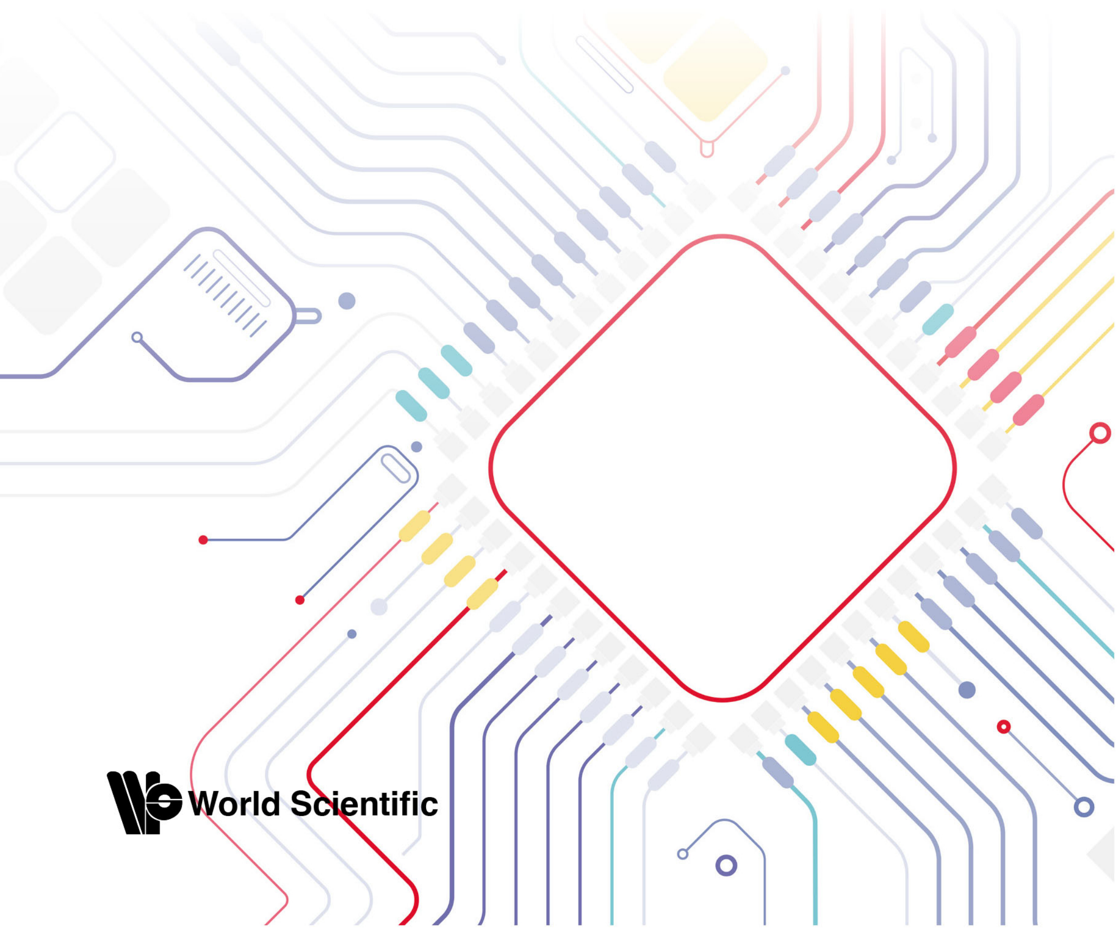
Logical Foundations of Computer Science

Vol 1: Propositional Logic

Peter A. Fejer & Dan A. Simovici



World Scientific

Logical Foundations of Computer Science

Vol 1: Propositional Logic

Logical Foundations of Computer Science

Vol 1: Propositional Logic

Peter A. Fejer & Dan A. Simovici

University of Massachusetts Boston, USA

World Scientific

NEW JERSEY • LONDON • SINGAPORE • BEIJING • SHANGHAI • HONG KONG • TAIPEI • CHENNAI • TOKYO

Published by

World Scientific Publishing Co. Pte. Ltd.
5 Toh Tuck Link, Singapore 596224
USA office: 27 Warren Street, Suite 401-402, Hackensack, NJ 07601
UK office: 57 Shelton Street, Covent Garden, London WC2H 9HE

Library of Congress Control Number: 2024016854

British Library Cataloguing-in-Publication Data
A catalogue record for this book is available from the British Library.

LOGICAL FOUNDATIONS OF COMPUTER SCIENCE
(In 2 volumes)
Volume 1: Propositional Logic
Volume 2: Predicate Logic

ISBN 978-981-12-8936-1 (set_hardcover)
ISBN 978-981-12-8934-7 (set_ebook for institutions)
ISBN 978-981-12-8935-4 (set_ebook for individuals)
ISBN 978-981-12-8765-7 (vol. 1_hardcover)
ISBN 978-981-12-8766-4 (vol. 2_hardcover)

For any available supplementary material, please visit
https://www.worldscientific.com/worldscibooks/10.1142/13714#t=suppl

Desk Editors: Balasubramanian Shanmugam/Steven Patt

Typeset by Stallion Press
Email: enquiries@stallionpress.com

Printed in Singapore

To our spouses
Elisabeth and Doina

Preface

In scientific reasoning, one starts with a collection of statements, the premises, in order to justify another statement, via a process of inference. Therefore, the study of logic is essential for students of computer science, mathematics, and all who use mathematical proofs.

Many of the fundamental computing concepts were created by logicians. The most famous such concept is the idea of a general-purpose computer, the Turing Machine. Computer programs are written in symbolic languages, e.g., Python, Java, and Lisp, that contain features of logical notations and symbolisms. Through such connections, the study of logic helps in the design of programs. Logic also has a role in the design of new programming languages, and it is essential for work in artificial intelligence.

An introductory chapter presents a set of theoretical and algebraic tools used throughout this book.

The syntactic and semantic concepts of propositional logic are discussed in the second chapter: formulas, truth assignments, truth tables, normal forms, clones of truth functions, and functional completeness. Some parts of logic are used by engineers in circuit design, a topic discussed extensively in the same chapter.

The object of the third chapter is to introduce a variety of propositional formal methods: Hilbert/Frege formal systems, tableaux, sequents, and natural deduction, and to examine transformation methods between these formalisms. Additionally, we present several variants of propositional resolution and the method of cutting planes.

The second part of the work deals with predicate logic also known as first-order logic. The development of this part parallels broadly the presentation of propositional logic. The first chapter of this part presents the syntax and semantics of predicate logic starting with the first-order formulas and structures. Various syntactic aspects specific to predicate logic are presented and then the focus shifts to semantics. We discuss normal forms for formulas and present certain special sets of formulas such as Hintikka sets and first-order theories. Also, the reduction of first-order logic to propositional logic is examined. This chapter concludes with a study of decidability in first-order logic.

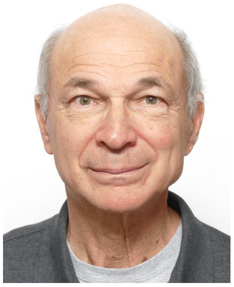
In the following chapter, several important corresponding formalisms for first-order logic are examined: Hilbert/Frege formal systems, tableaux, sequents, and natural deduction. Resolution which forms the basis for logic programming is discussed in various forms and special attention is paid to the method of paramodulation for languages with equality.

The last chapter includes the logical and mathematical analysis of programs, which allows proof of program correctness and analysis of the performance of programs. We discuss the use of logic for proving a variety of assertions concerning the correctness of programs and their performance.

The work contains more than 770 exercises and supplements that can be used to deepen the understanding of the material. We give detailed proofs and we do not shy away from technical difficulties. It is hoped that the readers would enjoy this introduction to logic and make good use of it in their own research.

Lexington and Brookline
Massachusetts
July 2023

About the Authors



Peter Fejer received his BA in Mathematics from Reed College and his SM and PhD degrees in Mathematics from the University of Chicago. He has held positions at Cornell University in the Mathematics Department and at the University of Massachusetts Boston in the Computer Science Department. He was also a Visiting Professor at Heidelberg University on several occasions. At UMass Boston, he served as Chair for 18 years and is now an Emeritus Professor. Professor Fejer's published research is in the area of Computability Theory.

Dan Simovici obtained his PhD in Mathematics from the University of Bucharest, Romania. His main research interests are in machine learning, data mining, and multi-valued logic. Dr. Simovici is a Computer Science Professor and Graduate Program Director at the University of Massachusetts in Boston, was a Visiting Professor in France and Japan, and serves as Editor-in-Chief of the *Journal of Multiple-Valued Logic and Soft Computing*. He is the author or co-author of more than 200 research publications and of several books. Dr. Simovici directed so far 13 PhD dissertations.

Contents

This page intentionally left blank

Volume 1
Propositional Logic

Chapter 1

Preliminaries

1.1 Introduction

This chapter contains a rather eclectic collection of results, intended to make the book self-contained. We cover set-theoretical concepts (properties of finite character and closure systems), sequences of symbols (including terms, substitutions, and unification), and trees regarded as functions on certain sets of sequences. This chapter concludes with a discussion of formal systems which are tools for deducing conclusions from hypotheses — a fundamental process in mathematical logic.

The reader may want to skim this rather voluminous introduction and come back to it later, as needed.

1.2 Sequences, Occurrences, and Substitutions

In logic, the fundamental syntactic objects are sequences of symbols. We remind the reader that a finite sequence is a function whose domain is $\{0, \ldots, n-1\}$ for some $n \in \mathbf{N}$. An infinite sequence is a function whose domain is $\mathbf{N}$. If q is a finite sequence with domain $\{0, \ldots, n-1\}$, then n is called the length of q and is denoted by $|q|$; the elements of the range of q are called the *entries* of q. If the range of q is contained in a set D, we refer to q as a finite sequence over D. Similarly, if r is an infinite sequence whose range is contained in D, then we refer to r as an infinite sequence over D. The set of sequences of length n over D is denoted by $\mathrm{Seq}_n(D)$ or D^n.

The set of sequences of length less than or equal to n over D is denoted by $\mathrm{Seq}_{\leq n}(D)$ or $D^{\leq n}$. For any set D, the set $\mathrm{Seq}_0(D)$ consists of exactly one sequence, the *null sequence*, denoted by λ. In this volume, we often identify an object d and the sequence of length 1 whose only entry is d. Under this convention, the sets D and D^1 are naturally identified. The set of all finite sequences over D is denoted by $\mathrm{Seq}(D)$ or, if D is finite, by D^*. The set of all infinite sequences over D is denoted by $\mathrm{ISeq}(D)$.

There are two commonly used notations for a finite sequence q of length n: $(q_0, \ldots, q_{n-1})$ and $q_0 \cdots q_{n-1}$, where $q_i = q(i)$ for $0 \leq i \leq n-1$. The second notation is often reserved for sequences over some fixed finite set of symbols, but we will use it more generally in this volume. Observe that when $n = 1$, the above notations become (q_0) and q_0, respectively. This is consistent with the identification of a sequence of length one with its lone entry.

For $n \in \mathbf{N}$ and $q = (q_0, \ldots, q_{n-1}), r = (r_0, \ldots, r_{n-1}) \in \mathrm{Seq}_n(D)$, define the set

$$\Delta(q, r) = \{i \in \{0, \ldots, n-1\} | q_i \neq r_i\}$$

and define $\delta : \mathrm{Seq}_n(D) \times \mathrm{Seq}_n(D) \longrightarrow \mathbf{N}$ by $\delta(q, r) = |\Delta(q, r)|$ (see Exercises 1 and 2 for properties of Δ and δ).

Definition 1.2.1. Let q', q'' be two finite sequences of lengths n and m, respectively. Then, the *concatenation* of q' and q'' is the sequence q of length $n + m$ given by

$$q(i) = \begin{cases} q'(i) & \text{if } 0 \leq i \leq n-1 \\ q''(i-n) & \text{if } n \leq i \leq n+m-1. \end{cases}$$

We will denote the concatenation of q' and q'' by $q'q''$. It is easy to show that concatenation is associative and that $q\lambda = q = \lambda q$ for every finite sequence q.

Let S, S' be two sets of finite sequences. The *concatenation of S and S'* is the set of sequences $SS' = \{qq' \mid q \in S \text{ and } q' \in S'\}$. We will write qS and Sq for $\{q\}S$ and $S\{q\}$, respectively.

If S is a set of sequences and r is a sequence, the *quotient of S by r* is the set of sequences q such that $rq \in S$. We denote this quotient by $S_{[r]}$. □

Example 1.2.2. Let $S = \{xx, xxyy, xyx\}$ and $S' = \{yyxy, xy\}$. The concatenation of S and S' is the set of sequences

$$SS' = \{xxyyxy, xxyyyyxy, xyxyyxy, xxxy, xyxxy\}.$$

Also, we have $S_{[xx]} = \{\lambda, yy\}$. □

Definition 1.2.3. Let q and r be two finite sequences; r is a *subsequence* of q if there are two finite sequences q_1 and q_2 such that $q = q_1 r q_2$. If $q = r q_2$, then we refer to r as a *prefix* of q; if $q = q_1 r$, r is a *suffix* of q.

If $q = q_0 \cdots q_{n-1}$ is a finite sequence and $i \in \mathbf{N}$, let $\text{pref}(q, i)$ be given by

$$\text{pref}(q, i) = \begin{cases} q_0 \cdots q_{i-1} & \text{if } i \leq n \\ q & \text{if } i > n. \end{cases}$$

Similarly, $\text{suff}(q, i)$ is given by

$$\text{suff}(q, i) = \begin{cases} q_{n-i} \cdots q_{n-1} & \text{if } i \leq n \\ q & \text{if } i > n. \end{cases}$$

The sets of all prefixes and all suffixes of a finite sequence q will be denoted by $\text{PREF}(q)$ and $\text{SUFF}(q)$, respectively.

A sequence r is a *proper prefix* (*proper suffix*) of q if r is a prefix (suffix) of q, $r \neq \lambda$ and $r \neq q$.

An *occurrence* of r in q is a pair (r, i) such that $0 \leq i \leq |q| - |r|$ and $r(\ell) = q(i + \ell)$ for every ℓ, $0 \leq \ell \leq |r| - 1$.

The set of all occurrences of r in q is denoted by $\texttt{OCC}_r(q)$. □

Clearly, there is an occurrence (r, i) of r in q if and only if r is a subsequence of q. If $|r| = 1$, then an occurrence of r in q is called an *occurrence of the symbol* $r(0)$ in q.

$|\texttt{OCC}_{(d)}(q)|$ will be referred to as the number of occurrences of a symbol d in a finite sequence q and be denoted by $|q|_d$.

Definition 1.2.4. Let q be a finite sequence and let (r, i) and (r', j) be occurrences of r and r' in q. The occurrence (r', j) is a *part of the occurrence* (r, i) if $0 \leq j - i \leq |r| - |r'|$. □

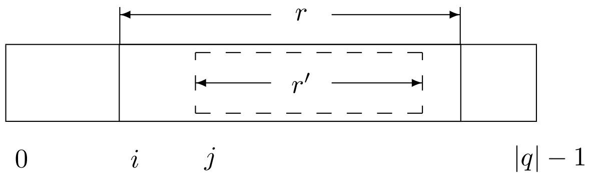


Fig. 1.1. Occurrence (r', j) is a part of occurrence (r, i).

The situation described in Definition 1.2.4 is illustrated in Figure 1.1.

Example 1.2.5. Consider the sequence $q = xxyxyxz$. The occurrences $(xy, 1), (yx, 2)$ and $(xy, 3)$ are parts of the occurrence $(xyxy, 1)$. □

Theorem 1.2.6. *If* $(r, j) \in \mathtt{OCC}_r(q)$ *and* $(s, i) \in \mathtt{OCC}_s(r)$*, then* $(s, i + j) \in \mathtt{OCC}_s(q)$*.*

Proof. The argument is left to the reader. □

Definition 1.2.7. Let q be a finite sequence and let $\zeta = (r, i)$ be an occurrence of r in q. If $q = q_0 r q_1$, where $|q_0| = i$, then the sequence which results from the *replacement* of the occurrence ζ in q by the finite sequence r' is the sequence $q_0 r' q_1$, denoted by $\mathtt{replace}\,(q, \zeta, r')$ or alternatively by $q[\zeta \to r']$. □

Example 1.2.8. For the occurrences $(xy, 1), (xy, 3)$ of the sequence xy in the sequence $q = xxyxyxz$, we have

$$\mathtt{replace}\,(q, (xy, 1), zxz) = xzxzxyxz,$$
$$\mathtt{replace}\,(q, (xy, 3), zxz) = xxyzxzxz.$$

□

Definition 1.2.9. A *substitution* is a function s whose range is a set of finite sequences. If $\mathrm{Dom}(s) = X$, we refer to s as a *substitution on* X.

The *carrier* of a substitution s is the set $\mathsf{carr}(s) = \{x \mid s(x) \neq (x)\}$. □

Definition 1.2.10. Let s be a substitution and let $q = q_0 \cdots q_{n-1}$ be a finite sequence. We define $\overline{s}(q)$ to be the finite sequence $q'_0 \cdots q'_{n-1}$, where

$$q'_i = \begin{cases} (q_i) & \text{if } q_i \notin \mathrm{Dom}(s), \\ s(q_i) & \text{otherwise.} \end{cases}$$

▯

Observe that if s is a substitution and $x \in \mathrm{Dom}(s)$, then $\overline{s}((x)) = s(x)$.

Theorem 1.2.11. *If s is a substitution and q, q' are finite sequences, then $\overline{s}(qq') = \overline{s}(q)\overline{s}(q')$.*

Proof. The argument is a direct consequence of the definition of $\overline{s}$, and it is left to the reader. □

Whenever there is no risk of confusion, we shall write $s(q)$ instead of $\overline{s}(q)$.

Example 1.2.12. Let q be the sequence $xyzxzyz$ and let s be the substitution $\{(x, xyz), (z, xxy)\}$. We have $s(q) = xyzyxxyxyzxx$ $yyxxy$. ▯

Theorem 1.2.13. *Let q, r_0, r_1 be three finite sequences such that there is an occurrence ζ of r_0 in q. If s is a substitution, then there is an occurrence ζ' of $s(r_0)$ in $s(q)$ such that*

$$s(\texttt{replace}\,(q, \zeta, r_1)) = \texttt{replace}\,(s(q), \zeta', s(r_1)).$$

Proof. We leave the argument to the reader since it is a simple application of the previous definitions. □

Definition 1.2.14. A substitution s such that $\mathsf{carr}(s)$ is a finite set is called a *finite substitution.* ▯

Suppose that we are working in a context where a certain set X is the domain of all substitutions under consideration. If $x_0, \ldots, x_{n-1}$ are distinct members of X and $r_0, \ldots, r_{n-1}$ are finite sequences, then we write $\mathsf{s}^{x_0 \cdots x_{n-1}}_{r_0 \cdots r_{n-1}}$ for the finite substitution s on X such that $s(x_i) = r_i$ for $0 \leq i \leq n-1$ and $s(x) = (x)$ for $x \in X - \{x_0, \ldots, x_{n-1}\}$. (For $n = 0$, $\mathsf{s}^{x_0 \cdots x_{n-1}}_{r_0 \cdots r_{n-1}}$ is the identity substitution on X, which maps $x \in X$

to (x) in accordance with our convention of identifying x with the sequence (x).)

Note that every finite substitution on X can be represented as $\mathsf{s}^{x_0 \cdots x_{n-1}}_{r_0 \cdots r_{n-1}}$ for appropriate $x_0, \ldots, x_{n-1}, r_0, \ldots, r_{n-1}$.

Our notation $\mathsf{s}^{x_0 \cdots x_{n-1}}_{r_0 \cdots r_{n-1}}$ is somewhat ambiguous in that it does not mention the domain X, but this ambiguity is harmless in the following sense. Let X, X' be two sets each containing $x_0, \ldots, x_{n-1}$ and let s be $\mathsf{s}^{x_0 \cdots x_{n-1}}_{r_0 \cdots r_{n-1}}$ considered as a substitution on X and s' be $\mathsf{s}^{x_0 \cdots x_{n-1}}_{r_0 \cdots r_{n-1}}$ considered as a substitution on X'. Then, for any sequence r, $s(r) = s'(r)$.

Example 1.2.15. We have $\mathsf{s}^{y}_{zz}(q) = xzzzxzzzz$ when q is the sequence $xyzxzyz$. ∎

Theorem 1.2.16. *For any finite sequence q and any x, y such that y does not occur in q, we have*

$$\mathsf{s}^{y}_{(x)}(\mathsf{s}^{x}_{(y)}(q)) = q.$$

Proof. The proof is a simple induction on $|q|$. □

Theorem 1.2.17. *Let $s = \mathsf{s}^{x_0 \cdots x_{n-1}}_{r_0 \cdots r_{n-1}}$ be a finite substitution such that $|r_i| > 0$ for $0 \leq i \leq n-1$. Then, for every finite sequence q, the set of finite sequences r such that $s(r) = q$ is finite.*

Proof. Suppose that $q = (q_0, \ldots, q_{m-1})$ is a sequence of length m and $Q = \{q_0, \ldots, q_{m-1}\}$ is the set of symbols that occur in q. If $s(r) = q$, then $|r| \leq m$ because none of the sequences r_i is null and $r \in \mathrm{Seq}(Q \cup \{x_0, \ldots, x_{n-1}\})$. Since $Q \cup \{x_0, \ldots, x_{n-1}\}$ is a finite set, it follows that there are only finitely many sequences r with this property. □

This theorem suggests the following definition:

Definition 1.2.18. A function $f : A \longrightarrow B$ is *finite-to-one* if the set $f^{-1}(b) = \{a \in A | f(a) = b\}$ is finite for every $b \in B$. ∎

Using the term introduced above, we can regard a substitution on X that satisfies the conditions of Theorem 1.2.17 as a finite-to-one function on $\mathrm{Seq}(X)$.

Theorem 1.2.19. *Let s_0, s_1 be two substitutions. Define a substitution s with domain* $\mathrm{Dom(s_0) \cup Dom(s_1)}$ *by*

$$s(x) = \begin{cases} \overline{s_0}(s_1(x)) & \text{if } \mathrm{x \in Dom(s_1)} \\ s_0(x) & \text{if } \mathrm{x \in Dom(s_0) - Dom(s_1)}. \end{cases}$$

Then, for every finite sequence q, we have $\overline{s_0}(\overline{s_1}(q)) = \overline{s}(q)$. Further, s is the unique substitution with domain $\mathrm{Dom(s_0) \cup Dom(s_1)}$ *that has the above property.*

Proof. We begin by showing that $\overline{s}((x)) = \overline{s_0}(\overline{s_1}((x)))$ for every symbol x. If $x \in \mathrm{Dom}(s_1)$, then $\overline{s_0}(\overline{s_1}((x))) = \overline{s_0}(s_1(x)) = s(x) = \overline{s}((x))$. If $x \in \mathrm{Dom}(s_0) - \mathrm{Dom}(s_1)$, then $\overline{s_0}(\overline{s_1}((x))) = \overline{s_0}((x)) = s_0(x) = s(x) = \overline{s}((x))$. In the remaining case, $x \notin \mathrm{Dom}(s_0) \cup \mathrm{Dom}(s_1) = \mathrm{Dom}(s)$, so $\overline{s_0}(\overline{s_1}((x))) = (x) = \overline{s}((x))$. Now the reader can verify immediately by induction on $|q|$ that $\overline{s_0}(\overline{s_1}(q)) = \overline{s}(q)$ for every finite sequence q.

Assume that s' is a substitution such that $\mathrm{Dom}(s') = \mathrm{Dom}(s)$ and $\overline{s_0}(\overline{s_1}(q)) = \overline{s'}(q)$ for every finite sequence q. Then, $s'(x) = \overline{s'}((x)) = \overline{s_0}(\overline{s_1}((x))) = \overline{s}((x)) = s(x)$ for every $x \in \mathrm{Dom}(s) = \mathrm{Dom}(s')$. Thus, $s = s'$. □

We will use the phrase "*composition of the substitutions s_0 and s_1*" to designate the substitution s whose existence is guaranteed by Theorem 1.2.19, even though s is not the functional composition of s_0 and s_1 as defined by set theory. We shall denote the composition (in this sense) of s_0 and s_1 by $s_0 * s_1$. Observe that the statement of Theorem 1.2.19 amounts to $\overline{s_0 * s_1} = \overline{s_0}\ \overline{s_1}$. Thus, for every $x \in \mathrm{Dom}(s_0) \cup \mathrm{Dom}(s_1)$, we have $s_0 * s_1(x) = \overline{s_0 * s_1}((x)) = \overline{s_0}(\overline{s_1}((x)))$.

Theorem 1.2.20. *Let ι be a substitution such that $\iota(x) = (x)$ for every $x \in$* $\mathrm{Dom}(\iota)$. *For every substitution s such that* $\mathrm{Dom}(\iota) \subseteq \mathrm{Dom(s)}$, *we have*

$$\iota * s = s = s * \iota.$$

If s_0, s_1, s_2 are any substitutions, then

$$s_2 * (s_1 * s_0) = (s_2 * s_1) * s_0.$$

In other words, composition of substitutions is associative.

Proof. We leave the easy first part to the reader.

To prove the second part, note that

$$\begin{aligned}
&\mathrm{Dom}(s_2 * (s_1 * s_0)) \\
&\quad = \mathrm{Dom}(s_2) \cup \mathrm{Dom}(s_1 * s_0) = \mathrm{Dom}(s_2) \cup \mathrm{Dom}(s_1) \cup \mathrm{Dom}(s_0) \\
&\quad = \mathrm{Dom}(s_2 * s_1) \cup \mathrm{Dom}(s_0) = \mathrm{Dom}((s_2 * s_1) * s_0).
\end{aligned}$$

Consider a symbol $x \in \mathrm{Dom}(s_2) \cup \mathrm{Dom}(s_1) \cup \mathrm{Dom}(s_0)$. We have

$$\begin{aligned}
&s_2 * (s_1 * s_0)(x) \\
&\quad = \overline{s_2 * (s_1 * s_0)}((x)) \\
&\quad = \overline{s_2}(\overline{s_1 * s_0}((x))) \\
&\quad = \overline{s_2}(\overline{s_1}(\overline{s_0}((x))))
\end{aligned}$$

because of Theorem 1.2.19 and of the definition of "$*$". Similarly, $(s_2 * s_1) * s_0(x) = \overline{s_2}(\overline{s_1}(\overline{s_0}((x))))$. □

Theorem 1.2.21. *The composition of two finite substitutions is a finite substitution. In fact, if* $s_0 = s^{y_0 \cdots y_{m-1}}_{r_0 \cdots r_{m-1}}$, $s_1 = s^{x_0 \cdots x_{n-1}}_{q_0 \cdots q_{n-1}}$, *and*

$$\{y_{j_0}, \ldots, y_{j_{k-1}}\} = \{y_0, \ldots, y_{m-1}\} - \{x_0, \ldots, x_{n-1}\},$$

then $s_0 * s_1 = s^{x_0 \cdots x_{n-1} y_{j_0} \cdots y_{j_{k-1}}}_{z_0 \cdots z_{n-1} r_{j_0} \cdots r_{j_{k-1}}}$, *where* $z_i = \overline{s_0}(q_i)$ *for* $0 \le i \le n-1$.

Proof. Let $s' = s^{x_0 \cdots x_{n-1} y_{j_0} \cdots y_{j_{k-1}}}_{z_0 \cdots z_{n-1} r_{j_0} \cdots r_{j_{k-1}}}$. We need to show that for every $x \in \mathrm{Dom}(s_0) \cup \mathrm{Dom}(s_1)$, $s'(x) = s_0 * s_1(x) = \overline{s_0}(\overline{s_1}((x)))$.

Suppose first that $x = x_i$. Then, $x \in \mathrm{Dom}(s_1)$ and $s_1(x) = q_i$, so $\overline{s_0}(\overline{s_1}((x))) = \overline{s_0}(q_i) = z_i = s'(x_i) = s'(x)$.

Now suppose that $x = y_{j_l}$. Regardless of whether $x \in \mathrm{Dom}(s_1)$ or not, we have $\overline{s_1}((x)) = (x)$. Therefore, $\overline{s_0}(\overline{s_1}((x))) = \overline{s_0}((x)) = r_{j_l} = s'(x)$.

Finally, if $x \notin \{x_0, \ldots, x_{n-1}, y_0, \ldots, y_{m-1}\}$, then

$$\overline{s_0}(\overline{s_1}((x))) = \overline{s_0}((x)) = (x) = s'(x).$$

□

Theorem 1.2.22. *Let $\{x_0, \ldots, x_{n-1}\}$ and $\{y_0, \ldots, y_{m-1}\}$ be two disjoint sets of symbols and let $q_0, \ldots, q_{n-1}, r_0, \ldots, r_{m-1}$ be sequences such that no symbol x_i occurs in any sequence r_j for $0 \leq i \leq n-1$ and $0 \leq j \leq m-1$. We have*

$$\mathsf{s}^{y_0 \cdots y_{m-1}}_{r_0 \cdots r_{m-1}} * \mathsf{s}^{x_0 \cdots x_{n-1}}_{q_0 \cdots q_{n-1}} = \mathsf{s}^{x_0 \cdots x_{n-1}}_{w_0 \cdots w_{n-1}} * \mathsf{s}^{y_0 \cdots y_{m-1}}_{r_0 \cdots r_{m-1}},$$

where $w_i = \mathsf{s}^{y_0 \cdots y_{m-1}}_{r_0 \cdots r_{m-1}}(q_i)$, for $0 \leq i \leq n-1$.

Proof. By Theorem 1.2.21, we can write

$$\mathsf{s}^{y_0 \cdots y_{m-1}}_{r_0 \cdots r_{m-1}} * \mathsf{s}^{x_0 \cdots x_{n-1}}_{q_0 \cdots q_{n-1}} = s^{x_0 \cdots x_{n-1} y_0 \cdots y_{m-1}}_{w_0 \cdots w_{n-1} q_0 \cdots q_{m-1}}.$$

By the same corollary, we have

$$\mathsf{s}^{x_0 \cdots x_{n-1}}_{w_0 \cdots w_{n-1}} * \mathsf{s}^{y_0 \cdots y_{m-1}}_{r_0 \cdots r_{m-1}} = s^{y_0 \cdots y_{m-1} x_0 \cdots x_{n-1}}_{z_0 \cdots z_{m-1} w_0 \cdots w_{n-1}},$$

where for each j, $z_j = \mathsf{s}^{x_0 \cdots x_{n-1}}_{w_0 \cdots w_{n-1}}(r_j) = r_j$ because no x_i occurs in a sequence r_j. This gives the desired equality. □

Composition of substitutions is not commutative in general. For example, let s_0, s_1 be the substitutions defined by $\text{Dom}(s_0) = \{x\}$ and $s_0(x) = (y)$, and $\text{Dom}(s_1) = \{y\}$ and $s_1(y) = (x)$, where $x \neq y$. We have $s_0 * s_1(x) = (y)$, while $s_1 * s_0(x) = (x)$. The following result gives sufficient conditions for two substitutions to commute.

Theorem 1.2.23. *Let s_0, s_1 be two substitutions whose domains are disjoint and no symbol that occurs in a sequence in $\text{Ran}(s_i)$ belongs to $\text{Dom}(\mathrm{s_j})$ for $i \neq j$, $i, j \in \{0, 1\}$. Then, we have $s_0 * s_1 = s_1 * s_0$.*

Proof. Since $\text{Dom}(s_0) \cap \text{Dom}(s_1) = \emptyset$, we can write

$$s_0 * s_1(x) = \begin{cases} \overline{s_0}(s_1(x)) & \text{if } x \in \text{Dom}(s_1) \\ s_0(x) & \text{if } x \in \text{Dom}(s_0), \end{cases}$$

$$s_1 * s_0(x) = \begin{cases} \overline{s_1}(s_0(x)) & \text{if } x \in \text{Dom}(s_0) \\ s_1(x) & \text{if } x \in \text{Dom}(s_1). \end{cases}$$

If $x \in \text{Dom}(s_1)$, the second condition guarantees that $\overline{s_0}(s_1(x)) = s_1(x)$. Similarly, if $x \in \text{Dom}(s_0)$, we have $\overline{s_1}(s_0(x)) = s_0(x)$. Thus, $s_0 * s_1(x) = s_1 * s_0(x)$ for every $x \in \text{Dom}(s_0) \cup \text{Dom}(s_1)$. □

1.3 Collections of Sets

In this section, we discuss two set-theoretical concepts — properties of finite character and closure systems — that will have later use in this volume.

As discussed in more detail in Section 1.2 of [13], we adopt the extensional point of view in defining properties. That is, we regard a property of the elements of a set S to be a subset of S. The set of all subsets of S is called the power set of S and is denoted by $\mathcal{P}(S)$. Thus, the set of all properties of the elements of S is $\mathcal{P}(S)$.

Definition 1.3.1. Let U be a set. A *property of finite character of the subsets of* U is a collection $\mathcal{C}$ of subsets of U such that A belongs to $\mathcal{C}$ if and only if every finite subset of A belongs to $\mathcal{C}$. (In other words, $\mathcal{C}(A)$ is true if and only if $\mathcal{C}(B)$ is true for every finite subset B of A.) ▯

It is easy to see that if C is a property of finite character of the subsets of U and $\mathcal{C}(A)$ is true, then $\mathcal{C}(B)$ is true for every subset B of A. We will refer to this by saying that $\mathcal{C}$ is *downward closed.*

Lemma 1.3.2. *Let U be a set and let $\mathcal{C}$ be a property of finite character of the subsets of U. If $(A_0, \ldots, A_n, \ldots)$ is a sequence of members of $\mathcal{C}$ such that $A_0 \subseteq \cdots \subseteq A_n \subseteq \cdots$, then $A = \bigcup\{A_i \mid i \geq 0\} \in \mathcal{C}$.*

Proof. Let $B = \{u_0, \ldots, u_{k-1}\}$ be a finite subset of A. For every $u_\ell \in B$, let q_ℓ be the least integer such that $u_\ell \in A_{q_\ell}$ for $0 \leq \ell \leq k-1$. If $q = \max\{q_0, \ldots, q_{k-1}\}$, then $B \subseteq A_q$, so $B \in \mathcal{C}$. Since every finite subset of A belongs to $\mathcal{C}$, we obtain $A \in \mathcal{C}$. □

If C is a property of the subsets of a set U, we call a set M a *maximal set in C* if $\mathcal{C}(M)$ is true and there is no set M' such that $M \subset M'$ and $\mathcal{C}(M')$ is true.

Theorem 1.3.3. *Let $U = \{u_0, \ldots, u_n, \ldots\}$ be a countable set and let C be a property of finite character of the subsets of U. Then, for every subset A of U such that $A \in \mathcal{C}$, there is a maximal set M in C that contains A.*

Proof. Consider the sequence $A_0, \ldots, A_n, \ldots$ of subsets of U defined recursively by $A_0 = A$ and

$$A_{n+1} = \begin{cases} A_n \cup \{u_n\} & \text{if } A_n \cup \{u_n\} \in \mathcal{C} \\ A_n & \text{otherwise.} \end{cases}$$

This sequence is nondecreasing and $A_n \in \mathcal{C}$ for every $n \in \mathbf{N}$. According to Lemma 1.3.2, we have $M = \bigcup\{A_i \mid i \in \mathbf{N}\} \in \mathcal{C}$ and, clearly $A \subseteq M$. We claim that M is a maximal set in C. Indeed, assume that $M \subset M'$, where $M' \in \mathcal{C}$. There exists $u_m \in U$ such that $u_m \in M' - M$. Since $u_m \notin M$, we have $u_m \notin A_{m+1}$. By the definition of the sequence $A_0, \ldots, A_n, \ldots$, this happens only if $A_m \cup \{u_m\} \notin \mathcal{C}$. Since $A_m \cup \{u_m\} \subseteq M' \in \mathcal{C}$, we obtain $A_m \cup \{u_m\} \in \mathcal{C}$ because $\mathcal{C}$ is of finite character. This contradiction implies that M is maximal. □

Definition 1.3.4. A *closure system* is a collection of sets $\mathcal{C}$ such that $\bigcup \mathcal{C} \in \mathcal{C}$ and for every nonempty collection $\mathcal{D} \subseteq \mathcal{C}$, we have $\bigcap \mathcal{D} \in \mathcal{C}$.

If $\bigcup \mathcal{C} = M$, we will refer to $\mathcal{C}$ as a *closure system on the set* M. ▯

Example 1.3.5. Let $\mathcal{C}$ be the collection of all intervals $[a, b] = \{x \in \mathbf{R} \mid a \leq x \leq b\}$ with $a, b \in \mathbf{R}$ and $a \leq b$, together with the empty set and the set $\mathbf{R}$. Note that $\bigcup \mathcal{C} = \mathbf{R} \in \mathcal{C}$, so the first condition of Definition 1.3.4 is satisfied.

Let $\mathcal{D}$ be a nonempty subcollection of $\mathcal{C}$. If $\emptyset \in \mathcal{D}$, then $\bigcap \mathcal{D} = \emptyset \in \mathcal{C}$. If $\mathcal{D} = \{\mathbf{R}\}$, then $\bigcap \mathcal{D} = \mathbf{R} \in \mathcal{C}$. Therefore, we need to consider only the case when $\mathcal{D} = \{[a_i, b_i] \mid i \in I\}$. Then, $\bigcap \mathcal{D} = \emptyset$, unless $a = \sup\{a_i \mid i \in I\}$ and $b = \inf\{b_i \mid i \in I\}$ both exist and $a \leq b$, in which case, $\bigcap \mathcal{D} = [a, b]$. Thus, $\mathcal{C}$ is a closure system. ▯

Example 1.3.6. Let M be a set and let F be a set of operations on M. A subset P of M is called *closed under* F or F*-closed* if P is closed under f for every $f \in F$, that is, for every operation $f \in F$, if f is n-ary and $p_0, \ldots, p_{n-1} \in P$, then $f(p_0, \ldots, p_{n-1}) \in P$. Note that M itself is closed under F. Further, if $\mathcal{C}$ is a nonempty collection of F-closed subsets of M, then $\bigcap \mathcal{C}$ is also F-closed. Therefore, the collection of all F-closed subsets of M is a closure system. ▯

Definition 1.3.7. A mapping $\kappa : \mathcal{P}(M) \longrightarrow \mathcal{P}(M)$ is a *closure operator* on a set M if it satisfies the following conditions:

- $H \subseteq \kappa(H)$ (expansiveness)
- $H \subseteq L$ implies $\kappa(H) \subseteq \kappa(L)$ (monotonicity)
- $\kappa(\kappa(H)) = \kappa(H)$ (idempotency)

for $H, L \in \mathcal{P}(M)$. ▯

Example 1.3.8. Let $\kappa : \mathcal{P}(\mathbf{R}) \longrightarrow \mathcal{P}(\mathbf{R})$ be defined by

$$\kappa(L) = \begin{cases} \emptyset & \text{if } L = \emptyset \\ [a, b] & \text{if both } a = \inf L \text{ and } b = \sup L \text{ exist} \\ \mathbf{R} & \text{otherwise.} \end{cases}$$

We leave to the reader the verification that κ is a closure operator. ▯

Closure operators generate closure systems, as shown by the following lemma.

Lemma 1.3.9. *Let $\kappa : \mathcal{P}(M) \longrightarrow \mathcal{P}(M)$ be a closure operator. Define the family of sets $\mathcal{C}_\kappa = \{H \in \mathcal{P}(M) \mid H = \kappa(H)\}$. Then, $\mathcal{C}_\kappa$ is a closure system on M.*

Proof. Since $M \subseteq \kappa(M) \subseteq M$, we have $M \in \mathcal{C}_\kappa$, so $\bigcup \mathcal{C}_\kappa = M \in \mathcal{C}_\kappa$.

Let $\mathcal{D} = \{D_i \mid i \in I\}$ be a nonempty collection of subsets of M such that $D_i = \kappa(D_i)$ for $i \in I$. Since $\bigcap \mathcal{D} \subseteq D_i$, we have $\kappa(\bigcap \mathcal{D}) \subseteq \kappa(D_i) = D_i$ for every $i \in I$. Therefore, $\kappa(\bigcap \mathcal{D}) \subseteq \bigcap \mathcal{D}$, which implies $\kappa(\bigcap \mathcal{D}) = \bigcap \mathcal{D}$. This proves our claim. □

Note that $\mathcal{C}_\kappa$, as defined in Lemma 1.3.9, equals the range of κ. Indeed, if $L \in \text{Ran}(\kappa)$, then $L = \kappa(H)$ for some $H \in \mathcal{P}(M)$, so $\kappa(L) = \kappa(\kappa(H)) = \kappa(H) = L$, which shows that $L \in \mathcal{C}_\kappa$. The reverse inclusion is obvious.

We shall refer to the sets in $\mathcal{C}_\kappa$ as the *κ-closed subsets of M*.

In the reverse direction to Lemma 1.3.9, we show that every closure system generates a closure operator.

Lemma 1.3.10. *Let $\mathcal{C}$ be a closure system on the set M. Define the mapping $\kappa_\mathcal{C} : \mathcal{P}(M) \longrightarrow \mathcal{P}(M)$ by $\kappa_\mathcal{C}(H) = \bigcap\{L \in \mathcal{C} \mid H \subseteq L\}$. Then, $\kappa_\mathcal{C}$ is a closure operator on the set M.*

Proof. Note that the collection $\{L \in \mathcal{C} \mid H \subseteq L\}$ is not empty since it contains at least M, so $\kappa_\mathcal{C}(H)$ is defined and it is clearly the smallest element of $\mathcal{C}$ that contains H. Also, by the definition of $\kappa_{\mathcal{C}}(H)$, it follows immediately that $H \subseteq \kappa_{\mathcal{C}}(H)$ for every $H \in \mathcal{P}(M)$.

Suppose that $H_1, H_2 \in \mathcal{P}(M)$ are such that $H_1 \subseteq H_2$. Since

$$\{L \in \mathcal{C} \mid H_2 \subseteq L\} \subseteq \{L \in \mathcal{C} \mid H_1 \subseteq L\},$$

we have

$$\bigcap\{L \in \mathcal{C} \mid H_1 \subseteq L\} \subseteq \bigcap\{L \in \mathcal{C} \mid H_2 \subseteq L\},$$

so $\kappa_{\mathcal{C}}(H_1) \subseteq \kappa_{\mathcal{C}}(H_2)$.

We have $\kappa_{\mathcal{C}}(H) \in \mathcal{C}$ for every $H \in \mathcal{P}(M)$ because $\mathcal{C}$ is a closure system. Therefore, $\kappa_{\mathcal{C}}(H) \in \{L \in \mathcal{C} \mid \kappa_{\mathcal{C}}(H) \subseteq L\}$, so $\kappa_{\mathcal{C}}(\kappa_{\mathcal{C}}(H)) \subseteq \kappa_{\mathcal{C}}(H)$. Since the reverse inclusion clearly holds, we obtain $\kappa_{\mathcal{C}}(\kappa_{\mathcal{C}}(H)) = \kappa_{\mathcal{C}}(H)$. □

Theorem 1.3.11. *Let M be a set. For every closure system $\mathcal{C}$ on M, we have $\mathcal{C} = \mathcal{C}_{\kappa_{\mathcal{C}}}$. For every closure operator κ on M, we have $\kappa = \kappa_{\mathcal{C}_\kappa}$.*

Proof. Let $\mathcal{C}$ be a closure system on M and let $H \subseteq M$. Then, we have the following equivalent statements:

(1) $H \in \mathcal{C}_{\kappa_{\mathcal{C}}}$,
(2) $\kappa_{\mathcal{C}}(H) = H$,
(3) $H \in \mathcal{C}$.

The equivalence between (2) and (3) follows from the fact that $\kappa_{\mathcal{C}}(H)$ is the smallest element of C that contains H.

Conversely, let κ be a closure operator on M. To prove the equality of κ and $\kappa_{\mathcal{C}_\kappa}$, consider the following list of equal sets, where $H \subseteq M$:

(1) $\kappa_{\mathcal{C}_\kappa}(H)$,
(2) $\bigcap\{L \in \mathcal{C}_\kappa \mid H \subseteq L\}$,
(3) $\bigcap\{L \in \mathcal{P}(M) \mid H \subseteq L = \kappa(L)\}$,
(4) $\kappa(H)$.

We need to justify only the equality of the last two members of the list. Since $H \subseteq \kappa(H) = \kappa(\kappa(H))$, we have $\kappa(H) \in \{L \in \mathcal{P}(M) \mid H \subseteq L = \kappa(L)\}$. Thus, $\bigcap\{L \in \mathcal{P}(M) \mid H \subseteq L = \kappa(L)\} \subseteq \kappa(H)$. To prove the reverse inclusion, note that for every $L \in \{L \in \mathcal{P}(M) \mid H \subseteq L = \kappa(L)\}$, we have $H \subseteq L$, so $\kappa(H) \subseteq \kappa(L) = L$. Therefore, $\kappa(H) \subseteq \bigcap\{L \in \mathcal{P}(M) \mid H \subseteq L = \kappa(L)\}$. □

Theorem 1.3.11 shows the existence of a natural bijection between the set of closure operators on a set M and the set of closure systems on M.

Example 1.3.12. Let κ be the closure operator given in Example 1.3.8. Since the closure system $\mathcal{C}_\kappa$ equals the range of κ, it follows that the members of $\mathcal{C}_\kappa$, the κ-closed sets, are $\emptyset$, $\mathbf{R}$, and all closed intervals $[a, b]$ with $a \leq b$. Thus, $\mathcal{C}_\kappa$ is the closure system $\mathcal{C}$ introduced in Example 1.3.5. Therefore, κ and $\mathcal{C}$ correspond to each other under the bijection of Theorem 1.3.11. $\square$

1.4 Decidable and Semidecidable Sets

In this section, we present informally the notions of decidable and semidecidable sets. Usually, these topics are treated in the context of computability theory, where they receive a formal treatment, based on the notion of partial recursive function. We undertake this informal treatment because we want to make this book self-contained. The informality is due to the fact that we base our presentation on the notion of algorithm, without defining this term precisely. When introducing an algorithm, we make implicitly two effectiveness assumptions: the inputs are given effectively and the algorithm provides an effective transformation of the inputs into the output. The notion of construction is somewhat weaker: if inputs were given effectively, we could use the construction to effectively produce the output from the input. However, we do not assume effectively given inputs.

A set M is *enumerable* if $M = \emptyset$ or M can be written as $\{m_0, m_1, \ldots\}$. An effective version of this notion is introduced next.

Definition 1.4.1. A set M is *effectively enumerable* if $M = \emptyset$ or there is an algorithm such that

(1) for every $n \in \mathbf{N}$, the algorithm produces an element m_n of M,
(2) $M = \{m_0, m_1, \ldots\}$.

We refer to the infinite sequence $(m_0, m_1, \ldots)$ as an *effective enumeration* of M. $\square$

Note that every finite set is effectively enumerable because the enumerating algorithm may repeat the elements it generates.

Theorem 1.4.2. *Let U be an effectively enumerable set. The set Seq(U) of finite sequences of elements of U is effectively enumerable.*

Proof. Let $(u_0, u_1, \ldots)$ be an effective enumeration of the set U. Consider the following algorithm for enumerating the set $\text{Seq}(U)$:

Input: A number $n \in \mathbf{N}$.
Output: A sequence in $\text{Seq}(U)$.
Method: If $n \in \{0, 1\}$, the output is λ. Otherwise, factor n as a product of primes, $n = p_0^{i_0} \cdots p_{k-1}^{i_{k-1}}$, where $i_{k-1} > 0$ and $(p_0, p_1, \ldots)$ is the sequence of prime numbers listed in increasing order. Let $(i'_0, \ldots, i'_{l-1})$ be the subsequence of the sequence $(i_0, \ldots, i_{k-1})$ that consists of its nonzero entries. Output the sequence $(u_{i'_0 - 1}, \ldots, u_{i'_{l-1} - 1})$.

We claim that the algorithm enumerates the set $\text{Seq}(U)$. Indeed, it is clear that the empty sequence will be generated by the algorithm (for $n = 0$ and $n = 1$). A sequence $(u_{i_0}, \ldots, u_{i_{k-1}})$, with $k > 0$, will be produced by the algorithm when the input is $n = p_0^{i_0+1} \cdots p_{k-1}^{i_{k-1}+1}$.

□

Definition 1.4.3. Let $(u_0, u_1, \ldots)$ be an effective enumeration of a set U, referred to in this context as a *universal set*. A subset A of U is *decidable* if there is an algorithm which, given an element u_i of U, returns 1 if $u_i \in A$ and 0 otherwise.

A subset A of U is *semidecidable* if there is an algorithm which, given an element u_i of U, returns 1 if $u_i \in A$ and does not return a value if $u_i \notin A$. ▯

If a subset A of U is decidable or semidecidable relative to an effective enumeration $(u_0, u_1, \ldots)$, then it is decidable or semidecidable, respectively, with respect to any other effective enumeration $(u'_0, u'_1, \ldots)$. Indeed, given an element u'_i, we run the enumerating algorithm for the effective enumeration $(u_0, u_1, \ldots)$ until we find the first j such $u_j = u'_i$. Then, using the decision algorithm for A determine whether $u_j \in A$. This clearly allows us to decide whether $u'_i \in A$. A similar argument works for semidecidability.

In Exercise 42, we ask the reader to show that if set A is the subset of two effectively enumerable sets U and U', then A is semidecidable a subset of U if and only if A is semidecidable as a subset of U'. However, the decidability of A as a subset of a universal set U

depends on this universal set. Thus, when dealing with decidability, it is important to specify the universal set. If the set U is clear from the context, then instead of saying that a subset A of U is a decidable or semidecidable subset of U, we will simply say that A is a decidable or semidecidable set.

Theorem 1.4.4. *Every decidable subset A of an effectively enumerable set U is semidecidable.*

Proof. We can produce a semidecidability algorithm for A starting from the decidability algorithm for A and entering an infinite loop when the decidability algorithm yields a 0. □

Theorem 1.4.5. *If A is a decidable subset of an effectively enumerable set U, then $U - A$ is also decidable. Further, if B is another decidable subset of U, then $A \cup B$ is decidable.*

Proof. To obtain a decidability algorithm for $U - A$, we can apply the decidability algorithm for A and replace 1 with 0 and 0 with 1 in the output.

We leave the second part to the reader. □

Theorem 1.4.6. *Let U be an effectively enumerable set. A subset A of U is decidable if and only if both A and $U - A$ are semidecidable.*

Proof. Let A be decidable. By Theorems 1.4.4 and 1.4.5, both A and $U - A$ are semidecidable.

Conversely, let A and $U - A$ be semidecidable and let $u_i \in U$. To decide whether $u_i \in A$, run the semidecidability algorithms for A and $U - A$ in parallel on u_i until one of them gives an output of 1. Then, decide that $u_i \in A$ if the one to return 1 was the semidecidability algorithm for A and decide that $u_i \notin A$ if 1 was returned by the semidecidability algorithm for $U - A$. □

Theorem 1.4.7. *A subset A of an effectively enumerable set U is semidecidable if and only if it is itself effectively enumerable.*

Proof. Suppose that A is an effectively enumerable subset of the effectively enumerable set U. If $A = \emptyset$, then A is clearly semidecidable. Otherwise, given an element u of U, apply the enumeration algorithm for A. Return 1 if a_i the ith output of the enumeration algorithm of A equals u. This is clearly a semidecidability algorithm for A.

Conversely, suppose that A is a semidecidable subset of the effectively enumerable set U. If $A = \emptyset$, then A is effectively enumerable. Otherwise, let $\mathcal{E}$ be an enumerating algorithm for U. To enumerate effectively the set A, consider the following algorithm. For $i = 0, 1, 2, \ldots$, run $\mathcal{E}$ to produce $u_0, \ldots, u_i$. Run the semideciding algorithm for A on each of $u_0, \ldots, u_i$ for i steps or until it halts, whichever comes first, and add to the enumeration of A all those u_js for which this algorithm returns 1. Note that every element of the set A will appear infinitely often in this enumeration, so A is effectively enumerable. □

The technique used in this informal proof is known in the computability literature as "dovetailing."

Definition 1.4.8. Let U, V be two effectively enumerable sets and let A, B be subsets of U, V, respectively. A is *many-one reducible* to B if there is an algorithm that computes a function $f : U \longrightarrow V$ such that $u \in A$ if and only if $f(u) \in B$ for all $u \in U$. This will be denoted by $A \leq_m B$. We will refer to the function f as a *reduction function.*

If we have both $A \leq_m B$ and $B \leq_m A$, then we say that the sets A and B are *m-equivalent* and write $A \equiv_m B$. ▯

If two sets are m-equivalent, membership in these sets is equally difficult to verify.

Because we do not consider other types of reducibility, we will often say that A is *reducible* to B when we mean that A is many-one reducible to B.

Theorem 1.4.9. *Let U, V be two effectively enumerable sets and let A, B be subsets of U, V, respectively, such that $A \leq_m B$:*

(1) *If B is decidable, then A is decidable.*
(2) *If B is semidecidable, then A is semidecidable.*

Proof. Let $f : U \longrightarrow V$ be a reduction function of A to B. Suppose that B is decidable. A decision algorithm for A works as follows: given an element u_i of U, apply the decision algorithm for B to $f(u_i)$ and return the same answer.

A similar argument works if B is semidecidable. □

1.5 Signatures and Terms

Terms as syntactic objects are built using two types of symbols: variables and function symbols. Terms are notations which can be used for describing computations by assigning values to variables and functions to function symbols. Any such assignment results in a value which is the result of the computation described by the term. This idea is fundamental for first-order logic and has applications in propositional logic as well.

Beginning with this section, we fix three objects denoted by

$$(\;) \; .$$

The set P consisting of these symbols will be called the set of *punctuation symbols*.

Definition 1.5.1. A *signature* is a pair $S = (F, \nu)$, where F is an arbitrary set, disjoint from P, whose elements are called the *function symbols* of $\mathtt{S}$ and $\nu : F \longrightarrow \mathbf{N}$ is called the *arity function* of $\mathtt{S}$.

If $f \in F$ and $\nu(f) = n$, we refer to f as an *n-ary function symbol* of $\mathtt{S}$. The set of all n-ary function symbols of S is denoted by F_n^S.

Elements of F_0^S are called *constant symbols* of S. ∎

Whenever there is no risk of confusion, we will write F_n rather than F_n^S.

Definition 1.5.2. Let $S = (F, \nu)$ be a signature and let V be a set disjoint from F and P. The elements of V are called *variables*.

We define the set $\mathrm{TERM}_S(V)$ of *terms* of S over the set of variables V to be the set of finite sequences of function symbols of S, variables, and punctuation symbols constructed inductively as follows:

(1) Every variable in V is in $\mathrm{TERM}_S(V)$.
(2) Every constant symbol of S is in $\mathrm{TERM}_S(V)$.
(3) For each function symbol f of S of positive arity (say of arity n) and each n-tuple $(t_0, \ldots, t_{n-1})$ of terms of $\mathrm{TERM}_S(V)$, $f(t_0, \ldots, t_{n-1})$ is in $\mathrm{TERM}_S(V)$.

∎

In the above definition, we identify an element of a set with the sequence of length 1 whose only entry is that element.

Note that Rule 3 of the definition actually consists of one rule for each function symbol f of S of positive arity n, which we denote by 3_f:

3_f For each n-tuple $(t_0, \ldots, t_{n-1})$ of terms of $\text{TERM}_S(V)$, the sequence $f(t_0, \ldots, t_{n-1})$ is in $\text{TERM}_S(V)$.

Since constant symbols are function symbols with arity 0, we could have omitted the second clause of the above definition if we allowed any arity in the third clause, but then we would have gotten terms of the form $a()$ instead of the more usual a.

If the signature S and the set of variables V are clear from the context, then we will refer to the terms of S over V as (S, V)-*terms* or simply as *terms*. In general, we will use the letters t, u, v, w to denote terms.

Example 1.5.3. Let $S = (\{a, f, g, h\}, \nu)$ be a signature with

$$\nu(a) = 0, \nu(f) = 1, \nu(g) = \nu(h) = 2$$

and let $V = \{x_0, x_1\}$. Since variables and constants' symbols of S are terms of S over V, we have $x_0, x_1, a \in \text{TERM}_S(V)$. Therefore, by Definition 1.5.2, we obtain the terms $t_1 = f(a)$, $t_2 = f(f(a))$, and $t_3 = f(x_1)$. The following are further terms:

$$\begin{aligned} g(t_1, t_2) &= g(f(a), f(f(a))), \\ h(t_1, x_0) &= h(f(a), x_0), \\ h(t_2, t_3) &= h(f(f(a)), f(x_1)). \end{aligned}$$

$\square$

Let $S = (F, \nu)$ be a signature and let V be a set of variables. If $t \in \text{TERM}_S(V)$, we write $\mathtt{V}_{S,V}(t)$ to denote the set of variables of V that occur in t. If S and V are understood from the context, we write $\mathtt{V}(t)$ rather than $\mathtt{V}_{S,V}(t)$. Similarly, we denote by $\mathtt{C}_{S,V}(t)$ or more simply, $\mathtt{C}(t)$, the set of constant symbols that occur in the term t.

Definition 1.5.4. Let $S = (F, \nu)$ be a signature. The set of *ground terms* of S, GTERM_S, is the set $\text{TERM}_S(\emptyset)$. $\square$

Note that the terms t_1, t_2, and $g(t_1, t_2)$ from Example 1.5.3 are ground terms of S, while $g(t_1, x_0)$ is not.

Example 1.5.5. If $S = (\emptyset, \nu)$, then ν is the empty function; in this case, we have $\mathrm{TERM}_S(V) = V$. ▯

Definition 1.5.6. Let $S = (F, \nu)$ and $S' = (F', \nu')$ be two signatures. Then, S is an *extension* of S', written $S' \preceq S$, if $F' \subseteq F$ and $\nu' = \nu \restriction F'$. If $S' \preceq S$, we say that S' is a *reduct* of S. ▯

Theorem 1.5.7. *Let S and S' be two signatures such that $S' \preceq S$. If $V' \subseteq V$, then $\mathrm{TERM}_{S'}(V') \subseteq \mathrm{TERM}_S(V)$.*

Proof. We leave this easy argument to the reader. □

Corollary 1.5.8. *Let S be a signature and V a set of variables. Then, $\mathrm{GTERM}_S \subseteq \mathrm{TERM}_S(V)$.*

Proof. Since $\mathrm{GTERM}_S = \mathrm{TERM}_S(\emptyset)$, the corollary follows immediately from Theorem 1.5.7. □

Theorem 1.5.9. *Let S be a signature and let V be a set of variables. Then GTERM_S consists of those terms of $\mathrm{TERM}_S(V)$ that do not contain any variables.*

Proof. The proof is left to the reader in Exercise 51. □

Because we will be introducing functions by recursive definitions, it is important to show that the definition of the set $\mathrm{TERM}_S(V)$ satisfies the unique readability condition. The following definition and lemma will be used for this purpose.

Definition 1.5.10. Let $S = (F, \nu)$ be a signature and let V be a set of variables. The function $K_{S,V} : F \cup V \cup P \longrightarrow \mathbf{Z}$ is given by the following table:

symbol s	variable	,	(	)	n-ary function symbol
$K_{S,V}(s)$	1	0	-1	1	$1-n$

▯

If the signature S and the set of variables V are clear from the context, we will write K for $K_{S,V}$.

Lemma 1.5.11. *Let $S = (F, \nu)$ be a signature and let V be a set of variables. If $t \in \mathrm{TERM}_S(V)$, then no proper prefix of t is in $\mathrm{TERM}_S(V)$.*

Proof. We extend K to have domain consisting of all sequences of variables, punctuation symbols, and function symbols of S by defining

$$K((s_0, \ldots, s_{n-1})) = \Sigma_{i=0}^{n-1} K(s_i).$$

Now we can show by induction on terms in $\mathrm{TERM}_S(V)$ that $K(t) = 1$ for each term. If t is a variable or constant symbol, then $K(t) = 1$ because a constant symbol is a 0-ary function symbol. If f is an n-ary function symbol with $n \geq 1$ and $t_0, \ldots, t_{n-1}$ are terms with $K(t_i) = 1$ for each i, $0 \leq i \leq n-1$, then

$$\begin{aligned} K(f(t_0, \ldots, t_{n-1})) &= K(f) + K(() + K(t_0) + K(,) + \cdots \\ &\quad + K(,) + K(t_{n-1}) + K()) \\ &= (1-n) - 1 + n \cdot 1 + 1 \\ &= 1. \end{aligned}$$

Next, we can show that if t is a term in $\mathrm{TERM}_S(V)$, then for every proper prefix u of t, $K(u) < 1$. This implies that a proper prefix of term cannot be a term (because for a term t, $K(t) = 1$).

The basis is immediate since variables and constant symbols have no proper prefixes. Now let $t = f(t_0, \ldots, t_{n-1})$, where f is an n-ary function symbol with $n > 0$. Suppose that $K(u) < 1$ for every proper prefix u of each term t_i for $0 \leq i \leq n-1$. A proper prefix u of t has one of the following forms:

- f,
- $f($,
- $f(t_0, \ldots, t_{i-1}, v$,
- $f(t_0, \ldots, t_i$,
- $f(t_0, \ldots, t_i$.

We assume here that $0 \leq i \leq n-1$ and that v is a proper prefix of t_i. In the first case, $K(u) = 1 - n < 1$, while in the second case, $K(u) = -n < 1$. In the third case, we have $K(u) = (i-n) + K(v) \leq -1 + K(v)$. By the inductive hypothesis, $K(v) < 1$, so $K(u) < 0$. Finally, in the last two cases, $K(u) = (i-n) + 1 \leq 0$. □

Theorem 1.5.12. *For every signature S and set of variables V, the definition of* $\text{TERM}_S(V)$ *(Definition 1.5.2) meets the unique readability condition.*

Proof. We need to show that every term t is put into $\text{TERM}_S(V)$ by exactly one rule of Definition 1.5.2 and that if it is put in by one of the rules 3_f, that is, if $t = f(t_0, \ldots, t_{n-1})$, then f and the terms $t_0, \ldots, t_{n-1}$ are uniquely determined.

If the first symbol of a term t is a variable, then t is put in $\text{TERM}_S(V)$ by the first rule; if the first symbol of t is a constant symbol, then t is put in by the second rule; finally, if the first symbol of t is a function symbol f of positive arity n, then t is put in by Rule 3_f. In other words, the rule used to put in t is determined by the first symbol of t.

Assume now that $t = f(t_0, \ldots, t_{n-1}) = f(t'_0, \ldots, t'_{n-1})$ for f a function symbol of S and $t_0, \ldots, t_{n-1}, t'_0, \ldots, t'_{n-1}$ terms of $\text{TERM}_S(V)$. Suppose that for some i, $0 \leq i \leq n-1$, we have $t_i \neq t'_i$. Let i_0 be the least i with this property. Of course, for $j < i$, we have $t_j = t'_j$ and this gives

$$t_{i_0}, \ldots, t_{n-1}) = t'_{i_0}, \ldots, t'_{n-1}).$$

Since t_{i_0} cannot be a proper prefix of t'_{i_0} nor vice versa, $t_{i_0} = t'_{i_0}$, which contradicts the definition of i_0. Therefore, $t_i = t'_i$ for each i, $0 \leq i \leq n-1$. □

There is an alternative notation for terms called Polish notation which dispenses with the use of punctuation symbols.

Definition 1.5.13. Let $S = (F, \nu)$ be a signature and let V be a set of variables.

We define the set $\text{POLTERM}_S(V)$ of *terms in Polish notation* of S over the set of variables V to be the set of finite sequences of function symbols of S and variables constructed inductively as follows:

(1) Every variable in V is in $\text{POLTERM}_S(V)$.
(2) $ft_0 \ldots t_{n-1}$ is in $\text{POLTERM}_S(V)$ for each n-ary function symbol f of S and each n-tuple $(t_0, \ldots, t_{n-1})$ of terms of $\text{POLTERM}_S(V)$.

□

Polish notation is more concise but also more difficult to read than the notation for terms previously introduced.

Example 1.5.14. Let $S = (\{a, f, g, h\}, \nu)$ be the signature introduced in Example 1.5.3. Since variables and constants symbols of S are terms in Polish notation of S over V, we have $x_0, x_1, a \in \mathrm{POLTERM}_S(V)$. Therefore, by Definition 1.5.13, we obtain the terms in Polish notation $t'_1 = fa$, $t'_2 = ffa$ and $t'_3 = fx_1$. The following are further terms in Polish notation:

$$\begin{aligned} gt'_1t'_2 &= gfaffa, \\ ht'_1x_0 &= hfax_0, \\ ht'_2t'_3 &= hffafx_1. \end{aligned}$$

∎

Theorem 1.5.15. *For every signature S and set of variables V, the definition of* $\mathrm{POLTERM}_S(V)$ *(Definition 1.5.13) meets the unique readability condition.*

Proof. Let K be the function introduced in Definition 1.5.10. An argument similar to one used in Lemma 1.5.11 shows that for every term t in Polish notation, we have $K(t) = 1$ and for every proper prefix u of a term in $\mathrm{POLTERM}_S(V)$, we have $K(u) < 1$. Consequently, no proper prefix of a term in $\mathrm{POLTERM}_S(V)$ can belong to $\mathrm{POLTERM}_S(V)$. The proof of unique readability now proceeds as in Theorem 1.5.12. □

Terms, in either our initial notation or in Polish notation, are descriptions of potential computations. The differences between the two notations are not essential, and in fact there is a "natural" bijection between $\mathrm{TERM}_S(V)$ and $\mathrm{POLTERM}_S(V)$. This remark is made precise by the following theorem.

Theorem 1.5.16. *Let $S = (F, \nu)$ be a signature and let V be a set of variables. Consider the mappings* $\Psi : \mathrm{POLTERM}_S(V) \longrightarrow \mathrm{TERM}_S(V)$ *and* $\Phi : \mathrm{TERM}_S(V) \longrightarrow \mathrm{POLTERM}_S(V)$ *defined recursively by*

$$\begin{aligned} \Psi(x) &= x, \\ \Psi(a) &= a, \\ \Psi(fu_0 \dots u_{n-1}) &= f(\Psi(u_0), \dots, \Psi(u_{n-1})) \end{aligned}$$

and

$$\begin{aligned} \Phi(x) &= x, \\ \Phi(a) &= a, \\ \Phi(f(t_0,\ldots,t_{n-1})) &= f\Phi(t_0)\ldots\Phi(t_{n-1}), \end{aligned}$$

for every variable x, constant symbol $a \in F$, $f \in F$ (with $n = \nu(f) > 0$), $u_0,\ldots,u_{n-1} \in \mathrm{POLTERM}_S(V)$ and $t_0,\ldots,t_{n-1} \in \mathrm{TERM}_S(V)$. Then, Φ and Ψ are bijections which are inverse of each other.

Proof. It suffices to show, for each $u \in \mathrm{POLTERM}_S(V)$, that $\Phi(\Psi(u)) = u$ and, for each $t \in \mathrm{TERM}_S(V)$, that $\Psi(\Phi(t)) = t$. We show the first claim by induction on u and leave the second for the reader. If u is a variable or constant symbol, then $\Phi(\Psi(u)) = \Phi(u) = u$, by the definitions of Ψ and Φ. This shows the basis steps of the induction. Now suppose that $u = fu_0\ldots u_{n-1}$ and that $\Phi(\Psi(u_i)) = u_i$ for $0 \le i \le n-1$, where $n > 0$. Then,

$$\begin{aligned} \Phi(\Psi(u)) &= \Phi(f(\Psi(u_0),\ldots,\Psi(u_{n-1}))) \\ &= f\Phi(\Psi(u_0))\ldots\Phi(\Psi(u_{n-1})) \\ &= fu_0\ldots u_{n-1} = u. \end{aligned}$$

□

Definition 1.5.17. Let S be a signature and let V be a set of variables. An (S,V)-*substitution* is a substitution whose domain is the set V and whose range is a subset of $\mathrm{TERM}_S(V)$, i.e., a mapping $s : V \longrightarrow \mathrm{TERM}_S(V)$. ▯

If the signature S and the set of variables V are clear from the context, we will use the term "substitution" rather than (S,V)-substitution.

Definition 1.5.18. Let S be a signature and let V be a set of variables. The *identity* (S,V)-*substitution* is the substitution $\iota_{(S,V)} : V \longrightarrow \mathrm{TERM}_S(V)$ given by $\iota_{(S,V)}(x) = x$, for all $x \in V$.

When the signature and set of variables are clear from the context, we will write ι rather than $\iota_{(S,V)}$. ▯

Lemma 1.5.19. *Let S be a signature, V be a set of variables, and s be an (S,V)-substitution. Then,*

$$\begin{aligned}\overline{s}(x) &= s(x),\\ \overline{s}(a) &= a,\\ \overline{s}(f(t_0,\ldots,t_{n-1})) &= f(\overline{s}(t_0),\ldots,\overline{s}(t_{n-1}))\end{aligned}$$

for all variables x, constant symbols a, n-ary function symbols f (with $n > 0$), and terms $t_0,\ldots,t_{n-1}$.

Proof. The argument follows from the definition of $\overline{s}$ and Theorem 1.2.11. □

Theorem 1.5.20. *Let s be an (S,V)-substitution and let t be a term in $\mathrm{TERM}_S(V)$. Then, $\overline{s}(t) \in \mathrm{TERM}_S(V)$.*

Proof. The argument is by structural induction on the definition of terms. If t is a variable, then $\overline{s}(t) = s(t)$ is a term by the definition of (S,V)-substitution. If t is a constant symbol, then $\overline{s}(t) = t$.

Let f be an n-ary function symbol ($n > 0$) and suppose that $\overline{s}(t_i)$ is a term for $0 \leq i \leq n-1$. Then, by Lemma 1.5.19 and the definition of term, $\overline{s}(f(t_0,\ldots,t_{n-1})) = f(\overline{s}(t_0),\ldots,\overline{s}(t_{n-1}))$ is a term. □

Whenever s is an (S,V)-substitution, we shall regard $\overline{s}$ as a function $\overline{s} : \mathrm{TERM}_S(V) \longrightarrow \mathrm{TERM}_S(V)$, rather than consider $\overline{s}(z)$ for arbitrary sequences z, as we did in Section 1.2.

We remind the reader that every finite substitution s can be written as $s = \mathsf{s}^{x_0\cdots x_{n-1}}_{t_0\cdots t_{n-1}}$ for some n, distinct variables x_i and terms t_i for $0 \leq i \leq n-1$. Since we do not assume that $t_i \neq x_i$, this representation is not unique. For instance, $s^{x_0}_{x_0}$, $s^{x_0x_1}_{x_0x_1}$, and $s^{x_0\cdots x_{n-1}}_{x_0\cdots x_{n-1}}$ all denote the identity (S,V)-substitution, where $x_0,\ldots,x_{n-1}$ all belong to V.

Example 1.5.21. Let $S = (\{a,f,g,h\},\nu)$ be the signature introduced in Example 1.5.3 and let $V = \{x_n | n \in \mathbf{N}\}$ be a set of variables.

Let $t \in \mathrm{TERM}_S(V)$ be the term $h(x_1,x_0)$. If $t_1 = g(f(a), f(f(a)))$, then $s^{x_0x_1}_{x_0t_1}(t) = h(g(f(a),f(f(a))),x_0)$. Of course, $s^{x_0x_1}_{x_0t_1}$ is the same substitution as $s^{x_1}_{t_1}$. □

Example 1.5.22. Let $S = (\{a,f,\},\nu)$ be a signature with $\nu(a) = 0$ and $\nu(f) = 3$ and let $V = \{x_0,x_1\}$. Consider the substitution $s^{x_0x_1}_{a\,x_0}$. If $t = f(x_0,x_1,a) \in \mathrm{TERM}_S(V)$, then $s^{x_0x_1}_{ax_0}(t) = f(a,x_0,a)$. □

Theorem 1.5.23. *Let s be an (S, V)-substitution and let t be a term in* $\mathrm{TERM}_S(V)$. *Then,* $\mathtt{V}(\overline{s}(t)) = \bigcup\{\mathtt{V}(s(x)) \mid x \in \mathtt{V}(t)\}$.

Proof. The argument is left to the reader. □

The following theorem shows that $\{\overline{s} \mid s$ is an (S, V)-substitution$\}$ is closed under composition.

Theorem 1.5.24. *Let s_0, s_1 be two (S, V)-substitutions. Then, $s_0 * s_1$ is an (S, V)-substitution.*

Proof. Let $s = s_0 * s_1$. By Theorem 1.2.19, since $\mathrm{Dom}(s_0) = \mathrm{Dom}(s_1) = V$, we have $s(x) = \overline{s_0}(s_1(x))$ for all $x \in V$. Since $s_1(x)$ is an (S, V)-term, by Theorem 1.5.20, $\overline{s_0}(s_1(x))$ is an (S, V)-term for every $x \in V$ and this proves that s is an (S, V)-substitution. □

Theorem 1.5.25. *Let S be a signature and let V' and V be two sets of variables such that $V' \subseteq V$. If s_0, s_1 are two (S, V)-substitutions such that $s_0 \upharpoonright V'$ and $s_1 \upharpoonright V'$ are (S, V')-substitutions, then $(s_0 * s_1) \upharpoonright V'$ is an (S, V')-substitution.*

Proof. For $x \in V'$, $s_1(x)$ is an (S, V')-term, so by Theorem 1.5.23, $\overline{s_0}(s_1(x))$ is an (S, V')-term. By the definition of substitution composition, it follows that $(s_0 * s_1) \upharpoonright V'$ is an (S, V')-substitution. □

Definition 1.5.26. Let t be an (S, V)-term. A *subterm* of t is a (S, V)-term which is a subsequence of t.

We denote the set of subterms of a term t by $\mathrm{SUBT}(t)$.

Any subterm of an (S, V)-term t distinct from t is a *proper subterm.* ▯

Note that we did not incorporate the signature into the notation $\mathrm{SUBT}(t)$ because the set of subterms of t does not depend on the signature.

Theorem 1.5.27 (Occurrence Theorem for Terms). *Let S be a signature and V be a set of variables. If an (S, V)-term t is a proper subterm of an (S, V)-term $u = f(t_0, \ldots, t_{n-1})$, then every occurrence (t, j) of t in u is part of some term t_i (or, more accurately, (t, j) is a part of one of the occurrences (t_i, m_i), where $0 \leq i \leq n-1$ and $m_i = 2 + i + \sum_{0 \leq k \leq i-1} |t_k|$).*

Proof. If $j = 0$, then t is a prefix of u, which implies by Lemma 1.5.11 that $t = u$, thus contradicting the assumption that t is a proper subterm of u. Since t may not begin with a parenthesis or comma, the occurrence of t must begin inside one of the terms t_i. If the occurrence of t extended beyond the end of this t_i, then a suffix of t_i would be a proper prefix of t which contradicts Exercise 53, Part (a). □

Theorem 1.5.28. *Let S be a signature and V be a set of variables. The function* SUBT *satisfies the following conditions:*

(1) *If t is a constant symbol or a variable, then* $\mathrm{SUBT}(t) = \{t\}$.
(2) *If f is an n-ary function symbol and $t = f(t_0, \ldots, t_{n-1})$, then*

$$\mathrm{SUBT}(t) = \{t\} \cup \bigcup_{0 \leq i \leq n-1} \mathrm{SUBT}(t_i).$$

Proof. This is an immediate consequence of Theorem 1.5.27 and we leave the details to the reader. □

Example 1.5.29. Let $S = (\{a, f, g, h\}, \nu)$ be the signature introduced in Example 1.5.3 and let $V = \{x_0, x_1\}$. The set of subterms of the term $t = h(f(f(a)), f(x_1))$ is given by

$$\begin{aligned}\mathrm{SUBT}(t) &= \{t\} \cup \mathrm{SUBT}(f(f(a))) \cup \mathrm{SUBT}(f(x_1)) \\ &= \{t, f(f(a)), f(a), a, f(x_1), x_1\}.\end{aligned}$$

▯

Theorem 1.5.30. *Let S be a signature and V be a set of variables. If t is an (S, V)-term and (u, j) is an occurrence of an (S, V)-term u in t, then for every (S, V)-term w,* `replace` $(t, (u, j), w)$ *is an (S, V)-term.*

Proof. The argument is by structural induction on the definition of terms. If t is a variable, that is, if $t = x$, then we have $u = x, j = 0$ and `replace` $(t, (u, j), w) = w$, so the statement clearly holds. If t is a constant symbol, a similar argument applies.

Let $t = f(t_0, \ldots, t_{n-1})$ be an (S, V)-term. Suppose that for every i, $0 \leq i \leq n-1$, `replace` $(t_i, (u, \ell), w)$ is an (S, V)-term for every $u, w \in \mathrm{TERM}_S(V)$ such that $(u, \ell) \in \mathtt{OCC}_u(t_i)$. Let (u, j) be an occurrence of u in t. If $u = t$, then $j = 0$ and `replace` $(t, (u, j), w) = w$, which is a term. If $u \neq t$, then Theorem 1.5.27 shows that the occurrence (u, j) is a part of some occurrence (t_i, m_i), where $0 \leq i \leq n-1$ and $m_i = 2 + i + \sum_{0 \leq k \leq i-1} |t_k|$. By Exercise 8,

$$\texttt{replace}\,(t,(u,j),w) = f(t_0,\ldots,\texttt{replace}\,(t_i,(u,\ell),w),\ldots,t_{n-1}),$$

where $\ell = j - (2 + i + \sum_{0 \leq k \leq i-1} |t_k|)$. Since $\texttt{replace}\,(t_i,(u,\ell),w)$ is a (S,V)-term by the inductive hypothesis, $\texttt{replace}\,(t,(u,j),w)$ is also a (S,V)-term. □

1.6 Term Unification

Unification is a process through which we decide whether there exists a substitution that maps all members of a finite set of terms into the same term. In this section, we will give an algorithm that enables us to make this decision. Unification is a tool used in the resolution proof method in first-order logic, as we shall see in Chapter 5.

Definition 1.6.1. Let S be a signature, S' be a reduct of S, V be a set of variables, and let $t_0,\ldots,t_{n-1} \in \text{TERM}_S(V)$. An (S,S',V)-*unifier* for $\{t_0,\ldots,t_{n-1}\}$ is an (S',V)-substitution s such that $s(t_0) = \cdots = s(t_{n-1})$. If s is a finite substitution, we refer to s as a *finite* (S,S',V)*-unifier.*

The finite set $T = \{t_0,\ldots,t_{n-1}\}$ of terms is (S,S',V)*-unifiable* if it has an (S,S',V)-unifier. We shall denote by $\mathbf{UNIF}^{S,V}_{S'}(T)$ the set of (S,S',V)-unifiers for the set T. An (S,S,V)-unifier will be referred to as an (S,V)*-unifier* and the set $\mathbf{UNIF}^{S,V}_{S}(T)$ will be written as $\mathbf{UNIF}^{S,V}(T)$. ▯

For any substitution s and any finite set of (S,V)-terms T, we have $|s(T)| \leq |T|$. Note that if s is a unifier for $T \neq \emptyset$, then $|s(T)| = 1$.

Example 1.6.2. Let f be a binary function symbol of the signature S, a, b be two constant symbols of S, and let V be a set of variables that contains u and v. The set $T = \{f(u,a), f(b,v)\}$ is (S,V)-unifiable because we have

$$s^{\,u\,v}_{\,b\,a}(f(u,a)) = s^{\,u\,v}_{\,b\,a}(f(b,v)) = f(b,a).$$

Observe that the set $\{f(a,v), f(b,u)\}$ is not (S,V)-unifiable because any substitution will leave intact the distinct constant symbols a and b.

If S' is the reduct of S that contains only a and b, then $s^{u\,v}_{b\,a}$ is an (S,S',V)-unifier of T. If, on the other hand, S'' is the reduct of S that consists only of a, then T has no (S,S'',V)-unifier. □

Example 1.6.3. Let f be a binary function symbol of the signature S'. Suppose now that a,b,u,v are all constant symbols of S' (contrast this with Example 1.6.2) and U is a set of variables. Then, the set of terms T of that example is not (S',U)-unifiable.

The point of this example is that unifiability is not an intrinsic property of the set of terms involved. Rather, it depends also on the signature and the set of variables involved because we need to be able to discriminate between variables and constant symbols. □

A set of terms T may have more than one unifier as shown by the following example.

Example 1.6.4. Suppose that f,g are function symbols of $S = (F,\nu)$, a is a constant symbol of S, and V is a set of variables that contains x_0 and x_1. Assume that $\nu(f) = 2$ and $\nu(g) = 1$. The set $T = \{f(g(x_0),x_1), f(g(a),x_2)\}$ is (S,V)-unifiable because both $s^{x_0\,x_1}_{a\;\,x_2}$ and $s^{x_0\,x_1\,x_2}_{a\;\,a\;\,a}$ are (S,V)-unifiers for T. □

Note that for the substitutions $s^{x_0\,x_1}_{a\;\,x_2}$ and $s^{x_0\,x_1\,x_2}_{a\;\,a\;\,a}$ considered in Example 1.6.4, we have $s^{x_0\,x_1\,x_2}_{a\;\,a\;\,a} = s^{x_2}_{a} * s^{x_0\,x_1}_{a\;\,x_2}$. We claim that for any (S,V)-unifier s for the terms $f(g(x_0),x_1), f(g(a),x_2)$ we must have a substitution s' such that $s = s' * s^{x_0\,x_1}_{a\;\,x_2}$. Indeed, arguing informally, if s is an (S,V)-unifier for these terms, then s must map x_0 into a and must transform x_1 and x_2 into the same term t. This can always be accomplished by transforming first x_1 into x_2 and, then, transforming x_2 into t. For instance, the (S,V)-unifier $s^{x_0\;\;x_1\;\;\;x_2}_{a\;\,g(b)\,g(b)}$ can be written as

$$s^{\begin{matrix} x_0 & x_1 & x_2 \end{matrix}}_{\begin{matrix} a & g(b) & g(b) \end{matrix}} = s^{x_2}_{g(b)} * s^{\begin{matrix} x_0 & x_1 \end{matrix}}_{\begin{matrix} a & x_2 \end{matrix}}.$$

These remarks motivate the following definition.

Definition 1.6.5. Let S be a signature, S' be a reduct of S, V be a set of variables, and let T be a finite subset of $\text{TERM}_S(V)$.

A *most general* (S, S', V)*-unifier* (mgu) for a finite set of (S, V)-terms T is an (S, S', V)-unifier s' for T such that any (S, S', V)-unifier s'_0 for T can be written as $s'_0 = s_1 * s'$ for some (S', V)-substitution s_1.

We shall denote by $\mathbf{MGUNIF}^{S,V}_{S'}(T)$ the set of most general (S, S', V)-unifiers for the set T. A most general (S, S, V)-unifier will be referred to as a *most general* (S, V)*-unifier* and the set $\mathbf{MGUNIF}^{S,V}_{S}(T)$ will be written as $\mathbf{MGUNIF}^{S,V}(T)$. $\square$

Let T be a set of (S, V)-terms. If s_0, s_1 are both most general (S, S', V)-unifiers for T, then $s_0 \equiv s_1$, where $\equiv$ is the equivalence on the set of (S', V)-substitutions introduced in Supplement 68. We extend this remark in the following theorem.

Theorem 1.6.6. *Let T be a set of (S, V)-terms and let s_0 be an (S, S', V) mgu of T. Then, an (S', V)-substitution s_1 is an (S, S', V)-mgu of T if and only if $s_0 \equiv s_1$.*

Proof. We need to prove only that if $s_0 \equiv s_1$, then s_1 is an mgu of T. Suppose that $s_1 = s' * s_0$ and $s_0 = s'' * s_1$. We have $|s_1(T)| = |s' * s_0(T)| = |s'(s_0(T))| = 1$, due to the fact that s_0 is a unifier of T. Now suppose that z is an arbitrary (S', V)-unifier of T. Then for some substitution z', $z = z' * s_0$ because s_0 is an mgu of T. Therefore, $z = z' * (s'' * s_1) = (z' * s'') * s_1$, which shows that s_1 is an mgu of T. $\square$

Let t, u be two distinct terms in $\text{TERM}_S(V)$. Since no term can be a proper prefix of another, there is a least number $i \in \mathbf{N}$ such that $0 \le i < \min\{|t|, |u|\}$ and the $i + 1$st symbols of t and u are different. We will denote this number (called the *disagreement position* of t and u) by $\text{DIS}(t, u)$.

Example 1.6.7. Let $t = f(x_0, a)$ and $u = f(f(x_0, a), x_0)$ be two terms. Then, $\text{DIS}(t, u) = 2$ because the first two symbols of t

and u are equal while the third symbols of t and u are x_0 and f, respectively. $\square$

Let $T = \{t_0, \ldots, t_{n-1}\}$ be a finite set of (S, V)-terms, where $n \geq 2$. The *disagreement position of* T, denoted by $\mathrm{DIS}(T)$, is defined by $\mathrm{DIS}(T) = \min\{\mathrm{DIS}(t, u) \mid t, u \in T \text{ and } t \neq u\}$. In other words, $\mathrm{DIS}(T)$ is the leftmost position in which not all terms of T have the same symbol.

Theorem 1.6.8. *Let t, u be two distinct (S, V)-terms. Then, there is a subterm t' of t beginning at position* $\mathrm{DIS}(t, u)$ *in t and there is a subterm u' of u beginning at position* $\mathrm{DIS}(t, u)$ *in u.*

Proof. We will show by induction on the term t that for every term u, $u \neq t$, there is a subterm t' of t that begins at position $\mathrm{DIS}(t, u)$ in t. This suffices because the roles of t and u are reversible.

The basis steps, when t is a variable or a constant symbol, are immediate because then $\mathrm{DIS}(t, u) = 0$ and $t' = t$.

Suppose now that $t = f(t_0, \ldots, t_{n-1})$, where $n > 0$, and the statement holds for $t_0, \ldots, t_{n-1}$. If u does not begin with the symbol f, then $\mathrm{DIS}(t, u) = 0$ and $t' = t$. Therefore, we can assume that $u = f(u_0, \ldots, u_{n-1})$ and so, $\mathrm{DIS}(t, u) \notin \{0, 1\}$. This leaves three cases to consider.

The first case occurs when the symbol of t in the disagreement position is a comma immediately following one of the subterms t_i of t. More precisely, if this is the ℓth such comma, where $1 \leq \ell \leq n-1$, we have $\mathrm{DIS}(t, u) = \ell + 1 + \sum\{|t_i| \mid 0 \leq i \leq \ell - 1\}$. Note that $t_0 = u_0$ because no term is a proper prefix of another. This implies $t_1 = u_1$, and so on, until we get $t_{\ell-1} = u_{\ell-1}$, which, in turn, means that the symbol of u at position $\mathrm{DIS}(t, u)$ is also a comma, thereby contradicting the definition of $\mathrm{DIS}(t, u)$.

The second case takes place when $\mathrm{DIS}(t, u) = |t| - 1$, so the symbol of t at position $\mathrm{DIS}(t, u)$ is a close parenthesis. By the argument of the previous case, we have $t_i = u_i$ for $0 \leq i \leq n-1$. Thus, the symbol of u at position $\mathrm{DIS}(t, u)$ is also a close parenthesis, thereby contradicting the definition of $\mathrm{DIS}(t, u)$.

The third case occurs when the disagreement position falls within one of the subterms t_i, say t_j. (More precisely, $j + 2 + \sum\{|t_i| \mid 0 \leq i \leq j - 1\} \leq \mathrm{DIS}(t, u) < j + 2 + \sum\{|t_i| \mid 0 \leq i \leq j\}$.) As in the previous case, $t_0 = u_0, \ldots, t_{j-1} = u_{j-1}$. Further, since u_j cannot be a proper prefix of t_j, $\mathrm{DIS}(t, u)$ falls within u_j. By the inductive hypothesis, there is a subterm t' of t_j starting at position

$$\mathrm{DIS}(t_j, u_j) = \mathrm{DIS}(t, u) - \left(j + 2 + \sum\{|t_i| \mid 0 \leq i \leq j - 1\}\right).$$

Since t', as a subterm of t, begins at position $\mathrm{DIS}(t, u)$ in t, this concludes our argument. □

Corollary 1.6.9. *Let T be a finite set of (S, V)-terms, where $|T| \geq 2$. Then, for each $t \in T$, there is a subterm t' of t that begins at position* $\mathrm{DIS}(T)$*.*

Proof. The definition of $\mathrm{DIS}(T)$ implies the existence of two distinct terms $u_0, u_1 \in T$ such that $\mathrm{DIS}(u_0, u_1) = \mathrm{DIS}(T)$. We can write $u_0 = xs_0u_0'$ and $u_1 = xs_1u_1'$, where $|x| = \mathrm{DIS}(T)$ and s_0, s_1 are two distinct symbols. Let now t be an arbitrary term in T. The definition of $\mathrm{DIS}(T)$ implies that t can be written as $t = xst'$ for some symbol s. Observe now that for some $i \in \{0, 1\}$, $s \neq s_i$, and thus $\mathrm{DIS}(t, u_i) = \mathrm{DIS}(u_0, u_1) = \mathrm{DIS}(T)$. Applying Theorem 1.6.8 to t and u_i gives the result. □

This corollary justifies the following definition.

Definition 1.6.10. Let T be a finite set of (S, V)-terms with $|T| \geq 2$. The *disagreement set* of T is the set $\Delta(T)$ that consists of all subterms beginning at position $\mathrm{DIS}(T)$ in each of the terms of T. If $|T| < 2$, then $\Delta(T) = \emptyset$. ▯

Note that, if $|T| > 1$, each term of T contributes exactly one of its subterms to $\Delta(T)$ because a position in a term determines at most one subterm of that term. Of course, some of the terms contributed may be the same, so in general $|\Delta(T)| \leq |T|$.

Example 1.6.11. Consider the set of (S, V)-terms

$$T = \{f(x_0, a), f(f(a, x_1), x_2), f(f(a, x_1), a)\},$$

where f is a binary function symbol, a is a constant symbol, and x_0, x_1 and x_2 are variables. We have $\mathrm{DIS}(T) = 2$ and $\Delta(T) = \{x_0, f(a, x_1)\}$. ▯

Theorem 1.6.12. *Let S be a signature, S' be a reduct of S, and V be a set of variables. For every finite set of (S, V)-terms T, we have*

$$\mathbf{UNIF}_{S'}^{S,V}(T) \subseteq \mathbf{UNIF}_{S'}^{S,V}(\Delta(T)).$$

Proof. Let T be a finite set of (S, V)-terms. If $s \in \mathbf{UNIF}_{S'}^{S,V}(T)$, then for some (S, V)-term w, $s(t) = w$, for every $t \in T$. Consider now two (S, V)-terms u, u' from $\Delta(T)$, such that $u \neq u'$. There exist $t, t' \in T$ such that $t = vur$ and $t' = vu'r'$, where v, r, r' are sequences of variables, function symbols, and punctuation symbols. Thus, we have $s(t) = s(v)s(u)s(r)$ and $s(t') = s(v)s(u')s(r')$, where $w = s(t) = s(t'), s(u)$ and $s(u')$ are all (S, V)-terms. Since both $s(u)$ and $s(u')$ are subterms of the term w that begin from the same position in w, we have $s(u) = s(u')$ (because, otherwise, one of these terms would be a proper prefix of the other). This shows that s is also an (S, S', V)-unifier for $\Delta(T)$. □

Corollary 1.6.13. *Let S be a signature, S' be a reduct of S, V be a set of variables, and T be a finite set of (S, V)-terms. If the disagreement set of T is not (S, S', V)-unifiable, then the set T is not (S, S', V)-unifiable.*

Proof. The corollary follows immediately from Theorem 1.6.12. □

Definition 1.6.14. Let S be a signature, V be a set of variables, x be a variable in V, and $t \neq x$ be an (S, V)-term. The pair (x, t) *passes the occurrence check* if x does not occur in t. Otherwise, we say that (x, t) *fails the occurrence check.*

A set of terms T *passes the occurrence check* if every pair of distinct terms (x, t) such that $x, t \in \Delta(T)$ and x is a variable passes the occurrence check. Otherwise, we say that T *fails the occurrence check.* ▯

Lemma 1.6.15. *Let S be a signature and V be a set of variables. If T is a finite set of (S, V)-terms that fails the occurrence check, then for every reduct S' of S, T is not (S, S', V)-unifiable.*

Proof. Let T be a finite set of (S, V)-terms that fails the occurrence check and let S' be a reduct of S. By Corollary 1.6.13, it suffices to show only that $\Delta(T)$ is not (S, S', V)-unifiable.

Let $x \in \Delta(T)$ be a variable and let $t \in \Delta(T)$ be a term such that $|t| > 1$ and x occurs in t. Suppose that $\Delta(T)$ has an (S, S', V)-unifier s. We must have $\bar{s}(x) = \bar{s}(t)$ or $s(x) = \bar{s}(t)$. Since the term t is not a variable it follows that it contains at least one function symbol and two parentheses besides the variable x. Therefore, $|\bar{s}(t)| \geq |s(x)|+3$ and this contradicts the equality $s(x) = \bar{s}(t)$. Thus, we conclude that T is not (S, S', V)-unifiable. □

Another situation where T is not unifiable is given in the following lemma.

Lemma 1.6.16. *Let S be a signature, S' be a reduct of S, V be a set of variables, and let T be a finite set of (S, V)-terms. If the disagreement set $\Delta(T)$ is nonempty and consists only of terms that are not variables, then T is not (S, S', V)-unifiable.*

Proof. Note that if a set of terms U contains two terms that begin with distinct function symbols, then U is not unifiable relative to any signature. If $\Delta(T)$ is not empty, then it contains at least two terms beginning with different symbols. Under the hypothesis of the lemma, $\Delta(T)$ contains two terms which begin with different function symbols, so $\Delta(T)$ is not (S, S', V)-unifiable, which implies that T is not (S, S', V)-unifiable, by Corollary 1.6.13. □

Lemma 1.6.17. *Let S be a signature, S' be a reduct of S, V be a set of variables, and let T be a finite set of (S, V)-terms. If the disagreement set $\Delta(T)$ contains a variable x and a term t that is not in $\mathrm{TERM}_{S'}(V)$, then T is not (S, S', V)-unifiable.*

Proof. The term t contains some function symbol that is not in the signature S'. Therefore, any (S', V)-substitution applied to t yields an (S, V)-term which is not in $\mathrm{TERM}_{S'}(V)$. The same substitution applied to x produces an (S', V)-term, which implies that $\Delta(T)$ is not (S, S', V)-unifiable and therefore, T is not (S, S', V)-unifiable, by Corollary 1.6.13. □

Algorithm 1.6.18 (Unification Algorithm).
Input: A signature S, a set of variables V, a reduct S' of S, and a finite set T of (S, V)-terms.
Output: A most general idempotent (S, S', V)-unifier for the set T, if T is (S, S', V)-unifiable; otherwise, an announcement that T is not (S, S', V)-unifiable.
Method: We construct a sequence of sets of (S, V)-terms $T_0, T_1, \ldots$ and a sequence of (S', V)-substitutions $s_0, s_1, \ldots$:

(1) Let $n = 0$, $T_0 = T$, and $s_0 = \iota$.
(2) If $|T_n| \leq 1$, output s_n and stop. Otherwise, compute $\Delta(T_n)$.
(3) If $\Delta(T_n)$ does not contain a variable x and an (S', V)-term $t \neq x$, then announce that T is not (S, S', V)-unifiable and stop. Otherwise, select one such pair (x, t). If (x, t) fails the occurrence check, announce that T is not (S, S', V)-unifiable and stop. If (x, t) passes the occurrence check, then define $s_{n+1} = s\substack{x\\ \\t} * s_n$, and construct $T_{n+1} = s\substack{x\\ \\t}(T_n)$.
(4) Increment n by 1 and go to Step 2.

Proof of Correctness: Observe that the number of variables that occur in the set T_{n+1} is strictly less than the number of variables that occur in the set T_n. Since T contains a finite number of variables, the algorithm terminates after a finite number of steps.

Note that for all n such that T_n is defined, $T_n = s_n(T)$. Therefore, if the algorithm terminates at Step 2 with $n = k$, then $|s_k(T)| \leq 1$, so the output s_k is clearly a unifier for T. (Note that $|s_k(T)| = 0$ can only happen if $T = \emptyset$.) It follows that if T is not (S, S', V)-unifiable, then the algorithm must halt at Step 3 with the announcement that T is not (S, S', V)-unifiable.

If the algorithm terminates at Step 3 with $n = k$, then the set T_k is not (S, S', V)-unifiable. Indeed, in this case, either T_k fails the occurrence check or $\Delta(T_k)$ contains no variables or $\Delta(T_k)$ contains a variable and a term u that is not an (S', V)-term. Thus, T_k is not (S, S', V)-unifiable as follows from Lemmas 1.6.15, 1.6.16, or 1.6.17, respectively.

We claim that if z is an (S, S', V)-unifier for T and T_n is defined, then $z = z * s_n$ and z is an (S, S', V)-unifier for T_n. The proof is by induction on n.

The basis step, $n = 0$, is immediate because $z = z * \iota$.

Suppose that the claim is true for n and assume that T_{n+1} is defined. Then, $T \neq \emptyset$ and, by inductive hypothesis, $z = z * s_n$ and z is an (S, S', V)-unifier for T_n. Let $x, t \in \Delta(T_n)$ be such that $s_{n+1} = s^{x}_{t} * s_n$. By Theorem 1.6.12, since z is an (S, S', V)-unifier for T_n, it is an (S, S', V)-unifier for $\Delta(T_n)$, so $z(x) = z(t)$. We have $z = z * s^{x}_{t}$ because $z * s^{x}_{t}(y) = z(y)$ if $y \neq x$ and $z * s^{x}_{t}(x) = z(t) = z(x)$. Thus, $z * s_{n+1} = z * (s^{x}_{t} * s_n) = (z * s^{x}_{t}) * s_n = z * s_n = z$. Finally, $|z(T_{n+1})| = |z(s_{n+1}(T))| = |z * s_{n+1}(T)| = |z(T)| = 1$, so z is an (S, S', V)-unifier for T_{n+1}.

Now, if T is (S, S', V)-unifiable, then the algorithm cannot stop at Step 3 because, as noted before, this would imply that the final set T_k is not (S, S', V)-unifiable and this would contradict what we have just shown. So, the algorithm must stop in this case at Step 2 and output a substitution s_k. Since $z = z * s_k$ for every (S, S', V)-unifier z for T, we conclude that s_k is a most general (S, S', V)-unifier for T. Note that, in particular, $s_k = s_k * s_k$, so s_k is also idempotent. □

Different strategies in picking the pair (x, t) at Step 3 of the Unification Algorithm will result in different variants of this algorithm. Among such strategies, one could consider the following:

- Pick the first pair (x, t) with $x \neq t$ in some systematic search through $\Delta(T_n)$; if T is (S, S', V)-unifiable, this would result in faster execution of the algorithm.
- Determine if T_n fails the occurrence check by searching through all pairs (x, t) that can be formed in $\Delta(T_n)$. If it fails, pick the first witness (x, t) to this failure; otherwise, pick the first pair (x, t). This strategy sometimes is more efficient when T is not (S, S', V)-unifiable but takes a longer time when T is (S, S', V)-unifiable because of the need to scan the entire set $\Delta(T_n)$ for each n.

In the interest of speeding up the algorithm, certain implementations associated with the PROLOG programming language omit the occurrence check for the selected pair (x, t) and just set $s_{n+1} = s_{t}^{x} * s_n$. This may cause the algorithm not to terminate when T is not unifiable; however, even with this omission, if T is unifiable, the algorithm will terminate and produce an mgu. If T is not unifiable and the algorithm terminates, it will correctly announce that T is not unifiable.

Example 1.6.19. Let

$$T = \{f(g(x_0), x_2, x_0), f(x_1, x_3, x_2), f(g(x_0), h(x_1), x_0)\}$$

be a set of (S, V)-terms where f is a ternary function symbol and g, h are two unary function symbols. We have $T_0 = T$ and $\Delta(T_0) = \{g(x_0), x_1\}$, so the set $T_1 = s_{g(x_0)}^{x_1}(T_0)$ is given by

$$T_1 = \{f(g(x_0), x_2, x_0), f(g(x_0), x_3, x_2), f(g(x_0), h(g(x_0)), x_0)\}.$$

The new disagreement set is $\Delta(T_1) = \{x_2, x_3, h(g(x_0))\}$ and here we have four choices for the substitution:

$$s_{x_3}^{x_2}, s_{x_2}^{x_3}, s_{h(g(x_0))}^{x_2} \quad \text{or} \quad s_{h(g(x_0))}^{x_3}.$$

Suppose that we chose to continue the application of the algorithm with the substitution $s_{h(g(x_0))}^{x_2}$. We have

$$T_2 = s_{h(g(x_0))}^{x_2}(T_1) = \{f(g(x_0), h(g(x_0)), x_0), f(g(x_0), x_3, h(g(x_0)))\}.$$

The new disagreement set is $\Delta(T_2) = \{x_3, h(g(x_0))\}$ and this means that the new substitution is $s_{h(g(x_0))}^{x_3}$. This gives

$$T_3 = \{f(g(x_0), h(g(x_0)), x_0), f(g(x_0), h(g(x_0)), h(g(x_0)))\}.$$

Finally, we conclude that the set T is not (S, V)-unifiable because the new disagreement set is $\Delta(T_3) = \{x_0, h(g(x_0))\}$ and x_0 occurs in $h(g(x_0))$.

If the occurrence check were omitted, the algorithm continues with the substitution $s_{h(g(x_0))}^{x_0}$. This would yield $\Delta(T_4) = \Delta(T_3)$ and the process continues indefinitely. $\square$

Example 1.6.20. Consider the set of terms

$$T = \{h(f(x_0), g(a, x_1)), h(f(f(x_2)), g(a, x_0))\},$$

where f is a unary function symbol, g, h are two binary function symbols, and a is a constant symbol of the signature S. We have $\Delta(T) = \{x_0, f(x_2)\}$. The new substitution is $s^{x_0}_{f(x_2)}$, so the set T_1 is

$$T_1 = s^{x_0}_{f(x_2)}(T) = \{h(f(f(x_2)), g(a, x_1)), h(f(f(x_2)), g(a, f(x_2)))\}.$$

Again $|T_1| > 1$ and $\Delta(T_1) = \{x_1, f(x_2)\}$. Thus, the new substitution is $s^{x_1}_{f(x_2)}$. Now, this gives

$$T_2 = s^{x_1}_{f(x_2)}(T_1) = \{h(f(f(x_2)), g(a, f(x_2)))\},$$

and, because $|T_2| = 1$, we conclude that T is (S, V)-unifiable. The substitution

$$s = s^{x_1}_{f(x_2)} * s^{x_0}_{f(x_2)} * \iota = s^{x_0\ \ x_1}_{f(x_2)\, f(x_2)}$$

is the mgu produced by the algorithm. $\square$

Example 1.6.21. Let $T = \{f(g(a), h(x_0)), f(x_1, x_1)\}$ be a set of (S, V)-terms, where f is a binary function symbol, g, h are unary function symbols, and a is a constant symbol. We have $T_0 = T$ and $\Delta(T_0) = \{g(a), x_1\}$. Therefore, $s_1 = s^{x_1}_{g(a)}$ and

$$T_1 = s_1(T_0) = \{f(g(a), h(x_0)), f(g(a), g(a))\}.$$

In turn, we obtain $\Delta(T_1) = \{h(x_0), g(a)\}$. Since there is no variable among the terms of $\Delta(T_1)$, we conclude that T is not (S, V)-unifiable. $\square$

Example 1.6.22. Let

$$T = \{f(g(x_2), x_2, x_0), f(x_1, x_3, x_2), f(g(x_2), h(x_0), x_0)\}$$

be a set of S-terms, where S is the signature considered in Example 1.6.19 and let S' be the reduct of S that contains just f and g.

If we apply the unification algorithm to find an (S, S', V)-unifier of T, we obtain $T_0 = T$, $\Delta(T_0) = \{g(x_2), x_1\}$, and the new set of terms is

$$T_1 = \{f(g(x_2), x_2, x_0), f(g(x_2), x_3, x_2), f(g(x_2), h(x_0), x_0)\}.$$

The new difference set is $\Delta(T_1) = \{x_2, x_3, h(x_0)\}$. Since $h(x_0)$ is not an (S', V)-term, we must replace x_2 with x_3 (or vice-versa), obtaining

$$T_2 = \{f(g(x_3), x_3, x_0), f(g(x_3), x_3, x_3), f((g(x_3), h(x_0), x_0)\}.$$

The new difference set is $\Delta(T_2) = \{x_3, h(x_0)\}$ and here the process halts with the conclusion that T is not (S, S', V)-unifiable, since h is not a function symbol in S'. Note however that the set T is (S, V)-unifiable. ▯

Let S be a signature and S_0, S_1 be two reducts of S. Define the signature $S_0 \sqcap S_1$ as the reduct of S that involves the function symbols in both S_0 and S_1. Note that $S_0 \sqcap S_1$ is the largest common reduct of S_0 and S_1.

Lemma 1.6.23. *Let T be a set of (S, V)-terms. Define the reduct $\mathtt{r}_{S,T,min}$ of S that consists of all function symbols that occur in terms of T.*

If S' is a reduct of S, then T is (S, S', V)-unifiable if and only if T is $(\mathtt{r}_{S,T,min}, S' \sqcap \mathtt{r}_{S,T,min}, \mathtt{V}(T))$-unifiable.

Proof. Let T be a set of (S, V)-terms such that T is (S, S', V)-unifiable. The application of the Unification Algorithm to S, V, S' and T yields an (S, S', V)-unifier. This substitution is s_p, where $s_0, s_1, \ldots, s_p$ and $T_0, T_1, \ldots, T_p$ are produced by the algorithm. We will prove by induction on n that each substitution $s_n \restriction \mathtt{V}(T)$ is an $(S' \sqcap \mathtt{r}_{S,T,\min}, \mathtt{V}(T))$-substitution and each T_n is a set of $(\mathtt{r}_{S,T,\min}, \mathtt{V}(T))$-terms.

For the basis step $n = 0$, this is obvious. Suppose that the result holds for $n < p$. Since T_n is a set of $(\mathtt{r}_{S,T,\min}, \mathtt{V}(T))$-terms, so is $\Delta(T_n)$, and, therefore, the term t chosen at the $(n+1)$st execution of the third step of the algorithm is an $(S' \sqcap \mathtt{r}_{S,T,\min}, \mathtt{V}(T))$-term. Since $s_{n+1} = s^{x}_{t} * s_n$, by Theorem 1.5.25, $s_{n+1} \restriction \mathtt{V}(T)$ is an $(S' \sqcap \mathtt{r}_{S,T,\min}, \mathtt{V}(T))$-substitution. Finally, since $T_{n+1} = s^{x}_{t}(T_n)$, it follows that T_{n+1} is a set of $(\mathtt{r}_{S,T,\min}, \mathtt{V}(T))$-terms.

The reverse direction is immediate. □

The following theorem shows that the unifiability of a set of terms is independent of the function symbols in the signature that do not occur in the set of terms.

Theorem 1.6.24. *Let S be a signature, V be a set of variables, S_0, S_1 be two reducts of S, and V_0, V_1 be two subsets of V. If T is a set of (S, V)-terms such that $T \subseteq \mathrm{TERM}_{S_0}(V_0) \cap \mathrm{TERM}_{S_1}(V_1)$, then T is (S_0, V_0)-unifiable if and only if T is (S_1, V_1)-unifiable.*

Proof. The requirement that S_0, S_1 are reducts of S serves to ensure that S_0, S_1 are coherent, that is, function symbols which belong to both have the same arity.

By Lemma 1.6.23, the following statements are easily seen to be equivalent:

(1) T is (S_0, V_0)-unifiable.
(2) T is (S_0, S_0, V_0)-unifiable.
(3) T is $(\mathtt{r}_{S,T,\min}, S_0 \sqcap \mathtt{r}_{S,T,\min}, \mathtt{V}(T))$-unifiable.
(4) T is $(\mathtt{r}_{S,T,\min}, \mathtt{V}(T))$-unifiable.

In the same way, we can prove that T is (S_1, V_1)-unifiable if and only if T is $(\mathtt{r}_{S,T,\min}, \mathtt{V}(T))$-unifiable, which allows us to reach the desired conclusion. □

Lemma 1.6.25. *Let S be a signature, V be a set of variables, S' be a reduct of S, and T be a set of (S, V)-terms. If T is unifiable, then the most general unifier produced by Algorithm 1.6.18 applied to S, V, S', and T is an $(S' \sqcap \mathtt{r}_{S,T,min}, V)$-substitution.*

Proof. This statement follows from the proof of Lemma 1.6.23. □

Theorem 1.6.26. *Let S be a signature, V be a set of variables, S' be a reduct of S, and T be a set of (S, V)-terms. If s is a most general (S, S', V)-unifier for T, then s is an $(S' \sqcap \mathtt{r}_{S,T,\min}, V)$-substitution.*

Proof. Since T is (S, S', V)-unifiable, an application of Algorithm 1.6.18 would yield a most general (S, S', V)-unifier z. By Lemma 1.6.25, z is an $(S' \sqcap \mathtt{r}_{S,T,\min}, V)$-substitution. Since s is a most general (S, S', V)-unifier, there is an (S', V)-substitution s_1 such that $z = s_1 * s$. It follows that s is an $(S' \sqcap \mathtt{r}_{S,T,\min}, V)$-substitution. Indeed, for every variable $x \in V$, $z(x) = \overline{s_1}(s(x))$ is an $(S' \sqcap \mathtt{r}_{S,T,\min}, V)$-term, which means that $s(x)$ must also be an $(S' \sqcap \mathbf{r}_{S,T,\min}, V)$-term because $\overline{s_1}$ does not affect any of the function symbols in $s(x)$. □

1.7 Labeled Ordered Trees

Rooted trees are usually defined as acyclic and connected directed graphs with a vertex selected as the root. In this section, we pursue a different approach that originated with Gorn [17]. There are two new features of the trees introduced here: the immediate descendants of each node are ordered and nodes are labeled. As we shall see, labeled ordered trees play an essential role in presenting proofs in various formalized versions of logical deduction.

A variant of labeled ordered tree — marked labeled ordered tree — is introduced to be used exclusively in natural deduction, a formalism later discussed in Chapters 3 and 5.

Definition 1.7.1. A *tree domain* is a subset D of Seq($\mathbf{N}$) that satisfies the following conditions:

(1) $D \neq \emptyset$.
(2) If $q \in D$, then every $r \in \text{PREF}(q)$ is also in D.
(3) For every $m \geq 0$, if $(q_0, \ldots, q_{m-1}, q_m) \in D$ and $l < q_m$, then we have $(q_0, \ldots, q_{m-1}, l) \in D$.

▯

Definition 1.7.2. A *labeled ordered tree* (or *lot* for short) is a function whose domain is a tree domain.

If $\mathtt{T}$ is a lot, U is a set, and $\text{Ran}(\mathtt{T}) \subseteq U$, then we refer to $\mathtt{T}$ as a *U-lot.*

The *label* of a node q of $\mathtt{T}$ is $\mathtt{T}(q)$.

A *sublot* a lot $\mathtt{T}$ is a lot $\mathtt{T}'$ such that $\text{Dom}(\mathtt{T}') \subseteq \text{Dom}(\mathtt{T})$ and $\mathtt{T}'(q) = \mathtt{T}(q)$ for $q \in \text{Dom}(\mathtt{T}')$. ▯

A U-lot can be regarded as a tree whose vertices have been labeled by elements of the set U and in which the descendants of every vertex have been numbered from left to right.

Note that a lot $\mathtt{T}$ is finite if and only if $\text{Dom}(\mathtt{T})$ is finite; in this case, we have $|\text{Dom}(\mathtt{T})| = |\mathtt{T}|$.

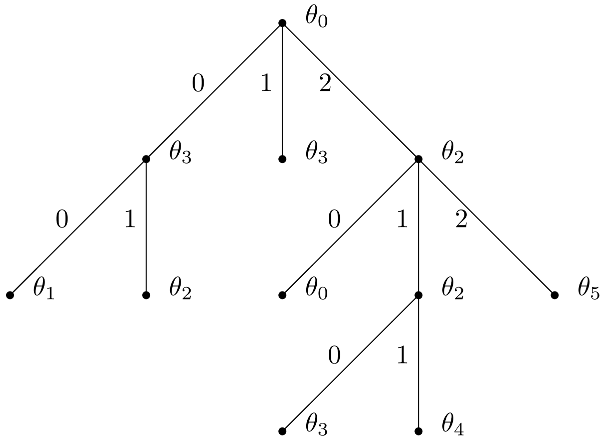


Fig. 1.2. Representation of a lot.

Example 1.7.3. Consider the lot $\mathtt{T}$ whose domain is

$$\begin{aligned}&\mathrm{Dom}(\mathtt{T})\\ &= \{\lambda, (0), (1), (2), (0,0), (0,1), (2,0), (2,1), (2,2), (2,1,0), (2,1,1)\}\end{aligned}$$

and whose values are given by

$$\begin{array}{llll} \mathtt{T}(\lambda) = \theta_0 & \mathtt{T}(0) = \theta_3 & \mathtt{T}(1) = \theta_3 & \mathtt{T}(2) = \theta_2 \\ \mathtt{T}(0,0) = \theta_1 & \mathtt{T}(0,1) = \theta_2 & \mathtt{T}(2,0) = \theta_0 & \mathtt{T}(2,1) = \theta_2 \\ \mathtt{T}(2,2) = \theta_5 & \mathtt{T}(2,1,0) = \theta_3 & \mathtt{T}(2,1,1) = \theta_4. & \end{array}$$

This lot is represented in Figure 1.2. □

Definition 1.7.4. Let D be a tree domain. The elements of D are called the *nodes* of D. If q and r are nodes of D and q is a prefix of r, then we refer to r as a *descendant* of q and to q as an *ancestor* of r. If $r = qi$ for some $i \in \mathbf{N}$, then we call r an *immediate descendant* of q and q the *immediate ancestor* of r.

The *root* of every tree domain is λ. A *leaf* of D is a node of D with no immediate descendants. We write LEAVES(D) to denote the set of leaves of D.

A *marked tree domain* is a pair (D, M), where D is a tree domain and $M \subseteq \mathrm{LEAVES}(D)$.

An *interior node* of a tree domain is a sequence $q \in D$ that is not a leaf of D. The set of interior nodes of D will be denoted by $\mathrm{INTN}(D)$.

The *level* of a node q of a tree domain D is $|q|$. The *i-th level* of D is the set of nodes at level i of D.

If every node of a tree domain D has only finitely many immediate descendants, we call D a *finitely branching* tree domain. □

Definition 1.7.5. Let D be a tree domain. A *path* of D is a non-empty subset $\mathtt{P}$ of D that satisfies the following conditions:

(1) If $q \in \mathtt{P}$, then every $r \in \mathrm{PREF}(q)$ is also in $\mathtt{P}$.
(2) For every $q, r \in \mathtt{P}$, either $q \in \mathrm{PREF}(r)$ or $r \in \mathrm{PREF}(q)$.

A *branch* of D is a path of D which is not properly contained in any other path of D.

The *length* of a finite path $\mathtt{P}$ of D is $|\mathtt{P}| - 1$.

For a finite tree domain D, we define $\mathtt{depth}(D)$ to be the length of the longest branch of D. □

If D is a tree domain and $q \in D$, then it is easy to see that $\mathrm{PREF}(q)$ is a path of D. We refer to this path as the *path in D leading to q*.

Lemma 1.7.6. *Let D be a tree domain and q be a node of D. Then,*

- *the path of D leading to q is a branch of D if and only if q is a leaf of D,*
- *a branch of D is finite if and only if it is the path of D leading to a leaf of D,*
- *an infinite path of D is a branch of D (so, a set of nodes of D is an infinite path if and only if it is an infinite branch).*

Proof. We leave the easy verification to the reader. □

Theorem 1.7.7 (König's Lemma[1]). *If a finitely branching tree domain has no infinite path, then it is finite.*

[1]Dénes König was born in 1884 in Budapest, Hungary, and died in 1944 in Budapest. König studied in Budapest and in Göttingen and obtained his Ph.D. in mathematics from the University of Budapest in 1907. He taught at the Technical University in Budapest. His main contributions were in graph theory, geometry, set theory, and partially ordered sets.

Proof. We show that in every infinite tree domain D which is finitely branching there is an infinite path $\mathtt{P}$, which clearly is equivalent to the statement of the theorem. We will construct recursively a sequence of nodes $\lambda = q_0, q_1, \ldots$ such that each q_{i+1} is an immediate descendant of q_i for all $i \in \mathbf{N}$ and then $\mathtt{P} = \{q_0, q_1, \ldots\}$ will be an infinite path of D. In order to make the construction work, we ensure that each q_i has infinitely many descendants in D. Define $q_0 = \lambda$. Since D is infinite, q_0 has infinitely many descendants in D. Now suppose that q_i is defined and has infinitely many descendants in D. Since D is finitely branching, there is a number k such that the immediate descendants of q_i are $q_i0, \ldots, q_ik$ and at least one of these immediate descendants has itself infinitely many descendants. Define $q_{i+1} = q_ij$, where j is the least number such that q_ij has infinitely many descendants. $\square$

Observe that by the third part of Lemma 1.7.6, König's lemma is equivalent to saying that a finitely branching tree domain with no infinite branch is finite.

If $\mathtt{T}$ is a lot, then by a node of $\mathtt{T}$ (or a node of $(\mathtt{T}, M)$), we mean a node of $\mathrm{Dom}(\mathtt{T})$. Similar terminology will be applied for the other notions introduced in Definition 1.7.4 and in the definitions that follow, where appropriate. Also, we write $\mathrm{LEAVES}(\mathtt{T})$ for $\mathrm{LEAVES}(\mathrm{Dom}(\mathtt{T}))$, $\mathrm{INTN}(\mathtt{T})$ for $\mathrm{INTN}(\mathrm{Dom}(\mathtt{T}))$, and $\mathrm{LEAVES}_\theta(\mathtt{T})$ for the set of leaves of $\mathtt{T}$ labeled by an object θ, that is, $\mathrm{LEAVES}_\theta(\mathtt{T}) = \mathtt{T}^{-1}(\theta) \cap \mathrm{LEAVES}(\mathtt{T})$.

Note that if $\mathtt{T}$ is a finite lot, then $|\mathtt{T}| = |\mathrm{Dom}(\mathtt{T})|$ is the number of nodes of $\mathtt{T}$.

Example 1.7.8. The leaves of the lot considered in Example 1.7.3 are $(0, 0)$, $(0, 1)$, (1), $(2, 0)$, $(2, 1, 0)$, $(2, 1, 1)$, and $(2, 2)$. The interior nodes of the same lot are $\lambda, (0), (2)$, and $(2, 1)$. ▯

Definition 1.7.9. A *marked lot* is a pair $\mathcal{T} = (\mathtt{T}, M)$, where $\mathtt{T}$ is a lot and $M \subseteq \mathrm{LEAVES}(\mathtt{T})$. ▯

For a marked lot $\mathcal{T} = (\mathtt{T}, M)$, we use the notions of leaves, internal nodes, etc. as they apply to the lot $\mathtt{T}$. For example, the set $\mathrm{LEAVES}(\mathcal{T})$ is the same as $\mathrm{LEAVES}(\mathtt{T})$.

Definition 1.7.10. Let $\mathcal{T} = (\mathtt{T}, M)$ be a marked lot and θ be an object. Then, $L_\theta(\mathcal{T})$ is the marked lot $(\mathtt{T}, M \cup \mathrm{LEAVES}_\theta(\mathtt{T}))$. ▯

The next result uses the notion of quotient of a set of sequences by a sequence introduced in Definition 1.2.1.

Lemma 1.7.11. *Let D be a tree domain and let $r \in D$. The quotient $D_{[r]} = \{q \in \mathrm{Seq}(\mathbf{N}) \mid \mathrm{rq} \in \mathrm{D}\}$ is a tree domain.*

Proof. The argument is straightforward and is left to the reader. □

Definition 1.7.12. Let r be a node of the tree domain D. If $D_{[r]}$ is a finite tree domain, then the *depth* of r in D is the depth of $D_{[r]}$. We will denote this number by $\mathtt{depth}(D)(r)$. □

Note that if D is a tree domain, $M \subseteq \mathrm{LEAVES}(D)$, and $r \in D$, then $M_{[r]} \subseteq \mathrm{LEAVES}(D_{[r]})$.

Definition 1.7.13. Let $\mathtt{T}$ be a lot with domain D and let $r \in D$. The *r-subtree* of $\mathtt{T}$ is the lot $\mathtt{T}_{[r]}$, where $\mathrm{Dom}(\mathtt{T}_{[r]}) = D_{[r]}$ and $\mathtt{T}_{[r]}(q) = \mathtt{T}(rq)$.

If $\mathcal{T} = (\mathtt{T}, M)$ is a marked lot, the *marked r-subtree* is the marked lot $\mathcal{T}_{[r]} = (\mathtt{T}_{[r]}, M_{[r]})$. □

In the above definition, we use a harmless abuse of the notation introduced in Definition 1.2.1 by writing $\mathtt{T}_{[r]}$ despite the fact that $\mathtt{T}$ is a lot and not a set of sequences.

Let $D_0, \ldots, D_{n-1}$ be n subsets of $\mathrm{Seq}(\mathbf{N})$. The set $\langle D_0, \ldots, D_{n-1}\rangle$ is defined by

$$\langle D_0, \ldots, D_{n-1}\rangle = \{\lambda\} \cup \bigcup_{0 \leq i \leq n-1} iD_i.$$

Observe that if $n = 0$, then $\langle D_0, \ldots, D_{n-1}\rangle = \{\lambda\}$.

Lemma 1.7.14. *If $D_0, \ldots, D_{n-1}$ are tree domains, then $\langle D_0, \ldots, D_{n-1}\rangle$ is also a tree domain.*

Proof. Let $D = \langle D_0, \ldots, D_{n-1}\rangle$. Clearly, $D \neq \emptyset$ because $\lambda \in D$. Suppose that $q \in D$ and let r be a prefix of q. If $r = \lambda$, then $r \in D$ by the definition of D. If $r \neq \lambda$, there exists $i \in \mathbf{N}$ such that $q = iq'$ and $q' \in D_i$. Since r is a nonnull prefix of q, there exists a sequence r' such that $r = ir'$ and r' is a prefix of q'. This implies $r' \in D_i$, so $r \in iD_i \subseteq D$, that is, $r \in D$.

Finally, suppose that $(q_0, \ldots, q_{m-1}, q_m) \in D$ for $m \in \mathbf{N}$ and $l < q_m$. If $m = 0$, then $0 < q_0 \leq n-1$, hence $0 \leq l \leq n-1$, which gives $(l) \in lD_l \subseteq D$. Otherwise, that is, if $m > 0$, we have $(q_1, \ldots, q_{m-1}, q_m) \in D_{q_0}$, so $(q_1, \ldots, q_{m-1}, l) \in D_{q_0}$. Therefore, $(q_0, \ldots, q_{m-1}, l) \in q_0 D_{q_0} \subseteq D$, which allows us to conclude that D is indeed a tree domain. □

Theorem 1.7.15. *Let $D_0, \ldots, D_{n-1}$ be n tree domains. Then,*

$$\begin{aligned}
&\text{LEAVES}(\langle D_0, \ldots, D_{n-1} \rangle) \\
&\quad = \begin{cases} \bigcup_{0 \le i \le n-1} i\, \text{LEAVES}(D_i) & \text{if } n > 0 \\ \{\lambda\} & \text{if } n = 0 \end{cases} \\
&\text{INTN}(\langle D_0, \ldots, D_{n-1} \rangle) \\
&\quad = \begin{cases} \{\lambda\} \cup \bigcup_{0 \le i \le n-1} i\, \text{INTN}(D_i) & \text{if } n > 0 \\ \emptyset & \text{if } n = 0. \end{cases}
\end{aligned}$$

Proof. The argument is straightforward and is left to the reader. □

Let $(D_0, M_0), \ldots, (D_{n-1}, M_{n-1})$ be n marked tree domains. We denote the pair

$$(\langle D_0, \ldots, D_{n-1} \rangle, 0M_0 \cup \cdots \cup (n-1)M_{n-1})$$

by $\langle (D_0, M_0), \ldots, (D_{n-1}, M_{n-1}) \rangle$.

Theorem 1.7.16. *Let $(D_0, M_0), \ldots, (D_{n-1}, M_{n-1})$ be n marked tree domains. Then the pair $\langle (D_0, M_0), \ldots, (D_{n-1}, M_{n-1}) \rangle$ is a marked tree domain.*

Proof. This statement follows immediately from Theorem 1.7.15. □

Definition 1.7.17. Let $\mathtt{T}_0, \ldots, \mathtt{T}_{n-1}$ be lots and let θ be an object. The function $\mathtt{T}$ with domain $\langle \text{Dom}(\mathtt{T}_0), \ldots, \text{Dom}(\mathtt{T}_{n-1}) \rangle$ given by

$$\mathtt{T}(\lambda) = \theta,$$
$$\mathtt{T}(iq) = \mathtt{T}_i(q)$$

for $0 \le i \le n-1$ and $q \in \text{Dom}(\mathtt{T}_i)$ will be denoted by $(\mathtt{T}_0, \ldots, \mathtt{T}_{n-1}; \theta)$ and referred to as the *θ-join of* $\mathtt{T}_0, \ldots, \mathtt{T}_{n-1}$. This construction is illustrated in Figure 1.3. ▯

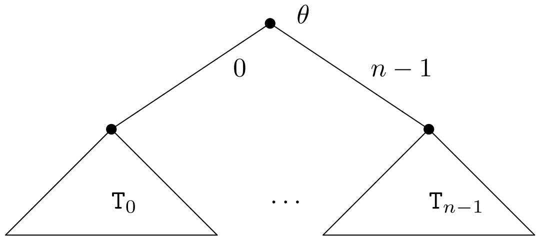


Fig. 1.3. Construction of the lot $(\mathtt{T}_0, \ldots, \mathtt{T}_{n-1}; \theta)$.

Theorem 1.7.18. *For all lots $\mathtt{T}_0, \ldots, \mathtt{T}_{n-1}$ and objects θ, the mapping $\mathtt{T} = (\mathtt{T}_0, \ldots, \mathtt{T}_{n-1}; \theta)$ is a lot.*

Proof. The theorem is an immediate consequence of Lemma 1.7.14. □

Let $\mathcal{T}_0, \ldots, \mathcal{T}_{n-1}$ be n marked lots, where $\mathcal{T}_i = (\mathtt{T}_i, M_i)$ for $0 \leq i \leq n-1$ and let θ be an object. We define $(\mathcal{T}_0, \ldots, \mathcal{T}_{n-1}; \theta)$ to be the pair

$$((\mathtt{T}_0, \ldots, \mathtt{T}_{n-1}; \theta), 0M_0 \cup \cdots \cup (n-1)M_{n-1}).$$

Theorems 1.7.16 and 1.7.18 imply that $(\mathcal{T}_0, \ldots, \mathcal{T}_{n-1}; \theta)$ is a marked lot.

Lemma 1.7.19. *Let D be a finite tree domain such that $|D| \geq 2$. Then, there is $m \in \mathbf{N}$ such that $D = \langle D_{[(0)]}, \ldots, D_{[(m)]} \rangle$.*

If $\mathtt{T}$ is a finite lot with $|\mathrm{Dom}(\mathtt{T})| \geq 2$, then there is $m \in \mathbf{N}$ such that $\mathtt{T} = (\mathtt{T}_{[(0)]}, \ldots, \mathtt{T}_{[(m)]}; \mathtt{T}(\lambda))$.

Proof. Since $|D| \geq 2$, there must be a node $q \neq \lambda$ in D. Let (i) be the prefix of length 1 of q. Since D is a tree domain, $(i) \in D$. In view of the fact that D is finite, we can define $m = \max\{i \in \mathbf{N} \mid (i) \in D\}$. Because D is a tree domain, $(0), (1), \ldots, (m) \in D$. We claim that $D = \langle D_{[(0)]}, \ldots, D_{[(m)]} \rangle$. Let $q \in D$. If $q = \lambda$, then, obviously, $q \in \langle D_{[(0)]}, \ldots, D_{[(m)]} \rangle$. Suppose, therefore, that $q \neq \lambda$ and let $q = (q_0)q'$. We have $(q_0) \in D$, so $0 \leq q_0 \leq m$. Also, $q' \in D_{[(q_0)]}$, which gives $q \in q_0 D_{[(q_0)]} \subseteq \langle D_{[(0)]}, \ldots, D_{[(m)]} \rangle$.

Conversely, let $q \in \langle D_{[(0)]}, \ldots, D_{[(m)]} \rangle$. If $q = \lambda$, then $q \in D$. Otherwise, there is i with $0 \leq i \leq m$ such that $q \in iD_{[(i)]}$, which immediately gives $q \in D$ by the definition of $D_{[(i)]}$.

We leave the proof of the second part of the lemma to the reader. □

Lemma 1.7.20. *Let D, D' be two tree domains and let $r \in D$. Then, the set $D[r \to D'] = (D - r\ \mathrm{Seq}(\mathbf{N})) \cup \mathrm{rD'}$ is a tree domain.*

Proof. Since $rD' \neq \emptyset$, it is clear that $D[r \to D'] \neq \emptyset$.

Let $q \in D[r \to D']$ and let q' be a prefix of q. If $q \in (D - r\mathrm{Seq}(\mathbf{N}))$, then q' cannot have r as a prefix, so $q' \in (D - r\ \mathrm{Seq}(\mathbf{N}))$. If $q \in rD'$, we can write $q = rq_1$ with $q_1 \in D'$. Observe that if q' is a prefix of r and $q' \neq r$, then $q' \in D - r\mathrm{Seq}(\mathbf{N})$. Otherwise, $q' = rr'$ for some prefix r' of $q_1 \in D'$. Since D' is prefix-closed, it follows that $r' \in D'$ which means that $q' \in rD'$.

To show that $D[r \to D']$ satisfies the third condition of Definition 1.7.1, let $q = (q_0, \ldots, q_m)$ be a sequence in $D[r \to D']$ and let $l \in \mathbf{N}$ be a number such that $l < q_m$. If $q \in (D - r\ \mathrm{Seq}(\mathbf{N}))$, then $(q_0, \ldots, q_{m-1}, l) = r$ or $(q_0, \ldots, q_{m-1}, l) \in (D - r\ \mathrm{Seq}(\mathbf{N}))$, so $(q_0, \ldots, q_{m-1}, l) \in D[r \to D']$. Otherwise, that is, if $(q_0, \ldots, q_{m-1}, q_m) \in rD'$, two cases are possible: either $q = r$ and $(q_0, \ldots, q_{m-1}, l) \in D[r \to D']$ because D is a tree domain or $q = rr'$, for some $r' \in D'$, $r' \neq \lambda$. In the last case, $(q_0, \ldots, q_{m-1}, l) \in D[r \to D']$ because D' is a tree domain. □

Definition 1.7.21. Let $\mathtt{T}$ and $\mathtt{T}'$ be two labeled ordered trees and let r be a node of $\mathtt{T}$. The lot $\mathtt{T}[r \to \mathtt{T}']$ *obtained by inserting* $\mathtt{T}'$ *in* $\mathtt{T}$ *at* r is the lot defined on $\mathrm{Dom}(\mathtt{T})[r \to \mathrm{Dom}(\mathtt{T}')]$ given by

$$\mathtt{T}[r \to \mathtt{T}'](q) = \begin{cases} \mathtt{T}(q) & \text{if } q \in \mathrm{Dom}(\mathtt{T}) - r\ \mathrm{Seq}(\mathbf{N}) \\ \mathtt{T}'(q') & \text{if } q = rq' \text{ for } q' \in \mathrm{Dom}(\mathtt{T}'). \end{cases}$$

□

Theorem 1.7.22. *Let $\mathcal{TD}$ be the collection of sets of sequences of natural numbers given be the following inductive definition:*

- $\{\lambda\} \in \mathcal{TD}$.
- *If $D_0, \ldots, D_k$ are in $\mathcal{TD}$, then $\langle D_0, \ldots, D_k \rangle \in \mathcal{TD}$, for every $k \geq 0$.*

Then, $\mathcal{TD}$ is the set of all finite tree domains.

Proof. Using structural induction and Lemma 1.7.14, it is clear that every member of $\mathcal{TD}$ is a finite tree domain.

Conversely, we shall prove by induction on $|D|$ that if D is a finite tree domain, then $D \in \mathcal{TD}$. If D is a tree domain such that $|D| = 1$, then $D = \{\lambda\}$, so $D \in \mathcal{TD}$. Now suppose that $n \geq 1$ and the result is true for tree domains of cardinality less than or equal to n and let D be a tree domain with $|D| = n+1$. By Lemma 1.7.19, we can write $D = \langle D_{[(0)]}, \dots, D_{[(m)]} \rangle$ for some $m \in \mathbf{N}$. Since $|D_{[(i)]}| < |D| = n+1$, for $0 \leq i \leq m$, by the inductive hypothesis, each such $D_{[(i)]} \in \mathcal{TD}$, which gives $D \in \mathcal{TD}$. □

Corollary 1.7.23 (Induction Principle for Finite Tree Domains). *Let P be a property of finite tree domains. Suppose that*

- *$P(\{\lambda\})$ is true,*
- *if $D_0, \dots, D_k$ are finite tree domains such that $P(D_i)$ is true for $0 \leq i \leq k$, then $P(\langle D_0, \dots, D_k \rangle)$ is true, for every $k \in \mathbf{N}$.*

Then, $P(D)$ is true for every finite tree domain D.

Proof. This follows immediately from Theorem 1.7.22. □

Corollary 1.7.24. *Let U be a set and let $\mathcal{LOT}(U)$ be the collection of lots given by the following inductive definition:*

- *If* $\mathrm{Dom}(\mathtt{T}) = \{\lambda\}$ *and $\mathtt{T}(\lambda) \in U$, then $\mathtt{T} \in \mathcal{LOT}(U)$.*
- *If $\mathtt{T}_0, \dots, \mathtt{T}_k \in \mathcal{LOT}(U)$ and $\theta \in U$, then $(\mathtt{T}_0, \dots, \mathtt{T}_k; \theta) \in \mathcal{LOT}(U)$.*

Then, $\mathcal{LOT}(U)$ is the set of all finite U-lots.

Proof. This is an immediate consequence of Theorem 1.7.22. □

Corollary 1.7.25 (Induction Principle for Finite *U*-Lots). *Let P be a property of finite U-lots. Suppose that*

- *$P(\mathtt{T})$ is true for every U-lot whose domain is $\{\lambda\}$,*
- *for every $k \in \mathbf{N}$, if $\mathtt{T}_0, \dots, \mathtt{T}_k$ are finite U-lots such that $P(\mathtt{T}_i)$ is true for $0 \leq i \leq k$, then $P((\mathtt{T}_0, \dots, \mathtt{T}_k; \theta))$ is true for every $\theta \in U$.*

Then, $P(\mathtt{T})$ is true for every finite U-lot $\mathtt{T}$.

Proof. This follows immediately from Theorem 1.7.22. □

Definition 1.7.26. A tree domain D' is an *extension* of a tree domain D if $D \subseteq D'$.

An extension D' of a tree domain D is a *leaf extension* of D if for every $q \in D' - D$ there is a leaf r of D that is a prefix of q. ▯

The reader can verify without difficulty that if D'' is a leaf extension of D' and D' is a leaf extension of D, then D'' is a leaf extension of D.

Since lots are functions, we can talk about one lot extending another one, namely, $\mathtt{T}'$ extends a lot $\mathtt{T}$ if $\mathrm{Dom}(\mathtt{T}) \subseteq \mathrm{Dom}(\mathtt{T}')$ and $\mathtt{T}(q) = \mathtt{T}'(q)$ for all $q \in \mathrm{Dom}(\mathtt{T})$.

Definition 1.7.27. An extension $\mathtt{T}'$ of a lot $\mathtt{T}$ is called a *leaf extension* of $\mathtt{T}$ if $\mathrm{Dom}(\mathtt{T}')$ is a leaf extension of $\mathrm{Dom}(\mathtt{T})$. ▯

Theorem 1.7.28. *Let $D_0 \subseteq D_1 \subseteq D_2 \subseteq \cdots$ be an increasing sequence of tree domains. Then, $D = \bigcup\{D_i \mid i \geq 0\}$ is a tree domain. Further, if, for each i, D_i is finite and D_{i+1} is a leaf extension of D_i and $\mathtt{B}$ is a branch of D, then, for every i, $\mathtt{B}$ contains a unique leaf q_i of D_i. If $\mathtt{B}_i$ is the branch of D_i that ends in q_i, then $\mathtt{B}_i = \mathtt{B} \cap D_i$ and $\mathtt{B} = \bigcup\{\mathtt{B}_i \mid i \geq 0\}$.*

Proof. The argument for the first part of the theorem is straightforward and is left for the reader.

For the second part, let $\mathtt{B}$ be a branch of D. For a given i, we distinguish two cases. If $\mathtt{B} \subseteq D_i$, then, certainly, $\mathtt{B}$ is a path of D_i. If $\mathtt{B}$ were not a branch of D_i, there would be a path $\mathtt{P}$ of D_i such that $\mathtt{B} \subset \mathtt{P}$. Since $\mathtt{P}$ would also be a path of D, this would contradict the assumption that $\mathtt{B}$ is a branch of D. Now, since D_i is finite, by Lemma 1.7.6, $\mathtt{B}$ contains a leaf q_i of D_i. If $\mathtt{B} \not\subseteq D_i$, then there is $r \in \mathtt{B} - D_i$. Therefore, there is $j > i$ with $r \in D_j - D_i$. Since D_j is a leaf extension of D_i, there is a leaf q_i of D_i that is a prefix of r and $q_i \in \mathtt{B}$ because $\mathtt{B}$ is prefix closed. Since no leaf of D_i can be a prefix of a different leaf of D_i, q_i is uniquely determined. We will prove that $\mathtt{B}_i = \mathtt{B} \cap D_i$. If $r \in \mathtt{B}_i$, then r is a prefix of q_i. Since q_i is both in $\mathtt{B}$ and in D_i, we have r in both of them, because they are both prefix closed, so $\mathtt{B}_i \subseteq \mathtt{B} \cap D_i$. Conversely, if $r \in \mathtt{B} \cap D_i$, then either r is a prefix of q_i or q_i is a proper prefix of r. The second case is impossible, since q_i is a leaf of D_i. Therefore, $r \in \mathtt{B}_i$, so $\mathtt{B} \cap D_i \subseteq \mathtt{B}_i$. This gives immediately $\bigcup\{\mathtt{B}_i \mid i \geq 0\} = \mathtt{B} \cap (\bigcup\{D_i \mid i \geq 0\}) = \mathtt{B}$. □

1.8 Formal Systems

In this section, we introduce the notion of formal system that will be useful later for exploring non-semantic approaches to propositional logic and first-order logic.

Definition 1.8.1. A *formal system* is a triple $\mathcal{F} = (U, A, I)$, where U is a set, called the set of *objects* of $\mathcal{F}$, A is a subset of U, called the set of *axioms* of $\mathcal{F}$, and I is a set whose elements are called the *rules of inference* of $\mathcal{F}$. If $R \in I$, then there is a number $n \in \mathbf{N}$, $n > 0$, such that R is a nonempty subset of $U^n \times U$. In this case, we refer to R as an *n-ary rule of inference* or, simply, as a *rule* of $\mathcal{F}$. □

If $\mathcal{F}$ is a formal system and R is an n-ary rule of $\mathcal{F}$, then we write

$$\frac{\theta_0, \ldots, \theta_{n-1}}{\theta}R$$

to mean $((\theta_0, \ldots, \theta_{n-1}), \theta) \in R$. When R is clear from the context, we omit it. We refer to $\theta_0, \ldots, \theta_{n-1}$ as the *hypotheses* or *premises* and to θ as the *conclusion* of this instance of the rule R and we say that θ is obtained by applying rule R to $\theta_0, \ldots, \theta_{n-1}$. Following established practice in logic, we will use the phrase "hypotheses of a rule of inference" rather than "hypotheses of an instance of a rule of inference" and similarly for the terms "premises" and "conclusion".

Definition 1.8.2. Let $\mathcal{F}$ be a formal system. The set $\mathrm{Thm}(\mathcal{F})$ of *theorems* of $\mathcal{F}$ is the set of objects of $\mathcal{F}$ given by the following inductive definition:

- Every axiom of $\mathcal{F}$ is a theorem of $\mathcal{F}$.
- If every hypothesis of an instance of a rule of $\mathcal{F}$ is a theorem of $\mathcal{F}$, then so is the conclusion of that instance.

□

A formal system amounts essentially to an inductive definition of its set of theorems. The axioms are objects which are theorems by the basis rules of the definition and the rules of inference give the inductive rules of the definition. More precisely, the rules of inference, as we have defined them, are constructors on the set of objects, as defined in [13], Section 4.6.

If θ is a theorem of a formal system $\mathcal{F}$, we write $\vdash_{\mathcal{F}} \theta$.

Definition 1.8.3. A formal system $\mathcal{F}'$ is an *extension* of a formal system $\mathcal{F}$ if $\text{Thm}(\mathcal{F}) \subseteq \text{Thm}(\mathcal{F}')$.

The formal systems $\mathcal{F}, \mathcal{F}'$ are *equivalent* if $\text{Thm}(\mathcal{F}) = \text{Thm}(\mathcal{F}')$. ▯

Theorem 1.8.4. *Let $\mathcal{F} = (U, A, I)$ and $\mathcal{F}' = (U', A', I')$ be formal systems such that $U \subseteq U'$ and $I = \{R' \cap (U^n \times U) \mid R' \in I'$ and R' is n-ary$\}$. Then, $\mathcal{F}'$ is an extension of $\mathcal{F}$ if and only if every axiom of $\mathcal{F}$ is a theorem of $\mathcal{F}'$.*

Proof. If $\mathcal{F}'$ is an extension of $\mathcal{F}$, then every axiom of $\mathcal{F}$ is a theorem of $\mathcal{F}$ and, therefore, is a theorem of $\mathcal{F}'$. Conversely, assume that $A \subseteq \text{Thm}(\mathcal{F}')$. We will prove, by induction on $\text{Thm}(\mathcal{F})$, that every theorem of $\mathcal{F}$ is a theorem of $\mathcal{F}'$. The basis is our assumption. Now, suppose that

$$\frac{\theta_0, \ldots, \theta_{n-1}}{\theta}\, R,$$

where $\theta_0, \ldots, \theta_{n-1}$ are theorems of $\mathcal{F}$ and $R \in I$ (which means that $R = R' \cap (U^n \times U)$ for some $R' \in I'$). If we assume, by inductive hypothesis, that $\theta_0, \ldots, \theta_{n-1}$ are theorems of $\mathcal{F}'$, then θ is a theorem of $\mathcal{F}'$ because $((\theta_0, \ldots, \theta_{n-1}), \theta) \in R'$. □

Corollary 1.8.5. *Let $\mathcal{F} = (U, A, I)$ and $\mathcal{F}' = (U, A', I)$ be formal systems such that $A \subseteq \text{Thm}(\mathcal{F}')$. Then, $\mathcal{F}'$ is an extension of $\mathcal{F}$.*

Proof. The corollary follows immediately from Theorem 1.8.4. □

Corollary 1.8.6. *Let $\mathcal{F} = (U, A, I), \mathcal{F}' = (U, A \cup A_1, I)$ be two formal systems, where $A_1 \subseteq \text{Thm}(\mathcal{F})$. Then, $\mathcal{F}$ and $\mathcal{F}'$ are equivalent.*

Proof. The corollary follows immediately from Corollary 1.8.5. □

Definition 1.8.7. Let $\mathcal{F} = (U, A, I)$ be a formal system and let G be a set of objects of $\mathcal{F}$. Then, $\mathcal{F}_G$ is the formal system $\mathcal{F}_G = (U, A \cup G, I)$ obtained from $\mathcal{F}$ by adding G to the set of axioms. ▯

We will write $G \vdash_{\mathcal{F}} \theta$ for $\vdash_{\mathcal{F}_G} \theta$. Observe that if $\theta \in G$, then $G \vdash_{\mathcal{F}} \theta$.

Corollary 1.8.8. *Let $\mathcal{F} = (U, A, I)$ be a formal system and let $G_0, G_1 \subseteq U$ with $G_0 \subseteq \mathrm{Thm}(\mathcal{F}_{G_1})$. Then, $G_0 \vdash_{\mathcal{F}} \theta$ implies $G_1 \vdash_{\mathcal{F}} \theta$.*

Proof. This statement follows immediately from Corollary 1.8.5 and Definition 1.8.7. □

To illustrate the notion of formal system, we will consider some simple examples.

Example 1.8.9. Our first formal system is

$$\mathcal{F}_{ab} = (\{a, b\}^*, \{\lambda\}, \{\,\mathsf{R}_0,\, \mathsf{R}_1,\, \mathsf{R}_2\}),$$

where R_0, R_1, R_2 are given by

$$\frac{u}{aub}\ \mathsf{R}_0 \quad \frac{u}{bua}\ \mathsf{R}_1 \quad \frac{u, v}{uv}\ \mathsf{R}_2$$

for all $u, v \in \{a, b\}^*$.

We can easily show that $abba \in \mathrm{Thm}(\mathcal{F}_{ab})$. Indeed, λ is a theorem, since it is an axiom. Then, by Rules R_0 and R_1, the words ab and ba are also theorems of $\mathcal{F}_{ab}$. Finally, by Rule R_2, we have $abba \in \mathrm{Thm}(\mathcal{F}_{ab})$.

It is easy to show, by induction on theorems, that every theorem of $\mathcal{F}_{ab}$ contains the same number of as and bs. In fact, $\mathrm{Thm}(\mathcal{F}_{ab})$ equals the set S defined inductively in Example 4.3.9 of [13]. Example 4.4.11 of [13] shows that S equals the set of all words over $\{a, b\}$ which contain an equal number of as and bs. ▯

Example 1.8.10. Consider the set of objects

$$U = \{(x_0, y_0, x_1, y_1) \in \mathbf{R}^4 \mid x_0 \leq x_1 \quad \text{and} \quad y_0 \leq y_1\}.$$

An object $\theta \in U$ can be interpreted as a rectangle whose sides are parallel to the axes; if $\theta = (x_0, y_0, x_1, y_1)$, then (x_0, y_0) is the southwest corner and (x_1, y_1) is the northeast corner of θ.

Consider the inference rules

$$\frac{(x_0, y_0, x_1, y_1), (x_1, y_0, x_2, y_1)}{(x_0, y_0, x_2, y_1)} \; \mathsf{R}_h$$

and

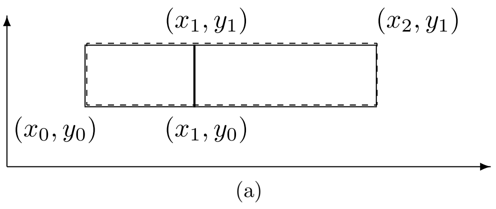
$$\frac{(x_0, y_0, x_1, y_1), (x_0, y_1, x_1, y_2)}{(x_0, y_0, x_1, y_2)} \; \mathsf{R}_v$$

for all $x_i, y_i \in R$, $0 \leq i \leq 2$ such that $x_0 \leq x_1 \leq x_2$, and $y_0 \leq y_1 \leq y_2$. The first rule involves two rectangles that have a common vertical side and it yields a new rectangle by eliminating that common side. The second rule involves two rectangles that have a common horizontal side (see Figure 1.4).

If the set of axioms A consists of all squares of side 1 whose vertices have integer coordinates,

$$A = \{(x_0, y_0, x_0 + 1, y_0 + 1) \mid x_0, y_0 \in \mathbf{Z}\},$$

then it is easy to see that the set of theorems of the formal system $\mathcal{F}_{\text{rec}} = (U, A, \{\mathsf{R}_h, \mathsf{R}_v\})$ consists of all rectangles whose vertices have integer coordinates and whose sides are parallel to the axes. □



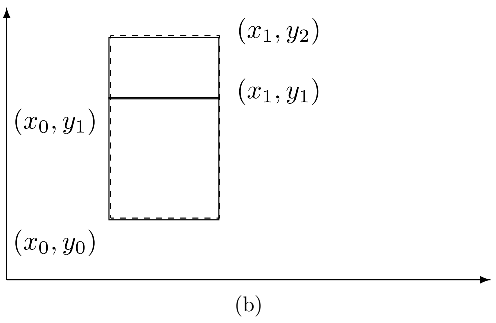


Fig. 1.4. Application of rules R_h (a) and R_v (b).

If $\mathcal{F} = (U, A, I)$ is a formal system, then $\mathcal{F}^{\emptyset}$ denotes the formal system $(U, \emptyset, I)$ obtained from $\mathcal{F}$ by removing the axioms. Note that if $G \subseteq U$, then $G \vdash_{\mathcal{F}} \theta$ if and only if $G \cup A \vdash_{\mathcal{F}^{\emptyset}} \theta$.

Example 1.8.11. Let M be a set. $\mathcal{F}_{\text{seq},M}$ is the formal system having $\text{Seq}(M)$ as the set of objects, $\emptyset$ as the set of axioms, and three rules $\mathsf{R}_{\text{intch}}$, R_{exp}, and R_{cont}, which are given by

$$\frac{(x_0, \ldots, x_i, x_{i+1}, \ldots, x_{n-1})}{(x_0, \ldots, x_{i+1}, x_i, \ldots, x_{n-1})}\ \mathsf{R}_{\text{intch}}$$
$$\frac{(x_0, \ldots, x_{n-1})}{(x_0, \ldots, x_{n-1}, y)}\ \mathsf{R}_{\text{exp}}$$
$$\frac{(x_0, \ldots, x_{n-1}, x_{n-1})}{(x_0, \ldots, x_{n-1})}\ \mathsf{R}_{\text{cont}}$$

and are called the interchange, expansion, and contraction rule, respectively. □

Definition 1.8.12. Let $\mathcal{F}$ be a formal system. A sequence $(\theta_0, \ldots, \theta_{n-1})$ of objects of $\mathcal{F}$ is a *proof in* $\mathcal{F}$ if for each i, $0 \leq i \leq n-1$, one of the following is true:

- θ_i is an axiom of $\mathcal{F}$ or
- there exist $j_0, \ldots, j_{m-1}$ with $j_k < i$ for $0 \leq k \leq m-1$ such that $((\theta_{j_0}, \ldots, \theta_{j_{m-1}}), \theta_i)$ is an instance of a rule of $\mathcal{F}$.

A *proof of the object* θ is proof whose last entry is θ. □

One simple measure of the complexity of a proof $(\theta_0, \ldots, \theta_{n-1})$ is its length n. However, this measure does not reflect the complexity of the objects that occur in the proof. To address this problem, we assume the existence of a function $\texttt{size} : U \longrightarrow \mathbf{N}$ which we think of intuitively as giving the size of objects. This function is extended to proofs by defining $\texttt{size}(\theta_0, \ldots, \theta_{n-1}) = \sum_{i=0}^{n-1} \texttt{size}(\theta_i)$. Whenever objects are or can be identified with strings over an alphabet, we will take $\texttt{size}(\theta)$ to be the length of the word that represents θ.

Example 1.8.13. Let $\theta_0 = \lambda, \theta_1 = ab, \theta_2 = ba$, and $\theta_3 = abba$. The sequence $(\theta_0, \theta_1, \theta_2, \theta_3)$ is a proof in the formal system $\mathcal{F}_{ab}$ introduced in Example 1.8.9 as shown in the following table:

Object	Justification
$\theta_0 = \lambda$	Axiom
$\theta_1 = ab$	Applying R_0 to θ_0
$\theta_2 = ba$	Applying R_1 to θ_0
$\theta_3 = abba$	Applying R_2 to θ_1 and θ_2

The previous proof has length 4 and size 8. □

Example 1.8.14. The sequence $(\theta_0, \ldots, \theta_6)$ whose entries are specified by the following table is a proof in the formal system $\mathcal{F}_{\text{rec}}$ given in Example 1.8.10:

Object	Justification
$\theta_0 = (2, 2, 3, 3)$	Axiom
$\theta_1 = (3, 2, 4, 3)$	Axiom
$\theta_2 = (2, 1, 3, 2)$	Axiom
$\theta_3 = (3, 1, 4, 2)$	Axiom
$\theta_4 = (2, 2, 4, 3)$	Applying R_h to θ_0 and θ_1
$\theta_5 = (2, 1, 4, 2)$	Applying R_h to θ_2 and θ_3
$\theta_6 = (2, 1, 4, 3)$	Applying R_v to θ_4 and θ_5

□

Example 1.8.15. Let $\mathcal{F}_{\text{seq},M}$ be the formal system introduced in Example 1.8.11. We claim that if $q = (x_0, \ldots, x_{n-1}), r = (y_0, \ldots, y_{m-1})$ are two sequences in $\text{Seq}(M)$ such that $\{x_0, \ldots, x_{n-1}\} \subseteq \{y_0, \ldots, y_{m-1}\}$, then $q \vdash_{\mathcal{F}_{\text{seq},M}} r$. We begin by observing that if $\dfrac{q_0}{q_1}$ is an instance of a rule of $\mathcal{F}_{\text{seq},M}$, then $\frac{sq_0}{sq_1}$ is an instance of the same rule for every $s \in \text{Seq}(M)$. Therefore, if $q' \vdash_{\mathcal{F}_{\text{seq},M}} q''$, then $sq' \vdash_{\mathcal{F}_{\text{seq},M}} sq''$.

The justification of the claim is by induction on $n = |q|$. If $n = 0$, we have $\lambda \vdash_{\mathcal{F}_{\text{seq},M}} r$ by applying the expansion rule m times, namely, we have the proof

$$(\lambda, (y_0), (y_0, y_1), \ldots, (y_0, \ldots, y_{m-1})).$$

Suppose that the claim is valid for sequences of length n and let $q = (x_0, \ldots, x_n)$ be a sequence of length $n + 1$ such that

$$\{x_0, \ldots, x_n\} \subseteq \{y_0, \ldots, y_{m-1}\}.$$

Let $q_1 = (x_1, \ldots, x_n)$. By the inductive hypothesis, we have $q_1 \vdash_{\mathcal{F}_{\mathrm{seq},M}} r$, so $q \vdash_{\mathcal{F}_{\mathrm{seq},M}} (x_0)r$, by our initial observation. Note that x_0 occurs in the sequence r, say $x_0 = y_i$. The initial proof

$$((x_0, \ldots, x_n), \ldots, (x_0, y_0, \ldots, y_{m-1}))$$

can be continued as follows:

$$\begin{array}{l}
(x_0, \ldots, x_n) \\
\qquad\qquad \vdots \text{ (initial proof)} \\
(x_0, y_0, \ldots, y_i = x_0, \ldots y_{m-1}) \\
\qquad\qquad \vdots \text{ repeated application of } \mathsf{R}_{\text{intch}} \\
(y_0, \ldots, y_{i-1}, y_{i+1}, \ldots, y_{m-1}, x_0, x_0) \\
\qquad\qquad \mathsf{R}_{\text{cont}} \\
(y_0, \ldots, y_{i-1}, y_{i+1}, \ldots, y_{m-1}, x_0) \\
\qquad\qquad \vdots \text{ repeated application of } \mathsf{R}_{\text{intch}} \\
(y_0, \ldots, y_{i-1}, x_0 = y_i, y_{i+1}, \ldots, y_{m-1})
\end{array}$$

which completes the argument. □

It is clear that any initial segment of a proof in $\mathcal{F}$ is a proof in $\mathcal{F}$.

Definition 1.8.16. An *$\mathcal{F}$-deduction tree*, or a *deduction tree in $\mathcal{F}$*, where $\mathcal{F} = (U, A, I)$ is a formal system, is a U-lot $\mathtt{T}$ such that

(1) $\mathtt{T}$ is finite,
(2) if q is an interior node of $\mathtt{T}$, and its immediate descendants are $q0, \ldots, qn$, then

$$\frac{\mathtt{T}(q0), \ldots, \mathtt{T}(qn)}{\mathtt{T}(q)}$$

is an instance of a rule of $\mathcal{F}$.

The set of deduction trees in the formal system $\mathcal{F}$ will be denoted by $\mathcal{DT}_{\mathcal{F}}$.

If a U-lot $\mathtt{T}$ satisfies only the second condition, then we refer to $\mathtt{T}$ as a *general $\mathcal{F}$-deduction tree*, or as a *general deduction tree in $\mathcal{F}$*. The set of general deduction trees in $\mathcal{F}$ will be denoted by $\mathcal{GDT}_{\mathcal{F}}$.

An *$\mathcal{F}$-proof tree* or a *proof tree in $\mathcal{F}$* is an F-deduction tree $\mathtt{T}$ such that if q is a leaf of $\mathtt{T}$, then $\mathtt{T}(q)$ is an axiom of $\mathcal{F}$. We will denote the set of proof trees in the formal system $\mathcal{F}$ by $\mathcal{PT}_{\mathcal{F}}$.

If $\mathtt{T}$ is an $\mathcal{F}$-deduction (-proof) tree and $\mathtt{T}(\lambda) = \theta$, then we refer to $\mathtt{T}$ as an *$\mathcal{F}$-deduction (-proof) tree of θ.*

The *size* of an $\mathcal{F}$-deduction tree $\mathtt{T}$ is

$$\mathtt{size}(\mathtt{T}) = \sum\{\mathtt{size}(\mathtt{T}(q)) \mid q \in \mathrm{Dom}(\mathtt{T})\}.$$

□

Example 1.8.17. We obtain the $\mathcal{F}_{ab}$-proof tree shown in Figure 1.5 from the proof of *abba* given in Example 1.8.13. Its size is 8.

Similarly, we have the $\mathcal{F}_{\mathrm{rec}}$-proof tree of Figure 1.6 corresponding to the proof considered in Example 1.8.14. □

Lemma 1.8.18. *Let $\mathcal{F} = (U, A, I)$ be a formal system. If $\mathtt{T}_0, \ldots, \mathtt{T}_{n-1}$ are $\mathcal{F}$-deduction (-proof) trees and there are a rule $R \in I$ and an object $\theta \in U$ such that $((\mathtt{T}_0(\lambda), \ldots, \mathtt{T}_{n-1}(\lambda)), \theta) \in R$, then the U-lot $\mathtt{T} = (\mathtt{T}_0, \ldots, \mathtt{T}_{n-1}; \theta)$ is an $\mathcal{F}$-deduction (-proof) tree of θ.*

Proof. Suppose that $\mathtt{T}_0, \ldots, \mathtt{T}_{n-1}$ are $\mathcal{F}$-deduction trees. Theorem 1.7.15 implies that if $r \neq \lambda$ is an internal node of $\mathtt{T}$ whose immediate descendants are $r0, \ldots, rm$, then there exists i, $0 \leq i \leq n-1$

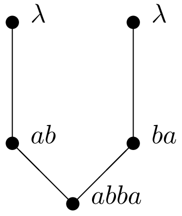


Fig. 1.5. Proof tree for *abba*.

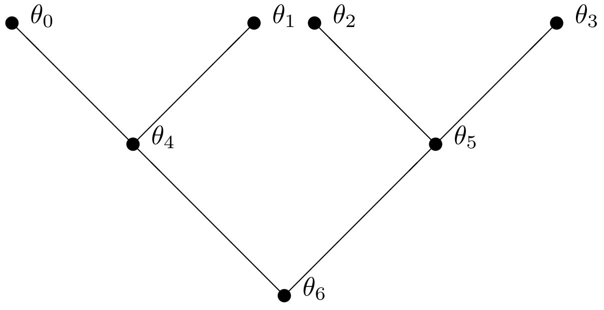


Fig. 1.6. Proof tree for θ_6.

such that $r = iq$ for some internal node q of $\mathtt{T}_i$ whose immediate descendants are $q0, \dots, qm$. Clearly, we have $rj = iqj$ for $0 \leq j \leq m$. Since $\mathtt{T}_i$ is an $\mathcal{F}$-deduction tree, there exists a rule R' such that

$$\frac{\mathtt{T}_i(q0), \dots, \mathtt{T}_i(qm)}{\mathtt{T}_i(q)}$$

is an instance of this rule. By the definition of $\mathtt{T}$, we obtain that

$$\frac{\mathtt{T}(iq0), \dots, \mathtt{T}(iqm)}{\mathtt{T}(iq)} = \frac{\mathtt{T}(r0), \dots, \mathtt{T}(rm)}{\mathtt{T}(r)}$$

is an instance of R'. When $r = \lambda$, the needed condition is clearly satisfied. Thus, $\mathtt{T}$ is an $\mathcal{F}$-deduction tree.

Further, suppose that $\mathtt{T}_0, \dots, \mathtt{T}_{n-1}$ are F-proof trees. The existence of a rule R such that $((\mathtt{T}_0(\lambda), \dots, \mathtt{T}_{n-1}(\lambda)), \theta) \in R$ means that $n > 0$. Therefore, we have

$$\text{LEAVES}(\mathtt{T}) = \bigcup_{0 \leq i \leq n-1} i\, \text{LEAVES}(\mathtt{T}_i),$$

according to Theorem 1.7.15. If r is a leaf of $\mathtt{T}$, then $r = iq$, where q is a leaf of $\mathtt{T}_i$ for some i, $0 \leq i \leq n-1$. Consequently, $\mathtt{T}(r) = \mathtt{T}(iq) = \mathtt{T}_i(q)$. Since $\mathtt{T}_i(q) \in A$, it follows that the labels of the leaves of $\mathtt{T}$ are axioms of $\mathcal{F}$. □

Lemma 1.8.19. *Let $\mathcal{F}$ be a formal system and let $\mathtt{T}$ be an $\mathcal{F}$-deduction (-proof) tree. If r is a node of $\mathtt{T}$, then $\mathtt{T}_{[r]}$ is an $\mathcal{F}$-deduction (-proof) tree.*

Proof. We leave this proof to the reader. □

Another way of defining proof trees is provided by the following theorem.

Theorem 1.8.20. *Let $\mathcal{F} = (U, A, I)$ be a formal system and let $\mathcal{S}$ be the set of U-lots given by the following inductive definition:*

- *If $\mathtt{T}$ is a lot such that $\text{Dom}(\mathtt{T}) = \{\lambda\}$ and $T(\lambda) \in A$, then $\mathtt{T}$ belongs to $\mathcal{S}$.*
- *If $\mathtt{T}_0, \dots, \mathtt{T}_{n-1}$ belong to $\mathcal{S}$ and there is a rule R of $\mathcal{F}$ such that $((\mathtt{T}_0(\lambda), \dots, \mathtt{T}_{n-1}(\lambda)), \theta) \in R$ for some object $\theta \in U$, then the U-lot $\mathtt{T} = (\mathtt{T}_0, \dots, \mathtt{T}_{n-1}; \theta)$ also belongs to $\mathcal{S}$.*

Then, $\mathcal{S} = \mathcal{PT}_{\mathcal{F}}$.

Proof. We show first that every member of $\mathcal{S}$ is an $\mathcal{F}$-proof tree. If $\mathtt{T}$ is a lot such that $\text{Dom}(\mathtt{T}) = \{\lambda\}$ and $\mathtt{T}(\lambda) \in A$, then clearly $\mathtt{T}$ is an $\mathcal{F}$-proof tree.

Suppose now that $\mathtt{T}_0, \ldots, \mathtt{T}_{n-1}$ are $\mathcal{F}$-proof trees and there is a rule R of $\mathcal{F}$ such that $((\mathtt{T}_0(\lambda), \ldots, \mathtt{T}_{n-1}(\lambda)), \theta) \in R$. Then, $(\mathtt{T}_0, \ldots, \mathtt{T}_{n-1}; \theta)$ is an $\mathcal{F}$-proof tree by Lemma 1.8.18.

Conversely, we prove by strong induction on $|\text{Dom}(\mathtt{T})|$ that every $\mathcal{F}$-proof tree $\mathtt{T}$ is a member of $\mathcal{S}$. If $|\text{Dom}(\mathtt{T})| = 1$, then $\text{Dom}(\mathtt{T}) = \{\lambda\}$ and $\mathtt{T}(\lambda) \in A$, so $\mathtt{T} \in \mathcal{S}$.

Assume now that $n \geq 1$ and that every $\mathcal{F}$-proof tree with at most n nodes is in $\mathcal{S}$ and let $\mathtt{T}$ be an $\mathcal{F}$-proof tree with $n+1$ nodes. Then, by Lemma 1.7.19, we have $\mathtt{T} = (\mathtt{T}_{[(0)]}, \ldots, \mathtt{T}_{[(m)]}; \mathtt{T}(\lambda))$ for some $m \in \mathbf{N}$. Since each $\mathtt{T}_{[(i)]}$ has fewer nodes than $\mathtt{T}$ and is a proof tree by Lemma 1.8.19, by inductive hypothesis, $\mathtt{T}_{[(i)]} \in \mathcal{S}$ for $0 \leq i \leq m$. By the definition of $\mathcal{F}$-proof tree, we have $((\mathtt{T}_{[(0)]}(\lambda), \ldots, \mathtt{T}_{[(m)]}(\lambda)), \mathtt{T}(\lambda)) = ((\mathtt{T}(0), \ldots, \mathtt{T}(m-1)), \mathtt{T}(\lambda)) \in R$ for some rule R of $\mathcal{F}$. This allows us to conclude that $\mathtt{T} \in \mathcal{S}$. □

Theorem 1.8.21. *Let $\mathcal{F} = (U, A, I)$ be a formal system and let S be the set of U-lots given by the following inductive definition:*

- *If $\mathtt{T}$ is a one node U-lot, then $\mathtt{T}$ belongs to $\mathcal{S}$.*
- *If $\mathtt{T}_0, \ldots, \mathtt{T}_{n-1}$ belong to $\mathcal{S}$ and there is a rule R of $\mathcal{F}$ such that $((\mathtt{T}_0(\lambda), \ldots, \mathtt{T}_{n-1}(\lambda)), \theta) \in R$ for some object $\theta \in U$, then the U-lot $\mathtt{T} = (\mathtt{T}_0, \ldots, \mathtt{T}_{n-1}; \theta)$ also belongs to $\mathcal{S}$.*

Then, $\mathcal{S}$ is the set of all $\mathcal{F}$-deduction trees.

Proof. The argument is a simpler variant of the one made in the previous theorem. □

Theorem 1.8.22 (Principle of Induction for Deduction (Proof) Trees). *Let $\mathcal{F}$ be a formal system, $\mathcal{F} = (U, A, I)$, and let P be a property of $\mathcal{F}$-deduction (-proof) trees. Suppose that*

- *if $\mathtt{T}$ is a one-node $\mathcal{F}$-deduction (-proof) tree, then $P(\mathtt{T})$ is true,*
- *if $\mathtt{T}_0, \ldots, \mathtt{T}_{n-1}$ are $\mathcal{F}$-deduction (-proof) trees such that $P(\mathtt{T}_i)$ is true for $0 \leq i \leq n-1$ and $((\mathtt{T}_0(\lambda), \ldots, \mathtt{T}_{n-1}(\lambda)), \theta) \in R$ for some rule R of $\mathcal{F}$, then $P((\mathtt{T}_0, \ldots, \mathtt{T}_{n-1}; \theta))$ is true for every $n \in \mathbf{N}$ and $\theta \in U$.*

Then, $P(\mathtt{T})$ is true for every $\mathcal{F}$-deduction (-proof) tree $\mathtt{T}$.

Proof. This follows immediately from Theorems 1.8.21 and 1.8.20. □

Theorem 1.8.23. *Let $\mathcal{F} = (U, A, I)$ be a formal system and let $\theta \in U$. The following statements are equivalent:*

(1) *θ is a theorem of $\mathcal{F}$.*
(2) *There exists a proof of θ in $\mathcal{F}$.*
(3) *There exists an $\mathcal{F}$-proof tree of θ.*

Proof. `(1) implies (2)`. First we show, by induction on the definition of theorems of $\mathcal{F}$ (Definition 1.8.2), that every theorem of $\mathcal{F}$ has a proof in $\mathcal{F}$. The basis step, that every axiom of $\mathcal{F}$ has a proof in $\mathcal{F}$, is obvious since for every axiom θ, the sequence (θ) is a proof of θ. Now suppose that we have the following instance of a rule of $\mathcal{F}$

$$\frac{\theta_0, \ldots, \theta_{n-1}}{\theta}$$

and that, by inductive hypothesis, $\varpi_0, \ldots, \varpi_{n-1}$ are proofs in the formal system $\mathcal{F}$ of $\theta_0, \ldots, \theta_{n-1}$, respectively. Then, a proof of θ in $\mathcal{F}$ can be obtained by concatenating the sequences $\varpi_0, \ldots, \varpi_{n-1}$ and (θ).

`(2) implies (3)`. Let θ be an object from U such that $(\theta_0, \ldots, \theta_{n-1})$ is a proof of θ in $\mathcal{F}$ for some $n \geq 1$. We prove, by strong induction on n, that there exists an $\mathcal{F}$-proof tree of θ.

For $n = 1$, $\theta = \theta_0$, so θ is an axiom. Therefore, the one-node lot $\mathtt{T}$ defined by $\mathtt{T}(\lambda) = \theta$ is an $\mathcal{F}$-proof tree of θ.

Assume that for every object that has a proof in $\mathcal{F}$ of length less than $n + 1$ there exists an $\mathcal{F}$-proof tree of the object and let θ be an object that has a proof $(\theta_0, \ldots, \theta_n)$ of length $n + 1$, where $\theta_n = \theta$. If θ is an axiom, the implication is immediate. Otherwise, the definition of proof implies the existence of $j_0, \ldots, j_{m-1}$, all less than n, such that $((\theta_{j_0}, \ldots, \theta_{j_{m-1}}), \theta_n)$ is an instance of a rule of $\mathcal{F}$ and $(\theta_0, \ldots, \theta_{j_\ell})$ is a proof of length $j_\ell + 1 < n + 1$ of θ_{j_ℓ} for $0 \leq \ell \leq m - 1$. By the inductive hypothesis, we have $\mathcal{F}$-proof trees $\mathtt{T}_{j_0}, \ldots, \mathtt{T}_{j_{m-1}}$ of $\theta_{j_0}, \ldots, \theta_{j_{m-1}}$, respectively. Lemma 1.8.18 implies that $\mathtt{T} = (\mathtt{T}_{j_0}, \ldots, \mathtt{T}_{j_{m-1}}; \theta)$ is an $\mathcal{F}$-proof tree of θ.

`(3) implies (1)`. Suppose that θ is an object for which there is an $\mathcal{F}$-proof tree $\mathtt{T}$. Using the principle of induction for proof trees, we will prove that θ is a theorem of $\mathcal{F}$.

Let $\mathtt{T}$ be a one-node $\mathcal{F}$-proof tree of θ. We have $\mathrm{Dom}(\mathtt{T}) = \{\lambda\}$ and $\mathtt{T}(\lambda) = \theta$. Therefore, θ is an axiom, so $\theta \in \mathrm{Thm}(\mathcal{F})$.

Let now $\mathtt{T}_0, \ldots, \mathtt{T}_{n-1}$ be n $\mathcal{F}$-proof trees such that $\mathtt{T}_i(\lambda) \in \mathrm{Thm}(\mathcal{F})$ for $0 \leq i \leq n-1$ and suppose that $((\mathtt{T}_0(\lambda), \ldots, \mathtt{T}_{n-1}(\lambda)), \theta)$ is an instance of a rule of $\mathcal{F}$. Let $\mathtt{T} = (\mathtt{T}_0, \ldots, \mathtt{T}_{n-1}; \theta)$. Then, $\mathtt{T}(\lambda) = \theta$ is a theorem of $\mathcal{F}$. □

Usually, when a formal system $\mathcal{F}$ is introduced, there is an externally defined set S of objects and the purpose of introducing the formal system is to have $\mathrm{Thm}(\mathcal{F}) = S$. The proof of this equality requires showing two containments. Namely, we need to show that $\mathrm{Thm}(\mathcal{F}) \subseteq S$, which is called the *soundness* of $\mathcal{F}$ with respect to S, and that $S \subseteq \mathrm{Thm}(\mathcal{F})$, which is called the *completeness* of $\mathcal{F}$ with respect to S. When the set S is understood from the context, we will use the terms "soundness of $\mathcal{F}$" and "completeness of $\mathcal{F}$", without mentioning the set S.

To prove the soundness of a formal system, one usually uses induction on the definition of the theorems, that is, one shows that the axioms are in S, and that for every instance of a rule, if the hypotheses are in S, then so is the conclusion. Proofs of completeness are generally more difficult and require techniques which are specific to the externally defined set S.

Definition 1.8.24. Let $\mathcal{F} = (U, A, I)$ be a formal system and let R be a nonempty subset of $U^n \times U$ for some $n > 0$. We refer to R as a *derived rule* of $\mathcal{F}$ if for every $((\theta_0, \ldots, \theta_{n-1}), \theta) \in R$ we have $\{\theta_0, \ldots, \theta_{n-1}\} \vdash_{\mathcal{F}} \theta$. ▯

Example 1.8.25. Example 1.8.15 shows that

$$\frac{(x_0, \ldots, x_{n-1})}{(y_0, \ldots, y_{m-1})}\ \mathsf{R}_{\text{struc}},$$

where $\{x_0, \ldots, x_{n-1}\} \subseteq \{y_0, \ldots, y_{m-1}\}$ is a derived rule of $\mathcal{F}_{\text{seq},M}$. We refer to the rule $\mathsf{R}_{\text{struc}}$ as the *structural rule*. ▯

A characterization of derived rules is given in the following theorem.

Theorem 1.8.26. *Let $\mathcal{F} = (U, A, I)$ be a formal system, $R \neq \emptyset$ be a subset of $U^n \times U$ for some $n > 0$, and $\mathcal{F}'$ be the formal system $(U, A, I \cup \{R\})$. Then, R is a derived rule of $\mathcal{F}$ if and only if $\mathcal{F}_G$ and $\mathcal{F}'_G$ are equivalent formal systems for all sets of objects G.*

Proof. Let R be a derived rule and let $G \subseteq U$. Since the inclusion $\mathrm{Thm}(\mathcal{F}_G) \subseteq \mathrm{Thm}(\mathcal{F}'_G)$ is obvious, we are left with showing the reverse inclusion by induction on the theorems of $\mathcal{F}'_G$. The basis is immediate. For the inductive step, suppose that $((\theta_0, \ldots, \theta_{n-1}), \theta)$ is an instance of a rule of $\mathcal{F}'$ and $\theta_i \in \mathrm{Thm}(\mathcal{F}_G)$ for $0 \leq i \leq n-1$. If the rule belongs to I, then it is immediate that $\theta \in \mathrm{Thm}(\mathcal{F}_G)$. If the rule is R, then, by applying Corollary 1.8.8 with $G_0 = \{\theta_0, \ldots, \theta_{n-1}\}$ and $G_1 = G$, we obtain $\theta \in \mathrm{Thm}(\mathcal{F}_G)$.

Conversely, suppose that $\mathcal{F}_G$ and $\mathcal{F}'_G$ are equivalent formal systems for all $G \subseteq U$ and $((\theta_0, \ldots, \theta_{n-1}), \theta) \in R$. Let $G = \{\theta_0, \ldots, \theta_{n-1}\}$. Then $\theta \in \mathrm{Thm}(\mathcal{F}'_G)$, so, by hypothesis, $\theta \in \mathrm{Thm}(\mathcal{F}_G)$, i.e., $\{\theta_0, \ldots, \theta_{n-1}\} \vdash_{\mathcal{F}} \theta$, as desired. □

Definition 1.8.27. An *effectively specified formal system* is a formal system $\mathcal{F} = (U, A, I)$ which satisfies the following conditions:

(1) U is an effectively enumerable set.
(2) A is a decidable subset of U.
(3) The set $P(\mathcal{F})$ is decidable, where $P(\mathcal{F})$ consists of those sequences of the form $(\theta_0, \ldots, \theta_n)$, where there is a rule of inference $R \in I$ such that $((\theta_{i_0}, \ldots, \theta_{i_{l-1}}), \theta_n) \in R$ for some $\theta_{i_0}, \ldots, \theta_{i_{l-1}}$ that occur in $(\theta_0, \ldots, \theta_{n-1})$.

▯

Lemma 1.8.28. *If $\mathcal{F} = (U, A, I)$ is an effectively specified formal system, then the set of proofs in $\mathcal{F}$ is a decidable subset of Seq(U).*

Proof. A sequence $(\theta_0, \ldots, \theta_{n-1})$ is a proof if for every i, $0 \leq i \leq n-1$, we have either $\theta_i \in A$ or $(\theta_0, \ldots, \theta_i) \in P(\mathcal{F})$. Since A and $P(\mathcal{F})$ are decidable, it follows that the set of proofs is also decidable. □

Theorem 1.8.29. *If $\mathcal{F} = (U, A, I)$ is an effectively specified formal system, then the set of its theorems* $\mathrm{Thm}(\mathcal{F})$ *is semidecidable.*

Proof. Consider the following semideciding algorithm for $\mathrm{Thm}(\mathcal{F})$. Given an object θ, enumerate $\mathrm{Seq}(U)$. Using the decidability of the set of proofs of $\mathcal{F}$, determine for each such sequence whether it is a proof. If so, test whether the last entry of the proof equals θ. If it does, output 1 and stop. □

Corollary 1.8.30. *If $\mathcal{F} = (U, A, I)$ is an effectively specified formal system and G is a decidable subset of U, then* $\mathrm{Thm}(\mathcal{F}_G)$ *is semidecidable.*

Proof. Since $\mathcal{F}$ is an effectively specified formal system, the formal system $\mathcal{F}_G = (U, A \cup G, I)$ is also effectively specified, by the second part of Theorem 1.4.5. The statement follows immediately from Theorem 1.8.29. □

1.9 Linear Orders

Definition 1.9.1. A *partial order* on a set M is a binary relation $\preceq$ that is reflexive, antisymmetric, and transitive. In other words, we have the following:

- $x \preceq x$ for all $x \in M$.
- $x \preceq y$ and $y \preceq x$ imply $x = y$ for all $x, y \in M$.
- $x \preceq y$ and $y \preceq z$ imply $x \preceq z$ for all $x, y, z \in M$.

If in addition, we have $x \preceq y$ or $y \preceq x$ for all $x, y \in M$, we say that the partial order $\preceq$ is a *total order* or a *linear order*. The pair $(M, \preceq)$ is said to be a *linearly ordered set*. ▯

Example 1.9.2. Let S be a signature and V be a set of variables. Define a binary relation $\preceq$ on $\mathrm{TERM}_S(V)$ as consisting of pairs (t, t'), where t is a subterm of t'. It is immediate to verify that $\preceq$ is a partial order on $\mathrm{TERM}_S(V)$. ▯

Example 1.9.3. The relation $\leq$ defined on $\mathbf{R}$ by $x \leq y$ if $y - x$ is a nonnegative number is a linear order on $\mathbf{R}$. ▯

Definition 1.9.4. A *strict partial order* on a set M is a binary relation $\prec$ that is irreflexive and transitive. In other words, we have the following:

- $x \not\prec x$ for all $x \in M$.
- $x \prec y$ and $y \prec z$ imply $x \prec z$ for all $x, y, z \in M$.

If in addition, we have $x \prec y$ or $y \prec x$ for all $x, y \in M$ with $x \neq y$, we say that the strict partial order $\prec$ is a *strict total order* or a *strict linear order*. The pair $(M, \prec)$ is said to be a *strictly linearly ordered set*. ▯

Example 1.9.5. The relation $<$ defined on $\mathbf{R}$ by $x < y$ if $y - x$ is a positive number is a strict linear order on $\mathbf{R}$. ▯

We denote the relation $\{(x, x) \mid x \in M\}$ on a set M by ι_M.

Theorem 1.9.6. *If $\preceq$ is a partial order (linear order) on a set M, then $\preceq - \iota_M$ is a strict partial order (strict linear order) on M. Furthermore, if $\prec$ is a strict partial order (strict linear order) on M, then $\prec \cup \iota_M$ is a partial order (linear order) on M.*

Proof. We leave this argument to the reader. □

Theorem 1.9.7. *If $\preceq$ is a linear order on a set M and S is a finite nonempty subset of M, then S has a greatest and a least element under $\preceq$.*

Proof. The argument is by induction on $n = |S|$.

For the basis step, $n = 1$, the unique element of S is both the greatest and the least element of S.

For the inductive step, suppose that the result is true for $n \geq 1$ and that $|S| = n+1$. Let a be an element of S and let $S' = S - \{a\}$. By the inductive hypothesis, S' has a greatest element b. If $b \preceq a$, then a is the greatest element of S, and if $a \preceq b$, then b is the greatest element of S. A similar argument proves that S has a least element. □

We say that b is a *predecessor* of a relative to the linear order $\preceq$ if $b \preceq a$.

Theorem 1.9.8. *If $\preceq$ is a linear order on a set M and every element of M has only finitely many predecessors under $\preceq$, then every nonempty subset of M has a least element.*

Proof. Let S be a nonempty subset of M, $a \in S$ and let $S' = \{b \in S \mid b \preceq a\}$. By hypothesis, S' is finite and nonempty because $a \in S'$. Thus, by Theorem 1.9.7, S' has a least element b. If $c \in S - S'$, we have $a \preceq c$ because $\preceq$ is a linear order and we do not have $c \preceq a$ and so by transitivity, we have $b \preceq c$. Thus, b is the least element of S under $\preceq$. □

Definition 1.9.9. Let $(M, \preceq)$ and $(M', \preceq')$ be two linearly order sets. A bijection $f : M \longrightarrow M'$ is an *isomorphism of linearly ordered sets* if $x \preceq y$ implies $f(x) \preceq' f(y)$. ▯

Theorem 1.9.10. *If* $(M, \preceq)$ *is a linearly ordered set where* M *is finite, then* $(M, \preceq)$ *is isomorphic to the linearly ordered set* $(\{0, \ldots, n-1\}, \leq)$*, where* $n = |M|$*.*

Proof. The argument is by induction on n.

For the basis step, $n = 0$, we have $M = \emptyset$ and $(M, \preceq)$ is clearly isomorphic to $(\emptyset, \leq)$.

For the inductive step, suppose that $n \geq 0$ and the result holds for n and $|M| = n + 1$. By Theorem 1.9.7, M has a greatest element a. Let $M' = M - \{a\}$ and let $\preceq'$ be the restriction of $\preceq$ to M'. By inductive hypothesis, there is a function $f' : \{0, \ldots, n-1\} \longrightarrow M'$ that is an isomorphism between $(\{0, \ldots, n-1\}, \leq)$ and $(M', \preceq')$. Extend f' to $f : \{0, \ldots, n\} \longrightarrow M$ by defining $f(n) = a$. The function f is clearly an isomorphism between $(\{0, \ldots, n\}, \leq)$ and $(M, \preceq)$. $\square$

Theorem 1.9.11. *Let* $\preceq$ *be a linear order on an infinite set* M *such that every element of* M *has only finitely many predecessors under* $\preceq$*. Then,* $(M, \preceq)$ *is isomorphic to* $(\mathbf{N}, \leq)$*.*

Proof. Let $f : \mathbf{N} \longrightarrow M$ be the function defined by taking $f(n)$ to be the least element under $\preceq$ of $M - \{f(0), \ldots, f(n-1)\}$ which exists by Theorem 1.9.8.

We claim that for $n \geq 0$, we have $\{x \in M \mid x \preceq f(n)\} = \{f(0), \ldots, f(n)\}$. The argument is by induction on n. In the basis step, $f(0)$ is the least element of M under $\preceq$, so the result holds for $n = 0$. For the inductive step, suppose the result holds for $n \geq 0$. Let $x \in M$ be such that $x \preceq f(n+1)$. If $x \preceq f(n)$, then, by induction hypothesis, $x \in \{f(0), \ldots, f(n), f(n+1)\}$. If $x \not\preceq f(n)$, then, by inductive hypothesis, $x \in M - \{f(0), \ldots, f(n)\}$, so by the definition of $f(n+1)$, $f(n+1) \preceq x$. By antisymmetry, $x = f(n+1)$.

Conversely, suppose that $x \in \{f(0), \ldots, f(n+1)\}$. If $x = f(n+1)$, then $x \preceq f(n+1)$ by reflexivity. If $x = f(i)$ for some i, $0 \leq i \leq n$, then by the definition of f, $f(n+1) \notin \{f(0), \ldots, f(n)\}$ so by the inductive hypothesis, $f(n+1) \not\preceq f(n)$. Since $\preceq$ is a linear order, $f(n) \preceq f(n+1)$. By inductive hypothesis, $x \preceq f(n)$, so $x \preceq f(n+1)$ by transitivity.

To prove the injectivity of f, suppose that $n, m \in \mathbf{N}$ and $n < m$. By the definition of f, we have $f(m) \notin \{f(0), \ldots, f(m-1)\}$, so $f(n) \neq f(m)$.

To prove that f is surjective, suppose that $a \in M - \text{Ran}(f)$. Then, by the previous claim, for each $n \in \mathbf{N}$, $a \not\preceq f(n)$. Since $\preceq$ is total, $f(n) \preceq a$ for all $n \geq 0$. Since f is injective, this means that a has infinitely many predecessors under $\preceq$ contradicting the hypothesis of the theorem.

To prove that f is monotonic, suppose that $n, m \in \mathbf{N}$ and $n < m$. By the definition of f, $f(m) \notin \{f(0), \ldots, f(m-1)\}$, so $f(m) \not\preceq f(n)$ by the previous claim. Since $\preceq$ is total, we have $f(n) \preceq f(m)$. □

1.10 Exercises and Supplements

Sequences, Occurrences, and Substitutions

(1) Prove that for $q, q', q'' \in \text{Seq}_n(D)$, we have

$$\Delta(q, q'') \subseteq \Delta(q, q') \cup \Delta(q', q'').$$

(2) Prove that δ is a metric on $\text{Seq}_n(D)$, that is, prove that

$$(i)\ \delta(q, q') = 0 \text{ if and only if } q = q',$$
$$(ii)\ \delta(q, q') = \delta(q', q),$$
$$(iii)\ \delta(q, q'') \leq \delta(q, q') + \delta(q', q'')$$

for every $q, q', q'' \in \text{Seq}_n(D)$.

(3) Show that for every finite sequence s, there are $|s| + 1$ occurrences of the empty sequence λ in s.

(4) Prove that every mapping $f : \{0, \ldots, p-1\} \longrightarrow \{0, \ldots, q-1\}$ is the composition of a surjection $f_0 : \{0, \ldots, p-1\} \longrightarrow \{0, \ldots, r-1\}$ and an injection $f_1 : \{0, \ldots, r-1\} \longrightarrow \{0, \ldots, q-1\}$, where $r \leq \min\{p, q\}$.
Solution. Let $f(\{0, \ldots, p-1\}) = \{l_0, \ldots, l_{r-1}\}$, where $l_0 < \cdots < l_{r-1}$. Define $f_0 : \{0, \ldots, p-1\} \longrightarrow \{0, \ldots, r-1\}$ by $f_0(i) = j$ for j such that $l_j = f(i)$ and $f_1 : \{0, \ldots, r-1\} \longrightarrow \{0, \ldots, q-1\}$ by $f_1(i) = l_i$.

(5) Let p, q be two natural numbers such that $p < q$. Prove that every injection $f : \{0, \ldots, p-1\} \longrightarrow \{0, \ldots, q-1\}$ is the composition of injections $f = g_{q-p-1} \cdots g_0$ where $g_\ell : \{0, \ldots, p+\ell-1\} \longrightarrow \{0, \ldots, p+\ell\}$ for $0 \leq \ell \leq q-p-1$.
Hint. The argument is by induction on $q - p$.

(6) Let q, q', q'' be finite sequences such that q'' is a subsequence of q' and q' is a subsequence of q. Show that q'' is a subsequence of q.

(7) Let q be a finite sequence and let $OS(q)$ be the set of occurrences of subsequences of q in q. Define the binary relation "$\sqsubseteq$" on $OS(q)$ by $(r, i) \sqsubseteq (r', j)$ if the occurrence (r, i) is a part of the occurrence (r', j). Prove that "$\sqsubseteq$" is a partial order on $OS(q)$.

(8) Let q be a finite sequence and let $\zeta = (r, i) \in \mathtt{OCC}_r(q)$. Show that if $q = q_0q_1q_2$ and ζ is a part of the occurrence $(q_1, |q_0|)$ of q_1 in q, then $\mathtt{replace}\,(q, \zeta, r') = q_0\mathtt{replace}\,(q_1, \zeta', r')q_2$, where $\zeta' = (r, i - |q_0|)$, for every sequence r'.

(9) Suppose that (r, i) is an occurrence of a sequence r in a sequence q and that (r'', j) is an occurrence of a sequence r'' in a sequence r'. Prove that

(a) $(r'', i + j)$ is an occurrence of r'' in $\mathtt{replace}\,(q, (r, i), r')$,

(b) we have

$$\begin{aligned}&\mathtt{replace}\,(q, (r, i), \mathtt{replace}\,(r', (r'', j), s))\\ &= \mathtt{replace}\,(\mathtt{replace}\,(q, (r, i), r'), (r'', i + j), s)\end{aligned}$$

for every sequence s.

We extend the definition of $\mathtt{replace}\,(\cdot, \cdot, \cdot)$ to allow the replacement of multiple nonoverlapping occurrences in parallel.

Let q be a sequence and let $(r_0, k_0), \ldots, (r_{m-1}, k_{m-1})$ with $k_0 < \cdots < k_{m-1}$ be nonoverlapping occurrences in q, that is, $k_i + |r_i| \leq k_{i+1}$, for $0 \leq i \leq m - 2$. If $q = q_0r_0q_1r_1 \cdots r_{m-1}q_m$, where $k_i = |q_0r_0 \cdots q_i|$, for $0 \leq i \leq m - 1$, then the sequence

$$\mathtt{replace}\,(q, ((r_0, k_0), \ldots, (r_{m-1}, k_{m-1})), (r'_0, \ldots, r'_{m-1}))$$

is the sequence $q_0r'_0q_1r'_1 \cdots r'_{m-1}q_m$.

(10) Using the previous notation, define the sequences $s_0, s_1, \ldots, s_m$ by $s_0 = q$ and $s_p = \mathtt{replace}\,(s_{p-1}, (r_{m-p}, k_{m-p}), r'_{m-p})$ for $1 \leq p \leq m$.

(a) Justify the definition of s_p by showing that (r_{m-p}, k_{m-p}) is an occurrence in s_{p-1} for $1 \leq p \leq m$,

(b) prove that

$$s_m = \texttt{replace}\,(q, ((r_0, k_0), \ldots, (r_{m-1}, k_{m-1})), (r'_0, \ldots, r'_{m-1})).$$

Hint. Prove by induction on p, $0 \leq p \leq m$, that

$$s_p = q_0 r_0 \cdots q_{m-p-1} r_{m-p-1} q_{m-p} r'_{m-p} q_{m-p+1} r'_{m-p+1} \cdots r'_{m-1} q_m.$$

(11) Let q, r be two finite sequences and let x, y be such that y occurs in neither q nor r. Prove that if $\mathsf{s}^x_{(y)}(q) = \mathsf{s}^x_{(y)}(r)$, then $q = r$.
Solution. By Theorem 1.2.16, we have

$$q = \mathsf{s}^y_{(x)}(\mathsf{s}^x_{(y)}(q)) = \mathsf{s}^y_{(x)}(\mathsf{s}^x_{(y)}(r)) = r.$$

(12) Let s be a substitution.
(a) Show that if $\lambda \notin \text{Ran}(s)$, then, for every finite sequence z, $|\overline{s}(z)| \geq |z|$.
(b) Show that if $\lambda \notin \text{Ran}(s)$ and $z = z_0 \cdots z_{n-1}$ is a finite sequence such that $\overline{s}(z) = z$, then $s(z_i) = z_i$ for all $0 \leq i \leq n-1$ such that $z_i \in \text{Dom}(s)$.
(c) Show that if the hypothesis $\lambda \notin \text{Ran}(s)$ is dropped, then the statement of the previous part becomes false.

(13) Let W be a set with at least two elements. Consider a substitution s with domain W such that if $x, y \in W$ and $x \neq y$, then $s(x)$ is not a suffix of $s(y)$. Prove that $\overline{s}$ is injective on $\text{Seq}(W)$.
Solution. Note that $s(x) \neq \lambda$ for every $x \in W$. We will show, by induction on $n = |q|$, that if $s(q) = s(q')$, then $q = q'$ for any $q, q' \in \text{Seq}(W)$. The basis, $n = 0$, is immediate because $s(x) \neq \lambda$ for all $x \in W$. Suppose now that the statement holds for sequences of length n and let q be a sequence of length $n+1$, $q = rx$, where $x \in W$ such that $\overline{s}(q) = \overline{s}(q')$. Then, $q' \neq \lambda$, so we may write $q' = r'x'$, where $x' \in W$. Thus, we have $\overline{s}(r)\overline{s}(x) = \overline{s}(r')\overline{s}(x')$ and since $s(x) = \overline{s}(x)$ cannot be a suffix of $s(x') = \overline{s}(x')$ and vice versa, if $x \neq x'$, we must have $x = x'$, which implies $\overline{s}(r) = \overline{s}(r')$. By the inductive hypothesis, we have $r = r'$, so $q = q'$.

(14) Explain why in Supplement 13 we assume that W has at least two elements.

(15) Let W be a set. Consider an injective substitution s with domain W such that if $x \in W$, then $|s(x)| = 1$. Prove that $\overline{s}$ is injective on $\mathrm{Seq}(W)$.
Solution. When W has at least two elements, the result follows from Supplement 13. If W has only one element x, and $s(x) = (y)$, then $|\overline{s}(x^n)| = |(y)^n| = n$, which implies that $\overline{s}$ is injective on $\mathrm{Seq}(W)$. Finally, when $W = \emptyset$, $\mathrm{Seq}(W) = \{\lambda\}$ and every function defined on this set is injective.

(16) Formulate and prove a result similar to the one in Supplement 13 in which the word "suffix" is replaced by "prefix".

Collections of Sets

(17) Let U be a set.
  (a) Show that if $\{\mathcal{C}_i \mid i \in I\}$ is a collection of properties of the subsets of U such that each $\mathcal{C}_i$ is of finite character, then $\bigcap\{\mathcal{C}_i \mid i \in I\}$ is a property of the subsets of U that has finite character.
  (b) Let $\mathcal{C}$ be any property of the subsets of U. Prove that there is a smallest property of the subsets of U, $\mathcal{C}'$, such that $\mathcal{C} \subseteq \mathcal{C}'$ and $\mathcal{C}'$ is of finite character.
  (c) Show that property $\mathcal{C}'$ introduced in Part (b) consists of those subsets A of U such that for every finite subset A_0 of A, there is a $B \in \mathcal{C}$ that contains A_0.

(18) Let $\mathcal{R}_r(M), \mathcal{R}_t(M), \mathcal{R}_{rt}(M)$ be the collections of all reflexive, transitive, reflexive, and transitive relations on a set M, respectively. Prove that these collections are closure systems on $M \times M$. Determine the closure operators that correspond to these closure systems.

A *pre-closure operator* on a set M is a mapping $\mu : \mathcal{P}(M) \longrightarrow \mathcal{P}(M)$ that is expansive and monotonic; in other words, a pre-closure operator is a mapping which satisfies the first two properties of the definition of closure operator.

(19) Let $\mu : \mathcal{P}(\mathbf{N}) \longrightarrow \mathcal{P}(\mathbf{N})$ be the mapping given by

$$\mu(L) = L \cup \{2n \mid n \in L\}.$$

Prove that μ is a pre-closure operator on $\mathbf{N}$ but not a closure operator.

(20) Let μ be a pre-closure operator on a set M and let $\mathcal{C}_\mu = \{H \subseteq M \mid \mu(H) = H\}$. Show that $\mathcal{C}_\mu$ is a closure system on M.
Solution. By analyzing the proof of Lemma 1.3.9, it is easily seen that idempotency is not used in the argument.

(21) Let μ be a pre-closure operator on M.
  (a) Show that the unique closure operator κ on M such that $\mathcal{C}_\kappa = \mathcal{C}_\mu$ is given by
$$\kappa(L) = \bigcap\{H \subseteq M \mid L \subseteq H = \mu(H)\}.$$
  (b) Prove that $\kappa(L)$, defined above, is the least set H such that $L \subseteq H$ and $H = \mu(H)$.
  (c) Determine explicitly the closure operator that corresponds to the pre-closure operator defined in Exercise 19.

A *topology* on a set M is a collection $\mathcal{T} \subseteq \mathcal{P}(M)$ of subsets of M, such that

(i) $M \in \mathcal{T}$,
(ii) for every subcollection $\mathcal{C} \subseteq \mathcal{T}$, we have $\bigcup \mathcal{C} \in \mathcal{T}$, and
(iii) if $\mathcal{D}$ is a finite, nonempty subcollection of $\mathcal{T}$, then $\bigcap \mathcal{D} \in \mathcal{T}$.

We will refer to the pair $(M, \mathcal{T})$ as a *topological space*. The members of the topology $\mathcal{T}$ are called the *open sets* of the topological space, while $\mathrm{CL}(M, \mathcal{T}) = \{M - L | L \in \mathcal{T}\}$ is the collection of *closed sets* of $(M, \mathcal{T})$.

(22) (a) Prove that condition (iii) in the definition of a topology can be replaced with the following:
   (iii') if $D, D' \in \mathcal{T}$, then $D \cap D' \in \mathcal{T}$.
  (b) Show, by considering $\mathcal{C} = \emptyset$ in (ii), that for every topological space $(M, \mathcal{T})$, $\emptyset \in \mathcal{T}$.
  (c) Show that in a topological space $(M, \mathcal{T})$, $\mathrm{CL}(M, \mathcal{T})$ is a closure system such that the union of every finite collection of sets in $\mathrm{CL}(M, \mathcal{T})$ belongs to $\mathrm{CL}(M, \mathcal{T})$.

(23) Prove that the following two statements regarding a closure operator κ on a set M are equivalent:
  (a) $\kappa(H \cup L) = \kappa(H) \cup \kappa(L)$ for all $H, L \subseteq M$,
  (b) the union of any two κ-closed subsets of M is a κ-closed subset of M.

(24) Let κ be a closure operator on M. Show that there is a topological space $(M, \mathcal{T})$ such that $\mathcal{C}_\kappa = \mathrm{CL}(M, \mathcal{T})$ if and only if $\kappa(H \cup L) = \kappa(H) \cup \kappa(L)$ for every $H, L \subseteq M$ and $\kappa(\emptyset) = \emptyset$.
Hint. Use Exercise 23.

A collection $\mathcal{C}$ of sets has the *finite intersection property* if for every nonempty finite subcollection $\mathcal{D}$ of $\mathcal{C}$, we have $\bigcap \mathcal{D} \neq \emptyset$.

(25) Let M be a set. For each $a \in M$, let $\mathcal{I}(a)$ be the collection of all finite subsets of M that contain a.
(a) Prove that for every $a \in M$, $\mathcal{I}(a)$ has the finite intersection property.
(b) Prove that $\{\mathcal{I}(a) \mid a \in M\}$ has the finite intersection property.

Solution. Part (a) is immediate. For Part (b), observe that

$$\{a_0, \ldots, a_{n-1}\} \subseteq \mathcal{I}(a_0) \cap \cdots \cap \mathcal{I}(a_{n-1}),$$

for every $n > 0$ and $a_0, \ldots, a_{n-1} \in M$.

A *filter* on a set M is a nonempty family $\mathcal{F}$ of subsets of M such that the intersection of two members of $\mathcal{F}$ belongs to $\mathcal{F}$ and $A \in \mathcal{F}$ and $A \subseteq B \subseteq M$ imply $B \in \mathcal{F}$. If $\mathcal{F}$ is a filter and $\emptyset \notin \mathcal{F}$, then we refer to $\mathcal{F}$ as a *proper filter*.

A proper filter $\mathcal{F}$ in M is an *ultrafilter* if it is strictly contained in no proper filter on M.

The collection of all filters on the set M will be denoted by $\mathtt{FIL}(M)$.

(26) Let $\mathcal{F}$ be a filter on a set M. Prove that for all $A, B \subseteq M$, we have $A \cap B \in \mathcal{F}$ if and only if $A \in \mathcal{F}$ and $B \in \mathcal{F}$.
(27) Prove that for every set M, the collection $\{M\}$ is a filter on M. Moreover, prove that for every filter $\mathcal{F} \in \mathtt{FIL}(M)$, we have $\{M\} \subseteq \mathcal{F}$.
(28) Let M be a set. A subset L of M is *cofinite* if $M - L$ is finite. Prove that the collection of all cofinite subsets of M is a filter in M.
(29) Let $\{\mathcal{F}_i \mid i \in I\}$ be a nonempty family of filters on a set M. Prove that $\bigcap\{\mathcal{F}_i \mid i \in I\}$ is a filter on M. Conclude that for every collection $\mathcal{C}$ of subsets of a set M, there is a least filter $\mathcal{F}_{\mathcal{C}}$ that contains $\mathcal{C}$.

(30) Let $\mathcal{C}$ be a collection of subsets of a set M. Prove that $\mathcal{F}_{\mathcal{C}}$ consists of all subsets X of M such that $X = M$ or $\bigcap \mathcal{D} \subseteq X$ for some nonempty finite subcollection $\mathcal{D}$ of $\mathcal{C}$. Also, show that if $\mathcal{C} \neq \emptyset$, then the alternative $X = M$ can be dropped from the description of $\mathcal{F}_{\mathcal{C}}$.

(31) Let $\mathcal{C}$ be a nonempty collection of subsets of a set M. Prove that there is a proper filter on M that includes $\mathcal{C}$ if and only if $\mathcal{C}$ has the finite intersection property.

(32) Let $\mathcal{F}$ be a filter on a set M, let $A \in \mathcal{P}(M) - \mathcal{F}$, and let $\mathcal{C} = \mathcal{F} \cup \{A\}$. Prove that

$$\mathcal{F}_{\mathcal{C}} = \mathcal{F} \cup \{Y \in \mathcal{P}(M) \mid A \cap X \subseteq Y \text{ for some } X \in \mathcal{F}\}.$$

(33) Prove that a filter $\mathcal{F} \in \texttt{FIL}(M)$ is an ultrafilter on M if and only if for every $A \subseteq M$ exactly one of A and $M - A$ is in $\mathcal{F}$.

(34) Let $\mathcal{U}$ be an ultrafilter on a set M and let $A, B \subseteq M$. Prove that

  (a) $A \cup B \in \mathcal{U}$ if and only if $A \in \mathcal{U}$ or $B \in \mathcal{U}$,
  (b) $A - B \in \mathcal{U}$ if and only if $A \in \mathcal{U}$ and $B \notin \mathcal{U}$.

(35) Let M be a set and let L be a subset of M.

  (a) Prove that $\mathcal{F}_{\{L\}} = \{A \in \mathcal{P}(M) \mid L \subseteq A\}$. We refer to $\mathcal{F}_{\{L\}}$ as the *principal filter generated* by L and we denote it by $\mathcal{F}_L$.
  (b) Prove that $\mathcal{F}_L$ is an ultrafilter on M if and only if $|L| = 1$.

(36) Let M be a set and let $m : \mathcal{P}(M) \longrightarrow \{0, 1\}$ be a function such that $m(M) = 1$ and $A \cap B = \emptyset$ implies $m(A \cup B) = m(A) + m(B)$ for every $A, B \in \mathcal{P}(M)$. We refer to such an m as a *two-valued measure* on M.
Prove that if m is a two-valued measure on M, then the collection $\{A \in \mathcal{P}(M) \mid m(A) = 1\}$ is an ultrafilter on M. Conversely, if $\mathcal{U}$ is an ultrafilter on a set M, prove that there is a two-valued measure m on M such that $\mathcal{U} = \{A \in \mathcal{P}(M) \mid m(A) = 1\}$.

(37) By considering inclusion among filters, we obtain the partially ordered set $(\texttt{FIL}(M), \subseteq)$.

  (a) Prove that if $\{\mathcal{F}_i \mid i \in I\} \subseteq \texttt{FIL}(M)$ is a nonempty collection of filters such that $i, j \in I$ implies $\mathcal{F}_i \subseteq \mathcal{F}_j$ or $\mathcal{F}_j \subseteq \mathcal{F}_i$, then $\bigcup\{\mathcal{F}_i \mid i \in I\}$ is also a filter on M.

(b) Using Zorn's lemma (see Chapter 3 of [13]), show that if $\mathcal{F}$ is a proper filter on a set M, then there exists an ultrafilter $\mathcal{U}$ on M such that $\mathcal{F} \subseteq \mathcal{U}$.

Decidable and Semidecidable Sets

(38) Prove that if U, V are two effectively enumerable sets, then $U \cup V$, $U \cap V$, and $U \times V$ are effectively enumerable.

(39) Let U be an effectively enumerable set and let A, B be decidable subsets of U. Prove that $A \cap B$ is a decidable subset of U.

(40) Let U be an effectively enumerable set and let A, B be semidecidable subsets of U. Prove that $A \cup B$ and $A \cap B$ are semidecidable subsets of U.

(41) Let U, U' be effectively enumerable sets such that $U \subseteq U'$. Prove that

(a) a subset of U is semidecidable as a subset of U if and only if it is semidecidable as a subset of U',

(b) if a subset of U is decidable as a subset of U', then it is decidable as a subset of U.

(42) Let U, U' be two effectively enumerable sets and A be a subset of $U \cap U'$. Prove that A is semidecidable as a subset of U if and only if A is semidecidable as a subset of U'.
Solution. By Exercise 38, $U \cup U'$ is effectively enumerable. A double application of the first part of Exercise 41 yields the desired conclusion.

(43) Let U, U' be effectively enumerable sets such that $U \subseteq U'$ and U is decidable as a subset of U'. Prove that a subset of U is decidable as a subset of U' if and only if it is decidable as a subset of U.

(44) Let U be an effectively enumerable set. Prove that a subset of U is semidecidable if and only if it is effectively enumerable.

(45) Let U, V, W be three effectively enumerable sets and let A, B, C be three subsets of U, V, W, respectively. Show that if $A \leq_m B$ and $B \leq_m C$, then $A \leq_m C$.

(46) Let A, B, C be subsets of an effectively enumerable set U. Prove that

(a) $A \equiv_m A$,

(b) if $A \equiv_m B$, then $B \equiv_m A$,

(c) if $A \equiv_m B$ and $B \equiv_m C$, then $A \equiv_m C$.

(47) Let U, V be effectively enumerable sets and let A, B be subsets of U, V respectively. Prove that if $A \leq_m B$, then $(U - A) \leq_m (V - B)$.

Signatures and Terms

(48) Let S, S', S'' be three signatures. Prove that

(a) $S \preceq S$,
(b) if $S \preceq S'$ and $S' \preceq S$, then $S = S'$, and
(c) if $S'' \preceq S'$ and $S' \preceq S$, then $S'' \preceq S$.

(49) Let S be a finite signature.

(a) Prove that if V is a nonempty set of variables, then there is a term $t \in \mathrm{TERM}_S(V)$ that contains all the function symbols of S if and only if both of the following hold:

i. S is nonempty and
ii. if S contains more than one constant symbol, then S contains a function symbol of arity greater than 1.

(b) Prove that there is a term $t \in \mathrm{GTERM}_S$ that contains all function symbols of S if and only if both of the following conditions hold:

i. S contains at least one constant symbol and
ii. if S contains more than one constant symbol, then S contains a function symbol of arity greater than 1.

(50) Let $S = (F, \nu)$ be a signature, V be a set of variables, and let $\zeta = (s, i)$ be an occurrence of a function symbol or a variable in a term $t \in \mathrm{TERM}_S(V)$. Then, there is an occurrence of an (S, V)-term u in t starting at the same position, $\xi = (u, i)$.
Solution. The argument is by induction on the term t. If t is a variable or constant symbol, then $s = t$ and $i = 0$ and we can take $u = t$. Suppose that the statement holds for $t_0, \ldots, t_{n-1}$ and that $t = f(t_0, \ldots, t_{n-1})$. If $i = 0$, we can take $u = t$. Otherwise, there is a term t_j and an occurrence $\zeta' = (s, i - k)$ in t_j, where $k = j + 2 + \sum_{l=0}^{j-1} |t_l|$, so by inductive hypothesis, there is an occurrence $\xi' = (u, i - k)$ in t_j, so an occurrence $\xi = (u, i)$ in t.

(51) Let $S = (F, \nu)$ be a signature, V be a set of variables, and let M be a subset of $\mathrm{TERM}_S(V)$ which contains all the constant symbols of F. Suppose that for every function symbol f of

positive arity n in F and n-tuple $(t_0, \dots, t_{n-1})$ of terms of $\text{TERM}_S(V)$, we have

$$\begin{aligned} &f(t_0, \dots, t_{n-1}) \in M \text{ if and only if} \\ &t_i \in M \text{ for all } i,\ 0 \leq i \leq n-1. \end{aligned} \tag{1.1}$$

(a) Prove that M equals $\text{TERM}_S(V \cap M)$.
(b) Derive Theorem 1.5.9 from the first part of this exercise.

Solution. (a) We prove by structural induction on the terms t of $\text{TERM}_S(V)$ that if t is in M, then $t \in \text{TERM}_S(V \cap M)$. If t is a variable or a constant symbol and is in M, then $t \in \text{TERM}_S(V \cap M)$ by definition. Assume now that $t = f(t_0, \dots, t_{n-1})$ is a term and, by inductive hypothesis, if $t_i \in M$, then $t_i \in \text{TERM}_S(V \cap M)$ for every i, $0 \leq i \leq n-1$. If $t \in M$, then by 1.1, we have $t_i \in M$ for $0 \leq i \leq n-1$ and therefore, by the inductive hypothesis, $t_0, \dots, t_{n-1}$ belong to $\text{TERM}_S(V \cap M)$. Then, by the definition of $\text{TERM}_S(V \cap M)$, we have $t \in \text{TERM}_S(V \cap M)$.

To prove the reverse inclusion, we show by induction on the definition of $\text{TERM}_S(V \cap M)$ that every member of $\text{TERM}_S(V \cap M)$ is in M. If u is put in $\text{TERM}_S(V \cap M)$ by one of the first two rules, then $u \in M$. Assume that u is put into $\text{TERM}_S(V \cap M)$ by the third rule, that is, $u = f(u_0, \dots, u_{n-1})$, where $(u_0, \dots, u_{n-1})$ is an n-tuple of terms in $\text{TERM}_S(V \cap M)$ that belong to M (by the inductive hypothesis). By (1.1), $u = f(u_0, \dots, u_{n-1}) \in M$.

(b) Let M be the set of terms in $\text{TERM}_S(V)$ that do not contain variables. The result follows immediately from Part (a) since $V \cap M = \emptyset$.

(52) Let $S = (F, \nu)$ be a signature and let V be a set of variables. Show that if u and v are two sequences of symbols in $F \cup V \cup P$, then $K(uv) = K(u) + K(v)$, where K is the function introduced in Definition 1.5.10, extended to sequences as in the proof of Lemma 1.5.11.

(53) Let $S = (F, \nu)$ be a signature and let V be a set of variables.

(a) Use Exercise 52 and the argument of Lemma 1.5.11 to prove that if w is a proper suffix of a term from $\text{TERM}_S(V)$, then $K(w) \geq 1$. Conclude that no suffix of a term can be a proper prefix of a term.

(b) Prove that no proper suffix of a term can be a term.

Hint for Part (b): If w is a proper suffix of $f(t_0, \ldots, t_{n-1})$ other than) and $(t_0, \ldots, t_{n-1})$, show that $K(w) \geq 2$; deal separately with) and $(t_0, \ldots, t_{n-1})$.

(54) Let $S = (F, \nu)$ be a signature and let V be a set of variables.

(a) Prove that Part (a) of Exercise 53 also holds for terms in Polish notation, that is, for terms in $\text{POLTERM}_S(V)$.

(b) Let s be a sequence of symbols from $F \cup V$. Prove that if $K(w) > 0$ for every nonempty suffix w of s, then s is a concatenation of $K(s)$ terms from $\text{POLTERM}_S(V)$. Here K is the function introduced in Definition 1.5.10.

(c) Show that a proper suffix of a term in Polish notation can be a term in Polish notation. In fact, show that every nonempty suffix of a term in Polish notation is a concatenation of terms in Polish notation.

Hint for Part (b): Use induction on the length of s.

(55) Formulate and prove an analog of the Occurrence Theorem for terms in Polish notation.

(56) Let $S = (F, \nu)$ be a signature and let V be a set of variables. The set of terms in reverse Polish notation of the signature S over the set of variables V, $\text{RPOLTERM}_S(V)$, is the set of finite sequences of symbols from F and V defined inductively as follows:

1. Every variable in V is in $\text{RPOLTERM}_S(V)$.
2. For each function symbol f of S (say of arity n) and each n-tuple $(t_0, \ldots, t_{n-1})$ of terms of $\text{RPOLTERM}_S(V)$, $t_0 \ldots t_{n-1} f$ is in $\text{RPOLTERM}_S(V)$.

(a) Consider the signature S introduced in Example 1.5.3 and the set of variables $V = \{x_0, x_1\}$. Show that the sequences

$$afaffg, afx_0h, affx_1fh$$

are terms in reverse Polish notation of S over V.

(b) Prove that the definition of the set $\text{RPOLTERM}_S(V)$ satisfies the unique readability condition.

(c) Consider the mappings $\Psi' : \text{RPOLTERM}_S(V) \longrightarrow \text{TERM}_S(V)$ and $\Phi' : \text{TERM}_S(V) \longrightarrow \text{RPOLTERM}_S(V)$

defined recursively by

$$\Psi'(x) = x,$$
$$\Psi'(a) = a,$$
$$\Psi'(u_0 \dots u_{m-1} g) = g(\Psi'(u_0), \dots, \Psi'(u_{m-1}))$$

and

$$\Phi'(x) = x,$$
$$\Phi'(a) = a,$$
$$\Phi'(f(t_0, \dots, t_{n-1})) = \Phi'(t_0) \dots \Phi'(t_{n-1}) f$$

for every variable x, constant symbol $a \in F$, $g \in F$ (with $m = \nu(g) > 0$), $f \in F$ (with $n = \nu(f) > 0$), $u_0, \dots, u_{m-1} \in \mathrm{RPOLTERM}_S(V)$, and $t_0, \dots, t_{n-1} \in \mathrm{TERM}_S(V)$. Then, prove that Φ' and Ψ' are bijections which are inverses of each other.

(d) Define *explicitly* two bijections

$$\Psi'' : \mathrm{RPOLTERM}_S(V) \longrightarrow \mathrm{POLTERM}_S(V)$$

and

$$\Phi'' : \mathrm{POLTERM}_S(V) \longrightarrow \mathrm{RPOLTERM}_S(V)$$

which are inverses of each other.

Hint for Part (b): Show that, for a given signature S and set of variables V, no proper suffix of a term in reverse Polish notation can be a term in reverse Polish notation by proving that for every proper suffix u of such a term we have $K(u) < 1$. (Here K is the function introduced in Definition 1.5.10.)

(57) Let S be a signature and V be a set of variables. Show that for every term t in $\mathrm{TERM}_S(V)$, $|\mathtt{OCC}_((t)| = |\mathtt{OCC}_)(t)|$.

(58) Let $S = (F, \nu)$ be a signature and V be a set of variables. For $t \in \mathrm{TERM}_S(V)$ and an occurrence of a symbol $(s, i) \in \mathtt{OCC}_s(t)$, define the *level* of the occurrence (s, i) to be $|\mathtt{OCC}_((u)| - |\mathtt{OCC}_)(u)|$, where u is the prefix of t of length i. In other words, the level of the occurrence (s, i) is given by the difference between the number of left parentheses and the number of right parentheses that precede the occurrence in t.

(a) Prove that the level of every occurrence of a comma in t is at least equal to 1.

(b) Prove that for every term $t \in \text{TERM}_S(V)$ that begins with a function symbol, the number of commas in t at level 1 is $\max\{\nu(f) - 1, 0\}$, where f is the first symbol of t.

(c) Give another proof of unique readability of Definition 1.5.2 using Part (b).

(59) (a) Give an example of two signatures $S = (F, \nu)$ and $S' = (F, \nu')$, a set of variables V, and a term t in both $\text{POLTERM}_S(V)$ and $\text{POLTERM}_{S'}(V)$ such that t contains a function symbol f for which $\nu(f) \neq \nu'(f)$.

(b) Show that no example as in Part (a) exists if we replace the sets $\text{POLTERM}_S(V)$ and $\text{POLTERM}_{S'}(V)$ by $\text{TERM}_S(V)$ and $\text{TERM}_{S'}(V)$, respectively, i.e., prove that if $t \in \text{TERM}_S(V) \cap \text{TERM}_{S'}(V)$ and f occurs in t, then $\nu(f) = \nu'(f)$.

Hint for Part (b): Use Part (b) of Exercise 58 and induction.

(60) Let $S = (F, \nu)$ and $t \in \text{POLTERM}_S(V)$. Let $\Phi_S(t)$ be the sum of the arities of the function symbols (counting multiplicities of these symbols) that occur in t.

(a) Prove that $\Phi_S(t) = |t| - 1$.

(b) Conclude that if $t \in \text{POLTERM}_S(V) \cap \text{POLTERM}_{S'}(V')$, then $\Phi_S(t) = \Phi_{S'}(t)$.

(61) Recall that we defined $\mathtt{r}_{S,T,\min}$ as the reduct of S that consists of all function symbols that occur in terms of T.

Let S_0, S_1 be two signatures, V_0, V_1 be two sets of variables, and T be a set of terms in $\text{TERM}_{S_0}(V_0) \cap \text{TERM}_{S_1}(V_1)$. Show that if $\mathtt{r}_{S_0,T,\min} = \mathtt{r}_{S_1,T,\min}$, then T is an (S_0, V_0)-unifiable set of terms if and only if it is an (S_1, V_1)-unifiable set of terms. Explain why the condition $\mathtt{r}_{S_0,T,\min} = \mathtt{r}_{S_1,T,\min}$ is necessary.

Hint. The statement follows from a double application of Theorem 1.6.24 and from the fact that the equality $\mathtt{r}_{S_0,T,\min} = \mathtt{r}_{S_1,T,\min}$ means that $\mathtt{V}(T)$ is the same regardless of whether it is computed relative to S_0 or to S_1.

(62) Let S be a signature and let V be a set of variables. Is $\iota_{(S,V)}$ the same as the identity map 1_V?

Solution. No, because, strictly speaking, the range of $\iota_{(S,V)}$ is not V but rather V^1, the set of all sequences of length 1 over V;

of course, in practice, we identify V and V^1 and, under this identification, $\iota_{(S,V)}$ and 1_V are the same.

(63) Let $s : V \longrightarrow \mathrm{TERM}_S(V)$ be a substitution.

(a) Let $t \in \mathrm{TERM}_S(V)$ be a term. Prove that $\bar{s}(t) = t$ if and only if $s(x) = x$ for every variable x that occurs in t.

(b) Let z be an (S, V)-substitution such that $z(x) = x$ for every x in $\mathtt{V}(\mathrm{Ran}(s))$. Prove that $z * s = s$.

(64) Show that if s is an (S, V)-substitution, then s is idempotent, that is, $s * s = s$, if and only if for all variables x and x' if x' occurs in $s(x)$, then $s(x') = x'$.
Solution. If s is idempotent, we have $\overline{s}(s(x)) = s(x)$ for all variables x and so, by Exercise 63, we have $s(x') = x'$ for every variable x' that occurs in $s(x)$.
Conversely, suppose that the condition is satisfied. Then, for every variable x, we have $\overline{s}(s(x)) = s(x)$ because all variables occurring in $s(x)$ are fixed by s and other symbols (function symbols and punctuation) are not in the domain of s.

(65) An (S, V)-substitution s is a *variable-pure substitution* if $s(x) \in V$ for every variable $x \in V$.
Show that an (S, V)-substitution s is invertible, that is $s * s' = s' * s = \iota$ for some (S, V)-substitution s', if and only if s is variable-pure and s is a bijection from V to itself.
Solution. Let s be an invertible substitution, and let s' be a substitution such that $s * s' = s' * s = \iota$. Then, for every variable x, $\overline{s}(s'(x)) = \overline{s'}(s(x)) = \iota(x) = x$. Since $\overline{s'}(s(x)) = x$, by Exercise 12, we have $1 = |x| = |\overline{s'}(s(x))| \geq |s(x)|$, so $s(x)$ is either a variable or a constant symbol. But, if $s(x)$ is a constant symbol a, we have $\overline{s'}(s(x)) = \overline{s'}(a) = a \neq x$. Thus, s is variable-pure and, therefore, for all variables x, $x = \overline{s'}(s(x)) = s'(s(x))$. Similarly, one can show that s' is variable-pure and for all variables x, $x = s(s'(x))$. It follows that $s : V \longrightarrow V$ is a bijection (with s' as its inverse).
Conversely, suppose that $s : V \longrightarrow V$ is a bijection and let s' be its inverse bijection. Then, for every $x \in \mathrm{VAR}$, we have $s * s'(x) = \overline{s}(s'(x)) = s(s'(x)) = x = \iota(x)$, so $s * s' = \iota$. Similarly, $s' * s = \iota$, so s is an invertible substitution.

(66) Let A be a finite set, B be a superset of A, and $f : A \longrightarrow B$ be an injection. Prove that there is a bijection $g : B \longrightarrow B$ such that $g \restriction A = f$ and $\{b \in B \mid b \neq g(b)\}$ is finite.

Solution. The function f is a bijection between A and $f(A)$, which means that $|A| = |f(A)|$. Since A is finite and is the disjoint union $A = (A - f(A)) \cup (A \cap f(A))$, we have

$$\begin{aligned} |A - f(A)| &= |A| - |A \cap f(A)| = |f(A)| - |A \cap f(A)| \\ &= |f(A) - A|. \end{aligned}$$

It follows that there is a bijection $h : f(A) - A \longrightarrow A - f(A)$. Now, g can be defined as

$$g(b) = \begin{cases} f(b) & \text{if } b \in A \\ h(b) & \text{if } b \in f(A) - A \\ b & \text{if } b \notin (A \cup f(A)). \end{cases}$$

It is clear that g extends f and $\{b \in B \mid b \neq g(b)\}$ is finite. Suppose that $g(b_0) = g(b_1)$. Note that b_0, b_1 cannot be co-located in any of the sets A, $f(A) - A$, or $B - (A \cup f(A))$ because f, h, and the identity map are injections. Since $b \in A$ implies $g(b) = f(b) \in f(A)$, $b \in f(A) - A$ implies $g(b) = h(b) \in A - f(A)$, and $b \in B - (A \cup f(A))$ implies $g(b) = b \in B - (A \cup f(A))$, it follows that g is injective. Observe that if $c \in f(A)$, then it is the image of some $b \in A$ under g, if $c \in A - f(A)$, then it is the image of an element in $f(A) - A$, and if $c \notin A \cup f(A)$, then $c = g(c)$. Thus, g is surjective and therefore it is a bijection.

(67) Prove that the statement in Supplement 66 can fail if A is not finite.

(68) Define the binary relation $\equiv$ on the set $\text{SUBST}(S, V)$ of all (S, V)-substitutions by $s_0 \equiv s_1$ if there are two (S, V)-substitutions s' and s'' such that $s_1 = s' * s_0$ and $s_0 = s'' * s_1$.

(a) Show that $\equiv$ is an equivalence on the set $\text{SUBST}(S, V)$.

(b) Show that for every term $t \in \text{TERM}_S(V)$ we have $|\overline{s_0}(t)| = |\overline{s_1}(t)|$, if $s_0 \equiv s_1$.

(c) Suppose that $s_0 \equiv s_1$ and that $s_1 = s' * s_0$ and $s_0 = s'' * s_1$ for some substitutions s' and s''. Prove that for all variables x, x', x'', if x' occurs in $s_0(x)$, then $s'(x') \in V$ and if x'' occurs in $s_1(x)$, then $s''(x'') \in V$. Furthermore, s' is injective on $\mathtt{V}(\text{Ran}(s_0))$ and s'' is injective on $\mathtt{V}(\text{Ran}(s_1))$.

(d) Show that if $s_1 \equiv s_0$, then there exist two variable-pure substitutions s_{10} and s_{01} such that $s_1 = s_{10} * s_0$ and

$s_0 = s_{01} * s_1$; furthermore, s_{10} is injective on $\mathtt{V}(\mathrm{Ran}(s_0))$ and s_{01} is injective on $\mathtt{V}(\mathrm{Ran}(s_1))$.

(e) Prove that $s_0 \equiv s_1$ if and only if $s_1 = s_{10} * s_0$ for some variable-pure substitution s_{10} that is injective on $\mathtt{V}(\mathrm{Ran}(s_0))$.

(f) Prove that if s_0 is a finite substitution, then s_1 is a finite substitution with $s_1 \equiv s_0$ if and only if there is a variable-pure, finite bijection z from V to V with $s_1 = z * s_0$.

Solution. We leave the easy verification of the Parts (a) and (b) to the reader. The hypothesis of Part (c) implies that for every variable x, $s_1(x) = \overline{s'}(s_0(x))$ and $s_0(x) = \overline{s''}(s_1(x))$, so $s_0(x) = \overline{s''}(\overline{s'}(s_0(x))) = \overline{s'' * s'}(s_0(x))$. By Part (a) of Exercise 63, we have $\overline{s''}(s'(x')) = \overline{s'' * s'}(x') = x'$ for every variable x' in $s_0(x)$. By an argument similar to the one used in Supplement 65, this implies that $s'(x')$ is a variable for all such x'. Similarly, if x'' occurs in $s_1(x)$, then $s''(x'')$ is a variable. Suppose that $s'(x_0) = s'(x_1)$, where $x_0, x_1 \in \mathtt{V}(\mathrm{Ran}(s_0))$. Then, $x_0 = \overline{s'' * s'}(x_0) = \overline{s''}(s'(x_0)) = \overline{s''}(s'(x_1)) = \overline{s'' * s'}(x_1) = x_1$, which shows that s' is injective on $\mathtt{V}(\mathrm{Ran}(s_0))$. In an analogous way, one shows that s'' is injective on $\mathtt{V}(\mathrm{Ran}(s_1))$.

To prove Part (d), suppose that s_1 and s_0 are substitutions such that $s_1 = s' * s_0$ and $s_0 = s'' * s_1$ for some substitutions s' and s''. Consider the substitution s_{10} given by

$$s_{10}(x') = \begin{cases} s'(x') & \text{if } x' \text{ occurs in a term in } \mathrm{Ran}(s_0), \\ x' & \text{otherwise.} \end{cases}$$

From Part (c), it follows that s_{10} is a variable-pure substitution and is injective on $\mathtt{V}(\mathrm{Ran}(s_0))$. For all variables x, since $\overline{s_{10}}(s_0(x)) = \overline{s'}(s_0(x))$, it follows that $s_1(x) = \overline{s_{10}}(s_0(x)) = s_{10} * s_0(x)$, so $s_1 = s_{10} * s_0$. In a similar way, we define s_{01}.

For Part (e), note that $s_0 \equiv s_1$ implies the existence of the variable-pure substitution s_{10} that is injective on $\mathtt{V}(\mathrm{Ran}(s_0))$ such that $s_1 = s_{10} * s_0$, by Part (d). Conversely, suppose s_{10} is a variable-pure substitution that is injective on $\mathtt{V}(\mathrm{Ran}(s_0))$ such that $s_1 = s_{10} * s_0$. Define s_{01} as

$$s_{01}(x) = \begin{cases} y & \text{if } y \in \mathtt{V}(\mathrm{Ran}(s_0)) \text{ and } s_{10}(y) = x \\ x & \text{otherwise.} \end{cases}$$

Note that the substitution s_{10} is well defined because s_{10} is injective on the set $\mathtt{V}(\mathrm{Ran}(s_0))$. By this definition, if $y \in \mathtt{V}(\mathrm{Ran}(s_0))$, we have

$$s_{01} * s_{10}(y) = \bar{s_{01}}(s_{10}(y)) = y.$$

By Exercise 63, we have $s_0 = (s_{01} * s_{10}) * s_0 = s_{01} * (s_{10} * s_0) = s_{01} * s_1$. Therefore, $s_0 \equiv s_1$.
To prove Part (f), suppose $s_1 = z * s_0$ where z is a finite, variable-pure substitution that is a bijection considered as a function from V to V. By Theorem 1.2.21, s_1 is finite and by Part (e), $s_1 \equiv s_0$.
Conversely, let s_1 be a finite substitution such that $s_1 \equiv s_0$. By Part (e), there is a variable-pure substitution z such that z is injective on $\mathtt{V}(\mathrm{Ran}(s_0))$ and $s_1 = z * s_0$. We shall prove that there is a finite bijection $z' : V \longrightarrow V$ which agrees with z on $\mathtt{V}(\mathrm{Ran}(s_0))$ (and therefore $s_1 = z' * s_0$). Note that z is a finite substitution, that is, $\mathsf{carr}(z)$ is a finite set, because $\mathsf{carr}(z) \subseteq \mathsf{carr}(s_0) \cup \mathsf{carr}(s_1)$ and both $\mathsf{carr}(s_0)$ and $\mathsf{carr}(s_1)$ are finite sets. Let $F_z = V - \mathsf{carr}(z)$ be the cofinite set of variables that are invariant under z. The finiteness of s_0 also implies that the set $\mathtt{V}(\mathrm{Ran}(s_0))$ is cofinite and therefore, the set $F_z \cap \mathtt{V}(\mathrm{Ran}(s_0))$ is also cofinite. Define the finite set $B = V - (F_z \cap \mathtt{V}(\mathrm{Ran}(s_0)))$ and let $A = \mathsf{carr}(z) \cap \mathtt{V}(\mathrm{Ran}(s_0))$. Note that $A \subseteq B$. Also, $z(A) \subseteq B$. Indeed, suppose that $v \in z(A)$ but $v \notin B$, that is, $v \in F_z \cap \mathtt{V}(\mathrm{Ran}(s_0))$, which implies $z(v) = v$. On the other hand, $v = z(x)$ for some $x \in A \subseteq B$. Note that $x \neq v$ because $x \in B$ and $v \notin B$. Since both x and v are in $\mathtt{V}(\mathrm{Ran}(s_0))$, this contradicts the injectivity of z on $\mathtt{V}(\mathrm{Ran}(s_0))$. Also, $z\restriction A$ is an injection since $A \subseteq \mathtt{V}(\mathrm{Ran}(s_0))$. By Supplement 66, there is a bijection $z'' : B \longrightarrow B$ that extends $z\restriction A$. The desired bijection $z' : V \longrightarrow V$ is defined by

$$z'(x) = \begin{cases} z(x) & \text{if } x \in V - B \\ z''(x) & \text{otherwise} \end{cases} = \begin{cases} x & \text{if } x \in V - B \\ z''(x) & \text{otherwise.} \end{cases}$$

Let s be an (S, V)-substitution, where S is a signature and V is a set of variables. Denote by $\mathsf{SIG}(s, S)$ the reduct of S to the set of function symbols that appear in some $s(x)$ for $x \in V$.

(69) Let s', z, s be three (S, V)-substitutions such that $s' = z * s$. Prove that $\mathsf{SIG}(s, S)$ is a reduct of $\mathsf{SIG}(s', S)$. Conclude that if $s \equiv s'$, then $\mathsf{SIG}(s, S) = \mathsf{SIG}(s', S)$.

Term Unification

(70) Let T be a set of (S, V)-terms with no more than one element.
   (a) Show that every (S, V)-substitution is a unifier of T.
   (b) Show that the identity substitution ι is a most general (S, V)-unifier for T.
   (c) Furthermore, show that an (S, V)-substitution is a most general (S, V)-unifier if and only if it is an injective mapping from V to V.

(71) Show that there exists a unifiable set of terms T and a substitution s such that $s(T)$ is not unifiable.

(72) Let T be a unifiable finite set of terms and let s be a most general unifier for T. Show that if z is a unifier for T and $t \in T$, then $|s(t)| \leq |z(t)|$. Conclude that if s' is another most general unifier for T, then $|s(t)| = |s'(t)|$ for all $t \in T$.

(73) Consider the terms

$$t = f(x_1, x_2, \ldots, x_n),$$
$$t' = f(g(x_0, x_0), g(x_1, x_1), \ldots, g(x_{n-1}, x_{n-1})),$$

where f is an n-ary function symbol and g is a binary function symbol. Prove that the set $\{t, t'\}$ is unifiable and that a term u resulting from the unification of t and t' through the application of a most general unifier of the set $\{t, t'\}$ is of length $10 \cdot 2^n - 3n - 8$ for $n \geq 1$.

(74) Let S be a signature and V be a set of variables. Prove that if s, z, s' are (S, V)-substitutions such that $s' = z * s$, then $\mathsf{SIG}(s, S) \preceq \mathsf{SIG}(s', S)$.

(75) Let S, S' be two signatures with $S' \preceq S$, V be a set of variables, and T be a finite set of (S, V)-terms. Prove that for any two most general (S, S', V)-unifiers s_0 and s_1 of T, we have $\mathsf{SIG}(s_0, S) = \mathsf{SIG}(s_1, S)$.

Let T be a finite set of (S, V)-terms and S' be a reduct of S. Suppose that T is (S, S', V)-unifiable. Denote by $\mathsf{MGUSIG}(T, S, S')$ the common signatures of all most general (S, S', V)-unifiers (which exists by Exercise 75). Also, we denote $\mathsf{MGUSIG}(T, S, S)$ by $\mathsf{MGUSIG}(T, S)$.

(76) Let T be a finite set of (S,V)-terms and S' be a reduct of S. Suppose that T is (S,S',V)-unifiable. Prove that $\mathsf{MGUSIG}(T,S,S')$ is a reduct of $\mathtt{r}_{S,T,\min} \sqcap S'$.
Hint. Apply Theorem 1.6.26.

(77) Let S, S_0, S_1 be three signatures with $S_0, S_1 \preceq S$, V be a set of variables, and T be a finite set of $(S_0 \sqcap S_1, V)$-terms. Prove that $\mathbf{MGUNIF}^{S_0,V}(T) = \mathbf{MGUNIF}^{S_1,V}(T)$ and therefore T is (S_0,V)-unifiable if and only if it is (S_1,V)-unifiable.
Solution. Let $z \in \mathbf{MGUNIF}^{S_0,V}(T)$. Since z is an $\mathsf{MGUSIG}(T,S_0)$-substitution, it follows that it is an $(\mathtt{r}_{S_0,T,\min}, V)$-substitution, by Exercise 76, and therefore is an $(\mathtt{r}_{S_1,T,\min}, V)$-substitution, because $\mathtt{r}_{S_0,T,\min} = \mathtt{r}_{S_1,T,\min}$. Thus, z is an (S_1,V)-substitution, so is an (S_1,V)-unifier of T.
To show that z is most general as an (S_1,V)-unifier, let z_1 be an (S_1,V)-unifier of T. We need to prove that z_1 can be factored through z as $z_1 = w_1 * z$, for some (S_1,V)-substitution w_1. Let z_1' be an (S_1,V)-most general unifier of T. Note that z_1' is also an (S_0,V)-unifier by the first part of the argument and therefore, $z_1' = w * z$ for some (S_0,V)-substitution w and w can be chosen to be also an (S_1,V)-substitution since z_1' is an (S_1,V)-substitution. By the definition of most general unifier, there is an (S_1,V)-substitution w' with $z_1 = w' * z_1'$. We now have $z_1 = w' * z_1' = w' * (w * z) = (w' * w) * z$, and therefore $w_1 = w' * w$ is the desired (S_1,V)-substitution.
We have thus shown that $\mathbf{MGUNIF}^{S_0,V}(T) \subseteq \mathbf{MGUNIF}^{S_1,V}(T)$. The reverse inclusion can be shown in a similar manner.

(78) Let T be a finite set of (S,V)-terms and S' be a reduct of S. Prove that

(a) T is (S,S',V)-unifiable if and only if T is (S,V)-unifiable and $\mathsf{MGUSIG}(T,S)$ is a reduct of S',
(b) if T is (S,S',V)-unifiable, then

$$\mathbf{MGUNIF}^{S,V}_{S'}(T) = \mathbf{MGUNIF}^{S,V}(T).$$

Solution. Suppose first that T is (S,S',V)-unifiable and let s' be a most general (S,S',V)-unifier of T. Then, s' is also an (S,V)-unifier of T. Let s be a most general (S,V)-unifier of T. Then, $s' = z * s$ for some (S,V)-substitution z.

It follows from Exercise 74 that $\mathsf{MGUSIG}(T, S) = \mathsf{SIG}(s, S) \preceq \mathsf{SIG}(s', S) \preceq S'$. We leave to the reader the argument for the reverse implication.

For the second part, let $s \in \mathbf{MGUNIF}^{S,V}_{S'}(T)$. This means that s is an (S', V)-substitution, hence an (S, V)-substitution and $|s(T)| \leq 1$, so s is an (S, V)-unifier of T, and, consequently, there is a most general (S, V)-unifier s_1 of T. By Part (a), s_1 is an (S', V)-substitution. In turn, this implies that s_1 is also an (S, S', V)-unifier of T. Hence, there is an (S', V)-substitution z_1 (which is also an (S, V)-substitution because S' is a reduct of S) such that $s_1 = z_1 * s$. Now, let s' be any (S, V)-unifier of T. Since s_1 is a most general (S, V)-unifier of T, there is an (S, V)-substitution z with $s' = z * s_1 = z * (z_1 * s) = (z * z_1) * s$, which shows that s is a most general (S, V)-unifier of T.

Conversely, suppose that $s \in \mathbf{MGUNIF}^{S,V}(T)$, that is, s is an (S, V)-substitution, $|s(T)| \leq 1$, and for every (S, V)-substitution z with $|z(T)| \leq 1$, these is an (S, V)-substitution z'' such that $z = z'' * s$. Since $\mathsf{MGUSIG}(T, S)$ is a reduct of S', by the first part, it follows that s is an (S', V)-substitution. Let z be an (S', V)-substitution with $|z(T)| \leq 1$. Note that z is also an (S, V)-substitution, so there is an (S, V)-substitution z'' with $z = z'' * s$. Let z' be the (S', V)-substitution obtained from z'' by defining $z'(x) = x$ for all variables x that do not appear in the range of s. Then, $z = z' * s$, so $s \in \mathbf{MGUNIF}^{S,V}_{S'}(T)$.

(79) Let S, S', S_0, S'_0 be four signatures such that

$$S' \preceq S, S'_0 \preceq S_0, S_0 \preceq S, S'_0 \preceq S',$$

and let V be a set of variables. Suppose that T is a finite set of (S_0, V)-terms such that $\mathtt{r}_{S_0,T,\min} \sqcap S' \preceq S'_0$. Show that T is (S, S', V)-unifiable if and only if it is (S_0, S'_0, V)-unifiable; furthermore, prove that

$$\mathbf{MGUNIF}^{S,V}_{S'}(T) = \mathbf{MGUNIF}^{S_0,V}_{S'_0}(T). \qquad (1.2)$$

Solution. Clearly, if T is (S_0, S'_0, V)-unifiable, then it is (S, S', V)-unifiable. Conversely, suppose that T is (S, S', V)-unifiable. Let s be a most general (S, S', V)-unifier of T. By Exercise 76, $\mathsf{SIG}(s, S)$ is a reduct of $\mathtt{r}_{S,T,\min} \sqcap S' = \mathtt{r}_{S_0,T,\min} \sqcap S' \preceq S'_0$. Thus, s is an (S_0, S'_0, V)-unifier of T, which establishes the reverse implication.

If T is not (S, S', V)-unifiable, then it is not (S_0, S'_0, V)-unifiable by the first part, and hence both sides of Equality (1.2) are empty. On the other hand, if T is (S, S', V)-unifiable, then, again by the first part, it is (S_0, S'_0, V)-unifiable and Equality (1.2) follows from Supplements 78 and 77.

(80) Let S, S' be signatures such that S' is a reduct of S, V be a set of variables, T be a finite, unifiable set of (S, V)-terms, and s be a most general (S, S', V)-unifier for T. Show that if $y_0, y_1 \in V - \mathtt{V}(T)$, then $s(y_0), s(y_1)$ are distinct variables in V. Conclude that for a finite, unifiable set T of (S, V)-terms and a most general (S, V)-unifier s for T, $s(y)$ is a variable for every $y \in V - \mathtt{V}(T)$ and furthermore the restriction of s to $V - \mathtt{V}(T)$ is injective.

Solution. Let s_0 be the most general (S, S', V)-unifier of T produced by the unification algorithm. By the proof of Lemma 1.6.23, $s_0(y_0) = y_0$ and $s_0(y_1) = y_1$. Since s is a most general (S, S', V)-unifier, there is an (S', V)-substitution z such that $s_0 = z * s$. In particular, $y_0 = s_0(y_0) = \bar{z}(s(y_0))$ and $y_1 = s_0(y_1) = \bar{z}(s(y_1))$. Since $\bar{z}$ does not affect function symbols, it follows that $s(y_0), s(y_1)$ are variables, and they are distinct.

(81) Let S, S' be signatures such that S' is a reduct of S, V be a set of variables, T be a finite, unifiable set of (S, V)-terms, and s be a most general (S, S', V)-unifier for T. Prove that

(a) $|\mathtt{V}(s(T))| \leq |\mathtt{V}(T)|$,
(b) $|\mathtt{V}(T) - \mathtt{V}(s(T))| \geq |\mathtt{V}(s(T)) - \mathtt{V}(T)|$.

Solution. Let s^* be an mgu of T produced by the unification algorithm. Note that for the sequence of sets of terms $T_0, T_1, \ldots, T_n$ produced by the algorithm, $\mathtt{V}(T) = \mathtt{V}(T_0) \supset \mathtt{V}(T_1) \supset \cdots \supset \mathtt{V}(T_n) = \mathtt{V}(s^*(T))$, so the result holds for s^*. If s is an arbitrary mgu, then by Supplement 68, there is a substitution z such that $s^* = z * s$ and z is variable-pure and injective on $\mathtt{V}(\mathrm{Ran}(s))$. This implies that

$$|\mathtt{V}(T)| \geq |\mathtt{V}(s^*(T))| = |\mathtt{V}(z * s(T))| = |\mathtt{V}(z(s(T)))| = |\mathtt{V}(s(T))|.$$

The second part follows from elementary set-theoretical arguments.

(82) Prove the following generalization of Supplement 81: Let S, S' be signatures such that S' is a reduct of S, V be a set of

variables, T be a finite, unifiable set of (S, V)-terms, and V_0 be a finite set of variables such that $\mathtt{V}(T) \subseteq V_0$. Let s be a most general (S, S', V)-unifier for T. Prove that

(a) $|\mathtt{V}(s(V_0))| \leq |V_0|$,

(b) $|V_0 - \mathtt{V}(s(V_0))| \geq |\mathtt{V}(s(V_0)) - V_0|$.

Solution. Clearly, $|V_0 - \mathtt{V}(T)| \geq |s(V_0 - \mathtt{V}(T))|$. By Supplement 80, since s is an mgu of T, we have $|V_0 - \mathtt{V}(T)| \geq |\mathtt{V}(s(V_0 - \mathtt{V}(T))|$. This allows us to write

$$\begin{aligned} |V_0| &= |\mathtt{V}(T)| + |V_0 - \mathtt{V}(T)| \\ &\geq |\mathtt{V}(s(T))| + |\mathtt{V}(s(V_0 - \mathtt{V}(T)))| \\ &\quad \text{(by Supplement 81)} \\ &\geq |\mathtt{V}(s(V_0))|. \end{aligned}$$

We leave to the reader the proof of the second part.

(83) Let S, S' be signatures such that S' is a reduct of S, V be a set of variables, T be a finite, unifiable set of (S, V)-terms, and s be a most general (S, S', V)-unifier for T. If V_0 is a finite set of variables such that $\mathtt{V}(T) \subseteq V_0$, prove that there is a finite most general (S, S', V)-unifier s' of T such that $s'(v) = s(v)$ for every $v \in V_0$.

Solution. The second inequality of Supplement 82 allows us to infer the existence of an injection $f : \mathtt{V}(s(V_0) - V_0) \longrightarrow V_0 - \mathtt{V}(s(V_0))$. This, in turn, allows us to define the (S', V)-substitution s' as

$$s'(x) = \begin{cases} s(x) & \text{if } x \in V_0 \\ f(x) & \text{if } x \in \mathtt{V}(s(V_0)) - V_0 \\ x & \text{if } x \notin V_0 \cup \mathtt{V}(s(V_0)). \end{cases}$$

By definition, s' agrees with s on V_0 and since $\mathtt{V}(T) \subseteq V_0$, it follows that s' is a unifier of T. Moreover, since $V_0 \cup \mathtt{V}(s(V_0))$ is finite, it follows that s' is a finite substitution. Since s is an mgu of T and s' is a unifier of T, we have $s' = z' * s$, for some substitution z'. To complete the proof that s' is an mgu of T, we will show that there is a substitution z such that $s = z * s'$.

This establishes that $s' \equiv s$ and hence by Theorem 1.6.6, s' is an mgu of T. Let z be the (S', V)-substitution given by

$$z(y) = \begin{cases} y & \text{if } y \in \mathtt{V}(s(V_0)) \\ s(f^{-1}(y)) & \text{if } y \in \mathrm{Ran}(f) \\ s(y) & \text{if } y \notin V_0 \cup \mathtt{V}(s(V_0)) \\ y & \text{if } y \in V_0 - (\mathrm{Ran}(f) \cup \mathtt{V}(s(V_0))). \end{cases}$$

We need to consider three cases for a variable $x \in V$ and show that $s(x) = z(s'(x))$.
`Case 1:` $x \in V_0$. We have $z(s'(x)) = z(s(x)) = s(x)$ since $z(y) = y$ for all $y \in \mathtt{V}(s(V_0))$.
`Case 2:` $x \in \mathtt{V}(s(V_0)) - V_0$. Then,

$$z(s'(x)) = z(f(x)) = s(f^{-1}(f(x))) = s(x).$$

`Case 3:` $x \notin V_0 \cup \mathtt{V}(s(V_0))$. This allows us to write $z(s'(x)) = z(x) = s(x)$.

(84) Let S, S' be signatures such that S' is a reduct of S and let V be a set of variables. Suppose that V_0, V_1 are two disjoint subsets of V and let T_0, T_1 be two finite sets of (S, V)-terms such that $\mathtt{V}(T_i) \subseteq V_i$, for $i = 0, 1$. Further, for $i = 0, 1$, let s_i be a most general (S, S', V)-unifier of T_i such that $\mathtt{V}(s_i(T_i)) \subseteq V_i$. Let z be a most general (S, S', V)-unifier of $s_0(T_0) \cup s_1(T_1)$. Define

$$s(x) = \begin{cases} s_0(x) & \text{if } x \in V_0 \\ s_1(x) & \text{if } x \in V_1 \\ x & \text{if } x \notin V_0 \cup V_1. \end{cases}$$

Prove that $z_{01} = z * s$ is a most general (S, S', V)-unifier of $T_0 \cup T_1$.

Solution. We have $z_{01}(T_0 \cup T_1) = z * s(T_0 \cup T_1) = z(s(T_0) \cup s(T_1)) = z(s_0(T_0) \cup s_1(T_1))$. Since z is a unifier of $s_0(T_0) \cup s_1(T_1)$, we have $|z_{01}(T_0 \cup T_1)| = 1$, so z_{01} is a unifier of $T_0 \cup T_1$ (see Figure 1.7).

Let z' be an (S, S', V)-unifier of $T_0 \cup T_1$. Since z' unifies both T_0 and T_1, there are z'_0 and z'_1 such that $z' = z'_0 * s_0$ and

$$\begin{tikzcd}
T_0 \arrow[d, "s_0\text{: an }(S,S',V)\text{-mgu}"] & (S,V_0)\text{-terms} & T_1 \arrow[d, "s_1\text{: an }(S,S',V)\text{-mgu}"] & (S,V_1)\text{-terms} & T_1 \cup T_1 \arrow[d] \\
s_0(T_0) \arrow[dr] & & s_0(T_0) \arrow[dl] & & z_{01}(T_0 \cup T_1) \\
& z(s_0(T_0) \cup s_1(T_1)) & & & \\
& z\text{: an }(S,S',V)\text{-mgu} & & &
\end{tikzcd}$$

Fig. 1.7. $z_{01} = z * s$ is a most general (S,S',V)-unifier of $T_0 \cup T_1$.

$z' = z'_1 * s_1$. Define z'' by

$$z''(x) = \begin{cases} z'_0(x) & \text{if } x \in V_0 \\ z'_1(x) & \text{if } x \in V_1 \\ z'(x) & \text{if } x \notin V_0 \cup V_1. \end{cases}$$

Then, $z''(s_0(T_0) \cup s_1(T_1)) = z''(s_0(T_0)) \cup z''(s_1(T_1)) = z'_0(s_0(T_0)) \cup z'_1(s_1(T_1)) = z'(T_0) \cup z'(T_1) = z'(T_0 \cup T_1)$. Therefore, $|z''(s_0(T_0) \cup s_1(T_1))| = |z'(T_0 \cup T_1)| = 1$, so we can conclude that z'' is an (S,S',V)-unifier of $s_0(T_0) \cup s_1(T_1)$. Since z is a most general unifier of $s_0(T_0) \cup s_1(T_1)$, we can write $z'' = \hat{z} * z$. We claim that $z' = \hat{z} * z_{01}$. First, note that $\hat{z} * z_{01} = \hat{z} * (z * s) = (\hat{z} * z) * s = z'' * s$. Hence, it suffices to show that $z' = z'' * s$. For $x \in V_0$, we have $z'' * s(x) = z''(s_0(x)) = z'_0(s_0(x)) = z'(x)$; similarly, for $x \in V_1$, we have $z'' * s(x) = z''(s_1(x)) = z'_1(s_1(x)) = z'(x)$. If $x \notin V_0 \cup V_1$, then $z'' * s(x) = z''(x) = z'(x)$. We have shown that z' is factorable through z_{01} and hence z_{01} is an (S,S',V)-mgu of $T_0 \cup T_1$.

Labeled Ordered Trees

(85) Prove that the set $D = \{(a_0, \ldots, a_{n-1}) \in \mathrm{Seq}(\mathbf{N}) \mid n \in \mathbf{N}$ and $0 \leq a_j \leq j$ for all $j, 0 \leq j \leq n-1\}$ is a tree domain.

(86) Let D, E be tree domains.

(a) Prove that $D \cup E$ and $D \cap E$ are tree domains.

(b) Prove that if $q, qr \in D$, then $(D_{[q]})_{[r]} = D_{[qr]}$.

(c) Prove that if $q \in D \cap E$, then

$$(D \cup E)_{[q]} = D_{[q]} \cup E_{[q]},$$
$$(D \cap E)_{[q]} = D_{[q]} \cap E_{[q]}.$$

(d) Prove that if $D_0, \ldots, D_{n-1}$ are n tree domains and $0 \leq i \leq n-1$, then

$$\langle D_0, \ldots, D_{n-1} \rangle_{[i]} = D_i.$$

(87) Let D, D' be tree domains and let $r \in D$. Prove that

$$\begin{aligned}&\mathrm{LEAVES}(D[r \to D'])\\ &\quad = (\mathrm{LEAVES}(D) - r\mathrm{Seq}(\mathbf{N})) \cup r\mathrm{LEAVES}(D').\end{aligned}$$

(88) Let D, D_1, D_2 be three tree domains and let $r_1, r_2 \in D$ be two sequences such that neither of them is a prefix of the other. Prove that

$$(D[r_1 \to D_1])[r_2 \to D_2] = (D[r_2 \to D_2])[r_1 \to D_1].$$

(89) Let $D_0, \ldots, D_{n-1}, D$ be tree domains. Prove that

$$\langle D_0, \ldots, D_{n-1} \rangle [ir \to D] = \langle D_0, \ldots, D_i[r \to D], \ldots, D_{n-1} \rangle$$

for $0 \leq i \leq n-1$ and $r \in D_i$.

(90) Let D be a tree domain such that no path of D has length greater than k and no node has more than n immediate descendants. Prove that $|D| \leq 1 + n + \cdots + n^k$ and D has no more than n^k leaves.

Hint. Use strong induction on k.

(91) Prove that the recursive definition of the set of finite tree domains given in Theorem 1.7.22 satisfies the unique readability condition.

A *tree semi-domain* is a subset D of $\mathrm{Seq}(\mathbf{N})$ that satisfies the first two conditions of Definition 1.7.1. The notions introduced in Definitions 1.7.4 and 1.7.5 carry over to tree semi-domains.

(92) Show that König's lemma holds for tree semi-domains.

(93) Let $\mathtt{T}_0, \ldots, \mathtt{T}_{n-1}$ be n lots and r be a node of $\mathtt{T}_i$. Prove that

$$(\mathtt{T}_0, \ldots, \mathtt{T}_{n-1}; \theta)_{[ir]} = (\mathtt{T}_i)_{[r]}.$$

Conclude that

$$(\mathtt{T}_0, \dots, \mathtt{T}_{n-1}; \theta)_{[i]} = \mathtt{T}_i$$

for $0 \leq i \leq n-1$ by taking $r = \lambda$.

(94) Let $\mathcal{T}_0, \dots, \mathcal{T}_{n-1}$ be n marked lots and r be a node of $\mathcal{T}_i$. Prove that

$$(\mathcal{T}_0, \dots, \mathcal{T}_{n-1}; \theta)_{[ir]} = (\mathcal{T}_i)_{[r]}.$$

Conclude that

$$(\mathcal{T}_0, \dots, \mathcal{T}_{n-1}; \theta)_{[i]} = \mathcal{T}_i$$

for $0 \leq i \leq n-1$ by taking $r = \lambda$.

(95) Let $\mathtt{T}, \mathtt{T}'$ be lots and let r be a node of $\mathtt{T}$. Show that

$$\begin{aligned} &\mathrm{LEAVES}_\theta(\mathtt{T}[r \to \mathtt{T}']) \\ &\quad = (\mathrm{LEAVES}_\theta(\mathtt{T}) - r\mathrm{Seq}(\mathbf{N})) \cup r\mathrm{LEAVES}_\theta(\mathtt{T}') \end{aligned}$$

for every object θ.

(96) Prove that the recursive definition of the set of finite U-lots given in Corollary 1.7.24 satisfies the unique readability condition.

Let $\mathtt{T}$ be a lot and let f be a function such that $\mathrm{Ran}(\mathtt{T}) \subseteq \mathrm{Dom}(f)$. Then $f(\mathtt{T})$ denotes the lot $f \circ \mathtt{T}$. If $\mathcal{T} = (\mathtt{T}, M)$ is a marked lot and f is a function which satisfies the same condition, then $f(\mathcal{T})$ is the marked lot $(f(\mathtt{T}), M)$.

(97) Let $\mathcal{T} = (\mathtt{T}, M)$ be a marked lot, θ be an object, and f be a function such that $\mathrm{Ran}(\mathtt{T}) \cup \{\theta\} \subseteq \mathrm{Dom}(f)$. Prove that if f is injective on the set $\{\mathtt{T}(q) \mid q \in \mathrm{LEAVES}(\mathtt{T})\} \cup \{\theta\}$, then

$$f(L_\theta(\mathcal{T})) = L_{f(\theta)}(f(\mathcal{T})).$$

Solution. We have

$$f(L_\theta(\mathcal{T})) = (f(\mathtt{T}), M \cup \mathrm{LEAVES}_\theta(\mathtt{T})) \text{ and}$$
$$L_{f(\theta)}(f(\mathcal{T})) = (f(\mathtt{T}), M \cup \mathrm{LEAVES}_{f(\theta)}(f(\mathtt{T}))).$$

Thus, it suffices to show that

$$\mathrm{LEAVES}_\theta(\mathtt{T}) = \mathrm{LEAVES}_{f(\theta)}(f(\mathtt{T})).$$

If $q \in \mathrm{LEAVES}_\theta(\mathtt{T})$, then $q \in \mathrm{LEAVES}(\mathtt{T}) = \mathrm{LEAVES}(f(\mathtt{T}))$ and $\mathtt{T}(q) = \theta$, so $f(\mathtt{T})(q) = f(\theta)$ and $q \in \mathrm{LEAVES}_{f(\theta)}(f(\mathtt{T}))$.

Conversely, if $q \in \text{LEAVES}_{f(\theta)}(f(\mathtt{T}))$, then $q \in \text{LEAVES}(f(\mathtt{T})) = \text{LEAVES}(\mathtt{T})$ and $f(\mathtt{T}(q)) = f(\theta)$. Since f is injective on $\{\mathtt{T}(q) \mid q \in \text{LEAVES}(\mathtt{T})\} \cup \{\theta\}$, it follows that $\mathtt{T}(q) = \theta$, so $q \in \text{LEAVES}_\theta(\mathtt{T})$.

(98) Let $\mathtt{T}_0, \ldots, \mathtt{T}_{n-1}, \mathtt{T}$ be lots. Prove that

$$(\mathtt{T}_0, \ldots, \mathtt{T}_{n-1}; \theta)[ir \to \mathtt{T}] = (\mathtt{T}_0, \ldots, \mathtt{T}_i[r \to \mathtt{T}], \ldots, \mathtt{T}_{n-1}; \theta)$$

for all $i \in \{0, \ldots, n-1\}$, $r \in \text{Dom}(\mathtt{T}_i)$ and objects θ.

Let $\mathcal{T} = (\mathtt{T}, M)$, $\mathcal{T}' = (\mathtt{T}', M')$ be two marked lots and let $r \in D = \text{Dom}(\mathtt{T})$. Denote the pair $(\mathtt{T}[r \to \mathtt{T}'], (M - r\text{Seq}(\mathbf{N})) \cup rM')$ by $\mathcal{T}[r \to \mathcal{T}']$.

(99) Let $\mathcal{T}, \mathcal{T}'$ be two marked lots and let r be a node of $\mathcal{T}$. Prove that the pair $\mathcal{T}[r \to \mathcal{T}']$ is a marked lot.

(100) Let θ be an object. Show that $L_\theta(L_\theta(\mathcal{T})) = L_\theta(\mathcal{T})$ for every marked lot $\mathcal{T}$.

(101) Let $\mathcal{T}, \mathcal{T}'$ be two marked lots and let θ be an object. If r is a node of $\mathcal{T}$, show that

$$L_\theta(\mathcal{T}[r \to \mathcal{T}']) = L_\theta(\mathcal{T})[r \to L_\theta(\mathcal{T}')].$$

(102) Let $\mathcal{T}_0, \ldots, \mathcal{T}_{n-1}, \mathcal{T}$ be marked lots. If r is a node of $\mathcal{T}_i$, prove that

$$(\mathcal{T}_0, \ldots, \mathcal{T}_{n-1}; \theta)[ir \to \mathcal{T}] = (\mathcal{T}_0, \ldots, \mathcal{T}_i[r \to \mathcal{T}], \ldots, \mathcal{T}_{n-1}; \theta)$$

for every object θ.
Hint. Use Exercise 98.

(103) Let $S = (F, \nu)$ be a signature and let V be a set of variables. An (S, V)-*term tree* is a finite lot $\mathtt{T}$ satisfying the following properties:

- If q is a leaf of $\mathtt{T}$, then $\mathtt{T}(q) \in V \cup F_0^S$.
- If q is an interior node of $\mathtt{T}$, and its immediate descendants are $q0, \ldots, qn$, then there exists an $n+1$-ary function symbol $f \in F$ such that $\mathtt{T}(q) = f(\mathtt{T}(q0), \ldots, \mathtt{T}(qn))$.

(a) Prove that a finite sequence is an (S, V)-term if and only if there is an (S, V)-term tree whose root is labeled with the sequence.

(b) Prove that if T is an (S, V)- term tree with root labeled t, then the set of labels of T equals $\mathrm{SUBT}(t)$.

Formal Systems

(104) Show the following converse of Corollary 1.8.8. Let $\mathcal{F} = (U, A, I)$ be a formal system and let $G_0, G_1 \subseteq U$. Prove that if $G_0 \vdash_{\mathcal{F}} \theta$ implies $G_1 \vdash_{\mathcal{F}} \theta$ for every $\theta \in U$, then $G_0 \subseteq \mathrm{Thm}(\mathcal{F}_{G_1})$.

(105) Give an example of a formal system $\mathcal{F} = (U, A, I)$ and a non-empty subset R of $U^n \times U$ for some $n > 0$ such that the formal system $\mathcal{F}' = (U, A, I \cup \{R\})$ is equivalent to $\mathcal{F}$, but R is not a derived rule of $\mathcal{F}$.

(106) Let $\mathcal{F} = (U, A, I)$ be a formal system and let $\kappa : \mathcal{P}(U) \longrightarrow \mathcal{P}(U)$ be the mapping given by $\kappa(G) = \mathrm{Thm}(\mathcal{F}_G)$. Prove that κ is a closure operator.

(107) Consider the formal system $\mathcal{F}_{td} = (\mathcal{P}(\mathrm{Seq}(\mathbf{N})), \{\{\lambda\}\}, \{\, \mathsf{R}_k \mid k \geq 1\})$, where R_k is the rule

$$\frac{D_0, \ldots, D_{k-1}}{\langle D_0, \ldots, D_{k-1} \rangle}.$$

Prove that $\mathrm{Thm}(\mathcal{F}_{td})$ equals the set of all finite tree domains.

(108) The game of tick-tack-toe can be regarded as a formal system as follows. The set of objects U is the set of 3×3-matrices whose entries are in the set $\{X, O, \flat\}$. (The symbol $\flat$ stands for blank.) The number of X entries (O entries) of an object M is denoted by M_X (M_O). An object M is

- X-winning (O-winning) if it has a row, column, or diagonal of Xs (Os),
- winning if it is either X-winning or O-winning,
- X-turn if $M_X = M_O$,
- O-turn if $M_X = M_O + 1$.

The single axiom is the matrix all of whose entries are $\flat$ and the set of rules is $\{\, \mathsf{R}_X, \mathsf{R}_O\}$, where

- the rule R_X is defined by

$$\frac{M}{M'}\ \mathsf{R}_X$$

if and only if M is X-turn and not winning, and M' is obtained from M by changing a $\flat$ into an X,

- the rule R_O is defined by

$$\frac{M}{M'}\ \mathsf{R}_O$$

  if and only if M is O-turn and not winning, and M' is obtained from M by changing a $\flat$ into an O.

Prove that M is a theorem of this formal system if and only if either M is X-turn and not X-winning, or M is O-turn and not O-winning; in other words, prove the soundness and completeness of the formal system for the above set of objects.

(109) Recall that (see [13]) a context-free grammar is a quadruple $G = (N, T, S, P)$, where N and T are disjoint alphabets referred to as the nonterminal and terminal alphabet, respectively, $S \in N$ is the start symbol, and P is a finite subset of $N \times (N \cup T)^*$ whose elements are referred to as productions. We also defined the relations $\underset{G}{\Rightarrow}$ and $\overset{*}{\underset{G}{\Rightarrow}}$. A word $\alpha \in (N \cup T)^*$ is called a sentential form of G if $S \overset{*}{\underset{G}{\Rightarrow}} \alpha$.
Starting from a context-free grammar G, define a formal system $\mathcal{F}_G = ((N \cup T)^*, S, \{\, \mathsf{R}_{X\to\alpha} \mid X \to \alpha \in P\})$, where

$$\frac{\gamma X \gamma'}{\gamma\alpha\gamma'}\ \mathsf{R}_{X\to\alpha}$$

for every $\gamma, \gamma' \in (N \cup T)^*$.
Prove that $\mathrm{Thm}(\mathcal{F}_G)$ equals the set of sentential forms of G.

(110) A formal system $\mathcal{F} = (U, A, I)$ is *semi-effectively specified* if U is an effectively enumerable set, A is a semidecidable subset of U, and $P(\mathcal{F})$ is a semidecidable set, where $P(\mathcal{F})$ is the set introduced in Definition 1.8.27.
Prove that

(a) if $\mathcal{F} = (U, A, I)$ is a semi-effectively specified formal system, then the set of proofs in $\mathcal{F}$ is a semidecidable subset of $\mathrm{Seq}(U)$,

(b) under the conditions of Part (a), show that $\mathrm{Thm}(\mathcal{F})$ is also semidecidable,

(c) if $\mathcal{F} = (U, A, I)$ is an effectively specified formal system and G is a semidecidable subset of U, then $\{\theta \mid G \vdash_{\mathcal{F}} \theta\}$ is a semidecidable subset of U.

Linear Orders

(111) Let $(M, \preceq)$ be a linearly ordered set. Define a relation $\prec'$ on $\text{Seq}(M)$ by $q \prec r$ if either q is a proper prefix of r or for some i such that $i < \min(|q|, |r|)$ we have $q \restriction \{0, \ldots, i-1\} = r \restriction \{0, \ldots, i-1\}$ and $q(i) \prec r(i)$.
Prove that

(a) $\prec'$ is a strict linear order on $\text{Seq}(M)$,
(b) $(\text{Seq}(M), \preceq')$ is isomorphic to $(\mathbf{N}, \leq)$ if and only if $|M| = 1$.

(112) Let $(M, \preceq)$ be a linearly ordered set. Define a relation $\prec''$ on $\text{Seq}(M)$ by $q \prec'' r$ if either $|q| < |r|$ or $|q| = |r|$ and $q \prec' r$, where $\prec'$ was introduced in Exercise 111.
Prove that

(a) $\prec''$ is a strict linear order on $\text{Seq}(M)$,
(b) $(\text{Seq}(M), \preceq'')$ is isomorphic to $(\mathbf{N}, \leq)$ if and only if M is a finite, nonempty set.

1.11 Bibliographical Comments

The notion of formal system is due to David Hilbert [19–22] who built on Frege's work [14]. See also Smullyan's book [35].

The notion of property of finite character is due to Smullyan [36]. Closure systems are presented in [8].

Chapter 2

Propositional Logic–Syntax and Semantics

2.1 Introduction

Propositional logic is a mathematical model of a certain simple type of reasoning. A brief introduction to this area is contained in Section 4.8 of [13]. Here we undertake a more detailed study of the subject. The presentation will be self-contained, but we will refer occasionally to the previous reference, mainly for motivation.

Propositional logic deals with *statements* (also known as *propositions*). For us, a statement is simply an assertion which is either true or false; statements are usually expressed by declarative sentences. Statements can be built from simpler statements using connectives; the truth value of a statement built up in this manner is determined in a fixed way by the truth values of the simpler statements.

We will be concerned primarily with the following connectives: conjunction, disjunction, implication (also called conditional), biimplication (also called biconditional), and negation. The linguistic counterparts of these connectives are given in the following table:

Connective	English Counterpart
conjunction	and
disjunction	or
implication	implies
	if . . . then
biimplication	if and only if
negation	not

Disjunction is used here in the inclusive sense; in other words, the disjunction of two statements is true if at least one of the statements is true. The alternative (the exclusive sense) generates a statement which is true when exactly one of the statements is true.

We refer the reader to Section 4.8 of [13] for a more detailed discussion of these connectives.

2.2 Formulas

Since we are interested only in the forms that statements can have, and not the actual statements themselves, we need to build symbolic representations for complex statements. These representations are known as formulas and we define them in this section.

We build formulas of propositional logic using the following components, called the *symbols of propositional logic*:

- a countably infinite set $SV = \{p_0, p_1, \ldots\}$ whose elements are called *statement variables*,
- five *connective symbols*: $\neg, \vee, \wedge, \rightarrow, \leftrightarrow$,
- two punctuation symbols denoted by the left and right parentheses.

We assume that all symbols specified above are distinct.

The five connective symbols are meant to stand for connectives according to the following table:

Connective Symbol	Connective
$\neg$	negation
$\vee$	disjunction
$\wedge$	conjunction
$\rightarrow$	implication
$\leftrightarrow$	biimplication

Definition 2.2.1. The set PLFORM of *formulas of propositional logic* (also called the set of *statement forms*) is the set of sequences of symbols of propositional logic given by the following inductive definition:

(1) Every statement variable is a formula of propositional logic.
(2) If φ is a formula of propositional logic, then so is $(\neg\varphi)$.
(3) If φ and ψ are formulas of propositional logic, then so are $(\varphi \vee \psi)$, $(\varphi \wedge \psi)$, $(\varphi \rightarrow \psi)$, and $(\varphi \leftrightarrow \psi)$.

We refer to the formula $(\varphi \vee \psi)$ as the *disjunction of* φ *and* ψ and to the formula $(\varphi \wedge \psi)$ as the *conjunction of* φ *and* ψ. □

In the above definition, we have followed the usual convention of identifying an element of a set with the sequence of length 1 whose only entry is that element, and we have denoted concatenation of sequences as juxtaposition even though the set over which the sequences are drawn is not an alphabet.

Observe that Rule 3 of the definition actually incorporates four rules: $3_\vee, 3_\wedge, 3_\rightarrow, 3_\leftrightarrow$, where, for example, $3_\vee$ is the rule:

$3_\vee$ If φ and ψ are formulas of propositional logic, then so is $(\varphi \vee \psi)$.

Example 2.2.2. The sequence $((\neg(p_0 \vee p_1)) \leftrightarrow p_6)$ is a formula because

(1) p_0 is a formula by Rule 1,
(2) p_1 is a formula by Rule 1,
(3) p_6 is a formula by Rule 1,
(4) $(p_0 \vee p_1)$ is a formula because of Steps 1 and 2 and Rule $3_\vee$,

(5) $(\neg(p_0 \vee p_1))$ is a formula because of Step 4 and Rule 2,
(6) $((\neg(p_0 \vee p_1)) \leftrightarrow p_6)$ is a formula because of Step 5, Step 3, and Rule $3_{\leftrightarrow}$.

□

We use the letters p, q, r to denote statement variables and $\varphi, \psi, \theta, \alpha, \beta, \gamma$ to denote formulas of propositional logic.

Definition 2.2.3. A *negative formula* is a formula φ such that $\varphi = (\neg\psi)$ for some formula $\psi \in$ PLFORM. A formula is *positive* if it is not negative. □

Note that if $\varphi = (\neg\psi)$, then the formula ψ is uniquely determined by φ as the substring obtained from φ by removing the first two symbols and the last symbol.

Definition 2.2.4. A *literal* is a formula ℓ which is either p or $(\neg p)$ for some statement variable p. In the former case, ℓ is a *positive literal* while in the latter case, ℓ is a *negative literal.*

The set of all literals will be denoted by LIT.

If S is a set of statement variables and $p \in S$, we refer to both p and $(\neg p)$ as *literals over S*. □

Next, we introduce a notation that is a sort of "optimized negation."

Definition 2.2.5. Let φ be a formula. Then $\overline{\varphi}$, the *complement* of φ, is defined by

$$\overline{\varphi} = \begin{cases} (\neg\varphi) & \text{if } \varphi \text{ is a positive formula,} \\ \psi & \text{if } \varphi = (\neg\psi) \text{ is a negative formula.} \end{cases}$$

□

Note that the complement of a literal is a literal.

The $\overline{\varphi}$ notation is naturally extended to sets of formulas and sequences of formulas. For example, if $k = (\varphi_0, \ldots, \varphi_{n-1})$, then $\overline{k} = (\overline{\varphi_0}, \ldots, \overline{\varphi_{n-1}})$.

It is sometimes important to encode formulas of propositional logic as words over a fixed alphabet. Here, we specify one way in which this can be done. We take our alphabet to be

$V = \{p, 0, 1, \neg, \vee, \wedge, \rightarrow, \leftrightarrow, (,)\}$ and we order the symbols of V as they were just listed. Let $\mathtt{k}_n$ be the usual binary representation for the natural number n (so $\mathtt{k}_0 = 0$ and for $n > 0$, $\mathtt{k}_n$ has no leading 0s). We encode every symbol x of propositional logic as a word $\mathsf{code}(x)$ over V as follows: every statement variable p_i is encoded as the word $p\mathtt{k}_i$; the other symbols are encoded as themselves. The code $\mathsf{code}(\varphi)$ of a formula φ is obtained by concatenating the codes of the symbols which make up the formula.

Example 2.2.6. The formula discussed in Example 2.2.2 is encoded as

$$((\neg(p0 \vee p1)) \leftrightarrow p110).$$

$\square$

Note that if x, y are two distinct symbols of propositional logic, then $\mathsf{code}(x)$ is not a suffix of $\mathsf{code}(y)$. Therefore, by Supplement 13 of Chapter 1, the mapping $\mathsf{code} : \mathrm{PLFORM} \longrightarrow V^*$ is injective, so PLFORM is countably infinite.

When an algorithm which takes formulas as inputs is being analyzed and it is desired to measure the size of the input in order to say something about the running time of the algorithm as a function of input size, formulas are coded as strings over V or some other (finite) alphabet.

Definition 2.2.7. The *size of a formula* $\varphi \in$ PLFORM is $\mathtt{size}(\varphi) = |\mathsf{code}(\varphi)|$. $\square$

Example 2.2.8. Let $\theta_i = (p_{2i-1} \vee p_{2i})$ for $i \geq 1$ and let $\theta_0 = p_0$. The sizes of p_{2i-1} and p_{2i} are $2 + \lfloor \log_2(2i-1) \rfloor$ and $2 + \lfloor \log_2(2i) \rfloor$, respectively.

Therefore, we have

$$\mathtt{size}(\theta_i) = \begin{cases} \lfloor \log_2(2i-1) \rfloor + \lfloor \log_2(2i) \rfloor + 7 & \text{if } i > 0 \\ 2 & \text{if } i = 0. \end{cases}$$

$\square$

We can define a well-ordering "$\preceq$" on PLFORM as follows: $\varphi \preceq \psi$ if $|\mathsf{code}(\varphi)| < |\mathsf{code}(\psi)|$ or if $|\mathsf{code}(\varphi)| = |\mathsf{code}(\psi)|$ and $\mathsf{code}(\varphi)$

precedes $\mathsf{code}(\psi)$ in the lexicographic order on V^*. We will refer to $\preceq$ as the *standard ordering* of the formulas.

If φ is a formula of propositional logic, then we will denote the set of variables that occur in φ by $SV(\varphi)$, that is,

$$SV(\varphi) = \{p \in SV \mid \mathsf{occ}_p(\varphi) \neq \emptyset\}.$$

If Γ is a set of formulas, then we denote by $SV(\Gamma)$ the set of statement variables that occur in some formula of Γ. In other words, $SV(\Gamma) = \bigcup\{SV(\varphi) \mid \varphi \in \Gamma\}$.

Definition 2.2.1 uses infix notation for connective symbols in formulas. This is possible since we use connective symbols of arity at most two. Later, when we wish to consider connective symbols of arbitrary arities, we will use prefix notation for connective symbols, by treating formulas as terms over an appropriate signature. We chose to use infix notation because this most common method yields easily readable formulas. However, this choice necessitates a distinct proof of unique readability.

Theorem 2.2.9. *For every formula $\varphi \in \mathrm{PLFORM}$, we have $|\varphi|_(= |\varphi|_)$. Furthermore, if w is a proper prefix of a formula, then $|w|_(> |w|_)$ and if w is a proper suffix of a formula, then $|w|_(< |w|_)$.*

Proof. To prove the first part of the theorem, we use induction on φ. If φ is a statement variable, then $|\varphi|_(= |\varphi|_) = 0$. Suppose now that $\varphi = (\neg\psi)$, where $|\psi|_(= |\psi|_)$. Clearly, we can write

$$|\varphi|_(= 1 + |\psi|_(= 1 + |\psi|_) = |\varphi|_).$$

Assume that $\varphi = (\psi C \theta)$, where C is one of the connective symbols $\vee, \wedge. \rightarrow$, or $\leftrightarrow$ and that $|\psi|_(= |\psi|_)$, $|\theta|_(= |\theta|_)$. By the inductive hypothesis, we have

$$|\varphi|_(= 1 + |\psi|_(+ |\theta|_(= 1 + |\psi|_) + |\theta|_) = |\varphi|_).$$

This concludes the argument for the first part.

Again, the argument for the second part of the theorem is by induction on φ. If φ is a statement variable, then it has no proper prefixes and the statement is vacuously true.

Consider a formula $\varphi = (\neg\psi)$, where $|w|_(> |w|_)$ for every proper prefix w of ψ. If u is a proper prefix of φ, then one of the cases shown in the following must hold, where w represents a proper prefix of ψ:

Proper prefix u	$\|u\|_($	$\|u\|_)$
$($	1	0
$(\neg$	1	0
$(\neg w$	$1 + \|w\|_($	$\|w\|_)$
$(\neg\psi$	$1 + \|\psi\|_($	$\|\psi\|_)$

By the inductive hypothesis and the first part of the theorem, we obtain $|u|_(> |u|_)$ in all cases.

Let now φ be $(\psi C\theta)$, where ψ and θ are formulas of propositional logic such that for every proper prefix v of ψ and for every proper prefix w of θ, we have $|v|_(> |v|_)$ and $|w|_(> |w|_)$, respectively. If u is a proper prefix of φ, one of the following cases must hold, where v is a proper prefix of ψ and w is a prefix of θ:

Proper prefix u	$\|u\|_($	$\|u\|_)$
$($	1	0
$(v$	$1 + \|v\|_($	$\|v\|_)$
$(\psi$	$1 + \|\psi\|_($	$\|\psi\|_)$
$(\psi C$	$1 + \|\psi\|_($	$\|\psi\|_)$
$(\psi C w$	$1 + \|\psi\|_(+ \|w\|_($	$\|\psi\|_) + \|w\|_)$
$(\psi C\theta$	$1 + \|\psi\|_(+ \|\theta\|_($	$\|\psi\|_) + \|\theta\|_)$

It is easy to see, by the inductive hypothesis and the first part of the theorem, that in each case $|u|_(> |u|_)$.

If w is a proper suffix of a formula φ, then we have $\varphi = uw$, where u is a proper prefix of φ. Using the first two parts of the theorem, we can write $|w|_) = |\varphi|_) - |u|_) > |\varphi|_(- |u|_(= |w|_($. □

Corollary 2.2.10. *No proper prefix of a formula of propositional logic is a suffix of a formula of propositional logic and therefore, no proper prefix of a formula of propositional logic is a formula.*

Proof. The proof is an immediate consequence of the previous theorem. □

Theorem 2.2.11. *Definition 2.2.1 of the set of formulas of propositional logic satisfies the unique readability condition.*

Proof. Note that if a formula φ enters the set PLFORM by Rule 1 of Definition 2.2.1, then φ is a variable. Otherwise, that is, if φ enters PLFORM by Rules 2 or 3, then φ begins with a left parenthesis.

This shows that a formula that enters PLFORM by Rule 1 may not be put there by any other rule.

Let φ be a formula that enters PLFORM by Rule 2. We have $\varphi = (\neg\psi)$ for some formula ψ. Observe that in this case φ may not be put in PLFORM by any of the Rules 3_C. Indeed, if φ enters PLFORM due to a Rule 3_C, we have $(\neg\psi) = (\alpha C \beta)$, so α begins with $\neg$, which is impossible. On the other hand, the formula ψ is uniquely determined, which shows that the unique readability is satisfied in this case.

Consider now a formula φ that enters PLFORM because of a Rule 3_C, that is, $\varphi = (\psi C \theta)$. Suppose that φ can also be obtained by applying another Rule $3_{C'}$. This implies $\varphi = (\psi C \theta) = (\psi' C' \theta')$. If $\psi \neq \psi'$, then either ψ is a proper prefix of ψ' or ψ' is a proper prefix of ψ. Neither case is possible because of Corollary 2.2.10, so $\psi = \psi'$. This implies $C = C'$ and $\theta = \theta'$ which means that in this last case the unique readability condition is satisfied. □

Theorem 2.2.12 (Occurrence Theorem for Propositional Logic). *Let φ, ψ, α be formulas. If $\alpha \neq (\neg\varphi)$, then every occurrence of α in $(\neg\varphi)$ is part of φ. (More exactly, each occurrence of α in $(\neg\varphi)$ is part of the occurrence $(\varphi, 2)$.)*

Let C be a binary connective symbol. If $\alpha \neq (\varphi C \psi)$, then every occurrence of α in $(\varphi C \psi)$ is either part of φ or part of ψ. (More exactly, each occurrence of α in $(\varphi C \psi)$ is either a part of $(\varphi, 1)$ or a part of $(\psi, |\varphi| + 2)$.)

Proof. We show only the second part of the theorem. Let (α, j) be an occurrence of α in $(\varphi C \psi)$. We have $j \neq 0$ because $\alpha \neq (\varphi C \psi)$ and no formula can be a proper prefix of another formula. Since no formula starts with a connective or a close parenthesis, the occurrence of α in $(\varphi C \psi)$ must begin either within φ or ψ, i.e., $1 \leq j \leq |\varphi|$ or $|\varphi| + 2 \leq j \leq |\varphi| + |\psi| + 1$. If the occurrence extends beyond the end of φ, in the first case, or the end of ψ, in the second case, this would imply that a suffix of a formula is a proper prefix of another formula, thus contradicting Corollary 2.2.10. □

Definition 2.2.13. Let φ be a formula. A *subformula* of φ is a formula ψ that is a subsequence of φ. If ψ is a subformula of φ and $\psi \neq \varphi$, then ψ is a *proper subformula* of φ.

If $\varphi = (\neg\psi)$, then we call ψ the *immediate subformula* of φ; similarly, if $\varphi = (\psi C \theta)$ for some binary connective symbol C, then we refer to ψ and θ as the *immediate subformulas* of φ.

The sets of subformulas and proper subformulas of a formula φ are denoted by $\mathrm{SUBF}(\varphi)$ and $\mathrm{PRSUBF}(\varphi)$, respectively.

If Γ is a set of formulas, then $\mathrm{SUBF}(\Gamma) = \bigcup\{\mathrm{SUBF}(\varphi) \mid \varphi \in \Gamma\}$ and $\mathrm{PRSUBF}(\Gamma) = \bigcup\{\mathrm{PRSUBF}(\varphi) \mid \varphi \in \Gamma\}$. $\square$

Observe that $\mathrm{SUBF}(\mathrm{SUBF}(\Gamma)) = \mathrm{SUBF}(\Gamma)$ for every set Γ of formulas.

We will now define the set $U(\Gamma)$ for every set of formulas Γ. The idea is that $\Gamma \cup U(\Gamma)$ will be the "analytical universe" of Γ, i.e., the set of formulas that can be obtained by analyzing the structure of formulas in Γ.

Theorem 2.2.14. *Let* $U : \mathcal{P}(\mathrm{PLFORM}) \longrightarrow \mathcal{P}(\mathrm{PLFORM})$ *be the function defined by*

$$U(\Gamma) = \mathrm{PRSUBF}(\Gamma) \cup \{(\neg\psi) \mid \psi \in \mathrm{PRSUBF}(\Gamma)\},$$

for every $\Gamma \subseteq \mathrm{PLFORM}$. *Then,*

(1) *for every* $\Gamma, \Gamma' \subseteq \mathrm{PLFORM}$, $U(\Gamma \cup \Gamma') = U(\Gamma) \cup U(\Gamma')$,
(2) *for every* $k \geq 1$, $U^k(\Gamma) \subseteq U(\Gamma)$.

Proof. We leave the argument to the reader. $\square$

Note that the first property of the theorem implies monotonicity of U. In other words, if $\Gamma_0 \subseteq \Gamma_1$, then $U(\Gamma_0) \subseteq U(\Gamma_1)$, for all $\Gamma_0, \Gamma_1 \subseteq$ PLFORM.

The notation $\texttt{replace}\,(\varphi, (\psi, i), \alpha)$ used in the following theorem was introduced in Section 1.2.

Theorem 2.2.15. *Let* φ, ψ, α *be formulas. If* (ψ, i) *is an occurrence of* ψ *in* φ*, then* $\texttt{replace}\,(\varphi, (\psi, i), \alpha)$ *is a formula.*

Proof. As is the case for similar theorems, the argument is by induction on the definition of formulas. If φ is a statement variable, then we have $\varphi = \psi = p$, where $p \in SV$ and $i = 0$. Clearly, in this case, we have $\texttt{replace}\,(\varphi, (\psi, i), \alpha) = \alpha$ and the statement of the theorem is true.

Suppose now that $\varphi = (\neg\varphi_0)$ and the statement is valid for φ_0. If $\psi = \varphi$, then $i = 0$, $\texttt{replace}\,(\varphi, (\psi, i), \alpha) = \alpha$, and the statement of the theorem is trivially true. Therefore, let us assume that $\psi \neq \varphi$. Theorem 2.2.12 implies that the occurrence (ψ, i) in φ is a part of the occurrence $(\varphi_0, 2)$ and we have

$$\texttt{replace}\,(\varphi, (\psi, i), \alpha) = (\neg\texttt{replace}\,(\varphi_0, (\psi, i-2), \alpha)).$$

By the inductive hypothesis, $\texttt{replace}\,(\varphi_0, (\psi, i-2), \alpha))$ is a formula and this shows that $\texttt{replace}\,(\varphi, (\psi, i), \alpha)$ is a formula.

Let $\varphi = (\varphi_0 C \varphi_1)$ be a formula, where C is a binary connective symbol and suppose that the statement holds for φ_0 and φ_1. If $\psi = \varphi$, then $i = 0$ and $\texttt{replace}\,(\varphi, (\psi, i), \alpha) = \alpha$, so the statement is true. If $\psi \neq \varphi$, then, by Theorem 2.2.12, we have one of the following cases:

(1) The occurrence (ψ, i) is a part of the occurrence $(\varphi_0, 1)$. In this case, we can write

$$\texttt{replace}\,(\varphi, (\psi, i), \alpha) = (\texttt{replace}\,(\varphi_0, (\psi, i-1), \alpha) C \varphi_1).$$

By the inductive hypothesis, $\texttt{replace}\,(\varphi_0, (\psi, i-1), \alpha)$ is a formula and this implies that $\texttt{replace}\,(\varphi, (\psi, i), \alpha)$ is a formula.

(2) The occurrence (ψ, i) is a part of the occurrence $(\varphi_1, |\varphi_0| + 2)$. We have

$$\texttt{replace}\,(\varphi, (\psi, i), \alpha) = (\varphi_0 C\, \texttt{replace}\,(\varphi_1, (\psi, i - (|\varphi_0| + 2)), \alpha)).$$

Again, by the inductive hypothesis, $\texttt{replace}\,(\varphi_1, (\psi, i - (|\varphi_0| + 2)), \alpha)$ is a formula and this implies that $\texttt{replace}\,(\varphi, (\psi, i), \alpha)$ is a formula.

$\square$

We introduce a notation for certain formulas that needs fewer parentheses than the standard notation. Let $\mathrm{DISJ}, \mathrm{CONJ} : \mathrm{Seq}^+(\mathrm{PLFORM}) \longrightarrow \mathrm{PLFORM}$ be defined by

$$\mathrm{DISJ}(\varphi_0) = \varphi_0,$$
$$\mathrm{DISJ}(\varphi_0, \ldots, \varphi_n) = (\mathrm{DISJ}(\varphi_0, \ldots, \varphi_{n-1}) \vee \varphi_n)$$

and

$$\mathrm{CONJ}(\varphi_0) = \varphi_0,$$
$$\mathrm{CONJ}(\varphi_0, \ldots, \varphi_n) = (\mathrm{CONJ}(\varphi_0, \ldots, \varphi_{n-1}) \wedge \varphi_n).$$

The formulas $\mathrm{DISJ}(\varphi_0, \ldots, \varphi_{n-1})$ and $\mathrm{CONJ}(\varphi_0, \ldots, \varphi_{n-1})$ will be denoted by $(\varphi_0 \vee \cdots \vee \varphi_{n-1})$ and $(\varphi_0 \wedge \cdots \wedge \varphi_{n-1})$, respectively. We shall also use the notations $\bigvee_{0 \leq i \leq n-1} \varphi_i$ and

$\bigwedge_{0 \le i \le n-1} \varphi_i$, respectively. We shall refer to $\alpha = \bigvee_{0 \le i \le n-1} \varphi_i$ as the *disjunction* of the sequence $(\varphi_0, \dots, \varphi_{n-1})$ and to $\beta = \bigwedge_{0 \le i \le n-1} \varphi_i$ as the *conjunction* of the same sequence of formulas.

Note that a formula can be the conjunction or disjunction on several sequences of formulas. For example, the sequences $(\varphi_0, \varphi_1, \varphi_2)$, $((\varphi_0 \wedge \varphi_1), \varphi_2)$, and $(((\varphi_0 \wedge \varphi_1) \wedge \varphi_2))$ have the same conjunction. However, we do have the following limited unique readability result.

Theorem 2.2.16. *If $(\varphi_0, \dots, \varphi_{n-1})$, $(\psi_0, \dots, \psi_{m-1})$ are two sequences of formulas that have the same conjunction (disjunction) and neither φ_0 nor ψ_0 is the conjunction (disjunction) of two formulas, then $n = m$ and $\varphi_i = \psi_i$ for $0 \le i \le n-1$.*

Proof. We discuss only the case of conjunctions. The argument is by induction on n. For the basis step, $n = 1$, we suppose that $\varphi_0 = (\psi_0 \wedge \cdots \wedge \psi_{m-1})$. If $m > 1$, φ_0 would be the conjunction of two formulas, which contradicts the hypothesis. Therefore, $m = 1 = n$ and so $\varphi_0 = \psi_0$.

Suppose now that the result is true for $n-1$ where $n \ge 2$ and that $(\varphi_0 \wedge \cdots \wedge \varphi_{n-1}) = (\psi_0 \wedge \cdots \wedge \psi_{m-1})$. If m were 1, the right side would be reduced to ψ_0 and thus, ψ_0 would be a conjunction of two formulas, contrary to hypothesis. The fact that both n and m are at least 2 allows us to write

$$((\varphi_0 \wedge \cdots \wedge \varphi_{n-2}) \wedge \varphi_{n-1}) = ((\psi_0 \wedge \cdots \wedge \psi_{m-2}) \wedge \psi_{m-1}).$$

By unique readability, we have $(\varphi_0 \wedge \cdots \wedge \varphi_{n-2}) = (\psi_0 \wedge \cdots \wedge \psi_{m-2})$ and $\varphi_{n-1} = \psi_{m-1}$. By the inductive hypothesis, we have $n-1 = m-1$ and $\varphi_i = \psi_i$ for $0 \le i \le n-2$. $\square$

We will refer to the formulas $\varphi_0, \dots, \varphi_{n-1}$ as the *disjuncts* of $(\varphi_0 \vee \cdots \vee \varphi_{n-1})$ and as the *conjuncts* of $(\varphi_0 \vee \cdots \vee \varphi_{n-1})$. The discussion preceding Theorem 2.2.16 shows that these notions are ambiguous. In practice, however, we will use these terms only when the context makes it clear which formulas are meant.

Observe that for $n = 1$, both the notations $(\varphi_0 \vee \cdots \vee \varphi_{n-1})$ and $(\varphi_0 \wedge \cdots \wedge \varphi_{n-1})$ become (φ_0). This ambiguity is harmless since, for $n = 1$ both $(\varphi_0 \vee \cdots \vee \varphi_{n-1})$ and $(\varphi_0 \wedge \cdots \wedge \varphi_{n-1})$ are notations for φ_0.

2.3 Truth Assignments

We want to specify how to assign a meaning to a formula of propositional logic in a certain context, so we must specify what a meaning is and what the context is in which we can assign such a meaning. It is easier to specify a meaning. For us, a meaning will be one of the two symbols **T** and **F**, called *truth values*. The symbols **T** and **F** can represent any two distinct objects; a common choice is for **T** to be 1 and for **F** to be 0, but any other choice would do. We denote $\{\mathbf{T}, \mathbf{F}\}$ by **Bool**. Our idea, of course, is that **T** stands for true and **F** stands for false.

We will also use the notation $\overline{b}$ defined as

$$\overline{b} = \begin{cases} \mathbf{T} & \text{if } b = \mathbf{F}, \\ \mathbf{F} & \text{if } b = \mathbf{T}. \end{cases}$$

A context in which we can assign a meaning to a formula is a mapping from SV to **Bool**.

Definition 2.3.1. A *truth assignment* is a mapping $v : SV \longrightarrow$ **Bool**. ▯

We will denote by TA the set of all truth assignments $SV \longrightarrow$ **Bool**.

The mathematical equivalent of a connective is a truth function.

Definition 2.3.2. An *n-ary truth function* is a function f : $\mathbf{Bool}^n \longrightarrow \mathbf{Bool}$, for $n \in \mathbf{N}$; 1-ary truth functions are called *unary truth functions*, and 2-ary truth functions are called *binary truth functions.* ▯

The connectives conjunction, disjunction, conditional, biconditional, and negation correspond to the truth functions $f_\wedge$, $f_\vee$, $f_\rightarrow$, $f_\leftrightarrow$, and $f_\neg$ given by the following tables:

a	b	$f_\wedge(a,b)$	$f_\vee(a,b)$	$f_\rightarrow(a,b)$	$f_\leftrightarrow(a,b)$
F	**F**	**F**	**F**	**T**	**T**
F	**T**	**F**	**T**	**T**	**F**
T	**F**	**F**	**T**	**F**	**F**
T	**T**	**T**	**T**	**T**	**T**

a	$f_\neg(a)$
F	**T**
T	**F**

Definition 2.3.3. Let v be a truth assignment. The *truth valuation generated by* v is the function $\bar{v}$: PLFORM $\longrightarrow$ **Bool** given by the following recursive definition:

$$\begin{aligned}
\bar{v}(q) &= v(q) \text{ for each statement variable } q,\\
\bar{v}((\neg\varphi)) &= f_\neg(\bar{v}(\varphi)),\\
\bar{v}((\varphi \wedge \psi)) &= f_\wedge(\bar{v}(\varphi), \bar{v}(\psi)),\\
\bar{v}((\varphi \vee \psi)) &= f_\vee(\bar{v}(\varphi), \bar{v}(\psi)),\\
\bar{v}((\varphi \rightarrow \psi)) &= f_\rightarrow(\bar{v}(\varphi), \bar{v}(\psi)),\\
\bar{v}((\varphi \leftrightarrow \psi)) &= f_\leftrightarrow(\bar{v}(\varphi), \bar{v}(\psi)).
\end{aligned}$$

$\square$

Theorem 2.2.11 ensures that there is a unique function $\bar{v}$ satisfying the conditions of the above definition. To simplify notation, we will often write $v(\varphi)$ instead of $\bar{v}(\varphi)$. We will refer to $v(\varphi)$ as the *truth value of* φ *under the assignment* v. If $v(\varphi) = \mathbf{T}$, then we say that v *satisfies* φ. If Γ is a set of formulas and v satisfies every formula in Γ, then we say that v satisfies Γ.

Truth valuations can be defined independently of the notion of truth assignment.

Definition 2.3.4. A *truth valuation* is a mapping w : PLFORM $\longrightarrow$ **Bool** which satisfies the following conditions:

$$\begin{aligned}
w((\neg\varphi)) &= f_\neg(w(\varphi)),\\
w((\varphi \wedge \psi)) &= f_\wedge(w(\varphi), w(\psi)),\\
w((\varphi \vee \psi)) &= f_\vee(w(\varphi), w(\psi)),\\
w((\varphi \rightarrow \psi)) &= f_\rightarrow(w(\varphi), w(\psi)),\\
w((\varphi \leftrightarrow \psi)) &= f_\leftrightarrow(w(\varphi), w(\psi)).
\end{aligned}$$

$\square$

We show now that there are no truth valuations except for the ones generated by truth assignments.

Theorem 2.3.5. *A function w : PLFORM $\longrightarrow$ **Bool** is a truth valuation if and only if there is a truth assignment v such that $w = \bar{v}$. Further, if w is a truth valuation, then there is a unique truth assignment v with $w = \bar{v}$.*

Proof. It is obvious, by Definition 2.3.3, that if v is a truth assignment, then $\bar{v}$ is a truth valuation. Conversely, let w be a truth valuation. Define the truth assignment v by $v(p) = w(p)$ for every variable p. Then, it is easy to show by induction on formulas that $w = \bar{v}$. The proof of the uniqueness of the truth assignment v is left to the reader. $\square$

Theorem 2.3.6. *For every truth assignment v and nonempty sequence of formulas $(\varphi_0, \ldots, \varphi_{n-1})$, we have*

$$v\left(\bigvee_{0\le i\le n-1} \varphi_i\right) = \begin{cases} \mathbf{T} & \text{if } v(\varphi_i) = \mathbf{T} \ \text{ for some } i, 0 \le i \le n-1 \\ \mathbf{F} & \text{otherwise} \end{cases}$$

and

$$v\left(\bigwedge_{0\le i\le n-1} \varphi_i\right) = \begin{cases} \mathbf{T} & \text{if } v(\varphi_i) = \mathbf{T} \ \text{ for every } i, 0 \le i \le n-1 \\ \mathbf{F} & \text{otherwise.} \end{cases}$$

Proof. The argument is by induction on n and is left to the reader. $\square$

Definition 2.3.7. Let φ and ψ be two formulas and let Γ be a set of formulas:

- φ is a *tautology* (denoted $\models \varphi$) if $v(\varphi) = \mathbf{T}$ for every truth assignment v.
- φ is *satisfiable* if there is some truth assignment v such that $v(\varphi) = \mathbf{T}$. Otherwise, φ is said to be *unsatisfiable*.
- φ is a *contradiction* if $v(\varphi) = \mathbf{F}$ for every truth assignment v.
- φ *logically implies* ψ (denoted by $\varphi \models \psi$) if for every truth assignment v, $v(\varphi) = \mathbf{T}$ implies $v(\psi) = \mathbf{T}$.
- φ is *logically equivalent* to ψ (denoted by $\varphi \equiv \psi$) if $v(\varphi) = v(\psi)$ for every truth assignment v.

- Γ *logically implies* ψ (denoted by $\Gamma \models \psi$) if every truth assignment that satisfies Γ also satisfies φ.
- Γ is *satisfiable* if there is a truth assignment v which satisfies every formula in Γ. Otherwise, we say that Γ is *unsatisfiable*.

$\square$

Example 2.3.8. Let $\varphi = ((p_1 \to p_2) \to ((p_0 \vee p_1) \to p_2))$, and let v be a truth assignment such that $v(p_0) = v(p_1) = v(p_2) = \mathbf{F}$. Applying Definition 2.3.3, we have $v(\varphi) = f_{\to}(v((p_1 \to p_2)), v(((p_0 \vee p_1) \to p_2)))$. In turn, $v((p_1 \to p_2)) = f_{\to}(v(p_1), v(p_2)) = f_{\to}(\mathbf{F}, \mathbf{F}) = \mathbf{T}$, and

$$\begin{aligned} v(((p_0 \vee p_1) \to p_2)) &= f_{\to}(f_{\vee}(v(p_0), v(p_1)), v(p_2)) \\ &= f_{\to}(f_{\vee}(\mathbf{F}, \mathbf{F}), \mathbf{F}) = f_{\to}(\mathbf{F}, \mathbf{F}) = \mathbf{T}. \end{aligned}$$

Thus, $v(\varphi) = f_{\to}(\mathbf{T}, \mathbf{T}) = \mathbf{T}$.

A similar analysis can be carried out for other choices of values of $v(p_0), v(p_1), v(p_2)$. Since there are two choices ($\mathbf{F}$ and $\mathbf{T}$) for each $v(p_i)$, we need to consider the following eight cases in order to compute $v(\varphi)$ for any truth assignment v:

$v(p_0)$	$v(p_1)$	$v(p_2)$	$v((p_1 \to p_2))$	$v(((p_0 \vee p_1) \to p_2))$	$v(\varphi)$
F	**F**	**F**	**T**	**T**	**T**
F	**F**	**T**	**T**	**T**	**T**
F	**T**	**F**	**F**	**F**	**T**
F	**T**	**T**	**T**	**T**	**T**
T	**F**	**F**	**T**	**F**	**F**
T	**F**	**T**	**T**	**T**	**T**
T	**T**	**F**	**F**	**F**	**T**
T	**T**	**T**	**T**	**T**	**T**

This shows that φ is not a tautology; however, φ is satisfiable.

Consider now the formula

$$\psi = ((p_0 \to p_2) \to ((p_1 \to p_2) \to ((p_0 \vee p_1) \to p_2))).$$

We claim that ψ is a tautology. Clearly, we can write $\psi = ((p_0 \to p_2) \to \varphi)$, where φ is the previous formula. The eight possible cases are enclosed in the following table:

$v(p_0)$	$v(p_1)$	$v(p_2)$	$v((p_0 \to p_2))$	$v(\varphi)$	$v(\psi)$
F	**F**	**F**	**T**	**T**	**T**
F	**F**	**T**	**T**	**T**	**T**
F	**T**	**F**	**T**	**T**	**T**
F	**T**	**T**	**T**	**T**	**T**
T	**F**	**F**	**F**	**F**	**T**
T	**F**	**T**	**T**	**T**	**T**
T	**T**	**F**	**F**	**T**	**T**
T	**T**	**T**	**T**	**T**	**T**

This confirms that ψ is a tautology. $\square$

Example 2.3.9. Let φ be a formula. For a truth assignment v, we have $v(\varphi) = \mathbf{T}$ if and only if $v((\neg\varphi)) = \mathbf{F}$. Also, $(\varphi \vee (\neg\varphi))$ is a tautology and $(\varphi \wedge (\neg\varphi))$ is a contradiction.

The first statement follows immediately from $v((\neg\varphi)) = f_\neg(v(\varphi))$ and the definition of the function $f_\neg$.

For the second part, there are two possible cases, depending on $v(\varphi)$:

$v(\varphi)$	$v((\neg\varphi))$	$v((\varphi \vee (\neg\varphi)))$	$v((\varphi \wedge (\neg\varphi)))$
F	**T**	**T**	**F**
T	**F**	**T**	**F**

For every truth assignment v, $v((\varphi \vee (\neg\varphi))) = \mathbf{T}$ and $v((\varphi \wedge (\neg\varphi))) = \mathbf{F}$. $\square$

Example 2.3.10. Let p be a statement variable. Example 2.3.9 implies that $(p \vee (\neg p))$ is a tautology and $(p \wedge (\neg p))$ is a contradiction.

On the other hand, if p, q are two distinct statement variables, then the formula $(p \wedge (\neg q))$ is satisfiable. Indeed, since $p \neq q$, there is a truth assignment v such that $v(p) = \mathbf{T}$ and $v(q) = \mathbf{F}$. This gives $v((\neg q)) = \mathbf{T}$ and

$$v(p \wedge (\neg q)) = f_\wedge(v(p), v((\neg q))) = f_\wedge(\mathbf{T}, \mathbf{T}) = \mathbf{T}.$$

$\square$

Example 2.3.11. The formula $(\varphi \to (\psi \to \varphi))$ is a tautology for any formulas φ, ψ. In order to justify this claim, note that for a truth assignment v there are four cases to consider depending on the values of $v(\varphi)$ and $v(\psi)$:

$v(\varphi)$	$v(\psi)$	$v((\psi \to \varphi))$	$v((\varphi \to (\psi \to \varphi)))$
F	**F**	**T**	**T**
F	**T**	**F**	**T**
T	**F**	**T**	**T**
T	**T**	**T**	**T**

Clearly, in all cases, $v((\varphi \to (\psi \to \varphi))) = \mathbf{T}$.

Consider a similar formula, $((\varphi \to \psi) \to \varphi)$. The four cases that occur for this formula are summarized in the following:

$v(\varphi)$	$v(\psi)$	$v((\varphi \to \psi))$	$v(((\varphi \to \psi) \to \varphi))$
F	**F**	**T**	**F**
F	**T**	**T**	**F**
T	**F**	**F**	**T**
T	**T**	**T**	**T**

Note that this formula is logically equivalent to φ. Of course, if φ is not a tautology, then neither is $((\varphi \to \psi) \to \varphi)$. □

Theorem 2.3.12. *Let φ, ψ, and θ be formulas. Then,*

(1) *(a)* $\varphi \models \varphi$,
(b) if $\varphi \models \psi$ *and* $\psi \models \theta$*, then* $\varphi \models \theta$,
(2) *(a)* $\varphi \equiv \varphi$,
(b) if $\varphi \equiv \psi$*, then* $\psi \equiv \varphi$,
(c) if $\varphi \equiv \psi$ *and* $\psi \equiv \theta$*, then* $\varphi \equiv \theta$,
(3) $\varphi \equiv \psi$ *if and only if* $\varphi \models \psi$ *and* $\psi \models \varphi$.

Proof. The arguments for all the parts are straightforward and are left to the reader. □

Theorem 2.3.13. *If* $\varphi_0, \varphi_1, \psi_0, \psi_1$ *are formulas such that* $\varphi_0 \models \varphi_1$ *and* $\psi_0 \models \psi_1$*, then*

$$(\neg\varphi_1) \models (\neg\varphi_0),$$

$$(\varphi_0 \vee \psi_0) \models (\varphi_1 \vee \psi_1),$$

$$(\varphi_0 \wedge \psi_0) \models (\varphi_1 \wedge \psi_1),$$

$$(\varphi_1 \to \psi_0) \models (\varphi_0 \to \psi_1).$$

Proof. We provide the argument only for the last part of the theorem. Suppose that v is a truth assignment such that $v((\varphi_1 \to \psi_0)) = \mathbf{T}$. We distinguish two cases:

`Case 1:` $v(\varphi_0) = \mathbf{F}$. Then, $v((\varphi_0 \to \psi_1)) = \mathbf{T}$.

`Case 2:` $v(\varphi_0) = \mathbf{T}$. Since $\varphi_0 \models \varphi_1$, we have $v(\varphi_1) = \mathbf{T}$. Thus, $v(\psi_0) = \mathbf{T}$ because $v((\varphi_1 \to \psi_0)) = \mathbf{T}$. This allows us to conclude that $v(\psi_1) = \mathbf{T}$, so $v((\varphi_0 \to \psi_1)) = \mathbf{T}$. □

Theorem 2.3.14. *If $\varphi, \varphi', \psi, \psi'$ are formulas such that $\varphi \equiv \varphi'$ and $\psi \equiv \psi'$, then*

$$(\neg\varphi) \equiv (\neg\varphi'),$$
$$(\varphi \vee \psi) \equiv (\varphi' \vee \psi'),$$
$$(\varphi \wedge \psi) \equiv (\varphi' \wedge \psi'),$$
$$(\varphi \to \psi) \equiv (\varphi' \to \psi'),$$
$$(\varphi \leftrightarrow \psi) \equiv (\varphi' \leftrightarrow \psi').$$

Proof. The statements of the theorem (except for the fifth one) are immediate consequences of Theorem 2.3.13. We can also give a direct argument for the last four logical equivalences. Let C be a binary connective symbol and let v be a truth assignment. Then, we have

$$v((\varphi C \psi)) = f_C(v(\varphi), v(\psi)) = f_C(v(\varphi'), v(\psi')) = v((\varphi' C \psi')),$$

which shows that $(\varphi C \psi) \equiv (\varphi' C \psi')$.

We leave for the reader the proof of the first logical equivalence of the theorem. □

Theorem 2.3.15. *Let φ, ψ, and θ be formulas and let Γ, Γ' be sets of formulas. Then,*

(1) $\emptyset \models \varphi$ *if and only if* $\models \varphi$,
(2) $\{\varphi\} \models \psi$ *if and only if* $\varphi \models \psi$,
(3) *if* Γ' *is satisfiable and* $\Gamma \subseteq \Gamma'$, *then* Γ *is satisfiable,*
(4) $\{\varphi\}$ *is satisfiable if and only if* φ *is satisfiable,*
(5) *if* $\Gamma \models \varphi$ *and* $\Gamma \subseteq \Gamma'$, *then* $\Gamma' \models \varphi$.

Proof. Again, we leave the simple arguments of the theorem to the reader. □

Lemma 2.3.16. *If* $\varphi, \psi \in$ PLFORM *and* v *is a truth assignment such that* $v(\varphi) = \mathbf{T}$ *and* $v((\varphi \to \psi)) = \mathbf{T}$*, then* $v(\psi) = \mathbf{T}$*.*

Proof. Indeed, we have

$$\mathbf{T} = v((\varphi \to \psi)) = f_{\to}(v(\varphi), v(\psi)) = f_{\to}(\mathbf{T}, v(\psi)).$$

According to the definition of function $f_{\to}$, this may happen only if $v(\psi) = \mathbf{T}$. □

Theorem 2.3.17. *Let* φ *and* ψ *be formulas and let* Γ *be a set of formulas. Then,*

(1) $\Gamma \models \varphi$ *if and only if* $\Gamma \cup \{\neg\varphi\}$ *is unsatisfiable,*
(2) *if* $\Gamma \models \varphi$ *and* $\Gamma \models (\varphi \to \psi)$*, then* $\Gamma \models \psi$*,*
(3) $\Gamma \cup \{\varphi\} \models \psi$ *if and only if* $\Gamma \models (\varphi \to \psi)$*.*

Proof. The argument is left to the reader. □

Theorem 2.3.18. *Let* Γ *be a set of formulas. The following statements are equivalent:*

(1) Γ *is unsatisfiable.*
(2) $\Gamma \models \varphi$ *for every formula* φ*.*
(3) $\Gamma \models \varphi$ *for every contradiction* φ*.*
(4) $\Gamma \models \varphi$ *for some contradiction* φ*.*

Proof. `(1) implies (2).` Let Γ be unsatisfiable. Since there is no truth assignment that satisfies Γ, $\Gamma \models \varphi$ holds vacuously for every formula φ.

It is immediate that (2) implies (3) and that (3) implies (4).

`(4) implies (1).` Let Γ be such that $\Gamma \models \varphi$ for some contradiction φ. No truth assignment could satisfy Γbecause any such truth assignment would have to satisfy φ. □

Theorem 2.3.19. *Let* $\Gamma = \{\varphi_0, \ldots, \varphi_{n-1}\}$ *be a nonempty, finite set of formulas. Then,* Γ *is unsatisfiable if and only if* $((\neg\varphi_0) \vee \cdots \vee (\neg\varphi_{n-1}))$ *is a tautology.*

Proof. The argument is straightforward and is left to the reader. □

A set Γ of formulas is *closed* if it contains both φ and $(\neg\varphi)$ for some formula φ. Clearly, any closed set of formulas is unsatisfiable, though an unsatisfiable set of formulas need not be closed.

Theorem 2.3.20. *Let φ, ψ be two formulas. Then, we have*

(1) *φ is a satisfiable if and only if $(\neg\varphi)$ is not a tautology,*
(2) *φ is a contradiction if and only if $(\neg\varphi)$ is a tautology,*
(3) *$\varphi \models \psi$ if and only if $(\varphi \to \psi)$ is a tautology,*
(4) *$\varphi \equiv \psi$ if and only if $(\varphi \leftrightarrow \psi)$ is a tautology.*

Proof. The first two parts of the theorem are straightforward and their proof is omitted.

The third part follows from Part 3 of Theorem 2.3.17 with $\Gamma = \emptyset$.

To prove the fourth part of the theorem, assume that $\varphi \equiv \psi$ and let v be an arbitrary truth assignment. We have

$$v((\varphi \leftrightarrow \psi)) = f_{\leftrightarrow}(v(\varphi), v(\psi)).$$

Since $v(\varphi) = v(\psi)$, according to the definition of $f_{\leftrightarrow}$, $f_{\leftrightarrow}(v(\varphi), v(\psi)) = \mathbf{T}$, hence $v((\varphi \leftrightarrow \psi)) = \mathbf{T}$ for every truth assignment v, so $(\varphi \leftrightarrow \psi)$ is a tautology.

Conversely, let φ, ψ be formulas such that $(\varphi \leftrightarrow \psi)$ is a tautology, that is, $v((\varphi \leftrightarrow \psi)) = \mathbf{T}$ for every v. We have $v((\varphi \leftrightarrow \psi)) = f_{\leftrightarrow}(v(\varphi), v(\psi)) = \mathbf{T}$. Since $f_{\leftrightarrow}(w, w') = \mathbf{T}$ if and only if $w = w'$ for $w, w' \in \mathbf{Bool}$, we have $v(\varphi) = v(\psi)$ for every truth assignment v, so $\varphi \equiv \psi$. $\square$

Corollary 2.3.21. *The formula $(\varphi \leftrightarrow \psi)$ is a tautology if and only if both formulas $(\varphi \to \psi)$ and $(\psi \to \varphi)$ are tautologies.*

Proof. The statement of the corollary follows immediately from the previous theorem and from Part 3 of Theorem 2.3.12. $\square$

The following theorem shows that the value of $\bar{v}(\varphi)$ is determined by the values of $v(p)$ for $p \in SV(\varphi)$.

Theorem 2.3.22 (Agreement Theorem for Propositional Logic). *Let $v, w \in TA$ and let φ be a formula such that $v(p) = w(p)$ for all $p \in SV(\varphi)$. Then $\bar{v}(\varphi) = \bar{w}(\varphi)$.*

Proof. The argument, by induction on φ, uses Exercise 1 and is left to the reader. $\square$

Definition 2.3.23. A *partial truth assignment* is a partial function from SV to **Bool**, i.e., a member of the set $SV \rightsquigarrow \mathbf{Bool}$.

If $S \subseteq SV$, then a *truth assignment over* S is a partial truth assignment whose domain is S.

The set $S \longrightarrow \mathbf{Bool}$ of truth assignments over S will be denoted by TA_S. □

Note that if S is a finite set of statement variables, then $|\mathrm{TA}_S| = 2^{|S|}$.

Let φ be a formula and v be a partial truth assignment such that $SV(\varphi) \subseteq \mathrm{Dom}(v)$. The Agreement Theorem shows that if $w_1, w_2 \in$ TA both extend v, then $w_1(\varphi) = w_2(\varphi)$. This allows us to make the following definition.

Definition 2.3.24. Let v be a partial truth assignment and let φ be a formula such that $SV(\varphi) \subseteq \mathrm{Dom}(v)$. The *truth value* $v(\varphi)$ equals $w(\varphi)$ for any truth assignment w that extends v. □

By Exercise 1, if $\varphi, \psi \in$ PLFORM, C is a binary connective symbol, and v is a partial truth assignment, then

$$v((\neg\varphi)) = f_\neg(v(\varphi)) \tag{2.1}$$

if $\mathrm{Dom}(v) \supseteq SV(\varphi)$ and

$$v((\varphi C \psi)) = f_C(v(\varphi), v(\psi)) \tag{2.2}$$

if $\mathrm{Dom}(v) \supseteq SV(\varphi) \cup SV(\psi)$. Also, if $\varphi_0, \ldots, \varphi_{n-1} \in$ PLFORM and $v : SV \rightsquigarrow \mathbf{Bool}$ is a partial truth assignment such that $\bigcup_{0 \le i \le n-1} SV(\varphi_i) \subseteq \mathrm{Dom}(v)$, then

$$v\left(\bigvee_{0 \le i \le n-1} \varphi_i\right) = \begin{cases} \mathbf{T} & \text{if } v(\varphi_i) = \mathbf{T} \text{ for some } i,\ 0 \le i \le n-1 \\ \mathbf{F} & \text{otherwise} \end{cases}$$

and

$$v\left(\bigwedge_{0 \le i \le n-1} \varphi_i\right) = \begin{cases} \mathbf{T} & \text{if } v(\varphi_i) = \mathbf{T} \text{ for every } i,\ 0 \le i \le n-1 \\ \mathbf{F} & \text{otherwise.} \end{cases}$$

Corollary 2.3.25. *Let v, w be two partial truth assignments and let φ be a formula such that $SV(\varphi) \subseteq \mathrm{Dom}(v) \cap \mathrm{Dom}(w)$. If $v(p) = w(p)$ for every $p \in SV(\varphi)$, then $v(\varphi) = w(\varphi)$.*

Proof. Let v_1, w_1 be truth assignments that extend v and w, respectively. Then, $v(\varphi) = v_1(\varphi)$ and $w(\varphi) = w_1(\varphi)$, by definition. By the Agreement Theorem, we have $v_1(\varphi) = w_1(\varphi)$ because $v_1(p) = v(p) = w(p) = w_1(p)$ for every $p \in SV(\varphi)$. $\square$

Definition 2.3.26. Let S be a set of statement variables. A *truth table over* S is a mapping $\tau : \text{TA}_S \longrightarrow \textbf{Bool}$. ▯

If S is a finite set of statement variables, $S = \{p_{i_0}, \ldots, p_{i_{n-1}}\}$, where $i_0 < \cdots < i_{n-1}$, then a truth table over S is represented graphically as follows:

$v(p_{i_0})$	$\cdots$	$v(p_{i_{n-1}})$	$\tau(v)$
$v_0(p_{i_0})$	$\cdots$	$v_0(p_{i_{n-1}})$	$\tau(v_0)$
$\vdots$	$\vdots$	$\vdots$	$\vdots$
$v_{2^n-1}(p_{i_0})$	$\cdots$	$v_{2^n-1}(p_{i_{n-1}})$	$\tau(v_{2^n-1})$

where $v_0, \ldots, v_{2^n-1}$ are the elements of TA_S listed in lexicographic order of the sequence $(v_i(p_{i_0}), \ldots, v_i(p_{i_{n-1}}))$. Note that in this representation of a truth table, each row corresponds to a truth assignment over S. For this reason, we will sometimes refer to truth assignments over finite sets of statement variables as rows of truth tables. We denote the set of truth tables over S by $\texttt{TT}_S$. Observe that $|\texttt{TT}_S| = 2^{2^{|S|}}$.

Definition 2.3.27. Let φ be a formula and let S be a set of statement variables such that $SV(\varphi) \subseteq S$. Then, $\tau_{\varphi,S}$, the *truth table of* φ *over* S, is defined by $\tau_{\varphi,S}(v) = v(\varphi)$.

The truth table τ_φ *of* φ is $\tau_{\varphi,SV(\varphi)}$. ▯

Example 2.3.28. Consider the formula $\varphi = (p_1 \vee (\neg p_4))$. The truth table of φ over the set of variables $S = \{p_1, p_4, p_{10}\}$ is

$v(p_1)$	$v(p_4)$	$v(p_{10})$	$\tau_{\varphi,S}(v)$
F	**F**	**F**	**T**
F	**F**	**T**	**T**
F	**T**	**F**	**F**
F	**T**	**T**	**F**
T	**F**	**F**	**T**
T	**F**	**T**	**T**
T	**T**	**F**	**T**
T	**T**	**T**	**T**

On the other hand, since $SV(\varphi) = \{p_1, p_4\}$, the truth table of formula φ is given by

$v(p_1)$	$v(p_4)$	$\tau_\varphi(v)$
F	**F**	**T**
F	**T**	**F**
T	**F**	**T**
T	**T**	**T**

We may abbreviate the notation for the heading of a truth table of a formula φ by writing $p_{i_0}, \ldots, p_{i_{n-1}}, \varphi$ instead of $v(p_{i_0}), \ldots, v(p_{i_{n-1}}), \tau_{\varphi,S}(v)$, respectively. With this convention, the first truth table considered in this example becomes

p_1	p_4	p_{10}	φ
F	**F**	**F**	**T**
F	**F**	**T**	**T**
F	**T**	**F**	**F**
F	**T**	**T**	**F**
T	**F**	**F**	**T**
T	**F**	**T**	**T**
T	**T**	**F**	**T**
T	**T**	**T**	**T**

▯

Theorem 2.3.29. *A formula φ is a tautology if and only if $\tau_\varphi(v) =$ **T**, for every row v of τ_φ.*

Proof. Suppose that φ is a tautology and let v be a row of the truth table of φ. Let w be a truth assignment such that $v = w\restriction SV(\varphi)$. Since $v(\varphi) = w(\varphi) = \mathbf{T}$, we have $\tau_\varphi(v) = \mathbf{T}$.

Conversely, suppose that for every row v of τ_φ, we have $v(\varphi) = \tau_\varphi(v) = \mathbf{T}$ and let w be a truth assignment. Observe that $u = w\restriction SV(\varphi)$ is one of the rows of τ_φ and therefore, $w(\varphi) = u(\varphi) = \mathbf{T}$. □

Since there are only finitely many rows in the truth table of a formula φ, and for each row v we can compute $v(\varphi)$ using equations (2.1) and (2.2), Theorem 2.3.29 gives an algorithm that allows us to check whether a formula is a tautology. Thus, we have shown that the set of tautologies is decidable.

As we saw in Theorem 2.3.20, we can reduce several problems of propositional logic to the determination of whether or not a certain formula is a tautology or a nontautology. Therefore, the algorithm to determine whether or not a formula is a tautology also provides algorithms for testing whether a formula is satisfiable or is a contradiction. We also obtain algorithms to test whether one formula logically implies another and if two formulas are equivalent.

These algorithms can be rephrased somewhat more directly as follows:

- φ is satisfiable if and only if there is a row v in its truth table such that $\tau_\varphi(v) = \mathbf{T}$.
- φ is a contradiction if and only if for every row v in its truth table, $\tau_\varphi(v) = \mathbf{F}$.
- $\varphi \models \psi$ if and only if for every row v of the truth tables $\tau_{\varphi,S}, \tau_{\psi,S}$ we have $\tau_{\psi,S}(v) = \mathbf{T}$ whenever $\tau_{\varphi,S}(v) = \mathbf{T}$, where $S = SV(\varphi) \cup SV(\psi)$.
- $\varphi \equiv \psi$ if and only if for every row v of the truth tables $\tau_{\varphi,S}, \tau_{\psi,S}$ we have $\tau_{\psi,S}(v) = \tau_{\varphi,S}(v)$, where $S = SV(\varphi) \cup SV(\psi)$.

We conclude that both the set of satisfiable formulas and the set of contradictions are decidable. Similarly, the set of pairs (φ, ψ) of formulas such that $\varphi \models \psi$ is decidable and so is the set of pairs (φ, ψ) of formulas such that $\varphi \equiv \psi$.

The following theorem presents a useful list of equivalent formulas.

Theorem 2.3.30. *Let p_0, p_1 and p_2 be statement variables. Then, we have the following:*

$$\begin{array}{rll}
(p_0 \wedge p_0) \equiv p_0 & & \text{(idempotency of } \wedge\text{)}, \\
(p_0 \vee p_0) \equiv p_0 & & \text{(idempotency of } \vee\text{)}, \\
(p_0 \wedge p_1) \equiv (p_1 \wedge p_0) & & \text{(commutativity of } \wedge\text{)}, \\
(p_0 \vee p_1) \equiv (p_1 \vee p_0) & & \text{(commutativity of } \vee\text{)}, \\
(p_0 \wedge (p_1 \wedge p_2)) \equiv ((p_0 \wedge p_1) \wedge p_2) & & \text{(associativity of } \wedge\text{)}, \\
(p_0 \vee (p_1 \vee p_2)) \equiv ((p_0 \vee p_1) \vee p_2) & & \text{(associativity of } \vee\text{)}, \\
(\neg(\neg p_0)) \equiv p_0 & & \text{(double negation)}, \\
(p_0 \wedge (p_0 \vee p_1)) \equiv p_0 & & \text{(absorption laws)}, \\
(p_0 \vee (p_0 \wedge p_1)) \equiv p_0 & & \quad\quad\text{''},
\end{array}$$

$$\begin{array}{rll}
(p_0 \wedge (p_1 \vee p_2)) \equiv ((p_0 \wedge p_1) \vee (p_0 \wedge p_2)) & & \text{(distributivity laws)},\\
((p_1 \vee p_2) \wedge p_0) \equiv ((p_1 \wedge p_0) \vee (p_2 \wedge p_0)) & & \text{''},\\
(p_0 \vee (p_1 \wedge p_2)) \equiv ((p_0 \vee p_1) \wedge (p_0 \vee p_2)) & & \text{''},\\
((p_1 \wedge p_2) \vee p_0) \equiv ((p_1 \vee p_0) \wedge (p_2 \vee p_0)) & & \text{''},\\
(\neg(p_0 \vee p_1)) \equiv ((\neg p_0) \wedge (\neg p_1)) & & \text{(De Morgan's laws)},\\
(\neg(p_0 \wedge p_1)) \equiv ((\neg p_0) \vee (\neg p_1)) & & \text{''},\\
(p_0 \rightarrow p_1) \equiv ((\neg p_0) \vee p_1), & & \\
(p_0 \leftrightarrow p_1) \equiv ((p_0 \rightarrow p_1) \wedge (p_1 \rightarrow p_0)). & &
\end{array}$$

Proof. We show only the first De Morgan's law by building the truth tables of the formulas $(\neg(p_0 \vee p_1))$ and $((\neg p_0) \wedge (\neg p_1))$:

p_0	p_1	$(p_0 \vee p_1)$	$(\neg(p_0 \vee p_1))$	$(\neg p_0)$	$(\neg p_1)$	$((\neg p_0) \wedge (\neg p_1))$
F	**F**	**F**	**T**	**T**	**T**	**T**
F	**T**	**T**	**F**	**T**	**F**	**F**
T	**F**	**T**	**F**	**F**	**T**	**F**
T	**T**	**T**	**F**	**F**	**F**	**F**

Thus, it is apparent that the truth tables of these formulas are identical and, therefore, they are equivalent. This proves the first De Morgan's law. Similar arguments can be used to prove the remaining logical equivalences of this theorem. □

We introduce now the notion of signed formula which allows us to give more elegant presentations of certain formal systems associated with propositional logic in Chapter 3.

Definition 2.3.31. A *signed formula* is a pair (b, φ) with $b \in \mathbf{Bool}$ and $\varphi \in$ PLFORM. A signed formula (b, φ) will be denoted as $b\varphi$.

The *size* $\texttt{size}(b\varphi)$ of the signed formula $b\varphi$ is $\texttt{size}(\varphi)$. ▯

Example 2.3.32. The signed formulas $(\mathbf{T}, (p_0 \rightarrow p_1))$ and $(\mathbf{F}, (p_0 \vee p_1))$ are denoted by $\mathbf{T}(p_0 \rightarrow p_1)$ and $\mathbf{F}(p_0 \vee p_1)$, respectively. ▯

The set of signed formulas will be denoted by SPLFORM.

We will introduce a *standard order* on SPLFORM as

$$\mathbf{T}\varphi_0, \mathbf{F}\varphi_0, \mathbf{T}\varphi_1, \mathbf{F}\varphi_1, \ldots,$$

where $\varphi_0, \varphi_1, \ldots$ is the standard order on PLFORM.

The notion of signed subformula of a signed formula is based on the notion of subformula of a formula.

Definition 2.3.33. Let $b\varphi$ be a signed formula. A *signed subformula* of $b\varphi$ is a signed formula $b'\psi$, where ψ is a subformula of φ. ▯

Example 2.3.34. The signed formula $\mathbf{T}((\alpha \vee \beta) \to \gamma)$ has $\mathbf{T}(\alpha \vee \beta)$, $\mathbf{F}(\alpha \vee \beta)$, $\mathbf{F}\alpha$, $\mathbf{T}\beta$, and $\mathbf{T}\gamma$ among its signed subformulas. ▯

Definition 2.3.35. A set Δ of signed formulas is *closed* if there is a formula φ such that both $\mathbf{T}\varphi$ and $\mathbf{F}\varphi$ belong to Δ. ▯

The notions of satisfiability and logical implication can be formulated for sets of signed formulas.

Definition 2.3.36. Let φ be a formula and let $b \in \mathbf{Bool}$. A truth assignment v *satisfies* the signed formula $b\varphi$ if $v(\varphi) = b$.

A truth assignment *satisfies a set* Δ *of signed formulas* if it satisfies all signed formulas of Δ.

A set Δ of signed formulas is *satisfiable* if there is a truth assignment that satisfies Δ.

A set of signed formulas Δ *logically implies* a signed formula $b\varphi$ if every truth assignment that satisfies Δ also satisfies $b\varphi$. ▯

Example 2.3.37. The set of signed formulas $\{\mathbf{T}p_0, \mathbf{T}(p_0 \to p_1), \mathbf{F}(p_1 \to p_2)\}$ is satisfiable because any truth assignment v with $v(p_0) = \mathbf{T}, v(p_1) = \mathbf{T}$, and $v(p_2) = \mathbf{F}$ satisfies every signed formula of the set. ▯

A closed set of signed formulas is unsatisfiable. Note however that a set may be unsatisfiable without being closed, as illustrated by $\Delta = \{\mathbf{F}\varphi, \mathbf{T}(\varphi \wedge \psi)\}$.

The following theorem relates logical implication for unsigned formulas to unsatisfiability of set of signed formulas. This is the basis for certain formal systems that we discuss in Chapter 3.

Theorem 2.3.38. *Let* Γ *be a set of formulas and let* φ *be a formula. Then,* $\Gamma \models \varphi$ *if and only if the set of signed formulas* $\{\mathbf{T}\psi \mid \psi \in \Gamma\} \cup \{\mathbf{F}\varphi\}$ *is unsatisfiable.*

Proof. We leave this simple proof to the reader. □

Theorem 2.3.39. *Let Δ be a set of signed formulas and let $b\varphi$ be a signed formula. The set of signed formulas $\Delta \cup \{b\varphi\}$ is unsatisfiable if and only if $\Delta \models \overline{b}\varphi$.*

Proof. The argument is left to the reader. □

2.4 The Compactness Theorem

There are two equivalent versions of this important result. In this section, we state both versions and show that they are equivalent.

Theorem 2.4.1. *The following two statements are equivalent:*

(1) *A set of formulas is satisfiable if and only if each of its finite subsets is satisfiable.*
(2) *If Γ is a set of formulas and φ is a formula, then $\Gamma \models \varphi$ if and only if there is a finite subset Γ_0 of Γ such that $\Gamma_0 \models \varphi$.*

Proof. `(1) implies (2).` Suppose that (1) holds. Then, for every set of formulas Γ and formula φ, we have the following sequence of equivalent statements:

(a) $\Gamma \models \varphi$.
(b) $\Gamma \cup \{(\neg\varphi)\}$ is unsatisfiable.
(c) There is a finite subset of $\Gamma \cup \{(\neg\varphi)\}$ that is unsatisfiable.
(d) $\Gamma_0 \cup \{(\neg\varphi)\}$ is unsatisfiable for some finite subset Γ_0 of Γ.
(e) $\Gamma_0 \models \varphi$ for some finite subset Γ_0 of Γ.

The equivalence of (a) and (b) and of (d) and (e) follows from Theorem 2.3.17. The first statement of the theorem implies that (b) and (c) are equivalent. The equivalence of (c) and (d) follows from the fact that every superset of an unsatisfiable set is also unsatisfiable.

`(2) implies (1).` Suppose that (2) holds. Then, for any set of formulas Γ, the following statements are equivalent:

(a) Γ is satisfiable.
(b) For every contradiction φ, we have $\Gamma \not\models \varphi$.
(c) For every contradiction φ, we have $\Gamma_0 \not\models \varphi$ for every finite subset Γ_0 of Γ.
(d) Every finite subset Γ_0 of Γ is satisfiable.

The equivalence of (a) and (b) and of (c) and (d) follows from Theorem 2.3.18, while the equivalence of (b) and (c) follows from the second statement of the theorem. □

Definition 2.4.2. A set of formulas is *finitely satisfiable* if each of its finite subsets is satisfiable. ▯

It is clear that if Γ is a set of formulas that is satisfiable, then Γ is finitely satisfiable. The following theorem shows that the converse of this statement also holds.

Theorem 2.4.3 (Compactness Theorem of Propositional Logic). *Let Γ be a set of formulas. Then, Γ is satisfiable if and only if Γ is finitely satisfiable.*

Proof. As stated above, it is obvious that if Γ is satisfiable, then Γ is finitely satisfiable.

Let Γ be a finitely satisfiable set of formulas. Define the set $\Gamma_n = \{\varphi \in \Gamma \mid SV(\varphi) \subseteq \{p_0, \ldots, p_{n-1}\}\}$ for $n \in \mathbf{N}$. Of course, $\Gamma_0 = \emptyset$ and $\Gamma_0 \subseteq \Gamma_1 \subseteq \cdots \subseteq \Gamma_n \subseteq \cdots$.

Observe that, in general, a set Γ_n may be infinite since its members can be arbitrarily long formulas. Nevertheless, formulas in Γ_n are written using a finite set of statement variables, namely $V_n = \{p_0, \ldots, p_{n-1}\}$ for $n \in \mathbf{N}$. Therefore, if $T_n = \{\tau_{\varphi, V_n} \mid \varphi \in \Gamma_n\}$, then T_n is a finite set (which contains at most 2^{2^n} elements). Consequently, there exists a finite subset Γ'_n of Γ_n such that $T_n = \{\tau_{\varphi, V_n} \mid \varphi \in \Gamma'_n\}$. By hypothesis, Γ'_n is satisfiable. Since every formula of Γ_n is logically equivalent to a formula of Γ'_n (because they have the same truth tables over V_n), it follows that every set Γ_n is satisfiable. Let w_n be a truth assignment that satisfies Γ_n. Clearly, w_n satisfies Γ_i for all $i \leq n$.

Define the truth assignment v recursively by

$$v(p_n) = \begin{cases} \mathbf{T} & \text{if } v \restriction V_n \cup \{(p_n, \mathbf{T})\} \subseteq w_m \text{ for infinitely many } m \\ \mathbf{F} & \text{otherwise.} \end{cases}$$

We claim that for all n, $v \restriction V_n \subseteq w_m$ for infinitely many m. For $n = 0$, this is trivial since $V_0 = \emptyset$. Suppose that the statement is true for n. Then, there are infinitely many m such that $v \restriction V_n \subseteq w_m$, and for each such m, either $v \restriction V_n \cup \{(p_n, \mathbf{T})\} \subseteq w_m$ or $v \restriction V_n \cup \{(p_n, \mathbf{F})\} \subseteq w_m$. Thus, if there are only finitely many m such that $v \restriction V_n \cup \{(p_n, \mathbf{T})\} \subseteq w_m$, then there are infinitely many m such that $v \restriction V_n \cup \{(p_n, \mathbf{F})\} \subseteq w_m$. Consequently, whichever alternative holds in the definition of $v(p_n)$, there are infinitely many m such that $v \restriction V_{n+1} \subseteq w_m$.

Let $\varphi \in \Gamma$. Then there is an n such that $\varphi \in \Gamma_n$ and for every $m \geq n$, $w_m(\varphi) = \mathbf{T}$. Since $v \restriction V_n \subseteq w_m$ for infinitely many m, it follows that for some $m \geq n$, we have $v(\varphi) = v \restriction V_n(\varphi) = w_m(\varphi) = \mathbf{T}$. □

Corollary 2.4.4. *If* Γ *is a set of formulas and* φ *is a formula, then* $\Gamma \models \varphi$ *if and only if there is a finite subset* Γ_0 *of* Γ *such that* $\Gamma_0 \models \varphi$.

Proof. The corollary follows directly from Theorems 2.4.1 and 2.4.3. □

As an application of the Compactness Theorem, we will prove a result from graph theory concerning the possibility of coloring the vertices of a graph so that no two adjacent vertices are colored with the same color. We remind the reader that a *graph* is a pair $G = (V, E)$, where V is a set and E is a collection of two-element subsets of V. The elements of V are called the *vertices* of G and the elements of E are called the *edges* of G. The graph G is called finite (countable) if V is finite (countable). A graph (V', E') is called a *subgraph* of a graph (V, E) if $V' \subseteq V$ and $E' \subseteq E$.

Definition 2.4.5. Let $G = (V, E)$ be a graph and let k be a positive integer. A *k-coloring of* G is a function $c : V \longrightarrow \{0, \ldots, k-1\}$ such that for every $x, y \in V$, if $\{x, y\} \in E$, then $c(x) \neq c(y)$. If there is a k-coloring of G, we call G k-colorable.

If $c(x) = i$, we say that the vertex x has been colored with the color i by c.

▯

Theorem 2.4.6. *Let* $G = (V, E)$ *be a countable graph and* k *be a positive integer. If every finite subgraph of* G *is* k*-colorable, then* G *is* k*-colorable.*

Proof. Since G is countable, we can attach to every vertex $x \in V$ and color $i \in \{0, \ldots, k-1\}$ a distinct statement variable p_x^i. Let Γ

be the set of formulas which consists of the following families of formulas:

- $\{(p_x^0 \vee \cdots \vee p_x^{k-1}) \mid x \in V\}$,
- $\{(\neg(p_x^i \wedge p_x^j)) \mid x \in V, i, j \in \{0, \ldots, k-1\}$ and $i \neq j\}$,
- $\{(\neg(p_x^i \wedge p_y^i)) \mid x, y \in V, i \in \{0, \ldots, k-1\}$ and $\{x, y\} \in E\}$.

We claim that every finite subset of Γ is satisfiable. To see this, let $\Gamma' \subseteq \Gamma$ be finite and let $V' = \{x \in V \mid p_x^i \in SV(\Gamma')$ for some $i\}$. Consider the finite subgraph $G' = (V', E')$ of G, where E' consists of those edges $\{x, y\}$ of E such that $x, y \in V'$. By hypothesis, G' is k-colorable. Let c' be a k-coloring of G'. Define a partial truth assignment v' over $SV(\Gamma')$ by letting $v'(p_x^i) = \mathbf{T}$ if $c'(x) = i$ and $v'(p_x^i) = \mathbf{F}$, otherwise. It is easy to verify that v' satisfies Γ'. Therefore, by the Compactness Theorem, Γ is satisfiable, so let v be a truth assignment that satisfies Γ. Since v satisfies the formulas $(p_x^0 \vee \cdots \vee p_x^{k-1})$ and $(\neg(p_x^i \wedge p_x^j))$ for every x in V and all colors i and j, it follows that for each $x \in V$ there is a unique i with $v(p_x^i) = \mathbf{T}$. This allows us to define a function c by $c(x) = i$, where $v(p_x^i) = \mathbf{T}$. Since v also satisfies $\{(\neg(p_x^i \wedge p_y^i)) \mid x, y \in V, i \in \{0, \ldots, k-1\}$ and $\{x, y\} \in E\}$, the mapping c is a k-coloring of G. □

The previous result was obtained in 1951 using graph theoretical methods in [29]. Two years later, Beth pointed out in [1] that the result could be obtained as an easy consequence of the compactness theorem. Our proof is a modified version of Beth's proof.

2.5 Normal Forms for Formulas

In this section, we show that for every formula φ there exist logically equivalent formulas that satisfy certain syntactic prescriptions. These restricted formulas are referred to as *normal forms* for φ.

Definition 2.5.1. Let φ be a formula and let $b \in \mathbf{Bool}$. The formula φ^b is given by

$$\varphi^b = \begin{cases} \varphi & \text{if } b = \mathbf{T} \\ (\neg\varphi) & \text{if } b = \mathbf{F}. \end{cases}$$

At times, we will use φ^1 for $\varphi^{\mathbf{T}}$ and φ^0 for $\varphi^{\mathbf{F}}$. ▯

Lemma 2.5.2. *Let φ be a formula and let v and w be two partial truth assignments, $v, w \in TA_S$, where $SV(\varphi) \subseteq S$. Then, we have*

$$w(\varphi^{v(\varphi)}) = \begin{cases} \mathbf{T} & \text{if } w(\varphi) = v(\varphi) \\ \mathbf{F} & \text{otherwise.} \end{cases}$$

Proof. The reader can verify the statement by examining the four possible cases determined by $v(\varphi)$ and $w(\varphi)$. □

Definition 2.5.3. Let S be a nonempty set of n statement variables, $S = \{p_{i_0}, \ldots, p_{i_{n-1}}\}$ with $i_0 < \cdots < i_{n-1}$.

A conjunction $\mu = (p_{i_0}^{a_0} \wedge \cdots \wedge p_{i_{n-1}}^{a_{n-1}})$, where $a_0, \ldots, a_{n-1} \in \mathbf{Bool}$ is called a *minterm over* S. The set of literals $\{p_{i_0}^{a_0}, \ldots, p_{i_{n-1}}^{a_{n-1}}\}$ will be denoted by $\mathrm{LIT}(\mu)$. The set of minterms over S will be denoted by $\mathrm{MINTRM}(S)$. A *partial minterm over* S is a minterm over a nonempty subset of S. The set of partial minterms over S will be denoted by $\mathrm{PMINTRM}(S)$.

A *maxterm over* S is a disjunction $(p_{i_0}^{a_0} \vee \cdots \vee p_{i_{n-1}}^{a_{n-1}})$, where $a_0, \ldots, a_{n-1} \in \mathbf{Bool}$. The set of maxterms over S will be denoted by $\mathrm{MAXTRM}(S)$. A *partial maxterm over* S is a maxterm over a nonempty subset of S. The set of partial maxterms over S will be denoted by $\mathrm{PMAXTRM}(S)$.

A formula φ is a *minterm* (*maxterm*) if there is a set S of statement variables such that φ is minterm (maxterm) over S. □

The set $\mathrm{PMINTRM}(SV)$ is the set of all minterms. Observe also that if S is a finite set of statement variables, then $|\mathrm{PMINTRM}(S)| = 3^{|S|} - 1$.

Example 2.5.4. Let $S = \{p_0, p_2, p_3\}$. Then $(p_0 \wedge (\neg p_2) \wedge p_3)$ and $((\neg p_0) \wedge p_2 \wedge p_3)$ are both minterms over S.

The formula $((\neg p_0) \vee (\neg p_2) \vee p_3)$ is a maxterm over S. □

Definition 2.5.5. Let $S = \{p_{i_0}, \ldots, p_{i_{n-1}}\}$ with $i_0 < \cdots < i_{n-1}$ be a nonempty set of statement variables and let $v \in \mathrm{TA}_S$.

The minterm $(p_{i_0}^{v(p_{i_0})} \wedge \cdots \wedge p_{i_{n-1}}^{v(p_{i_{n-1}})})$ will be denoted by μ_v and the maxterm $(p_{i_0}^{v'(p_{i_0})} \vee \cdots \vee p_{i_{n-1}}^{v'(p_{i_{n-1}})})$ where $v'(p) = f_\neg(v(p))$ for every $p \in S$ will be denoted by ν_v.

For v defined by $v(p_{i_j}) = \mathbf{T}$, for $0 \leq j \leq n-1$, we use the alternative notations μ_S, ν_S for μ_v, ν_v, respectively. In other words, we have $\mu_S = (p_{i_0} \wedge \cdots \wedge p_{i_{n-1}})$ and $\nu_S = ((\neg p_{i_0}) \vee \cdots \vee (\neg p_{i_{n-1}}))$. □

Note that a formula is a minterm (maxterm) over a set of variables S if and only if it equals μ_v (ν_v) for some truth assignment v over S and in this case v is unique.

Example 2.5.6. Let $S = \{p_0, p_1, p_2\}$ and let $v \in \mathrm{TA}_S$ be given by $v(p_0) = \mathbf{T}$, $v(p_1) = \mathbf{F}$, $v(p_2) = \mathbf{F}$. The minterm and the maxterm corresponding to this truth assignment are

$$\mu_v = (p_0^{\mathbf{T}} \wedge p_1^{\mathbf{F}} \wedge p_2^{\mathbf{F}}) = (p_0 \wedge (\neg p_1) \wedge (\neg p_2))$$

and

$$\nu_v = (p_0^{\mathbf{F}} \vee p_1^{\mathbf{T}} \vee p_2^{\mathbf{T}}) = ((\neg p_0) \vee p_1 \vee p_2),$$

respectively. ▯

Theorem 2.5.7. *Let v be a truth assignment over a finite, nonempty set S of statement variables. Then, for all $w \in TA_S$, we have*

$$w(\mu_v) = \begin{cases} \mathbf{T} & \text{if } v = w \\ \mathbf{F} & \text{otherwise,} \end{cases}$$

$$w(\nu_v) = \begin{cases} \mathbf{T} & \text{if } v \neq w \\ \mathbf{F} & \text{otherwise.} \end{cases}$$

Proof. Let $S = \{p_{i_0}, \ldots, p_{i_{n-1}}\}$ and let $v, w \in \mathrm{TA}_S$. We have $w(\mu_v) = \mathbf{T}$ if and only if $w(p_{i_k}^{v(p_{i_k})}) = \mathbf{T}$ for all k, $0 \leq k \leq n-1$. By Lemma 2.5.2, we can rephrase this as $w(\mu_v) = \mathbf{T}$ if and only if $w(p_{i_k}) = v(p_{i_k})$ for every k, $0 \leq k \leq n-1$, which amounts to $v = w$.

We leave the second part of the theorem to the reader. □

Definition 2.5.8. A formula is in *disjunctive normal form* (dnf) if it is a disjunction of conjunctions of literals. Each conjunction is called a *disjunct* of the formula.

A formula is in *conjunctive normal form* (cnf) if it is a conjunction of disjunctions of literals. Each disjunction is called a *conjunct* of the formula. ▯

Example 2.5.9. The formula $((p_0 \wedge (\neg p_1)) \vee (p_1 \wedge (\neg p_2)))$ is in dnf, while $((p_0 \vee (\neg p_1)) \wedge (p_1 \vee (\neg p_2)))$ is in cnf.

If p is a statement variable, then the formula p is in both dnf and cnf. Indeed, if we regard p as a disjunction of one literal, then we can view the formula p as a conjunction whose single conjunct is the disjunction p. A similar argument allows us to view p as a disjunction whose single disjunct is the one-literal conjunction p.

By a similar argument, $(p_0 \wedge (\neg p_1))$ is both in dnf and cnf. ▯

Theorem 2.5.10. *Let S be a finite, nonempty set of statement variables and let $\tau \in \mathtt{TT}_S$. Then, there is a formula ψ in disjunctive normal form with $SV(\psi) = S$ such that $\tau_{\psi,S} = \tau$. In addition, if $\tau(v) = \mathbf{T}$ for at least one $v \in TA_S$, then such a ψ exists each of whose disjuncts is a minterm over S.*

Proof. Suppose first that there is at least one truth assignment v over S such that $\tau(v) = \mathbf{T}$. Then, let $\{v_0, \ldots, v_{n-1}\} = \{v \in \mathrm{TA}_S \mid \tau(v) = \mathbf{T}\}$. Define $\psi = \bigvee_{0 \leq i \leq n-1} \mu_{v_i}$. For each $w \in \mathrm{TA}_S$, we have

$$\begin{aligned}
\tau_{\psi,S}(w) &= w(\psi) \\
&= w\left(\bigvee_{0 \leq i \leq n-1} \mu_{v_i}\right) \\
&= \begin{cases} \mathbf{T} & \text{if } w(\mu_{v_i}) = \mathbf{T} \text{ for some } i,\ 0 \leq i \leq n-1 \\ \mathbf{F} & \text{otherwise} \end{cases} \\
&= \begin{cases} \mathbf{T} & \text{if } w = v_i \text{ for some } i,\ 0 \leq i \leq n-1 \\ \mathbf{F} & \text{otherwise} \end{cases} \\
&= \tau(w).
\end{aligned}$$

Therefore, $\tau_{\psi,S} = \tau$.

If there is no truth assignment $v \in \mathrm{TA}_S$ such that $\tau(v) = \mathbf{T}$, then we can take for ψ the contradiction $\psi = (p_{i_0} \wedge (\neg p_{i_0}) \wedge p_{i_1} \wedge \cdots \wedge p_{i_{n-1}})$, where $S = \{p_{i_0}, \ldots, p_{i_{n-1}}\}$. □

Note that the proof of Theorem 2.5.10 provides an effective method of producing a disjunctive normal form for a given formula.

Example 2.5.11. Consider the set of variables $S = \{p_0, p_4, p_6\}$ and the truth table τ over S given in the following:

$v(p_0)$	$v(p_4)$	$v(p_6)$	$\tau(v)$
F	**F**	**F**	**F**
F	**F**	**T**	**T**
F	**T**	**F**	**F**
F	**T**	**T**	**F**
T	**F**	**F**	**F**
T	**F**	**T**	**T**
T	**T**	**F**	**F**
T	**T**	**T**	**T**

A formula in disjunctive normal form obtained using the method of Theorem 2.5.10 whose truth table is τ is

$$\psi = (((\neg p_0) \wedge (\neg p_4) \wedge p_6) \vee (p_0 \wedge (\neg p_4) \wedge p_6) \vee (p_0 \wedge p_4 \wedge p_6)).$$

▯

Corollary 2.5.12. *Let* $\varphi \in$ PLFORM. *Then, there is a formula* ψ *in disjunctive normal form such that* $SV(\varphi) = SV(\psi)$ *and* $\varphi \equiv \psi$. *If* φ *is not a contradiction, then such a* ψ *exists each of whose disjuncts is a minterm over* $SV(\varphi)$.

Proof. By applying Theorem 2.5.10, to the set $S = SV(\varphi)$ and to the truth table $\tau_\varphi = \tau_{\varphi,S}$, we get a formula ψ in dnf such that $SV(\psi) = S$ and $\tau_{\psi,S} = \tau_{\varphi,S}$, which implies that $\varphi \equiv \psi$. From the second part of Theorem 2.5.10, it follows that if φ is not a contradiction, then we can chose ψ to be a disjunction of minterms over S.

□

A formula ψ which is in disjunctive normal form and is logically equivalent to φ is called a *disjunctive normal form* for φ.

Example 2.5.13. Let $\varphi = ((p_4 \rightarrow p_0) \wedge p_6)$. Then, the truth table of φ is the one given in Example 2.5.11. Consequently, the formula ψ given in this example is a disjunctive normal form for φ. ▯

Theorem 2.5.14. *If $\psi = \mu_0 \vee \cdots \vee \mu_{n-1}$ is a disjunctive normal form for φ such that each μ_i is a minterm over $SV(\varphi)$, then $\{\mu_i \mid 0 \leq i \leq n-1\} = \{\mu_v \mid v \in TA_{SV(\varphi)}$ and $v(\varphi) = \mathbf{T}\}$.*

Proof. For every minterm μ_i, there exists a truth assignment v over $SV(\varphi)$ such that $\mu_i = \mu_v$. Then, $v(\mu_i) = \mathbf{T}$, so $v(\varphi) = v(\psi) = \mathbf{T}$, which means that $\mu_i \in \{\mu_v \mid v \in \mathrm{TA}_{\mathrm{SV}(\varphi)}$ and $v(\varphi) = \mathbf{T}\}$. Conversely, if $v \in \mathrm{TA}_{\mathrm{SV}(\varphi)}$ is such that $v(\varphi) = \mathbf{T}$ and $\mu_v \neq \mu_i$ for all i, $0 \leq i \leq n-1$, then $v(\mu_i) = \mathbf{F}$ for all i (since each $\mu_i = \mu_w$ for some $w \in \mathrm{TA}_{\mathrm{SV}(\varphi)}$), so $v(\psi) = \mathbf{F}$, which contradicts $\varphi \equiv \psi$. □

Theorem 2.5.14 shows that the disjunctive normal form for a satisfiable formula φ as constructed in Corollary 2.5.12 is the unique (up to order and repetitions of minterms) disjunctive normal form for φ which consists of minterms over $SV(\varphi)$. If ψ is a disjunctive normal form for φ that consists of distinct minterms over $SV(\varphi)$, we refer to ψ as a *standard disjunctive normal form* for φ. Observe that any formula obtained from a standard disjunctive normal form for φ by permuting the minterms is also a standard disjunctive normal form for φ. We shall refer to the minterms that occur in a standard disjunctive normal form for φ (i.e., to $\{\mu_v \mid v \in \mathrm{TA}_{\mathrm{SV}(\varphi)}$ and $v(\varphi) = \mathbf{T}\}$) as the *minterms of* φ. We will denote the set of minterms of φ by M_φ.

Theorem 2.5.15. *Let φ be a formula and let $S = SV(\varphi)$. Then,*

$$M_\varphi = \{\mu \in \text{ MINTRMS} S \ \mid \mu \models \varphi\}.$$

Proof. Let $\mu \in M_\varphi$. Then, $\mu \in \mathrm{MINTRM}(S)$ and $\mu = \mu_v$ for some $v \in \mathrm{TA}_{\mathrm{SV}(\varphi)}$ such that $v(\varphi) = \mathbf{T}$. If $w \in \mathrm{TA}_{\mathrm{SV}(\varphi)}$ is such that $w(\mu) = \mathbf{T}$, then $w = v$, so $w(\varphi) = \mathbf{T}$. Thus, $\mu \models \varphi$, and we have $M_\varphi \subseteq \{\mu \in \mathrm{MINTRM}(S) \mid \mu \models \varphi\}$.

Conversely, suppose that $\mu \in \mathrm{MINTRM}(S)$ and $\mu \models \varphi$. There is $v \in \mathrm{TA}_{\mathrm{SV}(\varphi)}$ such that $\mu = \mu_v$. Since $v(\mu) = \mathbf{T}$, $v(\varphi) = \mathbf{T}$ and this shows the reverse inclusion. □

Although the standard disjunctive normal form for a satisfiable formula is essentially unique, several disjunctive normal forms can exist if we allow minterms over subsets of $SV(\varphi)$.

Example 2.5.16. The reader can easily verify that the following formulas

$$(((\neg p_0) \wedge (\neg p_4) \wedge p_6) \vee (p_0 \wedge p_6))$$
$$(((\neg p_4) \wedge p_6) \vee (p_0 \wedge p_4 \wedge p_6))$$

are alternative disjunctive normal forms for the formula ψ introduced in Example 2.5.11. ▯

Given a formula φ of propositional logic, it is often desirable to find the "simplest" formula ψ which is logically equivalent to φ, where "simplest" is a term which we do not define precisely here. (One possible definition of "simplest" is "shortest".) We now give techniques which allow for the solution of this problem, if we restrict our attention to formulas ψ in disjunctive normal form.

Definition 2.5.17. Let φ be a formula. An *implicant* of φ is a partial minterm μ over $SV(\varphi)$ such that $\mu \models \varphi$.

The set of implicants of a formula φ will be denoted by $\mathrm{IMPL}(\varphi)$. ▯

Note that $|\mathrm{IMPL}(\varphi)| < 3^{|\mathrm{SV}(\varphi)|}$. The following theorem helps us to explain the role of implicants in optimizing disjunctive normal forms for formulas.

Theorem 2.5.18. *If $\psi = \mu_0 \vee \cdots \vee \mu_{n-1}$ is a disjunctive normal form for φ, where $\mu_0, \ldots, \mu_{n-1}$ are partial minterms over $SV(\varphi)$, then each μ_i is an implicant of φ, for $0 \leq i \leq n-1$.*

Proof. We leave the argument to the reader. □

Observe that μ is an implicant of φ with $SV(\mu) = SV(\varphi)$ if and only if $\mu = \mu_v$ for some $v \in \mathrm{TA}_{\mathrm{SV}(\varphi)}$ such that $v(\varphi) = \mathbf{T}$.

Definition 2.5.19. The *domination relation* is the relation $\leq$ defined on the set of all minterms by $\mu \leq \mu'$ if $\mu \models \mu'$. If $\mu \leq \mu'$, we shall say that μ' *dominates* μ. ▯

Theorem 2.5.20. *If μ, μ' are minterms, then $\mu \models \mu'$ if and only if every literal conjunct of μ' is also a conjunct of μ.*

Proof. If every literal conjunct of μ' is also a conjunct of μ, it is clear that $\mu \models \mu'$ because both μ and μ' are conjunctions of literals.

Conversely, suppose that $\mu \models \mu'$ and suppose that there exists a literal conjunct of μ' that is not a conjunct of μ. This means that there exists a statement variable p_j such that p_j^b is a conjunct of μ' and either $p_j \notin SV(\mu)$, or $p_j^{f_\neg(b)}$ is a conjunct of μ, where $\mu = (p_{i_0}^{a_0} \wedge \cdots \wedge p_{i_{m-1}}^{a_{m-1}})$.

Suppose, for instance, that p_j is a conjunct of μ' and $p_j \notin SV(\mu)$. Consider $v \in \mathrm{TA}$ such that $v(p_{i_k}) = a_k$ for $0 \leq k \leq m-1$ and $v(p_j) = \mathbf{F}$. We have $v(\mu) = \mathbf{T}$ and this implies $v(\mu') = \mathbf{T}$. This leads to a contradiction because p_j is a conjunct of μ'. A similar argument works when $(\neg p_j)$ is a conjunct of μ' and $p_j \notin SV(\mu)$. Suppose now that p_j^b is a conjunct of μ' and $p_j^{f_\neg(b)}$ is a conjunct of μ. If v is a truth assignment such that $v(\mu) = \mathbf{T}$, then $v(p_j) = f_\neg(b)$ and $v(\mu') = \mathbf{T}$. This, in turn, implies $v(p_j) = b$, which is a contradiction. Therefore, every literal conjunct of μ' is also a conjunct of μ. □

Corollary 2.5.21. *The pair* $(PMINTRM(SV), \leq)$ *is a partially ordered set.*

Proof. We need to verify that "$\leq$" is reflexive, antisymmetric, and transitive. The reflexivity and transitivity properties are immediate consequences of Theorem 2.3.12, so we prove only the antisymmetry property.

Let $\mu, \mu' \in \mathrm{PMINTRM}(SV)$ be such that $\mu \leq \mu'$ and $\mu' \leq \mu$. From Theorem 2.5.20, we infer that these conjunctions consist of exactly the same literals. Since every member of $\mathrm{PMINTRM}(SV)$ is determined by the set of literals it contains, we conclude that $\mu = \mu'$ which proves that $\leq$ is antisymmetric. □

Corollary 2.5.22. *If S, S' are two finite nonempty sets of variables such that $S \subseteq S'$, then $\mu_{S'} \models \mu_S$.*

Proof. This is an immediate consequence of Theorem 2.5.20. □

Corollary 2.5.23. *If* $\mu, \mu' \in \mathrm{IMPL}(\varphi)$, *then* $\mu \leq \mu'$ *if and only if every literal conjunct of* μ' *is also a conjunct of* μ.

Proof. This statement follows immediately from Theorem 2.5.20. □

Corollary 2.5.24. *If* $\mu, \mu' \in \mathrm{IMPL}(\varphi)$, *then* $\mu < \mu'$ *implies* $SV(\mu') \subset SV(\mu)$.

Proof. This statement follows immediately from Theorem 2.5.20. □

Theorem 2.5.25. *The minimal elements of the finite poset* $(\text{IMPL}(\varphi), \leq)$ *are the minterms of* φ.

Proof. It is clear that every minterm μ_v with $v \in \text{TA}_{\text{SV}(\varphi)}$ and $v(\varphi) = \mathbf{T}$ is a minimal element of the poset $(\text{IMPL}(\varphi), \leq)$. Conversely, let μ be a minimal element of $(\text{IMPL}(\varphi), \leq)$ and let $v \in \text{TA}_{\text{SV}(\varphi)}$ be such that $v(\mu) = \mathbf{T}$. This gives $\mu_v \leq \mu$ and the minimality of μ implies $\mu = \mu_v$. $\square$

Definition 2.5.26. The *prime implicants* of the formula φ are the maximal elements of the poset $(\text{IMPL}(\varphi), \leq)$. ▯

Definition 2.5.27. The *rank of an implicant* $\mu \in \text{IMPL}(\varphi)$ is $r(\mu) = |SV(\varphi)| - |SV(\mu)|$.

The *n-th implicant layer of the formula* φ is the set $L_n(\varphi)$ that consists of all implicants of rank n of φ for $0 \leq n \leq |SV(\varphi)| - 1$. ▯

Corollary 2.5.24 implies that if $\mu < \mu'$ for $\mu, \mu' \in \text{IMPL}(\varphi)$, then $r(\mu) < r(\mu')$.

It follows from a previous comment that if φ is satisfiable, then $L_0(\varphi)$ consists of all the minterms of φ. Note that

$$\text{IMPL}(\varphi) = \bigcup_{0 \leq i \leq |\text{SV}(\varphi)|-1} L_i(\varphi).$$

Definition 2.5.28. Let μ_0, μ_1 be minterms such that $SV(\mu_0) = SV(\mu_1)$, and

$$\mu_0 = (p_{i_0}^{a_0} \wedge \cdots \wedge p_{i_{m-1}}^{a_{m-1}}),$$

$$\mu_1 = (p_{i_0}^{b_0} \wedge \cdots \wedge p_{i_{m-1}}^{b_{m-1}}),$$

where $m > 1$, $a_k = b_k$ for $0 \leq k \leq m-1$, $k \neq j$, and $a_j = f_\neg(b_j)$. The formula $\mu_0 \star \mu_1$ is the conjunction

$$(p_{i_0}^{a_0} \wedge \cdots \wedge p_{i_{j-1}}^{a_{j-1}} \wedge p_{i_{j+1}}^{a_{j+1}} \wedge \cdots \wedge p_{i_{m-1}}^{a_{m-1}}).$$

▯

Lemma 2.5.29. *If* $\mu_0, \mu_1 \in \text{IMPL}(\varphi)$ *are such that* $\mu_0 \star \mu_1$ *is defined, then* $\mu_0 \star \mu_1 \in \text{IMPL}(\varphi)$*; furthermore,* $\mu_0 \star \mu_1 = \sup\{\mu_0, \mu_1\}$ *in the poset* $(\text{IMPL}(\varphi), \leq)$.

Proof. Let $v \in \mathrm{TA}_{\mathrm{SV}(\varphi)}$ be such that $v(\mu_0 \star \mu_1) = \mathbf{T}$, where

$$\mu_0 = (p_{i_0}^{a_0} \wedge \cdots \wedge p_{i_j}^{a_j} \wedge \cdots \wedge p_{i_{m-1}}^{a_{m-1}}),$$

$$\mu_1 = (p_{i_0}^{a_0} \wedge \cdots \wedge p_{i_j}^{\overline{a_j}} \wedge \cdots \wedge p_{i_{m-1}}^{a_{m-1}}),$$

and $\overline{a_j} = f_\neg(a_j)$. If $v(p_{i_j}) = a_j$, we have $v(\mu_0) = \mathbf{T}$, so $v(\varphi) = \mathbf{T}$; otherwise, that is, if $v(p_{i_j}) = f_\neg(a_j)$, we have $v(\mu_1) = \mathbf{T}$, so again $v(\varphi) = \mathbf{T}$. Thus, $\mu_0 \star \mu_1$ is an implicant of φ. We leave it to the reader to verify that $\mu_0 \star \mu_1 = \sup\{\mu_0, \mu_1\}$ in $(\mathrm{IMPL}(\varphi), \leq)$. $\square$

Lemma 2.5.29 shows that $\star$ is a partial operation on $\mathrm{IMPL}(\varphi)$; observe that $r(\mu_0 \star \mu_1) = r(\mu_0) + 1 = r(\mu_1) + 1$.

Theorem 2.5.30. *For every formula φ, we have*

$$L_{n+1}(\varphi) = \{\mu_0 \star \mu_1 \mid \mu_0, \mu_1 \in L_n(\varphi) \text{ and } \mu_0 \star \mu_1 \text{ is defined}\}$$

for $0 \leq n \leq |SV(\varphi)| - 2$.

Proof. Lemma 2.5.29 implies

$$\{\mu_0 \star \mu_1 \mid \mu_0, \mu_1 \in L_n(\varphi) \text{ and } \mu_0 \star \mu_1 \text{ is defined}\} \subseteq L_{n+1}(\varphi)$$

for $0 \leq n \leq |SV(\varphi)| - 2$. Therefore, we need to prove only the reverse inclusion.

Let $\mu \in L_{n+1}(\varphi)$. Since $|SV(\varphi)| - |SV(\mu)| = n + 1 \geq 1$, there exists a variable $p_l \in SV(\varphi)$ that does not occur in μ. If $\mu = (p_{i_0}^{a_0} \wedge \cdots \wedge p_{i_{m-1}}^{a_{m-1}})$, then both μ_0 and μ_1 belong to $L_n(\varphi)$, where

$$\mu_0 = (p_{i_0}^{a_0} \wedge \cdots \wedge (\neg p_l) \wedge \cdots \wedge p_{i_{m-1}}^{a_{m-1}}),$$

$$\mu_1 = (p_{i_0}^{a_0} \wedge \cdots \wedge p_l \wedge \cdots \wedge p_{i_{m-1}}^{a_{m-1}})$$

have been obtained from μ by inserting $(\neg p_l)$ and p_l, respectively, in the appropriate position. Also, it is clear that $\mu = \mu_0 \star \mu_1$ and this proves the reverse inclusion. $\square$

Theorem 2.5.30 proves that the following algorithm can be used to generate the set of all implicants of a formula φ:

Algorithm 2.5.31 (Quine[1]–McCluskey[2] Algorithm for constructing the set IMPL(φ)**).**
Input: A satisfiable formula $\varphi \in$ PLFORM.
Output: The collection of all implicant layers of φ.
Method:

(A) Let $L_0(\varphi)$ be the set of minterms of φ.
(B) For every i, $0 \leq i \leq |SV(\varphi)|-2$, construct the layer $L_{i+1}(\varphi) = \{\mu_0 \star \mu_1 \mid \mu_0, \mu_1 \in L_i(\varphi) \text{ and } \mu_0 \star \mu_1 \text{ is defined}\}$.
(C) Output the collection $\{L_i | 0 \leq i \leq |SV(\varphi)| - 1\}$.

Proof of Correctness: The correctness of the algorithm follows immediately from Theorem 2.5.30. □

Example 2.5.32. Consider the formula $\varphi = (p_0 \to (p_1 \leftrightarrow p_2))$. Its truth table is

p_0	p_1	p_2	φ
F	**F**	**F**	**T**
F	**F**	**T**	**T**
F	**T**	**F**	**T**
F	**T**	**T**	**T**
T	**F**	**F**	**T**
T	**F**	**T**	**F**
T	**T**	**F**	**F**
T	**T**	**T**	**T**

The set of minterms obtained in Step (A) of Algorithm 2.5.31 is

$$L_0 = \{\mu_{v_0}, \mu_{v_1}, \mu_{v_2}, \mu_{v_3}, \mu_{v_4}, \mu_{v_7}\},$$

[1]Willard Van Orman Quine was born on June 15, 1908, in Akron, Ohio, and died on December 25, 2000, in Boston, Massachusetts. Quine studied at Oberlin College and Harvard where he got his Ph.D. in 1932 and taught at Harvard. Quine's contributions were in both philosophy and mathematical logic and he was the author of several influential books. He was a former president of the American Philosophical Association.
[2]Edward J. McCluskey was born on October 16, 1929, and died on February 13, 2016. He studied at Bowdoin College and received his doctorate from MIT in 1956. McCluskey worked at Bell Labs and then taught at Princeton and Stanford. He served as the Director of the Center for Reliable Computing at Stanford. His research interests were in the areas of fault-tolerant computing, computer architecture, and logic design.

where

$$\begin{aligned}
\mu_{v_0} &= (p_0^{\mathbf{F}} \wedge p_1^{\mathbf{F}} \wedge p_2^{\mathbf{F}}),\\
\mu_{v_1} &= (p_0^{\mathbf{F}} \wedge p_1^{\mathbf{F}} \wedge p_2^{\mathbf{T}}),\\
\mu_{v_2} &= (p_0^{\mathbf{F}} \wedge p_1^{\mathbf{T}} \wedge p_2^{\mathbf{F}}),\\
\mu_{v_3} &= (p_0^{\mathbf{F}} \wedge p_1^{\mathbf{T}} \wedge p_2^{\mathbf{T}}),\\
\mu_{v_4} &= (p_0^{\mathbf{T}} \wedge p_1^{\mathbf{F}} \wedge p_2^{\mathbf{F}}),\\
\mu_{v_7} &= (p_0^{\mathbf{T}} \wedge p_1^{\mathbf{T}} \wedge p_2^{\mathbf{T}}).
\end{aligned}$$

Layer L_1, constructed in Step (C), consists of the conjunctions listed in the following table:

Conjunction	Obtained from
$(p_0^{\mathbf{F}} \wedge p_1^{\mathbf{F}})$	$\mu_{v_0} \star \mu_{v_1}$
$(p_0^{\mathbf{F}} \wedge p_2^{\mathbf{F}})$	$\mu_{v_0} \star \mu_{v_2}$
$(p_1^{\mathbf{F}} \wedge p_2^{\mathbf{F}})$	$\mu_{v_0} \star \mu_{v_4}$
$(p_0^{\mathbf{F}} \wedge p_2^{\mathbf{T}})$	$\mu_{v_1} \star \mu_{v_3}$
$(p_0^{\mathbf{F}} \wedge p_1^{\mathbf{T}})$	$\mu_{v_2} \star \mu_{v_3}$
$(p_1^{\mathbf{T}} \wedge p_2^{\mathbf{T}})$	$\mu_{v_3} \star \mu_{v_7}$

Layer L_2 consists of

Conjunction	Obtained from
$p_0^{\mathbf{F}}$	$(p_0^{\mathbf{F}} \wedge p_1^{\mathbf{F}}) \star (p_0^{\mathbf{F}} \wedge p_1^{\mathbf{T}})$
	or
	$(p_0^{\mathbf{F}} \wedge p_2^{\mathbf{F}}) \star (p_0^{\mathbf{F}} \wedge p_2^{\mathbf{T}})$

The Hasse diagram of the poset $(\mathrm{IMPL}(\varphi), \leq)$ is given in Figure 2.1. The prime implicants of φ are $(\neg p_0)$, $(p_1 \wedge p_2)$ and $((\neg p_1) \wedge (\neg p_2))$. □

Definition 2.5.33. A nonempty set $Q = \{\mu_0, \ldots, \mu_{m-1}\}$ of implicants of φ is a *cover* of φ if $\varphi \equiv (\mu_0 \vee \cdots \vee \mu_{m-1})$, i.e., if $(\mu_0 \vee \cdots \vee \mu_{m-1})$ is a disjunctive normal form for φ.

Q is a *minimal cover* of φ if Q is a cover of φ and no proper subset of Q is a cover of φ. □

Observe that for every nonempty set $Q = \{\mu_0, \ldots, \mu_{m-1}\}$ of implicants of φ we have $(\mu_0 \vee \cdots \vee \mu_{m-1}) \models \varphi$. Therefore, Q is a cover for φ if and only if $\varphi \models (\mu_0 \vee \cdots \vee \mu_{m-1})$.

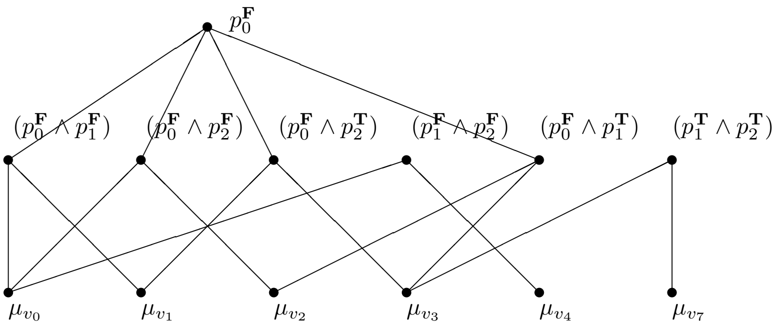

$p_0^{\mathbf{F}}$
$(p_0^{\mathbf{F}} \wedge p_1^{\mathbf{F}})$
$(p_0^{\mathbf{F}} \wedge p_2^{\mathbf{F}})$
$(p_0^{\mathbf{F}} \wedge p_2^{\mathbf{T}})$
$(p_1^{\mathbf{F}} \wedge p_2^{\mathbf{F}})$
$(p_0^{\mathbf{F}} \wedge p_1^{\mathbf{T}})$
$(p_1^{\mathbf{T}} \wedge p_2^{\mathbf{T}})$
μ_{v_0}
μ_{v_1}
μ_{v_2}
μ_{v_3}
μ_{v_4}
μ_{v_7}

Fig. 2.1. Hasse diagram of $(\text{IMPL}(\varphi), \leq)$.

Let φ be a satisfiable formula. The set of all minterms of φ is clearly a cover of φ. However, other covers may exist for φ and it is important from the point of view of circuit designers (as we shall see in Section 2.11) to be able to find covers for formulas which have as few occurrences of statement variables as possible.

Since $(\text{IMPL}(\varphi), \leq)$ is a finite poset, for every $\mu \in \text{IMPL}(\varphi)$, there is a prime implicant μ' such that $\mu \leq \mu'$.

Theorem 2.5.34. *Let φ be a satisfiable formula. A set of implicants of φ, $Q = \{\mu_0, \ldots, \mu_{m-1}\}$, is a cover of φ if and only if for every minterm μ_v of φ there is an implicant $\mu_i \in Q$ such that $\mu_v \leq \mu_i$ (i.e., $\mu_v \models \mu_i$).*

Proof. First, suppose that the condition of the theorem is met. Let $\{v \in \text{TA}_{\text{SV}(\varphi)} \mid v(\varphi) = \mathbf{T}\} = \{v_0, \ldots, v_{k-1}\}$. Then, since

$$\varphi \equiv \bigvee_{0 \leq i \leq k-1} \mu_{v_i} \models \bigvee_{0 \leq l \leq m-1} \mu_l \models \varphi,$$

it is immediate that Q is a cover for φ.

Conversely, let Q be a cover of φ and suppose that $v \in \text{TA}_{\text{SV}(\varphi)}$ is such that $v(\varphi) = \mathbf{T}$ and $\mu_v \not\models \mu_l$ for all l, $0 \leq l \leq m-1$. Then, for every l, there is a truth assignment $w_l \in \text{TA}_{\text{SV}(\varphi)}$ such that $w_l(\mu_v) = \mathbf{T}$ and $w_l(\mu_l) = \mathbf{F}$. By Theorem 2.5.7, we have $w_l = v$ for every l, $0 \leq l \leq m-1$, which implies $v(\mu_l) = \mathbf{F}$ for every l, $0 \leq l \leq m-1$. Since Q is a cover, we obtain that $v(\varphi) = \mathbf{F}$, thereby contradicting the choice of v. □

Corollary 2.5.35. *Let φ be a satisfiable formula. If $Q = \{\mu_0, \ldots, \mu_{m-1}\}$ is a cover for φ and μ is an implicant of φ*

such that $\mu_i < \mu$ *for some* i*,* $0 \leq i \leq m-1$*, then* $Q' = \{\mu_0, \ldots, \mu_{i-1}, \mu, \mu_{i+1}, \ldots, \mu_{m-1}\}$ *is also a cover for* φ*.*

Proof. The statement follows immediately from Theorem 2.5.34. □

Since for every nonprime implicant μ of a satisfiable formula φ there is a prime implicant μ' with $\mu < \mu'$, Corollary 2.5.35 shows that, by any reasonable definition of simplicity, the cover of φ which gives the simplest disjunctive normal form for φ must consist of prime implicants. In general, there are several sets of prime implicants that satisfy the condition of Theorem 2.5.34. The following algorithm allows us to identify the minimal covers consisting of prime implicants for a satisfiable formula φ.

Algorithm 2.5.36 (Quine–McCluskey Tabular Algorithm).
Input: The set of all prime implicants and the set of all minterms of a satisfiable formula $\varphi \in$ PLFORM.
Output: All minimal covers of prime implicants of φ.
Method:

(A) Prepare table T_A having one row for each prime implicant and one column for each minterm of φ. For each prime implicant μ and each minterm μ_v, place a check mark at the intersection of the line of μ and the column of μ_v if $\mu_v \leq \mu$.

(B) Examine the columns of table T_A. If a column contains a single check mark that corresponding prime implicant will be referred to as an *essential prime implicant*. Construct table T_B by eliminating from T_A all essential prime implicants and the columns corresponding to the minterms they dominate.

(C) Construct table T_C as follows: first, if the set of rows of the table T_B in which a column μ_v has check marks strictly includes the set of rows in which some other column $\mu_{v'}$ has check marks, then eliminate column μ_v; second, if, among the remaining columns, several have the same check mark pattern, then retain only one of them.

(D) Construct table T_D by eliminating from T_C all rows that contain no check marks.

(E) The output consists of every minimal set of rows in T_D such that at least one check mark exists in these rows for every column, to each of which we add the set of essential prime implicants determined at Step (B).

Proof of Correctness: If μ is an essential prime implicant for φ, then there exists a minterm of φ that is not dominated by any other implicant. Thus, the set E of essential prime implicants of φ must be contained in any cover by prime implicants of φ. Since, the columns of T_B correspond to the set of minterms not covered by the essential prime implicants of φ, it follows that any set of prime implicants Q is a minimal cover if and only if $Q = E \cup Q'$, where each minterm corresponding to a column of T_B is covered by an element of Q' and Q' is minimal with this property.

If, in table T_B, the column that corresponds to μ_v has check marks on every row in which the column that corresponds to $\mu_{v'}$ has such check marks, this means that any prime implicant that dominates $\mu_{v'}$ also dominates μ_v. Consequently, we can use table T_C in place of T_B when computing the minimal covers as in the previous paragraph. Finally, it is clear that a subset Q' as above cannot include any prime implicant removed in Step (D). Thus the output in Step (E) is correct. □

It should be noted that the set of all minimal covers of prime implicants of φ could be obtained directly from table T_A of the previous algorithm. The point of the algorithm is to make this task somewhat simpler by reducing the number of rows and columns involved.

Example 2.5.37. Let φ be the formula considered in Example 2.5.32. The minterms over $SV(\varphi)$ are

$$\begin{aligned}
\mu_{v_0} &= ((\neg p_0) \wedge (\neg p_1) \wedge (\neg p_2)),\\
\mu_{v_1} &= ((\neg p_0) \wedge (\neg p_1) \wedge p_2),\\
\mu_{v_2} &= ((\neg p_0) \wedge p_1 \wedge (\neg p_2)),\\
\mu_{v_3} &= ((\neg p_0) \wedge p_1 \wedge p_2),\\
\mu_{v_4} &= (p_0 \wedge (\neg p_1) \wedge (\neg p_2)),\\
\mu_{v_7} &= (p_0 \wedge p_1 \wedge p_2).
\end{aligned}$$

The prime implicants of φ are $(\neg p_0)$, $(p_1 \wedge p_2)$, and $((\neg p_1) \wedge (\neg p_2))$.

(A) Table T_A is given by

T_A	μ_{v_0}	μ_{v_1}	μ_{v_2}	μ_{v_3}	μ_{v_4}	μ_{v_7}
$(\neg p_0)$	$\surd$	$\surd$	$\surd$	$\surd$		
$(p_1 \wedge p_2)$				$\surd$		$\surd$
$((\neg p_1) \wedge (\neg p_2))$	$\surd$				$\surd$	

(B) Note that every prime implicant is essential; the column that corresponds to μ_{v_1} contains a unique check mark on the line corresponding to $(\neg p_0)$, the column that corresponds to μ_{v_7} contains a single check mark on the line corresponding to $(p_1 \wedge p_2))$, and the column that corresponds to μ_{v_4} contains a single check mark on the line corresponding to $((\neg p_1) \wedge (\neg p_2))$.

Therefore, table T_B is empty as are all the subsequent tables and the output is the original set of prime implicants. The only disjunctive normal forms for φ that consist of prime implicants are the disjunctions of this set. $\square$

Example 2.5.38. Consider the formula

$$\varphi = ((p_0 \wedge p_1 \wedge p_2) \vee (p_0 \wedge (\neg p_2)) \vee (p_0 \wedge p_1 \wedge (\neg p_3))$$
$$\vee((\neg p_0) \wedge p_2) \vee ((\neg p_0) \wedge (\neg p_1) \wedge (\neg p_2) \wedge (\neg p_3))).$$

Its truth table is

p_0	p_1	p_2	p_3	φ
F	**F**	**F**	**F**	**T**
F	**F**	**F**	**T**	**F**
F	**F**	**T**	**F**	**T**
F	**F**	**T**	**T**	**T**
F	**T**	**F**	**F**	**F**
F	**T**	**F**	**T**	**F**
F	**T**	**T**	**F**	**T**
F	**T**	**T**	**T**	**T**
T	**F**	**F**	**F**	**T**
T	**F**	**F**	**T**	**T**
T	**F**	**T**	**F**	**F**
T	**F**	**T**	**T**	**F**
T	**T**	**F**	**F**	**T**
T	**T**	**F**	**T**	**T**
T	**T**	**T**	**F**	**T**
T	**T**	**T**	**T**	**T**

The set of minterms L_0 obtained in Step (A) of Algorithm 2.5.31 is

$$\begin{aligned}
\mu_{v_0} &= (p_0^{\mathbf{F}} \wedge p_1^{\mathbf{F}} \wedge p_2^{\mathbf{F}} \wedge p_3^{\mathbf{F}}), & \mu_{v_9} &= (p_0^{\mathbf{T}} \wedge p_1^{\mathbf{F}} \wedge p_2^{\mathbf{F}} \wedge p_3^{\mathbf{T}}),\\
\mu_{v_2} &= (p_0^{\mathbf{F}} \wedge p_1^{\mathbf{F}} \wedge p_2^{\mathbf{T}} \wedge p_3^{\mathbf{F}}), & \mu_{v_{12}} &= (p_0^{\mathbf{T}} \wedge p_1^{\mathbf{T}} \wedge p_2^{\mathbf{F}} \wedge p_3^{\mathbf{F}}),\\
\mu_{v_3} &= (p_0^{\mathbf{F}} \wedge p_1^{\mathbf{F}} \wedge p_2^{\mathbf{T}} \wedge p_3^{\mathbf{T}}), & \mu_{v_{13}} &= (p_0^{\mathbf{T}} \wedge p_1^{\mathbf{T}} \wedge p_2^{\mathbf{F}} \wedge p_3^{\mathbf{T}}),\\
\mu_{v_6} &= (p_0^{\mathbf{F}} \wedge p_1^{\mathbf{T}} \wedge p_2^{\mathbf{T}} \wedge p_3^{\mathbf{F}}), & \mu_{v_{14}} &= (p_0^{\mathbf{T}} \wedge p_1^{\mathbf{T}} \wedge p_2^{\mathbf{T}} \wedge p_3^{\mathbf{F}}),\\
\mu_{v_7} &= (p_0^{\mathbf{F}} \wedge p_1^{\mathbf{T}} \wedge p_2^{\mathbf{T}} \wedge p_3^{\mathbf{T}}), & \mu_{v_{15}} &= (p_0^{\mathbf{T}} \wedge p_1^{\mathbf{T}} \wedge p_2^{\mathbf{T}} \wedge p_3^{\mathbf{T}}),\\
\mu_{v_8} &= (p_0^{\mathbf{T}} \wedge p_1^{\mathbf{F}} \wedge p_2^{\mathbf{F}} \wedge p_3^{\mathbf{F}}).
\end{aligned}$$

The layers produced by Algorithm 2.5.31 are shown in Figure 2.2. A check mark next to an implicant means that the implicant was used to produce an implicant in the next layer and therefore is not a prime implicant. Thus, the prime implicants are

$$\begin{gathered}
(p_0^{\mathbf{F}} \wedge p_1^{\mathbf{F}} \wedge p_3^{\mathbf{F}}), (p_1^{\mathbf{F}} \wedge p_2^{\mathbf{F}} \wedge p_3^{\mathbf{F}}), (p_0^{\mathbf{F}} \wedge p_2^{\mathbf{T}}),\\
(p_0^{\mathbf{T}} \wedge p_2^{\mathbf{F}}), (p_1^{\mathbf{T}} \wedge p_2^{\mathbf{T}}), (p_0^{\mathbf{T}} \wedge p_1^{\mathbf{T}}).
\end{gathered}$$

(A) The table T_A is given in Figure 2.3.

(B) The essential prime implicants are the formulas $(p_0^{\mathbf{F}} \wedge p_2^{\mathbf{T}})$ and $(p_0^{\mathbf{T}} \wedge p_2^{\mathbf{F}})$ because the column of μ_{v_3} contains a single check mark which corresponds to the first formula, and the column of μ_{v_9}

L_0	L_1	L_2
$(p_0^{\mathbf{F}} \wedge p_1^{\mathbf{F}} \wedge p_2^{\mathbf{F}} \wedge p_3^{\mathbf{F}})$ $\surd$	$(p_0^{\mathbf{F}} \wedge p_1^{\mathbf{F}} \wedge p_3^{\mathbf{F}})$	$(p_0^{\mathbf{F}} \wedge p_2^{\mathbf{T}})$
$(p_0^{\mathbf{F}} \wedge p_1^{\mathbf{F}} \wedge p_2^{\mathbf{T}} \wedge p_3^{\mathbf{F}})$ $\surd$	$(p_0^{\mathbf{F}} \wedge p_1^{\mathbf{F}} \wedge p_2^{\mathbf{T}})$ $\surd$	$(p_0^{\mathbf{T}} \wedge p_2^{\mathbf{F}})$
$(p_0^{\mathbf{F}} \wedge p_1^{\mathbf{F}} \wedge p_2^{\mathbf{T}} \wedge p_3^{\mathbf{T}})$ $\surd$	$(p_0^{\mathbf{F}} \wedge p_2^{\mathbf{T}} \wedge p_3^{\mathbf{F}})$ $\surd$	$(p_1^{\mathbf{T}} \wedge p_2^{\mathbf{T}})$
$(p_0^{\mathbf{F}} \wedge p_1^{\mathbf{T}} \wedge p_2^{\mathbf{T}} \wedge p_3^{\mathbf{F}})$ $\surd$	$(p_0^{\mathbf{F}} \wedge p_2^{\mathbf{T}} \wedge p_3^{\mathbf{T}})$ $\surd$	$(p_0^{\mathbf{T}} \wedge p_1^{\mathbf{T}})$
$(p_0^{\mathbf{F}} \wedge p_1^{\mathbf{T}} \wedge p_2^{\mathbf{T}} \wedge p_3^{\mathbf{T}})$ $\surd$	$(p_0^{\mathbf{F}} \wedge p_1^{\mathbf{T}} \wedge p_2^{\mathbf{T}})$ $\surd$	
$(p_0^{\mathbf{T}} \wedge p_1^{\mathbf{F}} \wedge p_2^{\mathbf{F}} \wedge p_3^{\mathbf{F}})$ $\surd$	$(p_1^{\mathbf{F}} \wedge p_2^{\mathbf{F}} \wedge p_3^{\mathbf{F}})$	
$(p_0^{\mathbf{T}} \wedge p_1^{\mathbf{F}} \wedge p_2^{\mathbf{F}} \wedge p_3^{\mathbf{T}})$ $\surd$	$(p_0^{\mathbf{T}} \wedge p_1^{\mathbf{F}} \wedge p_2^{\mathbf{F}})$ $\surd$	
$(p_0^{\mathbf{T}} \wedge p_1^{\mathbf{T}} \wedge p_2^{\mathbf{F}} \wedge p_3^{\mathbf{F}})$ $\surd$	$(p_0^{\mathbf{T}} \wedge p_2^{\mathbf{F}} \wedge p_3^{\mathbf{F}})$ $\surd$	
$(p_0^{\mathbf{T}} \wedge p_1^{\mathbf{T}} \wedge p_2^{\mathbf{F}} \wedge p_3^{\mathbf{T}})$ $\surd$	$(p_0^{\mathbf{T}} \wedge p_2^{\mathbf{F}} \wedge p_3^{\mathbf{T}})$ $\surd$	
$(p_0^{\mathbf{T}} \wedge p_1^{\mathbf{T}} \wedge p_2^{\mathbf{T}} \wedge p_3^{\mathbf{F}})$ $\surd$	$(p_0^{\mathbf{T}} \wedge p_1^{\mathbf{T}} \wedge p_2^{\mathbf{F}})$ $\surd$	
$(p_0^{\mathbf{T}} \wedge p_1^{\mathbf{T}} \wedge p_2^{\mathbf{T}} \wedge p_3^{\mathbf{T}})$ $\surd$	$(p_1^{\mathbf{T}} \wedge p_2^{\mathbf{T}} \wedge p_3^{\mathbf{F}})$ $\surd$	
	$(p_0^{\mathbf{T}} \wedge p_1^{\mathbf{T}} \wedge p_3^{\mathbf{F}})$ $\surd$	
	$(p_1^{\mathbf{T}} \wedge p_2^{\mathbf{T}} \wedge p_3^{\mathbf{T}})$ $\surd$	
	$(p_0^{\mathbf{T}} \wedge p_1^{\mathbf{T}} \wedge p_3^{\mathbf{T}})$ $\surd$	
	$(p_0^{\mathbf{T}} \wedge p_1^{\mathbf{T}} \wedge p_2^{\mathbf{T}})$ $\surd$	

Fig. 2.2. Layers of implicants for the formula of Example 2.5.38.

T_A	μ_{v_0}	μ_{v_2}	μ_{v_3}	μ_{v_6}	μ_{v_7}	μ_{v_8}	μ_{v_9}	$\mu_{v_{12}}$	$\mu_{v_{13}}$	$\mu_{v_{14}}$	$\mu_{v_{15}}$
$(p_0^{\mathbf{F}} \wedge p_1^{\mathbf{F}} \wedge p_3^{\mathbf{F}})$	$\surd$	$\surd$									
$(p_1^{\mathbf{F}} \wedge p_2^{\mathbf{F}} \wedge p_3^{\mathbf{F}})$	$\surd$					$\surd$					
$(p_0^{\mathbf{F}} \wedge p_2^{\mathbf{T}})$		$\surd$	$\surd$	$\surd$	$\surd$						
$(p_0^{\mathbf{T}} \wedge p_2^{\mathbf{F}})$						$\surd$	$\surd$	$\surd$	$\surd$		
$(p_1^{\mathbf{T}} \wedge p_2^{\mathbf{T}})$				$\surd$	$\surd$					$\surd$	$\surd$
$(p_0^{\mathbf{T}} \wedge p_1^{\mathbf{T}})$								$\surd$	$\surd$	$\surd$	$\surd$

Fig. 2.3. Table T_A for Example 2.5.38.

contains a single check mark which corresponds to the second formula. The table T_B is given in the following:

T_B	μ_{v_0}	$\mu_{v_{14}}$	$\mu_{v_{15}}$
$(p_0^{\mathbf{F}} \wedge p_1^{\mathbf{F}} \wedge p_3^{\mathbf{F}})$	$\surd$		
$(p_1^{\mathbf{F}} \wedge p_2^{\mathbf{F}} \wedge p_3^{\mathbf{F}})$	$\surd$		
$(p_1^{\mathbf{T}} \wedge p_2^{\mathbf{T}})$		$\surd$	$\surd$
$(p_0^{\mathbf{T}} \wedge p_1^{\mathbf{T}})$		$\surd$	$\surd$

(C) Table T_C can be obtained by eliminating either column $\mu_{v_{14}}$ or column $\mu_{v_{15}}$. We chose to eliminate the latter column and obtain the table

T_C	μ_{v_0}	$\mu_{v_{14}}$
$(p_0^{\mathbf{F}} \wedge p_1^{\mathbf{F}} \wedge p_3^{\mathbf{F}})$	$\surd$	
$(p_1^{\mathbf{F}} \wedge p_2^{\mathbf{F}} \wedge p_3^{\mathbf{F}})$	$\surd$	
$(p_1^{\mathbf{T}} \wedge p_2^{\mathbf{T}})$		$\surd$
$(p_0^{\mathbf{T}} \wedge p_1^{\mathbf{T}})$		$\surd$

(D) Table T_D coincides with T_C since there is no row without check marks.

(E) There are four minimal sets of rows in T_D such that each column contains one check mark in one of the rows. By adding the essential prime implicants, we obtain the following four sets of implicants:

$$\begin{gathered}
\{(p_0^{\mathbf{F}} \wedge p_2^{\mathbf{T}}), (p_0^{\mathbf{T}} \wedge p_2^{\mathbf{F}}), (p_0^{\mathbf{F}} \wedge p_1^{\mathbf{F}} \wedge p_3^{\mathbf{F}}), (p_1^{\mathbf{T}} \wedge p_2^{\mathbf{T}})\},\\
\{(p_0^{\mathbf{F}} \wedge p_2^{\mathbf{T}}), (p_0^{\mathbf{T}} \wedge p_2^{\mathbf{F}}), (p_0^{\mathbf{F}} \wedge p_1^{\mathbf{F}} \wedge p_3^{\mathbf{F}}), (p_0^{\mathbf{T}} \wedge p_1^{\mathbf{T}})\},\\
\{(p_0^{\mathbf{F}} \wedge p_2^{\mathbf{T}}), (p_0^{\mathbf{T}} \wedge p_2^{\mathbf{F}}), (p_1^{\mathbf{F}} \wedge p_2^{\mathbf{F}} \wedge p_3^{\mathbf{F}}), (p_1^{\mathbf{T}} \wedge p_2^{\mathbf{T}})\},\\
\{(p_0^{\mathbf{F}} \wedge p_2^{\mathbf{T}}), (p_0^{\mathbf{T}} \wedge p_2^{\mathbf{F}}), (p_1^{\mathbf{F}} \wedge p_2^{\mathbf{F}} \wedge p_3^{\mathbf{F}}), (p_0^{\mathbf{T}} \wedge p_1^{\mathbf{T}})\}.
\end{gathered}$$

∎

Example 2.5.39. The truth table of the formula $\psi = ((p_0 \leftrightarrow p_1) \vee (p_1 \leftrightarrow p_2))$ is

p_0	p_1	p_2	ψ
F	**F**	**F**	**T**
F	**F**	**T**	**T**
F	**T**	**F**	**F**
F	**T**	**T**	**T**
T	**F**	**F**	**T**
T	**F**	**T**	**F**
T	**T**	**F**	**T**
T	**T**	**T**	**T**

The set of minterms L_0 obtained in Step (A) of Algorithm 2.5.31 is

$$\begin{aligned}
\mu_{v_0} &= ((\neg p_0) \wedge (\neg p_1) \wedge (\neg p_2)),\\
\mu_{v_1} &= ((\neg p_0) \wedge (\neg p_1) \wedge p_2),\\
\mu_{v_3} &= ((\neg p_0) \wedge p_1 \wedge p_2),\\
\mu_{v_4} &= (p_0 \wedge (\neg p_1) \wedge (\neg p_2)),\\
\mu_{v_6} &= (p_0 \wedge p_1 \wedge (\neg p_2)),\\
\mu_{v_7} &= (p_0 \wedge p_1 \wedge p_2).
\end{aligned}$$

The Hasse diagram of the poset $(\text{IMPL}(\psi), \leq)$ is given in Figure 2.4.

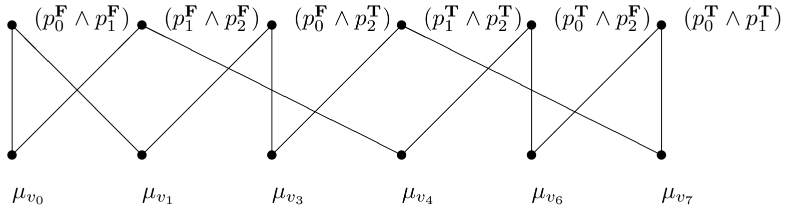

$(p_0^{\mathbf{F}} \wedge p_1^{\mathbf{F}})$
$(p_1^{\mathbf{F}} \wedge p_2^{\mathbf{F}})$
$(p_0^{\mathbf{F}} \wedge p_2^{\mathbf{T}})$
$(p_1^{\mathbf{T}} \wedge p_2^{\mathbf{T}})$
$(p_0^{\mathbf{T}} \wedge p_2^{\mathbf{F}})$
$(p_0^{\mathbf{T}} \wedge p_1^{\mathbf{T}})$
μ_{v_0}
μ_{v_1}
μ_{v_3}
μ_{v_4}
μ_{v_6}
μ_{v_7}

Fig. 2.4. Hasse diagram of $(\text{IMPL}(\psi), \leq)$.

(A) Table T_A having one row for each prime implicant and one column for each minterm over $SV(\varphi)$ is

T_A	μ_{v0}	μ_{v1}	μ_{v3}	μ_{v4}	μ_{v6}	μ_{v7}
$\theta_0 = ((\neg p_0) \wedge (\neg p_1))$	✓	✓				
$\theta_1 = ((\neg p_1) \wedge (\neg p_2))$	✓			✓		
$\theta_2 = ((\neg p_0) \wedge p_2)$		✓	✓			
$\theta_3 = (p_1 \wedge p_2)$			✓			✓
$\theta_4 = (p_0 \wedge (\neg p_2))$				✓	✓	
$\theta_5 = (p_0 \wedge p_1)$					✓	✓

(B) No essential prime implicants exist. Table T_B coincides with T_A.

(C) No columns can be eliminated at Step (C). Therefore, table T_C coincides with table T_B.

(D) No rows can be eliminated in this step. Table T_D coincides with table T_C and, therefore, with table T_A.

(E) There are five minimal sets of rows in T_D such that at least one check mark exists in these rows for every column. They correspond to the sets of implicants

$$\begin{aligned}
&\{((\neg p_0) \wedge (\neg p_1)), (p_1 \wedge p_2), (p_0 \wedge (\neg p_2))\},\\
&\{((\neg p_1) \wedge (\neg p_2)), ((\neg p_0) \wedge p_2), (p_0 \wedge p_1)\},\\
&\{((\neg p_1) \wedge (\neg p_2)), ((\neg p_0) \wedge p_2), (p_1 \wedge p_2), (p_0 \wedge (\neg p_2))\},\\
&\{((\neg p_0) \wedge (\neg p_1)), ((\neg p_0) \wedge p_2), (p_0 \wedge (\neg p_2)), (p_0 \wedge p_1)\},\\
&\{((\neg p_0) \wedge (\neg p_1)), ((\neg p_1) \wedge (\neg p_2)), (p_1 \wedge p_2), (p_0 \wedge p_1)\},
\end{aligned}$$

respectively. By any reasonable measure of "simplicity", the first two sets of implicants will result in the simplest disjunctive normal forms.

$\square$

We gave no explanation in Step (E) of Algorithm 2.5.36 of how to find the minimal covers of prime implicants. Sometimes, this is easily done by inspection (as in Example 2.5.38). In other cases (such as Example 2.5.39), it is not so easy to find the minimal covers and it is useful to have a systematic search procedure. We describe one such in the following after we introduce some helpful notation.

Definition 2.5.40. Let $S \subseteq SV$ and let $Z \subseteq \mathrm{MINTRM}(S)$. The set L_Z is the set $\bigcap\{\mathrm{LIT}(\mu) \mid \mu \in Z\}$.

Let $\mu \in \mathrm{MINTRM}(S)$ and let $T \subseteq SV$ be a set of variables such that $T \cap S \neq \emptyset$. Then, the *projection of μ on T* is the minterm $\mu_T \in \mathrm{MINTRM}(T)$ that is the conjunction of those literals over T that are conjuncts of μ. The set Z_T is $\{\mu_T \mid \mu \in Z\}$.

For $\theta \in \mathrm{MINTRM}(T)$, we define the set of minterms

$$\begin{aligned} B_{\theta,S} &= \{\mu \in \mathrm{MINTRM}(S) \mid \mu_T = \theta\} \\ &= \{\mu \in \mathrm{MINTRM}(S) \mid \mu \models \theta\}. \end{aligned}$$

$\Box$

The equivalence of the two expressions for $B_{\theta,S}$ is immediate from Theorem 2.5.20. When S is clear from context, we will write B_θ for $B_{\theta,S}$.

Example 2.5.41. Let $S = \{p_0, p_1, p_2, p_3\}$ and $T = \{p_1, p_3\}$. The projection of the minterm $\mu = (p_0^{\mathbf{F}} \wedge p_1^{\mathbf{T}} \wedge p_2^{\mathbf{F}} \wedge p_3^{\mathbf{F}}) \in \mathrm{MINTRM}(S)$ on T is the minterm $\theta = (p_1^{\mathbf{T}} \wedge p_3^{\mathbf{F}}) \in \mathrm{MINTRM}(T)$. The set $B_{\theta,S}$ consists of the following minterms:

$$\begin{aligned} &(p_0^{\mathbf{F}} \wedge p_1^{\mathbf{T}} \wedge p_2^{\mathbf{F}} \wedge p_3^{\mathbf{F}}), \\ &(p_0^{\mathbf{F}} \wedge p_1^{\mathbf{T}} \wedge p_2^{\mathbf{T}} \wedge p_3^{\mathbf{F}}), \\ &(p_0^{\mathbf{T}} \wedge p_1^{\mathbf{T}} \wedge p_2^{\mathbf{F}} \wedge p_3^{\mathbf{F}}), \\ &(p_0^{\mathbf{T}} \wedge p_1^{\mathbf{T}} \wedge p_2^{\mathbf{T}} \wedge p_3^{\mathbf{F}}). \end{aligned}$$

$\Box$

Algorithm 2.5.42.
Input: The table T_D produced by Algorithm 2.5.36 for a satisfiable formula $\varphi \in \mathrm{PLFORM}$ with rows corresponding to the prime implicants $\theta_0, \ldots, \theta_{m-1}$ and columns corresponding to minterms $\mu_{v_0}, \ldots, \mu_{v_{n-1}}$.
Output: Every minimal set of rows in T_D such that at least one check mark exists in these rows for every column.
Method:

(A) For $0 \leq j \leq n-1$, define $P_j = \{\theta_i \mid \mu_{v_j} \in B_{\theta_i,S}\}$.
(B) For each tuple $t \in P_0 \times \cdots \times P_{n-1}$, let $C(t)$ be the set of all implicants θ_i that occur in t. Define $\mathcal{C} = \{C(t) \mid t \in P_0 \times \cdots \times P_{n-1}\}$.
(C) Find the minimal-size members of $\mathcal{C}$ and add them to the output. Eliminate these sets and their supersets from $\mathcal{C}$. If $\mathcal{C} = \emptyset$, then halt; otherwise, repeat step (C).

Proof of Correctness: We call a set of rows sufficient if at least one check mark exists in these rows for every column. It is clear that for the value of $\mathcal{C}$ at the end of Step (B), every member of $\mathcal{C}$ is sufficient and each sufficient set of rows contains a subset that belongs to $\mathcal{C}$ (and, therefore, each minimal sufficient set of rows belongs to $\mathcal{C}$). Thus, Step (C), which finds the minimal elements of the original $\mathcal{C}$, yields the desired output. □

The above algorithm is clearly inefficient in that it may require time exponential in the number of minterms. This inefficiency is inherent in the nature of the problem.

Example 2.5.43. In this example, we apply Algorithm 2.5.42 to compute the minimal sets of prime implicants for the formula considered in Example 2.5.39. In Step (A), we obtain the sets

$$\begin{aligned} &P_0 = \{\theta_0, \theta_1\}\ P_3 = \{\theta_1, \theta_4\},\\ &P_1 = \{\theta_0, \theta_2\}\ P_4 = \{\theta_4, \theta_5\},\\ &P_2 = \{\theta_2, \theta_3\}\ P_5 = \{\theta_3, \theta_5\}. \end{aligned}$$

For $t_0 = (\theta_0, \theta_0, \theta_2, \theta_1, \theta_4, \theta_3)$, we have $C(t_0) = \{\theta_0, \theta_1, \theta_2, \theta_3, \theta_4\}$. Continuing this process for the remaining 63 6-tuples contained in the product $P_0 \times \cdots \times P_5$, we obtain the following collection of sets $\mathcal{C}$:

$$\begin{array}{lll} \{\theta_0, \theta_3, \theta_4\} & \{\theta_1, \theta_2, \theta_5\} & \{\theta_0, \theta_1, \theta_2, \theta_5\}\\ \{\theta_0, \theta_1, \theta_3, \theta_4\} & \{\theta_0, \theta_1, \theta_3, \theta_5\} & \{\theta_0, \theta_2, \theta_3, \theta_4\}\\ \{\theta_0, \theta_2, \theta_4, \theta_5\} & \{\theta_0, \theta_3, \theta_4, \theta_5\} & \{\theta_1, \theta_2, \theta_3, \theta_4\}\\ \{\theta_1, \theta_2, \theta_3, \theta_5\} & \{\theta_1, \theta_2, \theta_4, \theta_5\} & \{\theta_0, \theta_1, \theta_2, \theta_3, \theta_4\}\\ \{\theta_0, \theta_1, \theta_2, \theta_3, \theta_5\} & \{\theta_0, \theta_1, \theta_2, \theta_4, \theta_5\} & \{\theta_0, \theta_1, \theta_3, \theta_4, \theta_5\}\\ \{\theta_0, \theta_2, \theta_3, \theta_4, \theta_5\} & \{\theta_1, \theta_2, \theta_3, \theta_4, \theta_5\} & \{\theta_0, \theta_1, \theta_2, \theta_3, \theta_4, \theta_5\}. \end{array}$$

In the first application of Step (C), the minimal-size members of $\mathcal{C}$ are $\{\theta_0, \theta_3, \theta_4\}$ and $\{\theta_1, \theta_2, \theta_5\}$. After eliminating the supersets of these sets, the value of $\mathcal{C}$ for the new application of Step (C) consists of the following sets:

$$\{\theta_0, \theta_1, \theta_3, \theta_5\}, \quad \{\theta_0, \theta_2, \theta_4, \theta_5\}, \quad \{\theta_1, \theta_2, \theta_3, \theta_4\}.$$

All these sets are of minimal-size, so we add these to the output. The new value of $\mathcal{C}$ is $\emptyset$ and the algorithm halts. ▯

We will now present an alternative to the Quine–McCluskey algorithms (Algorithms 2.5.31 and 2.5.36) which is the Karnaugh[3] map. While the Quine–McCluskey algorithms are suitable for any number of statement variables and lend themselves to automation, the Karnaugh map is appropriate for a small number of statement variables (usually less than six) and is inherently visual.

Theorem 2.5.44. *Let φ be a formula and $S = SV(\varphi)$. Let θ, θ' be partial minterms over S:*

(1) $\theta \in \mathrm{IMPL}(\varphi)$ *if and only if* $B_{\theta,S} \subseteq M_\varphi$, *where* M_φ *is the set of minterms of* φ.
(2) $\theta \models \theta'$ *if and only if* $B_{\theta,S} \subseteq B_{\theta',S}$.
(3) *If* $\theta \neq \theta'$, *then* $B_{\theta,S} \neq B_{\theta',S}$.
(4) θ *is a prime implicant of* φ *if and only if* $B_{\theta,S} \subseteq M_\varphi$ *and there is no* $\theta_1 \in PMINTRM(S)$ *such that* $B_{\theta,S} \subset B_{\theta_1,S} \subseteq M_\varphi$.

Proof. For the first part, suppose that $\theta \in \mathrm{IMPL}(\varphi)$ and let $\mu \in B_{\theta,S}$. Since $\mu \models \theta \models \varphi$, it follows from Theorem 2.5.15 that $\mu \in M_\varphi$. Conversely, suppose $B_{\theta,S} \subseteq M_\varphi$ and let $v \in \mathrm{TA}_S$ such that $v(\theta) = \mathbf{T}$. Then, $\mu_v \models \theta$ and thus $\mu_v \in B_{\theta,S} \subseteq M_\varphi$. Consequently, $\mu_v \models \varphi$ which means that $v(\varphi) = \mathbf{T}$. We conclude that $\theta \in \mathrm{IMPL}(\varphi)$.

For the second part, suppose that $\theta \models \theta'$ and let $\mu \in B_{\theta,S}$. We have $\mu \models \theta \models \theta'$, so $\mu \in B_{\theta',S}$. Conversely, suppose that $B_{\theta,S} \subseteq B_{\theta',S}$ and let $v \in \mathrm{TA}_S$ be such that $v(\theta) = \mathbf{T}$. Then, $\mu_v \in B_{\theta,S} \subseteq B_{\theta',S}$, so $\mu_v \models \theta'$ which implies $v(\theta') = \mathbf{T}$. Thus, $\theta \models \theta'$.

For the third part, suppose that $B_{\theta,S} = B_{\theta',S}$. Then, by Part (2), $\theta \models \theta'$ and $\theta' \models \theta$, which implies $\mathrm{LIT}(\theta) = \mathrm{LIT}(\theta')$, by Theorem 2.5.20. Since both θ and θ' are minterms, this amounts to $\theta = \theta'$.

The fourth part follows from the previous parts. □

[3]Maurice Karnaugh was born on October 4, 1924, in New York City and died on November 8, 2022, in New York City. He studied at City College of New York and received his Ph. D. from Yale. He was affiliated with Bell Labs and AT&T and later joined IBM. His research interests span areas from applications of computers in telecommunications to artificial intelligence.

The following algorithm uses a top-down approach to find the prime implicants of a formula.

Algorithm 2.5.45.
Input: A formula φ.
Output: All prime implicants of φ.
Method:

(A) Obtain a standard disjunctive normal form for φ and use it to obtain the set M_φ as defined above.
(B) Set $I = \emptyset$ and $i = 1$.
(C) For each $T \subseteq SV(\varphi)$ with $|T| = i$ and $\theta \in \mathrm{MINTRM}(T)$, if $B_{\theta,S} \subseteq M_\varphi$ and if for every $\theta' \in I$, $B_{\theta,S} \not\subseteq B_{\theta',S}$, then add θ to I.
(D) If $i = n$, halt with I as output. Otherwise, increase i by one and go to (C).

Proof of Correctness: The correctness of the algorithm follows from Theorem 2.5.44. □

Algorithm 2.5.45, as presented above, is quite impractical, as the following example will show. The problem is that even for a small number of variables, there are too many sets of the form $B_{\theta,S}$ to consider. Specifically, there are $3^n - 1$ such sets, where $n = |S|$. However, we presented the algorithm because, together with a display technique which allows for the visual recognition of the sets $B_{\theta,S}$, it serves as the basis for a method which can be used when n is not too large (usually no larger than 5).

Example 2.5.46. Consider the formula φ introduced in Example 2.5.38, where the set M_φ was determined. In Figure 2.5, we display the set M_φ together with the sets $B_{\theta,S}$ for minterms θ that contain one variable in $S = \{p_0, p_1, p_2, p_3\}$. As can be seen from the figure, none of the one-variable minterms is a prime implicant. There are 24 two-variable minterms and 32 three-variable minterms to consider, which makes an exhaustive examination prohibitive. In Figure 2.6, we illustrate the case of two-variable minterms. By computing the remaining 20 columns of this table, the reader could reach the conclusion that the two-variable prime implicants of φ are $\bar{p}_0 \wedge p_2$, $p_0 \wedge \bar{p}_2$, $p_1 \wedge p_2$, and $p_0 \wedge p_1$.

Minterm	M_φ	$B_{\theta,S}$							
		p_0	$\bar{p}_0$	p_1	$\bar{p}_1$	p_2	$\bar{p}_2$	p_3	$\bar{p}_3$
$\bar{p}_0 \wedge \bar{p}_1 \wedge \bar{p}_2 \wedge \bar{p}_3$	✓		✓		✓		✓		✓
$\bar{p}_0 \wedge \bar{p}_1 \wedge \bar{p}_2 \wedge p_3$			✓		✓		✓	✓	
$\bar{p}_0 \wedge \bar{p}_1 \wedge p_2 \wedge \bar{p}_3$	✓		✓		✓	✓			✓
$\bar{p}_0 \wedge \bar{p}_1 \wedge p_2 \wedge p_3$	✓		✓		✓	✓		✓	
$\bar{p}_0 \wedge p_1 \wedge \bar{p}_2 \wedge \bar{p}_3$			✓	✓			✓		✓
$\bar{p}_0 \wedge p_1 \wedge \bar{p}_2 \wedge p_3$			✓	✓			✓	✓	
$\bar{p}_0 \wedge p_1 \wedge p_2 \wedge \bar{p}_3$	✓		✓	✓		✓			✓
$\bar{p}_0 \wedge p_1 \wedge p_2 \wedge p_3$	✓		✓	✓		✓		✓	
$p_0 \wedge \bar{p}_1 \wedge \bar{p}_2 \wedge \bar{p}_3$	✓	✓			✓		✓		✓
$p_0 \wedge \bar{p}_1 \wedge \bar{p}_2 \wedge p_3$	✓	✓			✓		✓	✓	
$p_0 \wedge \bar{p}_1 \wedge p_2 \wedge \bar{p}_3$		✓			✓	✓			✓
$p_0 \wedge \bar{p}_1 \wedge p_2 \wedge p_3$		✓			✓	✓		✓	
$p_0 \wedge p_1 \wedge \bar{p}_2 \wedge \bar{p}_3$	✓	✓		✓			✓		✓
$p_0 \wedge p_1 \wedge \bar{p}_2 \wedge p_3$	✓	✓		✓			✓	✓	
$p_0 \wedge p_1 \wedge p_2 \wedge \bar{p}_3$	✓	✓		✓		✓			✓
$p_0 \wedge p_1 \wedge p_2 \wedge p_3$	✓	✓		✓		✓		✓	

Fig. 2.5. The sets M_φ and $B_{\theta,S}$ for one-variable minterms.

Minterm	M_φ	$B_{\theta,S}$					
		$p_0 \wedge p_1$	$\cdots$	$p_0 \wedge \bar{p}_2$	$p_1 \wedge p_2$	$\cdots$	$\bar{p}_2 \wedge \bar{p}_3$
$\bar{p}_0 \wedge \bar{p}_1 \wedge \bar{p}_2 \wedge \bar{p}_3$	✓						✓
$\bar{p}_0 \wedge \bar{p}_1 \wedge \bar{p}_2 \wedge p_3$							
$\bar{p}_0 \wedge \bar{p}_1 \wedge p_2 \wedge \bar{p}_3$	✓						
$\bar{p}_0 \wedge \bar{p}_1 \wedge p_2 \wedge p_3$	✓						
$\bar{p}_0 \wedge p_1 \wedge \bar{p}_2 \wedge \bar{p}_3$							✓
$\bar{p}_0 \wedge p_1 \wedge \bar{p}_2 \wedge p_3$							
$\bar{p}_0 \wedge p_1 \wedge p_2 \wedge \bar{p}_3$	✓				✓		
$\bar{p}_0 \wedge p_1 \wedge p_2 \wedge p_3$	✓				✓		
$p_0 \wedge \bar{p}_1 \wedge \bar{p}_2 \wedge \bar{p}_3$	✓			✓			✓
$p_0 \wedge \bar{p}_1 \wedge \bar{p}_2 \wedge p_3$	✓			✓			
$p_0 \wedge \bar{p}_1 \wedge p_2 \wedge \bar{p}_3$							
$p_0 \wedge \bar{p}_1 \wedge p_2 \wedge p_3$							
$p_0 \wedge p_1 \wedge \bar{p}_2 \wedge \bar{p}_3$	✓	✓		✓			✓
$p_0 \wedge p_1 \wedge \bar{p}_2 \wedge p_3$	✓	✓		✓			
$p_0 \wedge p_1 \wedge p_2 \wedge \bar{p}_3$	✓	✓			✓		
$p_0 \wedge p_1 \wedge p_2 \wedge p_3$	✓	✓			✓		

Fig. 2.6. The sets M_φ and $B_{\theta,S}$ for two-variable minterms θ.

We can eliminate 13 three-variable minterms θ' because $B_{\theta',S} \subseteq B_{\theta,S}$ for some previously identified two-variable prime implicant θ. Among the remaining 19 three-variable minterms, using the tabular method, we find the prime implicants $\bar{p}_0 \wedge \bar{p}_1 \wedge \bar{p}_3$ and $\bar{p}_1 \wedge \bar{p}_2 \wedge \bar{p}_3$.

Since M_φ is contained in the union of the sets of the form $B_{\theta,S}$ for the prime implicants θ obtained so far, there are no four-variable prime implicants. $\square$

We now proceed to establish the basis for the visual method previously mentioned. There are two steps in this process. The first is to partition the set S of variables into two sets U and V and use a bijection between MINTRM(S) and MINTRM(U) $\times$ MINTRM(V) to give a two-dimensional representation of the sets M_φ and $B_{\theta,S}$. The second is to use a special ordering of the minterms over U and V that makes the sets $B_{\theta,S}$ easy to identify visually.

Theorem 2.5.47. *Let $S \subseteq SV$, $T \subseteq S$, and $\theta, \theta' \in$ MINTRM(T). Then, LIT(θ) $= L_{B_{\theta,S}}$. Furthermore, if $B_{\theta,S} = B_{\theta',S}$, then $\theta = \theta'$.*

Proof. We leave the easy proof of the first equality to the reader. If $B_{\theta,S} = B_{\theta',S}$, then LIT($\theta$) = LIT($\theta'$), which implies $\theta = \theta'$. $\square$

Suppose that $S = \{p_{i_0}, \ldots, p_{i_{n-1}}\}$, where $i_0 < \cdots < i_{n-1}$ and that $U = \{p_{i_0}, \ldots, p_{i_{k-1}}\}$, $V = \{p_{i_k}, \ldots, p_{i_{n-1}}\}$ is a partition of S with $0 < k < n$. There is a bijection $F : \text{MINTRM}(U) \times \text{MINTRM}(V) \longrightarrow \text{MINTRM}(S)$ given by

$$F(p_{i_0}^{a_0} \wedge \cdots \wedge p_{i_{k-1}}^{a_{k-1}}, p_{i_k}^{a_k} \wedge \cdots \wedge p_{i_{n-1}}^{a_{n-1}}) = p_{i_0}^{a_0} \wedge \cdots \wedge p_{i_{n-1}}^{a_{n-1}}.$$

For $\xi \in$ MINTRM(U) and $\zeta \in$ MINTRM(V), we will abuse notation slightly and write $F(\xi, \zeta)$ as $\xi \wedge \zeta$.

Theorem 2.5.48. *Let $S \subseteq SV$, $T, U \subseteq S$, and $\theta \in$ MINTRM(T):*

(1) *We have*

$$(B_{\theta,S})_U = \begin{cases} B_{\theta_U,U} & \text{if } U \cap T \neq \emptyset \\ \text{MINTRM}U & \text{otherwise.} \end{cases}$$

(2) *Let $\{U, V\}$ be a partition of $S = \{p_{i_0}, \ldots, p_{i_{n-1}}\}$, where $i_0 < \cdots < i_{n-1}$. Suppose that $U = \{p_{i_0}, \ldots, p_{i_{k-1}}\}$ and $V = \{p_{i_k}, \ldots, p_{i_{n-1}}\}$. Then,*

$$B_{\theta,S} = \{\xi \wedge \zeta \mid \xi \in (B_{\theta,S})_U, \zeta \in (B_{\theta,S})_V\}.$$

Proof. Suppose that $U \cap T \neq \emptyset$. Let $\eta \in (B_{\theta,S})_U$. Then, $\eta = \mu_U$ for some $\mu \in \text{MINTRM}(S)$ such that $\mu_T = \theta$. Then, $\eta \in \text{MINTRM}(U)$ and since $\eta_{T\cap U} = (\mu_U)_{T\cap U} = \mu_{T\cap U} = (\mu_T)_{T\cap U} = \theta_{T\cap U} = \theta_U$, we have $\eta \in B_{\theta_U,U}$. Conversely, suppose that $\eta \in B_{\theta_U,U}$. We have $\eta \in \text{MINTRM}(U)$ and $\eta_{T\cap U} = \theta_U = \theta_{T\cap U}$. Therefore, there exists a minterm $\mu \in \text{MINTRM}(S)$ such that $\mu_T = \theta$ and $\mu_U = \eta$, which shows that $\eta \in (B_{\theta,S})_U$.

Suppose that $U \cap T = \emptyset$ and let $\eta \in \text{MINTRM}(U)$. The disjointness of U and T implies the existence of $\mu \in \text{MINTRM}(S)$ such that $\mu_U = \eta$ and $\mu_T = \theta$. Thus, $\eta \in (B_{\theta,S})_U$, so $\text{MINTRM}(U) \subseteq (B_{\theta,S})_U$. Since the reverse inclusion is obvious, this completed the proof of the first part of the theorem.

Let now $\{U, V\}$ be a partition of S as defined prior to Theorem 2.5.48, and let $\mu \in B_{\theta,S}$. Since $\mu = \mu_U \wedge \mu_V$ and $\mu_U \in (B_{\theta,S})_U$, $\mu_V \in (B_{\theta,S})_V$, we have $B_{\theta,S} \subseteq \{\xi \wedge \zeta \mid \xi \in (B_{\theta,S})_U, \zeta \in (B_{\theta,S})_V\}$.

Conversely, let $\xi \in (B_{\theta,S})_U, \zeta \in (B_{\theta,S})_V$. Then $\xi = \mu_U$, $\zeta = \mu'_V$, where $\mu, \mu' \in \text{MINTRM}(S)$ and $\mu_T = \mu'_T = \theta$. Therefore, $\xi \wedge \zeta \in \text{MINTRM}(S)$ and

$$\begin{aligned}(\xi \wedge \zeta)_T &= \xi_T \wedge \zeta_T = \xi_{T\cap U} \wedge \zeta_{T\cap V}\\ &= \mu_{T\cap U} \wedge \mu'_{T\cap V} = \theta_{T\cap U} \wedge \theta_{T\cap V} = \theta.\end{aligned}$$

Thus, $\{\xi \wedge \zeta \mid \xi \in (B_{\theta,S})_U, \zeta \in (B_{\theta,S})_V\} \subseteq B_{\theta,S}$. □

Theorem 2.5.49. *Let $S \subseteq SV$, $\{U, V\}$ be a partition of S as above, and let $Z \subseteq$ MINTRM(S). There is $\theta \in$ PMINTRM(S) such that $Z = B_{\theta,S}$ if and only if the following conditions are met:*

(i) *either Z_U = MINTRM(U) or $Z_U = B_{\theta',U}$ for some partial minterm θ' over U,*
(ii) *either Z_V = MINTRM(V) or $Z_V = B_{\theta'',V}$ for some partial minterm θ'' over V,*
(iii) *either $Z_U \neq$ MINTRM(U) or $Z_V \neq$ MINTRM(V),*
(iv) *$Z = \{\xi \wedge \zeta \mid \xi \in Z_U$ and $\zeta \in Z_V\}$.*

Furthermore, if such a θ exists, then one of the following three cases occurs:

- *$\theta = \theta' \wedge \theta''$ if $Z_U = B_{\theta',U}$ for some $\theta' \in$ PMINTRM(U) and $Z_V = B_{\theta'',V}$ for some $\theta'' \in$ PMINTRM(V),*

- *$\theta = \theta'$ if $Z_U = B_{\theta',U}$ for some $\theta' \in PMINTRM(U)$ and $Z_V = MINTRM(V)$,*
- *$\theta = \theta''$ if $Z_V = B_{\theta'',V}$ for some $\theta'' \in PMINTRM(V)$ and $Z_U = MINTRM(U)$.*

Proof. Suppose that there is $\theta \in \text{MINTRM}(T)$, with $T \subseteq S$ such that $Z = B_{\theta,S}$. By Theorem 2.5.48, we have conditions (i), (ii), and (iv). Condition (iii) is also satisfied for otherwise, T would be empty.

Conversely, suppose that conditions (i)–(iv) are satisfied. We need to consider three cases:

(A) Suppose that $Z_U = B_{\theta',U}$ and $Z_V = B_{\theta'',V}$, where $\theta' \in \text{MINTRM}(T')$, $\theta'' \in \text{MINTRM}(T'')$, and $T' \subseteq U, T'' \subseteq V$. Let $\theta = \theta' \wedge \theta'' \in \text{MINTRM}(T' \cup T'')$. By Theorem 2.5.48, we have

$$(B_{\theta,S})_U = B_{\theta_U,U} = B_{\theta',U} = Z_U,$$
$$(B_{\theta,S})_V = B_{\theta_V,V} = B_{\theta'',V} = Z_V,$$

so, again by Theorem 2.5.48, we obtain

$$\begin{aligned} B_{\theta,S} &= \{\xi \wedge \zeta \mid \xi \in (B_{\theta,S})_U, \zeta \in (B_{\theta,S})_V\} \\ &= \{\xi \wedge \zeta \mid \xi \in Z_U, \zeta \in Z_V\} \\ &= Z. \end{aligned}$$

(B) Now, suppose that $Z_U = B_{\theta',U}$, $\theta' \in \text{MINTRM}(T')$, $T' \subseteq U$, and $Z_V = \text{MINTRM}(V)$. Define $\theta = \theta'$. We have

$$(B_{\theta,S})_U = B_{\theta_U,U} = B_{\theta',U} = Z_U,$$
$$(B_{\theta,S})_V = \text{MINTRM}(V) = Z_V,$$

which, as in Case (A), implies $B_{\theta,S} = Z$.

(C) Suppose that $Z_U = \text{MINTRM}(U)$ and $Z_V = B_{\theta'',V}$, where $\theta'' \in \text{MINTRM}(T'')$ and $T'' \subseteq V$. Then, we let $\theta = \theta''$ and we proceed as in Case (B).

The second part of the theorem follows from the proof of the first part and Theorem 2.5.47, which also establishes the uniqueness of θ. □

Corollary 2.5.50. *Let $S \subseteq SV$, $\{U, V\}$ be a partition of S as above, and let $Z \subseteq MINTRM(S)$. If there is $\theta \in PMINTRM(S)$ such that $Z = B_{\theta,S}$, then θ is the minterm defined as follows:*

(1) *If* $L_{Z_U} \neq \emptyset$ *and* $L_{Z_V} \neq \emptyset$*, then* $\theta = \theta' \wedge \theta''$*, where* θ' *is the minterm that is the conjunction of the literals in* L_{Z_U} *and* θ'' *is the minterm that is the conjunction of the literals in* L_{Z_V}*.*
(2) *If* $L_{Z_U} \neq \emptyset$ *and* $L_{Z_V} = \emptyset$*, then* θ *is the conjunction of the literals in* L_{Z_U}*.*
(3) *If* $L_{Z_U} = \emptyset$ *and* $L_{Z_V} \neq \emptyset$*, then* θ *is the conjunction of the literals in* L_{Z_V}*.*

Proof. The result follows immediately from Theorem 2.5.49 and the fact that $L_Z = L_{Z_U} \cup L_{Z_V}$. □

Let φ be a formula such that the set $S = SV(\varphi)$ has at least two elements and let U, V be a partition of S as defined in the notation introduced before Theorem 2.5.48. Let

$$\begin{aligned}\text{MINTRM}(U) &= \{\xi_0, \ldots, \xi_{2^k-1}\},\\ \text{MINTRM}(V) &= \{\zeta_0, \ldots, \zeta_{2^{n-k}-1}\}\end{aligned}$$

be two fixed orderings of MINTRM(U) and MINTRM(V). We denote $F(\xi_i, \zeta_j)$ as μ_{ij}.

A minterm $\mu \in$ MINTRM(S) is represented graphically as in Figure 2.7. Namely, if $\mu = \xi_i \wedge \zeta_j$, we mark the intersection of the row labeled ξ_i and the column labeled ζ_j. A set of minterms over S is represented by marking each square that corresponds to a minterm in the set. We refer to a set of squares that represents a set $B_{\theta,S}$ as

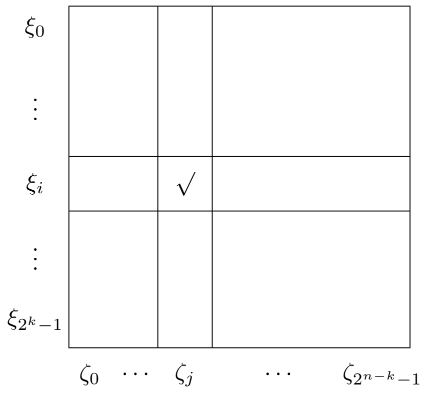


Fig. 2.7. Representation of a minterm.

the *θ-region*. Any set of squares that is the θ-region for some minterm θ is referred to as a *region*.

The ease of finding regions visually can vary greatly depending on the order chosen for MINTRM(U) and MINTRM(V). For example, let $S = \{p_0, p_1, p_2, p_3\}$, $U = \{p_0, p_1\}$, and $V = \{p_2, p_3\}$. Assume initially that

$$\text{MINTRM}(U) = \{\bar{p}_0 \wedge \bar{p}_1, \bar{p}_0 \wedge p_1, p_0 \wedge \bar{p}_1, p_0 \wedge p_1\},$$
$$\text{MINTRM}(V) = \{\bar{p}_2 \wedge \bar{p}_3, \bar{p}_2 \wedge p_3, p_2 \wedge \bar{p}_3, p_2 \wedge p_3\}.$$

In Figure 2.8, we show two 2×2 areas marked with $*$ and $\diamondsuit$. The one marked with asterisks corresponds to the set $B_{p_0 \wedge \bar{p}_2}$, while the one marked with diamonds is not a region. With the current choice of the orderings on MINTRM(U) and MINTRM(V), there is no easily stateable method for identifying those areas that are regions. By choosing the orderings

$$\text{MINTRM}(U) = \{\bar{p}_0 \wedge \bar{p}_1, p_0 \wedge \bar{p}_1, p_0 \wedge p_1, \bar{p}_0 \wedge p_1\},$$
$$\text{MINTRM}(V) = \{\bar{p}_2 \wedge \bar{p}_3, p_2 \wedge \bar{p}_3, p_2 \wedge p_3, \bar{p}_2 \wedge p_3\},$$

as illustrated in Figure 2.9, any 2×2 area is a region. For example, the area marked with asterisks is the region corresponding to $p_0 \wedge p_2$.

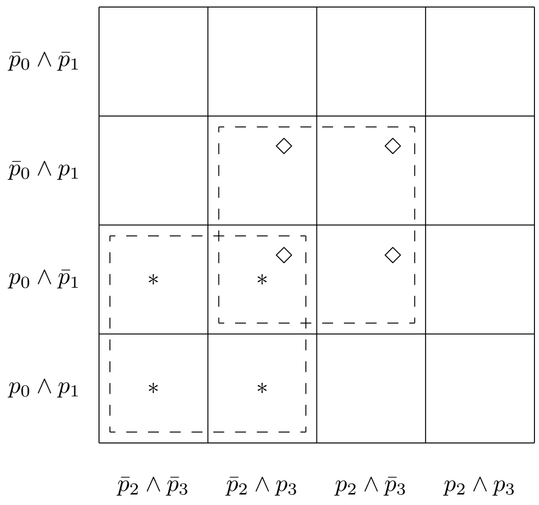


Fig. 2.8. A region and a nonregion.

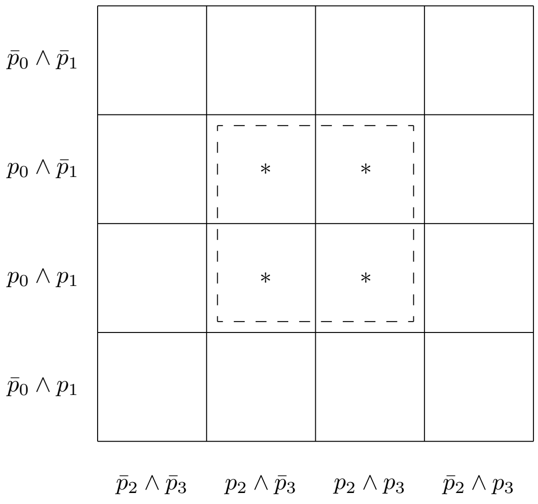


Fig. 2.9. Representation of a region.

We will show next how to systematically generate such orderings and how to identify regions visually.

Definition 2.5.51. Let k be a positive natural number. The *Gray sequence of order k* is the sequence gray^k in $\mathrm{Seq}(\mathbf{Bool}^k)$ of length 2^k defined recursively as follows:

- $\mathsf{gray}^1 = (\mathbf{F}, \mathbf{T})$,
- for $k > 1$,

$$\mathsf{gray}^k = (\mathsf{gray}_0^{k-1}\mathbf{F}, \dots, \mathsf{gray}_{2^{k-1}-1}^{k-1}\mathbf{F}, \mathsf{gray}_{2^{k-1}-1}^{k-1}\mathbf{T}, \dots, \mathsf{gray}_0^{k-1}\mathbf{T}).$$

▯

Theorem 2.5.52. *Let* $k \geq 1$. *Then,* $\{\mathsf{gray}_i^k \mid 0 \leq i \leq 2^k - 1\} = \mathbf{Bool}^k$. *Further,* $\delta(\mathsf{gray}_i^k, \mathsf{gray}_{i+1}^k) = 1$ *for* $0 \leq i \leq 2^k - 2$ *and* $\delta(\mathsf{gray}_{2^k-1}^k, \mathsf{gray}_0^k) = 1$.

Proof. We leave this easy proof by induction on k to the reader. □

Example 2.5.53. Starting from the Gray sequence of order 1, $\mathsf{gray}^1 = (\mathbf{F}, \mathbf{T})$, we obtain the Gray sequence of order 2:

$$\mathsf{gray}^2 = ((\mathbf{F}, \mathbf{F}), (\mathbf{T}, \mathbf{F}), (\mathbf{T}, \mathbf{T}), (\mathbf{F}, \mathbf{T})).$$

Then, the Gray sequence of order 3 is given by

$$\begin{aligned}\mathsf{gray}^3 &= ((\mathbf{F},\mathbf{F},\mathbf{F}),(\mathbf{T},\mathbf{F},\mathbf{F}),(\mathbf{T},\mathbf{T},\mathbf{F}),(\mathbf{F},\mathbf{T},\mathbf{F})\\ &= (\mathbf{F},\mathbf{T},\mathbf{T}),(\mathbf{T},\mathbf{T},\mathbf{T}),(\mathbf{T},\mathbf{F},\mathbf{T}),(\mathbf{F},\mathbf{F},\mathbf{T})).\end{aligned}$$

□

Let $S = \{p_{i_0},\ldots,p_{i_{n-1}}\}$ be a set of statement variables, where $i_0 < \cdots < i_{n-1}$ and let $\vec{a} = (a_0,\ldots,a_{n-1}) \in \mathbf{Bool}^n$. We denote by $v_{\vec{a},S}$ the partial truth assignment given by $v_{\vec{a},S}(p_{i_k}) = a_k$ for every k, $0 \leq k \leq n-1$.

Let $S = \{p_{i_0},\ldots,p_{i_{k-1}}\}$, with $i_0 < \cdots < i_{k-1}$, be a nonempty, finite set of statement variables. The *Gray ordering of MINTRM*(S) is given by

$$\mathrm{MINTRM}(S) = \{\mu_0,\ldots,\mu_{2^k-1}\} \quad \text{where} \quad \mu_i = \mu_{v_{\mathsf{gray}^k_i,W}}.$$

Example 2.5.54. Let $S = \{p_{i_0},p_{i_1},p_{i_2}\}$, where $i_0 < i_1 < i_2$. Denote these variables as p,q,r. The Gray ordering of MINTRM(S) is

$$\begin{aligned}\mathrm{MINTRM}(S) = \{&\bar{p}\wedge\bar{q}\wedge\bar{r},\quad p\wedge\bar{q}\wedge\bar{r},\quad p\wedge q\wedge\bar{r},\quad \bar{p}\wedge q\wedge\bar{r},\\ &\bar{p}\wedge q\wedge r,\quad p\wedge q\wedge r,\quad p\wedge\bar{q}\wedge r,\quad \bar{p}\wedge\bar{q}\wedge r\}.\end{aligned}$$

□

Let φ be a formula such that the set $S = SV(\varphi)$ has at least two elements. Suppose that $S = \{p_{i_0},\ldots,p_{i_{n-1}}\}$, where $i_0 < \cdots < i_{n-1}$. The *Karnaugh map* that corresponds to the partition $U = \{p_{i_0},\ldots,p_{i_{k-1}}\}$, $V = \{p_{i_k},\ldots,p_{i_{n-1}}\}$ of S with $0 < k < n$ is the representation of the set M_φ obtained as above where MINTRM(U) and MINTRM(V) are ordered by the Gray ordering.

The usefulness of Karnaugh maps is based on the ease of identifying the sets $B_{\theta,S}$ and to this end, by Theorem 2.5.49, we need to be able to recognize sets of the form $B_{\theta',U}$ and $B_{\theta'',V}$, when MINTRM(U) and MINTRM(V) are ordered according to the Gray sequence. The following two examples describe such sets when $|U| = 2$ or $|U| = 3$. The same considerations apply to V.

Example 2.5.55. Let $U = \{p_0,p_1\}$. Figure 2.10 shows a table where the columns correspond to the minterms over U, listed in the Gray ordering and the rows correspond to sets of the form $B_{\theta',U}$, where the presence of an asterisk in column μ, row θ' means that $\mu \in B_{\theta',U}$.

$\mu_0 = \bar{p}_0 \wedge \bar{p}_1$
$\mu_1 = p_0 \wedge \bar{p}_1$
$\mu_2 = p_0 \wedge p_1$
$\mu_3 = \bar{p}_0 \wedge p_1$

μ_0	μ_1	μ_2	μ_3	θ'
	*	*		p_0
		*	*	p_1
*			*	$\bar{p}_0$
*	*			$\bar{p}_1$
*				$\bar{p}_0 \wedge \bar{p}_1$
	*			$p_0 \wedge \bar{p}_1$
		*		$p_0 \wedge p_1$
			*	$\bar{p}_0 \wedge p_1$

Fig. 2.10. Representation of the sets $B_{\theta',U}$, where $|U| = 2$.

As can be seen, the sets $B_{\theta',U}$, where θ' contains a single literal, are those sets consisting of two consecutive minterms, where we regard the last and first minterms as being consecutive. We describe this succinctly by saying that these regions consist of two consecutive minterms with wraparound. Obviously, the sets of the form $B_{\theta',U}$, where θ' is a conjunction of two literals, are the sets consisting of a single minterm.

It is clear that the same representation would work for any set U of two variables. ∎

Example 2.5.56. Figure 2.11 shows the sets $B_{\theta',U}$, for $U = \{p_0, p_1, p_2\}$, using a format similar to the one used in Figure 2.10. To save space, the table omits θ' consisting of three literals. The description of the regions is now more intricate; nevertheless, it is simple enough to be practical. We refer to the first four columns (μ_0 through μ_3) as the first half of MINTRM(U) and the remaining

$\mu_0 = \bar{p}_0 \wedge \bar{p}_1 \wedge \bar{p}_2$
$\mu_1 = p_0 \wedge \bar{p}_1 \wedge \bar{p}_2$
$\mu_2 = p_0 \wedge p_1 \wedge \bar{p}_2$
$\mu_3 = \bar{p}_0 \wedge p_1 \wedge \bar{p}_2$
$\mu_4 = \bar{p}_0 \wedge p_1 \wedge p_2$
$\mu_5 = p_0 \wedge p_1 \wedge p_2$
$\mu_6 = p_0 \wedge \bar{p}_1 \wedge p_2$
$\mu_7 = \bar{p}_0 \wedge \bar{p}_1 \wedge p_2$

μ_0	μ_1	μ_2	μ_3	μ_4	μ_5	μ_6	μ_7	θ'
	*	*			*	*		p_0
		*	*	*	*			p_1
*			*	*			*	$\bar{p}_0$
*	*					*	*	$\bar{p}_1$
				*	*	*	*	p_2
*	*	*	*					$\bar{p}_2$
*							*	$\bar{p}_0 \wedge \bar{p}_1$
	*					*		$p_0 \wedge \bar{p}_1$
		*			*			$p_0 \wedge p_1$
			*	*				$\bar{p}_0 \wedge p_1$
					*	*		$p_0 \wedge p_2$
				*	*			$p_1 \wedge p_2$
				*			*	$\bar{p}_0 \wedge p_2$
						*	*	$\bar{p}_1 \wedge p_2$
	*	*						$p_0 \wedge \bar{p}_2$
		*	*					$p_1 \wedge \bar{p}_2$
*			*					$\bar{p}_0 \wedge \bar{p}_2$
*	*							$\bar{p}_1 \wedge \bar{p}_2$
								$\vdots$

Fig. 2.11. Representation of the sets $B_{\theta',U}$, where $|U| = 3$.

columns (μ_4 through μ_7) as the second half. The six regions corresponding to θ' with one literal consist of the two halves plus four regions obtained by taking two consecutive columns in the first half (with wraparound) and their symmetric images in the second half. The twelve regions that correspond to θ' with two literals consist of either two consecutive columns in one half (with wraparound) or one column in one half and its symmetric image in the other half. $\square$

Let S be a set of four variables and let φ be a formula such that $SV(\varphi) = S$. Depending on the partition $\{U, V\}$ we choose for S, we can have three possible Karnaugh maps for φ. The most practical is the one where $|U| = |V| = 2$. The following example describes the regions for such a Karnaugh map.

Example 2.5.57. Let $S = \{p_0, p_1, p_2, p_3\}$ be partitioned as $U = \{p_0, p_1\}$ and $V = \{p_2, p_3\}$. The Gray orderings of MINTRM(U) and MINTRM(V) are given by

$$\begin{aligned} \xi_0 &= \bar{p}_0 \wedge \bar{p}_1, & \zeta_0 &= \bar{p}_2 \wedge \bar{p}_3, \\ \xi_1 &= p_0 \wedge \bar{p}_1, & \zeta_1 &= p_2 \wedge \bar{p}_3, \\ \xi_2 &= p_0 \wedge p_1, & \zeta_2 &= p_2 \wedge p_3, \\ \xi_3 &= \bar{p}_0 \wedge p_1, & \zeta_3 &= \bar{p}_2 \wedge p_3. \end{aligned}$$

There are 80 regions corresponding to the minterms over subsets of S, which are described in the following table. In this table, the term "consecutive" should be understood as "consecutive with wraparound".

Number of literals in θ	Number of such θ	Size of $B_{\theta,S}$	Description of $B_{\theta,S}$
1	8	8	2 consecutive rows or columns
2	24	4	Single rows, single columns, and products of two consecutive rows with two consecutive columns
3	32	2	Two consecutive cells, horizontally or vertically
4	16	1	Single cell

In Figures 2.12–2.14, we give examples of regions and the minterms they correspond to, for one, two, and three literals, respectively.

∎

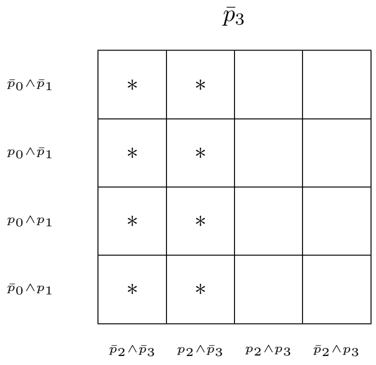


Fig. 2.12. Region for minterm $\bar{p}_3$.

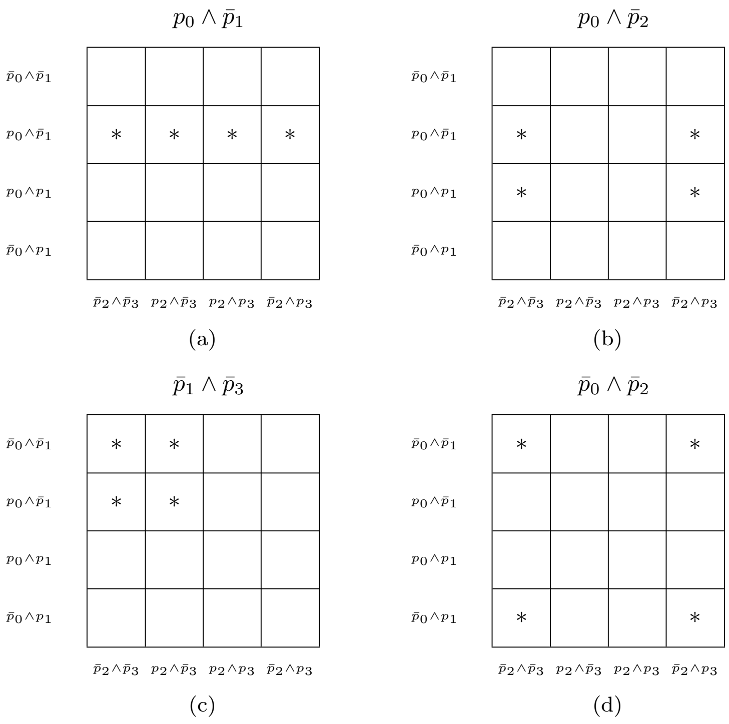


Fig. 2.13. Regions for two-literal minterms.

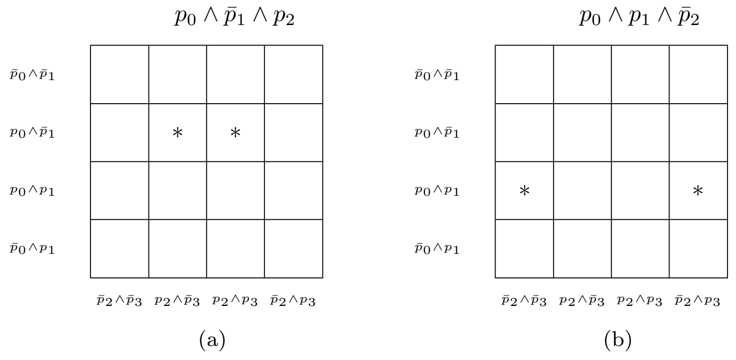


Fig. 2.14. Regions for three-literal minterms.

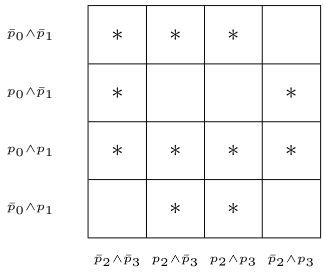


Fig. 2.15. The Karnaugh map for the formula φ.

When the number of variables in the formula φ is less than 5, we now have a practical alternative to Algorithm 2.5.31 to find the prime implicants of φ, using Algorithm 2.5.45 and the Karnaugh map. We illustrate this approach by finding the prime implicants of the formula φ considered in Example 2.5.38.

Example 2.5.58. Figure 2.15 shows the Karnaugh map for the formula φ of Example 2.5.38.

It is clear that M_φ does not contain any two consecutive rows or columns and so φ has no prime implicants that contain one literal. In the next step, we look for regions corresponding to minterms with two literals that are contained in M_φ and we find four such regions: one row and three 2×2 regions, two of which have wraparound (see Figure 2.16).

We can use Theorems 2.5.47 and 2.5.49 to identify the minterms that correspond to each of the regions. For instance, for the region Z in Figure 2.16(a), $L_{Z_U} = \{p_0, p_1\}$, so $Z_U = B_{p_0 \wedge p_1, U}$ and

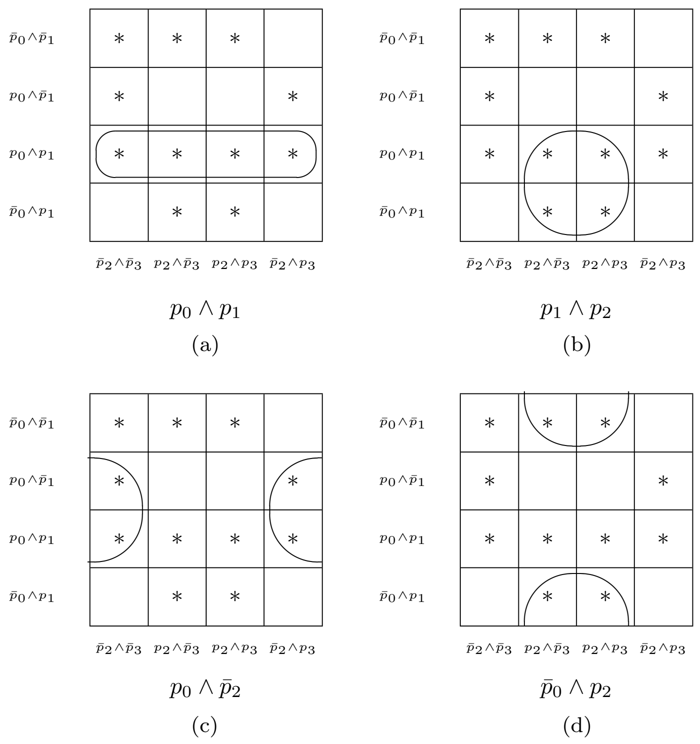


Fig. 2.16. Regions corresponding to two-literal minterms.

$Z_V = \text{MINTRM}(V)$, so $Z = B_{p_0\wedge p_1,S}$. Further, for the region Z in Figure 2.16(c), we have $L_{Z_U} = \{p_0\}$, $L_{Z_V} = \{\bar{p}_2\}$, so $Z = B_{p_0\wedge\bar{p}_2,S}$. Each of these regions corresponds to a prime implicant. Among the regions that are contained in M_φ and correspond to minterms with three literals, only the two shown in Figure 2.17 are not contained in the region of a prime implicant found earlier. Thus, these regions correspond to the last two prime implicants. $\square$

So far, the Karnaugh map has been used as an alternative to the first Quine–McCluskey algorithm (Algorithm 2.5.31) as a method for identifying the prime implicants of a formula. We can also use the Karnaugh map to identify minimal covers. To this end, we locate those cells in M_φ that are contained in a single prime implicant of φ. The corresponding prime implicants are the essential ones and they must appear in every cover of φ. After this, the minimal covers

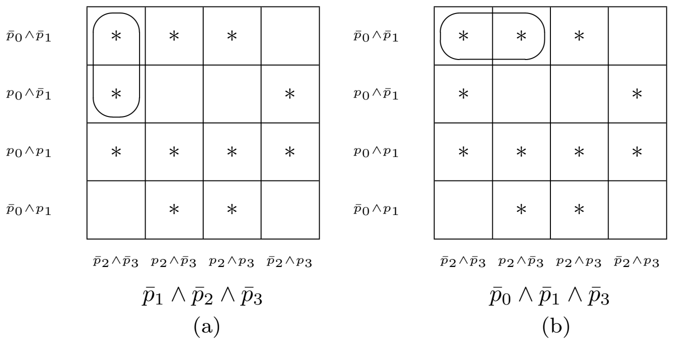

$\bar{p}_0 \wedge \bar{p}_1$
$p_0 \wedge \bar{p}_1$
$p_0 \wedge p_1$
$\bar{p}_0 \wedge p_1$
$\bar{p}_2 \wedge \bar{p}_3$ $p_2 \wedge \bar{p}_3$ $p_2 \wedge p_3$ $\bar{p}_2 \wedge p_3$
$\bar{p}_1 \wedge \bar{p}_2 \wedge \bar{p}_3$
(a)
$\bar{p}_0 \wedge \bar{p}_1 \wedge \bar{p}_3$
(b)

Fig. 2.17. Regions corresponding to three-literal minterms.

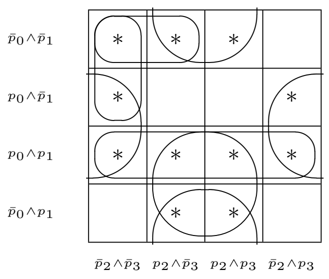

$\bar{p}_0 \wedge \bar{p}_1$
$p_0 \wedge \bar{p}_1$
$p_0 \wedge p_1$
$\bar{p}_0 \wedge p_1$
$\bar{p}_2 \wedge \bar{p}_3$ $p_2 \wedge \bar{p}_3$ $p_2 \wedge p_3$ $\bar{p}_2 \wedge p_3$

Fig. 2.18. The prime implicants of φ.

can often be identified visually. The following example illustrates this.

Example 2.5.59. Figure 2.18 shows all the prime implicants of the formula φ considered most recently in Example 2.5.58. Observe that the cells that correspond to $\bar{p}_0 \wedge \bar{p}_1 \wedge p_2 \wedge p_3$ and $p_0 \wedge \bar{p}_1 \wedge \bar{p}_2 \wedge p_3$ are contained in the regions corresponding to the prime implicants $\bar{p}_0 \wedge p_2$ and $p_0 \wedge \bar{p}_2$, respectively, and in no other region. Thus, these two prime implicants are essential. The cells of M_φ not covered by the essential prime implicants correspond to the minterms

$$\mu_{v_0} = \bar{p}_0 \wedge \bar{p}_1 \wedge \bar{p}_2 \wedge \bar{p}_3,$$
$$\mu_{v_{14}} = p_0 \wedge p_1 \wedge p_2 \wedge \bar{p}_3,$$
$$\mu_{v_{15}} = p_0 \wedge p_1 \wedge p_2 \wedge p_3.$$

The cell μ_{v_0} can be covered with either $\bar{p}_1 \wedge \bar{p}_2 \wedge \bar{p}_3$ or $\bar{p}_0 \wedge \bar{p}_1 \wedge \bar{p}_3$. Both $\mu_{v_{14}}$ and $\mu_{v_{15}}$ can be covered with either $p_0 \wedge p_1$ or $p_1 \wedge p_2$. Thus, we retrieve the minimal covers found in Example 2.5.38. ▯

Theorem 2.5.60. *Let S be a finite, nonempty set of statement variables and let $\tau \in \mathtt{TT}_S$. Then, there is a formula ψ in conjunctive normal form with $SV(\psi) = S$ such that $\tau_{\psi,S} = \tau$. In addition, if $\tau(v) = \mathbf{F}$ for at least one $v \in TA_S$, then such a ψ exists each of whose conjuncts is a maxterm over S.*

Proof. Suppose first that there is at least one truth assignment v over S such that $\tau(v) = \mathbf{F}$. Then, let $\{v_0, \ldots, v_{n-1}\} = \{v \in \mathrm{TA}_S \mid \tau(v) = \mathbf{F}\}$. Define $\psi = \bigwedge_{0 \le i \le n-1} \nu_{v_i}$. For each $w \in \mathrm{TA}_S$, we have

$$\begin{aligned}
\tau_{\psi,S}(w) &= w(\psi) \\
&= w\left(\bigwedge_{0\le i\le n-1} \nu_{v_i}\right) \\
&= \begin{cases} \mathbf{T} & \text{if } w(\nu_{v_i}) = \mathbf{T} \text{ for all } i,\ 0 \le i \le n-1 \\ \mathbf{F} & \text{otherwise} \end{cases} \\
&= \begin{cases} \mathbf{T} & \text{if } w \ne v_i \text{ for all } i,\ 0 \le i \le n-1 \\ \mathbf{F} & \text{otherwise} \end{cases} \\
&= \tau(w).
\end{aligned}$$

Therefore, $\tau_{\psi,S} = \tau$.

If there is no truth assignment $v \in \mathrm{TA}_S$ such that $\tau(v) = \mathbf{F}$, then we can take for ψ the tautology $\psi = (p_{i_0} \vee (\neg p_{i_0}) \vee p_{i_1} \vee \cdots p_{i_{n-1}})$, where $S = \{p_{i_0}, \ldots, p_{i_{n-1}}\}$. □

Note that the proof of Theorem 2.5.60 provides an effective method of producing a conjunctive normal form for a given formula.

Corollary 2.5.61. *Let $\varphi \in$ PLFORM. Then there is a formula ψ in conjunctive normal form such that $\varphi \equiv \psi$. If φ is not a tautology, then such a ψ exists each of whose conjuncts is a maxterm over $SV(\varphi)$.*

Proof. By applying Theorem 2.5.60, to the set $S = SV(\varphi)$ and to the truth table $\tau_\varphi = \tau_{\varphi,S}$, we get a formula ψ in cnf such that

$SV(\psi) = S$ and $\tau_{\psi,S} = \tau_{\varphi,S}$, which implies that $\varphi \equiv \psi$. From the second part of Theorem 2.5.60, it follows that if φ is not a tautology, then we can chose ψ to be a conjunction of maxterms over S. □

A formula ψ which is in conjunctive normal form and is logically equivalent to φ is called a *conjunctive normal form* for φ. If, in addition, ψ consists of distinct maxterms over $SV(\varphi)$, we refer to ψ as a *standard conjunctive normal form* for φ. Observe that any formula obtained from a standard conjunctive normal form for φ by permuting the maxterms is also a standard conjunctive normal form for φ.

Example 2.5.62. Let τ be the truth table considered in Example 2.5.11. A formula in conjunctive normal form whose truth table is τ is

$$\begin{aligned}\psi = ((p_0 \vee p_4 \vee p_6) \wedge (p_0 \vee (\neg p_4) \vee p_6) \wedge (p_0 \vee (\neg p_4) \vee (\neg p_6))\\ \wedge((\neg p_0) \vee p_4 \vee p_6) \wedge ((\neg p_0) \vee (\neg p_4) \vee p_6)).\end{aligned}$$

We have thus found a conjunctive normal form for the formula $((p_4 \to p_0) \wedge p_6)$. ▯

Definition 2.5.63. A formula $\varphi = \bigwedge_{0 \leq i \leq n-1} \bigvee_{0 \leq j \leq m_i - 1} \ell_{ij}$ in conjunctive normal form, where each ℓ_{ij} is a literal, is a *Horn*[4] *formula* if every conjunct $\bigvee_{0 \leq j \leq m_i - 1} \ell_{ij}$ contains at most one positive literal. ▯

Example 2.5.64. The formula

$$\psi = (((\neg p_0) \vee (\neg p_1) \vee p_2) \wedge p_1 \wedge ((\neg p_2) \vee (\neg p_1) \vee (\neg p_0)) \wedge (\neg p_2))$$

is a Horn formula, while $\theta = (p_0 \vee p_1)$ is not. ▯

For checking satisfiability for arbitrary formulas, even ones in conjunctive normal form, we have no method essentially more efficient than the construction of their truth table. For Horn formulas,

[4]Alfred Horn, born in 1918 in New York City and deceased in Los Angeles in 2001, was an American mathematician whose main contributions were in the areas of logic, lattice theory, and universal algebra. Horn was educated at City College of New York and New York University and received his Ph.D. from the University of California, Berkeley. He taught at the University of California at Los Angeles.

however, the following algorithm gives an efficient test for satisfiability. The algorithm involves a process of marking variables that occur in the formula.

Algorithm 2.5.65 (Satisfiability of Horn Formulas).
Input: A Horn formula φ.
Output: "Yes" if φ is satisfiable and "No" otherwise.
Method:

(A) If there is a conjunct of φ containing only negative literals whose variables are marked, then output "No" and stop.
(B) If there is a conjunct of φ containing a positive literal whose variable is unmarked and all variables occurring in the negative literals of the conjunct (if any) are marked, then, for each such conjunct, mark the variable of the positive literal of the conjunct and then go to step (A).
(C) Output "Yes" and stop.

Proof of Correctness: The algorithm always halts because each time step (B) is executed (except the last time) a new variable is marked and there are only finitely many variables which occur in φ.

Suppose that v is a truth assignment such that $v(\varphi) = \mathbf{T}$. We claim that for every marked variable p, we have $v(p) = \mathbf{T}$. We prove this claim by course-of-values induction on the order in which the variables are marked. Suppose that the variable p is marked at some point in the algorithm and $v(q) = \mathbf{T}$ for all variables q marked earlier in the algorithm. Then, p occurs positively in some conjunct of φ in which all variables in negative literals have been already marked. Because φ is a Horn formula, p is the only variable appearing in a positive literal of that conjunct. By the inductive hypothesis, for all of the other variables q occurring in the conjunct, $v(q) = \mathbf{T}$. Since these variables occur in negative literals, we must have $v(p) = \mathbf{T}$.

If the algorithm halts with output "No", then there is a conjunct of φ which consists of negative literals whose variables are marked. Suppose that φ is satisfiable. Then, by the previous argument, for any truth assignment v such that $v(\varphi) = \mathbf{T}$, we would have $v(p) = \mathbf{T}$ for all variables occurring in this conjunct and this would imply that

v makes the conjunct (and therefore the formula) false. Hence, there can be no such v and φ is unsatisfiable.

Now suppose that the algorithm outputs "Yes". Let $v_0 \in \mathrm{TA}_{SV(\varphi)}$ be defined by $v_0(p) = \mathbf{T}$ if and only if p is a marked variable. We show that $v_0(\varphi) = \mathbf{T}$ and, therefore, φ is satisfiable. Let ψ be a conjunct of φ. We consider two cases. In the first case, suppose that ψ contains a positive literal p. If p is marked, then $v_0(p) = \mathbf{T}$, so $v_0(\psi) = \mathbf{T}$. If p is not marked, then, there must be a variable q in φ which occurs in a negative literal of this conjunct and is unmarked. Then, $v_0(q) = \mathbf{F}$, so $v_0(\psi) = \mathbf{T}$. In the second case, ψ does not contain any positive literal. Observe that step (C) can be reached only after executing steps (A) and (B) in succession and this means that at the last execution of step (A), ψ could not contain only marked variables. Therefore, ψ contains at least one unmarked variable in a negative literal and thus $v_0(\psi) = \mathbf{T}$. $\square$

Example 2.5.66. Consider the Horn formula

$$\varphi = (p_0 \wedge ((\neg p_0) \vee p_1) \wedge ((\neg p_1) \vee p_2) \wedge ((\neg p_2) \vee (\neg p_0))).$$

The application of Algorithm 2.5.65 to φ is given in the following table. Marked variables are underlined.

Step	Effect
A	go to B
B	$(\underline{p_0} \wedge ((\neg \underline{p_0}) \vee p_1) \wedge ((\neg p_1) \vee p_2) \wedge ((\neg p_2) \vee (\neg \underline{p_0})))$ (mark p_0 and go to A)
A	go to B
B	$(\underline{p_0} \wedge ((\neg \underline{p_0}) \vee \underline{p_1}) \wedge ((\neg \underline{p_1}) \vee p_2) \wedge ((\neg p_2) \vee (\neg \underline{p_0})))$ (mark p_1 and go to A)
A	go to B
B	$(\underline{p_0} \wedge ((\neg \underline{p_0}) \vee \underline{p_1}) \wedge ((\neg \underline{p_1}) \vee \underline{p_2}) \wedge ((\neg \underline{p_2}) \vee (\neg \underline{p_0})))$ (mark p_2 and go to A)
A	output "No" and stop

We conclude that φ is not satisfiable.

Let ψ be the Horn formula considered in Example 2.5.64:

$$\psi = (((\neg p_0) \vee (\neg p_1) \vee p_2) \wedge p_1 \wedge ((\neg p_2) \vee (\neg p_1) \vee (\neg p_0)) \wedge (\neg p_2)).$$

By applying Algorithm 2.5.65, we conclude that ψ is satisfiable.

Step	Effect
A	go to B
B	$(((\neg p_0) \vee (\neg \underline{p_1}) \vee p_2) \wedge \underline{p_1} \wedge ((\neg p_2) \vee (\neg \underline{p_1}) \vee (\neg p_0)) \wedge (\neg p_2))$ (mark p_1 and go to A)
A	go to B
B	no variable can be marked; go to C
C	output "Yes" and stop

▯

If S is a set of statement variables and $v, w \in \mathrm{TA}_S$, we define $v \leq w$ to mean that $w(p) = \mathbf{T}$ for every $p \in S$ such that $v(p) = \mathbf{T}$. It is easy to verify that "$\leq$" is a partial order on TA_S.

Theorem 2.5.67. *For every satisfiable Horn formula φ, there exists a truth assignment $v_0 \in TA_{SV(\varphi)}$ such that $v_0(\varphi) = \mathbf{T}$ and for every truth assignment $v \in TA_{SV(\varphi)}$ with $v(\varphi) = \mathbf{T}$, we have $v_0 \leq v$.*

Proof. The correctness proof of Algorithm 2.5.65 contains the argument. □

Corollary 2.5.68. *For every satisfiable Horn formula φ, there exists a truth assignment $w_0 \in TA$ such that $w_0(\varphi) = \mathbf{T}$ and for every truth assignment $w \in TA$ with $w(\varphi) = \mathbf{T}$, we have $w_0 \leq w$.*

Proof. Let v_0 be as in Theorem 2.5.67. Define w_0 by

$$w_0(p) = \begin{cases} v_0(p) & \text{if } p \in SV(\varphi) \\ \mathbf{F} & \text{otherwise.} \end{cases}$$

Since w_0 and v_0 agree on all variables in φ, we have $w_0(\varphi) = v_0(\varphi) = \mathbf{T}$. Let w be a truth assignment such that $w(\varphi) = \mathbf{T}$ and let $v = w \restriction SV(\varphi)$. Again, since v agrees with w on all variables in φ, we have $v(\varphi) = \mathbf{T}$. Theorem 2.5.67 implies that $v_0 \leq v$ and this gives $w_0 \leq w$. □

Note that if $\varphi \equiv \psi$ and φ is a Horn formula, then Corollary 2.5.68 applies to ψ.

Example 2.5.69. Not every formula is equivalent to a Horn formula. For instance, let $\varphi = (p_0 \vee p_1)$. Consider the truth assignments u and u' given by

$$u(p) = \begin{cases} \mathbf{T} & \text{if } p = p_0 \\ \mathbf{F} & \text{otherwise} \end{cases} \quad \text{and} \quad u'(p) = \begin{cases} \mathbf{T} & \text{if } p = p_1 \\ \mathbf{F} & \text{otherwise.} \end{cases}$$

Note that $u(\varphi) = u'(\varphi) = \mathbf{T}$. If φ were equivalent to a Horn formula, this would imply the existence of a truth assignment w_0 such that $w_0 \leq u, u'$ and $w_0(\varphi) = \mathbf{T}$. But, the single truth assignment w_0 such that $w_0 \leq u, u'$ is the one given by $w_0(p) = \mathbf{F}$ for all $p \in SV$ and for this w_0, $w_0(\varphi) = \mathbf{F}$. ▯

2.6 Substitutions and Formulas

In this section, we examine the effects that substitutions have on formulas of propositional logic. Naturally, we are interested in substitutions that replace statement variables with formulas.

Definition 2.6.1. A *propositional substitution* is a substitution defined on SV with values in PLFORM. ▯

For the remainder of this section, we will use the word "substitution" rather than "propositional substitution". (Notations and conventions involving substitutions can be found in Section 1.2 of Chapter 1.)

Lemma 2.6.2. *Let s be a substitution. Then, for the extension $\overline{s}$ of s, we have*

$$\overline{s}(v) = s(v),$$
$$\overline{s}((\neg\varphi)) = (\neg\overline{s}(\varphi)),$$
$$\overline{s}((\varphi\ C\ \psi)) = (\overline{s}(\varphi)\ C\ \overline{s}(\psi)),$$

for every statement variable v, all formulas φ and ψ, and every binary connective symbol C.

Proof. The argument follows from the definition of $\overline{s}$ and Theorem 1.2.11. □

Theorem 2.6.3. *If s is a substitution and φ is a formula, then $\overline{s}(\varphi)$ is a formula.*

Proof. Using Lemma 2.6.2, this is easily shown by induction on formulas. The details are left to the reader. □

We define the effect of a substitution s on a signed formula $b\varphi$ by $\bar{s}(b\varphi) = b\bar{s}(\varphi)$.

Example 2.6.4. Let s be a substitution such that $s(p_0) = (p_2 \wedge p_3)$ and $s(p_1) = (\neg p_3)$. If $\varphi = ((p_0 \wedge p_1) \vee p_0)$, then

$$s(\varphi) = (((p_2 \wedge p_3) \wedge (\neg p_3)) \vee (p_2 \wedge p_3)),$$
$$s(\mathbf{F}\varphi) = \mathbf{F}(((p_2 \wedge p_3) \wedge (\neg p_3)) \vee (p_2 \wedge p_3)).$$

▯

Theorem 2.6.5. *Let s be a substitution. Then for every formula φ,*

$$SV(s(\varphi)) = \bigcup_{p \in SV(\varphi)} SV(s(p)).$$

Proof. The argument is by induction on the definition of formulas and is left to the reader. □

The following theorem shows that the truth value under a given truth assignment of a formula obtained by substitution from another is the same as the truth value of the original formula under a modified truth assignment.

Theorem 2.6.6. *Let s be a substitution and v be a truth assignment. For every formula φ, we have $v(s(\varphi)) = v'(\varphi)$, where v' is the truth assignment given by $v'(p) = v(s(p))$ for every $p \in SV$.*

Proof. The proof is by induction of φ. The basis step ($\varphi = p$) is immediate by the definition of v'. Suppose that the statement holds for α and β and let $\varphi = (\alpha C \beta)$, where C is a binary connective symbol. Then, we can write

$$\begin{aligned} v(s(\varphi)) &= v((s(\alpha) C s(\beta))) \\ &= f_C(v(s(\alpha)), v(s(\beta))) \\ &= f_C(v'(\alpha), v'(\beta)) \\ &= v'(\varphi). \end{aligned}$$

We leave to the reader the case $\varphi = (\neg\alpha)$. □

Corollary 2.6.7. *Let s be a substitution and v be a partial truth assignment over a set S. For every formula φ such that $SV(s(\varphi)) \subseteq S$, we have $v(s(\varphi)) = v'(\varphi)$ for every partial truth assignment v' such that* $\mathrm{Dom}(v') \supseteq SV(\varphi)$ *and $v'(p) = v(s(p))$ for every $p \in SV(\varphi)$.*

Proof. Let v_1 be a truth assignment that extends v and let v'_1 be the truth assignment defined by $v'_1(p) = v_1(s(p))$, for $p \in SV$. By Definition 2.3.24, $v(s(\varphi)) = v_1(s(\varphi))$. By Theorem 2.6.6, $v_1(s(\varphi)) = v'_1(\varphi)$. If $p \in SV(\varphi)$, we have $v'(p) = v(s(p)) = v_1(s(p)) = v'_1(p)$. Thus, by Corollary 2.3.25, $v'_1(\varphi) = v'(\varphi)$. □

Corollary 2.6.8. *If φ is a tautology, then $s(\varphi)$ is also a tautology for every substitution s.*

Proof. For every truth assignment v, we have $v(s(\varphi)) = v'(\varphi) = \mathbf{T}$, where $v'(p) = v(s(p))$ for $p \in SV$, because of Theorem 2.6.6. □

Example 2.6.9. The formula $\alpha = ((p \to (\neg p)) \to (\neg p))$ is a tautology, as shown by the following table:

$v(p)$	$v((p \to (\neg p))$	$v(\alpha)$
T	**F**	**T**
F	**T**	**T**

Consequently, by Corollary 2.6.8, the formula $((\varphi \to (\neg\varphi)) \to (\neg\varphi))$ is a tautology for every formula φ. ▯

Substitutions preserve logical equivalence of formulas according to the following corollary.

Corollary 2.6.10. *If φ, ψ are logically equivalent formulas, then $s(\varphi)$ and $s(\psi)$ are also logically equivalent for every substitution s.*

Proof. Let v be a truth assignment. Because of Theorem 2.6.6, we have $v(s(\theta)) = v'(\theta)$ for every formula θ, where v' is the truth assignment given by $v'(p) = v(s(p))$ for every $p \in SV$. This allows us to write

$$\begin{aligned} v(s(\varphi)) &= v'(\varphi) \\ &= v'(\psi) \qquad \text{since } \varphi, \psi \text{ are logically equivalent} \\ &= v(s(\psi)), \end{aligned}$$

for every truth assignment v, which proves that $s(\varphi)$ and $s(\psi)$ are logically equivalent. □

Corollary 2.6.10 and Theorem 2.3.30 allow us to obtain the list of equivalent formulas given in the following theorem.

Theorem 2.6.11. *Let φ, ψ, and θ be formulas. Then, we have*

$$\begin{aligned}
(\varphi \wedge \varphi) &\equiv \varphi && (\textit{idempotency of } \wedge),\\
(\varphi \vee \varphi) &\equiv \varphi && (\text{idempotency of } \vee),\\
(\varphi \wedge \psi) &\equiv (\psi \wedge \varphi) && (\text{commutativity of } \wedge),\\
(\varphi \vee \psi) &\equiv (\psi \vee \varphi) && (\text{commutativity of } \vee),\\
(\varphi \wedge (\psi \wedge \theta)) &\equiv ((\varphi \wedge \psi) \wedge \theta) && (\text{associativity of } \wedge),\\
(\varphi \vee (\psi \vee \theta)) &\equiv ((\varphi \vee \psi) \vee \theta) && (\text{associativity of } \vee),\\
(\neg(\neg\varphi)) &\equiv \varphi && (\text{double negation}),\\
(\varphi \wedge (\varphi \vee \psi)) &\equiv \varphi && (\text{absorption laws}),\\
(\varphi \vee (\varphi \wedge \psi)) &\equiv \varphi && \text{''},\\
(\varphi \wedge (\psi \vee \theta)) &\equiv ((\varphi \wedge \psi) \vee (\varphi \wedge \theta)) && (\text{distributivity laws}),\\
((\psi \vee \theta) \wedge \varphi) &\equiv ((\psi \wedge \varphi) \vee (\theta \wedge \varphi)) && \text{''},\\
(\varphi \vee (\psi \wedge \theta)) &\equiv ((\varphi \vee \psi) \wedge (\varphi \vee \theta)) && \text{''},\\
((\psi \wedge \theta) \vee \varphi) &\equiv ((\psi \vee \varphi) \wedge (\theta \vee \varphi)) && \text{''},\\
(\neg(\varphi \vee \psi)) &\equiv ((\neg\varphi) \wedge (\neg\psi)) && (\text{De Morgan's laws}),\\
(\neg(\varphi \wedge \psi)) &\equiv ((\neg\varphi) \vee (\neg\psi)) && \text{''},\\
(\varphi \to \psi) &\equiv ((\neg\varphi) \vee \psi), &&\\
(\varphi \leftrightarrow \psi) &\equiv ((\varphi \to \psi) \wedge (\psi \to \varphi)). &&
\end{aligned}$$

Proof. Let φ, ψ, and θ be three formulas and let s be the substitution defined by $s(p_0) = \varphi$, $s(p_1) = \psi$, $s(p_2) = \theta$, and $s(p_i) = p_i$ for $i > 2$. Applying this substitution to the pairs of equivalent formulas from Theorem 2.3.30 and taking into account Corollary 2.6.10, we obtain immediately the equivalent formulas mentioned in this theorem. □

Theorem 2.6.12 (Replacement Theorem for PLFORM**).** *Let φ be a formula. If α, β are logically equivalent formulas and (α, i) is an occurrence of α in φ, then φ is logically equivalent to* `replace` $(\varphi, (\alpha, i), \beta)$.

Proof. Note that by Theorem 2.2.15, `replace` $(\varphi, (\alpha, i), \beta)$ is a formula. If $\varphi = \alpha$, then $i = 0$ and `replace` $(\varphi, (\alpha, i), \beta) = \beta$, so the result follows immediately.

The argument is by induction on the definition of φ. If φ is a statement variable, then $\varphi = \alpha$ and we are done by the previous remark.

Suppose that $\varphi = (\neg\varphi_1)$, where φ_1 is a formula for which the statement holds. If $\varphi = \alpha$, the conclusion is immediate. If $\varphi \neq \alpha$, we have shown in the proof of Theorem 2.2.15 that

$$\texttt{replace}\,(\varphi, (\alpha, i), \beta) = (\neg\,\texttt{replace}\,(\varphi_1, (\alpha, i-2), \beta)).$$

We have $\varphi_1 \equiv \texttt{replace}\,(\varphi_1, (\alpha, i-2), \beta)$ by the inductive hypothesis. Therefore, by Theorem 2.3.14, we have

$$\begin{aligned}\varphi = (\neg\varphi_1) &\equiv (\neg\,\texttt{replace}\,(\varphi_1, (\alpha, i-2), \beta)) \\ &= \texttt{replace}\,(\varphi, (\alpha, i), \beta).\end{aligned}$$

We leave to the reader the argument for the last case, that is, for the case when $\varphi = (\varphi_1 C \varphi_2)$, where C is a binary connective symbol and $\varphi_1, \varphi_2 \in$ PLFORM. □

2.7 Truth Sets and Hintikka Sets

In this section, we first introduce the notion of constituent of a formula. A formula that is not a literal has one constituent for each distinct way that it can be satisfied. Further, since a constituent of a formula φ consists of subformulas or negated subformulas of φ (that is of members of the set $U(\{\varphi\})$, where U is the function defined in Theorem 2.2.14), constituents of formulas can be built by "analyzing" their structure. This property of constituents is known as their *analyticity*. Constituents allow us to introduce truth sets and Hintikka[5] sets, a generalization of truth sets.

We will see that every satisfiable set Γ is contained in a truth set Γ'. This is interesting because truth sets are not merely satisfiable, but they explicitly specify the satisfying truth assignment.

[5]Jaakko Hintikka was born in Helsinki, Finland, in 1929 and died in Porvoo, Finland, in 2015. He received his Ph.D. from the University of Helsinki in 1956. Hintikka's contributions were in general philosophy, philosophy of mathematics, and logic. He taught at the University of Helsinki, Florida State University, Stanford, and Boston University.

However, the truth set Γ' may contain formulas that have no relation to the formulas in Γ. Hintikka sets maintain the property of explicitly specifying which truth assignments satisfy them. (In general, several satisfying truth assignments exist for a Hintikka set.) The advantage of Hintikka sets over truth sets is that every satisfiable set Γ can be extended to a Hintikka set Γ' such that the formulas of Γ' have a simple syntactic relationship to the formulas of Γ.

Definition 2.7.1. The function

$$\mathtt{d} : \text{PLFORM} - \{p, (\neg p) \mid p \in SV\} \longrightarrow \text{Seq}(\mathcal{P}(\text{PLFORM}))$$

is given by the following table:

Formula α	$\mathtt{d}(\alpha)$
$(\neg(\neg\varphi))$	$(\{\varphi\})$
$(\varphi \wedge \psi)$	$(\{\varphi, \psi\})$
$(\neg(\varphi \wedge \psi))$	$(\{(\neg\varphi)\}, \{(\neg\psi)\})$
$(\varphi \vee \psi)$	$(\{\varphi\}, \{\psi\})$
$(\neg(\varphi \vee \psi))$	$(\{(\neg\varphi), (\neg\psi)\})$
$(\varphi \rightarrow \psi)$	$(\{(\neg\varphi)\}, \{\psi\})$
$(\neg(\varphi \rightarrow \psi))$	$(\{\varphi, (\neg\psi)\})$
$(\varphi \leftrightarrow \psi)$	$(\{\varphi, \psi\}, \{(\neg\varphi), (\neg\psi)\})$
$(\neg(\varphi \leftrightarrow \psi))$	$(\{\varphi, (\neg\psi)\}, \{(\neg\varphi), \psi\})$

The *constituent sequence* of φ is the sequence $\mathtt{d}(\varphi)$. The *constituent set* of φ is the set $\mathtt{D}(\varphi)$ that consists of all sets of formulas that occur in $\mathtt{d}(\varphi)$. Every such set of formulas is called a *constituent* of φ. ▯

Theorem 2.7.2. *If φ is not a literal, then a truth assignment v satisfies φ if and only if it satisfies at least one of the constituents of φ.*

Proof. We leave the easy verification to the reader. □

Note that the length of each formula in a constituent of a formula φ is smaller than $|\varphi|$. Also, the formulas in any constituent K of φ are proper subformulas of φ or negations of proper subformulas of φ. In other words, $K \subseteq U(\{\varphi\})$, where U is the mapping defined in Theorem 2.2.14.

Theorem 2.7.3. *Let Γ be a set of formulas such that for every formula φ, $(\neg\varphi) \in \Gamma$ if and only if $\varphi \notin \Gamma$ and let C be a* *binary connective symbol. Then, the following pairs of statements are equivalent, where in each condition φ ranges over the set $\{(\alpha C \beta) \mid \alpha, \beta \in \text{PLFORM}\}$:*

- C_{pd}*:* $\varphi \in \Gamma$ *implies* $K \subseteq \Gamma$ *for some* $K \in \mathtt{D}(\varphi)$*,*
- C_{npu}*:* $H \subseteq \Gamma$ *for some* $H \in \mathtt{D}((\neg\varphi))$ *implies* $(\neg\varphi) \in \Gamma$

and

- C_{pu}*:* $K \subseteq \Gamma$ *for some* $K \in \mathtt{D}(\varphi)$ *implies* $\varphi \in \Gamma$*,*
- C_{npd}*:* $(\neg\varphi) \in \Gamma$ *implies* $H \subseteq \Gamma$ *for some* $H \in \mathtt{D}((\neg\varphi))$*.*

Proof. The argument follows by inspection of the definition of constituent. □

The subscripts p, np of the indices of the statements of Theorem 2.7.3 indicate that the formula involved is positive or negated positive, respectively. The subscript d refers to a "downward" implication from a formula to its constituents, while u refers to an "upward" implication. In addition, we will use the following two conditions:

- K_{nnd} for every formula φ, $(\neg(\neg\varphi)) \in \Gamma$ implies $\varphi \in \Gamma$.
- K_{nnu} for every formula φ, $\varphi \in \Gamma$ implies $(\neg(\neg\varphi)) \in \Gamma$.

Definition 2.7.4. A set of formulas Γ is

- *p-downward closed* if it satisfies the condition C_{pd} for every binary connective symbol C,
- *np-downward closed* if it satisfies the condition C_{npd} for every binary connective symbol C,
- *nn-downward closed* if it satisfies the condition K_{nnd},
- *p-upward closed* if it satisfies the condition C_{pu} for every binary connective symbol C,
- *np-upward closed* if it satisfies the condition C_{npu} for every binary connective symbol C,
- *nn-upward closed* if it satisfies the condition K_{nnu},
- *downward closed* if it is p-downward, np-downward, and nn-downward closed,
- *upward closed* if it is p-upward, np-upward, and nn-upward closed,
- *saturated* if for every formula φ, $\varphi \in \Gamma$ if and only if $(\neg\varphi) \notin \Gamma$ and Γ is both upward and downward closed.

□

Theorem 2.7.5. *Let* Γ *be a set of formulas such that for every formula* φ*, exactly one of* φ *and* $(\neg\varphi)$ *belongs to* Γ*. Then, we have*

(1) Γ *is upward closed if and only if* Γ *is both p-upward and np-upward closed,*
(2) Γ *is downward closed if and only if* Γ *is both p-downward and np-downward closed.*

Proof. It is clear that if Γ is upward closed, then it is both p-upward and np-upward closed. Conversely, let Γ be both p-upward and np-upward closed. We only need to show that the condition K_{nnu} is satisfied, that is, if $\{\varphi\} \subseteq \Gamma$, then $(\neg(\neg\varphi)) \in \Gamma$. Since $\varphi \in \Gamma$, we have $(\neg\varphi) \notin \Gamma$, so $(\neg(\neg\varphi)) \in \Gamma$. This concludes the proof of Part (1). Part (2) is entirely similar. □

Definition 2.7.6. A set of formulas Γ is a *truth set* if the following conditions are satisfied:

(1) for every formula φ, $(\neg\varphi) \in \Gamma$ if and only if $\varphi \notin \Gamma$,
(2) for every positive formula φ that is not a variable, we have $\varphi \in \Gamma$ if and only if at least one of its constituents is included in Γ.

▯

Theorem 2.7.7. *By replacing the second condition of the definition of truth set (Definition 2.7.6) by any of the following statements, one obtains an equivalent definition:*

(1) Γ *is both np-upward and np-downward closed.*
(2) Γ *is both p-upward and np-upward closed.*
(3) Γ *is both p-downward and np-downward closed.*
(4) Γ *is upward closed.*
(5) Γ *is downward closed.*
(6) Γ *is both upward and downward closed.*

Proof. The equivalence of the original definition with the first three modified definitions follows immediately from Theorem 2.7.3. The equivalence of the second and fourth modified definitions, on one hand, and of the third and fifth modified definitions, on the other hand, follows from Theorem 2.7.5. The equivalence of the last modified definition with the others is now immediate. □

Note that this shows that Γ is saturated if and only if it is a truth set.

We will now investigate the connection between truth sets and truth valuations.

Definition 2.7.8. For every function $w : \text{PLFORM} \longrightarrow \mathbf{Bool}$, let $\Gamma_w = \{\varphi \mid w(\varphi) = \mathbf{T}\}$.

For $\Gamma \subseteq \text{PLFORM}$, define the function $w_\Gamma : \text{PLFORM} \longrightarrow \mathbf{Bool}$ by

$$w_\Gamma(\varphi) = \begin{cases} \mathbf{T} & \text{if } \varphi \in \Gamma \\ \mathbf{F} & \text{otherwise.} \end{cases}$$

▯

It is straightforward to show that for any $w : \text{PLFORM} \longrightarrow \mathbf{Bool}$ and for any set of formulas Γ we have $\Gamma_{w_\Gamma} = \Gamma$ and $w_{\Gamma_w} = w$. Further, w is a truth valuation if and only if Γ_w is a truth set.

Theorem 2.7.9. *Γ is a truth set if and only if $\Gamma = \Gamma_w$ for some truth valuation w and w is a truth valuation if and only if $w = w_\Gamma$ for some truth set Γ.*

Proof. The proof is left to the reader. □

Corollary 2.7.10. *Every satisfiable set Γ is contained in a truth set.*

Proof. If Γ is satisfied by the truth assignment w, then $\Gamma \subseteq \Gamma_{\overline{w}}$ and $\Gamma_{\overline{w}}$ is a truth set by Theorem 2.7.9. □

Definition 2.7.11. A *maximally satisfiable set of formulas* is a set of formulas Γ that is satisfiable and for which there is no satisfiable set of formulas Γ' such that $\Gamma \subset \Gamma'$. ▯

Theorem 2.7.12. *Let Γ be a satisfiable set of formulas. Then, Γ is maximally satisfiable if and only if exactly one of the formulas φ and $(\neg\varphi)$ belongs to Γ for every formula $\varphi \in \text{PLFORM}$.*

Proof. Let Γ be a satisfiable set such that exactly one of the formulas φ and $(\neg\varphi)$ belongs to Γ for every formula $\varphi \in \text{PLFORM}$. Suppose that there exists a satisfiable set Γ_1 such that $\Gamma \subset \Gamma_1$ and let $\psi \in \Gamma_1 - \Gamma$. Since $\psi \notin \Gamma$, we have $(\neg\psi) \in \Gamma$. Therefore, both ψ

and $(\neg\psi)$ belong to Γ_1 and this is a contradiction because Γ_1 was supposed to be satisfiable. This shows that Γ is maximally satisfiable.

Conversely, suppose that Γ is a maximally satisfiable set. If there exists a formula φ such that neither φ nor $(\neg\varphi)$ belong to Γ, then one of the sets $\Gamma \cup \{\varphi\}$ or $\Gamma \cup \{(\neg\varphi)\}$ is satisfiable and strictly includes Γ. This contradicts the maximality of Γ. Since Γ may not contain both φ and $(\neg\varphi)$, it follows that exactly one of the formulas $\varphi, (\neg\varphi)$ belongs to Γ. □

Theorem 2.7.13. *A set of formulas Γ is maximally satisfiable if and only if Γ is a truth set.*

Proof. Let Γ be a maximally satisfiable set. Then, there is some truth valuation w such that $\Gamma \subseteq \Gamma_w$. Since Γ is maximally satisfiable, we have $\Gamma = \Gamma_w$, so Γ is a truth set by Theorem 2.7.9. Conversely, if Γ is a truth set, then $\Gamma = \Gamma_w$ for some truth valuation w, so Γ is satisfiable. Since for every formula φexactly one of φ and $(\neg\varphi)$ belongs to Γ, by Theorem 2.7.12, Γ is maximally satisfiable. □

We focus now on the generalization of truth set mentioned in the introduction.

Definition 2.7.14. A *Hintikka set* is a set Γ of formulas that satisfies the following conditions:

(1) for every statement variable p, at most one of the literals p and $(\neg p)$ is in Γ,
(2) Γ is downwards closed.

▯

Theorem 2.7.15. *Every truth set is a Hintikka set.*

Proof. Clearly, the first condition of the definition of truth set implies the first condition of the definition of Hintikka set. By Part (5) of Theorem 2.7.7, every truth set is downwards closed. □

Theorem 2.7.16. *Let Γ be a Hintikka set of formulas and let v be a truth assignment. Then, v satisfies Γ if and only if for each $p \in SV$, $v(p) = \mathbf{T}$ if $p \in \Gamma$ and $v(p) = \mathbf{F}$ if $(\neg p) \in \Gamma$.*

Proof. Suppose first that $v(p) = \mathbf{T}$ if $p \in \Gamma$ and $v(p) = \mathbf{F}$ if $(\neg p) \in \Gamma$ for each $p \in SV$. We must show that if $\varphi \in \Gamma$, then v satisfies φ. We prove this statement using course-of-values induction on the length of φ. Suppose that the result is true for formulas shorter than φ and that $\varphi \in \Gamma$. If φ is a literal, then $v(\varphi) = \mathbf{T}$ by our assumption about v. Otherwise, since Γ is a Hintikka set, there is a constituent K of φ such that $K \subseteq \Gamma$. Since each element of K is shorter than φ, by inductive hypothesis, v satisfies K, so by Theorem 2.7.2, v satisfies φ.

The converse implication is straightforward. □

Corollary 2.7.17. *Every Hintikka set of formulas is satisfiable.*

Proof. Since a Hintikka set may not contain both p and $(\neg p)$ for any statement variable p, a truth assignment as in Theorem 2.7.16 always exists. □

Definition 2.7.18. A *consistency property* is a collection $\mathcal{C}$ of sets of formulas such that

- no set with property $\mathcal{C}$ contains both p and $(\neg p)$ for any statement variable p,
- if Γ has property $\mathcal{C}$, $\varphi \in \Gamma$ and φ is not a literal, then $\Gamma \cup K$ has property $\mathcal{C}$ for some constituent K of φ.

A collection of sets of formulas $\mathcal{I}$ is an *inconsistency property* if the collection of sets $\mathcal{P}(\text{PLFORM}) - \mathcal{I}$ is a consistency property. ▯

Note that a collection of sets of formulas $\mathcal{I}$ is an inconsistency property if and only if the following conditions are satisfied:

- Every set of formulas including $\{p, (\neg p)\}$ for some statement variable p belongs to $\mathcal{I}$.
- If Γ is a set of formulas and $\varphi \in \Gamma$ is such that for every constituent K of φ, $\Gamma \cup K \in \mathcal{I}$, then $\Gamma \in \mathcal{I}$.

Example 2.7.19. Let $\mathcal{C}$ be the collection of all satisfiable sets of formulas. Then, by Theorem 2.7.2, $\mathcal{C}$ is a consistency property. ▯

Example 2.7.20. Let φ be a formula and let $\mathcal{I}_\varphi$ be the collection of all sets of formulas Γ such that $\Gamma \models \varphi$. We leave to the reader the easy verification that $\mathcal{I}_\varphi$ is an inconsistency property. This shows that $\mathcal{C}_\varphi$, the collection of all sets of formulas Γ such that $\Gamma \not\models \varphi$, is a consistency property. ▯

Theorem 2.7.21. *Every member of a consistency property is a satisfiable set of formulas. In fact, if* Γ *is a member of a consistency property, there is a Hintikka set* Γ' *such that* $\Gamma \subseteq \Gamma'$ *and every formula in* Γ' *is either a subformula or the negation of a proper subformula of a formula of* Γ.

Proof. Let $\mathcal{C}$ be a consistency property and let Γ have property $\mathcal{C}$. We shall construct a Hintikka set Γ' such that $\Gamma \subseteq \Gamma'$. By Corollary 2.7.17, we can then conclude that Γ is satisfiable. We build recursively a sequence of sets $\Gamma_0, \Gamma_1, \ldots$, each with property $\mathcal{C}$ such that every formula in Γ_n is either a subformula or a negation of a proper subformula of a formula in Γ. Then, we set $\Gamma' = \bigcup\{\Gamma_n \mid n \in \mathbf{N}\}$.

We define $\Gamma_0 = \Gamma$. Suppose that Γ_n is defined and is in $\mathcal{C}$. Then, there are two cases:

- If Γ_n is a Hintikka set, $\Gamma_{n+1} = \Gamma_n$.
- If Γ_n is not a Hintikka set, then, since no set with property $\mathcal{C}$ may contain both a variable and its negation, there must be a formula $\varphi \in \Gamma_n$ such that none of its constituents is included in Γ_n. Let φ be the first such formula in the standard ordering and let K be the first constituent of φ in the sequence $\mathtt{d}(\varphi)$ such that $\Gamma_n \cup K$ is in $\mathcal{C}$. Define $\Gamma_{n+1} = \Gamma_n \cup K$.

First, we show that every $\Gamma_n \subseteq \Gamma \cup U(\Gamma)$, that is, every formula in Γ_n, is either a subformula or a negation of a proper subformula of a formula in Γ. The argument is by induction on n. The basis, $n = 0$, is obvious. Suppose that $\Gamma_n \subseteq \Gamma \cup U(\Gamma)$. If $\Gamma_{n+1} = \Gamma_n$, there is nothing to prove. Otherwise, $\Gamma_{n+1} = \Gamma_n \cup K$, where K is a constituent of a formula $\varphi \in \Gamma_n \subseteq \Gamma \cup U(\Gamma)$. Since $K \subseteq U(\{\varphi\})$, we have, by Theorem 2.2.14, $K \subseteq U(\Gamma \cup U(\Gamma)) = U(\Gamma) \cup U(U(\Gamma)) = U(\Gamma)$. Thus, $\Gamma_{n+1} \subseteq \Gamma \cup U(\Gamma)$, so $\Gamma' \subseteq \Gamma \cup U(\Gamma)$.

Note that Γ' cannot contain both p and $(\neg p)$ for any $p \in SV$. Suppose that Γ' is not a Hintikka set. Then, there exists a formula in Γ' that is not a literal and none of whose constituents is contained in Γ'. Let φ be the first such formula in the standard order. Choose n large enough to satisfy the following conditions:

- $\varphi \in \Gamma_n$.
- For all predecessors ψ of φ in the standard order that are not literals and are in Γ', some constituent of ψ is included in Γ_n.

Such an n exists because for all predecessors of φ in the standard order that are not literals and belong to Γ', one of their constituents is included in Γ' and each such constituent is finite. By construction, Γ_{n+1} will include a constituent of φ, contradicting our choice of φ, in view of the fact that $\Gamma_{n+1} \subseteq \Gamma'$. $\Box$

The fact that the set Γ' constructed in the previous theorem consists of subformulas and negated proper subformulas of formulas of Γ allows us to say that Γ' was obtained analytically from Γ, that is, the members of Γ' were obtained by analyzing the structure of the formulas of Γ.

Corollary 2.7.22. *For every satisfiable set* Γ*, there is a Hintikka set* Γ' *such that* $\Gamma \subseteq \Gamma'$ *and every formula in* Γ' *is either a subformula or the negation of a proper subformula of a formula of* Γ*.*

Proof. Since satisfiability is a consistency property, this statement follows immediately from Theorem 2.7.21. $\Box$

Example 2.7.23. We can use Theorem 2.7.21 to give another proof of the Compactness Theorem (Theorem 2.4.3). In order to do this, we will prove (without using the Compactness Theorem) that finite satisfiability is a consistency property. Then, Theorem 2.7.21 will establish the nontrivial part of the Compactness Theorem. In fact, we will prove that the property of not being finitely satisfiable is an inconsistency property.

It is clear that any set that contains both p and $(\neg p)$, where p is a statement variable, is not finitely satisfiable.

Now, we suppose that Γ is a set of formulas, $\varphi \in \Gamma$ is not a literal and for no constituent K of φ is $\Gamma \cup K$ finitely satisfiable, and we show that Γ is not finitely satisfiable. By our hypotheses, for each constituent K of φ, there is an unsatisfiable, finite subset Ξ_K of $\Gamma \cup K$. Let $\Gamma_K = \Gamma \cap \Xi_K$. Then, Γ_K is a finite subset of Γ and $\Gamma_K \cup K$ is unsatisfiable. Let $\Gamma_0 = \bigcup\{\Gamma_K \mid K \text{ is a constituent of } \varphi\} \cup \{\varphi\}$. Clearly, Γ_0 is a finite subset of Γ. We claim that Γ_0 is unsatisfiable. Suppose that v were a truth assignment that satisfied Γ_0. Then v would satisfy φ and, therefore, it would satisfy one of its constituents K_0. Since v also satisfies Γ_{K_0}, it follows that v satisfies $\Gamma_{K_0} \cup K_0$ which is known to be unsatisfiable. $\Box$

We introduce now the notion of constituent for signed formulas. This will allow us to introduce Hintikka sets of signed formulas.

Definition 2.7.24. The mapping

$$\mathtt{d} : (\text{SPLFORM} - (\mathbf{Bool} \times SV)) \longrightarrow \text{Seq}(\mathcal{P}(\text{SPLFORM}))$$

is given by the following table:

Signed Formula $b\alpha$	$\mathtt{d}(b\alpha)$
$\mathbf{T}(\neg\varphi)$	$(\{\mathbf{F}\varphi\})$
$\mathbf{F}(\neg\varphi)$	$(\{\mathbf{T}\varphi\})$
$\mathbf{T}(\varphi \wedge \psi)$	$(\{\mathbf{T}\varphi, \mathbf{T}\psi\})$
$\mathbf{F}(\varphi \wedge \psi)$	$(\{\mathbf{F}\varphi\}, \{\mathbf{F}\psi\})$
$\mathbf{T}(\varphi \vee \psi)$	$(\{\mathbf{T}\varphi\}, \{\mathbf{T}\psi\})$
$\mathbf{F}(\varphi \vee \psi)$	$(\{\mathbf{F}\varphi, \mathbf{F}\psi\})$
$\mathbf{T}(\varphi \to \psi)$	$(\{\mathbf{F}\varphi\}, \{\mathbf{T}\psi\})$
$\mathbf{F}(\varphi \to \psi)$	$(\{\mathbf{T}\varphi, \mathbf{F}\psi\})$
$\mathbf{T}(\varphi \leftrightarrow \psi)$	$(\{\mathbf{T}\varphi, \mathbf{T}\psi\}, \{\mathbf{F}\varphi, \mathbf{F}\psi\})$
$\mathbf{F}(\varphi \leftrightarrow \psi)$	$(\{\mathbf{T}\varphi, \mathbf{F}\psi\}, \{\mathbf{F}\varphi, \mathbf{T}\psi\})$

Let $b\varphi$ be a signed formula such that φ is not a variable. The *constituent sequence* of $b\varphi$ is the sequence $\mathtt{d}(b\varphi)$. The *constituent set* of $b\varphi$ is the set $\mathtt{D}(b\varphi)$ that consists of all sets of formulas that occur in $\mathtt{d}(b\varphi)$. Every such set of formulas is called a *constituent* of $b\varphi$. ▯

The signed formulas of a constituent of a signed formula $b\varphi$ are signed subformulas of φ. Thus, constituents of signed formulas are obtained analytically, just as constituents of unsigned formulas.

Theorem 2.7.25. *If φ is not a statement variable, then a truth assignment v satisfies the signed formula $b\varphi$ if and only if it satisfies at least one constituent of $b\varphi$.*

Proof. We leave this easy verification to the reader. □

Corollary 2.7.26. *A set of signed formulas $\Delta \cup \{b\varphi\}$ is unsatisfiable if and only if the set of signed formulas $\Delta \cup K$ is unsatisfiable for every constituent K of $b\varphi$.*

Proof. This follows immediately from Theorem 2.7.25. □

It is easy to verify that if $b'\theta$ belongs to a constituent of the signed formula $b\varphi$, then θ is an immediate subformula of φ.

Definition 2.7.27. A *Hintikka set of signed formulas* is a set Δ of signed formulas that satisfies the following conditions:

(1) for every statement variable p, at most one of the signed formulas $\mathbf{T}p$ and $\mathbf{F}p$ is in Δ,
(2) if $b\varphi \in \Delta - (\mathbf{Bool} \times SV)$, then there is a constituent K of $b\varphi$ such that $K \subseteq \Delta$.

▯

Theorem 2.7.28. *Let Δ be a Hintikka set of signed formulas and let v be a truth assignment. Then, v satisfies Δ if and only if $v(p) = b$ for each signed variable $bp \in \Delta$.*

Proof. Suppose first that $v(p) = b$ for each signed variable $bp \in SV$. We will show by induction on formulas φ that if $b\varphi \in \Delta$, then v satisfies $b\varphi$. If φ is a statement variable, then the statement holds by our assumption about v. Now, suppose that the statement is true for φ and $b(\neg\varphi) \in \Delta$. Since Δ is a Hintikka set, $f_{\neg}(b)\varphi \in \Delta$. By inductive hypothesis, $v(\varphi) = f_{\neg}(b)$, so $v(\neg\varphi) = b$, which means that v satisfies $b(\neg\varphi)$.

Now suppose that the result holds for φ and ψ and $b(\varphi C \psi) \in \Delta$ for some binary connective symbol C. Since Δ is a Hintikka set, there is a constituent K of $b(\varphi C \psi)$ which is contained in Δ. Since K consists of signed formulas involving φ and ψ, by the inductive hypothesis, v satisfies K and hence, v satisfies $b(\varphi C \psi)$ by Theorem 2.7.25.

The converse implication is immediate. □

Corollary 2.7.29. *Every Hintikka set of signed formulas is satisfiable.*

Proof. Since a Hintikka set may not contain both $\mathbf{T}p$ and $\mathbf{F}p$ for any statement variable p, a truth assignment as in Theorem 2.7.28 always exists. □

2.8 Truth Functions

In Section 2.3, we defined four binary operations $f_{\wedge}, f_{\vee}, f_{\rightarrow}, f_{\leftrightarrow}$ and one unary operation $f_{\neg}$ on the set $\mathbf{Bool}$. This section is dedicated to a study of the algebraic properties of these operations and, in general, of operations that can be defined on **Bool**. Properties of truth functions discussed in this section are important not only for propositional logic but also for a number of applications that interest engineers and computer scientists.

The set of truth functions $\mathbf{Bool}^n \longrightarrow \mathbf{Bool}$ will be denoted in the sequel by $\mathtt{TF}_n$ for $n \in \mathbf{N}$. The set of all truth functions $\bigcup_{n\in\mathbf{N}} \mathtt{TF}_n$ will be denoted by $\mathtt{TF}_*$; we shall use frequently the notation $\mathtt{TF} = \bigcup_{n\geq 1} \mathtt{TF}_n$.

Note that there are exactly two 0-ary truth functions, $f_\top() = \mathbf{T}$ and $f_\perp() = \mathbf{F}$ and hence $\mathtt{TF}_* = \mathtt{TF} \cup \{f_\top, f_\perp\}$.

For every $m \in \mathbf{N}$, let $c^m_{\mathbf{F}}, c^m_{\mathbf{T}}$ be the constant m-ary truth functions given by

$$\begin{aligned} c^m_{\mathbf{F}}(x_0, \dots, x_{m-1}) &= \mathbf{F}, \\ c^m_{\mathbf{T}}(x_0, \dots, x_{m-1}) &= \mathbf{T}, \end{aligned} \tag{2.3}$$

for every $(x_0, \dots, x_{m-1}) \in \mathbf{Bool}^m$. Note that $c^0_{\mathbf{F}} = f_\perp$ and $c^0_{\mathbf{T}} = f_\top$.

Note that despite the simplicity of the set **Bool**, the number of truth functions of n arguments that can be defined on **Bool** grows very fast. Indeed, there are 2^{2^n} functions in the set $\mathtt{TF}_n$; even for $n = 10$ we have 2^{1024} such functions, a number of the order of magnitude of 10^{308}! This shows that even for a relatively small number of arguments listing all the truth functions that have certain properties might be very costly.

Definition 2.8.1. Let $n, i \in \mathbf{N}$ be such that $0 \leq i \leq n-1$. The *projection* π^n_i is the truth function $\pi^n_i : \mathbf{Bool}^n \longrightarrow \mathbf{Bool}$ which is given by $\pi^n_i(a_0, \dots, a_{n-1}) = a_i$ for all $a_0, \dots, a_{n-1} \in \mathbf{Bool}$.

The set of all projections will be denoted by PROJ. □

Example 2.8.2. The projection π^1_0 is the identity function on **Bool**. The projections π^2_0 and π^2_1 are given by $\pi^2_0(a_0, a_1) = a_0$ and $\pi^2_1(a_0, a_1) = a_1$ for every $a_0, a_1 \in \mathbf{Bool}$. □

We remind the reader of the definition of composition of functions as it pertains to truth functions.

Definition 2.8.3. Let $f : \mathbf{Bool}^m \longrightarrow \mathbf{Bool}$, where $m > 0$, and let $g_i : \mathbf{Bool}^n \longrightarrow \mathbf{Bool}$ for $0 \leq i \leq m-1$ be truth functions. The composition of f with $g_0, \ldots, g_{m-1}$ is the function $f(g_0, \ldots, g_{m-1}) : \mathbf{Bool}^n \longrightarrow \mathbf{Bool}$ given by

$$\begin{aligned} &f(g_0, \ldots, g_{m-1})(a_0, \ldots, a_{n-1}) \\ &\quad = f(g_0(a_0, \ldots, a_{n-1}), \ldots, g_{m-1}(a_0, \ldots, a_{n-1})) \end{aligned}$$

for every $a_0, \ldots, a_{m-1} \in \mathbf{Bool}$. ∎

Let a, b be two arbitrary elements of **Bool**. We use the infix notation $a \wedge b, a \vee b, a \rightarrow b$ and $a \leftrightarrow b$ instead of $f_\wedge(a,b), f_\vee(a,b), f_\rightarrow(a,b)$ and $f_\leftrightarrow(a,b)$, respectively. Instead of $a \leftrightarrow b$, it is convenient to use sometimes the notation a^b. Also, we shall write $\bar{a}$ instead of $f_\neg(a)$.

It is easy to check that for every $a, b \in \mathbf{Bool}$:

$$a^b = \begin{cases} a & \text{if } b = \mathbf{T}, \\ \bar{a} & \text{if } b = \mathbf{F}. \end{cases}$$

Theorem 2.8.4. *The operations* $\vee$, $\wedge$, *and* $\neg$ *on* **Bool** *have the following properties:*

$$\begin{array}{rl}
a \wedge a = a & \text{(idempotency of } \wedge\text{)}, \\
a \vee a = a & \text{(idempotency of } \vee\text{)}, \\
a \wedge b = b \wedge a & \text{(commutativity of } \wedge\text{)}, \\
a \vee b = b \vee a & \text{(commutativity of } \vee\text{)}, \\
(a \wedge b) \wedge c = a \wedge (b \wedge c) & \text{(associativity of } \wedge\text{)}, \\
(a \vee b) \vee c = a \vee (b \vee c) & \text{(associativity of } \vee\text{)}, \\
\neg\neg a = a & \text{(involutive property of } \neg\text{)} \\
a \wedge (a \vee b) = a & \text{(absorption property of } \wedge\text{)}, \\
a \vee (a \wedge b) = a & \text{(absorption property of } \vee\text{)}, \\
a \wedge (b \vee c) = (a \wedge b) \vee (a \wedge c) & \text{(distributivity)}, \\
(b \vee c) \wedge a = (b \wedge a) \vee (c \wedge a) & \quad ", \\
a \vee (b \wedge c) = (a \vee b) \wedge (a \vee c) & \quad ", \\
(b \wedge c) \vee a = (b \vee a) \wedge (c \vee a) & \quad ", \\
\neg(a \wedge b) = (\neg a) \vee (\neg b) & \text{(De Morgan properties)}, \\
\neg(a \vee b) = (\neg a) \wedge (\neg b) & \quad ",
\end{array}$$

for every $a, b, c \in \mathbf{Bool}$.

Proof. The argument parallels the one for Theorem 2.3.30. □

The binary operation "+" is defined by

$$a + b = \begin{cases} \mathbf{T} & \text{if } a \neq b, \\ \mathbf{F} & \text{if } a = b, \end{cases}$$

for every $a, b \in \mathbf{Bool}$.

Theorem 2.8.5. *The operation "+" is commutative and associative. In other words, we have*

$$\begin{aligned} a + b &= b + a && \text{(commutativity of } +\text{)}, \\ (a + b) + c &= a + (b + c) && \text{(associativity of } +\text{)}, \end{aligned}$$

for every $a, b, c \in \mathbf{Bool}$. Also, for every $a \in \mathbf{Bool}$, we have

$$a + a = \mathbf{F},$$

$$a + \mathbf{F} = a,$$

$$a + \mathbf{T} = \bar{a}.$$

The operation "$\wedge$" is distributive with respect to "+", that is,

$$a \wedge (b + c) = (a \wedge b) + (a \wedge c),$$

$$(b + c) \wedge a = (b \wedge a) + (c \wedge a),$$

for every $a, b, c \in \mathbf{Bool}$.

Proof. We shall prove only the first distributivity equality and leave the proof of the other parts to the reader. Consider the cases summarized by the following table:

a	b	c	$b + c$	$a \wedge (b + c)$	$a \wedge b$	$a \wedge c$	$(a \wedge b) + (a \wedge c)$
F	**F**	**F**	**F**	**F**	**F**	**F**	**F**
F	**F**	**T**	**T**	**F**	**F**	**F**	**F**
F	**T**	**F**	**T**	**F**	**F**	**F**	**F**
F	**T**	**T**	**F**	**F**	**F**	**F**	**F**
T	**F**	**F**	**F**	**F**	**F**	**F**	**F**
T	**F**	**T**	**T**	**T**	**F**	**T**	**T**
T	**T**	**F**	**T**	**T**	**T**	**F**	**T**
T	**T**	**T**	**F**	**F**	**T**	**T**	**F**

By comparing the fifth and the last column of the table, we conclude that the first distributivity property is satisfied. □

The operations introduced thus far on **Bool** allow us to define similar operations on the set of truth functions.

Definition 2.8.6. Let $f, g : \mathbf{Bool}^n \longrightarrow \mathbf{Bool}$ be two n-ary truth functions for $n \in \mathbf{N}$. The functions $f \vee g$, $f \wedge g$, $f \rightarrow g$, $f \leftrightarrow g$, $f + g$, and $\neg f$ are given by

$$\begin{aligned}
(f \vee g)(x_0, \dots, x_{n-1}) &= f(x_0, \dots, x_{n-1}) \vee g(x_0, \dots, x_{n-1}),\\
(f \wedge g)(x_0, \dots, x_{n-1}) &= f(x_0, \dots, x_{n-1}) \wedge g(x_0, \dots, x_{n-1}),\\
(f \rightarrow g)(x_0, \dots, x_{n-1}) &= f(x_0, \dots, x_{n-1}) \rightarrow g(x_0, \dots, x_{n-1}),\\
(f \leftrightarrow g)(x_0, \dots, x_{n-1}) &= f(x_0, \dots, x_{n-1}) \leftrightarrow g(x_0, \dots, x_{n-1}),\\
(f + g)(x_0, \dots, x_{n-1}) &= f(x_0, \dots, x_{n-1}) + g(x_0, \dots, x_{n-1}),\\
(\neg f)(x_0, \dots, x_{n-1}) &= \neg f(x_0, \dots, x_{n-1})
\end{aligned}$$

for every $(x_0, \dots, x_{n-1}) \in \mathbf{Bool}^n$. □

An alternative notation for $(\neg f)$ is $\bar{f}$.

Consider the function $\mathbf{OR} : \mathrm{Seq}(\mathbf{Bool}) \longrightarrow \mathbf{Bool}$ defined recursively by

$$\begin{aligned}
\mathbf{OR}(\lambda) &= \mathbf{F},\\
\mathbf{OR}(a_0, \dots, a_n) &= \mathbf{OR}(a_0, \dots, a_{n-1}) \vee a_n,
\end{aligned}$$

for every $a_0, \dots, a_n \in \mathbf{Bool}$ and $n \in \mathbf{N}$. The usual notations for $\mathbf{OR}(\vec{a})$ (where $\vec{a} = (a_0, \dots, a_{n-1})$) are $a_0 \vee \cdots \vee a_{n-1}$ and $\bigvee_{0 \leq i \leq n-1} a_i$.

Similarly, define the function $\mathbf{AND} : \mathrm{Seq}(\mathbf{Bool}) \longrightarrow \mathbf{Bool}$ by

$$\begin{aligned}
\mathbf{AND}(\lambda) &= \mathbf{T},\\
\mathbf{AND}(a_0, \dots, a_n) &= \mathbf{AND}(a_0, \dots, a_{n-1}) \wedge a_n,
\end{aligned}$$

for every $a_0, \dots, a_n \in \mathbf{Bool}$ and $n \in \mathbf{N}$. The usual notations for $\mathbf{AND}(\vec{a})$ (where $\vec{a} = (a_0, \dots, a_{n-1})$) are $a_0 \wedge \cdots \wedge a_{n-1}$ and $\bigwedge_{0 \leq i \leq n-1} a_i$.

The operation "+" can be extended to sequences by defining the function $\mathbf{SUM} : \mathrm{Seq}(\mathbf{Bool}) \longrightarrow \mathbf{Bool}$:

$$\begin{aligned}
\mathbf{SUM}(\lambda) &= \mathbf{F},\\
\mathbf{SUM}(a_0, \dots, a_n) &= \mathbf{SUM}(a_0, \dots, a_{n-1}) + a_n,
\end{aligned}$$

for every $a_0, \ldots, a_n \in \mathbf{Bool}$ and $n \in \mathbf{N}$. The usual notations for $\mathbf{SUM}(\vec{a})$ (where $\vec{a} = (a_0, \ldots, a_{n-1})$) are $a_0 + \cdots + a_{n-1}$ and $\sum_{0 \leq i \leq n-1} a_i$.

Lemma 2.8.7. *Let* $(a_0, \ldots, a_{n-1})$ *be a sequence of elements of* $\mathbf{Bool}$. *The following statements hold:*

(1) $\bigvee_{0 \leq i \leq n-1} a_i = \mathbf{T}$ *if and only if there is an* i, $0 \leq i \leq n-1$, *such that* $a_i = \mathbf{T}$,
(2) $\bigwedge_{0 \leq i \leq n-1} a_i = \mathbf{T}$ *if and only if for every* i, $0 \leq i \leq n-1$, $a_i = \mathbf{T}$,
(3) $\sum_{0 \leq i \leq n-1} a_i = \mathbf{T}$ *if and only if* $|\{i \mid 0 \leq i \leq n-1$ *and* $a_i = \mathbf{T}\}|$ *is odd.*

Proof. The proof of all three statements is by induction on n. We give here only the proof of the third statement. The basis step, $n = 0$, is immediate because we defined the sum of the empty sequence to be $\mathbf{F}$. Suppose that the third statement is true for sequences of length n. Then, the following statements are equivalent:

(i) $\sum_{0 \leq i \leq n} a_i = \mathbf{T}$,
(ii) either $\sum_{0 \leq i \leq n-1} a_i = \mathbf{T}$ and $a_n = \mathbf{F}$ or $\sum_{0 \leq i \leq n-1} a_i = \mathbf{F}$ and $a_n = \mathbf{T}$,
(iii) either $|\{i \mid 0 \leq i \leq n-1$ and $a_i = \mathbf{T}\}|$ is odd and $a_n = \mathbf{F}$ or $|\{i \mid 0 \leq i \leq n-1$ and $a_i = \mathbf{T}\}|$ is even and $a_n = \mathbf{T}$,
(iv) $|\{i \mid 0 \leq i \leq n$ and $a_i = \mathbf{T}\}|$ is odd.

□

Theorem 2.8.8. *Let* $(a_0, \ldots, a_{n-1})$ *be a sequence in Seq*$(\mathbf{Bool})$. *For every permutation* $(a_{i_0}, \ldots, a_{i_{n-1}})$ *of the sequence, we have*

$$\bigvee_{0 \leq j \leq n-1} a_j = \bigvee_{0 \leq j \leq n-1} a_{i_j},$$

$$\bigwedge_{0 \leq j \leq n-1} a_j = \bigwedge_{0 \leq j \leq n-1} a_{i_j},$$

$$\sum_{0 \leq j \leq n-1} a_j = \sum_{0 \leq j \leq n-1} a_{i_j}.$$

Proof. The proof follows from Lemma 2.8.7. □

Theorem 2.8.9. *Let* $(a_0, \ldots, a_{n-1}), (b_0, \ldots, b_{m-1})$ *be in Seq*(**Bool**). *We have*

$$\bigvee\nolimits_{0\le i\le n-1} a_i \vee \bigvee\nolimits_{0\le j\le m-1} b_j = a_0 \vee \cdots \vee a_{n-1} \vee b_0 \vee \cdots \vee b_{m-1},$$

$$\bigwedge\nolimits_{0\le i\le n-1} a_i \wedge \bigwedge\nolimits_{0\le j\le m-1} b_j = a_0 \wedge \cdots \wedge a_{n-1} \wedge b_0 \wedge \cdots \wedge b_{m-1},$$

$$\sum_{0\le i\le n-1} a_i + \sum_{0\le j\le m-1} b_j = a_0 + \cdots + a_{n-1} + b_0 + \cdots + b_{m-1}.$$

Proof. We prove only the third assertion of the theorem, by giving a semantic argument. Let p_a be the number of i, $0 \le i \le n-1$, such that $a_i = \mathbf{T}$ and let p_b be the analogous number for the sequence $(b_0, \ldots, b_{n-1})$. The following statements are easily seen to be equivalent:

- $\sum_{0\le i\le n-1} a_i + \sum_{0\le j\le m-1} b_j = \mathbf{F}$,
- the numbers p_a and p_b have the same parity,
- $p_a + p_b$ is an even number,
- $a_0 + \cdots + a_{n-1} + b_0 + \cdots + b_{m-1} = \mathbf{F}$.

$\square$

Theorem 2.8.10. *Let* $(a_0, \ldots, a_{n-1}), (b_0, \ldots, b_{n-1})$ *be in Seq*(**Bool**). *We have*

$$\bigvee\nolimits_{0\le i\le n-1} a_i \vee \bigvee\nolimits_{0\le j\le n-1} b_j = \bigvee\nolimits_{0\le i\le n-1} (a_i \vee b_i),$$

$$\bigwedge\nolimits_{0\le i\le n-1} a_i \wedge \bigwedge\nolimits_{0\le j\le n-1} b_j = \bigwedge\nolimits_{0\le i\le n-1} (a_i \wedge b_i),$$

$$\sum_{0\le i\le n-1} a_i + \sum_{0\le j\le n-1} b_j = \sum_{0\le i\le n-1} (a_i + b_i).$$

Proof. Again, we prove only the third part of the theorem. The basis step, $n = 0$, of the induction argument on n is trivial. Suppose the statement holds for n. We have

$$\begin{aligned}\sum_{0\le i\le n} a_i + \sum_{0\le j\le n} b_j &= \left(\sum_{0\le i\le n-1} a_i + a_n\right) + \left(\sum_{0\le j\le n-1} b_j + b_n\right)\\ &= \sum_{0\le i\le n-1} a_i + a_n + \sum_{0\le j\le n-1} b_j + b_n\end{aligned}$$

$$\begin{aligned}
&\quad \text{(by Theorem 2.8.9, Part (3))}\\
&= \sum_{0\le i\le n-1} a_i + \sum_{0\le j\le n-1} b_j + a_n + b_n\\
&\quad \text{(by Theorem 2.8.8)}\\
&= \left(\sum_{0\le i\le n-1} a_i + \sum_{0\le j\le n-1} b_j\right) + (a_n + b_n)\\
&= \sum_{0\le i\le n-1} (a_i + b_i) + (a_n + b_n)\\
&\quad \text{(by inductive hypothesis)}\\
&= \sum_{0\le i\le n} (a_i + b_i).
\end{aligned}$$

□

Theorem 2.8.11. *Let* $(b_0, \ldots, b_{n-1})$ *be a sequence in* $\mathrm{Seq}(\mathbf{Bool})$ *and let* $a \in \mathbf{Bool}$. *We have the following generalized distributivity laws:*

$$\begin{gathered}
a \wedge \left(\bigvee\nolimits_{0\le i\le n-1} b_i\right) = \bigvee\nolimits_{0\le i\le n-1} (a \wedge b_i),\\
\left(\bigvee\nolimits_{0\le i\le n-1} b_i\right) \wedge a = \bigvee\nolimits_{0\le i\le n-1} (b_i \wedge a),\\
a \vee \left(\bigwedge\nolimits_{0\le i\le n-1} b_i\right) = \bigwedge\nolimits_{0\le i\le n-1} (a \vee b_i),\\
\left(\bigwedge\nolimits_{0\le i\le n-1} b_i\right) \vee a = \bigwedge\nolimits_{0\le i\le n-1} (b_i \vee a),\\
a \wedge \left(\sum_{0\le i\le n-1} b_i\right) = \sum_{0\le i\le n-1} (a \wedge b_i),\\
\left(\sum_{0\le i\le n-1} b_i\right) \wedge a = \sum_{0\le i\le n-1} (b_i \wedge a).
\end{gathered}$$

Proof. The argument for each of the equalities is by induction on n and is left to the reader. □

Let $f_0, \ldots, f_{m-1}$ be a sequence of n-ary truth functions, where $m, n \in \mathbf{N}$. We define the n-ary truth functions $\bigvee_{0 \leq i \leq m-1} f_i$ and $\bigwedge_{0 \leq i \leq m-1} f_i$, and $\sum_{0 \leq i \leq m-1} f_i$ by

$$\left(\bigvee\nolimits_{0 \leq i \leq m-1} f_i\right)(\vec{b}) = \bigvee\nolimits_{0 \leq i \leq m-1} f_i(\vec{b}),$$

$$\left(\bigwedge\nolimits_{0 \leq i \leq m-1} f_i\right)(\vec{b}) = \bigwedge\nolimits_{0 \leq i \leq m-1} f_i(\vec{b}),$$

$$\left(\sum_{0 \leq i \leq m-1} f_i)(\vec{b}\right) = \sum_{0 \leq i \leq m-1} f_i(\vec{b}),$$

for every $\vec{b} \in \mathbf{Bool}^n$.

In order to simplify the notation for truth functions, we shall assume that the following order of priority has been imposed on the operations mentioned so far:

(1) $\neg$ has the highest priority,
(2) $\vee, \wedge, \rightarrow$ and $\leftrightarrow$ have the second highest priority, and
(3) $+$ has the lowest priority.

Parentheses will be used whenever we need to eliminate ambiguities.

Let $\vec{a} = (a_0, \ldots, a_{n-1})$ and $\vec{b} = (b_0, \ldots, b_{n-1})$ be two sequences over **Bool** having the same length. We use the notation $\vec{a}^{\,\vec{b}}$ for

$$\vec{a}^{\,\vec{b}} = a_0^{b_0} \wedge \cdots \wedge a_{n-1}^{b_{n-1}},$$

where we use the notation a^b introduced just before Theorem 2.8.4. Note that $\vec{a}^{\,\vec{b}} = \mathbf{T}$ if and only if $\vec{a} = \vec{b}$, that is, if and only if $a_i = b_i$ for all i, $0 \leq i \leq n-1$.

Definition 2.8.12. Let $f \in \mathtt{TF}_n$ be a truth function for $n \in \mathbf{N}$. The *dual* of f is the function $f^d \in \mathtt{TF}_n$ given by

$$f^d(a_0, \ldots, a_{n-1}) = \overline{f(\overline{a_0}, \ldots, \overline{a_{n-1}})},$$

for every $(a_0, \ldots, a_{n-1}) \in \mathbf{Bool}^n$. ▯

Example 2.8.13. The dual of the function $f_\wedge$ is the function $f_\vee$ since, by the De Morgan laws, we have

$$\overline{\overline{a} \wedge \overline{b}} = \overline{\overline{a}} \vee \overline{\overline{b}} = a \vee b,$$

for every $a, b \in \mathbf{Bool}$. In a similar manner, it is easy to see that the dual of $f_\vee$ is $f_\wedge$. ▯

It is easy to prove that the dual of the dual of a function f : $\mathbf{Bool}^n \longrightarrow \mathbf{Bool}$ is f itself. Indeed, we have

$$\begin{aligned}(f^d)^d(a_0, \dots, a_{n-1}) &= \overline{f^d(\overline{a_0}, \dots, \overline{a_{n-1}})}\\ &= \overline{\overline{f(\overline{\overline{a_0}}, \dots, \overline{\overline{a_{n-1}}})}}\\ &= f(a_0, \dots, a_{n-1}),\end{aligned}$$

for every $a_0, \dots, a_{n-1} \in \mathbf{Bool}$.

Definition 2.8.14. A truth function f is *self-dual* if $f = f^d$. The sets $\mathcal{SD}_*$, $\mathcal{SD}$, and $\mathcal{SD}_n$ (for $n \in \mathbf{N}$) are given by

$$\begin{aligned}\mathcal{SD}_* &= \{f \in \mathtt{TF}_* \mid f \text{ is self-dual}\},\\ \mathcal{SD} &= \mathcal{SD}_* \cap \mathtt{TF},\\ \mathcal{SD}_n &= \mathcal{SD}_* \cap \mathtt{TF}_n.\end{aligned}$$

□

Note that a function $f \in \mathtt{TF}_n$ is self-dual if and only if

$$f(\overline{a_0}, \dots, \overline{a_{n-1}}) = \overline{f(a_0, \dots, a_{n-1})},$$

for every $a_0, \dots, a_{n-1} \in \mathbf{Bool}$.

Since neither of the functions $f_\top$ and $f_\bot$ is self-dual, we have $\mathcal{SD}_* = \mathcal{SD}$.

Example 2.8.15. The function $f_\neg$ is self-dual. Indeed, we have

$$f_\neg^d(x) = \overline{f_\neg(\overline{x})} = \overline{\overline{\overline{x}}} = \overline{x} = f_\neg(x),$$

for every $x \in \mathbf{Bool}$.

It is easy to see that the identity π_0^1 is also self-dual. In general, every projection π_i^n is self-dual.

Note that the set $\mathtt{TF}_1$ consists of four functions, $\mathtt{TF}_1 = \{f_\neg, \pi_0^1, f_0, f_1\}$, where f_0 and f_1 are the two constant functions, given by $f_0(x) = \mathbf{F}$ and $f_1(x) = \mathbf{T}$ for $x \in \mathbf{Bool}$, respectively. Therefore, among the truth functions in $\mathtt{TF}_1$, only f_0 and f_1 are not self-dual.

□

Definition 2.8.16. For $n \in \mathbf{N}$, a function $f \in \mathtt{TF}_n$ is *linear* if there exist $n+1$ elements $k, k_0, \ldots, k_{n-1}$ of **Bool** such that

$$f(a_0, \ldots, a_{n-1}) = k + (k_0 \wedge a_0) + \ldots + (k_{n-1} \wedge a_{n-1})$$

for every $a_0, \ldots, a_{n-1} \in \mathbf{Bool}$. The sets $\mathcal{LIN}_*$, $\mathcal{LIN}$, and $\mathcal{LIN}_n$ (for $n \in \mathbf{N}$) are given by

$$\begin{aligned}
\mathcal{LIN}_* &= \{f \in \mathtt{TF}_* \mid f \text{ is linear}\},\\
\mathcal{LIN} &= \mathcal{LIN}_* \cap \mathtt{TF},\\
\mathcal{LIN}_n &= \mathcal{LIN}_* \cap \mathtt{TF}_n.
\end{aligned}$$

$\Box$

Note that both $f_\top$ and $f_\perp$ are linear, so $\mathcal{LIN}_* = \mathcal{LIN} \cup \{f_\top, f_\perp\}$.

Example 2.8.17. The truth function $f_\neg$ is linear; indeed, we have

$$f_\neg(a_0) = \mathbf{T} + (\mathbf{T} \wedge a_0) = \mathbf{T} + a_0$$

for every $a \in \mathbf{Bool}$.

The truth function $f_\leftrightarrow$ is also linear because

$$f_\leftrightarrow(a_0, a_1) = \mathbf{T} + a_0 + a_1$$

for $a_0, a_1 \in \mathbf{Bool}$.

For $n \geq 1$, the n-ary *parity function* $f_{P_n} \in \mathtt{TF}_n$ takes the value $\mathbf{T}$ if an even number of its arguments have the value $\mathbf{T}$. Every such function is linear since we can write

$$f_{P_n}(a_0, \ldots, a_{n-1}) = \mathbf{T} + a_0 + \cdots + a_{n-1}$$

for every $a_0, \ldots, a_{n-1} \in \mathbf{Bool}$. Note that $f_\neg = f_{P_1}$ and $f_\leftrightarrow = f_{P_2}$, so this observation generalizes the previous ones.

None of the functions $f_\wedge, f_\vee, f_\to$ is linear. We show this here only for $f_\to$. Later, in Example 2.8.33, we prove that $f_\wedge$ and $f_\vee$ are nonlinear, using a characterization of linear binary truth functions.

Suppose that $f_\to$ were linear. Then, there are $k, k_0, k_1 \in \mathbf{Bool}$ such that

$$a_0 \to a_1 = k + (k_0 \wedge a_0) + (k_1 \wedge a_1)$$

for every $a_0, a_1 \in \mathbf{Bool}$. This gives the following equalities:

$$\begin{aligned}
\mathbf{F} \to \mathbf{F} = \mathbf{T} &= k,\\
\mathbf{F} \to \mathbf{T} = \mathbf{T} &= k + k_1,\\
\mathbf{T} \to \mathbf{F} = \mathbf{F} &= k + k_0,\\
\mathbf{T} \to \mathbf{T} = \mathbf{T} &= k + k_0 + k_1.
\end{aligned}$$

The first three equalities give $k = \mathbf{T}$, $k_1 = \mathbf{F}$, and $k_0 = \mathbf{T}$. These values, however, do not satisfy the last equality, which is a contradiction.

In a similar way, one can show that the *minority function* $f_{\min} \in \mathtt{TF}_3$ defined by $f_{\min}(a_0, a_1, a_2) = \mathbf{T}$ if at most one of its arguments is $\mathbf{T}$ is not linear. ▯

We introduce a relation "$\leq$" on $\mathbf{Bool}$ by

$$\leq = \{(\mathbf{F},\mathbf{F}), (\mathbf{T},\mathbf{T}), (\mathbf{F},\mathbf{T})\}.$$

It is easy to check that $\leq$ is a partial order on $\mathbf{Bool}$. Also, for $n \in \mathbf{N}$, we use the partially ordered set $(\mathbf{Bool}^n, \leq)$, where $(a_0, \ldots, a_{n-1}) \leq (b_0, \ldots, b_{n-1})$ if $a_i \leq b_i$ for all i, $0 \leq i \leq n-1$.

Let $n \in \mathbf{N}$. We define a bijection $\beta_n : \mathbf{Bool}^n \longrightarrow \{0, \ldots, 2^n - 1\}$, which allows us to define a total order $\lhd$ on $\mathbf{Bool}^n$, by

$$\beta_n(a_0, \ldots, a_{n-1}) = 2^{n-1}d_0 + \cdots + 2d_{n-2} + d_{n-1},$$

where

$$d_i = \begin{cases} 1 & \text{if } a_i = \mathbf{T} \\ 0 & \text{if } a_i = \mathbf{F} \end{cases}$$

for $0 \leq i \leq n-1$. The relation $\lhd$ on $\mathbf{Bool}^n$ is given by $\vec{a} \lhd \vec{b}$ if $\beta_n(\vec{a}) \leq \beta_n(\vec{b})$. Since $(\mathbf{Bool}^n, \lhd)$ is a chain, we can list its elements as

$$\overrightarrow{b_0^n} \lhd \overrightarrow{b_1^n} \lhd \cdots \lhd \overrightarrow{b_{2^n-1}^n}\ .$$

Example 2.8.18. For the totally ordered set $(\mathbf{Bool}^3, \lhd)$, we have

$$\begin{gathered}
(\mathbf{F},\mathbf{F},\mathbf{F}) \lhd (\mathbf{F},\mathbf{F},\mathbf{T}) \lhd (\mathbf{F},\mathbf{T},\mathbf{F}) \lhd (\mathbf{F},\mathbf{T},\mathbf{T}) \lhd\\
(\mathbf{T},\mathbf{F},\mathbf{F}) \lhd (\mathbf{T},\mathbf{F},\mathbf{T}) \lhd (\mathbf{T},\mathbf{T},\mathbf{F}) \lhd (\mathbf{T},\mathbf{T},\mathbf{T}).
\end{gathered}$$

▯

Recall the notation $\delta(\vec{a}, \vec{b})$ introduced in Section 1.2.

Lemma 2.8.19. *Let $n \in \mathbf{N}$ and let $\vec{a}, \vec{b} \in \mathbf{Bool}^n$ be two sequences. If $\delta(\vec{a}, \vec{b}) = k$, then there is a sequence $\vec{c}(0), \ldots, \vec{c}(k)$ in $(\mathbf{Bool}^n, \leq)$ such that $\vec{c}(0) = \vec{a}$, $\vec{c}(k) = \vec{b}$ and $\delta(\vec{c}(i), \vec{c}(i+1)) = 1$ for $0 \leq i \leq k-1$. Furthermore, if $\vec{a} \leq \vec{b}$, then we may choose the sequence to be a chain $\vec{c}(0) < \cdots < \vec{c}(k)$ in $(\mathbf{Bool}^n, \leq)$.*

Proof. The proof is by induction on k and it is left to the reader. □

Definition 2.8.20. For $n \in \mathbf{N}$, a function $f \in \mathtt{TF}_n$ is *monotonic* if for every $(a_0, \ldots, a_{n-1}), (b_0, \ldots, b_{n-1}) \in \mathbf{Bool}^n$ we have

$$(a_0, \ldots, a_{n-1}) \leq (b_0, \ldots, b_{n-1}) \text{ implies}$$
$$f(a_0, \ldots, a_{n-1}) \leq f(b_0, \ldots, b_{n-1}).$$

The sets $\mathcal{MON}_*$, $\mathcal{MON}$, and $\mathcal{MON}_n$ (for $n \in \mathbf{N}$) are given by

$$\begin{aligned}\mathcal{MON}_* &= \{f \in \mathtt{TF}_* \mid f \text{ is monotonic}\},\\ \mathcal{MON} &= \mathcal{MON}_* \cap \mathtt{TF},\\ \mathcal{MON}_n &= \mathcal{MON}_* \cap \mathtt{TF}_n.\end{aligned}$$

▯

Both $f_\top$ and $f_\perp$ are trivially monotonic functions and hence $\mathcal{MON}_* = \mathcal{MON} \cup \{f_\top, f_\perp\}$.

A technical result that is a consequence of Lemma 2.8.19 is given in the following.

Lemma 2.8.21. *Let $f \in \mathtt{TF}_n - \mathcal{MON}$ be a nonmonotonic truth function. There exist $\vec{c}, \vec{d} \in \mathbf{Bool}^n$ such that $\vec{c} \leq \vec{d}$, $\delta(\vec{c}, \vec{d}) = 1$, $f(\vec{c}) = \mathbf{T}$ and $f(\vec{d}) = \mathbf{F}$.*

Proof. Since f is nonmonotonic, there is a pair of sequences

$$\vec{a} = (a_0, \ldots, a_{n-1}), \vec{b} = (b_0, \ldots, b_{n-1})$$

such that $\vec{a} \leq \vec{b}$, $f(a_0, \ldots, a_{n-1}) = \mathbf{T}$, and $f(b_0, \ldots, b_{n-1}) = \mathbf{F}$.

By Lemma 2.8.19, there is a chain $\vec{c}(0) < \cdots < \vec{c}(k)$ in $(\mathbf{Bool}^n, \leq)$ such that $\vec{c}(0) = \vec{a}$, $\vec{c}(k) = \vec{b}$ and $\delta(\vec{c}(i), \vec{c}(i+1)) = 1$ for $0 \leq i \leq k-1$. Since $f(\vec{a}) = \mathbf{T}$ and $f(\vec{b}) = \mathbf{F}$, there is j such that $0 \leq j \leq k-1$, $f(\vec{c}(j)) = \mathbf{T}$ and $f(\vec{c}(j+1)) = \mathbf{F}$. In this case, $\vec{c}(j)$ and $\vec{c}(j+1)$ are sequences that satisfy the conditions of the lemma. □

Definition 2.8.22. A function $f \in \mathtt{TF}_n$ is *symmetric* if for every permutation π of $\{0, \ldots, n-1\}$, we have $f(a_0, \ldots, a_{n-1}) = f(a_{\pi(0)}, \ldots, a_{\pi(n-1)})$ for every $a_0, \ldots, a_{n-1} \in \mathbf{Bool}$.

Let $1 \leq k \leq n$. The *threshold function* $\mathrm{th}_{k,n}$ is the n-ary truth function defined by

$$\mathrm{th}_{k,n}(a_0, \ldots, a_{n-1}) = \begin{cases} \mathbf{T} & \text{if at least } k \text{ among } a_0, \ldots, a_{n-1} \text{ are } \mathbf{T} \\ \mathbf{F} & \text{otherwise.} \end{cases}$$

□

Note that the threshold functions are exactly the nonconstant, monotonic, symmetric truth functions.

Definition 2.8.23. For each $n \in \mathbf{N}$ and $\vec{b} \in \mathbf{Bool}^n$, the *$n$-ary minterm function* generated by $\vec{b}$ is the function $f_{\vec{b}} \in \mathtt{TF}_n$ given by

$$f_{\vec{b}}(x_0, \ldots, x_{n-1}) = (x_0, \ldots, x_{n-1})^{\vec{b}}$$

for every $(x_0, \ldots, x_{n-1}) \in \mathbf{Bool}^n$.

Similarly, for every $n \in \mathbf{N}$ and $\vec{b} \in \mathbf{Bool}^n$, the *$n$-ary maxterm function* generated by $\vec{b}$ is the function $g_{\vec{b}} \in \mathtt{TF}_n$ given by

$$g_{\vec{b}}(x_0, \ldots, x_{n-1}) = x_0^{b_0} \vee \cdots \vee x_{n-1}^{b_{n-1}}$$

for every $(x_0, \ldots, x_{n-1}) \in \mathbf{Bool}^n$. □

Note that if $\vec{b} = (b_0, \ldots, b_{n-1})$, then

$$f_{\vec{b}}(x_0, \ldots, x_{n-1}) = \begin{cases} \mathbf{T} & \text{if } x_0 = b_0, \ldots x_{n-1} = b_{n-1}, \\ \mathbf{F} & \text{otherwise,} \end{cases}$$

for every $x_0, \ldots, x_{n-1} \in \mathbf{Bool}$.

For a sequence $\vec{b} = (b_0, \ldots, b_{n-1}) \in \mathbf{Bool}^n$, we denote by $\neg\vec{b}$ the sequence $\neg\vec{b} = (\overline{b_0}, \ldots, \overline{b_{n-1}})$. It is easy to see that

$$g_{\vec{b}} = \overline{f_{\neg\vec{b}}},$$
$$f_{\vec{b}} = \overline{g_{\neg\vec{b}}},$$
$$\overline{g_{\vec{b}}} = f_{\neg\vec{b}},$$
$$\overline{f_{\vec{b}}} = g_{\neg\vec{b}}.$$

Also, $g_{\vec{b}}$ is the dual of the mintern function $f_{\vec{b}}$.

There are 2^n distinct n-ary minterm functions and an equal number of n-ary maxterm functions.

For $f \in \mathtt{TF}_n$, we denote by $T(f)$ and $F(f)$ the sets

$$T(f) = \{\vec{b} \in \mathbf{Bool}^n | f(\vec{b}) = \mathbf{T}\},$$

$$F(f) = \{\vec{b} \in \mathbf{Bool}^n | f(\vec{b}) = \mathbf{F}\}.$$

Definition 2.8.24. Let $m, n \in \mathbf{N}$ with $n \geq 1$.

An n-ary conjunction is a truth function $f \in \mathtt{TF}_n$ such that $f(\vec{x}) = \ell_0(\vec{x}) \wedge \cdots \wedge \ell_{m-1}(\vec{x})$ for all $\vec{x} \in \mathbf{Bool}^n$, where each ℓ_i, $0 \leq i \leq m-1$, is either an n-ary projection π_j^n or is $f_\neg(\pi_j^n)$, for some $j \in \mathbf{N}$. If every ℓ_i is an n-ary projection, then f is a *positive n-ary conjunction.*

An n-ary disjunction is a truth function $f \in \mathtt{TF}_n$, such that $f(\vec{x}) = \ell_0(\vec{x}) \vee \cdots \vee \ell_{m-1}(\vec{x})$ for all $\vec{x} \in \mathbf{Bool}^n$, where each ℓ_i, $0 \leq i \leq m-1$, is either an n-ary projection π_j^n or is $f_\neg(\pi_j^n)$, for some $j \in \mathbf{N}$. If every ℓ_i is an n-ary projection, then f is a *positive n-ary disjunction.* $\square$

Example 2.8.25. The function $f \in \mathtt{TF}_4$ given by $f(x_0, x_1, x_2, x_3) = x_1 \wedge \overline{x_3}$ for $(x_0, x_1, x_2, x_3) \in \mathbf{Bool}^4$ is a 4-ary conjunction since we have

$$f(x_0, x_1, x_2, x_3) = \pi_1^4(x_0, x_1, x_2, x_3) \wedge f_\neg(\pi_3^4(x_0, x_1, x_2, x_3))$$

for every $x_0, x_1, x_2, x_3 \in \mathbf{Bool}$.

The function $g \in \mathtt{TF}_4$ given by

$$g(x_0, x_1, x_2, x_3) = \overline{x_0} \vee x_2 \vee \overline{x_3}$$

for every $x_0, x_1, x_2, x_3 \in \mathbf{Bool}$ is a 4-ary disjunction since

$$\begin{aligned} &g(x_0, x_1, x_2, x_3) \\ &\quad = f_\neg(\pi_0^4(x_0, x_1, x_2, x_3)) \vee \pi_2^4(x_0, x_1, x_2, x_3) \vee f_\neg(\pi_3^4(x_0, x_1, x_2, x_3)) \end{aligned}$$

for every $x_0, x_1, x_2, x_3 \in \mathbf{Bool}$. $\square$

Let I be a finite set with $|I| = n$, and let $h : I \longrightarrow \mathbf{Bool}$. We define

$$\bigvee\{f(i) \mid i \in I\} = \bigvee\nolimits_{0 \leq j \leq n-1} h(i_j),$$

$$\bigwedge\{f(i) \mid i \in I\} = \bigwedge\nolimits_{0 \leq j \leq n-1} h(i_j),$$

$$\sum\{f(i) \mid i \in I\} = \sum_{0 \leq j \leq n-1} h(i_j),$$

where $(i_0, \ldots, i_{n-1})$ is an arbitrary permutation of I. By Theorem 2.8.8, these definitions are independent of the permutation chosen.

Definition 2.8.26. Let f be an n-ary truth function. If

$$f(\vec{x}) = f_0(\vec{x}) \vee \cdots \vee f_{k-1}(\vec{x}), \tag{2.4}$$

for every $\vec{x} \in \mathbf{Bool}^n$, where $f_0, \ldots, f_{k-1}$ are n-ary conjunctions, then the right member of equation (2.4) is called a *disjunctive normal form of f*.

If

$$f(\vec{x}) = f_0(\vec{x}) \wedge \cdots \wedge f_{k-1}(\vec{x}), \tag{2.5}$$

for every $\vec{x} \in \mathbf{Bool}^n$, where $f_0, \ldots, f_{k-1}$ are n-ary disjunctions, then the right member of equation (2.5) is called a *conjunctive normal form of f*. ▯

There exists at least one disjunctive normal form for any truth function and one conjunctive normal form, as shown by the following two theorems.

Theorem 2.8.27 (Full Disjunctive Normal Form Theorem). *For every function $f \in \mathtt{TF}_n$, we have*

$$f(x_0, \ldots, x_{n-1}) = \bigvee \{f_{\vec{b}}(x_0, \ldots, x_{n-1}) | \vec{b} \in T(f)\} \tag{2.6}$$

*for every $x_0, \ldots, x_{n-1} \in$ **Bool**. The right member of equality (2.6) is called* the full disjunctive normal form of the function f.

Proof. Consider an n-tuple $\vec{x} = (x_0, \ldots, x_n) \in \mathbf{Bool}^n$. Two cases are possible:

(1) If $\vec{x} \notin T(f)$, then $f(\vec{x}) = \mathbf{F}$. Since $\vec{x} \neq \vec{b}$ for every $\vec{b} \in T(f)$, we have $f_{\vec{b}}(\vec{x}) = \mathbf{F}$, so both sides are equal to $\mathbf{F}$.

(2) If $\vec{x} \in T(f)$, then $f_{\vec{x}}(\vec{x}) = \mathbf{T}$. Since the right member of the equality of the theorem is a disjunction, it follows that it is equal to $\mathbf{T}$ and so, the equality is satisfied.

□

Theorem 2.8.28 (Full Conjunctive Normal Form Theorem). *For every function $f \in \mathtt{TF}_n$, we have*

$$f(x_0, \ldots, x_{n-1}) = \bigwedge \{g_{\vec{b}'}(x_0, \ldots, x_{n-1}) | \vec{b} \in F(f)\} \tag{2.7}$$

for every $x_0, \ldots, x_{n-1} \in \mathbf{Bool}$*, where* $\vec{b'} = \neg\vec{b}$ *for every* $\vec{b} \in \mathbf{Bool}^n$*. The right member of equality (2.7) is called* the full conjunctive normal form of the function f*.*

Proof. By applying the full disjunctive normal form theorem to the function $\bar{f}$, we obtain

$$\begin{aligned}\bar{f}(\vec{x}) &= \bigvee\{f_{\vec{b}}(x_0, \ldots, x_{n-1}) | \vec{b} \in T(\bar{f})\}\\ &= \bigvee\{f_{\vec{b}}(x_0, \ldots, x_{n-1}) | \vec{b} \in F(f)\}.\end{aligned}$$

This, in turn, implies, by the De Morgan laws

$$\begin{aligned}f(\vec{x}) &= \bigwedge\{\bar{f}_{\vec{b}}(\vec{x}) | \vec{b} \in F(f)\}\\ &= \bigwedge\{g_{\vec{b'}}(\vec{x}) | \vec{b} \in F(f)\},\end{aligned}$$

for every $\vec{x} \in \mathbf{Bool}^n$. □

Example 2.8.29. Consider the function $f : \mathbf{Bool}^3 \longrightarrow \mathbf{Bool}$ defined by the following table:

x_0	x_1	x_2	$f(x_0, x_1, x_2)$
F	**F**	**F**	**T**
F	**F**	**T**	**F**
F	**T**	**F**	**F**
F	**T**	**T**	**T**
T	**F**	**F**	**F**
T	**F**	**T**	**T**
T	**T**	**F**	**T**
T	**T**	**T**	**T**

The sets $T(f)$ and $F(f)$ are given by

$$\begin{aligned}T(f) = \{&(\mathbf{F}, \mathbf{F}, \mathbf{F}), (\mathbf{F}, \mathbf{T}, \mathbf{T}), (\mathbf{T}, \mathbf{F}, \mathbf{T}),\\ &(\mathbf{T}, \mathbf{T}, \mathbf{F}), (\mathbf{T}, \mathbf{T}, \mathbf{T})\},\\ F(f) = \{&(\mathbf{F}, \mathbf{F}, \mathbf{T}), (\mathbf{F}, \mathbf{T}, \mathbf{F}), (\mathbf{T}, \mathbf{F}, \mathbf{F})\}.\end{aligned}$$

Consequently, the full disjunctive normal form of f is given by

$$\begin{aligned}f(x_0, x_1, x_2) = &(\bar{x_0} \wedge \bar{x_1} \wedge \bar{x_2}) \vee (\bar{x_0} \wedge x_1 \wedge x_2)\\ &\vee (x_0 \wedge \bar{x_1} \wedge x_2) \vee (x_0 \wedge x_1 \wedge \bar{x_2})\\ &\vee (x_0 \wedge x_1 \wedge x_2),\end{aligned}$$

while its full disjunctive normal form is

$$f(x_0, x_1, x_2) = (x_0 \lor x_1 \lor \bar{x_2}) \land (x_0 \lor \bar{x_1} \lor x_2)$$
$$\land (\bar{x_0} \lor x_1 \lor x_2).$$

∎

Let us denote by P_m^n the set of all subsets of $\{0, \ldots, n-1\}$ having m elements, for $m, n \in \mathbf{N}$ and $0 \leq m \leq n$. P^n will denote the power set of $\{0, \ldots, n-1\}$. Observe that, for $n \geq 1$, the function taking $L \in P^{n-1}$ to $L \cup \{n-1\}$ is a bijection between P^{n-1} and $P^n - P^{n-1}$ for $n \geq 1$. For $\vec{x} = (x_0, \ldots, x_{n-1}) \in \mathbf{Bool}^n$ and $L = \{i_0, \ldots, i_{m-1}\}$, $0 \leq i_0 < \cdots < i_{m-1} \leq n-1$, denote by $\vec{x}_L$ the conjunction $x_{i_0} \land \ldots \land x_{i_{m-1}}$.

For monotonic truth functions, we have a refinement of Theorem 2.8.27.

Theorem 2.8.30 (Monotonic Disjunctive Normal Form Theorem). *For every monotonic function $f \in \mathtt{TF}_n$, there is a disjunctive normal form of f that consists of positive n-ary conjunctions.*

Proof. If $g = \ell_0 \land \cdots \land \ell_{m-1}$ is an n-ary conjunction, we denote by g^+ the n-ary conjunction $\ell_{i_0} \land \cdots \land \ell_{i_{p-1}}$, where $\ell_{i_0}, \ldots \ell_{i_{p-1}}$ are those members of $\{\ell_0, \ldots, \ell_{m-1}\}$ that are n-ary projections. It is easy to see that $g(\vec{x}) \leq g^+(\vec{x})$, for all $\vec{x} \in \mathbf{Bool}^n$. (Note that when $p = 0$, then $g^+(x_0, \ldots, x_{n-1}) = \mathbf{T}$ for all $x_0, \ldots, x_{n-1} \in \mathbf{Bool}$.)

Let $f = \bigvee_{0 \leq i \leq k-1} f_i$ be the full disjunctive normal form of f and let $h = \bigvee_{0 \leq i \leq k-1} f_i^+$. It is clear that $f(\vec{x}) \leq h(\vec{x})$ for all $\vec{x} \in \mathbf{Bool}^n$. To show the opposite inequality, let $\vec{x} \in \mathbf{Bool}^n$ be such that $h(\vec{x}) = \mathbf{T}$. Then, there is an i such that $0 \leq i \leq k-1$ and $f_i^+(\vec{x}) = \mathbf{T}$. We can write $f_i(\vec{x}) = \ell_0(\vec{x}) \land \cdots \land \ell_{n-1}(\vec{x})$, where ℓ_r is either π_r^n or $f_\neg(\pi_r^n)$, and

$$f_i^+(\vec{x}) = \ell_{j_0}(\vec{x}) \land \cdots \land \ell_{j_{p-1}}(\vec{x}) = x_{j_0} \land \cdots \land x_{j_{p-1}} = \mathbf{T}.$$

Define $\vec{x}' \in \mathbf{Bool}^n$ by

$$x_i' = \begin{cases} \mathbf{T} & \text{if } i \in \{j_0, \ldots, j_{p-1}\} \\ \mathbf{F} & \text{otherwise.} \end{cases}$$

Since $\ell_0(\vec{x}') = \cdots = \ell_{n-1}(\vec{x}') = \mathbf{T}$, we have $f_i(\vec{x}') = \mathbf{T}$, so $f(\vec{x}') = \mathbf{T}$. The inequality $\vec{x}' \leq \vec{x}$ and the monotonicity of f imply $f(\vec{x}) = \mathbf{T}$, as desired. □

Theorem 2.8.31 (Polynomial Normal Form Theorem). *Let $f \in \mathtt{TF}_n$, $n \geq 1$ be a truth function. There is an indexed set $\{a_L \mid L \in P^n\}$ of elements of* **Bool** *such that*

$$f(\vec{x}) = \sum \{a_L \wedge \vec{x}_L | L \in P^n\},$$

for every $\vec{x} \in \mathbf{Bool}^n$. The coefficients a_L are uniquely determined by the function f.

If $L = \{i_0, \ldots, i_{m-1}\}$, $i_0 < \cdots < i_{m-1}$, we shall denote the coefficient a_L by $a_{i_0 \ldots i_{m-1}}$. Also, $a_\emptyset$ will be denoted simply by a.

Proof. The argument is by induction on n, the number of arguments of the truth function f. If $n = 1$, we have the cases summarized as follows:

f	a	a_0	Polynomial Form
$f = \pi_0^1$	**F**	**T**	$f(x) = \mathbf{F} + \mathbf{T} \wedge x$
$f = f_0$	**F**	**F**	$f(x) = \mathbf{F} + \mathbf{F} \wedge x$
$f = f_1$	**T**	**F**	$f(x) = \mathbf{T} + \mathbf{F} \wedge x$
$f = f_\neg$	**T**	**T**	$f(x) = \mathbf{T} + \mathbf{T} \wedge x$

Suppose that for $n \geq 2$ every function of $n - 1$ arguments can be written as above and consider the functions $f_0, f_1 : \mathbf{Bool}^{n-1} \longrightarrow \mathbf{Bool}$ given by

$$f_0(x_0, \ldots, x_{n-2}) = f(x_0, \ldots, x_{n-2}, \mathbf{F}),$$
$$f_1(x_0, \ldots, x_{n-2}) = f(x_0, \ldots, x_{n-2}, \mathbf{T}),$$

for every $x_0, \ldots, x_{n-2} \in \mathbf{Bool}$. Note that

$$f(x_0, \ldots, x_{n-1}) = f_0(x_0, \ldots, x_{n-2}) + (f_0(x_0, \ldots, x_{n-2}) + f_1(x_0, \ldots, x_{n-2})) \wedge x_{n-1}$$

for every $x_0, \ldots, x_{n-2} \in \mathbf{Bool}$. By the inductive hypothesis, there are a_L^0, a_L^1 such that

$$f_0(x_0, \ldots, x_{n-2}) = \sum \{a_L^0 \wedge x_L | L \in P^{n-1}\}$$

and

$$f_1(x_0, \ldots, x_{n-2}) = \sum \{a_L^1 \wedge x_L | L \in P^{n-1}\},$$

for every $x_0, \ldots, x_{n-2} \in \mathbf{Bool}$.

By combining the expansions of the truth functions, we obtain the needed expansion for the function f:

$$\begin{aligned} f(x_0,\ldots,x_{n-1}) = {} & \sum\{a^0_L \wedge x_L \mid L \in P^{n-1}\} \\ & + \Big(\sum\{a^0_L \wedge x_L \mid L \in P^{n-1}\} \\ & + \sum\{a^1_L \wedge x_L \mid L \in P^{n-1}\}\Big) \wedge x_{n-1}. \end{aligned}$$

The last summand can be rewritten as follows:

$$\begin{aligned} & \Big(\sum\{a^0_L \wedge x_L \mid L \in P^{n-1}\} + \sum\{a^1_L \wedge x_L \mid L \in P^{n-1}\}\Big) \wedge x_{n-1} \\ & = \sum\{a^0_L \wedge x_L + a^1_L \wedge x_L \mid L \in P^{n-1}\} \wedge x_{n-1} \\ & \quad \text{(by Part (3) of Theorem 2.8.10)} \\ & = \sum\{(a^0_L + a^1_L) \wedge x_L \mid L \in P^{n-1}\} \wedge x_{n-1} \\ & = \sum\{(a^0_L + a^1_L) \wedge (x_L \wedge x_{n-1}) \mid L \in P^{n-1}\} \\ & \quad \text{(by Theorem 2.8.11 and associativity of } \wedge) \\ & = \sum\{(a^0_L + a^1_L) \wedge x_{L \cup \{n-1\}} \mid L \in P^{n-1}\} \\ & = \sum\{(a^0_{L-\{n-1\}} + a^1_{L-\{n-1\}}) \wedge x_L \mid L \in P^n - P^{n-1}\}. \end{aligned}$$

Thus, we can write

$$\begin{aligned} & f(x_0,\ldots,x_{n-1}) \\ & \quad = \sum\{a^0_L \wedge x_L \mid L \in P^{n-1}\} \\ & \qquad + \sum\{(a^0_{L-\{n-1\}} + a^1_{L-\{n-1\}}) \wedge x_L \mid L \in P^n - P^{n-1}\} \\ & \quad = \sum\{a_L \wedge x_L \mid L \in P^n\}, \end{aligned}$$

where

$$a_L = \begin{cases} a^0_L & \text{if } n-1 \notin L, \\ a^0_{L-\{n-1\}} + a^1_{L-\{n-1\}} & \text{if } n-1 \in L. \end{cases}$$

The uniqueness of the polynomial normal form for truth functions follows from the fact that for $n \geq 1$ there are 2^{2^n} distinct expressions of the form $\sum\{a_L \wedge \vec{x}_L \mid L \in P^n\}$ and an equal number of n-ary truth functions. Since every n-ary truth function has a polynomial normal form, it follows that no such function can have more than one such form. $\square$

Example 2.8.32. The functions $f_\wedge, f_\vee$ are not linear because their polynomial normal forms are given by

$$\begin{aligned}f_\wedge(x_0, x_1) &= \mathbf{F} + (\mathbf{F} \wedge x_0) + (\mathbf{F} \wedge x_1) + (\mathbf{T} \wedge x_0 \wedge x_1),\\ f_\vee(x_0, x_1) &= \mathbf{F} + (\mathbf{T} \wedge x_0) + (\mathbf{T} \wedge x_1) + (\mathbf{T} \wedge x_0 \wedge x_1)\end{aligned}$$

for $x_0, x_1 \in \mathbf{Bool}$. If either of these functions were linear, this would contradict the uniqueness of polynomial normal form. ▯

Example 2.8.33. For a binary truth function $f : \mathbf{Bool}^2 \longrightarrow \mathbf{Bool}$, the polynomial normal form is

$$f(x_0, x_1) = a + a_0 \wedge x_0 + a_1 \wedge x_1 + a_{01} \wedge x_0 \wedge x_1,$$

for every $x_0, x_1 \in \mathbf{Bool}$. This gives the following equalities:

$$\begin{aligned}f(\mathbf{F}, \mathbf{F}) &= a,\\ f(\mathbf{F}, \mathbf{T}) &= a + a_1,\\ f(\mathbf{T}, \mathbf{F}) &= a + a_0,\\ f(\mathbf{T}, \mathbf{T}) &= a + a_0 + a_1 + a_{01}.\end{aligned}$$

These equalities allow us to compute the coefficients:

$$\begin{aligned}a &= f(\mathbf{F}, \mathbf{F}),\\ a_0 &= f(\mathbf{F}, \mathbf{F}) + f(\mathbf{T}, \mathbf{F}),\\ a_1 &= f(\mathbf{F}, \mathbf{F}) + f(\mathbf{F}, \mathbf{T}),\\ a_{01} &= f(\mathbf{F}, \mathbf{F}) + f(\mathbf{F}, \mathbf{T}) + f(\mathbf{T}, \mathbf{F}) + f(\mathbf{T}, \mathbf{T}).\end{aligned}$$

This shows that a binary truth function f is linear if and only if $f(\mathbf{F}, \mathbf{F}) + f(\mathbf{F}, \mathbf{T}) + f(\mathbf{T}, \mathbf{F}) + f(\mathbf{T}, \mathbf{T}) = \mathbf{F}$. In other words, f is linear if and only if the sequence $(f(\mathbf{F}, \mathbf{F}), f(\mathbf{F}, \mathbf{T}), f(\mathbf{T}, \mathbf{F}), f(\mathbf{T}, \mathbf{T}))$ contains an even number of entries that are $\mathbf{T}$. This implies that neither $f_\wedge$ nor $f_\vee$ is linear. In fact, the binary truth functions that are linear are the constant functions, the projections and their negations, $f_\leftrightarrow$, and its negation. ▯

The next normal form, called Lupanov's (k, s)-representation, was introduced in [28] and will be used in Section 2.11 to derive minimal-size implementations of truth functions.[6] If $B \subseteq \mathbf{Bool}^n$ and $f \in \mathtt{TF}_n$, define the *trace of f on the set B* as $f_B \in \mathtt{TF}_n$, where

$$f_B(x) = \begin{cases} f(x) & \text{if } x \in B \\ \mathbf{F} & \text{otherwise.} \end{cases}$$

Lemma 2.8.34. *Let $n \in \mathbf{N}$ and let $B_0, \ldots, B_{\ell-1}$ be a collection of subsets of $\mathbf{Bool}^n$ whose union is $\mathbf{Bool}^n$. For $f \in \mathtt{TF}_n$, we have*

$$f = \bigvee\{f_{B_i} \mid 0 \leq i \leq \ell - 1\}.$$

Proof. Let $x \in \mathbf{Bool}^n$. If $f(x) = \mathbf{T}$, then $f_{B_i}(x) = \mathbf{T}$ for those B_i such that $x \in B_i$ and there is at least one such B_i because $\bigcup\{B_i \mid 0 \leq i \leq \ell - 1\} = \mathbf{Bool}^n$. If $f(x) = \mathbf{F}$, then $f_{B_i}(x) = \mathbf{F}$ for every B_i. The result follows from these observations. $\square$

Definition 2.8.35. Let $n, k, s \in \mathbf{N}$ be such that $1 \leq k \leq n$ and $1 \leq s \leq 2^k$. An *s-partition of* $\mathbf{Bool}^k$ is a partition $A_0, \ldots, A_{\ell-1}$ of $\mathbf{Bool}^k$ such that $|A_0| = \cdots = |A_{\ell-2}| = s$ and $|A_{\ell-1}| \leq s$.

If $A_0, \ldots, A_{\ell-1}$ is an s-partition of $\mathbf{Bool}^k$, then the *(k, s)-partition of* $\mathbf{Bool}^n$ induced by the s-partition is $B_0, \ldots, B_{\ell-1}$, where

$$B_i = \{(x_0, \ldots, x_{n-1}) \mid (x_0, \ldots, x_{k-1}) \in A_i\}$$

for $0 \leq i \leq \ell - 1$. $\square$

Note that an s-partition of $\mathbf{Bool}^k$, and therefore, a (k, s)-partition of $\mathbf{Bool}^n$, has $\ell = \lceil \frac{2^k}{s} \rceil$ blocks.

Example 2.8.36. The *standard s-partition of* $\mathbf{Bool}^k$ is the partition whose blocks consist of consecutive k-tuples in lexicographic order.

[6]O. B. Lupanov was born in St. Petersburg in 1932 and died in Moscow in 2006. He graduated from Moscow State University where he served as a professor and a dean. He is known for the representation of Boolean functions which bears his name.

For example, the standard 3-partition of $\mathbf{Bool}^3$ has the following blocks:

$$\begin{aligned}
A_0 &= \{(\mathbf{F},\mathbf{F},\mathbf{F}),(\mathbf{F},\mathbf{F},\mathbf{T}),(\mathbf{F},\mathbf{T},\mathbf{F})\},\\
A_1 &= \{(\mathbf{F},\mathbf{T},\mathbf{T}),(\mathbf{T},\mathbf{F},\mathbf{F}),(\mathbf{T},\mathbf{F},\mathbf{T})\},\\
A_2 &= \{(\mathbf{T},\mathbf{T},\mathbf{F}),(\mathbf{T},\mathbf{T},\mathbf{T})\}.
\end{aligned}$$

The $(3,3)$-partition of $\mathbf{Bool}^5$ induced by the partition A_0, A_1, A_2 consists of three blocks B_0, B_1, B_2, where $|B_0| = |B_1| = 12$ and $|B_2| = 8$. □

Definition 2.8.37. Let $f \in \mathtt{TF}_n$ and let $A_0, \ldots, A_{\ell-1}$ be an s-partition of $\mathbf{Bool}^k$, where $1 \leq k \leq n$ and $1 \leq s \leq 2^k$. Denote the induced (k,s)-partition of $\mathbf{Bool}^n$ by $B_0, \ldots, B_{\ell-1}$.

The *first Lupanov decomposition* of f generated by $A_0, \ldots, A_{\ell-1}$ is the set of functions $\{f_0, \ldots, f_{\ell-1}\}$, where $f_i = f_{B_i}$ for $0 \leq i \leq \ell-1$. □

By Lemma 2.8.34, f equals the disjunction of the functions in any of its first Lupanov decompositions.

Example 2.8.38. Let $f \in \mathtt{TF}_4$ be given by the following table:

x_0	x_1	x_2	x_3	$f(x_0,x_1,x_2,x_3)$
F	**F**	**F**	**F**	**F**
F	**F**	**F**	**T**	**T**
F	**F**	**T**	**F**	**F**
F	**F**	**T**	**T**	**F**
F	**T**	**F**	**F**	**T**
F	**T**	**F**	**T**	**F**
F	**T**	**T**	**F**	**T**
F	**T**	**T**	**T**	**F**
T	**F**	**F**	**F**	**F**
T	**F**	**F**	**T**	**T**
T	**F**	**T**	**F**	**T**
T	**F**	**T**	**T**	**T**
T	**T**	**F**	**F**	**T**
T	**T**	**F**	**T**	**T**
T	**T**	**T**	**F**	**T**
T	**T**	**T**	**T**	**T**

Consider the 2-partition $A_0 = \{(\mathbf{F},\mathbf{F}),(\mathbf{F},\mathbf{T})\}, A_1 = \{(\mathbf{T},\mathbf{F}), (\mathbf{T},\mathbf{T})\}$ of $\mathbf{Bool}^2$ and the corresponding $(2,2)$-partition of $\mathbf{Bool}^4$. To obtain the first Lupanov decomposition of f, it is convenient to redraw the table defining f as follows:

		x_2	**F**	**F**	**T**	**T**
		x_3	**F**	**T**	**F**	**T**
	x_0	x_1				
A_0	**F**	**F**	**F**	**T**	**F**	**F**
	F	**T**	**T**	**F**	**T**	**F**
A_1	**T**	**F**	**F**	**T**	**T**	**T**
	T	**T**	**T**	**T**	**T**	**T**

The decomposition of f consists of the two functions f_0, f_1 given by

f_0		x_2	**F**	**F**	**T**	**T**
		x_3	**F**	**T**	**F**	**T**
	x_0	x_1				
A_0	**F**	**F**	**F**	**T**	**F**	**F**
	F	**T**	**T**	**F**	**T**	**F**
A_1	**T**	**F**	**F**	**F**	**F**	**F**
	T	**T**	**F**	**F**	**F**	**F**

f_1		x_2	**F**	**F**	**T**	**T**
		x_3	**F**	**T**	**F**	**T**
	x_0	x_1				
A_0	**F**	**F**	**F**	**F**	**F**	**F**
	F	**T**	**F**	**F**	**F**	**F**
A_1	**T**	**F**	**F**	**T**	**T**	**T**
	T	**T**	**T**	**T**	**T**	**T**

$\square$

We now proceed to further decompose each of the functions f_i in a first Lupanov decomposition of f.

Definition 2.8.39. Let $f \in \mathtt{TF}_n$ and let $A_0, \ldots, A_{\ell-1}$ be an s-partition of $\mathbf{Bool}^k$, where $1 \leq k \leq n$ and $1 \leq s \leq 2^k$, and let $\{f_0, \ldots, f_{\ell-1}\}$ be the first Lupanov decomposition of f generated by the partition.

For $\omega : A_i \longrightarrow \mathbf{Bool}$, let $C_{A_i,\omega}$ be the subset of $\mathbf{Bool}^{n-k}$ given by

$$\begin{aligned} C_{A_i,\omega} = \{(y_0, \ldots, y_{n-k-1}) \mid f(&x_0, \ldots, x_{k-1}, y_0, \ldots, y_{n-k-1}) \\ &= \omega(x_0, \ldots, x_{k-1}) \text{ for every } (x_0, \ldots, x_{k-1}) \in A_i\} \end{aligned}$$

and let $D_{A_i,\omega}$ be the subset of $\mathbf{Bool}^n$ defined by

$$\begin{aligned} D_{A_i,\omega} = \{(x_0, \ldots, x_{k-1}, y_0, \ldots, y_{n-k-1}) \mid (x_0, \ldots, x_{k-1}) \in \mathbf{Bool}^k, \\ (y_0, \ldots, y_{n-k-1}) \in C_{A_i,\omega}\}. \end{aligned}$$

The function $f_{i,\omega}$ is the trace of the function f_i on the set $D_{A_i,\omega}$. □

Observe that for a truth function $f \in \mathtt{TF}_n$ and each i, the block A_i of a (k,s)-partition generates a partition $\{C_{A_i,\omega} \mid \omega : A_i \longrightarrow \mathbf{Bool} \text{ and } C_{A_i,\omega} \neq \emptyset\}$ of $\mathbf{Bool}^{n-k}$. Therefore, the collection

$$\{D_{A_i,\omega} \mid \omega : A_i \longrightarrow \mathbf{Bool} \text{ and } C_{A_i,\omega} \neq \emptyset\}$$

is a partition of $\mathbf{Bool}^n$.

Theorem 2.8.40. *Let $f \in \mathtt{TF}_n$ and let $\{f_0, \ldots, f_{\ell-1}\}$ be the first Lupanov decomposition generated by a (k,s)-partition $A_0, \ldots, A_{\ell-1}$. We have*

$$f_i = \bigvee \{f_{i,\omega} \mid \omega : A_i \longrightarrow \mathbf{Bool} \text{ and } C_{A_i,\omega} \neq \emptyset\}$$

for $0 \leq i \leq \ell - 1$ and

$$f = \bigvee \{f_{i,\omega} \mid 0 \leq i \leq \ell - 1, \omega : A_i \longrightarrow \mathbf{Bool} \text{ and } C_{A_i,\omega} \neq \emptyset\}.$$

Proof. The first equality of the theorem follows from Lemma 2.8.34 and the fact that $\bigcup \{D_{A_i,\omega} \mid \omega : A_i \longrightarrow \mathbf{Bool} \text{ and } C_{A_i,\omega} \neq \emptyset\} = \mathbf{Bool}^n$. Since f is the disjunction of the functions f_i, the second equality follows immediately from the first. □

Definition 2.8.41. Let $f \in \mathtt{TF}_n$ and let $A_0, \ldots, A_{\ell-1}$ be an s-partition of $\mathbf{Bool}^k$, where $1 \leq k \leq n$ and $1 \leq s \leq 2^k$. The *second Lupanov decomposition* generated by $A_0, \ldots, A_{\ell-1}$ is the set of functions $\{f_{i,\omega} \mid 0 \leq i \leq \ell - 1, \omega : A_i \longrightarrow \mathbf{Bool} \text{ and } C_{A_i,\omega} \neq \emptyset\}$. □

Example 2.8.42. For the function considered in Example 2.8.38, the sets $C_{A_i,\omega}$ are given by the following tables:

		ω_0	ω_1	ω_2	ω_3
A_0	**F F**	**F**	**F**	**T**	**T**
	F T	**F**	**T**	**F**	**T**
$C_{A_0,\omega}$		$\begin{pmatrix}\mathbf{T}\\ \mathbf{T}\end{pmatrix}$	$\begin{pmatrix}\mathbf{F}\\ \mathbf{F}\end{pmatrix}, \begin{pmatrix}\mathbf{T}\\ \mathbf{F}\end{pmatrix}$	$\begin{pmatrix}\mathbf{F}\\ \mathbf{T}\end{pmatrix}$	$\emptyset$

and

			ω_4	ω_5	ω_6	ω_7
A_1	**T**	**F**	**F**	**F**	**T**	**T**
	T	**T**	**F**	**T**	**F**	**T**
$C_{A_1,\omega}$			$\emptyset$	$\begin{pmatrix}\mathbf{F}\\ \mathbf{F}\end{pmatrix}$	$\emptyset$	$\begin{pmatrix}\mathbf{F}\\ \mathbf{T}\end{pmatrix}, \begin{pmatrix}\mathbf{T}\\ \mathbf{F}\end{pmatrix}, \begin{pmatrix}\mathbf{T}\\ \mathbf{T}\end{pmatrix}$

If $C_{A_i,\omega} = \emptyset$, then $f_{i,\omega}$ is the n-ary constant truth function whose value is **F**. These functions are not part of the second Lupanov decomposition. Thus, the second Lupanov decomposition of f consists of the functions $\{f_{0,\omega_0}, f_{0,\omega_1}, f_{0,\omega_2}, f_{1,\omega_5}, f_{1,\omega_7}\}$, whose tables are given in Figure 2.19.

$\square$

f_{0,ω_0}		x_2	**F**	**F**	**T**	**T**
		x_3	**F**	**T**	**F**	**T**
	x_0	x_1				
A_0	**F**	**F**	**F**	**F**	**F**	**F**
	F	**T**	**F**	**F**	**F**	**F**
A_1	**T**	**F**	**F**	**F**	**F**	**F**
	T	**T**	**F**	**F**	**F**	**F**

f_{1,ω_5}		x_2	**F**	**F**	**T**	**T**
		x_3	**F**	**T**	**F**	**T**
	x_0	x_1				
A_0	**F**	**F**	**F**	**F**	**F**	**F**
	F	**T**	**F**	**F**	**F**	**F**
A_1	**T**	**F**	**F**	**F**	**F**	**F**
	T	**T**	**T**	**F**	**F**	**F**

f_{0,ω_1}		x_2	**F**	**F**	**T**	**T**
		x_3	**F**	**T**	**F**	**T**
	x_0	x_1				
A_0	**F**	**F**	**F**	**F**	**F**	**F**
	F	**T**	**T**	**F**	**T**	**F**
A_1	**T**	**F**	**F**	**F**	**F**	**F**
	T	**T**	**F**	**F**	**F**	**F**

f_{1,ω_7}		x_2	**F**	**F**	**T**	**T**
		x_3	**F**	**T**	**F**	**T**
	x_0	x_1				
A_0	**F**	**F**	**F**	**F**	**F**	**F**
	F	**T**	**F**	**F**	**F**	**F**
A_1	**T**	**F**	**F**	**T**	**T**	**T**
	T	**T**	**F**	**T**	**T**	**T**

f_{0,ω_2}		x_2	**F**	**F**	**T**	**T**
		x_3	**F**	**T**	**F**	**T**
	x_0	x_1				
A_0	**F**	**F**	**F**	**T**	**F**	**F**
	F	**T**	**F**	**F**	**F**	**F**
A_1	**T**	**F**	**F**	**F**	**F**	**F**
	T	**T**	**F**	**F**	**F**	**F**

Fig. 2.19. Second Lupanov decomposition of f.

Theorem 2.8.43. *Let $f \in \mathtt{TF}_n$ and let $A_0, \ldots, A_{\ell-1}$ be an s-partition of $\mathbf{Bool}^k$, where $1 \leq k \leq n$ and $1 \leq s \leq 2^k$. For every function $f_{i,\omega}$ in the corresponding second Lupanov decomposition of f, we have $f_{i,\omega} = f^0_{i,\omega} \wedge f^1_{i,\omega}$, where*

$$f^0_{i,\omega}(x_0, \ldots, x_{n-1}) = \begin{cases} \omega(x_0, \ldots, x_{k-1}) & \text{if } (x_0, \ldots, x_{k-1}) \in A_i \\ \mathbf{F} & \text{otherwise} \end{cases}$$

and

$$f^1_{i,\omega}(x_0, \ldots, x_{n-1}) = \begin{cases} \mathbf{T} & \text{if } (x_k, \ldots, x_{n-1}) \in C_{A_i,\omega} \\ \mathbf{F} & \text{otherwise.} \end{cases}$$

Proof. Because $f_{i,\omega}$ is the trace of f_i on $D_{A_i,\omega}$, we have

$$f_{i,\omega}(x_0, \ldots, x_{n-1}) = \begin{cases} f_i(x_0, \ldots, x_{n-1}) & \text{if } (x_0, \ldots, x_{n-1}) \in D_{A_i,\omega} \\ \mathbf{F} & \text{otherwise.} \end{cases}$$

The definition of $D_{A_i,\omega}$ allows us to write

$$f_{i,\omega}(x_0, \ldots, x_{n-1}) = \begin{cases} f_i(x_0, \ldots, x_{n-1}) & \text{if } (x_k, \ldots, x_{n-1}) \in C_{A_i,\omega} \\ \mathbf{F} & \text{otherwise.} \end{cases}$$

Since f_i is the trace of f on B_i, taking into account the definition of B_i, we obtain

$$f_{i,\omega}(x_0, \ldots, x_{n-1}) = \begin{cases} f(x_0, \ldots, x_{n-1}) & \text{if } (x_k, \ldots, x_{n-1}) \in C_{A_i,\omega} \\ & \text{and } (x_0, \ldots, x_{k-1}) \in A_i \\ \mathbf{F} & \text{otherwise.} \end{cases}$$

The definition of $C_{A_i,\omega}$ allows us to further write

$$f_{i,\omega}(x_0, \ldots, x_{n-1}) = \begin{cases} \omega(x_0, \ldots, x_{k-1}) & \text{if } (x_k, \ldots, x_{n-1}) \in C_{A_i,\omega} \\ & \text{and } (x_0, \ldots, x_{k-1}) \in A_i \\ \mathbf{F} & \text{otherwise.} \end{cases}$$

Taking into account the definitions of $f^0_{i,\omega}$ and $f^1_{i,\omega}$, we have

$$\begin{aligned} f_{i,\omega}(x_0, \ldots, x_{n-1}) &= \begin{cases} f^0_{i,\omega}(x_0, \ldots, x_{n-1}) & \text{if } (x_k, \ldots, x_{n-1}) \in C_{A_i,\omega} \\ \mathbf{F} & \text{otherwise} \end{cases} \\ &= f^0_{i,\omega}(x_0, \ldots, x_{n-1}) \wedge f^1_{i,\omega}(x_0, \ldots, x_{n-1}). \end{aligned}$$

□

We stress that in the above theorem the function $f^0_{i,\omega}$ depends only on the first k arguments while $f^1_{i,\omega}$ depends only on the last $n-k$ arguments.

Theorem 2.8.43 implies the equality

$$f = \bigvee \{ f^0_{i,\omega} \wedge f^1_{i,\omega} \mid 0 \leq i \leq \ell - 1, \omega : A_i \longrightarrow \mathbf{Bool} \text{ and } C_{A_i,\omega} \neq \emptyset \}. \tag{2.8}$$

Definition 2.8.44. Let $f \in \texttt{TF}_n$ and let $A_0, \ldots, A_{\ell-1}$ be the standard s-partition of $\mathbf{Bool}^k$, where $1 \leq k \leq n$ and $1 \leq s \leq 2^k$. *Lupanov's* (k,s)*-representation of* f is the representation given by Equation (2.8). □

Example 2.8.45. For the functions $f_{0,\omega_1}, f_{1,\omega_7}$ of Example 2.8.42, the tables of $f^0_{0,\omega_1}, f^0_{1,\omega_7}$ and $f^1_{0,\omega_1}, f^1_{1,\omega_7}$ of Lupanov's $(2,2)$-representation of f are given in Figure 2.20. The full representation can be obtained by applying the same process to all functions $f_{i,\omega}$ of the second Lupanov decomposition.

□

We examine now the relationship that exists between truth tables and truth functions.

f^0_{0,ω_1}		x_2	**F**	**F**	**T**	**T**
		x_3	**F**	**T**	**F**	**T**
	x_0	x_1				
A_0	**F**	**F**	**F**	**F**	**F**	**F**
	F	**T**	**T**	**T**	**T**	**T**
A_1	**T**	**F**	**F**	**F**	**F**	**F**
	T	**T**	**F**	**F**	**F**	**F**

f^0_{1,ω_7}		x_2	**F**	**F**	**T**	**T**
		x_3	**F**	**T**	**F**	**T**
	x_0	x_1				
A_0	**F**	**F**	**F**	**F**	**F**	**F**
	F	**T**	**F**	**F**	**F**	**F**
A_1	**T**	**F**	**T**	**T**	**T**	**T**
	T	**T**	**T**	**T**	**T**	**T**

f^1_{0,ω_1}		x_2	**F**	**F**	**T**	**T**
		x_3	**F**	**T**	**F**	**T**
	x_0	x_1				
A_0	**F**	**F**	**T**	**F**	**T**	**F**
	F	**T**	**T**	**F**	**T**	**F**
A_1	**T**	**F**	**T**	**F**	**T**	**F**
	T	**T**	**T**	**F**	**T**	**F**

f^1_{1,ω_7}		x_2	**F**	**F**	**T**	**T**
		x_3	**F**	**T**	**F**	**T**
	x_0	x_1				
A_0	**F**	**F**	**F**	**T**	**T**	**T**
	F	**T**	**F**	**T**	**T**	**T**
A_1	**T**	**F**	**F**	**T**	**T**	**T**
	T	**T**	**F**	**T**	**T**	**T**

Fig. 2.20. Some functions of the Lupanov's $(2,2)$-representation.

Definition 2.8.46. Let $S = \{p_{i_0}, \ldots, p_{i_{n-1}}\}$ be a set of statement variables, where $i_0 < \cdots < i_{n-1}$ and $n \in \mathbf{N}$, and let $f \in \mathtt{TF}_n$. The *truth table defined by f and S* is the truth table τ_S^f given by

$$\tau_S^f(v) = f(v(p_{i_0}), \ldots, v(p_{i_{n-1}}))$$

for every $v \in \mathrm{TA}_S$. ∎

Let $S = \{p_{i_0}, \ldots, p_{i_{n-1}}\}$ be a set of statement variables, where $i_0 < \cdots < i_{n-1}$ and let $\vec{a} = (a_0, \ldots, a_{n-1}) \in \mathbf{Bool}^n$. Recall from Section 2.5 that $v_{\vec{a},S}$ is the partial truth assignment given by $v_{\vec{a},S}(p_{i_k}) = a_k$ for every k, $0 \leq k \leq n-1$.

Definition 2.8.47. Let $\tau : \mathrm{TA}_S \longrightarrow \mathbf{Bool}$ be a truth table over the set S of statement variables, where $S = \{p_{i_0}, \ldots, p_{i_{n-1}}\}$ and $i_0 < \cdots < i_{n-1}$.

The *truth function defined by τ* is the function $f_\tau : \mathbf{Bool}^n \longrightarrow \mathbf{Bool}$, where $f_\tau(\vec{a}) = \tau(v_{\vec{a},S})$ for every $\vec{a} \in \mathbf{Bool}^n$. ∎

Theorem 2.8.48. *Let S be a finite set of statement variables with $|S| = n$. Define $\Phi_S : \mathtt{TT}_S \longrightarrow \mathtt{TF}_n$ by $\Phi_S(\tau) = f_\tau$ for all $\tau \in \mathtt{TT}_S$ and define $\Psi_S : \mathtt{TF}_n \longrightarrow \mathtt{TT}_S$ by $\Psi_S(f) = \tau_S^f$ for all $f \in \mathtt{TF}_n$. Then Φ_S and Ψ_S are bijections which are inverses of each other.*

Proof. We must show that $\Psi_S \circ \Phi_S = 1_{\mathtt{TT}_S}$ and that $\Phi_S \circ \Psi_S = 1_{\mathtt{TF}_n}$, in other words, that $\tau_S^{f_\tau} = \tau$ for every $\tau \in \mathtt{TT}_S$ and that $f_{\tau_S^f} = f$ for every $f \in \mathtt{TF}_n$.

In order to show the first of these equalities, we define, for each $v \in \mathrm{TA}_S$, the sequence $\vec{a}_v = (v(p_{i_0}), \ldots, v(p_{i_{n-1}})) \in \mathbf{Bool}^n$. It is easy to see that $v = v_{\vec{a}_v,S}$. For each $\tau \in \mathtt{TT}_S$ and $v \in \mathrm{TA}_S$, we have $\tau(v) = \tau(v_{\vec{a}_v,S}) = f_\tau(\vec{a}_v) = \tau_S^{f_\tau}(v)$, which establishes the first equality.

To establish the second equality, let $f \in \mathtt{TF}_n$ be an arbitrary n-ary truth function. Then, for every $\vec{a} \in \mathbf{Bool}^n$, we have $f_{\tau_S^f}(\vec{a}) = \tau_S^f(v_{\vec{a},S}) = f(v_{\vec{a},S}(p_{i_0}), \ldots, v_{\vec{a},S}(p_{i_{n-1}})) = f(\vec{a})$, which finishes the proof. □

2.9 Clones and Functional Completeness

In this section, we characterize those sets of truth functions such that every truth function can be built starting from these functions by composition. Following standard practice in this field, we initially exclude 0-ary truth functions from our discussion. However, in the latter part of this section, we extend the usual definition of composition of functions in order to include 0-ary functions.

Definition 2.9.1. Let T be a set of truth functions, $T \subseteq \mathtt{TF}$. T is a *clone* if it contains all projections and is closed under composition. $\square$

Note that the set of truth functions $\mathtt{TF}$ is a clone and so is the set PROJ.

If T is a clone and f is an n-ary truth function that belongs to T, then any function that can be obtained from f by permuting or "fusing" of variables also belongs to T. In other words, let us consider a sequence $(i_0, \ldots, i_{m-1})$ of elements of the set $\{0, \ldots, n-1\}$. The truth function $g : \mathbf{Bool}^n \longrightarrow \mathbf{Bool}$ defined by

$$g = f(\pi^n_{i_0}, \ldots, \pi^n_{i_{m-1}})$$

is also a member of T. For instance, if $f : \mathbf{Bool}^4 \longrightarrow \mathbf{Bool}$ is a function from T, then $g(x_0, x_1) = f(x_1, x_0, x_0, x_0)$ for every $x_0, x_1 \in \mathbf{Bool}$ is also a function in T because $g = f(\pi^2_1, \pi^2_0, \pi^2_0, \pi^2_0)$. To make this concept more precise, consider the following definition.

Definition 2.9.2. Let $\wp : \{0, \ldots, k-1\} \longrightarrow \{0, \ldots, n-1\}$ and let $g \in \mathtt{TF}_k$. The *$\wp$-conjugate of g* is the truth function $g^{\wp} \in \mathtt{TF}_n$ defined by

$$g^{\wp}(x_0, \ldots, x_{n-1}) = g(x_{\wp(0)}, \ldots, x_{\wp(k-1)})$$

for $x_0, \ldots, x_{n-1} \in \mathbf{Bool}$.

If f is the $\wp$-conjugate of g for some $\wp$, then we refer to f as a *conjugate* of g. $\square$

It is easy to see that any conjugate of a member of a clone is also a member of the clone.

In the sequel, we consider five all-important examples of clones.

Example 2.9.3. Let $\mathcal{T}_{0,*} = \{f \in \mathtt{TF}_* | f(\mathbf{F}, \ldots, \mathbf{F}) = \mathbf{F}\}$ be the set of all truth functions that preserve $\mathbf{F}$ and let $\mathcal{T}_0 = \mathcal{T}_{0,*} \cap \mathtt{TF}$.

Since $f_\perp$ is in $\mathcal{T}_{0,*}$ and $f_\top$ is not, we have $\mathcal{T}_{0,*} = \mathcal{T}_0 \cup \{f_\perp\}$.

Note that for every $\pi_i^n \in$ PROJ we have $\pi_i^n(\mathbf{F}, \ldots, \mathbf{F}) = \mathbf{F}$, so PROJ $\subseteq \mathcal{T}_0$.

If $f \in \mathtt{TF}_m \cap \mathcal{T}_0$ and $g_i \in \mathtt{TF}_n \cap \mathcal{T}_0$ for $0 \leq i \leq m-1$, then

$$\begin{aligned} & f(g_0, \ldots, g_{m-1})(\mathbf{F}, \ldots, \mathbf{F}) \\ & \quad = f(g_0(\mathbf{F}, \ldots, \mathbf{F}), \ldots, g_{m-1}(\mathbf{F}, \ldots, \mathbf{F})) \\ & \quad = f(\mathbf{F}, \ldots, \mathbf{F}) = \mathbf{F}, \end{aligned}$$

which proves that $\mathcal{T}_0$ is closed under composition. Therefore, $\mathcal{T}_0$ is indeed a clone. □

Example 2.9.4. Let $\mathcal{T}_{1,*} = \{f \in \mathtt{TF}_* | f(\mathbf{T}, \ldots, \mathbf{T}) = \mathbf{T}\}$ be the set of all truth functions that preserve $\mathbf{T}$ and let $\mathcal{T}_1 = \mathcal{T}_{1,*} \cap \mathtt{TF}$.

Since $f_\top$ is in $\mathcal{T}_{1,*}$ and $f_\perp$ is not, we have $\mathcal{T}_{1,*} = \mathcal{T}_1 \cup \{f_\top\}$.

Using an argument similar to the one in Example 2.9.3, the reader can easily show that $\mathcal{T}_1$ is a clone. □

Example 2.9.5. The set of self-dual truth functions $\mathcal{SD}$ is a clone. We pointed out that every projection is self-dual. Therefore, we need to show only that if $f \in \mathcal{SD}_m, g_i \in \mathcal{SD}_n$ for $0 \leq i \leq m-1$, then $f(g_0, \ldots, g_{m-1}) \in \mathcal{SD}_n$. We have

$$\begin{aligned} & (f(g_0, \ldots, g_{m-1}))^d(a_0, \ldots, a_{n-1}) \\ & \quad = \overline{f(g_0, \ldots, g_{m-1})(\overline{a_0}, \ldots, \overline{a_{m-1}})} \\ & \quad = \overline{f(g_0(\overline{a_0}, \ldots, \overline{a_{n-1}}), \ldots, g_{m-1}(\overline{a_0}, \ldots, \overline{a_{n-1}}))} \\ & \quad = \overline{f(\overline{g_0(a_0, \ldots, a_{n-1})}, \ldots, \overline{g_{m-1}(a_0, \ldots, a_{n-1})})} \\ & \quad = \overline{\overline{f(g_0(a_0, \ldots, a_{n-1}), \ldots, g_{m-1}(a_0, \ldots, a_{n-1}))}} \\ & \quad = f(g_0, \ldots, g_{m-1}))(a_0, \ldots, a_{n-1}), \end{aligned}$$

for every $a_0, \ldots, a_{n-1} \in \mathbf{Bool}$. This proves that $f(g_0, \ldots, g_{m-1})$ is self-dual, so $\mathcal{SD}$ is a clone. □

Example 2.9.6. The set of monotonic truth functions $\mathcal{MON}$ is a clone. We leave it to the reader to prove that every projection is monotonic and that the composition of monotonic truth functions is monotonic. □

Example 2.9.7. The set $\mathcal{LIN}$ of linear truth functions $\mathcal{LIN}$ is a clone. Indeed, for π_i^n we have

$$\begin{aligned}\pi_i^n(a_0, \ldots, a_{n-1}) &= a_i \\ &= \mathbf{F} + (\mathbf{F} \wedge a_0) + \cdots (\mathbf{F} \wedge a_{i-1}) \\ &\quad +(\mathbf{T} \wedge a_i) + (\mathbf{F} \wedge a_{i+1}) + \cdots + (\mathbf{F} \wedge a_{n-1})\end{aligned}$$

for every $a_0, \ldots, a_{n-1} \in \mathbf{Bool}$, which shows that every projection is linear. Using elementary properties of the operations $\wedge$ and $+$, it is tedious but straightforward to verify that if $f \in \mathcal{LIN}_m$ and $g_0, \ldots, g_{m-1} \in \mathcal{LIN}_n$, then $f(g_0, \ldots, g_{m-1}) \in \mathcal{LIN}_n$, so $\mathcal{LIN}$ is indeed a clone. $\square$

Lemma 2.9.8. *The clones $\mathcal{T}_0$, $\mathcal{T}_1$, $\mathcal{SD}$, $\mathcal{LIN}$, $\mathcal{MON}$ are pairwise distinct and proper subsets of* `TF`.

Proof. Consider the functions f_0, f_1, and $f_\neg$. The following table shows the memberships of these functions in the above clones:

f	$\mathcal{T}_0$	$\mathcal{T}_1$	$\mathcal{SD}$	$\mathcal{LIN}$	$\mathcal{MON}$
f_0	yes	no	no	yes	yes
f_1	no	yes	no	yes	yes
$f_\neg$	no	no	yes	yes	no

We have seen in Example 2.8.17 that $f_\vee \notin \mathcal{LIN}$. Thus, the above classes are distinct and properly included in `TF`. $\square$

An immediate application of the fact that $\mathcal{LIN}$ is a clone is contained in the following example.

Example 2.9.9. We saw in Example 2.8.17 that $f_\neg \in \texttt{TF}_1$ is a linear function while $f_\vee \in \texttt{TF}_2$ is not. Consider the function $f_|$ defined by $f_|(a, b) = f_\neg(f_\wedge(a, b))$ for every $a, b \in \mathbf{Bool}$. This function is also known as *the Sheffer function*. We have $f_\vee = f_|(f_\neg, f_\neg)$ and this shows that $f_|$ is not linear since otherwise, $f_\vee$ would be linear and this is not the case. $\square$

It is easy to verify that the intersection of an arbitrary nonempty collection of clones is also a clone. Thus, the collection of all clones is a closure system.

Definition 2.9.10. Let F be a set of truth functions, $F \subseteq \mathtt{TF}$. The *clone generated by* F is the set $\widehat{F}$ given by the following inductive definition:

(1) every projection $\pi^n_j : \mathbf{Bool}^n \longrightarrow \mathbf{Bool}$ belongs to $\widehat{F}$ for $n \in \mathbf{N}$, $n \geq 1$ and $0 \leq i \leq n-1$,
(2) every function f from F belongs to $\widehat{F}$, and
(3) if $f \in \mathtt{TF}_m$ and $g_0, \ldots, g_{m-1} \in \mathtt{TF}_n$ such that $f, g_0, \ldots, g_{m-1} \in \widehat{F}$, then their composition $f(g_0, \ldots, g_{m-1})$ also belongs to $\widehat{F}$.

$\square$

Let $F \subseteq \mathtt{TF}$. By the definition of $\widehat{F}$, $\widehat{F}$ is the intersection of all clones that contain F. In other words, $\widehat{F}$ is the closure of the set F under the closure operator corresponding to the closure system of all clones. Note that $\widehat{\emptyset} = \mathrm{PROJ}$.

Definition 2.9.11. A set of truth functions $F \subseteq \mathtt{TF}$ is *complete* if $\widehat{F} = \mathtt{TF}$. $\square$

In order to discuss a characterization of complete sets of truth functions, we need several technical results.

If $f \in \mathtt{TF}_2$, then, for $n \geq 1$, denote by $f^{(n)}$ the n-ary truth functions defined inductively by

$$f^{(1)}(x_0) = x_0,$$
$$f^{(n+1)}(x_0, \ldots, x_n) = f(f^{(n)}(x_0, \ldots, x_{n-1}), x_n),$$

for every $x_0, \ldots, x_n \in \mathbf{Bool}$.

Lemma 2.9.12. *If T is a clone and $f \in T \cap \mathtt{TF}_2$, then $f^{(n)} \in T$ for $n \geq 1$.*

Proof. The argument is by induction on n, where $n \geq 1$. For $n = 1$, the statement is obviously true since $f^{(1)} = \pi^1_0$. Suppose that $f^{(n)} \in T$. The function $h \in \mathtt{TF}_n$ given by

$$h(x_0, \ldots, x_n) = f^{(n)}(\pi_0^{(n+1)}(x_0, \ldots, x_n), \ldots, \pi_{n-1}^{(n+1)}(x_0, \ldots, x_n))$$

for $x_0, \ldots, x_{n-1} \in \mathbf{Bool}$, clearly belongs to T. Since

$$f^{(n+1)}(x_0, \ldots, x_n) = f(h(x_0, \ldots, x_n), \pi_n^{(n+1)}(x_0, \ldots, x_n)),$$

we infer that $f^{(n+1)} \in T$. $\square$

Theorem 2.9.13. *The set of truth functions $F = \{f_\neg, f_\wedge\}$ is complete.*

Proof. Let us note that $f_\vee \in \widehat{F}$ since, according to the De Morgan laws, we have

$$f_\vee(x_0, x_1) = f_\neg(f_\wedge(f_\neg(x_0), f_\neg(x_1))),$$

for every $x_0, x_1 \in \mathbf{Bool}$. Therefore, by Lemma 2.9.12, $f_\vee^{(n)} \in \widehat{F}$ for $n \geq 1$.

Consider the function $g_b \in \mathtt{TF}_1$ given by $g_b(x_0) = x_0^b$ for $x_0, b \in \mathbf{Bool}$. We have $g_b \in \widehat{F}$ because $g_b = \pi_0^1$ if $b = \mathbf{T}$ and $g_b = f_\neg$ when $b = \mathbf{F}$.

By Lemma 2.9.12, the function $f_\wedge^{(n)}$ belongs to $\widehat{F}$, for all $n \geq 1$. We claim that every minterm function $f_{\vec{b}}$, where $\vec{b} = (b_0, \ldots, b_{n-1})$ and $n \geq 1$, also belongs to $\widehat{F}$. Indeed, the minterm function $f_{\vec{b}}$ can be written as

$$f_{\vec{b}}(x_0, \ldots, x_{n-1}) = f_\wedge^{(n)}(g_{b_0}(x_0), \ldots, g_{b_{n-1}}(x_{n-1})),$$

so $f_{\vec{b}} \in \widehat{F}$.

Let $f \in \mathtt{TF}_n$ and let $T(f) = \{\vec{b}_0, \ldots, \vec{b}_{k-1}\}$. Suppose initially that $k \geq 1$. Then, according to Theorem 2.8.27, we can write

$$f = f_\vee^{(k)}(f_{\vec{b}_0}, \ldots, f_{\vec{b}_{k-1}}),$$

which proves that $f \in \widehat{F}$. If $k = 0$, then f is the n-ary constant function whose value is $\mathbf{F}$ and we have $f = f_\wedge(\pi_0^n, f_\neg(\pi_0^n))$, so $f \in \widehat{F}$. $\square$

Lemma 2.9.14. *If a clone T contains a truth function $f \in \mathtt{TF}_n$, $n \geq 1$, that is not self-dual and the function $f_\neg$, then it also contains the one-argument constant truth functions f_0 and f_1.*

Proof. Let $f \in T \cap \mathtt{TF}_n$ be a truth function that is not self-dual. There is $(c_0, \ldots, c_{n-1}) \in \mathbf{Bool}^n$ such that

$$f(c_0, \ldots, c_{n-1}) = f(\overline{c_0}, \ldots, \overline{c_{n-1}}).$$

Consider the one-argument truth functions g_i given by $g_i(x) = x^{c_i}$ for $0 \leq i \leq n-1$. Note that we have either $g_i = \pi_0^1$ or $g_i = f_\neg$, so $g_i \in T$ for $0 \leq i \leq n-1$. Therefore, the one-argument function $h = f(g_0, \ldots, g_{n-1})$ is in the clone T. On the other hand,

$$h(\mathbf{T}) = f(\mathbf{T}^{\,c_0}, \ldots, \mathbf{T}^{\,c_{n-1}}) = f(c_0, \ldots, c_{n-1})$$

and

$$h(\mathbf{F}) = f(\mathbf{F}^{\,c_0}, \ldots, \mathbf{F}^{\,c_{n-1}}) = f(\overline{c_0}, \ldots, \overline{c_{n-1}}),$$

which shows that h is a constant function. If $h = f_0$, we have $f_1 = f_\neg(f_0)$, so both f_0 and f_1 belong to T. A similar argument works when $h = f_1$. □

Lemma 2.9.15. *If a clone T contains a nonmonotonic truth function $f \in \mathtt{TF}_n$, $n \geq 1$, and the constant functions f_0 and f_1, then it also contains $f_\neg$.*

Proof. Since f is nonmonotonic, by Lemma 2.8.21, there are

$$(a_0, \ldots, a_{n-1}), (b_0, \ldots, b_{n-1}) \in \mathbf{Bool}^n$$

such that $a_i \leq b_i$ for $0 \leq i \leq n-1$, $|\Delta(\vec{a}, \vec{b})| = 1$ and $f(a_0, \ldots, a_{n-1}) = \mathbf{T}$ and $f(b_0, \ldots, b_{n-1}) = \mathbf{F}$.

There exists $\ell \in \{0, \ldots, n-1\}$ such that $b_i = a_i$ for $i \neq \ell$, $a_\ell = \mathbf{F}$ and $b_\ell = \mathbf{T}$. If h_i is the one-argument constant function given by

$$h_i = \begin{cases} f_0 & \text{if } a_i = \mathbf{F}, \\ f_1 & \text{if } a_i = \mathbf{T}, \end{cases}$$

for $0 \leq i \leq n-1$ and $i \neq \ell$, it is clear that all functions h_i belong to T. Therefore, the function g defined by

$$g(x_0) = f(h_0(x_0), \ldots, h_{\ell-1}(x_0), \pi_0^1(x_0), h_{\ell+1}(x_0), \ldots, h_{n-1}(x_0))$$

for $x_0 \in \mathbf{Bool}$ is a member of T. Since $g(\mathbf{F}) = \mathbf{T}$ and $g(\mathbf{T}) = \mathbf{F}$, we have $g = f_\neg$, so $f_\neg \in T$. □

Lemma 2.9.16. *Let T be a clone that contains a nonlinear function $f \in \mathtt{TF}_n$, $n \geq 1$. If T contains all one-argument functions, then it also contains $f_\wedge$.*

Proof. Consider the polynomial normal form expansion of the function f:

$$f(\vec{x}) = \sum \{a_L \wedge \vec{x}_L \mid L \in P^n\},$$

Since f is nonlinear, $n > 1$, and there is $L \in P^n$ such that $|L| \geq 2$ and $a_L = \mathbf{T}$. To simplify notation, we assume that $\{x_0, x_1\} \subseteq L$. The argument would be similar if different variables were involved, but the notation would become messier. By applying the distributivity of $\wedge$ over $+$, we obtain the existence of the truth functions $g, h, k, l \in \mathtt{TF}_{n-2}$ such that

$$\begin{aligned} & f(x_0, x_1, \ldots, x_{n-1}) \\ & \quad = x_0 \wedge x_1 \wedge g(x_2, \ldots, x_{n-1}) + x_0 \wedge h(x_2, \ldots, x_{n-1}) \\ & \qquad + x_1 \wedge k(x_2, \ldots, x_{n-1}) + l(x_2, \ldots, x_{n-1}), \end{aligned}$$

for $x_0, \ldots, x_{n-1} \in \mathbf{Bool}$. The function g cannot be the constant $\mathbf{F}$ function since otherwise we could produce a different polynomial normal form for f where the coefficient of every term containing both x_0 and x_1 is $\mathbf{F}$, which would contradict the uniqueness of the polynomial normal form. Therefore, there are $a_2, \ldots, a_{n-1} \in \mathbf{Bool}$ such that $g(a_2, \ldots, a_{n-1}) = \mathbf{T}$. Consider the function $f' \in \mathtt{TF}_2$ defined by $f'(x_0, x_1) = f(x_0, x_1, a_2, \ldots, a_{n-1})$. Since T contains the projections and the constant functions, we have $f' \in T$ and we can write

$$f'(x_0, x_1) = x_0 \wedge x_1 + x_0 \wedge a + x_1 \wedge b + c,$$

where

$$\begin{aligned} a &= h(a_2, \ldots, a_{n-1}), \\ b &= k(a_2, \ldots, a_{n-1}), \\ c &= l(a_2, \ldots, a_{n-1}). \end{aligned}$$

Let $r_d \in \mathtt{TF}_1$ be the function given by $r_d = \pi_0^1$ if $d = \mathbf{F}$ and $r_d = f_\neg$ if $d = \mathbf{T}$. It is easy to see that $r_d(x_0) = x_0 + d$ for $d \in \mathbf{Bool}$. By hypothesis, it is clear that $r_d \in T$ for $d \in \mathbf{Bool}$. Therefore, the truth function $\ell \in \mathtt{TF}_2$ given by $\ell(x_0, x_1) = f'(r_b(x_0), r_a(x_1))$ belongs to T. We have

$$\begin{aligned} \ell(x_0, x_1) &= (x_0 + b) \wedge (x_1 + a) + (x_0 + b) \wedge a + (x_1 + a) \wedge b + c \\ &= x_0 \wedge x_1 + a \wedge b + c. \end{aligned}$$

If $e = a \wedge b + c$, then $x_0 \wedge x_1 = \ell(x_0, x_1) + e$, or $f_\wedge(x_0, x_1) = r_e(\ell(x_0, x_1))$, so $f_\wedge \in T$. $\square$

Theorem 2.9.17 (Post[7]Completeness Theorem). *A set F of truth functions, $F \subseteq \mathtt{TF}$, is complete if and only if it is not included in any of the clones $\mathcal{T}_0$, $\mathcal{T}_1$, $\mathcal{SD}$, $\mathcal{LIN}$, or $\mathcal{MON}$.*

Proof. Suppose that F is not included in any of the clones mentioned above. In this case, the clone generated by F, $\widehat{F}$ is also not included in any of these clones. This means that $\widehat{F}$ contains

- a function f_{nf} that does not preserve $\mathbf{F}$,
- a function f_{nt} that does not preserve $\mathbf{T}$,
- a function f_{nsd} that is not self-dual,
- a function f_{nl} that is not linear,
- a function f_{nm} that is not monotonic.

We claim initially that $\widehat{F}$ contains the constant functions f_0, f_1 from $\mathtt{TF}_1$. Define the function $f \in \mathtt{TF}_1$ by $f = f_{nf}(\pi_0^1, \ldots, \pi_0^1)$. Then, $f \in \widehat{F}$. Let $b = f_{nf}(\mathbf{T}, \ldots, \mathbf{T})$. If $b = \mathbf{T}$, then, since $f_{nf}(\mathbf{F}, \ldots, \mathbf{F}) = \mathbf{T}$, the function f is the constant function f_1. If $b = \mathbf{F}$, we have $f(\mathbf{F}) = \mathbf{T}$ and $f(\mathbf{T}) = \mathbf{F}$, so $f = f_\neg$.

A similar argument applied to the function f_{nt} shows that $\widehat{F}$ contains one of the functions f_0 or $f_\neg$. Note that the following cases may occur:

(1) $f_\neg \in \widehat{F}$,
(2) $f_\neg \notin \widehat{F}$ and, therefore, $f_0, f_1 \in \widehat{F}$, which justifies our claim.

In the first case, we obtain the existence of the constant functions in $\widehat{F}$ from Lemma 2.9.14. Therefore, the initial claim is justified.

By Lemma 2.9.15, $\widehat{F}$ also contains $f_\neg$. Therefore, $\widehat{F}$ contains all four one-argument truth functions and, by Lemma 2.9.16, it also contains $f_\wedge$. Since $\{f_\neg, f_\wedge\} \subseteq \widehat{F}$, by Theorem 2.9.13, we have $\widehat{F} = \mathtt{TF}$, hence F is a complete set.

In order to prove the necessity of the condition, suppose that F would be included in any of the clones $\mathcal{T}_0$, $\mathcal{T}_1$, $\mathcal{SD}$, $\mathcal{LIN}$, or $\mathcal{MON}$.

[7]Emil Leon Post was born in Augustòw, Poland, on February 11, 1897, and died in New York on April 21, 1954. He attended the College of the City of New York and received his Ph.D. from Columbia University in 1920. He was affiliated with Princeton, Columbia, Cornell, and eventually with the City College of New York, where he taught from 1932 on. His main contributions were in proof theory, algebra, and computability.

This would imply that $\widehat{F}$ is included in one of these clones and by Lemma 2.9.8, this would prevent F from being complete. $\square$

Example 2.9.18. Let $f_+ \in \mathtt{TF}_2$ be defined by $f_+(a_0, a_1) = a_0 + a_1$ for every $(a_0, a_1) \in \mathbf{Bool}^2$. Based on Theorem 2.8.31, we could make an argument similar to the proof of Theorem 2.9.13 to show that the set of truth functions $\{f_0, f_1, f_+, f_\wedge\}$ is complete. Here we give an argument based on Theorem 2.9.17 to show the slightly stronger result that $\{f_1, f_+, f_\wedge\}$ is complete.

The function f_1 does not preserve $\mathbf{F}$ and the function f_+ does not preserve $\mathbf{T}$. The function f_1 is not self-dual, $f_\wedge$ is not linear, by Example 2.8.33, and f_+ is not monotonic. Therefore, $\{f_1, f_+, f_\wedge\}$ is complete. ▯

Example 2.9.19. Every function $f \in \mathtt{TF} - (\mathcal{T}_0 \cup \mathcal{T}_1 \cup \mathcal{SD} \cup \mathcal{LIN} \cup \mathcal{MON})$ defines a one-element complete set $\{f\}$. Consider, for instance, the truth function $f_|$ defined by $f_|(a, b) = f_\neg(f_\wedge(a, b))$ for every $a, b \in \mathbf{Bool}$. Note that

(1) $f_| \notin \mathcal{T}_0$ because $f_|(\mathbf{F}, \mathbf{F}) = \mathbf{T}$,
(2) $f_| \notin \mathcal{T}_1$ because $f_|(\mathbf{T}, \mathbf{T}) = \mathbf{F}$,
(3) $f_| \notin \mathcal{SD}$ because $f_|^d(a_0, a_1) = \overline{a_0} \wedge \overline{a_1}$,
(4) $f_| \notin \mathcal{LIN}$ as we have shown in Example 2.9.9,
(5) $f_| \notin \mathcal{MON}$ because $(\mathbf{F}, \mathbf{F}) \leq (\mathbf{T}, \mathbf{T})$, while $f_|(\mathbf{F}, \mathbf{F}) = \mathbf{T}$ and $f_|(\mathbf{T}, \mathbf{T}) = \mathbf{F}$.

This shows that $\{f_|\}$ is a complete set.

Using a similar argument, it is possible to show that the function $f_\downarrow$ defined by $f_\downarrow(a_0, a_1) = f_\neg(f_\vee(a_0, a_1))$ for every $a_0, a_1 \in \mathbf{Bool}$ also defines a one-element complete set $\{f_\downarrow\}$. ▯

Example 2.9.20. Consider the set of truth functions $\{f_{P_4}, f_{\min}\}$, where f_{P_4} is the four-argument parity function and $f_{\min}$ is the minority function both introduced in Example 2.8.17. It is easy to verify that f_{P_4} does not belong to $\mathcal{T}_0, \mathcal{SD}$, or $\mathcal{MON}$. It is clear that $f_{\min}$ does not belong to $\mathcal{T}_1$ and we saw in Example 2.8.17 that $f_{\min}$ is not linear. Thus, by Theorem 2.9.17, the set $\{f_{P_4}, f_{\min}\}$ is complete. ▯

The exclusion of the 0-ary truth functions in the discussion of clones is a common but unnecessary practice in the literature. We now present an extension of the idea of functional completeness that incorporates 0-ary truth functions. We will need this extension in Section 2.10.

We begin by extending the idea of composition.

Definition 2.9.21. Let f be a 0-ary truth function and let g be an n-ary truth function with $n > 0$. We say that g is obtained from f by *degenerate composition* if $g(x_0, \ldots, x_{n-1}) = f()$ for every $(x_0, \ldots, x_{n-1}) \in \mathbf{Bool}^n$.

A function is obtained by *extended composition* from a set of functions if it is obtained from these functions by composition in the sense of Definition 2.8.3 or by degenerate composition. ∎

Definition 2.9.22. Let T be a set of truth functions, $T \subseteq \mathtt{TF}_*$. T is an *extended clone* if it contains all projections and is closed under extended composition. ∎

It is obvious that $\mathtt{TF}_*$, $\mathtt{TF}$, and PROJ are extended clones.

Theorem 2.9.23. *Every clone is an extended clone. If C is an extended clone, then $C \cap \mathtt{TF}$ is a clone.*

Proof. The first part of the theorem is immediate since every clone is vacuously closed under degenerate composition because it does not contain any 0-ary functions.

For the second part, note that PROJ $\subseteq C \cap \mathtt{TF}$ because PROJ $\subseteq C$. In addition, the composition of non-0-ary functions from C results in a non-0-ary function in C, which shows that $C \cap \mathtt{TF}$ is closed under composition. □

Note that if an extended clone contains $f_\perp$, then it contains for every $n \in \mathbf{N}$ the n-ary truth function g such that $g(x_0, \ldots, x_{n-1}) = \mathbf{F}$ for every $(x_0, \ldots, x_{n-1}) \in \mathbf{Bool}^n$. Similarly, if an extended clone contains $f_\top$, then for every $n \in \mathbf{N}$, it contains the function h such that $h(x_0, \ldots, x_{n-1}) = \mathbf{T}$ for every $(x_0, \ldots, x_{n-1}) \in \mathbf{Bool}^n$.

In Section 2.8, we introduced the following families of functions:

$$\begin{aligned}
\mathcal{T}_{0,*} &= \mathcal{T}_0 \cup \{f_\perp\},\\
\mathcal{T}_{1,*} &= \mathcal{T}_1 \cup \{f_\top\},\\
\mathcal{SD}_* &= \mathcal{SD},\\
\mathcal{MON}_* &= \mathcal{MON} \cup \{f_\top, f_\perp\},\\
\mathcal{LIN}_* &= \mathcal{LIN} \cup \{f_\top, f_\perp\}.
\end{aligned}$$

If $\mathcal{C}$ is one of the clones $\mathcal{T}_0, \mathcal{T}_1, \mathcal{SD}, \mathcal{MON}$, or $\mathcal{LIN}$, then we denote by $\mathcal{C}_*$ the corresponding "starred" set given above.

Theorem 2.9.24. *The sets $\mathcal{T}_{0,*}, \mathcal{T}_{1,*}, \mathcal{SD}_*, \mathcal{MON}_*$, and $\mathcal{LIN}_*$ are all extended clones.*

Proof. We give the argument for $\mathcal{T}_{0,*}$ and leave the others to the reader. Since $\mathcal{T}_0$ was shown to be a clone in Example 2.9.3, we need only show that $f(f_\perp, \ldots, f_\perp) \in \mathcal{T}_{0,*}$ when $f \in \mathcal{T}_0$ in order to show that $\mathcal{T}_{0,*}$ is closed under composition. This is clearly the case because $f(f_\perp, \ldots, f_\perp) = f_\perp$. In addition, $\mathcal{T}_{0,*}$ is closed under degenerate composition because if g is obtained from $f_\perp$ by degenerate composition, then $g(\mathbf{F}, \ldots, \mathbf{F}) = f_\perp() = \mathbf{F}$. □

Definition 2.9.25. Let $F \subseteq \mathtt{TF}_*$. The *extended clone generated by* F is the set $\widehat{\widehat{F}}$ given by the following inductive definition:

(1) every projection $\pi_j^n : \mathbf{Bool}^n \longrightarrow \mathbf{Bool}$ belongs to $\widehat{\widehat{F}}$ for $n \geq 1$ and $0 \leq i \leq n-1$,
(2) every function f from F belongs to $\widehat{\widehat{F}}$,
(3) for every 0-ary function h from F and $n \in \mathbf{N}$, the n-ary function g obtained from h by degenerate composition is a member of $\widehat{\widehat{F}}$, and
(4) if $f \in \mathtt{TF}_m$ and $g_0, \ldots, g_{m-1} \in \mathtt{TF}_n$ are such that $f, g_0, \ldots, g_{m-1} \in \widehat{\widehat{F}}$, for $m \geq 1$, then their composition $f(g_0, \ldots, g_{m-1})$ also belongs to $\widehat{\widehat{F}}$.

▯

Observe that $\widehat{\widehat{F}}$ is the intersection of all extended clones containing F.

The next result is the analog of Lemma 2.9.12 for extended clones.

Lemma 2.9.26. *If T is an extended clone and $f \in T \cap \mathtt{TF}_2$, then $f^{(n)} \in T$ for $n \geq 1$.*

Proof. Since $f \in T \cap \mathtt{TF}$, it follows by Lemma 2.9.12 that $f^{(n)} \in T \cap \mathtt{TF} \subseteq T$ because $T \cap \mathtt{TF}$ is clone by Theorem 2.9.23. □

Theorem 2.9.27. *Let $F = \{f_\wedge, f_\vee, f_\top, f_\perp\}$. The extended clone generated by F, $\widehat{\widehat{F}}$, equals $\mathcal{MON}_*$.*

Proof. Since $f_\wedge, f_\vee, f_\top, f_\bot \in \mathcal{MON}_*$, it is clear that $\widehat{\widehat{F}} \subseteq \mathcal{MON}_*$.

Conversely, let $f \in \mathtt{TF}_n$ be monotonic. Note that by degenerate composition, both $c^n_{\mathbf{T}}$ and $c^n_{\mathbf{F}}$ belong to $\widehat{\widehat{F}}$, where the functions $c^m_{\mathbf{F}}, c^m_{\mathbf{T}}$ were introduced in Equalities (2.3). Lemma 2.9.26 implies that $f^{(m)}_\vee$ and $f^{(m)}_\wedge$ belong to $\widehat{\widehat{F}}$ for all $m \geq 1$. This allows us to conclude that every positive n-ary conjunction $\ell_0 \wedge \cdots \wedge \ell_{p-1}$ belongs to $\widehat{\widehat{F}}$. (For the special case $p = 0$, this follows because the n-ary conjunction reduces to $c^n_{\mathbf{T}}$.) By Theorem 2.8.30, f can be written as $\bigvee_{0 \leq i \leq k-1} g_i$, where each g_i is a positive n-ary conjunction. If $k = 0$, $f \in \widehat{\widehat{F}}$ because $f = c^n_{\mathbf{F}}$. If $k > 0$, the same conclusion can be reached from the facts established above. $\Box$

Theorem 2.9.28. *If $F \subseteq \mathtt{TF}$, then $\widehat{\widehat{F}} = \widehat{F}$.*

Proof. Since every clone is an extended clone, $\widehat{\widehat{F}} \subseteq \widehat{F}$. To prove the converse inclusion, observe that, by Theorem 2.9.23, $\widehat{\widehat{F}} \cap \mathtt{TF}$ is a clone and F is a subset of this clone because $F \subseteq \mathtt{TF}$. Therefore, $\widehat{\widehat{F}} \supseteq \widehat{\widehat{F}} \cap \mathtt{TF} \supseteq \widehat{F}$. $\Box$

We can now extend the notion of complete set of non-0-ary truth functions to arbitrary sets of truth functions.

Definition 2.9.29. A set $F \subseteq \mathtt{TF}_*$ is *complete* if $\widehat{\widehat{F}} \supseteq \mathtt{TF}$. ▯

A seemingly more natural notion of completeness could be defined by requiring that $\widehat{\widehat{F}} = \mathtt{TF}_*$. The current definition has the advantage of being conservative, as shown by the following theorem. (The relationship between these two possible definitions of completeness is discussed in Supplement 143.)

Theorem 2.9.30. *Let $F \subseteq \mathtt{TF}$. Then F is complete in the sense of Definition 2.9.11 if and only if it is complete in the sense of Definition 2.9.29.*

Proof. This theorem follows immediately from Theorem 2.9.28 and the fact that we always have $\widehat{F} \subseteq \mathtt{TF}$. $\Box$

Lemma 2.9.31. *Let $\mathcal{C}$ be one of the clones $\mathcal{T}_0, \mathcal{T}_1, \mathcal{SD}, \mathcal{MON}$, or $\mathcal{LIN}$, let f be a 0-ary truth function and let g be a non-*0*-ary truth* *function obtained from f by degenerate composition. Then, we have $f \in \mathcal{C}_*$ if and only if $g \in \mathcal{C}$.*

Proof. Suppose first that $\mathcal{C} = \mathcal{T}_0$. If $f = f_\perp$, then $f \in \mathcal{C}_*$ and g is a constant function with range $\{\mathbf{F}\}$, so $g \in \mathcal{C}$. If $f = f_\top$, then $f \notin \mathcal{C}_*$ and g is a constant function with range $\{\mathbf{T}\}$, so $g \notin \mathcal{C}$.

Now suppose that $\mathcal{C} = \mathcal{SD}$. Observe that no 0-ary function is in $\mathcal{SD}_*$ and no constant function is in $\mathcal{SD}$, which gives immediately the result.

The remaining cases are left to the reader. □

We can now extend the Post Completeness Theorem to arbitrary sets of truth functions.

Theorem 2.9.32. *A set F of truth functions is complete if and only if it is not included in any of the extended clones $\mathcal{T}_{0,*}$, $\mathcal{T}_{1,*}$, $\mathcal{SD}_*$, $\mathcal{LIN}_*$, or $\mathcal{MON}_*$.*

Proof. If F is contained in one of the given extended clones, then $\widehat{\widehat{F}}$ is also contained in the same clone, so F is not complete.

Suppose now that F is not contained in any of the given extended clones. Then F must contain

- a function f_{nf} not in $\mathcal{T}_{0,*}$,
- a function f_{nt} not in $\mathcal{T}_{1,*}$,
- a function f_{nsd} not in $\mathcal{SD}_*$,
- a function f_{nl} not in $\mathcal{LIN}_*$,
- a function f_{nm} not in $\mathcal{MON}_*$.

Let f'_{nf} be the truth function defined by

$$f'_{nf} = \begin{cases} f_{nf} \text{ if } f_{nf} \text{ is not 0-ary,} \\ \text{the unary function obtained from } f_{nf} \text{ by} \\ \qquad \text{degenerate composition, otherwise.} \end{cases}$$

Then, $f'_{nf} \in \widehat{\widehat{F}}$ and, by Lemma 2.9.31, $f'_{nf} \notin \mathcal{T}_0$. Let $f'_{nt}, f'_{nsd}, f'_{nl}$, and f'_{nm} be defined similarly. By the same argument as the one used for f'_{nf}, we have $f'_{nt}, f'_{nsd}, f'_{nl}, f'_{nm} \in \widehat{\widehat{F}}$ and $f'_{nt} \notin \mathcal{T}_1$, $f'_{nsd} \notin \mathcal{SD}$, $f'_{nl} \notin \mathcal{LIN}$ and $f'_{nm} \notin \mathcal{MON}$. Thus, by the Post Completeness Theorem, $\widehat{\widehat{F}} \cap \mathtt{TF}$ is complete, i.e., the clone generated by $\widehat{\widehat{F}} \cap \mathtt{TF}$

equals TF. Since, by Theorem 2.9.23, $\widehat{\widehat{F}} \cap \mathtt{TF}$ is itself a clone, we have $\widehat{\widehat{F}} \cap \mathtt{TF} = \mathtt{TF}$, so $\widehat{\widehat{F}} \supseteq \mathtt{TF}$, which shows that F is complete. □

2.10 Complete Sets of Connectives

The notion of connective, as a tool for building complex statements from simpler ones, has a linguistic nature. In previous sections, in our mathematical modeling of this notion, we have used connective symbols equipped with an appropriate semantics, which was expressed by truth functions attached to the connective symbol.

Intuitively, a complete set of connectives is one such that every statement can be built up using only connectives from the set. When we talk about complete sets of connectives, we want to consider arbitrary connectives, not necessarily the ones that we used initially (negation, conjunction, disjunction, implication, and biconditional). In formalizing this notion, we need to consider arbitrary collections of connective symbols and their semantics given by truth functions, not necessarily unary or binary. Because these connective symbols can have arbitrary arities, we must introduce an alternative way of building formulas of propositional logic.

Let $F = \{f_C \mid C \in \mathbf{C}\}$ be a collection of truth functions indexed by a set of symbols $\mathbf{C}$. We shall refer to the elements of $\mathbf{C}$ as *connective symbols of* F or just *connective symbols* when F is understood from the context.

Definition 2.10.1. Let $F = \{f_C \mid C \in \mathbf{C}\}$ be a collection of truth functions indexed by $\mathbf{C}$. The *F-signature of propositional logic* is $S_F = (\mathbf{C}, \nu)$ where for each $C \in \mathbf{C}$, $\nu(C)$ is the arity of f_C.

The set of *F-formulas of propositional logic* is

$$\mathrm{PLFORM}_F = \mathrm{TERM}_{S_F}(SV).$$

▯

If φ is a formula from PLFORM_F, then again we will denote the set of variables that occur in φ by $SV(\varphi)$.

Example 2.10.2. Consider the set $F = \{f_\neg, f_\wedge, f_\vee, f_\rightarrow, f_\leftrightarrow\}$ indexed by the standard collection of connective symbols $\mathbf{C} = \{\neg, \wedge, \vee, \rightarrow, \leftrightarrow\}$. The set PLFORM_F differs from PLFORM because in the former set, connective symbols are treated as function symbols and formulas are terms. For instance, the formula

$$(((p_0 \to p_1) \wedge (p_1 \to p_2)) \to (p_0 \to p_2))$$

from PLFORM corresponds to the F-formula

$$\to (\wedge(\to (p_0, p_1), \to (p_1, p_2)), \to (p_0, p_2)).$$

Recall that we have the unique readability property for PLFORM by Theorem 2.2.11 and the same property for PLFORM_F by Theorem 1.5.12. This justifies the definitions of Φ and Ψ given next.

Although the sets PLFORM and PLFORM_F are not the same, there are natural bijections $\Phi : \text{PLFORM} \longrightarrow \text{PLFORM}_F$ and $\Psi : \text{PLFORM}_F \longrightarrow \text{PLFORM}$ which allow us to go back and forth between these sets while preserving the semantics of the formulas as defined in the following (see Exercise 149). These mappings are recursively defined by

$$\begin{aligned} \Phi(p) &= p, \\ \Phi((\neg\alpha)) &= \neg(\Phi(\alpha)), \\ \Phi((\alpha C \beta)) &= C(\Phi(\alpha), \Phi(\beta)), \end{aligned}$$

for all $\alpha, \beta \in \text{PLFORM}$, and

$$\begin{aligned} \Psi(p) &= p, \\ \Psi(\neg(\alpha)) &= (\neg\Psi(\alpha)), \\ \Psi(C(\alpha, \beta)) &= (\Psi(\alpha) C \Psi(\beta)), \end{aligned}$$

for all $\alpha, \beta \in \text{PLFORM}_F$, where C is a binary connective symbol in $\mathbf{C}$. We leave it to the reader (see Exercise 148) to verify that Φ and Ψ are inverse bijections. ▯

Example 2.10.3. Let $F = \{f_{P_4}, f_{\min}\}$ be the set truth functions, considered in Example 2.9.20, indexed by the set of connective symbols $\mathbf{C} = \{\min, P_4\}$. The following string of symbols

$$P_4(\min(p_0, p_1, p_2), p_3, p_4, \min(p_0, p_1, p_3))$$

is an F-formula. ▯

Definition 2.10.4. Let $F = \{f_C \mid C \in \mathbf{C}\}$ be a collection of truth functions indexed by $\mathbf{C}$. An *F-substitution* is a substitution defined on SV with values in PLFORM_F. $\square$

If $\varphi \in \text{PLFORM}_F$ and s is an F-substitution, then, according to Theorem 1.5.20, $s(\varphi) \in \text{PLFORM}_F$.

Definition 2.10.5. Let $F = \{f_C \mid C \in \mathbf{C}\}$ be a collection of truth functions indexed by $\mathbf{C}$ and let v be a truth assignment. The *truth valuation on* PLFORM_F *generated by* v is the function $\bar{v} : \text{PLFORM}_F \longrightarrow \mathbf{Bool}$ given by the following recursive definition:

$$\bar{v}(q) = v(q),$$
$$\bar{v}(C_0) = f_{C_0}(),$$
$$\bar{v}(C(\varphi_0, \ldots, \varphi_{n-1})) = f_C(\bar{v}(\varphi_0), \ldots, \bar{v}(\varphi_{n-1}))$$

for each statement variable q, 0-ary connective symbol C_0, n-ary connective symbol C with $n \geq 1$, and F-formulas $\varphi_0, \ldots, \varphi_{n-1} \in \text{PLFORM}_F$. $\square$

As before, we will often write $v(\varphi)$ instead of $\bar{v}(\varphi)$.

Theorem 2.10.6. *Let $F = \{f_C \mid C \in \mathbf{C}\}$ be a set of truth functions indexed by $\mathbf{C}$ and let $\varphi \in \text{PLFORM}_F$. If v, w are two truth assignments such that $v(p) = w(p)$ for every $p \in SV(\varphi)$, then $\bar{v}(\varphi) = \bar{w}(\varphi)$.*

Proof. The argument is by induction on φ and is left to the reader. $\square$

Let S be a set of statement variables, $F = \{f_C | C \in \mathbf{C}\}$ be a collection of truth functions, and let v be a partial truth assignment over S. For every formula $\varphi \in \text{PLFORM}_F$ with $SV(\varphi) \subseteq S$, we can define $v(\varphi)$ as we did in Section 2.3, by applying the Agreement Theorem. Again, if v is a partial truth assignment such that $SV(\varphi) \subseteq \text{Dom}(v)$, then for every connective symbol C we have

$$v(C(\varphi_0, \ldots, \varphi_{n-1})) = f_C(v(\varphi_0), \ldots, v(\varphi_{n-1})),$$

for $\varphi_0, \ldots, \varphi_{n-1} \in \text{PLFORM}_F$.

Corollary 2.10.7. *Let $F = \{f_C \mid C \in \mathbf{C}\}$ be a set of truth functions indexed by $\mathbf{C}$ and let $\varphi \in \text{PLFORM}_F$. If v, w are two partial truth assignments whose domains contain $SV(\varphi)$ such that $v(p) = w(p)$ for every $p \in SV(\varphi)$, then $\bar{v}(\varphi) = \bar{w}(\varphi)$.*

Proof. Let v', w' be truth assignments which extend v, w, respectively. Then, since $v'(p) = w'(p)$ for all $p \in SV(\varphi)$, we have $v'(\varphi) = w'(\varphi)$ because of Theorem 2.10.6. Since $v(\varphi) = v'(\varphi)$ and $w(\varphi) = w'(\varphi)$, the statement follows immediately. $\square$

Theorem 2.10.8. *Let s be an F-substitution and v be a truth assignment. For every formula $\varphi \in \mathrm{PLFORM}_F$, we have $v(s(\varphi)) = v'(\varphi)$, where v' is the truth assignment given by $v'(p) = v(s(p))$ for every $p \in SV$.*

Proof. The proof is by induction on formulas and is similar to the proof of Theorem 2.6.6. $\square$

Corollary 2.10.9. *Let $F = \{f_C | C \in \mathbf{C}\}$ be a collection of truth functions indexed by $\mathbf{C}$. If s is an F-substitution and v is a partial truth assignment over a set S, then, for every formula φ such that $SV(s(\varphi)) \subseteq S$, we have $v(s(\varphi)) = v'(\varphi)$ for every partial truth assignment v' such that $\mathrm{Dom}(v') \supseteq SV(\varphi)$ and $v'(p) = v(s(p))$ for every $p \in SV(\varphi)$.*

Proof. The statement is a consequence of Theorem 2.10.8; the argument is similar to that of Corollary 2.6.7. $\square$

Definition 2.3.27 can be reformulated in this new setting as follows.

Definition 2.10.10. Let $F = \{f_C | C \in \mathbf{C}\}$ be a collection of truth functions, φ be a formula in PLFORM_F, and let S be a set of statement variables such that $SV(\varphi) \subseteq S$. Then, $\tau_{\varphi,S}$, the *truth table of φ over S*, is defined by $\tau_{\varphi,S}(v) = v(\varphi)$.

The *truth table* τ_φ of φ is $\tau_{\varphi,SV(\varphi)}$. □

Now, we introduce the notion of completeness of a set of connectives. This will turn out to be closely related to the notion of completeness of a set of truth functions introduced in Section 2.8.

Definition 2.10.11. Let $F = \{f_C \mid C \in \mathbf{C}\}$ be a set of truth functions indexed by $\mathbf{C}$. The set $\mathbf{C}$ is *complete* if for every finite, nonempty set of statement variables S and $\tau \in \mathtt{TT}_S$, there is a formula $\varphi \in \mathrm{PLFORM}_F$ with $SV(\varphi) \subseteq S$ such that $\tau = \tau_{\varphi,S}$. □

We use the terminology "complete set $\mathbf{C}$" in order to follow standard but abusive practice. It would be more correct to refer to completeness of F.

To relate the notion of completeness just introduced to the notion of completeness from Section 2.9, we need the following preliminary results.

Lemma 2.10.12. *Let $F = \{f_C \mid C \in \mathbf{C}\}$ be a set of truth functions indexed by $\mathbf{C}$ and let $\mathcal{C}$ be the extended clone generated by the set $\{f_C \mid C \in \mathbf{C}\}$. For every n-ary truth function $f \in \mathcal{C}$ and set of variables S such that $|S| = n$, there exists a formula $\varphi \in \mathrm{PLFORM}_F$ such that $SV(\varphi) \subseteq S$ and $\tau_S^f = \tau_{\varphi,S}$.*

Proof. The proof is by induction on the definition of C as given in Definition 2.9.25.

First, let f be a projection, say $f = \pi_j^n$, and let $S = \{p_{i_0}, \ldots, p_{i_{n-1}}\}$. Then, if we take $\varphi = p_{i_j}$, we obtain the desired conclusion.

Now suppose that $f = f_C$ for some $C \in \mathbf{C}$, where f is n-ary. Let $S = \{p_{i_0}, \ldots, p_{i_{n-1}}\}$. If $n > 0$, take $\varphi = C(p_{i_0}, \ldots, p_{i_{n-1}})$. We have, for every $v \in \mathrm{TA}_S$,

$$\begin{aligned}\tau_S^f(v) &= f(v(p_{i_0}), \ldots, v(p_{i_{n-1}}))\\ &= f_C(v(p_{i_0}), \ldots, v(p_{i_{n-1}}))\\ &= v(C(p_{i_0}, \ldots, p_{i_{n-1}}))\\ &= v(\varphi)\\ &= \tau_{\varphi,S}(v).\end{aligned}$$

If $n = 0$, we can take $\varphi = C$.

Let g be a 0-ary function for which the statement holds and let f be an n-ary function obtained from g by degenerate composition. By the inductive hypothesis, there is a formula ψ with $SV(\psi) = \emptyset$ such that $\tau_\emptyset^g = \tau_{\psi,\emptyset}$, that is, $g() = v_0(\psi)$, where v_0 is the unique partial truth assignment defined on $\emptyset$. Let $S = \{p_{i_0}, \ldots, p_{i_{n-1}}\}$. We claim that $\tau_S^f = \tau_{\psi,S}$. Indeed, let $v \in \mathrm{TA}_S$. Observe that $v(\psi) = v_0(\psi)$ since v and v_0 coincide on $SV(\psi) = \emptyset$. Therefore,

$$\begin{aligned}\tau_S^f(v) &= f(v(p_{i_0}), \ldots, v(p_{i_{n-1}}))\\ &= g()\\ &= v_0(\psi)\\ &= v(\psi)\\ &= \tau_{\psi,S}(v).\end{aligned}$$

Suppose that the hypothesis is true for $g, h_0, \ldots, h_{m-1}$ where g is m-ary, $m \geq 1$, and $h_0, \ldots, h_{m-1}$ are n-ary functions and that the function f is given by $f = g(h_0, \ldots, h_{m-1})$. Let $S = \{p_{i_0}, \ldots, p_{i_{n-1}}\}$. By the inductive hypothesis, if $T = \{p_0, \ldots, p_{m-1}\}$, there is a formula ψ with $SV(\psi) \subseteq T$ such that $\tau^g_T = \tau_{\psi,T}$. Therefore, we have

$$w(\psi) = g(w(p_0), \ldots, w(p_{m-1}))$$

for every $w \in \mathrm{TA}_T$. For the same reason, there are formulas $\psi_0, \ldots, \psi_{m-1}$ with $SV(\psi_j) \subseteq S$ and $\tau^{h_j}_S = \tau_{\psi_j,S}$ for all j, $0 \leq j \leq m-1$. This amounts to

$$v(\psi_j) = h_j(v(p_{i_0}), \ldots, v(p_{i_{n-1}}))$$

for every $v \in \mathrm{TA}_S$ and $0 \leq j \leq m-1$.

Let s be the finite substitution $s^{p_0 \;\cdots\; p_{m-1}}_{\psi_0 \cdots \psi_{m-1}}$ and $\varphi = s(\psi)$. By Theorem 2.6.5, we have $SV(\varphi) \subseteq S$. We claim that $\tau^f_S = \tau_{\varphi,S}$. Indeed, let v be an arbitrary truth assignment from TA_S and let $w \in \mathrm{TA}_T$ be such that $w(p_j) = h_j(v(p_{i_0}), \ldots, v(p_{i_{n-1}})) = v(\psi_j)$ for $0 \leq j \leq m-1$. This implies

$$\begin{aligned} w(\psi) &= g(h_0(v(p_{i_0}), \ldots, v(p_{i_{n-1}})), \ldots, h_{m-1}(v(p_{i_0}), \ldots, v(p_{i_{n-1}}))) \\ &= f(v(p_{i_0}), \ldots, v(p_{i_{n-1}})) \\ &= \tau^f_S(v). \end{aligned}$$

On the other hand, $\tau_{\varphi,S}(v) = v(\varphi) = v(s(\psi)) = w(\psi)$ because of Corollary 2.10.9. This concludes our argument. □

Lemma 2.10.13. *Let $F = \{f_C \mid C \in \mathbf{C}\}$ be a set of truth functions indexed by $\mathbf{C}$ and let $\mathcal{C}$ be the extended clone generated by the set $\{f_C \mid C \in \mathbf{C}\}$. For every formula $\varphi \in \mathrm{PLFORM}_F$ and set of variables S such that $SV(\varphi) \subseteq S$ and $|S| = n$, there exists an n-ary truth function $f \in \mathcal{C}$ such that $\tau^f_S = \tau_{\varphi,S}$.*

Proof. The argument is by induction on the definition of formulas. For the basis step, let φ be a statement variable, $\varphi = p$, and let S be a set of statement variables, $S = \{p_{i_0}, \ldots, p_{i_{n-1}}\}$, where $p = p_{i_j}$ for some j, $0 \leq j \leq n-1$. We claim that $\tau_{p_{i_j},S} = \tau_S^{\pi_j^n}$. Indeed, let v be a truth assignment, $v \in \mathrm{TA}_S$. We have $\tau_{p_{i_j},S}(v) = v(p_{i_j})$ and

$$\tau_S^{\pi_j^n}(v) = \pi_j^n(v(p_{i_0}), \ldots, v(p_{i_{n-1}})) = v(p_{i_j}),$$

and this shows that the statement holds since $\pi_j^n \in \mathcal{C}$.

If $\varphi = C$ where C is a 0-ary connective symbol, we consider two cases. If $S = \emptyset$, then $f_C \in \mathcal{C}$ and we claim that $\tau_{C,\emptyset} = \tau_\emptyset^{f_C}$. Indeed, let v_0 be the unique partial truth assignment defined on $\emptyset$. Because of Definition 2.10.5, we have $\tau_{C,\emptyset}(v_0) = v_0(C) = f_C() = \tau_\emptyset^{f_C}(v_0)$. If $S = \{p_{i_0}, \ldots, p_{i_{n-1}}\}$ with $n > 0$, then let g be the n-ary function obtained from $f_C \in \mathcal{C}$ by degenerate composition. Clearly, $g \in \mathcal{C}$ and we claim that $\tau_{C,S} = \tau_S^g$. In fact, if $v \in \mathrm{TA}_S$, we have $\tau_{C,S}(v) = v(C) = f_C() = g(v(p_{i_0}), \ldots, v(p_{i_{n-1}})) = \tau_S^g(v)$.

Let now $\varphi = C(\psi_0, \ldots, \psi_{m-1})$, where C is an m-ary connective symbol, $m > 0$, and $\psi_0, \ldots, \psi_{m-1} \in \mathrm{PLFORM}_F$. Let $S = \{p_{i_0}, \ldots, p_{i_{n-1}}\}$ be a set of statement variables such that $SV(\varphi) \subseteq S$. Note that $SV(\psi_j) \subseteq S$ for $0 \leq j \leq m-1$. Suppose, by inductive hypothesis, that for each j, $0 \leq j \leq m-1$, there exists an n-ary truth function $h_j \in \mathcal{C}$ such that $\tau_{\psi_j,S} = \tau_S^{h_j}$ and let f be the truth function given by $f = f_C(h_0, \ldots, h_{m-1}) \in \mathcal{C}$. We claim that $\tau_S^f = \tau_{\varphi,S}$. Indeed, we can write

$$\begin{aligned}
\tau_S^f(v) &= f(v(p_{i_0}), \ldots, v(p_{i_{n-1}})) \\
&= f_C(h_0(v(p_{i_0}), \ldots, v(p_{i_{n-1}})), \ldots, h_{m-1}(v(p_{i_0}), \ldots, v(p_{i_{n-1}}))) \\
&= f_C(\tau_S^{h_0}(v), \ldots, \tau_S^{h_{m-1}}(v)) \\
&= f_C(\tau_{\psi_0,S}(v), \ldots, \tau_{\psi_{m-1},S}(v)) \\
&= f_C(v(\psi_0), \ldots, v(\psi_{m-1})) \\
&= v(C(\psi_0, \ldots, \psi_{m-1})) \\
&= v(\varphi) \\
&= \tau_{\varphi,S}(v),
\end{aligned}$$

for every $v \in \mathrm{TA}_S$, which justifies our claim. □

Theorem 2.10.14. *Let $F = \{f_C \mid C \in \mathbf{C}\}$ be a set of truth functions indexed by $\mathbf{C}$. The set $\mathbf{C}$ is complete in the sense of Definition 2.10.11 if and only if the set $\{f_C \mid C \in \mathbf{C}\}$ is complete in the sense of Definition 2.9.29.*

Proof. We will prove initially that algebraic completeness, that is, completeness in the sense of Definition 2.9.29, implies logical completeness, that is, completeness in the sense of Definition 2.10.11.

Suppose that $\{f_C \mid C \in \mathbf{C}\}$ is a complete set of truth functions indexed by $\mathbf{C}$. This means that the extended clone C generated by this set includes $\mathtt{TF}$. Let S be a finite, nonempty set of statement variables and let $\tau \in \mathtt{TT}_S$. The truth function f_τ belongs to $\mathcal{C}$ and, by Lemma 2.10.12, there exists a formula φ such that $SV(\varphi) \subseteq S$ and $\tau_S^{f_\tau} = \tau_{\varphi,S}$. Since, by Theorem 2.8.48, $\tau = \tau_S^{f_\tau}$, this proves that $\tau = \tau_{\varphi,S}$ and we obtain the completeness of $\mathbf{C}$ according to Definition 2.10.11.

Conversely, let $F = \{f_C \mid C \in \mathbf{C}\}$ be a set of truth functions indexed by $\mathbf{C}$ and suppose that $\mathbf{C}$ is a complete set of connectives. This means that for every finite, nonempty set of statement variables S and $\tau \in \mathtt{TT}_S$, there is a formula $\varphi \in \text{PLFORM}_F$ with $SV(\varphi) \subseteq S$ such that $\tau = \tau_{\varphi,S}$.

Let $f \in \mathtt{TF}_n$ be an arbitrary n-ary truth function, $n > 0$. If $S = \{p_0, \ldots, p_{n-1}\}$, consider the truth table τ_S^f. By the logical completeness of $\mathbf{C}$, there is a formula $\varphi \in \text{PLFORM}_F$ such that $SV(\varphi) \subseteq S$ and $\tau_S^f = \tau_{\varphi,S}$. There exists an n-ary truth function g in the extended clone $\mathcal{C}$ generated by $\{f_C \mid C \in \mathbf{C}\}$ such that $\tau_S^g = \tau_{\varphi,S} = \tau_S^f$ because of Lemma 2.10.13. By Theorem 2.8.48, we obtain $f = g \in \mathcal{C}$. This shows that the set F is algebraically complete. □

Example 2.10.15. Let $\mathbf{C} = \{P_4, \min\}$ be a set of connective symbols, where the truth functions f_{P_4} and $f_{\min}$ are as defined in Example 2.8.17. In Example 2.9.20, we have shown that the set $\{f_{P_4}, f_{\min}\}$ is complete in the sense of Definition 2.9.11. Theorem 2.9.30 implies that this set is complete in the sense of Definition 2.9.29, which means by Theorem 2.10.14 that $\mathbf{C}$ is logically complete. ▯

Example 2.10.16. The set $F = \{f_{P_3}, f_{P_4}\}$ indexed by $\mathbf{C} = \{P_3, P_4\}$ is not algebraically complete because both f_{P_3} and f_{P_4} are linear functions, as we saw in Example 2.8.17. Consequently, **C** is not a logically complete set of connectives. □

2.11 Circuits and Truth Functions

We begin by introducing a model of computations of truth functions.

Recall that we introduced the functions $c^m_{\mathbf{F}}, c^m_{\mathbf{T}}$ in Equalities 2.3.

Definition 2.11.1. Let $m \in \mathbf{N}$. An *m-ary data set* is a set of functions D that consists of m-ary projections and m-ary constant functions; in other words, $D \subseteq \{\pi^m_0, \ldots, \pi^m_{m-1}\} \cup \{c^m_{\mathbf{F}}, c^m_{\mathbf{T}}\}$. □

Note that if $m = 0$, then an m-ary data set is a subset of the set $\{c^0_{\mathbf{F}}, c^0_{\mathbf{T}}\}$ since no 0-ary projections exist.

Definition 2.11.2. Let **C** be a set of connective symbols and let $F = \{f_C \mid C \in \mathbf{C}\}$ be a set of truth functions indexed by the elements of **C**. If D is an m-ary data set, then an (F, D)*-computation* is a sequence of functions $\mathbf{c} = (f_0, \ldots, f_{n-1})$ such that $n \geq 1$ and for every i, $0 \leq i \leq n-1$, one of the following cases occurs:

(1) $f_i \in D$,
(2) $f_i = f_C(f_{j_0}, \ldots, f_{j_{k-1}})$ for some k-ary connective symbol $C \in \mathbf{C}$ and $0 \leq j_0, \ldots, j_{k-1} \leq i-1$.

In the last case, we refer to $\{f_{j_0}, \ldots, f_{j_{k-1}}\}$ as *a set of inputs of* f_i.

The *target of the computation* $\mathbf{c}$ is the function f_{n-1}. The number n is the *length of the computation.* □

Note that the target of an (F, D)-computation belongs to the extended clone generated by the set of functions $\{f_C \mid C \in \mathbf{C}\} \cup D$.

Example 2.11.3. Let $D = \{\pi^3_0, \pi^3_1, \pi^3_2, c^3_{\mathbf{T}}\}$ be a ternary data set. Consider the set $C = \{\wedge, +\}$ of binary connectives and the computation $\mathbf{c} = (f_0, f_1, f_2, f_3, f_4, f_5, f_6, f_7)$ such that

$$\begin{aligned} f_0 &= \pi^3_0, & f_4 &= f_+(f_0, f_3), \\ f_1 &= \pi^3_1, & f_5 &= f_\wedge(f_1, f_2), \\ f_2 &= \pi^3_2, & f_6 &= f_\wedge(f_4, f_5), \\ f_3 &= c^3_{\mathbf{T}}, & f_7 &= f_+(f_6, f_3). \end{aligned}$$

We have

$$
\begin{aligned}
f_0(x_0,x_1,x_2) &= x_0,\\
f_1(x_0,x_1,x_2) &= x_1,\\
f_2(x_0,x_1,x_2) &= x_2,\\
f_3(x_0,x_1,x_2) &= \mathbf{T},\\
f_4(x_0,x_1,x_2) &= x_0+\mathbf{T},\\
f_5(x_0,x_1,x_2) &= x_1 \wedge x_2,\\
f_6(x_0,x_1,x_2) &= (x_0+\mathbf{T}) \wedge (x_1 \wedge x_2),\\
f_7(x_0,x_1,x_2) &= ((x_0+\mathbf{T}) \wedge (x_1 \wedge x_2)) + \mathbf{T},
\end{aligned}
$$

for $x_0, x_1, x_2 \in \mathbf{Bool}$. The target of $\mathbf{c}$ is the function $f : \mathbf{Bool}^3 \longrightarrow \mathbf{Bool}$ given by

$$f(x_0,x_1,x_2) = ((x_0+\mathbf{T}) \wedge (x_1 \wedge x_2)) + \mathbf{T}$$

for $x_0, x_1, x_2 \in \mathbf{Bool}$. ∎

We remind the reader that a directed graph is a pair of sets (V, U), where V is the set of vertices of G and U is the set of edges of G, $U \subseteq V \times V$. For a vertex $\mathtt{v}$, the in-degree $d_G^-(\mathtt{v})$ is the number of incoming edges, $d_G^-(\mathtt{v}) = |\{(\mathtt{v}', \mathtt{v}) \mid (\mathtt{v}', \mathtt{v}) \in U\}|$, while the out-degree is the number of outgoing edges $d_G^+(\mathtt{v}) = |\{(\mathtt{v}, \mathtt{v}') \mid (\mathtt{v}, \mathtt{v}') \in U\}|$. We refer the reader to [13] for further terminology in graph theory.

Definition 2.11.4. Let S_m be the set $\{p_0, \ldots, p_{m-1}\}$ of statement variables, where $m \in \mathbf{N}$, and let $\mathbf{C}$ be a set of connective symbols that indexes the set of truth functions F. An *(F, m)-circuit with p outputs* is a quadruple $\mathcal{K} = (G, \varpi, q, r_{\text{out}})$, where $G = (V, U)$ is a finite acyclic digraph, $\varpi : V \longrightarrow \mathbf{C} \cup \mathbf{Bool} \cup S_m$ is a function called the *labeling function of* $\mathcal{K}$, the sequence $q = (\mathtt{e}_0, \ldots, \mathtt{e}_{n-1})$ is a listing of the edges of G, and r_{out} is a sequence of p distinct vertices of G called the *output sequence* of the circuit such that for every vertex $\mathtt{v}$ of G:

(1) if $\varpi(\mathtt{v}) \in S_m \cup \mathbf{Bool}$, then $d_G^-(\mathtt{v}) = 0$,
(2) if $\varpi(\mathtt{v}) = C$, where $C \in \mathbf{C}$ is a k-ary connective symbol, then $d_G^-(\mathtt{v}) = k$.

An *output vertex* of $\mathcal{K}$ is a member of the sequence r_{out}.

If $\mathrm{Ran}(\varpi) \subseteq \mathbf{C} \cup S_m$, then we refer to $\mathcal{K}$ as a *constant-free circuit.*

The numbers

$$\mathtt{fo}(\mathcal{K}) = \max\{d_G^+(\mathtt{v}) | \mathtt{v} \in V\},$$
$$\mathtt{fi}(\mathcal{K}) = \max\{d_G^-(\mathtt{v}) | \mathtt{v} \in V\}$$

are the *fan-out* and the *fan-in of the* (F, m)*-circuit* $\mathcal{K}$, respectively.

We say that $\mathcal{K}$ is an F*-circuit* if $\mathcal{K}$ is an (F, m)-circuit for some m.

□

If $\mathcal{K} = (G, \varpi, q, r_{\mathrm{out}})$ where $r_{\mathrm{out}} = (\mathtt{v}_{\mathrm{out}})$, then we denote the single-output circuit $\mathcal{K}$ simply by $(G, \varpi, q, \mathtt{v}_{\mathrm{out}})$.

Unless stated otherwise, we will consider only single-output circuits. Note that an (F, m)-circuit is also an $(F, m + k)$-circuit for every $k \in \mathbf{N}$.

The vertices of a circuit $\mathcal{K}$ labeled by connective symbols are alternatively referred to as the *gates* of the circuit. Those vertices labeled by constants or variables are *the inputs* of the circuit.

Definition 2.11.5. The *size* of a circuit $\mathcal{K}$ is the number of gates of $\mathcal{K}$. We denote the size of $\mathcal{K}$ by $||\mathcal{K}||$. □

Observe that the fan-out of a circuit $\mathcal{K}$ indicates the maximum number of times a gate of the circuit may be used to provide an input for another gate of the circuit. A circuit with fan-out 0 or 1 is called a *formula circuit.*

Definition 2.11.6. Let $\mathcal{K} = (G, \varpi, q, r_{\mathrm{out}})$ be an (F, m)-circuit. The *depth of a vertex* $\mathtt{v}$ of $\mathcal{K}$, $\mathtt{depth}(\mathtt{v})$ is defined to be the maximum length of a path ending in $\mathtt{v}$.

The *delay of the circuit* $\mathcal{K}$, $\mathtt{delay}(\mathcal{K})$, is the largest number $\mathtt{depth}(\mathtt{v})$ where $\mathtt{v}$ is an output vertex. □

The definition of the depth of a vertex is meaningful because of the acyclicity of the underlying graph of the circuit. Indeed, since the graph is acyclic, the length of any path has to be less than the number of vertices of the graph. Moreover, $\mathtt{delay}(\mathcal{K}) \leq ||\mathcal{K}||$ because the number of gates on a longest path that ends at an output vertex is either $\mathtt{delay}(\mathcal{K})$ or $\mathtt{delay}(\mathcal{K}) + 1$. (The latter case may occur when a longest path that ends in an output vertex begins with a gate labeled with a 0-ary connective.)

Starting from a circuit we define the function computed at each vertex of the circuit as follows.

Definition 2.11.7. Let $\mathcal{K} = (G, \varpi, q, r_{\text{out}})$ be an (F, m)-circuit, where $G = (V, U)$ and $q = (\mathtt{e}_0, \ldots, \mathtt{e}_{n-1})$.

Define the mapping $\Theta_{\mathcal{K}} : V \longrightarrow \mathtt{TF}_m$ as follows:

(1) If $\mathtt{depth}(\mathtt{v}) = 0$, then

$$\Theta_{\mathcal{K}}(\mathtt{v}) = \begin{cases} \pi_j^m & \text{if } \varpi(\mathtt{v}) = p_j \\ c_{\mathbf{T}}^m & \text{if } \varpi(\mathtt{v}) = \mathbf{T} \\ c_{\mathbf{F}}^m & \text{if } \varpi(\mathtt{v}) = \mathbf{F} \\ c_{\mathbf{T}}^m & \text{if } \varpi(\mathtt{v}) = C \quad \text{and} \quad f_C = c_{\mathbf{T}}^0 \\ c_{\mathbf{F}}^m & \text{if } \varpi(\mathtt{v}) = C \quad \text{and} \quad f_C = c_{\mathbf{F}}^0. \end{cases}$$

(2) If $\mathtt{depth}(\mathtt{v}) > 0$ and $\varpi(\mathtt{v}) = C$, where C is a k-ary connective symbol, then let the incoming edges of vertex $\mathtt{v}$ be $(\mathtt{e}_{\ell_0}, \ldots, \mathtt{e}_{\ell_{k-1}})$, where $\ell_0 < \cdots < \ell_{k-1}$ and edge $\mathtt{e}_{\ell_r}$ begins at vertex $\mathtt{v}_{\ell_r}$ for $0 \leq r \leq k-1$. We define $\Theta_{\mathcal{K}}(\mathtt{v}) = f_C(f_{\ell_0}, \cdots, f_{\ell_{k-1}})$, where $f_{\ell_r} = \Theta_{\mathcal{K}}(\mathtt{v}_{\ell_r})$ for $0 \leq r \leq k-1$.

The *function computed at vertex* $\mathtt{v}$ *by the circuit* $\mathcal{K}$ is the function $\Theta_{\mathcal{K}}(\mathtt{v})$. The *sequence of functions computed by* $\mathcal{K}$ is $(\Theta_{\mathcal{K}}(\mathtt{v}_0), \ldots, \Theta_{\mathcal{K}}(\mathtt{v}_{p-1}))$, where $r_{\text{out}} = (\mathtt{v}_0, \ldots, \mathtt{v}_{p-1})$ is the output sequence of $\mathcal{K}$. If $p = 1$, we refer to $\Theta_{\mathcal{K}}(\mathtt{v}_0)$ as the *function computed by the circuit* and we denote it by $f_{\mathcal{K}}$. □

Consider the m-ary data set

$$D = \{\Theta_{\mathcal{K}}(\mathtt{v}) \mid d_G^-(\mathtt{v}) = 0\}.$$

It is easy to verify that if $(\mathtt{v}_0, \ldots, \mathtt{v}_k)$ is a topological sort of G, then the sequence $(\Theta_{\mathcal{K}}(\mathtt{v}_0), \ldots, \Theta_{\mathcal{K}}(\mathtt{v}_k))$ is an (F, D)-computation.

Let $V_{\mathtt{v}}$ be the set that consists of those vertices $\mathtt{u}$ of the circuit $\mathcal{K} = (G, \varpi, q, r_{\text{out}})$ such that there is a path from $\mathtt{u}$ to $\mathtt{v}$. It is easy to see that if $\mathtt{u} \in V_{\mathtt{v}}$, then $V_{\mathtt{u}} \subseteq V_{\mathtt{v}}$. The subgraph of G determined by $V_{\mathtt{v}}$ will be denoted by $G_{\mathtt{v}}$ and the restriction of ϖ to $V_{\mathtt{v}}$ will be denoted by $\varpi_{\mathtt{v}}$. The sequence $q_{\mathtt{v}}$ obtained from q by removing those edges not connecting vertices in $V_{\mathtt{v}}$ is clearly a listing of the set of edges of $G_{\mathtt{v}}$. This justifies the following definition.

Definition 2.11.8. The *subcircuit of the circuit* $\mathcal{K} = (G, \varpi, q, r_{\text{out}})$ *determined by the vertex* $\mathtt{v}$ is the single-output circuit $\mathcal{K}_{\mathtt{v}} = (G_{\mathtt{v}}, \varpi_{\mathtt{v}}, q_{\mathtt{v}}, \mathtt{v})$. ▯

Note that the depth of a vertex in $\mathcal{K}_{\mathtt{v}}$ is the same as the depth of the vertex in $\mathcal{K}$.

Theorem 2.11.9. *Let* $\mathcal{K} = (G, \varpi, q, r_{\text{out}})$ *be an* (F, m)*-circuit. For all vertices* $\mathtt{v}, \mathtt{v}', \mathtt{w}$ *of* G *such that* $\mathtt{w} \in V_{\mathtt{v}} \cap V_{\mathtt{v}'}$*, we have* $\Theta_{\mathcal{K}_{\mathtt{v}}}(\mathtt{w}) = \Theta_{\mathcal{K}_{\mathtt{v}'}}(\mathtt{w})$.

Proof. The argument is by induction on the depth of $\mathtt{w}$ in $\mathcal{K}$. The basis step, $\mathtt{depth}(\mathtt{w}) = 0$, is immediate. Suppose that $\mathtt{depth}(\mathtt{w}) > 0$ and that the statement holds for vertices of smaller depth. Let the incoming edges of $\mathtt{w}$ be $(\mathtt{v}_{\ell_0}, \mathtt{w}), \ldots, (\mathtt{v}_{\ell_{n-1}}, \mathtt{w})$, where the edges are ordered according to q, and assume that $\varpi(\mathtt{w}) = C$. We have

$$\begin{aligned}\Theta_{\mathcal{K}_{\mathtt{v}}}(\mathtt{w}) &= f_C(\Theta_{\mathcal{K}_{\mathtt{v}}}(\mathtt{v}_{\ell_0}), \ldots, \Theta_{\mathcal{K}_{\mathtt{v}}}(\mathtt{v}_{\ell_{n-1}}))\\ &= f_C(\Theta_{\mathcal{K}_{\mathtt{v}'}}(\mathtt{v}_{\ell_0}), \ldots, \Theta_{\mathcal{K}_{\mathtt{v}'}}(\mathtt{v}_{\ell_{n-1}}))\\ &\quad \text{(by the inductive hypothesis)}\\ &= \Theta_{\mathcal{K}_{\mathtt{v}'}}(\mathtt{w}).\end{aligned}$$

□

Corollary 2.11.10. *Let* $\mathcal{K} = (G, \varpi, q, r_{\text{out}})$ *be an* (F, m)*-circuit. For all vertices* $\mathtt{v}, \mathtt{w}$ *of* G *such that* $\mathtt{w} \in V_{\mathtt{v}}$*, we have* $\Theta_{\mathcal{K}_{\mathtt{v}}}(\mathtt{w}) = \Theta_{\mathcal{K}_{\mathtt{w}}}(\mathtt{w})$.

Proof. This follows immediately from Theorem 2.11.9. □

Let F be a nonempty set of truth functions. Observe that every truth function that is computed by an (F, m)-circuit with s gates is also computed by an (F, m)-circuit with s' gates for all $s' \geq s$. Indeed, if $\mathcal{K}$ is an (F, m)-circuit with s gates that computes f, we can obtain an (F, m)-circuit $\mathcal{K}'$ that computes the same function f by adding $s' - s$ new gates with out-degrees 0 and connecting those gates to new input vertices.

Example 2.11.11. Let $F = \{f_{\to}, f_{\wedge}\}$. Figure 2.21 gives an $(F, 2)$-circuit $\mathcal{K} = (G, \varpi, (\mathtt{e}_0, \ldots, \mathtt{e}_5), \mathtt{v}_4)$. The gates of $\mathcal{K}$ are $\mathtt{v}_2, \mathtt{v}_3$, and $\mathtt{v}_4$ and the inputs are $\mathtt{v}_0$ and $\mathtt{v}_1$.

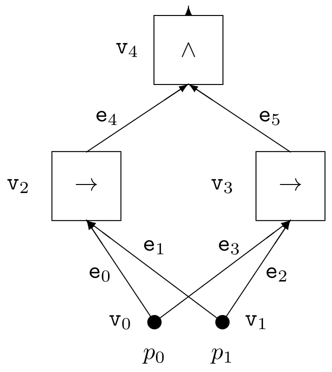


Fig. 2.21. The $(F, 2)$-circuit $\mathcal{K}$.

The functions $f_i = \Theta_{\mathcal{K}}(\mathtt{v}_i)$ are given by

$$\begin{aligned} f_0 &= \pi_0^2, \\ f_1 &= \pi_1^2, \\ f_2 &= f_{\to}(f_0, f_1), \\ f_3 &= f_{\to}(f_1, f_0), \\ f_4 &= f_{\wedge}(f_2, f_3). \end{aligned}$$

Since $f_0(b_0, b_1) = b_0$ and $f_1(b_0, b_1) = b_1$, it is easy to see that $f_4(b_0, b_1) = (b_0 \to b_1) \wedge (b_1 \to b_0) = b_0 \leftrightarrow b_1$. Note that the order of the incoming edges is significant here because $f_{\to}(b, b')$ is distinct from $f_{\to}(b', b)$. □

Example 2.11.12. Let $d : \mathbf{Bool}^4 \longrightarrow \mathbf{Bool}$ be the discriminator function defined by

$$d(a_0, a_1, a_2, a_3) = \begin{cases} a_2 & \text{if } a_0 = a_1, \\ a_3 & \text{otherwise,} \end{cases}$$

for every $(a_0, a_1, a_2, a_3) \in \mathbf{Bool}^4$ and let $F = \{f_+, f_\wedge, f_\vee\}$. An $(F, 4)$-circuit that computes d is given in Figure 2.22. Its fan-out is equal to 2. We omitted the listing of the edges because the functions in F are symmetric in their input arguments, that is, their value does not change when the inputs are permuted.

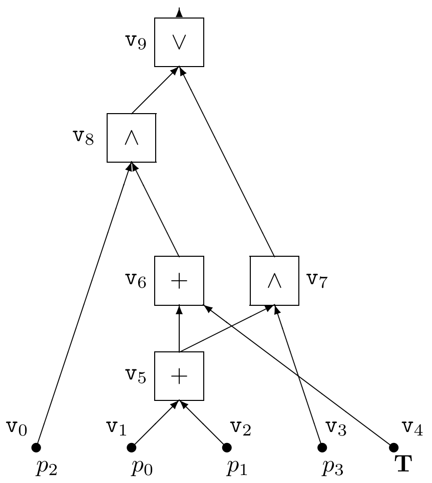


Fig. 2.22. Circuit for computing the function d.

Indeed, the functions $f_i = \Theta_{\mathcal{K}}(\mathtt{v}_i)$ are given by

$$\begin{aligned} f_0 &= \pi^4_2, \\ f_1 &= \pi^4_0, \\ f_2 &= \pi^4_1, \\ f_3 &= \pi^4_3, \\ f_4 &= c^4_{\mathbf{T}}, \\ f_5 &= f_+(f_1, f_2), \\ f_6 &= f_+(f_5, f_4), \\ f_7 &= f_\wedge(f_5, f_3), \\ f_8 &= f_\wedge(f_0, f_6), \\ f_9 &= f_\vee(f_8, f_7). \end{aligned}$$

It is not difficult to see that $d = f_9$. ∎

Example 2.11.13. The two-output circuit $\mathcal{K} = (G, \varpi, q, (\mathtt{v}_0, \mathtt{v}_1))$ given in Figure 2.23 computes the sequence of functions $(f_\wedge, f_\vee)$. This circuit is known as the *comparator circuit* since its two outputs give the smaller and larger inputs according to the order previously introduced on **Bool**. ∎

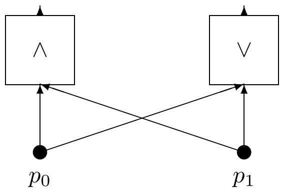


Fig. 2.23. The comparator circuit.

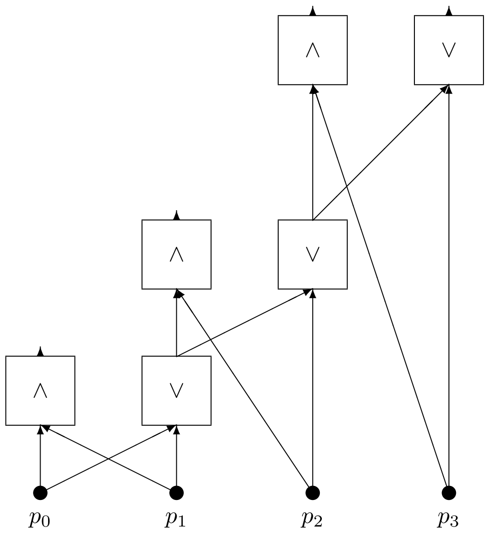


Fig. 2.24. Single-pass circuit for bubble sort for $n = 4$.

Example 2.11.14. The well-known bubble sort algorithm can be used to sort sequences of truth values, that is, to rearrange such a sequence in increasing order according to the order defined on **Bool**. For a sequence of length $n > 0$, $\vec{a}^0 = (a_0^0, \ldots, a_{n-1}^0)$, the algorithm makes $n-1$ passes over the sequence. The ith pass involves the first $n-i+1$ members of the output $\vec{a}^{i-1} = (a_0^{i-1}, \ldots, a_{n-1}^{i-1})$ of the $i-1$st pass and consists of comparing the first two elements, swapping them if they are not in order, comparing the second and the third element, swapping them if they are not in order, etc. At the end of the ith pass, the last i elements of $\vec{a}^i$ are in their final positions, as can be easily seen by the reader.

The circuit shown in Figure 2.24 implements the first pass of the bubble sort algorithm when $n = 4$. It can be seen that the gates in

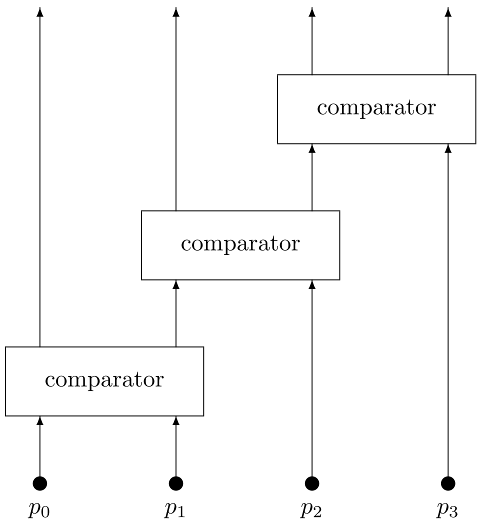


Fig. 2.25. Schematic representation of single-pass circuit for $n = 4$.

this circuit can be grouped together to form comparators, and this allows us to give a schematic representation of the circuit as shown in Figure 2.25. Whenever practical, we will use this alternative "black box" representation of circuits.

A circuit that performs a bubble sort pass over $n > 0$ inputs is shown in Figure 2.26. We will denote this circuit by $\mathcal{BP}_n$. Clearly, since the circuit involves $n-1$ comparators, we have $||\mathcal{BP}_n|| = 2n-2$, for $n \geq 1$. Using circuits of the form $\mathcal{BP}_k$ as building blocks, we can construct an (F, n)-circuit with n outputs that uses bubble sort to produce the sorted permutation of the values at the input nodes (see Figure 2.27). Here $F = \{f_\wedge, f_\vee\}$. The size of this sorting circuit is easily seen to be $n^2 - n$. With the natural ordering of the output nodes, the multiple-output circuit computes the sequence $(\mathrm{th}_{n,n}, \mathrm{th}_{n-1,n}, \ldots, \mathrm{th}_{1,n})$, where the functions th were introduced in Definition 2.8.22. □

Let F be a set of truth functions indexed by the set $\mathbf{C}$ of connective symbols. For a sequence of m-ary truth functions $(f_0, \ldots, f_{p-1})$, let $\mathbf{CIRC}^n_F(f_0, \ldots, f_{p-1})$ be the collection of all (F, m)-circuits with fan-out less than or equal to n that compute $(f_0, \ldots, f_{p-1})$. If $n, p \in \mathbf{N}$ and $n \leq p$, then

$$\mathbf{CIRC}^n_F(f) \subseteq \mathbf{CIRC}^p_F(f). \tag{2.9}$$

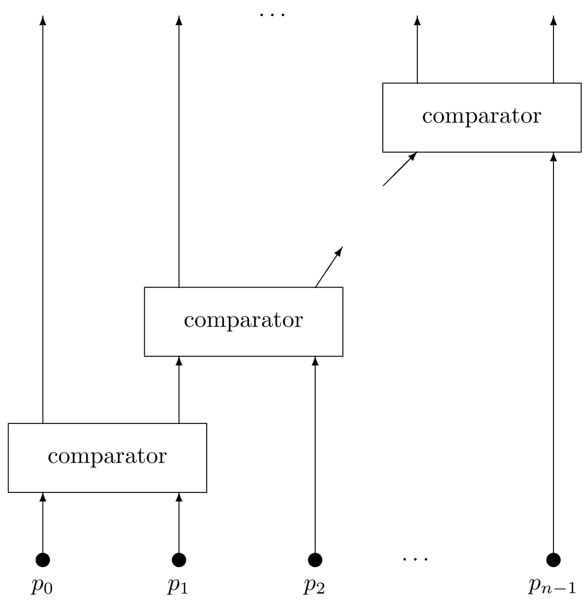


Fig. 2.26. Schematic representation of the circuit $\mathcal{BP}_n$.

The collection of all circuits that compute the function f will be denoted by $\mathbf{CIRC}_F(f)$, where

$$\mathbf{CIRC}_F(f) = \bigcup_{n \in \mathbf{N}} \mathbf{CIRC}_F^n(f).$$

Definition 2.11.15. The *combinational complexity with fan-out n* over the set F is a partial mapping $\mathsf{COMB}_F^n : \mathtt{TF} \longrightarrow \mathbf{N}$, where f belongs to $\mathrm{Dom}(\mathsf{COMB}_F^n)$ if and only if $\mathbf{CIRC}_F^n(f) \neq \emptyset$ and in this case,

$$\mathsf{COMB}_F^n(f) = \min\{||\mathcal{K}|| \mid \mathcal{K} \in \mathbf{CIRC}_F^n(f)\}.$$

$\mathsf{COMB}_F^1(f)$ is the *formula complexity* of f.

A circuit $\mathcal{K} \in \mathbf{CIRC}_F^n(f)$ such that $||\mathcal{K}|| = \mathsf{COMB}_F^n(f)$ is a *fan-out n, F-optimal circuit for f.*

The *combinational complexity with arbitrary fan-out* over the set F is a partial mapping $\mathsf{COMB}_F^\infty : \mathtt{TF} \longrightarrow \mathbf{N}$, where $f \in \mathrm{Dom}(\mathsf{COMB}_F^\infty)$ if and only if $\mathbf{CIRC}_F(f) \neq \emptyset$ and in this case,

$$\mathsf{COMB}_F^\infty(f) = \min\{||\mathcal{K}|| \mid \mathcal{K} \in \mathbf{CIRC}_F(f)\}.$$

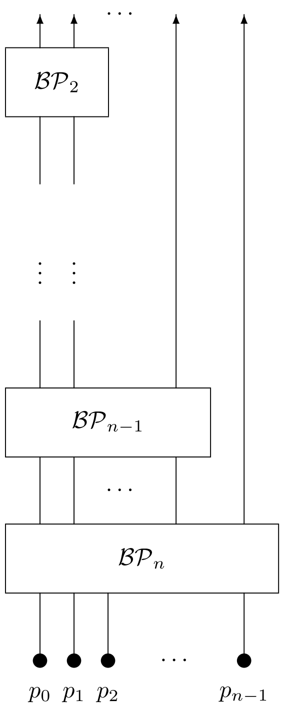


Fig. 2.27. Circuit for bubble sort.

A circuit $\mathcal{K} \in \mathbf{CIRC}_F^\infty(f)$ such that $||\mathcal{K}|| = \mathsf{COMB}_F^\infty(f)$ is an *unlimited fan-out, F-optimal circuit for f.*

The *delay complexity over F* is a partial mapping DELAY_F : $\mathtt{TF} \longrightarrow \mathbf{N}$, where $f \in \mathrm{Dom}(D_F)$ if and only if $\mathbf{CIRC}_F(f) \neq \emptyset$ and

$$\mathsf{DELAY}_F(f) = \min\{\mathtt{delay}(\mathcal{K}) \mid \mathcal{K} \in \mathbf{CIRC}_F(f)\}.$$

□

Whenever some of these complexity measures are undefined for a function f, we may say that their corresponding value is ∞. For example, if there is no (F, m)-circuit that computes $f \in \mathtt{TF}_m$ with fan-out n, then we write $\mathsf{COMB}_F^n(f) = \infty$.

The inclusion (2.9) implies

$$\mathsf{COMB}_F^\infty(f) \leq \mathsf{COMB}_F^p(f) \leq \mathsf{COMB}_F^n(f) \qquad (2.10)$$

for every $f \in \mathrm{Dom}(\mathsf{COMB}_F^n)$ and every $n, p \in \mathbf{N}$ such that $n \leq p$.

Theorem 2.11.16. *Let* $\mathbf{C}$ *be a set of connective symbols and* $F = \{f_C \mid C \in \mathbf{C}\}$ *be a set of truth functions indexed by* $\mathbf{C}$.

For every formula $\varphi \in \mathrm{PLFORM}_F$ *and* $m \in \mathbf{N}$ *such that* $SV(\varphi) \subseteq S_m$, *there is a constant-free single-output* (F, m)*-circuit* $\mathcal{K}_{\varphi,m}$ *that satisfies the following conditions:*

(1) $\mathcal{K}_{\varphi,m}$ *is a formula circuit and*
(2) *the function computed by the circuit equals* $f_{\tau_{\varphi,S_m}}$.

Proof. The argument is by induction on the definition of the formula φ. If $\varphi = p_i \in S_m$, let $\mathcal{K} = (G, \varpi, q, \mathtt{v})$, where G is the graph having a single vertex $\mathtt{v}$ and no edges, $\varpi(\mathtt{v}) = p_i$ and $q = \lambda$ is the empty sequence. Clearly, the fan-out of G is 0 and $\Theta_{\mathcal{K}}(\mathtt{v}) = \pi_i^m = f_{\tau_{p_i,S_m}}$.

Suppose now that $\varphi = C(\varphi_0, \ldots, \varphi_{n-1})$, where C is an n-ary connective symbol with $n \geq 1$, or $\varphi = C$, where C is a 0-ary connective symbol, and that for every formula φ_i there exists a circuit $\mathcal{K}_i = (G_i, \varpi_i, q_i, \mathtt{v}_i)$ such that the graph $G_i = (V_i, U_i)$ has fan-out no greater than 1, and the function computed by $\mathcal{K}_i$ is $f_{\tau_{\varphi_i,S_m}}$ for $0 \leq i \leq n-1$. We assume that the set of vertices V_i of the graphs G_i is pairwise disjoint and that $\mathtt{v} \notin \bigcup_{0 \leq i \leq n-1} V_i$.

Consider the graph $G = (V, U)$, where $V = \{\mathtt{v}\} \cup \bigcup_{0 \leq i \leq n-1} V_i$, $U = \{(\mathtt{v}_0, \mathtt{v}), \ldots, (\mathtt{v}_{n-1}, \mathtt{v})\} \cup \bigcup_{0 \leq i \leq n-1} U_i$, shown in Figure 2.28.

The sequence q is the concatenation of the sequences $q_0, \ldots, q_{n-1}$ followed by the sequence $((\mathtt{v}_0, \mathtt{v}), \ldots, (\mathtt{v}_{n-1}, \mathtt{v}))$, and the mapping ϖ is given by

$$\varpi(z) = \begin{cases} \varpi_i(z) & \text{if } z \in V_i \text{ for } 0 \leq i \leq n-1, \\ C & \text{if } z = \mathtt{v}. \end{cases}$$

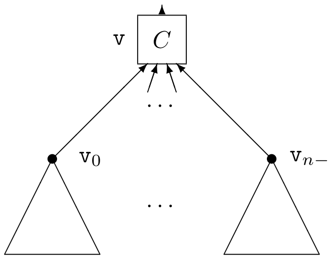


Fig. 2.28. The circuit $\mathcal{K}'$.

Let $\mathcal{K} = (G, \varpi, q, \mathtt{v})$. It is clear that $\mathcal{K}$ is a formula circuit. We claim that $\Theta_{\mathcal{K}}(\mathtt{v})$, the function computed by the circuit equals $f_{\tau_{\varphi,S_m}}$. Indeed, we have

$$\begin{aligned}
f_{\tau_{\varphi,S_m}}(\vec{a}) &= \tau_{\varphi,S_m}(v_{\vec{a},S_m})\\
&= v_{\vec{a},S_m}(\varphi)\\
&= f_C(v_{\vec{a},S_m}(\varphi_0), \ldots, v_{\vec{a},S_m}(\varphi_{n-1}))\\
&= f_C(f_{\tau_{\varphi_0,S_m}}(\vec{a}), \ldots, f_{\tau_{\varphi_{n-1},S_m}}(\vec{a}))\\
&= f_C(\Theta_{\mathcal{K}_0}(\mathtt{v}_0)(\vec{a}), \ldots, \Theta_{\mathcal{K}_{n-1}}(\mathtt{v}_{n-1})(\vec{a}))\\
&\quad \text{(by inductive hypothesis)}\\
&= f_C(\Theta_{\mathcal{K}}(\mathtt{v}_0)(\vec{a}), \ldots, \Theta_{\mathcal{K}}(\mathtt{v}_{n-1})(\vec{a}))\\
&\quad \text{(by Corollary 2.11.10)}\\
&= \Theta_{\mathcal{K}}(\mathtt{v})(\vec{a}).
\end{aligned}$$

$\square$

Note that if $f \in \mathtt{TF}_m$ and φ is a formula such $SV(\varphi) = S_m$ and $\tau_\varphi = \tau^f_{S_m}$, then $\mathsf{COMB}^1_F(f) \leq ||\mathcal{K}_{\varphi,m}||$.

Theorem 2.11.17. *Let* $\mathbf{C}$ *be a set of connective symbols,* $F = \{f_C \mid C \in \mathbf{C}\}$*, and* $\mathcal{K} = (G, \varpi, q, r_{\text{out}})$ *be an* (F, m)*-circuit:*

(a) *If* $\mathcal{K}$ *is constant-free, then for every vertex* $\mathtt{v}$ *of the graph* G *there exists a formula* $\varphi_{\mathtt{v}} \in \text{PLFORM}_F$ *such that* $\Theta_{\mathcal{K}}(\mathtt{v}) = f_{\tau_{\varphi_{\mathtt{v}},S_m}}$.

(b) *Let* $\mathbf{C}_0 = \mathbf{C} \cup \{\top, \bot\}$*, where* $\top, \bot \notin \mathbf{C}$*, and let* $F_0 = \{f_C \mid C \in \mathbf{C}_0\}$*, where* $f_\top = c^0_{\mathbf{T}}$ *and* $f_\bot = c^0_{\mathbf{F}}$*. Then, for every vertex* $\mathtt{v}$ *of the graph* G *there exists a formula* $\varphi_{\mathtt{v}} \in \text{PLFORM}_{F_0}$ *such that* $\Theta_{\mathcal{K}}(\mathtt{v}) = f_{\tau_{\varphi_{\mathtt{v}},S_m}}$.

Proof. For Part (a), the proof is by induction on $\mathtt{depth}(\mathtt{v})$. For the basis step, $\mathtt{depth}(\mathtt{v}) = 0$, the formula $\varphi_{\mathtt{v}}$ is given by $\varphi_{\mathtt{v}} = p_j$ if $\varpi(\mathtt{v}) = p_j$ and by $\varphi_{\mathtt{v}} = C$ if $\varpi(\mathtt{v}) = C$, where C is a 0-ary connective symbol.

For the inductive step, suppose that the statement holds for vertices of depth less than $\mathtt{depth}(\mathtt{v}) > 0$ and that $\varpi(\mathtt{v}) = C$. If the incoming edges of $\mathtt{v}$ are $(\mathtt{v}_0, \mathtt{v}), \ldots, (\mathtt{v}_{n-1}, \mathtt{v})$ listed in the order specified by q, then, by inductive hypothesis, we have formulas $\varphi_{\mathtt{v}_0}, \ldots, \varphi_{\mathtt{v}_{n-1}}$ such that $\Theta_{\mathcal{K}}(\mathtt{v}_i) = f_{\tau_{\varphi_{\mathtt{v}_i},S_m}}$ for $0 \leq i \leq n-1$. We leave it to the reader to prove that the formula $\varphi_{\mathtt{v}} = C(\varphi_{\mathtt{v}_0}, \ldots, \varphi_{\mathtt{v}_{n-1}})$ has the desired property.

The proof of Part (b) differs only in the basis step and is left to the reader. $\square$

Corollary 2.11.18. *Let F be an indexed set of truth functions and $f \in \mathtt{TF}_m$ be a truth function. The following four statements are equivalent:*

(1) $f \in \widehat{\widehat{F}}$,
(2) *there is a formula $\varphi \in \mathrm{PLFORM}_F$ such that $SV(\varphi) \subseteq S_m$ and $f = f_{\tau_{\varphi,S_m}}$,*
(3) *there is a constant-free single-output (F, m)-circuit $\mathcal{K}$ of fan-out less than or equal to one such that $f = f_{\mathcal{K}}$,*
(4) *there is a constant-free single-output (F, m)-circuit $\mathcal{K}$ such that $f = f_{\mathcal{K}}$.*

Proof. The equivalence of (1) and (2) is a consequence of Lemma 2.10.12, Lemma 2.10.13, and Theorem 2.8.48. The second statement implies (3) by Theorem 2.11.16, (3) implies (4) trivially, and (4) implies (2) by Theorem 2.11.17, Part (a). $\square$

Corollary 2.11.19. *Let F be an indexed set of truth functions, $F_0 = F \cup \{f_\top, f_\bot\}$, and $f \in \mathtt{TF}_m$ be a truth function. The following four statements are equivalent:*

(1) $f \in \widehat{\widehat{F_0}}$,
(2) *there is a formula $\varphi \in \mathrm{PLFORM}_{F_0}$ such that $SV(\varphi) \subseteq S_m$ and $f = f_{\tau_{\varphi,S_m}}$,*
(3) *there is a single-output (F, m)-circuit $\mathcal{K}$ of fan-out less than or equal to one such that $f = f_{\mathcal{K}}$,*
(4) *there is a single-output (F, m)-circuit $\mathcal{K}$ such that $f = f_{\mathcal{K}}$.*

Proof. By Corollary 2.11.18 applied to F_0, the first statement is equivalent to the second. Suppose now that (2) holds. By the same corollary, there is a constant-free (F_0, m)-circuit $\mathcal{K}'$ of fan-out less than or equal to one such that $f = f_{\mathcal{K}'}$. Let $\mathcal{K}$ be the (F, m)-circuit obtained from $\mathcal{K}'$ by replacing all labels $\bot$ with $\mathbf{F}$ and $\top$ with $\mathbf{T}$. Then, it is easy to see that $f_{\mathcal{K}} = f_{\mathcal{K}'} = f$, which shows that (2) implies (3). It is immediate that (3) implies (4) and (4) implies (2) by Theorem 2.11.17, Part (b). $\square$

Thus, if F_0 is a complete set of truth functions in the sense of Definition 2.9.29, then, for every m-ary truth function, there is an (F, m)-circuit that computes the function.

Corollary 2.11.20. *For every single-output (F, m)-circuit $\mathcal{K}$, there exists a single-output (F, m)-circuit $\mathcal{K}'$ such that $\mathcal{K}'$ has fan-out less than or equal 1 and computes the same m-ary function as $\mathcal{K}$.*

Proof. For $m = 0$, the function computed by $\mathcal{K}$ is necessarily $c^0_{\mathbf{F}}$ or $c^0_{\mathbf{T}}$. The one-vertex circuit labeled by $\mathbf{F}$ or $\mathbf{T}$, respectively, computes the same function and has fan-out 0.

Now suppose $m > 0$ and let f be the m-ary function computed by $\mathcal{K}$. By Theorem 2.11.17, there is an F-formula φ such that $f = f_{\tau_{\varphi}, S_m}$. By Theorem 2.11.16, there is an (F, m)-circuit $\mathcal{K}'$ of fan-out less than or equal to 1 that computes the function f. $\square$

The previous corollary implies that all functions COMB^n_F have the same domain for $n \in \mathbf{N} \cup \{\infty\}$ and $n > 0$. Any member of this domain is called a *function computable by F-circuits.*

Suppose we have an (F, m)-circuit $\mathcal{K}$ that computes a function $g \in \mathtt{TF}_m$ and that $f \in \mathtt{TF}_n$ is a conjugate of g, say $f = g^{\wp}$, where $\wp : \{0, \ldots, m-1\} \longrightarrow \{0, \ldots, n-1\}$. By Corollary 2.11.19, there is an (F, n)-circuit $\mathcal{K}'$ that computes f. Such a circuit can be constructed effectively from $\mathcal{K}$ by a suitable change of the statement variables that label the input nodes of $\mathcal{K}$. Namely, each label p_i of an input node is replaced by $p_{\wp(i)}$. The resulting circuit will be denoted by $\mathcal{K}^{\wp}$. We leave it to the reader to prove that the function computed by $\mathcal{K}^{\wp}$ is $g^{\wp}$ (see Exercise 173).

Theorem 2.11.21. *Let F, F' be two finite, complete indexed sets of truth functions. There is a constant c such that for all truth functions f,*

$$\mathrm{COMB}^{\infty}_F(f) \leq c \cdot \mathsf{COMB}^{\infty}_{F'}(f).$$

Proof. Let f be a truth function and let $\mathcal{K}'$ be an unlimited fan-out F'-optimal circuit for f. The completeness of F implies that for every $f_{C'} \in F'$, there is an unlimited fan-out, F-optimal circuit $\mathcal{K}_{C'}$ for $f_{C'}$. Let $\mathcal{K}''$ be the circuit obtained from $\mathcal{K}'$ by replacing each gate labeled by a connective symbol C' by a copy of $\mathcal{K}_{C'}$. In this circuit, the node linked to the ith input of the gate labeled C' is directly connected to the nodes of $\mathcal{K}_{C'}$ that were linked to the inputs of $\mathcal{K}_{C'}$ labeled p_{i-1}. Of course, the input nodes of $\mathcal{K}_{C'}$ labeled with variables are removed. Observe that $||\mathcal{K}''|| \leq c||\mathcal{K}||$, where $c = \max\{||\mathcal{K}_{C'}|| \mid f_{C'} \in F'\}$. Therefore, $\mathsf{COMB}^{\infty}_{F}(f) \leq c\mathsf{COMB}^{\infty}_{F'}(f)$. $\square$

A similar result for fan-out n where $1 < n < \infty$ is shown in Corollary 2.11.27.

Theorem 2.11.22. *Let F, F' be two finite, complete sets of truth functions. There is a constant d such that for all truth functions f, $\mathsf{DELAY}_F(f) \leq d \cdot \mathsf{DELAY}_{F'}(f)$.*

Proof. Let f be a truth function and let $\mathcal{K}'$ be an F'-circuit for f of minimal delay. The completeness of F implies that for every $f_{C'} \in F'$, there is an F-circuit $\mathcal{K}_{C'}$ for $f_{C'}$ that has minimal delay. Let $\mathcal{K}''$ be the circuit obtained from $\mathcal{K}'$ by replacing each gate labeled by a connective symbol C' by a copy of $\mathcal{K}_{C'}$, as in the previous theorem. Note that $\texttt{delay}(\mathcal{K}'') \leq d \cdot \texttt{delay}(\mathcal{K}')$, where $d = \max\{\texttt{delay}(\mathcal{K}_{C'}) \mid f_{C'} \in F'\}$. Therefore,

$$\mathsf{DELAY}_F(f) \leq d\mathsf{DELAY}_{F'}(f).$$

$\square$

We now investigate the relationships between $\mathsf{COMB}^2_F, \ldots, \mathsf{COMB}^{\infty}_F$ for given F.

Observe that if F consists only of constant functions, then the function computed by an (F, m)-circuit is either a projection or a constant function. Therefore, since each such function is computable by a one-vertex circuit without gates, we have $\mathsf{COMB}^n_F(f) = 0$ for every $f \in \mathrm{Dom}(\mathsf{COMB}^n_F)$. This shows that sets of connectives that denote only constant functions are of little interest for our study; thus, we will consider from now on sets F that contain at least one non-constant function.

Lemma 2.11.23. *Let F be an indexed set of truth functions. If $\mathcal{K}$ is a fan-out n, F-optimal circuit, or an unlimited fan-out F-optimal circuit for a truth function f, then the number of vertices of $\mathcal{K}$ with positive out-degree is at least $||\mathcal{K}||$.*

Proof. Since $\mathcal{K}$ is an optimal circuit with limited or unlimited fan-out, there is no gate with out-degree 0 except the output gate. (Note that the output vertex can be an input vertex.) Also, no vertex of $\mathcal{K}$ is labeled with a 0-ary connective symbol because any such gate could be replaced with an input vertex labeled with a truth value, thereby decreasing the number of gates of the circuit.

If the output vertex of $\mathcal{K}$ is an input vertex, then the result follows immediately because the optimality of the circuit implies $||\mathcal{K}|| = 0$. Therefore, suppose that the output vertex r_{out} is a gate. The acyclicity of the underlying graph of $\mathcal{K}$ implies the existence of a vertex $\mathtt{v}$ with in-degree 0 that is joined by a path to r_{out}. The vertex $\mathtt{v}$ is an input because it cannot be labeled by a 0-ary connective symbol. Consequently, $\mathtt{v} \neq r_{\text{out}}$ and this proves the existence of an input vertex with positive out-degree which implies the desired inequality. $\square$

Lemma 2.11.24. *Let F be a set of truth functions that contains at least one nonconstant function. Then, there is an $(F, 1)$-circuit $\mathcal{K}_{id,F}$ of depth one or two and size equal to the depth that computes the identity function.*

Proof. Let f_C be a nonconstant n-ary function in F. There are two n-tuples $\vec{c}, \vec{d}$ in $\mathbf{Bool}^n$ such that $f_C(\vec{c}) \neq f_C(\vec{d})$. By Lemma 2.8.19, there is a sequence $\vec{a}(0), \ldots, \vec{a}(k)$ with $|\Delta(\vec{a}(i), \vec{a}(i+1))| = 1$, $\vec{a}(0) = \vec{c}$, and $\vec{a}(k) = \vec{d}$. Clearly, there must be two consecutive vectors in this sequence $(b_0, \ldots, b_{j-1}, b_j, b_{j+1}, \ldots b_{n-1})$, $(b_0, \ldots, b_{j-1}, \overline{b}_j, b_{j+1}, \ldots b_{n-1})$ such that

$$f_C(b_0, \ldots, b_{j-1}, b_j, b_{j+1}, \ldots b_{n-1}) \neq f_C(b_0, \ldots, b_{j-1}, \overline{b_j}, b_{j+1}, \ldots b_{n-1}).$$

If $f_C(b_0, \ldots, b_j, \ldots, b_{n-1}) = b_j$, then the desired $(F, 1)$-circuit is given in Figure 2.29(a). Otherwise, that is, if $f_C(b_0, \ldots, b_j, \ldots, b_{n-1}) = \overline{b_j}$, then the desired $(F, 1)$-circuit is given in Figure 2.29(b). $\square$

Theorem 2.11.25. *Let F be a finite set of truth functions that contains at least one nonconstant function and let k be the maximum arity of a function in F. There is a number $\ell(F) = ||\mathcal{K}_{id,F}|| \in \{1, 2\}$ such that*

$$\mathsf{COMB}^n_F(f) \leq \left(1 + \ell(F)\frac{k-1}{n-1}\right) \mathsf{COMB}^\infty_F(f)$$

for every $f \in \mathrm{Dom}(\mathsf{COMB}^n_F)$ and $n \in \mathbf{N}$ such that $n \geq 2$.

Proof. Let $\mathsf{COMB}^\infty_F(f) = c$ and let $\mathcal{K} = (G, \varpi, q, r_{\text{out}})$ be a circuit with c gates that computes f. For each vertex $\mathtt{v}$ of $\mathcal{K}$, define $r_{\mathtt{v}}$ to be the fan-out of $\mathtt{v}$.

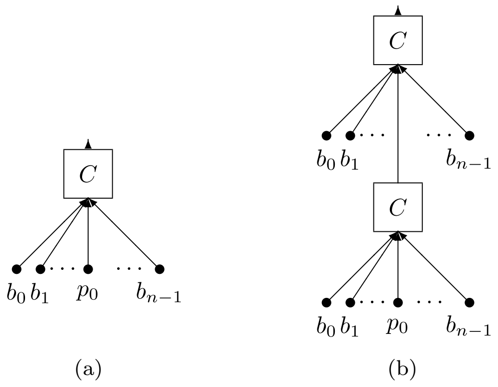


Fig. 2.29. Circuits that compute the identity function.

In order to create a circuit $\mathcal{K}'$ with fan-out bounded by n for f starting from $\mathcal{K}$, we rearrange the edges emerging from every vertex $\mathtt{v}$ as follows. If $r_{\mathtt{v}} \leq n$, no changes are needed at vertex $\mathtt{v}$. Otherwise, that is, if $r_{\mathtt{v}} > n$, the first $n-1$ edges are left alone, while the last $r_{\mathtt{v}} - (n-1)$ edges are replaced by a single edge connecting $\mathtt{v}$ to a copy of the circuit $\mathcal{K}_{id,F}$ defined in Lemma 2.11.24, where the input vertex of $\mathcal{K}_{id,F}$ labeled p_0 is replaced by the edge. If $r_{\mathtt{v}} \leq 2n-1$, then the output of the copy of $\mathcal{K}_{id,F}$, which is the same as the output of $\mathtt{v}$, is sent through $r_{\mathtt{v}} - (n-1)$ edges to the vertices of the circuit that would have received the last $r_{\mathtt{v}} - (n-1)$ original edges. Otherwise, we repeat the process as shown in Figure 2.30 until we have enough edges to compensate for the initial $r_{\mathtt{v}}$ edges. The number $m_{\mathtt{v}}$ of copies of $\mathcal{K}_{id,F}$ introduced by this process applied to the vertex $\mathtt{v}$ is defined by

$$(n-1)m_{\mathtt{v}} + n \geq r_{\mathtt{v}} > (n-1)(m_{\mathtt{v}} - 1) + n,$$

which gives $m_{\mathtt{v}} < (r_{\mathtt{v}} - 1)/(n-1)$.

For those vertices $\mathtt{v}$ such that $0 < r_{\mathtt{v}} \leq n$, define $m_{\mathtt{v}} = 0$. Then, for every vertex $\mathtt{v}$ with $r_{\mathtt{v}} > 0$, we have $m_{\mathtt{v}} \leq (r_{\mathtt{v}} - 1)/(n-1)$.

Since $||\mathcal{K}'|| = c + \sum\{m_{\mathtt{v}}\ell(F) \mid r_{\mathtt{v}} > 0\}$, we have

$$\begin{aligned}\mathsf{COMB}^n_F(f) &\leq c + \sum\{m_{\mathtt{v}}\ell(F) \mid r_{\mathtt{v}} > 0\} \\ &= c + \ell(F)\sum\{m_{\mathtt{v}} \mid r_{\mathtt{v}} > 0\}\end{aligned}$$

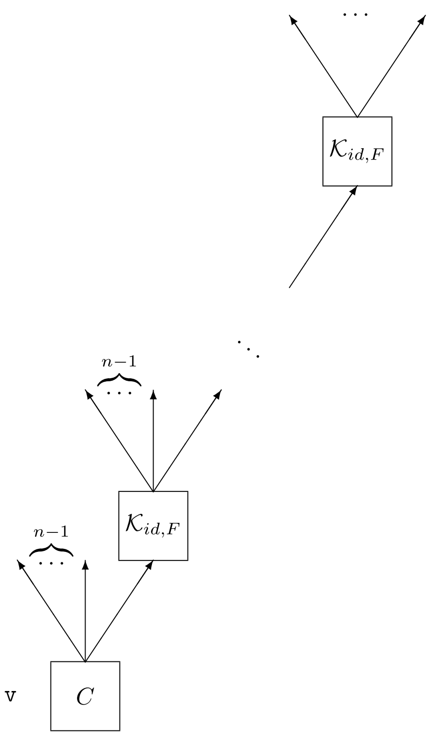


Fig. 2.30. Construction of $\mathcal{K}'$.

$$\leq c + \frac{\ell(F)}{n-1} \sum \{(r_{\mathtt{v}} - 1) \mid r_{\mathtt{v}} > 0\}$$
$$= c + \frac{\ell(F)}{n-1} \left(\sum \{r_{\mathtt{v}} \mid r_{\mathtt{v}} > 0\} - |\{\mathtt{v} \mid r_{\mathtt{v}} > 0\}| \right).$$

Observe that the number of outgoing edges of $\mathcal{K}$, $\sum\{r_{\mathtt{v}} \mid r_{\mathtt{v}} > 0\}$ equals the number of incoming edges and the latter number is bounded by ck; also, by Lemma 2.11.23, we have $c \leq |\{\mathtt{v} \mid r_{\mathtt{v}} > 0\}|$. Using the last inequality, we obtain

$$\mathsf{COMB}^n_F(f) \leq c + \frac{\ell(F)}{n-1}(ck - c)$$
$$= \left(1 + \ell(F)\frac{k-1}{n-1}\right) \mathsf{COMB}^{\infty}_F(f).$$

□

Corollary 2.11.26. *Let F be a finite set of truth functions that contains at least one nonconstant function. For every $p, q \in \mathbf{N} \cup \{\infty\}$ such that $2 \leq p, q$, there are constants c_F^{pq}, c_F^{qp} such that for every truth function computable by F-circuits,*

$$\mathsf{COMB}_F^p(f) \leq c_F^{pq} \mathsf{COMB}_F^q(f),$$
$$\mathsf{COMB}_F^q(f) \leq c_F^{qp} \mathsf{COMB}_F^p(f).$$

Proof. Suppose $p < q$. Then, we can take $c_F^{qp} = 1$. By Theorem 2.11.25, we have

$$\mathsf{COMB}_F^p(f) \leq c_F^{pq} \mathsf{COMB}_F^\infty(f) \leq c_F^{pq} \mathsf{COMB}_F^q(f),$$

where $c_F^{pq} = 1 + \ell(F)\frac{k-1}{p-1}$ and k is the maximum arity of a function in F. □

The last corollary shows that for finite F, there are essentially two different complexity measures up to multiplicative constants, namely, COMB_F^1, the formula complexity, and COMB_F^∞.

Corollary 2.11.27. *Let F, F' be two finite, complete sets of truth functions. For $n \in \mathbf{N}$ and $n \geq 2$, there is a constant c_n such that for all truth functions f, $\mathsf{COMB}_F^n(f) \leq c_n \mathsf{COMB}_{F'}^n(f)$.*

Proof. By Theorem 2.11.21 and Corollary 2.11.26, there exist constants c, $c_F^{n\infty}$ and $c_{F'}^{\infty n}$ such that

$$\begin{aligned}\mathsf{COMB}_F^n(f) &\leq c_F^{n\infty} \mathsf{COMB}_F^\infty(f)\\ &\leq c_F^{n\infty} c\, \mathsf{COMB}_{F'}^\infty(f) \leq c_{F'}^{\infty n} c_F^{n\infty} c\, \mathsf{COMB}_{F'}^n(f)\end{aligned}$$

for every $f \in \mathtt{TF}$. □

Theorem 2.11.28. *Let F be a finite set of truth functions that contains a non-constant function and let k be the maximum arity of a function in F. For $s \geq 2$, there are no more than*

$$c_F(s, m) = \frac{(|F|(s + m + 1)^k)^s s}{s!}$$

truth functions computable by (F, m)-circuits of size less than or equal to s.

Proof. We noted earlier that if a function is computed by a circuit with fewer than s gates, then it is also computed by a circuit with s gates. The output vertex of such a circuit is either an input vertex or a gate.

There are $m + 2$ functions that are computed by (F, m)-circuits that have an input vertex as the output. It is easy to see that any of these functions can also be computed by circuits whose output is a gate and have at most two gates. In fact, such circuits can be obtained from the circuit $\mathcal{K}_{id,F}$ by replacing the label of the input p_0 with a suitable truth value or variable p_i.

Therefore, it remains only to count those functions that are computed by (F, m)-circuits of size s whose output vertex is a gate. Since we have no limits on the fan-out of the circuits, for every (F, m)-circuit of size s, there is an (F, m)-circuit with equal size that computes the same function and has no more than $m + 2$ input vertices, so we give now an upper bound on the number of such circuits.

For each gate in a circuit of size s, there are $|F|$ ways of choosing the connective symbol that labels the gate; also, each of its input edges can be connected to one of the other vertices of which there are no more than $s+m+1$. This gives us no more than $|F|(s+m+1)^k$ ways of labeling and connecting each vertex, so we have no more than $(|F|(s + m + 1)^k)^s$ ways of labeling and "wiring" the graph. Finally, we have s choices for the output gate, which shows that there are no more than $((|F|(s + m + 1)^k)^s s$ (F, m)-circuits with size s and no more than $m + 2$ input vertices.

We claim that for each function f computable by an (F, m)-circuit with s gates whose output is a gate, there are at least $s!$ (F, m)-circuits of size s whose output is a gate, which have no more than $m + 2$ input vertices, and which compute f. Indeed, there is such a circuit having no two gates with the same label and sequence of input vertices. But then all of the $s!$ circuits obtained by permutations of the gates also compute f. Thus, the number of functions computable by an (F, m)-circuit with s gates whose output is a gate does not exceed $\frac{(|F|(s+m+1)^k)^s s}{s!}$. □

Theorem 2.11.29. *Let F be a finite set of truth functions and let k be the maximum arity of a function in F. For each $m \geq 3$, there is a truth function in $\mathtt{TF}_m$ that cannot be computed by any (F, m)-circuit with fewer than $\frac{2^m}{km+\log_2 |F|}$ gates.*

Proof. If F contains only constant functions, then no nonconstant function other than the projections can be computed by (F, m)-circuits, so the statement follows immediately.

Therefore, we may assume that F contains a nonconstant function and hence, that the maximum arity k of a function in F is at least 1. It is easy to observe that if $m \geq 3$, then we have both $\frac{1}{km+\log_2 |F|} < 0.5$ and $\frac{m+1}{2^m} \leq 0.5$, which gives $\frac{1}{km+\log_2 |F|} + \frac{m+1}{2^m} < 1$, or equivalently,

$$\frac{2^m}{km + \log_2 |F|} + m + 1 < 2^m.$$

By choosing $s = \frac{2^m}{km+\log_2 |F|}$, we have $s + m + 1 < 2^m$, so $\log_2(s + m + 1) < m$. In turn, this implies

$$\frac{2^m}{km + \log_2 |F|} \left(\log_2 |F| + k \log_2(s + m + 1)\right) < 2^m,$$

which amounts to $s (\log_2 |F| + k \log_2(s + m + 1)) < 2^m$. This inequality yields $\left(|F|(s + m + 1)^k\right)^s < 2^{2^m}$, so $c_F(s, m) < 2^{2^m}$ because $s \leq s!$. By Theorem 2.11.28 and the fact that there are 2^{2^m} m-ary truth functions, for $m \geq 3$, there is an m-ary truth function that cannot be computed by any (F, m)-circuit with fewer than $\frac{2^m}{km+\log_2 |F|}$ gates. $\square$

Lemma 2.11.30. *Let F be a finite set of truth functions that contains a nonconstant function and assume that the maximum arity k of a function of F is greater than 1. For $\epsilon > 0$ and m a positive integer, define $r_\epsilon(m) = \frac{2^m}{m} \frac{\log_2 m}{1+\epsilon}$. Then, for all m large enough, if $s_m = \frac{2^m}{(k-1)m}$, we have*

$$c_F(s_m, m) \leq 2^{2^m - r_\epsilon(m)}.$$

Proof. It is clear that for sufficiently large m, we have $s_m > m+1$. Therefore, we can write for all sufficiently large m:

$$\begin{aligned} c_F(s_m, m) &= \frac{(|F|(s_m + m + 1)^k)^{s_m} s_m}{s_m!} \\ &\leq \frac{(|F|(s_m + m + 1)^k)^{s_m} s_m}{\sqrt{2\pi} \frac{s_m^{s_m+0.5}}{e^{s_m}}} \\ &\qquad \text{(by Stirling's Formula)} \end{aligned}$$

$$\leq \frac{(e \cdot |F| \cdot (s_m + m + 1)^k)^{s_m} s_m}{s_m^{s_m + 0.5}}$$

$$\leq (e|F|2^k)^{s_m} \cdot s_m^{(k-1)s_m} \cdot s_m^{0.5}.$$

Thus it suffices to show that

$$(e|F|2^k)^{s_m} \cdot s_m^{(k-1)s_m} \cdot s_m^{0.5} \leq 2^{2^m - r_\epsilon(m)}$$

or equivalently

$$s_m \log_2(e \cdot |F| \cdot 2^k) + (k-1)s_m \log_2 s_m + 0.5 \log_2 s_m \leq 2^m - r_\epsilon(m).$$

Substituting the values of s_m and $r_\epsilon(m)$ in the last inequality, after elementary transformations we obtain the equivalent inequality

$$\begin{aligned}&0.5(m - \log_2(k-1) - \log_2 m)\\ &\quad\leq \frac{2^m}{m}\left(\log_2(k-1) + \frac{\epsilon \log_2 m}{1+\epsilon} - \frac{\log_2(e \cdot |F| \cdot 2^k)}{k-1}\right),\end{aligned}$$

which holds for m sufficiently large. □

Definition 2.11.31. Let P be a property of truth functions, that is, $P \subseteq \mathtt{TF}_*$, and let $P_n = P \cap \mathtt{TF}_n$, for $n \in \mathbf{N}$. *P holds for almost all truth functions* if

$$\lim_{n\to\infty} \frac{|P_n|}{2^{2^n}} = 1.$$

▯

The next result is both weaker and stronger than Theorem 2.11.29. It is weaker in that it holds only for large m, and it is stronger in that it shows that almost all m-ary truth functions require large (F, m)-circuits.

Theorem 2.11.32. *Let F be a finite collection of truth functions. Let K be the larger of 2 and the maximum arity of a function in F. Then, for almost all truth functions f, $\mathsf{COMB}_F^\infty(f) \geq \frac{2^m}{(K-1)m}$, where m is the arity of f.*

Proof. If F contains only constant functions, then for each m, there are at most $m + 2$ truth functions that are computable by (F, m)-circuits. If F contains no function of arity greater than 1, then for each m, there are at most $2m+2$ such functions. In either of these cases, the theorem holds. Thus, we can assume that F contains a nonconstant function and the maximum arity of a function in F is greater than 1.

Fix $\epsilon > 0$. By Theorem 2.11.28, the number of truth functions computable by (F, m)-circuits of size less than or equal to $s_m = \frac{2^m}{(K-1)m}$ does not exceed $c_F(s_m, m)$ and by Lemma 2.11.30, the number of these functions does not exceed $2^{2^m - r_\epsilon(m)}$. Consequently, for at least $2^{2^m} - 2^{2^m - r_\epsilon(m)}$ m-ary truth functions f, we have $\mathsf{COMB}_F^m(f) \geq \frac{2^m}{(K-1)m}$. Note that

$$\lim_{m\to\infty} \frac{2^{2^m} - 2^{2^m - r_\epsilon(m)}}{2^{2^m}} = 1$$

because $\lim_{m\to\infty} r_\epsilon(m) = \infty$, which yields the desired result. □

In the following corollary, we use the standard asymptotic notation Ω.[8]

Corollary 2.11.33. *Let F be a complete set of truth functions. Almost all truth functions f require (F, m)-circuits of size $\Omega(2^m/m)$, where m is the arity of f.*

Proof. The statement follows directly from Theorem 2.11.32. □

Next, we will establish that for a complete set of truth functions the asymptotic lower bound of Corollary 2.11.33 is in fact an asymptotic upper bound as well. Initially, we will establish this result for a particular complete set of truth functions, namely, $\{f_\neg, f_\wedge, f_\vee\}$. By Theorem 2.11.21, the result is translatable to any complete set of truth functions.

Lemma 2.11.34. *Let $\vec{b} = (b_0, \ldots, b_{n-1}) \in \mathbf{Bool}^n$ and let $f = f_{\vec{b}}$ be the minterm function that corresponds to $\vec{b}$. If $n = k + l$ and $\vec{b}_{i,j} = (b_i, \ldots, b_j)$ for $0 \leq i \leq j \leq n-1$, we have $f = (f_{\vec{b}_{0,k-1}})^{\wp'} \wedge (f_{\vec{b}_{k,n-1}})^{\wp''}$, where $\wp' : \{0, \ldots, k-1\} \longrightarrow \{0, \ldots, n-1\}$, $\wp'' : \{0, \ldots, l-1\} \longrightarrow \{0, \ldots, n-1\}$ are given by $\wp'(i) = i$ and $\wp''(j) = j+k$ for $0 \leq i \leq k-1$ and $0 \leq j \leq l-1$.*

[8]We write $f = \Omega(g)$, where $f, g : \mathbf{N} \longrightarrow \mathbf{R}$ if there is a positive real number c and a natural number n_0 such that $n \geq n_0$ implies $0 \leq cg(n) \leq f(n)$.

Proof. The proof amounts to the direct application of the definition of conjugate function. □

Example 2.11.35. Let $\vec{b} = (\mathbf{F}, \mathbf{T}, \mathbf{T}, \mathbf{F}, \mathbf{T}) \in \mathbf{Bool}^5$. We have the minterm function

$$f_{\vec{b}}(x_0, x_1, x_2, x_3, x_4) = \overline{x_0} \wedge x_1 \wedge x_2 \wedge \overline{x_3} \wedge x_4,$$

for $x_0, \ldots, x_4 \in \mathbf{Bool}$. For $k = 3$ and $l = 2$, the decomposition described in Lemma 2.11.34 amounts to the obvious equality

$$\overline{x_0} \wedge x_1 \wedge x_2 \wedge \overline{x_3} \wedge x_4 = (\overline{x_0} \wedge x_1 \wedge x_2) \wedge (\overline{x_3} \wedge x_4).$$

Indeed, with the choices of $\wp'$ and $\wp''$ of the lemma, for $f_{\vec{b}_{0,2}}(x_0, x_1, x_2) = \overline{x_0} \wedge x_1 \wedge x_2$ and $f_{\vec{b}_{3,4}}(x_0, x_1) = \overline{x_0} \wedge x_1$, we have

$$f^{\wp'}_{\vec{b}_{0,2}}(x_0, \ldots, x_4) = \overline{x_0} \wedge x_1 \wedge x_2,$$

$$f^{\wp''}_{\vec{b}_{3,4}}(x_0, \ldots, x_4) = \overline{x_3} \wedge x_4$$

for $x_0, \ldots, x_4 \in \mathbf{Bool}$. ▯

Lemma 2.11.36. *Let $F = \{f_\wedge, f_\neg\}$. For $n \in \mathbf{N}$, there is an (F, n)-circuit $\mathcal{K}_n$ of size $2^n + O(n2^{n/2})$ such that for every n-ary minterm function $f_{\vec{b}}$, there is a vertex $\mathtt{v}$ in $\mathcal{K}_n$ such that $\Theta_{\mathcal{K}_n}(\mathtt{v}) = f_{\vec{b}}$.*

Proof. Note that for each m there are 2^m m-ary minterm functions and each one can be computed by an (F, m)-circuit of size at most $2m-1$. Therefore, there is an (F, m)-circuit $\mathcal{H}_m$ of size $O(m2^m)$ such that for every m-ary minterm function there is a vertex $\mathtt{v}$ in $\mathcal{H}_m$ such that $\Theta_{\mathcal{H}_m}(\mathtt{v})$ is that minterm function.

Since $n = \lceil n/2 \rceil + \lfloor n/2 \rfloor$, by Lemma 2.11.34, for $n \geq 2$, every n-ary minterm function f can be expressed as $g^{\wp'} \wedge h^{\wp''}$, where g is an $\lceil n/2 \rceil$-ary minterm function, h is an $\lfloor n/2 \rfloor$-ary minterm function, and $\wp' : \{0, \ldots, \lceil n/2 \rceil - 1\} \longrightarrow \{0, \ldots, n-1\}$, $\wp'' : \{0, \ldots, \lfloor n/2 \rfloor - 1\} \longrightarrow \{0, \ldots, n-1\}$ are fixed functions independent of f. For $n \geq 2$, the circuit $\mathcal{K}_n$ consists of the circuits $\mathcal{H}^{\wp'}_{\lceil n/2 \rceil}$ and $\mathcal{H}^{\wp''}_{\lfloor n/2 \rfloor}$ supplemented by a set of 2^n $\wedge$-gates, $\{\mathtt{v}_f \mid f$ is an n-ary minterm function$\}$. More specifically, if f, g, h are as above and $g = \Theta_{\mathcal{H}_{\lceil n/2 \rceil}}(\mathtt{v}')$, and $h = \Theta_{\mathcal{H}_{\lfloor n/2 \rfloor}}(\mathtt{v}'')$, then we add edges from $\mathtt{v}'$ and $\mathtt{v}''$ to $\mathtt{v}_f$. The total number of gates of $\mathcal{K}_n$ is $2^n + O(\lceil n/2 \rceil 2^{\lceil n/2 \rceil}) + O(\lfloor n/2 \rfloor 2^{\lfloor n/2 \rfloor}) = 2^n + O(n2^{n/2})$. □

In the following theorem, we use the asymptotic notation o.[9]

Theorem 2.11.37. *For every truth function f,*

$$\mathsf{COMB}_F^\infty(f) \le \frac{2^n}{n} + o\left(\frac{2^n}{n}\right),$$

where $f \in \mathtt{TF}_n$ and $F = \{f_\neg, f_\wedge, f_\vee\}$.

Proof. Let $A_0, \dots, A_{\ell-1}$ be the standard s-partition of $\mathbf{Bool}^k$ for some k and s such that $1 \le k \le n$ and $1 \le s \le 2^k$. Here $\ell = \lceil 2^k/s \rceil$. Consider Lupanov's (k,s)-representation of f introduced by Equation (2.8):

$$f = \bigvee\{f_{i,\omega}^0 \wedge f_{i,\omega}^1 \mid 0 \le i \le \ell - 1, \omega : A_i \longrightarrow \mathbf{Bool} \text{ and } C_{A_i,\omega} \neq \emptyset\}.$$

Recall that

$$f_{i,\omega}^0(x_0,\dots,x_{n-1}) = \begin{cases} \omega(x_0,\dots,x_{k-1}) & \text{if } (x_0,\dots,x_{k-1}) \in A_i \\ \mathbf{F} & \text{otherwise} \end{cases}$$

and

$$f_{i,\omega}^1(x_0,\dots,x_{n-1}) = \begin{cases} \mathbf{T} & \text{if } (x_k,\dots,x_{n-1}) \in C_{A_i,\omega} \\ \mathbf{F} & \text{otherwise.} \end{cases}$$

Define the functions $g_{i,\omega} \in \mathtt{TF}_k$ and $h_{i,\omega}$ in $\mathtt{TF}_{n-k}$ by

$$g_{i,\omega}(x_0,\dots,x_{k-1}) = \begin{cases} \omega(x_0,\dots,x_{k-1}) & \text{if } (x_0,\dots,x_{k-1}) \in A_i \\ \mathbf{F} & \text{otherwise} \end{cases}$$

and

$$h_{i,\omega}(x_0,\dots,x_{n-k-1}) = \begin{cases} \mathbf{T} & \text{if } (x_0,\dots,x_{n-k-1}) \in C_{A_i,\omega} \\ \mathbf{F} & \text{otherwise.} \end{cases}$$

Note that $f_{i,\omega}^0 = g_{i,\omega}^{\wp'}$ and $f_{i,\omega}^1 = h_{i,\omega}^{\wp''}$, where $\wp' : \{0,\dots,k-1\} \longrightarrow \{0,\dots,n-1\}$, $\wp'' : \{0,\dots,n-k-1\} \longrightarrow \{0,\dots,n-1\}$ are given by $\wp'(j) = j$ and $\wp''(l) = k + l$. By Lemma 2.11.36, there is an (F,k)-circuit $\mathcal{K}_k$ of size $O(2^k)$ that computes every k-ary minterm function,

[9]We write $f = o(g)$, where $f, g : \mathbf{N} \longrightarrow \mathbf{R}_+$ if $\lim_{n\to\infty} \frac{f(n)}{g(n)} = 0$.

and there is an $(F, n-k)$-circuit $\mathcal{K}_{n-k}$ of size $O(2^{n-k})$ that computes every $n-k$-ary minterm function.

Next, we construct an (F, k)-circuit $\mathcal{H}_g$ that computes all of the functions $g_{i,\omega}$ using their disjunctive normal forms and the circuit $\mathcal{K}_k$. A k-ary minterm function is used in constructing $g_{i,\omega}$ only if its corresponding k-tuple belongs to A_i. Since $A_0, \ldots, A_\ell$ is a partition, and there are no more than 2^s choices for ω for a given A_i, each minterm function is used no more than 2^s times in building $\mathcal{H}_g$. It follows that the number of additional gates used in building $\mathcal{H}_g$ from $\mathcal{K}_k$ does not exceed $2^k 2^s$.

Similarly, we construct an $(F, n-k)$-circuit $\mathcal{H}_h$ that computes all the functions $h_{i,\omega}$ using their disjunctive normal forms and the circuit $\mathcal{K}_{n-k}$. Since for each i the sets $C_{i,\omega}$ form a partition of $\mathbf{Bool}^{n-k}$, no $n-k$-ary minterm function is used more than ℓ times in building $\mathcal{H}_h$. Thus, $\mathcal{H}_h$ is constructed from $\mathcal{K}_{n-k}$ using no more than $2^{n-k}\ell$ additional gates beyond the ones used in building $\mathcal{K}_{n-k}$.

The circuit $\mathcal{H}_g^{\wp'}$ computes all the functions $f^0_{i,\omega}$ and the circuit $\mathcal{H}_h^{\wp''}$ computes all the functions $f^1_{i,\omega}$.

Using Lupanov's (k, s)-representation, the function f is computed by a circuit $\mathcal{K}$ which we build by using copies of $\mathcal{H}_g^{\wp'}$ and $\mathcal{H}_h^{\wp''}$ and no more than $2\ell 2^s$ additional gates. The total number of gates in $\mathcal{K}$ is no more than

$$O(2^k) + O(2^{n-k}) + 2^k 2^s + \ell 2^{n-k} + 2\ell 2^s.$$

Since $\ell \leq 2^k/s + 1$, we obtain the upper bound

$$O(2^k + 2^{n-k}) + 2^{k+s} + \frac{2^n}{s} + 2^{n-k} + \frac{2^{s+1+k}}{s} + 2^{s+1}.$$

Let $k = \lceil 3\log_2 n \rceil$ and $s = n - \lceil 5\log_2 n \rceil$. For sufficiently large values of n, we have $k \leq n$ and $s \geq 1$. Therefore, $||\mathcal{K}|| \leq 2^n/n + o(2^n/n)$, which gives the desired result. □

Corollary 2.11.38. *For every complete set of connectives F and for every truth function f, $\mathit{COMB}_F^{\infty}(f) = O(\frac{2^n}{n})$, where $f \in \mathtt{TF}_n$.*

Proof. This statement is a consequence of Theorems 2.11.37 and 2.11.21. □

Next, we prove a lower bound on the combinational complexity of the threshold functions $\mathrm{th}_{2,n}$ introduced in Definition 2.8.22.

Lemma 2.11.39. *For $n \geq 3$ and $1 < k < n$, there are no numbers i and j and truth functions $f \in \mathtt{TF}_{n-1}$ and $g \in \mathtt{TF}_2$ such that $0 \leq i < j \leq n-1$ and*

$$\begin{aligned} &th_{k,n}(a_0, \ldots, a_{n-1}) \\ &\quad = f(a_0, \ldots, a_{i-1}, a_{i+1}, \ldots, a_{j-1}, a_{j+1}, \ldots, a_{n-1}, g(a_i, a_j)) \end{aligned}$$

for $a_0, \ldots, a_{n-1} \in$ **Bool***.*

Proof. For $0 \leq i < j \leq n-1$ and $a, b \in \mathbf{Bool}$, define $h^{i,j}_{a,b} \in \mathtt{TF}_{n-2}$ by

$$\begin{aligned} &h^{i,j}_{a,b}(a_0, \ldots, a_{n-3}) \\ &\quad = \mathrm{th}_{k,n}(a_0, \ldots, a_{i-1}, a, a_i, \ldots, a_{j-2}, b, a_{j-1}, \ldots, a_{n-3}) \end{aligned}$$

for $a_0, \ldots, a_{n-3} \in \mathbf{Bool}$. Observe that the functions $h^{i,j}_{0,0}, h^{i,j}_{0,1}, h^{i,j}_{1,1}$ are all distinct because $1 < k < n$.

If $\mathrm{th}_{k,n}$ had a representation as in the statement of the lemma, then there would be at most two distinct functions among the $h^{i,j}_{a,b}$, which shows the impossibility of such a decomposition of $\mathrm{th}_{k,n}$. □

Theorem 2.11.40. *For all $n \geq 2$, we have $\mathsf{COMB}^{\infty}_{\mathtt{TF}_2}(th_{2,n}) \geq 2n - 3$.*

Proof. The proof is by induction on n. The basis step, $n = 2$, is trivial. For the inductive step, assume that $n > 2$ and $\mathsf{COMB}^{\infty}_{\mathtt{TF}_2}(\mathrm{th}_{2,n-1}) \geq 2(n-1) - 3 = 2n - 5$ and let $\mathcal{K}$ be an optimal $\mathtt{TF}_2$-circuit with unlimited fan-out for $\mathrm{th}_{2,n}$. Since $\mathrm{th}_{2,n}$ is neither a constant function nor a projection, $\mathcal{K}$ must contain at least one gate. Hence, by the acyclicity of the graph underlying $\mathcal{K}$, there is a gate $\mathtt{v}$ directly connected to two input nodes. This gate cannot be the output gate for otherwise the function computed by $\mathcal{K}$ would depend on at most two arguments. We claim that the inputs to $\mathtt{v}$ are labeled by two distinct variables p_i and p_j. If this were not the case, the function $\Theta_{\mathcal{K}}(\mathtt{v})$ computed at $\mathtt{v}$ would be either a constant function, a projection, or the negation of a projection. We claim that in any of these cases, we could modify $\mathcal{K}$ into a circuit $\mathcal{K}'$ with one fewer gate that computes $\mathrm{th}_{2,n}$, thereby contradicting the optimality of $\mathcal{K}$. If $\Theta_{\mathcal{K}}(\mathtt{v})$ is a constant, we replace the gate $\mathtt{v}$ by an input labeled by this constant. If $\Theta_{\mathcal{K}}(\mathtt{v})$ is the projection π^n_i, then the gate $\mathtt{v}$ can be replaced by an input labeled p_i. Suppose now that $\Theta_{\mathcal{K}}(\mathtt{v}) = f_{\neg\pi_i^n}$ and the outgoing edges of $\mathtt{v}$ connect $\mathtt{v}$ to the gates $\mathtt{v}_0, \ldots, \mathtt{v}_{l-1}$, labeled by the connective symbols $C_0, \ldots, C_{l-1}$, where $l \geq 1$. The circuit $\mathcal{K}'$ is obtained by replacing the gate $\mathtt{v}$ with an input labeled p_i and changing the labels $C_0, \ldots, C_{l-1}$ to $C'_0, \ldots, C'_{l-1}$ as follows:

(1) if $\mathtt{v}$ is linked to the first input of $\mathtt{v}_i$ but not the second, let C'_i be a connective symbol such that $f_{C'_i}(a, b) = f_{C_i}(\overline{a}, b)$, for $a, b \in \mathbf{Bool}$,
(2) if $\mathtt{v}$ is linked to the second input of $\mathtt{v}_i$ but not the first, let C'_i be a connective symbol such that $f_{C'_i}(a, b) = f_{C_i}(a, \overline{b})$, for $a, b \in \mathbf{Bool}$,
(3) if $\mathtt{v}$ is linked to both inputs of $\mathtt{v}_i$, let C'_i be a connective symbol such that $f_{C'_i}(a, b) = f_{C_i}(\overline{a}, \overline{b})$, for $a, b \in \mathbf{Bool}$.

Next, we claim that there is another gate $\mathtt{v}'$ that is directly connected to an input labeled by one of the variables p_i, p_j that label the inputs directly connected to $\mathtt{v}$. Indeed, otherwise, $\mathrm{th}_{2,n}$ would be decomposable in the way shown to be impossible in Lemma 2.11.39. Without loss of generality, we may assume that it is p_i which labels nodes directly connected to $\mathtt{v}$ and $\mathtt{v}'$.

Let $\widetilde{\mathcal{K}}$ be the circuit obtained from $\mathcal{K}$ by relabeling each input labeled p_i with $\mathbf{F}$ and each input labeled p_k where $k > i$ with p_{k-1}. The function computed by $\widetilde{\mathcal{K}}$ is $\mathrm{th}_{2,n-1}$.

Next, we show that there is a circuit $\widehat{\mathcal{K}}$ equivalent to $\widetilde{\mathcal{K}}$ with two fewer gates.

Suppose initially that $\mathtt{v}'$ is not the output gate. Each of the functions $\Theta_{\widetilde{\mathcal{K}}}(\mathtt{v})$ and $\Theta_{\widetilde{\mathcal{K}}}(\mathtt{v}')$ can be a constant function, a projection, or the negation of a projection. Then, we can eliminate $\mathtt{v}, \mathtt{v}'$ using the same argument as above.

For the case when $\mathtt{v}'$ is the output gate, let $\mathtt{v}''$ be the other node directly connected to $\mathtt{v}'$. Observe first that $\mathtt{v}''$ is a gate since otherwise the function computed by $\mathcal{K}$ would depend on at most two variables. Second, $\mathtt{v}''$ is distinct from $\mathtt{v}$ for the same reason. The gate $\mathtt{v}$ can be eliminated as before. The gate $\mathtt{v}'$ is eliminated by making $\mathtt{v}''$ the new output gate and relabeling $\mathtt{v}''$ with $\widehat{C''}$ where

$$f_{\widehat{C''}}(a, b) = f_{C'}(f_{C''}(a, b), \mathbf{F}),$$

C'' was the previous label of $\mathtt{v}''$, C' is the label of $\mathtt{v}'$, and $a, b \in \mathbf{Bool}$. We assumed here that $\mathtt{v}''$ is connected to the first input of $\mathtt{v}'$.

A similar definition works when v'' is connected to the second input of v'.

We have

$$\begin{aligned}||\mathcal{K}|| &= ||\widehat{\mathcal{K}}|| + 2\\ &\geq 2(n-1) - 3 + 2 \text{ (by the inductive hypothesis)}\\ &= 2n - 3,\end{aligned}$$

which concludes the argument. □

Definition 2.11.41. A *family of truth functions* is a infinite sequence $\mathcal{F} = (f_0, f_1, \ldots, f_n, \ldots)$ of truth functions such that $f_n \in \mathtt{TF}_n$ for $n \in \mathbf{N}$.

Let F be a collection of truth functions. The *F-circuit complexity of* $\mathcal{F}$ is the function $\mathrm{comb}_{F,\mathcal{F}} : \mathbf{N} \longrightarrow \mathbf{N} \cup \{\infty\}$ given by $\mathrm{comb}_{F,\mathcal{F}}(n) = \mathsf{COMB}_F^{\infty}(f_n)$, for $n \in \mathbf{N}$.

A function $g : \mathbf{N} \longrightarrow \mathbf{N} \cup \{\infty\}$ is a *lower bound on the F-circuit complexity of* $\mathcal{F}$ if $g(n) \leq \mathrm{comb}_{F,\mathcal{F}}(n)$ for all $n \in \mathbf{N}$.

A function $h : \mathbf{N} \longrightarrow \mathbf{N} \cup \{\infty\}$ is an *upper bound on the F-circuit complexity of* $\mathcal{F}$ if $g(n) \geq \mathrm{comb}_{F,\mathcal{F}}(n)$ for all $n \in \mathbf{N}$.

▯

Note that if $F, F' \subseteq \mathtt{TF}$ are such that $F \subseteq F'$, then any lower bound on the F'-circuit complexity of a family $\mathcal{F}$ is also a lower bound on the F-circuit complexity of $\mathcal{F}$. Further, any upper bound on the F-circuit complexity of $\mathcal{F}$ is also an upper bound on the F'-circuit complexity of $\mathcal{F}$.

Example 2.11.42. Let $\mathcal{F}_2 = (\mathrm{th}_{2,0}, \mathrm{th}_{2,1}, \mathrm{th}_{2,2}, \ldots)$, where the functions $\mathrm{th}_{2,0}, \mathrm{th}_{2,1}$ are defined by $\mathrm{th}_{2,0} = c^0_{\mathbf{F}}$ and $\mathrm{th}_{2,1} = c^1_{\mathbf{F}}$. It follows from Example 2.11.14 that the function g given by

$$g(n) = \begin{cases} n^2 - n & \text{if } n \geq 2\\ 0 & \text{if } n < 2\end{cases}$$

is an upper bound on the F-circuit complexity of $\mathcal{F}_2$, where $F = \{f_\wedge, f_\vee\}$.

A slightly more complicated argument allows us to prove a stronger result about circuits that compute threshold functions. We start by modifying the circuit $\mathcal{BP}_n$ introduced in Example 2.11.14

by removing the $\vee$-gate of the last comparator. This eliminates the last output gate, thereby leaving us with a circuit $\mathcal{BP}'_n$ which has n inputs and $n-1$ outputs, for $n \geq 1$. We also use the circuits $\mathcal{K}^p_\vee$ which compute the functions $f_p(a_0, \ldots, a_{p-1}) = \bigvee_{0 \leq i \leq p-1} a_i$, for $a_0, \ldots, a_{p-1} \in \mathbf{Bool}$ and $p \geq 1$, using $p-1$ $\vee$-gates. The circuit shown in Figure 2.31 computes the function $\mathrm{th}_{k,n}$, where $1 \leq k \leq n$, because $\mathrm{th}_{k,n}(a_0, \ldots, a_{n-1}) = \mathbf{T}$ if and only if at least one of the least $n-k+1$ members of the sequence $(a_0, \ldots, a_{n-1})$ is $\mathbf{T}$. We leave to the reader the verification that the size of this circuit is $n(2k-1)-k^2-k+1$. For $k = 2$, the size of the circuit is $3n-5$. Thus, we can improve the result of the first paragraph to say that the family $\mathcal{F}_2$ has the function g' as an upper bound on its F-circuit complexity,

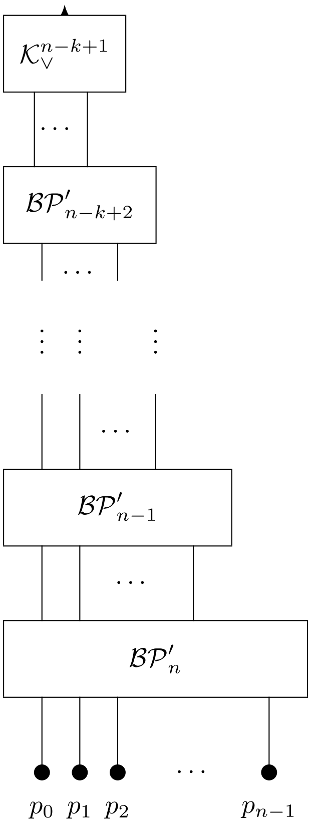


Fig. 2.31. Circuit for the function $\mathrm{th}_{k,n}$.

where

$$g'(n) = \begin{cases} 3n - 5 & \text{if } n \geq 2 \\ 0 & \text{if } n < 2. \end{cases}$$

On the other hand, Theorem 2.11.40 can be restated by saying that the family $\mathcal{F}_2$ has the function $h(n) = 2n-3$ as a lower bound on its $\mathtt{TF}_2$-circuit complexity. Thus, we have asymptotically tight bounds on both the F- and $\mathtt{TF}_2$-circuit complexities of $\mathcal{F}_2$. $\square$

There are explicitly given families of truth functions such that any superpolynomial lower bound on their $\mathtt{TF}_2$-circuit complexity would have profound implications for complexity theory. (Namely, it would settle the well-known $P = NP$? problem in the negative.) Thus, one is compelled to inquire about the existence of any family of truth functions with this property.

By Theorem 2.11.32, for every finite set of truth functions F, there is a number n_0 and a family of truth functions $\mathcal{F} = (f_0, \ldots, f_n, \ldots)$ such that for all $n \geq n_0$, we have $\mathsf{COMB}_F^\infty(f_n) \geq \frac{2^n}{kn}$ for some fixed constant k. (Note that $\frac{2^n}{kn} = \Omega(a^n)$, for any $a \in (1, 2)$, so this family, in fact, has an exponential lower bound on its circuit complexity.) To make the construction of the family $\mathcal{F}$ more specific, for $n \geq n_0$, we could define f_n as the first function f in $\mathtt{TF}_n$ in some fixed effective order of truth functions such that $\mathsf{COMB}_F^\infty(f) \geq \frac{2^n}{kn}$. Nevertheless, this definition does not meet the intuitive requirements of an explicit definition and one might ask what lower bounds could be obtained for explicitly defined families of functions. Here the results are rather poor since the best lower bound obtained so far on the $\mathtt{TF}_2$-circuit complexity of any explicitly defined family of truth functions is linear. In fact, the result obtained in Theorem 2.11.40 is close to the best known lower bound due to Blum [3], namely, $3n - 3$. In view of the difficulty of finding lower bounds for explicitly defined families of functions for general circuits, attention has focused on restricting the circuits in some way and then obtaining lower bounds. Some of the restrictions studied include bounding the delay of the circuits, restricting the circuits to have fan-out 1, or limiting the type of gates to $\wedge$ and $\vee$ gates (which limits the type of function that can be computed to monotonic functions). In each of these cases, higher lower bounds have been obtained, but this has not led, so far, to any progress in the general case.

2.12 Exercises and Supplements

Formulas

(1) Let p be a statement variable, C be a binary connective symbol, and $\varphi, \psi \in$ PLFORM. Prove that

$$SV(p) = \{p\},$$
$$SV((\neg\varphi)) = SV(\varphi),$$
$$SV((\varphi C \psi)) = SV(\varphi) \cup SV(\psi).$$

(2) Show that for every formula φ, $\{\varphi, (\neg\varphi)\} = \{\psi, \overline{\psi}\}$ for some formula ψ. Show also that for every formula φ, there is a formula ψ such that $\{\varphi, \overline{\varphi}\} = \{\psi, (\neg\psi)\}$.

(3) Prove that if $n > 0$, then for the binary representation $\mathtt{k}_n$ of n, we have $|\mathtt{k}_n| = \lfloor \log_2(n) \rfloor + 1$.

(4) Let (α, i), (β, j) be two occurrences of the formulas α, β in a formula φ. Prove that one of the following situations occurs:

(a) $(\alpha, i) \sqsubseteq (\beta, j)$, or
(b) $(\beta, j) \sqsubseteq (\alpha, i)$, or
(c) $|\alpha| + i \leq j$, or
(d) $|\beta| + j \leq i$.

We used here the notation "$\sqsubseteq$" introduced in Exercise 7 of Chapter 1. Note that in the first two cases, one occurrence is part of the other, while in the last two cases, the occurrences are disjoint.
Hint. Use Corollary 2.2.10.

(5) Prove that the standard ordering of the formulas of propositional logic $\preceq$ is a well-ordering on PLFORM.

(6) Prove that for every formula $\varphi \in$ PLFORM we have $|\varphi|_\neg + |\varphi|_\vee + |\varphi|_\wedge + |\varphi|_\to + |\varphi|_\leftrightarrow = |\varphi|_(= |\varphi|_)$. Also, for every proper prefix u of a formula, $|u|_(\geq |u|_\neg + |u|_\vee + |u|_\wedge + |u|_\to + |u|_\leftrightarrow \geq |u|_)$.

(7) Prove that for every formula $\varphi \in$ PLFORM, we have

$$\sum\{|\varphi|_p \mid p \in SV\} - 1 = |\varphi|_\vee + |\varphi|_\wedge + |\varphi|_\to + |\varphi|_\leftrightarrow.$$

(8) Prove that for every formula $\varphi \in \text{PLFORM}$, we have

$$|\text{SUBF}(\varphi)| = |\varphi|_\neg + |\varphi|_\vee + |\varphi|_\wedge + |\varphi|_\rightarrow + |\varphi|_\leftrightarrow \\ + \sum\{|\varphi|_p \mid p \in SV\}.$$

Conclude that $|\text{SUBF}(\varphi)| \leq |\varphi|$.

(9) Let PLFORM' be the set of formulas defined as follows:

1. For every statement variable p, $p \in \text{PLFORM}'$.
2. For every statement variable p, $(\neg p) \in \text{PLFORM}'$.
3. For all formulas $\varphi, \psi \in \text{PLFORM}'$ and every binary connective symbol, $(\varphi C \psi) \in \text{PLFORM}'$.
4. For every formula $\varphi \in \text{PLFORM}'$, $(\neg(\neg\varphi)) \in \text{PLFORM}'$.
5. For all formulas $\varphi, \psi \in \text{PLFORM}'$ and every binary connective symbol, $(\neg(\varphi C \psi)) \in \text{PLFORM}'$.

Observe that Rules 3 and 5 each incorporate four rules, one for each binary connective symbol:

(a) Show that $\text{PLFORM}' = \text{PLFORM}$.
(b) Show that the alternate definition of PLFORM given above satisfies the unique readability condition.

(10) Show that for $n \geq 1$ and $\varphi_0, \ldots, \varphi_{n-1} \in \text{PLFORM}$, we have

$$\texttt{size}\left(\bigwedge_{i=0}^{n-1} \varphi_i\right) = \sum_{i=0}^{n-1} \texttt{size}(\varphi_i) + 3(n-1).$$

We write $f = \Theta(g)$, where $f, g : \mathbf{N} \longrightarrow \mathbf{R}$ if there are positive real numbers c, d and a natural number n_0 such that $n \geq n_0$ implies $0 \leq cg(n) \leq f(n) \leq dg(n)$.

(11) Recall that we defined the formulas θ_i in Example 2.2.8 as $\theta_0 = p_0$ and $\theta_i = (p_{2i-1} \vee p_{2i})$ for $i \geq 1$. Define the formulas $\varphi_n = \bigwedge_{i=0}^{n-1} \theta_i$ for $n \geq 1$. Show that

$$\texttt{size}(\varphi_n) = (2n-1)\lfloor \log_2(2n-2) \rfloor - 2 \cdot 2^{\lfloor \log_2(2n-2) \rfloor} \\ +10n - 6,$$

for $n \geq 2$. Conclude that $\texttt{size}(\varphi_n) = \Theta(n \log n)$.

Hint. This follows from Example 2.2.8, Exercise 10, and Exercise 42 (b) of Section 1.2.4 of [27].

Let $\Gamma = \{\varphi_0, \dots, \varphi_{n-1}\}$ be a finite set of formulas. Define $\texttt{size}(\Gamma)$ as $\sum_{i=1}^{n-1} \texttt{size}(\varphi_i)$. If Δ is a finite set of signed formulas, then $\texttt{size}(\Delta)$ is the sum of the sizes of the signed formulas in Δ.

(12) Let α_i, β_i be the formulas defined by

$$\alpha_i = (\varphi_i \to p_{2i-1}), \quad \beta_i = (\varphi_i \to p_{2i})$$

for $i \geq 1$, where the formulas φ_i are the ones introduced in Exercise 11. Also, define the formulas γ_i for $i \in \mathbf{N}$ as $\gamma_0 = p_0$ and $\gamma_i = (\alpha_i \vee \beta_i)$, for $i \geq 1$. Prove that

$$\texttt{size}(\{\gamma_0, \dots, \gamma_n\}) = \Theta(n^2 \log n).$$

(13) Let $\varphi_0, \dots, \varphi_{n-1} \in$ PLFORM. Show that

$$SV\left(\bigvee_{0\leq i\leq n-1} \varphi_i\right) = \bigcup_{0\leq i\leq n-1} SV(\varphi_i),$$

$$SV\left(\bigwedge_{0\leq i\leq n-1} \varphi_i\right) = \bigcup_{0\leq i\leq n-1} SV(\varphi_i).$$

(14) Let $\varphi_0, \dots, \varphi_{n-1}, \psi_0, \dots, \psi_{m-1}$ be formulas, where $n \geq 1$ and $m \geq 0$. Prove that

$$\begin{aligned} &\text{DISJ}(\text{DISJ}(\varphi_0, \dots, \varphi_{n-1}), \psi_0, \dots, \psi_{m-1}) \\ &\quad = \text{DISJ}(\varphi_0, \dots, \varphi_{n-1}, \psi_0, \dots, \psi_{m-1}). \end{aligned}$$

Show a similar result for CONJ.
Hint. The proof is by induction on m.

We introduce the notion of one formula occurring positively or negatively in another formula.

Define the functions $\mathsf{Pos}, \mathsf{Neg} : \text{PLFORM} \longrightarrow \mathcal{P}(\text{PLFORM})$ by

$$\begin{aligned} \mathsf{Pos}(p) &= \{p\}, \\ \mathsf{Neg}(p) &= \emptyset, \\ \mathsf{Pos}((\neg\varphi)) &= \mathsf{Neg}(\varphi) \cup \{(\neg\varphi)\}, \\ \mathsf{Neg}((\neg\varphi)) &= \mathsf{Pos}(\varphi), \end{aligned}$$

$$\begin{aligned}
\mathsf{Pos}((\varphi C\psi)) &= \mathsf{Pos}(\varphi) \cup \mathsf{Pos}(\psi) \cup \{(\varphi C\psi)\},\\
\mathsf{Neg}((\varphi C\psi)) &= \mathsf{Neg}(\varphi) \cup \mathsf{Neg}(\psi),\\
\mathsf{Pos}((\varphi \to \psi)) &= \mathsf{Neg}(\varphi) \cup \mathsf{Pos}(\psi) \cup \{(\varphi \to \psi)\},\\
\mathsf{Neg}((\varphi \to \psi)) &= \mathsf{Pos}(\varphi) \cup \mathsf{Neg}(\psi),\\
\mathsf{Pos}((\varphi \leftrightarrow \psi)) &= \mathsf{Pos}(\varphi) \cup \mathsf{Neg}(\varphi) \cup \mathsf{Pos}(\psi) \cup \mathsf{Neg}(\psi) \cup \{(\varphi \leftrightarrow \psi)\},\\
\mathsf{Neg}((\varphi \leftrightarrow \psi)) &= \mathsf{Pos}(\varphi) \cup \mathsf{Neg}(\varphi) \cup \mathsf{Pos}(\psi) \cup \mathsf{Neg}(\psi),
\end{aligned}$$

for every $p \in SV$, $C \in \{\vee, \wedge\}$, $\varphi, \psi \in$ PLFORM. A formula θ *occurs positively* in φ if $\theta \in \mathsf{Pos}(\varphi)$; if $\theta \in \mathsf{Neg}(\varphi)$, we say that θ *occurs negatively* in φ.

A formula occurs positively (negatively) in a set of formulas if it occurs positively (negatively) in some formula of the set.

(15) Prove that for all formulas α, β, γ in PLFORM the following statements hold:

(a) if α occurs positively in β and β occurs positively in γ, then α occurs positively in γ,
(b) if α occurs positively in β and β occurs negatively in γ, then α occurs negatively in γ,
(c) if α occurs negatively in β and β occurs positively in γ, then α occurs negatively in γ,
(d) if α occurs negatively in β and β occurs negatively in γ, then α occurs positively in γ.

Hint. Use induction on γ to prove all four statements simultaneously.

The formula ψ *occurs positively* (*negatively*) in the signed formula $\mathbf{T}\varphi$ if ψ occurs positively (negatively) in φ. The formula ψ *occurs positively* (*negatively*) in the signed formula $\mathbf{F}\varphi$ if ψ occurs negatively (positively) in φ.

A formula occurs positively (negatively) in a set of signed formulas if it occurs positively (negatively) in some signed formula of the set.

(16) Identify the formulas that occur positively and the formulas that occur negatively in the signed formula $\mathbf{F}((p \to q) \to r)$.

Truth Assignments

(17) Use Theorem 2.3.5 to recast Definition 2.3.7 in terms of truth valuations.

(18) (a) Prove that for every formula φ, $\overline{\varphi} \equiv (\neg\varphi)$.
(b) Prove that for every formula φ, $\overline{\overline{\varphi}} \equiv \varphi$.
(c) For which formulas φ, do we have $\overline{\overline{\varphi}} = \varphi$?

(19) Let $(\varphi_0, \ldots, \varphi_{n-1})$ and $(\psi_0, \ldots, \psi_{m-1})$ be two nonempty sequences of formulas such that $\{\varphi_0, \ldots, \varphi_{n-1}\} = \{\psi_0, \ldots, \psi_{m-1}\}$. Prove that

$$\bigvee_{0 \le i \le n-1} \varphi_i \equiv \bigvee_{0 \le i \le n-1} \psi_i,$$

$$\bigwedge_{0 \le i \le n-1} \varphi_i \equiv \bigwedge_{0 \le i \le n-1} \psi_i.$$

Hint. Use Theorem 2.3.6.

(20) Let $\varphi = \bigvee_{0 \le i \le n-1} \varphi_i$, where $\varphi_{i_0} \models \varphi_{i_1}$ for some i_0, i_1 with $0 \le i_0, i_1 \le n-1$ and $i_0 \ne i_1$. Prove that for $\varphi' = \bigvee_{0 \le i \le n-1, i \ne i_0} \varphi_i$, we have $\varphi \equiv \varphi'$.

(21) Let $\varphi = \bigwedge_{0 \le i \le n-1} \varphi_i$, where $\varphi_{i_1} \models \varphi_{i_0}$ for some i_0, i_1 with $0 \le i_0, i_1 \le n-1$ and $i_0 \ne i_1$. Prove that for $\varphi' = \bigwedge_{0 \le i \le n-1, i \ne i_0} \varphi_i$, we have $\varphi \equiv \varphi'$.

(22) Let $m_0, \ldots, m_{n-1}$ be n positive natural numbers for some $n \ge 1$ and let $\{\varphi_{ij} \mid 0 \le i \le n-1, 0 \le j \le m_i - 1\}$ be a family of formulas. Prove that

$$\bigvee_{0 \le i \le n-1} \bigvee_{0 \le j \le m_i - 1} \varphi_{ij}$$
$$\equiv (\varphi_{00} \vee \cdots \vee \varphi_{0\, m_0-1} \vee \cdots \vee \varphi_{n-1\, 0} \vee \cdots \vee \varphi_{n-1\, m_{n-1}-1})$$
$$\bigwedge_{0 \le i \le n-1} \bigwedge_{0 \le j \le m_i - 1} \varphi_{ij}$$
$$\equiv (\varphi_{00} \wedge \cdots \wedge \varphi_{0\, m_0-1} \wedge \cdots \wedge \varphi_{n-1\, 0} \wedge \cdots \wedge \varphi_{n-1\, m_{n-1}-1}).$$

For $m \in \mathbf{N}$, let $S_m = \{0, \ldots, m-1\}$. This notation can be extended to a finite sequence $\mathbf{m} = (m_0, \ldots, m_{n-1}) \in \mathrm{Seq}_n(\mathbf{N})$, by defining $S_{\mathbf{m}} = S_{m_0} \times \cdots \times S_{m_{n-1}}$. If $s, t \in S_{\mathbf{m}}$, we write $s \leq t$ if $s = t$ or there is a number k, $0 \leq k \leq n-1$ such that for $0 \leq i \leq k-1$ we have $s(i) = t(i)$ and $s(k) < t(k)$. This relation is a total order known as the lexicographic order on $S_{\mathbf{m}}$. We can list the elements of $S_{\mathbf{m}}$ in this order as $(s_0, \ldots, s_{M-1})$, where $M = m_0 \cdot \cdots \cdot m_{n-1}$.

(23) Let $m_0, \ldots, m_{n-1}$ be n positive natural numbers for some $n \geq 1$, $\mathbf{m} = (m_0, \ldots, m_{n-1})$, and $M = m_0 \cdot \cdots \cdot m_{n-1}$. We use here the notations introduced above. For $0 \leq i \leq n-1$, let $\varphi_{i\,0}, \ldots, \varphi_{i\,m_i-1}$ be m_i formulas. Prove that

$$\bigvee_{i=0}^{n-1} \bigwedge_{j=0}^{m_i-1} \varphi_{ij} \equiv \bigwedge_{k=0}^{M-1} \bigvee_{i=0}^{n-1} \varphi_{i\,s_k(i)},$$

$$\bigwedge_{i=0}^{n-1} \bigvee_{j=0}^{m_i-1} \varphi_{ij} \equiv \bigvee_{k=0}^{M-1} \bigwedge_{i=0}^{n-1} \varphi_{i\,s_k(i)}.$$

Solution. We prove the first logical equivalence. Let $v \in \mathrm{TA}$ be such that $v\left(\bigvee_{i=0}^{n-1} \bigwedge_{j=0}^{m_i-1} \varphi_{ij}\right) = \mathbf{T}$. Then, there exists i_0 such that $v\left(\bigwedge_{j=0}^{m_{i_0}-1} \varphi_{i_0 j}\right) = \mathbf{T}$, which implies $v(\varphi_{i_0 0}) = \cdots = v(\varphi_{i_0\,m_{i_0}-1}) = \mathbf{T}$. For k with $0 \leq k \leq M-1$, $v(\varphi_{i_0\,s_k(i_0)}) = \mathbf{T}$, so $v\left(\bigvee_{i=0}^{n-1} \varphi_{i\,s_k(i)}\right) = \mathbf{T}$, which implies that $v\left(\bigwedge_{k=0}^{M-1} \bigvee_{i=0}^{n-1} \varphi_{i\,s_k(i)}\right) = \mathbf{T}$. On the other hand, if $v\left(\bigvee_{i=0}^{n-1} \bigwedge_{j=0}^{m_i-1} \varphi_{ij}\right) = \mathbf{F}$, then for every i, $0 \leq i \leq n-1$, we have $v\left(\bigwedge_{j=0}^{m_i-1} \varphi_{ij}\right) = \mathbf{F}$, which implies the existence of j_i such that $0 \leq j_i \leq m_i - 1$ and $v(\varphi_{i\,j_i}) = \mathbf{F}$. Consider the sequence $s_k = (j_0, \ldots, j_{n-1})$. We have $v\left(\bigvee_{i=0}^{n-1} \varphi_{i\,s_k(i)}\right) = \mathbf{F}$, so

$$v\left(\bigwedge_{k=0}^{M-1} \bigvee_{i=0}^{n-1} \varphi_{i\,s_k(i)}\right) = \mathbf{F}.$$

(24) Prove that if Γ, Γ' are sets of formulas such that $\Gamma \subseteq \Gamma'$ and φ, ψ are formulas such that $(\varphi \to \psi)$ is a tautology, then $\Gamma \models \varphi$ implies $\Gamma' \models \psi$.

(25) Let φ be a formula of propositional logic. Prove that the following statements are equivalent:

(a) φ is a tautology,
(b) $(\varphi \vee \psi) \equiv \varphi$ for every formula ψ,
(c) $(\varphi \wedge \psi) \equiv \psi$ for every formula ψ.

Also, prove that the following statements are equivalent:

(a) φ is a contradiction,
(b) $(\varphi \vee \psi) \equiv \psi$ for every formula ψ,
(c) $(\varphi \wedge \psi) \equiv \varphi$ for every formula ψ.

(26) Let φ, ψ be two formulas. Prove that the formula $(\varphi \rightarrow \psi)$ is logically equivalent to the formula $((\neg\varphi) \vee \psi)$. Also, prove that $(\varphi \leftrightarrow \psi) \equiv ((\varphi \rightarrow \psi) \wedge (\psi \rightarrow \varphi))$.

(27) Let $\Gamma = \{\varphi_0, \ldots, \varphi_{n-1}\}$ be a set of n formulas. If $\Gamma \models \psi$, show that for every bijection $f : \{0, \ldots, n-1\} \longrightarrow \{0, \ldots, n-1\}$ the formula $(\varphi_{f(0)} \rightarrow (\varphi_{f(1)} \rightarrow (\cdots(\varphi_{f(n-1)} \rightarrow \psi)\cdots)))$ is a tautology.

(28) Let φ and ψ be formulas:

(a) Show that if either φ is a contradiction or ψ is a tautology, then $(\varphi \rightarrow \psi)$ is a tautology.
(b) Show that if $SV(\varphi) \cap SV(\psi) = \emptyset$ and $(\varphi \rightarrow \psi)$ is a tautology, then either φ is a contradiction or ψ is a tautology.
(c) Show that the previous part can fail if $SV(\varphi) \cap SV(\psi) \neq \emptyset$.

(29) (a) Prove that for any formulas $\varphi_0, \varphi_1, \varphi_2$ we have

$$((\varphi_0 \leftrightarrow \varphi_1) \leftrightarrow \varphi_2) \equiv (\varphi_0 \leftrightarrow (\varphi_1 \leftrightarrow \varphi_2))$$

and that

$$(\varphi_0 \leftrightarrow \varphi_1) \equiv (\varphi_1 \leftrightarrow \varphi_0).$$

(b) Prove that if $(\varphi \leftrightarrow \psi)$ is a tautology and $SV(\varphi) \cap SV(\psi) = \emptyset$, then either both φ and ψ are tautologies or both are contradictions.
(c) Show that the previous part can fail if $SV(\varphi) \cap SV(\psi) \neq \emptyset$.
(d) Let BIC : $\text{Seq}^+(\text{PLFORM}) \longrightarrow \text{PLFORM}$ be defined by

$$\begin{aligned} \text{BIC}(\varphi_0) &= \varphi_0, \\ \text{BIC}(\varphi_0, \ldots, \varphi_n) &= (\text{BIC}(\varphi_0, \ldots, \varphi_{n-1}) \leftrightarrow \varphi_n), \\ &\quad \text{for } n > 0. \end{aligned}$$

The formula $\mathrm{BIC}(\varphi_0, \ldots, \varphi_{n-1})$ will be denoted by $(\varphi_0 \leftrightarrow \cdots \leftrightarrow \varphi_{n-1})$.

Show that for every truth assignment v, we have $v(\varphi_0 \leftrightarrow \cdots \leftrightarrow \varphi_{n-1}) = \mathbf{T}$ if and only if $|\{i \mid 0 \le i \le n-1 \text{ and } v(\varphi_i) = \mathbf{T}\}|$ has the same parity as n.

(e) Prove that for all $n \geq 1$ and all formulas $\varphi_0, \ldots, \varphi_{n-1}, \psi$, we have

$$(\varphi_0 \leftrightarrow \cdots \leftrightarrow \varphi_{n-1}) \leftrightarrow \psi \equiv (\varphi_0 \leftrightarrow \cdots \leftrightarrow (\varphi_{n-1} \leftrightarrow \psi)).$$

(f) Let φ be a formula whose only connective symbol is $\leftrightarrow$ and let p be a statement variable such that p occurs in φ and $\varphi \neq p$. Prove that there is a formula φ' such that $\varphi \equiv (\varphi' \leftrightarrow p)$, the only connective symbol in φ' is $\leftrightarrow$, and for all variables q, $|(\varphi' \leftrightarrow p)|_q = |\varphi|_q$.

Hint. The Replacement Theorem of Section 2.6 is useful in the argument.

(g) Let φ be a formula whose single connective symbol is $\leftrightarrow$. If $SV(\varphi) = \{p_{i_0}, \ldots, p_{i_{n-1}}\}$, show that $\varphi \equiv \psi$, where $\psi = (\varphi_0 \leftrightarrow \cdots \leftrightarrow \varphi_{n-1})$, $\varphi_j = (p_{i_j} \leftrightarrow \cdots \leftrightarrow p_{i_j})$, and $|\varphi_j|_{p_{i_j}} = |\varphi|_{p_{i_j}}$.

(h) Prove that a formula whose only connective symbol is $\leftrightarrow$ is a tautology if and only if every variable occurs an even number of times.

(30) Prove that for every set of formulas Γ, if Γ is unsatisfiable, then $\Gamma \models \varphi$ for every $\varphi \in \mathrm{PLFORM}$. Also, show that if $\Gamma \models \varphi$ and φ is a contradiction, then Γ is unsatisfiable.

(31) A formula φ *depends on a variable* p if there are truth assignments v_0, v_1 such that $v_0(q) = v_1(q)$ for all $q \neq p$ and $v_0(\varphi) \neq v_1(\varphi)$.

Prove that if φ depends on p and $\varphi \equiv \psi$, then $p \in SV(\psi)$.

Hint. Use the Agreement Theorem.

(32) Let φ be a formula and let $V_0(\varphi), V_1(\varphi)$ be the subsets of the set TA of truth assignments defined by

$$V_0(\varphi) = \{v \in \mathrm{TA} | v(\varphi) = \mathbf{F}\},$$

$$V_1(\varphi) = \{v \in \mathrm{TA} | v(\varphi) = \mathbf{T}\}.$$

(a) Show that $\varphi \models \psi$ if and only if $V_1(\varphi) \subseteq V_1(\psi)$ and that the following three conditions are equivalent:

(i) $\varphi \equiv \psi$,
(ii) $V_1(\varphi) = V_1(\psi)$,
(iii) $V_0(\varphi) = V_0(\psi)$.

(b) Prove that for any formulas $\varphi, \psi \in$ PLFORM we have

$$\begin{aligned}
V_1(\varphi) &= \mathrm{TA} - V_0(\varphi),\\
V_1((\neg\varphi)) &= V_0(\varphi),\\
V_1((\varphi \vee \psi)) &= V_1(\varphi) \cup V_1(\psi),\\
V_1((\varphi \wedge \psi)) &= V_1(\varphi) \cap V_1(\psi),\\
V_1((\varphi \rightarrow \psi)) &= V_0(\varphi) \cup V_1(\psi),\\
V_1((\varphi \leftrightarrow \psi)) &= (V_1(\varphi) \cap V_1(\psi)) \cup (V_0(\varphi) \cap V_0(\psi)).
\end{aligned}$$

(c) Give another proof of Theorem 2.3.14 using Parts (a) and (b).

(33) Let Γ be a set of formulas. Prove that Γ is unsatisfiable if and only if $\mathrm{TA} = \bigcup\{V_1((\neg\varphi)) | \varphi \in \Gamma\}$.
Solution. Suppose that $\mathrm{TA} = \bigcup\{V_1((\neg\varphi)) | \varphi \in \Gamma\}$. For every $v \in \mathrm{TA}$, there is a formula $\varphi \in \Gamma$ such that $v((\neg\varphi)) = \mathbf{T}$, or, equivalently, $v(\varphi) = \mathbf{F}$. This proves that Γ is not satisfiable. Conversely, if Γ is not satisfiable, for every truth assignment v, there is a formula φ in Γ such that $v(\varphi) = \mathbf{F}$, or $v((\neg\varphi)) = \mathbf{T}$ which means that $v \in V_1((\neg\varphi))$. This shows that $\mathrm{TA} = \bigcup\{V_1((\neg\varphi)) | \varphi \in \Gamma\}$.

(34) Let $\Delta_n = \{\mathbf{T}\gamma_0, \ldots, \mathbf{T}\gamma_n, \mathbf{F}p_{2n-1}, \mathbf{F}p_{2n}\}$, where the formulas γ_i were introduced in Exercise 12 and $n \geq 1$. Prove that for every $n \geq 1$, Δ_n is an unsatisfiable set of signed formulas.
Solution. Suppose that v were a truth assignment that satisfies Δ_n. We prove by strong induction on i with $0 \leq i \leq n$ that $v(\theta_i) = \mathbf{T}$, where $\theta_0 = p_0$ and $\theta_i = (p_{2i-1} \vee p_{2i})$ (see Exercise 11). The basis step, $i = 0$, is obvious because $\gamma_0 = p_0$. Suppose that $0 \leq i < n$ and $v(\theta_j) = \mathbf{T}$ for $0 \leq j \leq i$. Since $\varphi_{i+1} = \bigwedge_{j=0}^{i} \theta_j$, we have $v(\varphi_{i+1}) = \mathbf{T}$. We also have $v(\gamma_{i+1}) = v(((\varphi_{i+1} \rightarrow p_{2i+1}) \vee (\varphi_{i+1} \rightarrow p_{2i+2}))) = \mathbf{T}$ and this implies $v((p_{2i+1} \vee p_{2i+2})) = v(\theta_{i+1}) = \mathbf{T}$, completing the induction. In particular, $v((p_{2n-1} \vee p_{2n})) = \mathbf{T}$ and this conflicts with $v(\mathbf{F}p_{2n-1}) = v(\mathbf{F}p_{2n}) = \mathbf{T}$.

The Compactness Theorem

(35) Show that finite satisfiability is a property of finite character.

(36) Let Γ be a finitely satisfiable set of formulas and let φ be a formula. Prove that at least one of $\Gamma \cup \{\varphi\}$ and $\Gamma \cup \{(\neg\varphi)\}$ is finitely satisfiable.

Solution. Suppose that neither $\Gamma \cup \{\varphi\}$ nor $\Gamma \cup \{(\neg\varphi)\}$ is finitely satisfiable. Then, there are two finite subsets Γ' and Γ'' of Γ such that $\Gamma' \cup \{\varphi\}$ and $\Gamma'' \cup \{(\neg\varphi)\}$ are unsatisfiable. We claim that $\Gamma' \cup \Gamma''$ is unsatisfiable. Indeed, if v were a truth assignment that satisfies $\Gamma' \cup \Gamma''$, then v would satisfy one of the sets $\Gamma' \cup \{\varphi\}$ or $\Gamma'' \cup \{(\neg\varphi)\}$, which is impossible. This contradicts the finite satisfiability of Γ.

(37) Prove that if Γ is a maximal finitely satisfiable set, then, for every formula φ, $(\neg\varphi) \in \Gamma$ if and only if $\varphi \notin \Gamma$.

(38) Let Γ be a set of formulas such that every truth assignment satisfies at least one formula in Γ. Prove that there is a disjunction of formulas of Γ that is a tautology.

Hint. Consider the set $\{(\neg\varphi) \mid \varphi \in \Gamma\}$ and apply the Compactness Theorem.

(39) Let (M, μ) be a partially ordered set (see Chapter 3 of [13] for definitions of terms used here).

(a) Prove that if M is finite, then there is a total order μ' on M such that $\mu \subseteq \mu'$.

(b) Using the Compactness Theorem and Part (a) prove that if M is countable, then there is a total order μ' on M such that $\mu \subseteq \mu'$.

Solution. The argument for Part (a) is by induction on $n = |M|$. The basis step, $n = 1$, is trivial. Suppose that the statement holds for n and that $|M| = n + 1$. Since (M, μ) is a finite poset, there is a minimal element $z \in M$. Let $P = M - \{z\}$ and $\pi = \mu \cap (P \times P)$. Then, (P, π) is a poset and $|P| = n$, so, by inductive hypothesis, there is a total order π' on P such that $\pi \subseteq \pi'$. Let $\mu' = \pi' \cup \{(z, x) \mid x \in M\}$. (This argument is essentially the proof of the existence of a topological sort of a directed acyclic graph from Chapter 2 of [13].)

For the second part, let M be countable. Since $M \times M$ is countable, we can choose for every pair $(a, b) \in M \times M$ a distinct variable p_{ab}. Let Γ be the union of the following sets of formulas:

(I) $\{p_{ab} \mid (a,b) \in \mu\}$,
(II) $\{(p_{ab} \vee p_{ba}) \mid a,b \in M\}$,
(III) $\{((p_{ab} \wedge p_{bc}) \to p_{ac}) \mid a,b,c \in M\}$,
(IV) $\{(\neg(p_{ab} \wedge p_{ba})) \mid a,b \in M$ and $a \neq b\}$.

We show that Γ is finitely satisfiable. Let Γ_0 be a finite subset of Γ and let $V_0 = SV(\Gamma_0)$ be the set of variables that occur in Γ_0. Denote by M_0 the finite set of elements of M that occur in subscripts of variables in V_0 and by μ_0 the partial order $\mu \cap (M_0 \times M_0)$. By Part (a), there is a total order μ'_0 on M_0 such that $\mu_0 \subseteq \mu'_0$. Define a truth assignment v by $v(p) = \mathbf{T}$ if and only if $p = p_{ab}$, where $(a,b) \in \mu'_0$. Let φ be a formula in Γ_0. If φ belongs to the first group, then $\varphi = p_{ab}$, where $(a,b) \in \mu$ and $a,b \in M_0$. Thus, $(a,b) \in \mu_0 \subseteq \mu'_0$, so $v(p_{ab}) = \mathbf{T}$. Let now φ be in the second group, say $\varphi = (p_{ab} \vee p_{ba})$, where $a,b \in M_0$. Since μ'_0 is a total order on M_0, we have either $(a,b) \in \mu'_0$ or $(b,a) \in \mu'_0$, so either $v(p_{ab}) = \mathbf{T}$ or $v(p_{ba}) = \mathbf{T}$, which implies $v(\varphi) = \mathbf{T}$. We leave to the reader to prove that $v(\varphi) = \mathbf{T}$ when φ belongs to the last two groups of formulas.

By the Compactness Theorem, Γ is satisfiable, so there is a truth assignment w that satisfies Γ. Define $\mu' = \{(a,b) \mid w(p_{ab}) = \mathbf{T}\}$. It is easy to see that μ' is a total order such that $\mu \subseteq \mu'$.

(40) This exercise gives another proof of the Compactness Theorem.

Let Γ be a finitely satisfiable set of formulas. Construct a sequence of sets $\Gamma = \Gamma_0, \Gamma_1, \ldots$ as follows:

$$\Gamma_{n+1} = \begin{cases} \Gamma_n \cup \{p_n\} & \text{if } \Gamma_n \cup \{p_n\} \text{ is finitely satisfiable} \\ \Gamma_n \cup \{(\neg p_n)\} & \text{otherwise.} \end{cases}$$

(a) Prove that each Γ_n is finitely satisfiable.
(b) Define a truth assignment v by

$$v(p) = \begin{cases} \mathbf{T} & \text{if } p \in \bigcup_{n \in \mathbf{N}} \Gamma_n \\ \mathbf{F} & \text{otherwise.} \end{cases}$$

Prove that v satisfies Γ.

Hint. To prove Part (b), let $q_0, \ldots, q_{m-1}$ be the variables that occur in a formula $\varphi \in \Gamma$. Let ℓ_i be q_i if $q_i \in \bigcup_{n \in \mathbf{N}} \Gamma_n$ and be $(\neg q_i)$ otherwise, for $0 \leq i \leq m-1$. Note that there is a truth assignment w that satisfies the set $\{\varphi, \ell_0, \ldots, \ell_{m-1}\}$ because this set is included in some set Γ_k. Compare w and v.

(41) The proof given in the text for the Compactness Theorem is similar to the proof of König's lemma we gave. In this exercise, we show how the connection can be made more explicit.
If $q \in \mathrm{Seq}(\{0,1\})$, we define the partial truth assignment v_q by

$$v_q(p_i) = \begin{cases} \mathbf{T} & \text{if } q_i = 1 \\ \mathbf{F} & \text{otherwise,} \end{cases}$$

for $0 \leq i \leq |q| - 1$.
Let Γ be a finitely satisfiable set of formulas and let Γ_n be as defined in the proof of the Compactness Theorem. Define the set $D = \{q \in \mathrm{Seq}(\{0,1\}) \mid v_q \text{ satisfies } \Gamma_{|q|}\}$.

(a) Show that D is an infinite tree semi-domain.
(b) By applying König's lemma for tree semi-domains (see Exercise 92 of Chapter 1) to D, conclude that D has an infinite branch. Show that this infinite branch defines a truth assignment that satisfies Γ.

Let $V = \{v_i \mid i \in I\}$ be a set of truth assignments indexed by some set I. If $\mathcal{F}$ is a filter on the set I, $\mathcal{F} \in \mathtt{FIL}(I)$ (see page 67), then let $v_{V,\mathcal{F}}$ be the truth assignment given by

$$v_{V,\mathcal{F}}(p) = \mathbf{T} \quad \text{if and only if } \{i \in I \mid v_i(p) = \mathbf{T}\} \in \mathcal{F}.$$

The truth assignment $v_{V,\mathcal{F}}$ is called *the reduced product* of the indexed set V relative to the filter $\mathcal{F}$. If $\mathcal{U}$ is an ultrafilter, then we refer to the truth assignment $v_{V,\mathcal{U}}$ as an *ultraproduct* of the indexed set V of truth assignments relative to the ultrafilter $\mathcal{U}$.

(42) Let $V = \{v_i \mid i \in I\}$ be a set of truth assignments indexed by some set I and let $\mathcal{U}$ be an ultrafilter on I. Prove that for every formula φ

$$v_{V,\mathcal{U}}(\varphi) = \mathbf{T} \quad \text{if and only if } \{i \in I \mid v_i(\varphi) = \mathbf{T}\} \in \mathcal{U}.$$

Solution. The argument is by induction on formulas. The basis step, when φ is a statement variable, is immediate. There are several inductive steps. Suppose that the equality holds for the formulas α, β. If $\varphi = (\alpha \wedge \beta)$, then the following statements are equivalent:

- $v_{V,\mathcal{U}}(\varphi) = \mathbf{T}$.
- $v_{V,\mathcal{U}}(\alpha) = \mathbf{T}$ and $v_{V,\mathcal{U}}(\beta) = \mathbf{T}$.
- $\{i \in I \mid v_i(\alpha) = \mathbf{T}\} \in \mathcal{U}$ and $\{i \in I \mid v_i(\beta) = \mathbf{T}\} \in \mathcal{U}$.
- $\{i \in I \mid v_i(\alpha) = \mathbf{T}$ and $v_i(\beta) = \mathbf{T}\} \in \mathcal{U}$.
- $\{i \in I \mid v_i(\varphi) = \mathbf{T}\} \in \mathcal{U}$.

The equivalence of the third and fourth statements follows from Exercise 26 of Chapter 1.
Assume now that $\varphi = (\neg\alpha)$. The following statements are equivalent:

- $v_{V,\mathcal{U}}(\varphi) = \mathbf{T}$.
- $v_{V,\mathcal{U}}(\alpha) = \mathbf{F}$.
- $\{i \in I \mid v_i(\alpha) = \mathbf{T}\} \notin \mathcal{U}$.
- $\{i \in I \mid v_i(\alpha) = \mathbf{F}\} \in \mathcal{U}$.
- $\{i \in I \mid v_i((\neg\alpha)) = \mathbf{T}\} \in \mathcal{U}$.
- $\{i \in I \mid v_i(\varphi) = \mathbf{T}\} \in \mathcal{U}$.

The equivalence of the third and fourth statements follows from Exercise 33 of Chapter 1.
If $\varphi = (\alpha \to \beta)$, then we have the following equivalent statements:

- $v_{V,\mathcal{U}}(\varphi) = \mathbf{T}$.
- $v_{V,\mathcal{U}}(\alpha) = \mathbf{F}$ or $v_{V,\mathcal{U}}(\beta) = \mathbf{T}$.
- $\{i \in I \mid v_i(\alpha) = \mathbf{T}\} \notin \mathcal{U}$ or $\{i \in I \mid v_i(\beta) = \mathbf{T}\} \in \mathcal{U}$.
- $\{i \in I \mid v_i(\alpha) = \mathbf{F}\} \in \mathcal{U}$ or $\{i \in I \mid v_i(\beta) = \mathbf{T}\} \in \mathcal{U}$.
- $\{i \in I \mid v_i(\alpha) = \mathbf{F}$ or $v_i(\beta) = \mathbf{T}\} \in \mathcal{U}$.
- $\{i \in I \mid v_i((\alpha \to \beta)) = \mathbf{T}\} \in \mathcal{U}$.
- $\{i \in I \mid v_i(\varphi) = \mathbf{T}\} \in \mathcal{U}$.

The equivalence of the third and fourth statements follows from Exercise 33 of Chapter 1, while the equivalence of the fourth and fifth statements follows from Part (a) of Exercise 34 of the same chapter.
We leave to the reader the case when $\varphi = (\alpha \vee \beta)$ and $\varphi = (\alpha \leftrightarrow \beta)$.

Let Γ be a set of formulas that is finitely satisfiable. For $\Gamma_0 \in \mathcal{P}_{\text{fin}}(\Gamma)$, where $\mathcal{P}_{\text{fin}}(\Gamma)$ is the set of all finite subsets of Γ, let v_{Γ_0}

be a truth assignment that satisfies Γ_0. Using the notation of Supplement 25 of Chapter 1, $\mathcal{I}(\varphi)$ is the collection of all finite subsets of Γ that contain φ, where $\varphi \in \Gamma$; we define the collection $\mathcal{C}_\Gamma = \{\mathcal{I}(\varphi) \mid \varphi \in \Gamma\}$.

(43) Let Γ be a set of formulas that is finitely satisfiable.

(a) Prove, using Zorn's lemma, that there exists an ultrafilter $\mathcal{U}_\Gamma$ on $\mathcal{P}_{\text{fin}}(\Gamma)$ including $\mathcal{C}_\Gamma$, that is, for every $\varphi \in \Gamma, \mathcal{I}(\varphi) \in \mathcal{U}_\Gamma$. Show that for every $\varphi \in \Gamma$, $\{\Gamma_0 \in \mathcal{P}_{\text{fin}}(\Gamma) \mid v_{\Gamma_0}(\varphi) = \mathbf{T}\} \in \mathcal{U}_\Gamma$.

(b) Prove the following version of the Compactness Theorem: the ultraproduct of the collection of truth assignments $\{v_{\Gamma_0} \mid \Gamma_0 \in \mathcal{P}_{\text{fin}}(\Gamma)\}$ with respect to the ultrafilter $\mathcal{U}_\Gamma$ satisfies Γ.

Hint. For Part (a), use Supplement 25(b) and Exercises 31 and 37 of Chapter 1.

A subcollection $\mathcal{B}$ of a topology $\mathcal{T}$ is a base for $\mathcal{T}$ if every set $L \in \mathcal{T}$ can be written as a union of some sets in $\mathcal{B}$. The topology $\mathcal{T}$ is said to be *generated by the base* $\mathcal{B}$.

(44) (a) Prove that if $\mathcal{B}$ is a base for a topology $\mathcal{T}$ on a set M, then $\mathcal{T} = \{\bigcup \mathcal{C} \mid \mathcal{C} \subseteq \mathcal{B}\}$. Conclude that a collection $\mathcal{B} \subseteq \mathcal{P}(\mathcal{M})$ can be a base for at most one topology on M.

(b) Prove that a collection of subsets $\mathcal{B}$ of M is a base for a topology if and only if $M = \bigcup \mathcal{B}$ and, for every $P, Q \in \mathcal{B}$ and $x \in P \cap Q$, there is $R \in \mathcal{B}$ such that $x \in R \subseteq P \cap Q$.

(c) Prove that the collection $\mathcal{B}_l = \{V_1(\varphi) | \varphi \in \text{PLFORM}\}$ is a base for a topology on the set TA of truth assignments. Moreover, in this topology, for every $\varphi \in \text{PLFORM}$, $V_1(\varphi)$ is both open and closed.

A topological space $(M, \mathcal{T})$ is *compact* if every collection $\mathcal{C} \subseteq \mathcal{T}$ such that $M = \bigcup \mathcal{C}$ contains a finite subcollection $\mathcal{D}$ such that $M = \bigcup \mathcal{D}$.

(45) (a) Let $(M, \mathcal{T})$ be a topological space and let $\mathcal{B}$ be a base for $\mathcal{T}$. Prove that $(M, \mathcal{T})$ is compact if and only if for every collection $\mathcal{E} \subseteq \mathcal{B}$ such that $M = \bigcup \mathcal{E}$, $\mathcal{E}$ contains a finite subcollection $\mathcal{D}$ such that $M = \bigcup \mathcal{D}$.

(b) Let $\mathcal{T}_l$ be the topology on TA generated by the base $\mathcal{B}_l$. Prove that the Compactness Theorem is equivalent to the assertion that $(\text{TA}, \mathcal{T}_l)$ is a compact topological space.

Normal Forms

(46) Prove that if μ is a conjunction of variables, then $\mu \equiv \mu_S$ for some finite, nonempty set of variables S. Prove a similar result for disjunctions of negated variables.

(47) Let φ be a formula in disjunctive normal form that contains no negative literals. Show how to obtain a formula ψ such that $\psi \equiv \varphi$, $SV(\psi) \subseteq SV(\varphi)$, and $\psi = \bigvee_{0 \leq i \leq m-1} \mu_{S_i}$, where for $i \neq j$, $S_i \not\subseteq S_j$.

(48) Let $S = \{p_{i_0}, \ldots, p_{i_{n-1}}\}$ be a set of statement variables such that $n > 0$ and $i_0 < \cdots < i_{n-1}$. Show that

$$|\bigcup\{\text{MINTRM}(T) \mid \emptyset \neq T \subseteq S\}| = 3^n - 1.$$

(49) Prove that for every formula φ and $b \in \mathbf{Bool}$, we have $\varphi^{f_\neg(b)} \equiv (\neg\varphi^b)$.

The set NNF of formulas in *negation normal form* is given by the following inductive definition:

- For every $p \in SV$, both p and $(\neg p)$ are in NNF.
- If $\varphi, \psi \in$ NNF, then $(\varphi \vee \psi)$ and $(\varphi \wedge \psi)$ belong to NNF.

Observe that formulas in either disjunctive normal form or conjunctive normal form are in negation normal form. Therefore, for every formula $\varphi \in$ PLFORM, there is a logically equivalent formula in negation normal form.

(50) (a) Prove that if $(\varphi \vee \psi)$ or $(\varphi \wedge \psi)$ belongs to NNF, then both φ and ψ belong to NNF.

(b) Prove that if $n > 0$ and $\varphi_i \in$ NNF for $0 \leq i \leq n-1$, then both $\bigvee_{i=0}^{n-1} \varphi_i$ and $\bigwedge_{i=0}^{n-1} \varphi_i$ belong to NNF.

(51) Define a mapping $\sim$: NNF $\longrightarrow$ NNF such that $\sim(\varphi) \equiv (\neg\varphi)$ for every $\varphi \in$ NNF.

Hint. Define $\sim$ by

$$\begin{aligned}
\sim(p) &= (\neg p),\\
\sim((\neg p)) &= p,\\
\sim((\varphi \vee \psi)) &= (\sim(\varphi) \wedge \sim(\psi)),\\
\sim((\varphi \wedge \psi)) &= (\sim(\varphi) \vee \sim(\psi)).
\end{aligned}$$

(52) Prove that $\sim(\sim(\varphi)) = \varphi$ for every $\varphi \in \mathrm{NNF}$, where $\sim$ is as in Exercise 51.

(53) Using the inductive definition of formulas discussed in Exercise 9 and the function $\sim$ introduced in Exercise 51, give an inductive definition of a function $\Phi : \mathrm{PLFORM} \longrightarrow \mathrm{PLFORM}$ such that $\Phi(\varphi)$ is in negation normal form, and $\Phi(\varphi) \equiv \varphi$ for every formula φ.
Hint. For instance, $\Phi((\varphi \to \psi)) = (\sim(\Phi(\varphi)) \vee \Phi(\psi))$ and $\Phi((\neg(\varphi \to \psi))) = (\Phi(\varphi) \wedge \sim(\Phi(\psi)))$.

(54) Let φ be the formula

$$(((\neg p_0) \wedge p_3) \vee (p_1 \wedge p_2 \wedge p_3) \vee (p_0 \wedge p_3) \vee (\neg p_1)).$$

(a) Apply the Quine–McCluskey algorithms to obtain the set of prime implicants of φ and the collection of all its minimal covers.
(b) Use a Karnaugh map to compute the same sets.

(55) (a) Let φ be a formula with four variables. Using a Karnaugh map, prove that if φ has more than eight minterms, then there is a prime implicant of φ that is not a minterm of φ.
(b) Give a simple formula with four variables such that the formula has eight minterms, all of which are prime implicants.

Let $\mathcal{K}$ be a collection of nonempty subsets of a set U. A *hitting set* for $\mathcal{K}$ is a subset H of U such that $H \cap K \neq \emptyset$ for every $K \in \mathcal{K}$. A hitting set H for $\mathcal{K}$ is *minimal* if no proper subset of H is a hitting set.

(56) Explain how Algorithm 2.5.42 can be viewed in a more general context as an algorithm to find all the minimal hitting sets for a collection of subsets of a finite set.

Let $\mathcal{K}$ be a collection of subsets of a set U. Denote by $\mathrm{MIN}(\mathcal{K})$ the collection of all minimal elements of $\mathcal{K}$.
For $P \subseteq U$, let $\mathcal{K} * P = \{K \cup \{p\} \mid K \in \mathcal{K}, p \in P\}$.

(57) Let $\mathcal{K} = \{P_0, \ldots, P_{n-1}\}$ be a collection of subsets of a set U. Define the sequence of collections of sets $\mathcal{K}_0, \ldots, \mathcal{K}_n$ by

$$\mathcal{K}_0 = \{\emptyset\},$$
$$\mathcal{K}_{i+1} = \mathrm{MIN}(\mathcal{K}_i * P_i).$$

Prove that $\mathcal{K}_n$ consists of all minimal hitting sets for $\mathcal{K}$.

(58) Use Exercise 57 to obtain a variant of Algorithm 2.5.42. Apply this algorithm to the formula considered in Example 2.5.43 to obtain all its minimal covers.

(59) Let φ be a formula such that $SV(\varphi) = \{p_0, p_1, p_2\}$ and for every truth assignment v, $v(\varphi) = \mathbf{T}$ if and only if at least two of $v(p_0), v(p_1), v(p_2)$ are $\mathbf{T}$. Prove that there is no formula logically equivalent to φ that is a Horn formula.

(60) Find the least n such that the number of formulas in disjunctive normal form consisting of distinct minterms over some subset of a set S of statement variables with $|S| = n$ exceeds one U.S. quadrillion (10^{15}). (Here we identify two disjunctive normal forms if they consist of the same minterms in different orders.)

The problem of determining whether a formula in conjunctive normal form is satisfiable is believed to be an algorithmically hard problem. The next three items will show that this problem can be efficiently reduced to other problems, thereby showing that these problems are also probably algorithmically hard.

(61) Let φ be a nonempty disjunction of n distinct literals, $\varphi = (\ell_0 \vee \cdots \vee \ell_{n-1})$. Consider the functions $\mathtt{v} : \mathbf{N} \longrightarrow \mathbf{N}$ and $\mathtt{conj} : \mathbf{N} \longrightarrow \mathbf{N}$ given by

$$\mathtt{v}(n) = \begin{cases} 3-n & \text{if } n \leq 3 \\ n-3 & \text{if } n > 3 \end{cases} \quad \text{and} \quad \mathtt{conj}(n) = \begin{cases} 4 & \text{if } n = 1 \\ 2 & \text{if } n = 2 \\ 1 & \text{if } n = 3 \\ n-2 & \text{if } n > 3. \end{cases}$$

Let $q_0, \ldots, q_{\mathtt{v}(n)-1}$ be $\mathtt{v}(n)$ variables that do not occur in $SV(\varphi)$. Define the formula ψ in conjunctive normal form (whose conjuncts consist of three distinct literals each) as follows:

- If $n = 1$, then

$$\psi = (\ell_0 \vee q_0 \vee q_1) \wedge (\ell_0 \vee q_0 \vee \overline{q_1}) \\ \wedge (\ell_0 \vee \overline{q_0} \vee q_1) \wedge (\ell_0 \vee \overline{q_0} \vee \overline{q_1}).$$

- If $n = 2$, then

$$\psi = (\ell_0 \vee \ell_1 \vee q_0) \wedge (\ell_0 \vee \ell_1 \vee \overline{q_0}).$$

- If $n = 3$, $\psi = \varphi$.

- If $n \geq 4$, define

$$\psi = (\ell_0 \vee \ell_1 \vee q_0) \wedge (\overline{q_0} \vee \ell_2 \vee q_1)$$
$$\wedge(\overline{q_1} \vee \ell_3 \vee q_2) \wedge \cdots \wedge (\overline{q_{n-4}} \vee \ell_{n-2} \vee \ell_{n-1}).$$

(A formula in conjunctive normal form whose conjuncts each contain three literals is said to be in *3-cnf.*) Prove that the following conditions are satisfied by ψ:

(a) every truth assignment $v \in \mathrm{TA}_{SV(\varphi)}$ such that $v(\varphi) = \mathbf{T}$ can be extended to a truth assignment $v' \in \mathrm{TA}_{SV(\psi)}$ such that $v'(\psi) = \mathbf{T}$,

(b) for every truth assignment v' such that $v'(\psi) = \mathbf{T}$ there exists a literal ℓ that occurs in φ such that $v'(\ell) = \mathbf{T}$,

(c) the length of ψ is no greater than $19\mathtt{conj}(n)$.

Solution. Observe that in all cases $|SV(\psi) - SV(\varphi)| = \mathtt{v}(n)$ for $n \geq 1$ and that ψ is a formula in conjunctive normal form that consists of $\mathtt{conj}(n)$ conjuncts. Let $v \in \mathrm{TA}_{SV(\varphi)}$ be such that $v(\varphi) = \mathbf{T}$. If $n \leq 3$, then for any extension v' of v to $SV(\psi)$, we have $v'(\psi) = \mathbf{T}$. Assume, therefore, that φ is a disjunction of $n \geq 4$ literals and let v be a truth assignment such that $v(\varphi) = \mathbf{T}$. Let $j = \min\{i | 0 \leq i \leq n-1, v(\ell_i) = \mathbf{T}\}$. If $j \in \{0, 1\}$, then we define $v'(q_i) = \mathbf{F}$ for $0 \leq i \leq n-4$. If $j \in \{n-2, n-1\}$, then let $v'(q_i) = \mathbf{T}$ for $0 \leq i \leq n-4$. For the remaining case, that is, when $2 \leq j \leq n-3$, define

$$v'(q_i) = \begin{cases} \mathbf{T} & \text{if } 0 \leq i \leq j-2 \\ \mathbf{F} & \text{if } j-1 \leq i \leq n-4. \end{cases}$$

It is easy to see that $v'(\psi) = \mathbf{T}$.
We leave to the reader to prove that for every $v' \in \mathrm{TA}_{SV(\psi)}$ such that $v'(\psi) = \mathbf{T}$ there exists a literal ℓ in φ such that $v'(\ell) = \mathbf{T}$. Also, the evaluation of an upper bound on the length of ψ is left to the reader.

(62) Let $\varphi = \bigwedge_{i=0}^{m} \varphi_i$ be a formula in conjunctive normal form, where each conjunct φ_i contains n_i disjuncts for $0 \leq i \leq m$. Prove that there exists a formula ψ in 3-cnf such that the following conditions are satisfied:

(a) $SV(\varphi) \subseteq SV(\psi)$ and $SV(\psi) - SV(\varphi)$ contains $\sum_{i=0}^{m} \mathtt{v}(n_i)$ variables,

(b) ψ contains $\sum_{i=0}^{m} \mathtt{conj}(n_i)$ conjuncts, and
(c) ψ is satisfiable if and only if φ is satisfiable.

Hint. Use Exercise 61.

(63) Let $\varphi = \bigwedge_{0 \le i \le n-1} \alpha_i$ be a formula in conjunctive normal form. Assume that $\alpha_i = (\ell_{i0} \vee \cdots \vee \ell_{ik_i-1})$, where $\ell_{i0}, \ldots, \ell_{ik_i-1}$ are literals for $0 \le i \le n-1$. Consider the graph G_φ whose set of vertices consists of pairs of the form (α_i, ℓ_{ij}). An edge $((\alpha_i, \ell_{ij}), (\alpha_h, \ell_{hk}))$ exists in G_φ if and only if $i \neq h$ and $\ell_{ij} \neq \overline{\ell_{hk}}$. Prove that φ is satisfiable if and only if the graph G_φ contains a set of n vertices such that each pair of such vertices is connected by an edge.[10]

Solution. Suppose that φ is satisfiable and let v be a truth assignment such that $v(\varphi) = \mathbf{T}$. Then, $v(\alpha_i) = \mathbf{T}$ for $0 \le i \le n-1$. Each conjunct α_i must contain a literal ℓ_{im_i} such that $v(\ell_{im_i}) = \mathbf{T}$ for $0 \le i \le n-1$. We claim that $W = \{(\alpha_i, \ell_{im_i}) | 0 \le i \le n-1\}$ is the desired set of vertices. Indeed, if (α_i, ℓ_{im_i}) and (α_j, ℓ_{jm_j}) would be two distinct vertices not joined by an edge in G_φ, then we would have $i \neq j$ and $\ell_{im_i} = \overline{\ell_{jm_j}}$. This, however, would lead to a contradiction because

$$v(\ell_{im_i}) = v(\overline{\ell_{jm_j}}) = f_\neg(v(\ell_{jm_j})) = f_\neg(\mathbf{T}) = \mathbf{F}.$$

Conversely, assume that G_φ has a set W of n vertices such that each pair of distinct vertices is joined by an edge. The set of first components of these vertices must consist of n distinct elements, so $W = \{(\alpha_i, \ell_{im_i}) \mid 0 \le i \le n-1\}$. Note that a variable that occurs in a positive literal ℓ_{hm_h} of the set $L = \{\ell_{im_i} \mid 0 \le i \le n-1\}$ may not occur in a negative literal ℓ_{km_k} of the same set since, otherwise, the vertices (α_h, ℓ_{hm_h}) and (α_k, ℓ_{km_k}) would not be joined by an edge. This remark allows us to define the truth assignment $v \in \mathrm{TA}_{SV(\varphi)}$ by

$$v(p) = \begin{cases} \mathbf{T} & \text{if } p \text{ occurs in a positive literal of } L \\ \mathbf{F} & \text{if } p \text{ occurs in a negative literal of } L \\ \text{arbitrary} & \text{otherwise.} \end{cases}$$

Clearly, $v(\varphi) = \mathbf{T}$.

[10]The graph-theoretical term for such a set of vertices is *clique*.

(64) **Craig's[11] Interpolation Lemma** Let φ, ψ be two formulas such that $SV(\varphi) \cap SV(\psi) \neq \emptyset$. Prove that the following assertions are true:

(a) If $(\varphi \to \psi)$ is a tautology, then there exists a formula θ such that $SV(\theta) \subseteq SV(\varphi) \cap SV(\psi)$ and both $(\varphi \to \theta)$ and $(\theta \to \psi)$ are tautologies.
(b) If $\varphi \models \psi$, then there exists a formula θ such that $SV(\theta) \subseteq SV(\varphi) \cap SV(\psi)$ and both $\varphi \models \theta$ and $\theta \models \psi$.
(c) If $(\varphi \vee \psi)$ is a tautology, then there exists a formula θ such that $SV(\theta) \subseteq SV(\varphi) \cap SV(\psi)$ and both $(\varphi \vee \theta)$ and $(\psi \vee (\neg\theta))$ are tautologies.

Solution. Consider the sets of variables $V_0 = SV(\varphi) \cap SV(\psi)$ and $W_0 = SV(\varphi) \cup SV(\psi)$. Since $(\varphi \to \psi)$ is a tautology, for every truth assignment $w \in \mathrm{TA}_{W_0}$, if $w'(\varphi) = \mathbf{T}$, then $w''(\psi) = \mathbf{T}$, where $w' = w \restriction SV(\varphi)$ and $w'' = w \restriction SV(\psi)$. Define the truth table $\tau : \mathrm{TA}_{V_0} \longrightarrow \mathbf{Bool}$ by

$$\tau(v) = \begin{cases} \mathbf{T} & \text{if there exists an extension} \\ & w' \text{ of } v \text{ to } SV(\varphi) \text{ such that} \\ & w'(\varphi) = \mathbf{T} \\ \mathbf{F} & \text{otherwise.} \end{cases}$$

Let θ be a formula whose truth table is τ (for instance, θ could be taken to be in disjunctive normal form) and let v an arbitrary truth assignment. If $v(\varphi) = \mathbf{T}$, then $v(\theta) = \mathbf{T}$ due to the definition of θ so $v(\varphi \to \theta) = \mathbf{T}$, which proves that $(\varphi \to \theta)$ is a tautology.
Consider now a truth assignment u such that $u(\theta) = \mathbf{T}$ and let $u_0 = u \restriction V_0$. Since $u_0(\theta) = \mathbf{T}$, there is an extension w' of u_0

[11]William Craig was born in 1918 in Nuremberg, Germany, and emigrated to the United States in 1937. He studied at Cornell, the University of California at Berkeley, Harvard, the Swiss Federal Institute of Technology, and Princeton and got his Ph.D. from Harvard in 1951. Craig taught at Pennsylvania State University and the University of California at Berkeley where he was a professor of philosophy. Craig's interests were in both the mathematical and philosophical aspects of logic. He served as the president of the Association for Symbolic Logic. Craig died on January 13, 2016.

to $SV(\varphi)$ such that $w'(\varphi) = \mathbf{T}$. Observe that $u \restriction V_0 = w' \restriction V_0$. Define $w'' : W_0 \longrightarrow \mathbf{Bool}$ by

$$w''(p) = \begin{cases} w'(p) & \text{if } p \in SV(\varphi) \\ u(p) & \text{if } p \in SV(\psi) - SV(\varphi). \end{cases}$$

Since w'' coincides with w' on $SV(\varphi)$, we have $w''(\varphi) = \mathbf{T}$, so $w''(\psi) = \mathbf{T}$. Note that w'' coincides with u on $SV(\psi)$ because, if $p \in V_0 = SV(\varphi) \cap SV(\psi)$, then $w''(p) = w'(p) = u(p)$. This implies $u(\psi) = \mathbf{T}$, so $(\theta \to \psi)$ is a tautology.
The second and the third parts are immediate consequences of the first.

Substitutions and Formulas

(65) Prove that $s(\varphi)$ a tautology does not necessarily imply φ a tautology, where s is a substitution and φ is a formula.

(66) Prove that if φ is a contradiction and s is a substitution, then $s(\varphi)$ is a contradiction.

(67) Let φ be a formula in conjunctive normal form such that φ contains no negative literals. Prove that there is a formula ψ in disjunctive normal form such that $\psi \equiv \varphi$, $SV(\psi) = SV(\varphi)$, and ψ contains no negative literals.
Solution. The argument is by induction on n, the number of conjuncts of φ. The basis step, $n = 1$, is immediate. Suppose that the statement holds for n and let $\varphi = \alpha \wedge (p_{i_0} \vee \cdots \vee p_{i_{m-1}})$, where α is a formula in conjunctive normal form that contains n conjuncts and no negative literals. By inductive hypothesis, there is a formula β in disjunctive normal form, $\beta = (\mu_0 \vee \cdots \vee \mu_{l-1})$ such that $\beta \equiv \alpha$, $SV(\beta) = SV(\alpha)$, and each μ_i is a conjunction of statement variables. By Theorem 2.6.12, we have $\varphi \equiv (\mu_0 \vee \cdots \vee \mu_{l-1}) \wedge (p_{i_0} \vee \cdots \vee p_{i_{m-1}})$. An easy semantic argument shows that φ is equivalent to the formula

$$(\mu_0 \wedge p_{i_0}) \vee \cdots \vee (\mu_0 \wedge p_{i_{m-1}}) \vee \cdots \vee (\mu_{l-1} \wedge p_{i_{m-1}})$$

which has the desired form.
Note that this argument contains an algorithm for obtaining ψ from φ.

(68) Let φ be a formula and s be a substitution. Suppose that the list of all occurrences of variables in φ is

$(p_{j_0}, i_0), \ldots, (p_{j_{n-1}}, i_{n-1})$, where $i_0 > \cdots > i_{n-1}$. Prove that there is a sequence of formulas $\varphi_0, \ldots, \varphi_n$ such that the following conditions are satisfied:

(a) $\varphi_0 = \varphi$,
(b) $\varphi_{l+1} = \texttt{replace}(\varphi_l, (p_{j_l}, i_l), s(p_{j_l}))$, for $0 \leq l \leq n-1$, where (p_{j_l}, i_l) is an occurrence in φ_l,
(c) $\varphi_n = s(\varphi)$.

(69) Let $\varphi \in$ PLFORM and let p, q be two statement variables. Define the substitutions s^1_{pq}, s^0_{pq} by

$$s^1_{pq}(r) = \begin{cases} r & \text{if } r \neq p \\ (q \vee (\neg q)) & \text{if } r = p \end{cases}$$

and

$$s^0_{pq}(r) = \begin{cases} r & \text{if } r \neq p \\ (q \wedge (\neg q)) & \text{if } r = p, \end{cases}$$

respectively. Prove that φ is a tautology if and only if $(s^1_{pq}(\varphi) \wedge s^0_{pq}(\varphi))$ is a tautology.

(70) Let (α, i) be an occurrence of a formula α in a formula φ. Prove that

$$\begin{aligned} &SV(\beta) \cup (SV(\varphi) - SV(\alpha)) \\ &\subseteq SV(\texttt{replace}(\varphi, (\alpha, i), \beta) \subseteq SV(\varphi) \cup SV(\beta) \end{aligned}$$

for every formula β.

Let φ, ψ be two formulas such that $\varphi = q_0 \psi q_1 \cdots \psi q_n$, where ψ does not occur in any sequence q_i for $0 \leq i \leq n$. By Exercise 4, this representation is unique and there are n occurrences of ψ in φ. For a formula θ, define $\mathsf{R}(\varphi, \psi, \theta) = q_0 \theta q_1 \cdots \theta q_n$.

(71) Prove that the function R introduced above can be defined recursively by

$$\mathsf{R}(\varphi, \psi, \theta) = \begin{cases} \theta & \text{if } \varphi = \psi \\ \varphi & \text{if } \varphi = p \neq \psi, \quad \text{where} \quad p \in SV \\ (\neg \mathsf{R}(\alpha, \psi, \theta)) & \text{if } \varphi = (\neg \alpha) \quad \text{and} \quad \varphi \neq \psi \\ (\mathsf{R}(\alpha, \psi, \theta) C \mathsf{R}(\beta, \psi, \theta)) & \text{if } \varphi = (\alpha C \beta) \quad \text{and} \quad \varphi \neq \psi, \end{cases}$$

where $\varphi, \psi, \theta \in$ PLFORM. Further, prove that $\mathsf{R}(\varphi, \psi, \theta)$ is a formula.

Truth Sets and Hintikka Sets

(72) (a) Prove that φ is a tautology if and only if it belongs to every truth set.
(b) Prove that $\Gamma \models \varphi$ if and only if φ is a member of every truth set which contains Γ.
(c) Reformulate the remaining parts of Definition 2.3.7 using the notion of truth set.

(73) Prove that every satisfiable set is contained in a maximally satisfiable set.

(74) Let $\Gamma_0 \subseteq \Gamma_1 \subseteq \cdots$ be an ascending chain of Hintikka sets. Prove that $\bigcup\{\Gamma_i \mid i \geq 0\}$ is a Hintikka set.

(75) Give a semantic proof of Theorem 2.7.15.
Solution. Let Γ be a truth set. By Theorem 2.7.9, $\Gamma = \Gamma_w$ for some truth valuation w. It is also clear, by the definition of truth set, that exactly one of p and $(\neg p)$ belongs to Γ, for every statement variable p. Let $\varphi \in \Gamma$ be a formula that is not a literal. Then, since $w(\varphi) = \mathbf{T}$, by Theorem 2.7.2, w satisfies some constituent K of φ, so $K \subseteq \Gamma_w = \Gamma$.

(76) Let Γ be a Hintikka set that contains both formulas φ and $(\varphi \to \psi)$. Prove that $\psi \in \Gamma$.

(77) Let $\mathcal{C}$ be a consistency property. Prove that $\mathcal{C}'$, the smallest property of finite character of the subsets of PLFORM that contains $\mathcal{C}$, is a consistency property (see Exercise 17 of Chapter 1).
Solution. We claim that no set $\Gamma' \in \mathcal{C}'$ contains both a statement variable and its negation. Indeed, if Γ' were to contain both a statement variable p and $(\neg p)$, then $\{p, (\neg p)\}$, as a finite subset of Γ', would be included in a set Γ of $\mathcal{C}$, by the characterization of $\mathcal{C}'$ given in Exercise 17 of Chapter 1. This contradicts the assumption that $\mathcal{C}$ is a consistency property.
Let φ be a formula that is not a literal such that $\varphi \in \Gamma' \in \mathcal{C}'$. We must show that there is a constituent K of φ such that $\Gamma' \cup K \in \mathcal{C}'$.
Let $\varphi_0, \varphi_1, \ldots$ be the standard enumeration of PLFORM and consider the increasing sequence $\Gamma'_0, \Gamma'_1, \ldots$ of finite subsets of Γ' defined by $\Gamma'_n = (\{\varphi_0, \ldots, \varphi_{n-1}\} \cap \Gamma') \cup \{\varphi\}$, for $n \in \mathbf{N}$. Again, by Exercise 17 of Chapter 1, for each n, there is a set $\Gamma_n \in \mathcal{C}$ such that $\Gamma'_n \subseteq \Gamma_n$. Since $\varphi \in \Gamma'_n \subseteq \Gamma_n$, there is a constituent K_n of φ such that $\Gamma_n \cup K_n \in \mathcal{C}$. Since φ has only

finitely many constituents, there is a constituent K of φ such that $K_n = K$ for infinitely many n. We claim that $\Gamma' \cup K \in \mathcal{C}'$. Any finite subset of $\Gamma' \cup K$ can be written as $\hat{\Gamma} \cup \hat{K}$, where $\hat{\Gamma}$ is a finite subset of Γ' and $\hat{K}$ is a finite subset of K. For all n large enough, $\hat{\Gamma} \subseteq \Gamma'_n$ because $\Gamma'_0, \Gamma'_1, \ldots$ is an increasing sequence of sets whose union is Γ'. Since for infinitely many n we have $\hat{K} \subseteq K = K_n$, it follows that there is an n such that $\hat{\Gamma} \cup \hat{K} \subseteq \Gamma'_n \cup K_n \subseteq \Gamma_n \cup K_n \in \mathcal{C}$, which implies $\Gamma' \cup K \in \mathcal{C}'$, by Exercise 17 of Chapter 1.

(78) Let $\mathcal{C}$ be a consistency property. Prove that every maximal element of $\mathcal{C}$ is a Hintikka set.

(79) Use Supplement 77 and Exercise 78 to provide an alternative proof of the fact that every member of a consistency property is contained in a Hintikka set and therefore is satisfiable.
Hint. Use Theorem 1.3.3.

(80) Let $\sharp$: PLFORM $\longrightarrow$ PLFORM be a unary operation on formulas. We denote $\sharp(\varphi)$ by $\varphi^\sharp$. Extend $\sharp$ to signed formulas by $(b\varphi)^\sharp = b\varphi^\sharp$ and further extend the application of $\sharp$ to sequences of sets of signed formulas in the natural way:

(a) Prove that if $(\neg\varphi)^\sharp = (\neg\varphi^\sharp)$ for a formula φ, then for $b \in \mathbf{Bool}$, $\mathtt{d}((b(\neg\varphi))^\sharp) = (\mathtt{d}(b(\neg\varphi)))^\sharp$.

(b) Prove that if $(\varphi C \psi)^\sharp = (\varphi^\sharp C \psi^\sharp)$, for the formulas φ, ψ and the binary connective symbol C, then

$$\mathtt{d}((b(\varphi C \psi))^\sharp) = (\mathtt{d}(b(\varphi C \psi)))^\sharp.$$

(81) Let s be a propositional substitution and let φ be a formula:

(a) Prove that if φ is not a literal, then $\mathtt{d}(s(\varphi)) = s(\mathtt{d}(\varphi))$.

(b) Prove that if φ is not a statement variable and $b \in \mathbf{Bool}$, then $\mathtt{d}(s(b\varphi)) = s(\mathtt{d}(b\varphi))$.

Hint. One can prove this directly; the second part can also be shown by using Exercise 80.

(82) Let C be a connective symbol and let α, β be two formulas. If K is a constituent of the formula $\varphi = (\alpha C \beta)$ and H is a constituent of the formula $\psi = (\neg(\alpha C \beta))$, prove that there exists an immediate subformula γ of φ such that the set $K \cup H$ contains both γ and $(\neg\gamma)$.

Solution. The four possible cases are summarized by the following table:

C	constituent K of $(\alpha C\beta)$	constituent H of $(\neg(\alpha C\beta))$
$\wedge$	$\{\alpha, \beta\}$	$\{(\neg\alpha)\}$ or $\{(\neg\beta)\}$
$\vee$	$\{\alpha\}$ or $\{\beta\}$	$\{(\neg\alpha), (\neg\beta)\}$
$\to$	$\{(\neg\alpha)\}$ or $\{\beta\}$	$\{\alpha, (\neg\beta)\}$
$\leftrightarrow$	$\{\alpha, \beta\}$ or $\{(\neg\alpha), (\neg\beta)\}$	$\{\alpha, (\neg\beta)\}$ or $\{(\neg\alpha), \beta\}$

Note that in every case, $K \cup H$ contains a formula γ and its negation $(\neg\gamma)$. For instance, if $C = \wedge$, then for $H = \{(\neg\alpha)\}$, $\gamma = \alpha$; for $H = \{(\neg\beta)\}$, we have $\gamma = \beta$. We leave to the reader the easy verification of the remaining cases.

(83) Let φ be $(\alpha C\beta)$ or $(\neg(\alpha C\beta))$ for some formulas α, β and some connective symbol C. Suppose that $\mathtt{d}(\varphi) = (K_0, \ldots, K_{n-1})$ and that $\Gamma = \{\psi_i \mid 0 \leq i \leq n-1\}$ is a set of formulas such that $\psi_i \in K_i$ for $0 \leq i \leq n-1$ and Γ does not contain any pair of formulas $\gamma, (\neg\gamma)$. Prove that there exist formulas $\theta_i \equiv (\neg\psi_i)$ for $0 \leq i \leq n-1$ such that $\{\theta_i \mid 0 \leq i \leq n-1\}$ is a constituent of $\overline{\varphi}$.

Hint. An argument can be made by considering the eight cases that result from the four possible choices for C.

(84) (a) Let

$$\Gamma = \left\{ \underbrace{(\neg(\neg(\cdots(\neg}_{2k}\psi)\cdots))) \mid \psi \text{ is a positive formula and } k \in \mathbf{N} \right\}.$$

Show that for all formulas φ, $\varphi \in \Gamma$ if and only if $(\neg\varphi) \notin \Gamma$ and that Γ is p-upward closed but that Γ is not a truth set. (Hence, in Theorem 2.7.7, "Γ is p-upward closed" cannot be added to the list of equivalent conditions.)

(b) Show by a similar example that "Γ is np-upward closed" cannot be added to the list of equivalent conditions in Theorem 2.7.7.

(85) Show that by replacing the first condition in the definition of truth set (Definition 2.7.6) by

1′. for every literal ℓ, $\ell \in \Gamma$ if and only if $\overline{\ell} \notin \Gamma$, and the second condition by

$2'$. Γ is both upward and downward closed,

one obtains an equivalent definition of truth set.

Solution. Suppose that Γ satisfies both $1'$ and $2'$. We begin by proving (by induction on formulas) that for no formula φ can we have $\{\varphi, (\neg\varphi)\} \subseteq \Gamma$. The basis step, when $\varphi = p$, follows immediately from condition ($1'$). For the first inductive step, suppose that the statement holds for ψ and that $\varphi = (\neg\psi)$. If $\{\varphi, (\neg\varphi)\} \subseteq \Gamma$, that is, $\{(\neg\psi), (\neg(\neg\psi))\} \subseteq \Gamma$, then, by the downward closure of Γ, we would have $\{(\neg\psi), \psi\} \subseteq \Gamma$, which would contradict the inductive hypothesis. For the second inductive step, suppose that the statement holds for α and β and let $\varphi = (\alpha C \beta)$, where C is a binary connective symbol. If $\{\varphi, (\neg\varphi)\} \subseteq \Gamma$, then, by the downward closure of Γ, there would be constituents K and H of φ and $(\neg\varphi)$, respectively, with $K \cup H \subseteq \Gamma$. Then, by Supplement 82, there would be a formula $\gamma \in \{\alpha, \beta\}$ such that $\{\gamma, (\neg\gamma)\} \subseteq K \cup H \subseteq \Gamma$, contradicting the inductive hypothesis.

We show now, again by induction on formulas, that for every formula φ at least one of $\varphi, (\neg\varphi)$ belongs to Γ. The basis step ($\varphi = p$) follows immediately from condition ($1'$). For the first inductive step, suppose that the statement holds for ψ, that is, at least one of the formulas $\psi, (\neg\psi)$ is in Γ, and let $\varphi = (\neg\psi)$. By the upward closure of Γ, $\psi \in \Gamma$ implies $(\neg(\neg\psi)) \in \Gamma$, so at least one of $\varphi, (\neg\varphi)$ is in Γ. For the second inductive step, suppose that the statement holds for α and β and let $\varphi = (\alpha C \beta)$, where C is a binary connective symbol. We discuss the case when $C = \leftrightarrow$ and leave the remaining cases to the reader. If $(\alpha \leftrightarrow \beta) \in \Gamma$, we are done. Suppose that $(\alpha \leftrightarrow \beta) \notin \Gamma$. Then, by upward closure of Γ, $\{\alpha, \beta\} \not\subseteq \Gamma$ and $\{(\neg\alpha), (\neg\beta)\} \not\subseteq \Gamma$. Suppose $\alpha \notin \Gamma$. (The argument for $\beta \notin \Gamma$ is similar.) Then, $(\neg\alpha) \in \Gamma$, by the inductive hypothesis and this implies $(\neg\beta) \notin \Gamma$ because $\{(\neg\alpha), (\neg\beta)\} \not\subseteq \Gamma$. Therefore, $\beta \in \Gamma$, again by inductive hypothesis, so $\{(\neg\alpha), \beta\} \subseteq \Gamma$. Since $\{(\neg\alpha), \beta\}$ is a constituent of $(\neg(\alpha \leftrightarrow \beta))$, by upward closure of Γ, $(\neg\varphi) \in \Gamma$. Thus, Γ is a truth set.

Conversely, every truth set obviously satisfies $1'$ and satisfies condition $2'$ by Theorem 2.7.7.

(86) Show, with examples, that a set of formulas Γ such that $\ell \in \Gamma$ if and only if $\overline{\ell} \notin \Gamma$ for all literals ℓ, can satisfy any five of the following six conditions and fail to be a truth set:

- Γ is p-upward closed.
- Γ is np-upward closed.
- Γ is nn-upward closed.
- Γ is p-downward closed.
- Γ is np-downward closed.
- Γ is nn-downward closed.

Hint. Let Γ be the set of formulas defined by

- $p \in \Gamma$ for all $p \in SV$.
- If $\varphi = (\alpha C \beta)$ or $\varphi = (\neg(\alpha C \beta))$ for some connective symbol C and formulas α, β and some constituent of φ is included in Γ, then $\varphi \in \Gamma$.

It is easy to see that $(\neg p) \notin \Gamma$ for all $p \in SV$; also, Γ satisfies all the conditions except the third but does not satisfy the third condition (since, for example, $p \in \Gamma$ but $(\neg(\neg p)) \notin \Gamma$). Similar examples can be given to show that either of the first two conditions can be omitted.
Now let Γ be the set of formulas defined by the following:

- Every positive formula is in Γ.
- If a constituent of a formula is included in Γ, then the formula is in Γ.

Then Γ is clearly upward closed and is np- and nn-downward closed. Note that Γ is not p-downward closed since, for example, the positive formula $((\neg p) \vee (\neg q)) \in \Gamma$, while neither constituent $\{(\neg p)\}$, $\{(\neg q)\}$ is contained in Γ. We invite the reader to provide similar examples for the remaining cases.

(87) Prove that if Γ is a set of formulas such that $\ell \in \Gamma$ if and only if $\overline{\ell} \notin \Gamma$ for all literals ℓ and Γ is upward closed, then Γ contains a truth set.
Hint. For all $p \in SV$, define $w(p) = \mathbf{T}$ if and only if $p \in \Gamma$ and prove that $\Gamma_w \subseteq \Gamma$.

(88) Prove that a set of formulas is a truth set if and only if it is a maximal Hintikka set.

(89) Prove that a set of formulas is a Hintikka set if and only if it is downward closed subset of a truth set.

(90) Give an example of a Hintikka set of formulas that is not a maximally satisfiable set of formulas.

(91) A set of formulas Γ is *semantically consistent* if there is no formula φ such that $\Gamma \models \varphi$ and $\Gamma \models (\neg\varphi)$. Prove that for any set of formulas Γ, the following conditions are equivalent:

(a) Γ is satisfiable.
(b) Γ is semantically consistent.
(c) There is a formula φ such that $\Gamma \not\models \varphi$.

(92) Prove that for any set of formulas Γ, the following conditions are equivalent:

(a) Γ is satisfiable.
(b) Every finite subset of Γ is semantically consistent.
(c) For every finite subset Γ_1 of Γ, there is a formula φ such that $\Gamma_1 \not\models \varphi$.
(d) There is a formula φ such that for every finite subset Γ_1 of Γ, we have $\Gamma_1 \not\models \varphi$.

Solution. The equivalence of the first three conditions follows from the Compactness Theorem and Exercise 91. Condition (d) immediately implies Condition (c). On the other hand, Condition (a) implies Condition (d) because if Γ is satisfiable, by Condition (c) of Exercise 91, there is a formula φ such that $\Gamma \not\models \varphi$ and, therefore, $\Gamma_1 \not\models \varphi$ for every finite subset Γ_1 of Γ.

(93) (a) Let Γ be a maximal finitely satisfiable set of formulas. Prove that Γ is upward closed. Using Exercise 37 and Theorem 2.7.7, conclude that every maximal finitely satisfiable set of formulas is a truth set.
(b) Use Part (a) of this exercise, Exercise 35, and Theorem 1.3.3 to give another proof of the Compactness Theorem.

Solution. In order to prove Part (a), it suffices to prove that if $K \subseteq \Gamma$ is a constituent of a formula φ, then $\Gamma \cup \{\varphi\}$ is finitely satisfiable because then, the maximality of Γ would imply that $\varphi \in \Gamma$. To do this, it suffices to prove that if Γ_0 is a finite subset of Γ, then $\Gamma_0 \cup \{\varphi\}$ is satisfiable. Since $K \subseteq \Gamma$, the finite set $\Gamma_0 \cup K$ is satisfiable. By Theorem 2.7.2, any truth assignment that satisfies $\Gamma_0 \cup K$ also satisfies φ, and so it satisfies $\Gamma_0 \cup \{\varphi\}$.

The alternative proof of the Compactness Theorem mentioned in Part (b) is obtained as follows. Let Γ be a finitely satisfiable set. By Exercise 35 and Theorem 1.3.3, there is a maximal finitely satisfiable set Γ_1 that includes Γ. By Part (a), Γ_1 is a truth set and, therefore, it is satisfiable, which implies the satisfiability of Γ.

(94) Let $b\varphi$ be a signed formula. Prove that if the formula α occurs positively (negatively) in some constituent of $b\varphi$, then α occurs positively (negatively) in $b\varphi$.

(95) Let Γ be a set of formulas and let v be a truth assignment. Prove that the set of signed formulas $\{v(\varphi)\varphi \mid \varphi \in \Gamma\}$ is satisfiable.

Define the unsigning function $\mathsf{u} : \text{SPLFORM} \longrightarrow \text{PLFORM}$ and the signing function $\mathsf{s} : \text{PLFORM} \longrightarrow \text{SPLFORM}$ by

$$\mathsf{u}(b\varphi) = \begin{cases} \varphi & \text{if } b = \mathbf{T} \\ (\neg\varphi) & \text{if } b = \mathbf{F} \end{cases} \tag{2.11}$$

for every $b\varphi \in \text{SPLFORM}$ and

$$\mathsf{s}(\varphi) = \begin{cases} \mathbf{T}\varphi & \text{if } \varphi \text{ is positive} \\ \mathbf{F}\psi & \text{if } \varphi = (\neg\psi) \end{cases} \tag{2.12}$$

for every $\varphi \in \text{PLFORM}$.

(96) Prove that the following identities hold:

$$\mathsf{u}(\mathsf{s}(\varphi)) = \varphi,$$
$$\mathsf{s}(\mathsf{u}(\mathbf{F}\varphi)) = \mathbf{F}\varphi,$$
$$\mathsf{s}(\mathsf{u}(\mathbf{T}\varphi)) = \begin{cases} \mathbf{T}\varphi & \text{if } \varphi \text{ is positive} \\ \mathbf{F}\psi & \text{if } \varphi = (\neg\psi), \end{cases}$$

for every formula $\varphi \in \text{PLFORM}$. Conclude that s is a one-to-one function and u is an onto function.

(97) Prove that a truth assignment v satisfies $b\varphi$ if and only if v satisfies $\mathsf{u}(b\varphi)$; further, prove that v satisfies φ if and only it satisfies $\mathsf{s}(\varphi)$, for every $\varphi \in \text{PLFORM}$ and $b \in \mathbf{Bool}$.

(98) Define a function $\Xi : \mathcal{P}(\text{SPLFORM}) \longrightarrow \mathcal{P}(\text{PLFORM})$ by $\Xi(\Delta) = \{\varphi \mid \mathbf{T}\varphi \in \Delta\} \cup \{(\neg\varphi) \mid \mathbf{F}\varphi \in \Delta\}$. Observe that $\Xi(\Delta)$ is the extension of the function u, defined above, to sets of signed formulas:

(a) Prove that a truth assignment satisfies Δ if and only if it satisfies $\Xi(\Delta)$.
(b) Show that if φ is a positive formula, then $\varphi \in \Xi(\Delta)$ if and only if $\mathbf{T}\varphi \in \Delta$.
(c) Let Δ be a Hintikka set of signed formulas. Prove that $(\neg\varphi) \in \Xi(\Delta)$ if and only if $\mathbf{F}\varphi \in \Delta$.
(d) Show that if $b\varphi$ does not have the form $\mathbf{T}(\neg\alpha)$ for some α, and $\mathtt{d}(b\varphi) = (K_0, \ldots, K_{l-1})$, then $\Xi(\{b\varphi\}) = \{\psi\}$ for some ψ and $\mathtt{d}(\psi) = (\Xi(K_0), \ldots, \Xi(K_{l-1}))$.
(e) Prove that if Δ is a Hintikka set of signed formulas, then $\Xi(\Delta)$ is a Hintikka set of formulas.
(f) Give an example of a set of signed formulas Δ such that $\Xi(\Delta)$ is a Hintikka set of formulas although Δ is not a Hintikka set of signed formulas.
(g) Use the previous parts to derive Theorem 2.7.28 from Theorem 2.7.16.

Solution. We leave to the reader the arguments for Parts (a) and (b).
(c) Note that if $(\neg\varphi) \in \Xi(\Delta)$, then $\mathbf{T}(\neg\varphi) \in \Delta$ or $\mathbf{F}\varphi \in \Delta$. In the former case, since Δ is a Hintikka set of signed formulas, we have $\mathbf{F}\varphi \in \Delta$. The reverse implication is trivial.
(d) We leave the verification of this statement to the reader.
(e) Let Δ be a Hintikka set of signed formulas. By Parts (b) and (c), for no statement variable p we can have both p and $(\neg p)$ in $\Xi(\Delta)$. We need to verify that for every formula $\varphi \in \Xi(\Delta)$ that is not a literal there is a constituent which is included in $\Xi(\Delta)$. Suppose first that φ is a positive formula. By Part (b), $\mathbf{T}\varphi \in \Delta$, so some K in $\mathtt{d}(\mathbf{T}\varphi)$ is included in Δ. By Part (d), $\Xi(K)$ is a constituent of φ which is contained in $\Xi(\Delta)$. When φ is negative, the same result can be obtained using Parts (c) and (d).
(f) Take $\Delta = \{\mathbf{T}(\neg p)\}$ for p a statement variable.
(g) This part is left to the reader.

(99) Let $\Upsilon : \mathcal{P}(\text{PLFORM}) \longrightarrow \mathcal{P}(\text{SPLFORM})$ be given by $\Upsilon(\Gamma) = \{\mathbf{T}\varphi \mid \varphi \in \Gamma\} \cup \{\mathbf{F}\varphi \mid (\neg\varphi) \in \Gamma\}$:

(a) Prove that a truth assignment satisfies Γ if and only if it satisfies $\Upsilon(\Gamma)$.

(b) Show that $\Upsilon(\Gamma)$ is a Hintikka set of signed formulas if and only if Γ is a Hintikka set of formulas.
(c) Use the previous parts to derive Theorem 2.7.16 from Theorem 2.7.28.

Hint. Use an argument similar to that of Supplement 98.

(100) Let Ξ and Υ be as in Supplements 98 and 99. Prove that for every set Δ of signed formulas and for every set Γ of unsigned formulas, we have

$$\Xi(\Upsilon(\Gamma)) = \Gamma,$$
$$\Upsilon(\Xi(\Delta)) = \Delta \cup \{\mathbf{T}(\neg\varphi) \mid \mathbf{F}\varphi \in \Delta\} \cup \{\mathbf{F}\varphi \mid \mathbf{T}(\neg\varphi) \in \Delta\}.$$

A *maximally satisfiable set of signed formulas* is a set of signed formulas Δ that is satisfiable and for which there is no satisfiable set of signed formulas Δ' such that $\Delta \subset \Delta'$.

A set of signed formulas Δ is $\mathbf{T}$-*downward closed* if for every signed formula $\mathbf{T}\varphi \in \Delta - (\mathbf{Bool} \times SV)$ there exists a constituent K of $\mathbf{T}\varphi$ such that $K \subseteq \Delta$. A set of signed formulas Δ is $\mathbf{F}$-*downward closed* if for every signed formula $\mathbf{F}\varphi \in \Delta - (\mathbf{Bool} \times SV)$ there exists a constituent K of $\mathbf{F}\varphi$ such that $K \subseteq \Delta$. Δ is downward-closed if it is both $\mathbf{T}$- and $\mathbf{F}$-downward closed.

The notions of $\mathbf{T}$-upward, $\mathbf{F}$-upward, and upward closed are defined similarly.

Δ is a *saturated set* of signed formulas if it satisfies the following conditions:

- For every $\varphi \in$ PLFORM, $\mathbf{T}\varphi \in \Delta$ if and only if $\mathbf{F}\varphi \notin \Delta$.
- Δ is both upward and downward closed.

(101) Let Δ be a satisfiable set of signed formulas. Prove that Δ is maximally satisfiable if and only if exactly one of the formulas $\mathbf{T}\varphi$ and $\mathbf{F}\varphi$ belongs to Δ for every formula $\varphi \in$ PLFORM.

(102) If v is a truth assignment, define

$$\Delta_v = \{b\varphi \in \text{SPLFORM} \mid v \text{ satisfies } b\varphi\}.$$

Prove that a set of signed formulas Δ is maximally satisfiable if and only if $\Delta = \Delta_v$ for some truth assignment v.

Solution. Let Δ be a maximally satisfiable set of signed formulas and let v be a truth assignment that satisfies Δ. Then, $\Delta \subseteq \Delta_v$ and since Δ_v is clearly satisfiable, we obtain $\Delta = \Delta_v$. Conversely, since exactly one of the formulas $\mathbf{T}\varphi$ and $\mathbf{F}\varphi$ belongs to Δ_v, for every $\varphi \in$ PLFORM, by Exercise 101, Δ_v is maximally satisfiable.

(103) Show that a set of signed formulas is maximally satisfiable if and only if it is a saturated set.

Solution. Let Δ be a maximally satisfiable set of signed formulas. By Supplement 102, $\Delta = \Delta_v$ for some truth assignment v. It follows easily from Theorem 2.7.25 that $\Delta = \Delta_v$ is a saturated set.

Conversely, suppose that Δ is a saturated set of signed formulas. Since Δ is a Hintikka set, Δ is satisfiable and therefore, by Exercise 101, it is maximally satisfiable.

If $b\varphi \in$ SPLFORM, let $\overline{b\varphi}$ be the signed formula $f_{\neg}(b)\varphi$. Observe that $\overline{\overline{b\varphi}} = b\varphi$. This notation is extended naturally to sets and sequences of signed formulas.

(104) Let φ be a formula that is not a variable. Suppose that K is a constituent of the signed formula $\mathbf{T}\varphi$ and H is a constituent of the signed formula $\mathbf{F}\varphi$. Prove that there exists an immediate subformula γ of φ such that the set $K \cup H$ contains both $\mathbf{T}\gamma$ and $\mathbf{F}\gamma$.

(105) Let $b\varphi \in \text{SPLFORM} - (\mathbf{Bool} \times SV)$, $\mathtt{d}(b\varphi) = (K_0, \ldots, K_{m-1})$. Prove that if $b_j\psi_j \in K_j$ is a signed formula for each j, $0 \leq j \leq m-1$, such that the set $L = \{b_j\psi_j \mid 0 \leq j \leq m-1\}$ does not contain both $\mathbf{T}\gamma$ and $\mathbf{F}\gamma$ for any formula γ, then there is a constituent H of $\overline{b\varphi}$ such that $H = \{\overline{b_j\psi_j} \mid 0 \leq j \leq m-1\}$.

Solution. The proof consists in verifying all possible choices for the formulas $b_i\theta_i$ for the constituents K_i of $b\varphi$. We limit our discussion to $b\varphi = \mathbf{T}(\alpha \leftrightarrow \beta)$. The constituents of $b\varphi$ are $\{\mathbf{T}\alpha, \mathbf{T}\beta\}$ and $\{\mathbf{F}\alpha, \mathbf{F}\beta\}$ and there are four choices for L: $\{\mathbf{T}\alpha, \mathbf{F}\alpha\}$, $\{\mathbf{T}\alpha, \mathbf{F}\beta\}$, $\{\mathbf{F}\alpha, \mathbf{T}\beta\}$, $\{\mathbf{F}\beta, \mathbf{T}\beta\}$. (Of course, if $\alpha = \beta$, then not all of these choices are distinct.) The first and last choices contain both a formula and its negation. The remaining two choices yield two choices for $\{\overline{b\varphi} \mid b\varphi \in L\}$: $\{\mathbf{F}\alpha, \mathbf{T}\beta\}$, $\{\mathbf{T}\alpha, \mathbf{F}\beta\}$, which are constituents of

$\overline{b\varphi} = \mathbf{F}(\alpha \leftrightarrow \beta)$. We leave to the reader the verification of the other cases.

(106) Let Δ be a set of signed formula such that exactly one of the formulas $\mathbf{T}\varphi$ and $\mathbf{F}\varphi$ belongs to Δ for every $\varphi \in$ PLFORM. Prove the following:

(a) Δ is $\mathbf{T}$-upward closed if and only if Δ is $\mathbf{F}$-downward closed.
(b) Δ is $\mathbf{F}$-upward closed if and only if Δ is $\mathbf{T}$-downward closed.

Solution. Let Δ be a set of signed formulas such that Δ is $\mathbf{T}$-upward closed and exactly one of the formulas $\mathbf{T}\varphi$ and $\mathbf{F}\varphi$ belongs to Δ for every $\varphi \in$ PLFORM and let $\mathbf{F}\varphi \in \Delta$, where φ is not a variable. Suppose that $\mathtt{d}(\mathbf{T}\varphi) = (K_0, \ldots, K_{l-1})$. Since $\mathbf{T}\varphi \notin \Delta$, no K_i is included in Δ because Δ is $\mathbf{T}$-upward closed. Therefore, each K_i contains a signed formula $b_i\theta_i$ such that $b_i\theta_i \notin \Delta$. Observe that the set $L = \{\overline{b_i\theta_i} \mid 0 \leq i \leq l-1\}$ is included in Δ and may not contain both $c\gamma$ and $\overline{c\gamma}$ for some $c\gamma \in$ SPLFORM since this would violate the hypothesis made about Δ. Therefore, by Supplement 105, L is a constituent of $\mathbf{F}\varphi$ and $L \subseteq \Delta$, which allows us to conclude that Δ is $\mathbf{F}$-downward closed.
Conversely, suppose that Δ is an $\mathbf{F}$-downward closed set of signed formulas such that exactly one of the formulas $\mathbf{T}\varphi$ and $\mathbf{F}\varphi$ belongs to Δ for every $\varphi \in$ PLFORM and let $\mathbf{T}\varphi$ be a formula such that a constituent K of $\mathbf{T}\varphi$ is included in Δ. If $\mathbf{F}\varphi \in \Delta$, at least one constituent K' of $\mathbf{F}\varphi$ is also included in Δ because Δ is $\mathbf{F}$-downward closed. However, this contradicts the hypothesis, by Supplement 104. Therefore, $\mathbf{F}\varphi \notin \Delta$, so $\mathbf{T}\varphi \in \Delta$ which shows that Δ is $\mathbf{T}$-upward closed.
We leave to the reader the similar argument needed for Part (b).

(107) Let Δ be a set of signed formulas such that exactly one of the formulas $\mathbf{T}\varphi$ and $\mathbf{F}\varphi$ belongs to Δ for every formula $\varphi \in$ PLFORM. Prove that the following statements are equivalent:

(a) Δ is $\mathbf{T}$-upward closed and $\mathbf{F}$-upward closed, that is, Δ is upward closed.
(b) Δ is $\mathbf{T}$-upward and $\mathbf{T}$-downward closed.

(c) Δ is **F**-downward and **T**-downward closed, that is, Δ is downward closed.
(d) Δ is **F**-downward and **F**-upward closed.
(e) Δ is both upward and downward closed.

It follows that the second condition of the definition of saturated set can be replaced by any of the conditions given as follows.
Hint. Use Supplement 106.

(108) Let $\Delta = \{\mathbf{T}\varphi \mid \varphi \in \text{PLFORM}\}$. Show that for all signed formulas $\mathbf{T}\varphi$, $\mathbf{T}\varphi \in \Delta$ if and only if $\mathbf{F}\varphi \notin \Delta$ and that Δ is **T**-upward closed but that Δ is not a saturated set. (Hence, in Supplement 107, "Δ is **T**-upward closed" cannot be added to the list of equivalent conditions.)

(109) Show by a similar example that "Δ is **F**-upward closed" cannot be added to the list of equivalent conditions in Supplement 107.

(110) Prove that Δ is a saturated set of signed formulas if and only if for every statement variable p, $\mathbf{T}p \in \Delta$ if and only if $\mathbf{F}p \notin \Delta$ and Δ is both upward and downward closed.
Solution. The only nontrivial part of the solution is to prove that if Δ is a set of signed formulas such that for every statement variable p, $\mathbf{T}p \in \Delta$ if and only if $\mathbf{F}p \notin \Delta$ and Δ is both upward and downward closed, then for every formula φ, $\mathbf{T}\varphi \in \Delta$ if and only if $\mathbf{F}\varphi \notin \Delta$. We show this by induction on φ. The basis step, when φ is a statement variable, is immediate. Now suppose that this property holds for all immediate subformulas of a formula φ where $\varphi \notin SV$.
If $\mathbf{T}\varphi \in \Delta$, then, since Δ is downward closed, there is a constituent K of $\mathbf{T}\varphi$ such that $K \subseteq \Delta$. If $\mathbf{F}\varphi \in \Delta$, then there is a constituent H of $\mathbf{F}\varphi$ included in Δ. By Supplement 104, this contradicts the inductive hypothesis. Therefore, $\mathbf{F}\varphi \notin \Delta$.
Assume now that $\mathbf{F}\varphi \notin \Delta$. Since Δ is upward closed, no constituent of $\mathbf{F}\varphi$ is contained in Δ If $\mathsf{d}(\mathbf{F}\varphi) = (H_0, \ldots, H_{l-1})$, then for each i with $0 \leq i \leq l-1$, there is a signed formula $b_i\theta_i \in H_i - \Delta$ where each θ_i is an immediate subformula of φ. Let $L = \{\overline{b_i\theta_i} \mid 0 \leq i \leq l-1\}$. By inductive hypothesis, $L \subseteq \Delta$ and there is no i such that $\mathbf{T}\theta_i$ and $\mathbf{F}\theta_i$ belong to L. Therefore, by Supplement 105, L is a constituent of $\mathbf{T}\varphi$, so by upward closure of Δ, $\mathbf{T}\varphi \in \Delta$.

(111) Show with examples that a set of signed formulas Δ such that $\mathbf{T}p \in \Delta$ if and only if $\mathbf{F}p \notin \Delta$ for all $p \in SV$ can satisfy any three of the following four conditions and fail to be a saturated set:

- Δ is **T**-upward closed.
- Δ is **F**-upward closed.
- Δ is **T**-downward closed.
- Δ is **F**-downward closed.

(112) Prove that a set of signed formulas is a Hintikka set if and only if it is a downward closed subset of a saturated set of signed formulas.

(113) Give an example of a Hintikka set of signed formulas that is not a saturated set of signed formulas.

The notion of consistency property can be formulated for signed formulas. Namely, a *consistency property* is a collection $\mathcal{C}$ of sets of signed formulas such that we have the following:

- No set with property $\mathcal{C}$ contains both $\mathbf{T}p$ and $\mathbf{F}p$ for any statement variable p.
- If Δ has property $\mathcal{C}$, $b\varphi \in \Delta$ and φ is not a statement variable, then $\Delta \cup K$ has property $\mathcal{C}$ for some constituent K of $b\varphi$.

A collection of sets of signed formulas $\mathcal{I}$ is an *inconsistency property* if the collection of sets $\mathcal{P}(\text{SPLFORM}) - \mathcal{I}$ is a consistency property.

Note that a collection of sets of signed formulas I is an inconsistency property if and only if the following conditions are satisfied:

- Every set of signed formulas that includes $\{\mathbf{T}p, \mathbf{F}p\}$ for some statement variable p belongs to $\mathcal{I}$.
- If Δ is a set of signed formulas and $b\varphi \in \Delta$ is such that for every constituent K of φ, $\Delta \cup K \in \mathcal{I}$, then $\Delta \in \mathcal{I}$.

(114) Let $\mathcal{C}$ be the collection of all satisfiable sets of signed formulas. Show that $\mathcal{C}$ is a consistency property.

(115) Show that every member of a consistency property of signed formulas is a satisfiable set. In fact, prove that if Δ belongs to a consistency property, then there is a Hintikka set of signed

formulas Δ' such that $\Delta \subseteq \Delta'$ and every formula in Δ' is a signed subformula of a formula in Δ.
Hint. Adapt the proof of Theorem 2.7.21.

(116) Prove that every satisfiable set of signed formulas Δ is contained in a Hintikka set of signed formulas Δ' such that every formula in Δ' is a signed subformula of a formula in Δ.
Hint. Use both Exercises 114 and 115.

(117) Use Part (a) of Supplement 98 to show that if Δ is a set of signed formulas such that every finite subset of Δ is satisfiable, then Δ is satisfiable. (This is the Compactness Theorem for Signed Formulas. We will encounter this result later in this book in Theorem 3.3.41.)

(118) Use the notion of consistency property to show that if Δ is an unsatisfiable set of signed formulas, then there is a finite subset Δ' of Δ that is unsatisfiable. (This gives another proof of the Compactness Theorem for Signed Formulas.)

Truth Functions

(119) Let v be a truth assignment and let $\varphi_0, \ldots, \varphi_{n-1}$ be a nonnull sequence of formulas. Prove that

$$v\left(\bigvee_{0\leq i\leq n-1} \varphi_i\right) = \bigvee\nolimits_{0\leq i\leq n-1} v(\varphi_i),$$

$$v\left(\bigwedge_{0\leq i\leq n-1} \varphi_i\right) = \bigwedge\nolimits_{0\leq i\leq n-1} v(\varphi_i).$$

(120) A function $f \in \mathtt{TF}_n$ *depends essentially* on the ith argument, where $0 \leq i \leq n-1$, if

$$\begin{aligned} &f(a_0, \ldots, a_{i-1}, \mathbf{F}, a_{i+1}, \ldots, a_{n-1}) \\ &\quad \neq f(a_0, \ldots, a_{i-1}, \mathbf{T}, a_{i+1}, \ldots, a_{n-1}) \end{aligned}$$

for some $a_0, \ldots, a_{i-1}, a_{i+1}, \ldots, a_{n-1} \in \mathbf{Bool}$; f is *nondegenerate* if it depends essentially on all its arguments. The number of nondegenerate n-ary truth functions is denoted by $\text{nd}(n)$.

(a) Show that every 0-ary truth function is nondegenerate, that is, $\text{nd}(0) = 2$.

(b) Prove that

$$\sum_{k=0}^{n} \binom{n}{k} \mathrm{nd}(k) = |\mathtt{TF}_n|$$

for all $n \in \mathbf{N}$.

(121) Let f_+ be the binary truth function defined by $f_+(a,b) = a + b$ for $a, b \in \mathbf{Bool}$. Prove that f_+ is a linear function which is neither self-dual nor monotonic.

(122) Prove that the operation $\leftrightarrow$ is commutative and associative. Moreover, show that $a \leftrightarrow a = \mathbf{T}$, $a \leftrightarrow \mathbf{F} = \overline{a}$, and $a \leftrightarrow \mathbf{T} = a$, for $a \in \mathbf{Bool}$.

(123) The operation "$\leftrightarrow$" can be extended to sequences by defining the function $\mathbf{BIC} : \mathrm{Seq}(\mathbf{Bool}) \longrightarrow \mathbf{Bool}$:

$$\mathbf{BIC}(\lambda) = \mathbf{T},$$

$$\mathbf{BIC}(a_0, \ldots, a_n) = \mathbf{BIC}(a_0, \ldots, a_{n-1}) \leftrightarrow a_n,$$

for every $a_0, \ldots, a_n \in \mathbf{Bool}$ and $n \in \mathbf{N}$. $\mathbf{BIC}(a_0, \ldots, a_{n-1})$ will be denoted by $a_0 \leftrightarrow \cdots \leftrightarrow a_{n-1}$.

Prove that $a_0 \leftrightarrow \cdots \leftrightarrow a_{n-1} = \mathbf{T}$ if and only if $|\{i \mid 0 \leq i \leq n-1 \text{ and } a_i = \mathbf{T}\}|$ has the same parity as n.

(124) Let $f_{\mathrm{maj}} : \mathbf{Bool}^3 \longrightarrow \mathbf{Bool}$ be the majority truth function that takes the value $\mathbf{T}$ if and only if at least two of its arguments are $\mathbf{T}$. Prove that f_{maj} is a monotonic, self-dual function which is not linear.

(125) (a) Let A, B, C be three sets and let $\xi : A \longrightarrow B$ be a bijection. Define the functions $\Phi : (B \longrightarrow C) \longrightarrow (A \longrightarrow C)$ by $\Phi(f) = f\xi$ and $\Psi : (A \longrightarrow C) \longrightarrow (B \longrightarrow C)$ by $\Psi(g) = g\xi^{-1}$. Prove that Φ and Ψ are bijections that are inverses of each other.

(b) Let S be a finite set of statement variables with $|S| = n$. Consider the mappings $\xi_S : \mathbf{Bool}^n \longrightarrow \mathrm{TA}_S$ and $\zeta_S : \mathrm{TA}_S \longrightarrow \mathbf{Bool}^n$ given by $\xi_S(\vec{a}) = v_{\vec{a},S}$ and $\zeta_S(v) = \vec{a}_v$ for every $\vec{a} \in \mathbf{Bool}^n$ and $v \in \mathrm{TA}_S$. (The notation $v_{\vec{a},S}$ was introduced just prior to Definition 2.8.47 and the notation $\vec{a}_v$ in the proof of Theorem 2.8.48.)

Show that ξ_S and ζ_S are bijections which are inverse to each other.

(c) Let S be a finite set of statement variables. Reprove Theorem 2.8.48 using the previous two parts of this exercise.

(126) Let $\varphi_0, \ldots, \varphi_{m-1}$ be a nonnull sequence of formulas and let S be a set of statement variables such that $SV(\varphi_i) \subseteq S$ for $0 \leq i \leq m-1$. Show that

$$\Phi_S\left(\tau_{\bigvee_{0\leq i\leq m-1}\varphi_i,S}\right) = \bigvee_{0\leq i\leq m-1}\Phi_S(\tau_{\varphi_i,S}),$$

$$\Phi_S\left(\tau_{\bigwedge_{0\leq i\leq m-1}\varphi_i,S}\right) = \bigwedge_{0\leq i\leq m-1}\Phi_S(\tau_{\varphi_i,S}),$$

where the function Φ_S was defined in Theorem 2.8.48.

Solution. We prove the first equality and leave the second to the reader. Suppose that $|S| = n$ and that $\vec{b} \in \mathbf{Bool}^n$. Then, applying the definition of Φ_S, we can write

$$\begin{aligned}
\Phi_S\left(\tau_{\bigvee_{0\leq i\leq m-1}\varphi_i,S}\right)(\vec{b}) &= f_{\tau_{\bigvee_{0\leq i\leq m-1}\varphi_i,S}}(\vec{b})\\
&= \tau_{\bigvee_{0\leq i\leq m-1}\varphi_i,S}(v_{\vec{b},S})\\
&= v_{\vec{b},S}\left(\bigvee_{0\leq i\leq m-1}\varphi_i\right)\\
&= \bigvee_{0\leq i\leq m-1} v_{\vec{b},S}(\varphi_i)\\
&= \bigvee_{0\leq i\leq m-1} \tau_{\varphi_i,S}(v_{\vec{b},S})\\
&= \bigvee_{0\leq i\leq m-1} f_{\tau_{\varphi_i,S}}(\vec{b})\\
&= \bigvee_{0\leq i\leq m-1} \Phi_S(\tau_{\varphi_i,S})(\vec{b})
\end{aligned}$$

for every $\vec{b} \in \mathbf{Bool}^n$. This gives the first equality of the exercise.

(127) Let $n \geq 1$ and suppose that S is a set of n statement variables. Show that, for every $\vec{b} \in \mathbf{Bool}^n$, we have

$$f_{\vec{b}} = \Phi_S(\tau_{\mu_{v_{\vec{b},S}}}),$$
$$g_{\vec{b}} = \Phi_S(\tau_{\nu_{v_{\vec{b}',S}}}),$$

where $\vec{b}' = \neg\vec{b}$, and the notations $f_{\vec{b}}, g_{\vec{b}}$ were introduced in Definition 2.8.23.

Solution. Let $\varphi = \mu_{v_{\vec{b},S}}$. We have

$$\begin{aligned}\Phi_S(\tau_\varphi)(\vec{x}) &= f_{\tau_\varphi}(\vec{x})\\ &= \tau_\varphi(v_{\vec{x},S})\\ &= v_{\vec{x},S}(\varphi)\\ &= v_{\vec{x},S}(\mu_{v_{\vec{b},S}})\\ &= \begin{cases}\mathbf{T} & \text{if } v_{\vec{x},S} = v_{\vec{b},S}\\ \mathbf{F} & \text{otherwise}\end{cases}\\ &= \begin{cases}\mathbf{T} & \text{if } \vec{x} = \vec{b}\\ \mathbf{F} & \text{otherwise}\end{cases}\\ &= f_{\vec{b}}(\vec{x}).\end{aligned}$$

We leave to the reader the second part of the argument.

(128) Let $S = \{p_0, \ldots, p_{n-1}\}$ be a nonempty set of statement variables. Prove that an n-ary truth function f is an n-ary minterm (maxterm) function if and only if $f = \Phi_S(\tau_\varphi)$, where φ is a minterm (maxterm) over S.

Solution. Suppose first that f is an n-ary minterm function, say $f = f_{\vec{b}}$, with $\vec{b} \in \mathbf{Bool}^n$. Then, the existence of φ follows immediately from Exercise 127.

Conversely, let μ_v be a minterm over S. If $\vec{b} = (v(p_0), \ldots, v(p_{n-1}))$, then $v = v_{\vec{b},S}$ and, by Exercise 127, $\Phi_S(\tau_{\mu_v})$ is the n-ary minterm function $f_{\vec{b}}$.

The argument for maxterms is similar and is left to the reader.

(129) Obtain alternate proofs of Theorems 2.8.27 and 2.8.28 by using Exercises 126 and 127 and the proofs of Theorems 2.5.10 and 2.5.60.

(130) Prove that for $n \in \mathbf{N}$ there are 2^{n+1} linear n-ary truth functions.
Hint. Use the uniqueness of the polynomial normal form.

(131) Prove that for $n \in \mathbf{N}$ there are 2^n linear, self-dual n-ary truth functions.
Hint. Prove that an n-ary linear function presented as in Definition 2.8.16 is self-dual if and only if $k_0 + \cdots + k_{n-1} = \mathbf{T}$.

(132) Prove that if $f \in \mathtt{TF}_2$ and f is nonlinear, then there are $a, b \in \mathbf{Bool}$ such that the functions $f(a, x_1)$ and $f(x_0, b)$ are constant.
Hint. Use the polynomial normal form of f.

(133) Prove that for $n \in \mathbf{N}$, $f \in \mathtt{TF}_n$ is linear if and only if $f_\neg$ is linear.

(134) Let $f \in \mathtt{TF}_2$. Define the functions $g, h, l \in \mathtt{TF}_2$ by $g(a,b) = f(f_\neg(a), b)$, $h(a,b) = f(a, f_\neg(b))$, and $l(a,b) = f(f_\neg(a), f_\neg(b))$, for $a, b \in \mathbf{Bool}$. Prove that if any one of the functions f, g, h, l is linear, then the other three are also linear. Generalize this result to functions in $\mathtt{TF}_n$.

Clones and Functional Completeness

(135) Let $\wp : \{0, \ldots, k-1\} \longrightarrow \{0, \ldots, n-1\}$ be a function and let $f \in \mathtt{TF}_\ell$, $g_0, \ldots, g_{\ell-1} \in \mathtt{TF}_k$. Prove that

$$(f(g_0, \ldots, g_{\ell-1}))^\wp = f(g_0^\wp, \ldots, g_{\ell-1}^\wp).$$

(136) Let $\wp : \{0, \ldots, k-1\} \longrightarrow \{0, \ldots, n-1\}$ be a function. Prove that for every projection π_i^k, we have $(\pi_i^k)^\wp = \pi_{\wp(i)}^n$.

(137) Prove that if T is a clone and $T \subset \mathtt{TF}$, then T is contained in at least one of the clones $\mathcal{T}_0$, $\mathcal{T}_1$, $\mathcal{SD}$, $\mathcal{LIN}$ or $\mathcal{MON}$.

(138) Prove that none of the clones $\mathcal{T}_0$, $\mathcal{T}_1$, $\mathcal{SD}$, $\mathcal{LIN}$, or $\mathcal{MON}$ is included in any of the others, thereby obtaining a strengthening of Lemma 2.9.8.
Hint. For each pair of clones, it is necessary to find a truth function included in one but not the other. The required examples can be found among the truth functions $f_0, f_1, f_\neg, f_+, f_\leftrightarrow$, and f_{maj}. (Recall that f_+ and f_{maj} are introduced in Example 2.9.18 and Exercise 124, respectively.)

(139) We call a clone T *maximal* if $T \subset \mathtt{TF}$ and there is no clone T' such that $T \subset T' \subset \mathtt{TF}$. Using Exercises 137 and 138, prove

that the maximal clones are exactly the clones $\mathcal{T}_0$, $\mathcal{T}_1$, $\mathcal{SD}$, $\mathcal{LIN}$, and $\mathcal{MON}$.

(140) Prove that

$$\mathcal{SD} \cap (\mathcal{MON} \cup \mathcal{T}_1) \subseteq \mathcal{T}_0, \quad \mathcal{SD} \cap (\mathcal{MON} \cup \mathcal{T}_0) \subseteq \mathcal{T}_1.$$

(141) Let $T = \{f_\wedge, f_0, f_1, f\}$, where $f \in \mathtt{TF}_3$ is given by $f(x_0, x_1, x_2) = x_0 + x_1 + x_2$ for every $x_0, x_1, x_2 \in \mathbf{Bool}$:

(a) Prove that T is a complete set of functions.

(b) Prove that the set T is minimally complete, that is, no proper subset of T is complete.

Solution. The completeness of T follows from the fact that $f_1 \notin \mathcal{T}_0$, $f_0 \notin \mathcal{T}_1$, $f_0 \notin \mathcal{SD}$, $f \notin \mathcal{MON}$, and $f_\wedge \notin \mathcal{LIN}$.
For the second part, observe that every three-element subset of T is contained in one of the clones specified in Theorem 2.9.17. Namely, we have

$$\begin{aligned} \{f_0, f_1, f\} &\subseteq \mathcal{LIN}, & \{f_\wedge, f_1, f\} &\subseteq \mathcal{T}_1, \\ \{f_\wedge, f_0, f\} &\subseteq \mathcal{T}_0, & \{f_\wedge, f_0, f_1\} &\subseteq \mathcal{MON}. \end{aligned}$$

(142) Let $F \subseteq \mathtt{TF}_2$ be a complete set of functions. Prove that there is a subset T' of T such that T' is complete and $|T'| \leq 4$.
Solution. Since T is complete, Theorem 2.9.17 implies that T contains a function g_0 that does not belong to $\mathcal{T}_0$. By Exercise 140, either g_0 is not self-dual, or g_0 is neither monotonic nor a member of $\mathcal{T}_1$. Therefore, g_0 lies outside of at least two maximal clones, so we can form a subset T' of T that contains g_0 and at most three other functions such that T' is not contained in any of the maximal clones.

(143) Let $F \subseteq \mathtt{TF}_*$ be a set of functions. Prove that $\widehat{\widehat{F}} = \mathtt{TF}_*$ if and only if F is complete and contains a 0-ary truth function.

Solution. Suppose that F is complete, that is, $\widehat{\widehat{F}} \supseteq \mathtt{TF}$, and F contains a 0-ary truth function f. Since F is complete, $f_\neg \in \widehat{\widehat{F}}$ and, therefore, $g = f_\neg(f) \in \widehat{\widehat{F}}$. Since f and g are the two 0-ary truth functions, and both are in $\widehat{\widehat{F}}$, $\widehat{\widehat{F}} = \mathtt{TF}_*$.
Now suppose that $\widehat{\widehat{F}} = \mathtt{TF}_*$. Then F is complete. If $F \subseteq \mathtt{TF}$, then, by Theorem 2.9.28, we would have $\widehat{\widehat{F}} = \widehat{F} \subseteq \mathtt{TF}$. Thus, F must contain a 0-ary truth function.

(144) Show that every monotonic truth function that is not a constant function belongs to $\mathcal{T}_0 \cap \mathcal{T}_1$.

(145) Show that the set of all monotonic, nonconstant truth functions is a clone.
Hint. Prove that this set equals $\mathcal{MON} \cap \mathcal{T}_0 \cap \mathcal{T}_1$.

(146) Show that the clone generated by $\{f_\wedge, f_\vee\}$ is the clone of all monotonic, nonconstant functions.

(147) Prove that the set $F = \{f_\top, f_\bot, f_\wedge, f_\vee\}$ is a minimal generating set for the extended clone $\mathcal{MON}_*$ in the sense that no proper subset of F generates $\mathcal{MON}_*$.

Complete Sets of Connectives

(148) Prove that the mappings $\Phi : \text{PLFORM} \longrightarrow \text{PLFORM}_F$ and $\Psi : \text{PLFORM}_F \longrightarrow \text{PLFORM}$ of Example 2.10.2 are inverse bijections.

(149) Prove that the mappings Φ, Ψ of Example 2.10.2 preserve the semantics of formulas, that is, for every truth assignment v, $\alpha \in \text{PLFORM}$ and $\beta \in \text{PLFORM}_F$, we have $v(\Phi(\alpha)) = v(\alpha)$ and $v(\Psi(\beta)) = v(\beta)$.

Let $F = \{f_C | C \in \mathbf{C}\}$ be a collection of truth functions indexed by a set of connective symbols $\mathbf{C}$. F will remain fixed throughout this discussion.

Let S be a set of statement variables and let PTA_S be the set of all partial truth assignments defined on subsets of the set S, i.e.,

$$\text{PTA}_S = \bigcup_{S_1 \subseteq S} \text{TA}_{S_1}.$$

Similarly, we define PTA_S^+ to be the set of all partial truth assignments defined on nonempty subsets of S.

If $S = SV(\varphi)$ for some formula $\varphi \in \text{PLFORM}_F$, we shall use the alternative notation $\text{PTA}(\varphi)$ for $\text{PTA}_{\text{SV}(\varphi)}$.

Partial truth assignments over a finite set of statement variables $S = \{p_{i_0}, \ldots, p_{i_{n-1}}\}$, where $i_0 < \cdots < i_{n-1}$, can be represented as sequences over the set $\mathbf{Bool} \cup \{-\}$ as follows. If $v \in \text{PTA}_S$, define the sequence $q_v : \{0, \ldots, n-1\} \longrightarrow \mathbf{Bool} \cup \{-\}$ by

$$q_v(k) = \begin{cases} v(p_{i_k}) & \text{if } p_{i_k} \in \text{Dom}(v) \\ - & \text{otherwise} \end{cases}$$

for $0 \leq k \leq n-1$.

For instance, if $v \in \mathrm{PTA}_S$, where $S = \{p_2, p_5, p_6, p_8\}$ and $\mathrm{Dom}(v) = \{p_2, p_6\}$, then $q_v = (v(p_2), -, v(p_6), -)$.

For $v, w \in \mathrm{PTA}_S$, define $v \leq w$ if $v \subseteq w$. In other words, $v \leq w$ if $\mathrm{Dom}(v) \subseteq \mathrm{Dom}(w)$ and $v = w\restriction_{\mathrm{Dom}(v)}$. It is immediate that $\leq$ is a partial order on PTA_S. The least element of the poset $(\mathrm{PTA}_S, \leq)$ is the empty truth assignment $v_\emptyset$.

If $\varphi \in \mathrm{PLFORM}_F$, define the set of partial truth assignments $\mathrm{PTA}^b(\varphi)$ by

$$\mathrm{PTA}^b(\varphi) = \{v \in \mathrm{PTA}(\varphi) \mid w(\varphi) = b \text{ for all } w \in \mathrm{TA}_{\mathrm{SV}(\varphi)} \text{ such that } v \leq w\}$$

for $b \in \mathbf{Bool}$. In other words, $\mathrm{PTA}^b(\varphi)$ consists of those partial truth assignments of φ all of whose extensions to $SV(\varphi)$ satisfy $b\varphi$.

Let $u, v \in \mathrm{PTA}_S$ be two partial truth assignments such that $\mathrm{Dom}(u) = \mathrm{Dom}(v)$ and for some variable $q \in \mathrm{Dom}(u) = \mathrm{Dom}(v)$ we have $u(p) = v(p)$ for all $p \in \mathrm{Dom}(u) = \mathrm{Dom}(v)$ with $p \neq q$, and $u(q) \neq v(q)$. Define the partial truth assignment w by $\mathrm{Dom}(w) = \mathrm{Dom}(u) - \{q\} = \mathrm{Dom}(v) - \{q\}$ and $w(p) = u(p) = v(p)$ for every $p \in \mathrm{Dom}(w)$. We will denote w by $u \star v$.

Another useful order on PTA_S is introduced starting from the total order $\sqsubseteq_0$ defined on $\mathbf{Bool} \cup \{-\}$ by $\mathbf{T} \sqsubseteq_0 \mathbf{F} \sqsubseteq_0 -$. Namely, we define $v \sqsubseteq w$ if q_v is less than or equal to q_w in the lexicographic order on $(\mathbf{Bool} \cup \{-\})^{|S|}$ induced by $\sqsubseteq_0$. For instance, if $q_v = (-, \mathbf{F}, \mathbf{F})$ and $q_w = (-, -, \mathbf{T})$, then $v \sqsubseteq w$.

(150) Show that if $v, w \in \mathrm{PTA}_S$, then $v \leq w$ implies $w \sqsubseteq v$.

(151) Show that if S, S' are two sets of statement variables such that $v, w \in \mathrm{PTA}_S \cap \mathrm{PTA}_{S'}$, then $v \leq w$ in the poset $(\mathrm{PTA}_S, \leq)$ if and only if $v \leq w$ in the poset $(\mathrm{PTA}_{S'}, \leq)$. Prove that a similar result holds when $\leq$ is replaced by $\sqsubseteq$.

(152) Show that if $v_0, v_1 \in \mathrm{PTA}^b(\varphi)$ and $v_0 \star v_1$ is defined, then $v_0 \star v_1 \in \mathrm{PTA}^b(\varphi)$. Furthermore, $v_0 \star v_1 = \inf\{v_0, v_1\}$ in the poset $(\mathrm{PTA}(\varphi), \leq)$.

Solution. Suppose that $v_0, v_1 \in \mathrm{PTA}^b(\varphi)$ and let $v = v_0 \star v_1$. Then, $\mathrm{Dom}(v) = \mathrm{Dom}(v_0) - \{q\} = \mathrm{Dom}(v_1) - \{q\}$, where q is a statement variable from $\mathrm{Dom}(v_0) = \mathrm{Dom}(v_1)$ and $v_0(q) \neq v_1(q)$. If w is a truth assignment such that $\mathrm{Dom}(w) = SV(\varphi)$ and $v \leq w$, then $v(p) = w(p)$ for every $p \in \mathrm{Dom}(v)$ and, since $v_0(q) \neq v_1(q)$, we have either $w(q) = v_0(q)$ or $w(q) = v_1(q)$.

In the first case, w is an extension of v_0; in the second, w is an extension of v_1, so in either case, $w(\varphi) = b$. The second part of the statement is left to the reader.

The *rank* of a partial truth assignment v with finite domain is $r(v) = |\mathrm{Dom}(v)|$.

The *n-th layer of the* $\mathbf{T}$*-partial truth assignments of* φ is the set

$$\mathrm{PTA}^{\mathbf{T}}_n(\varphi) = \{v \in \mathrm{PTA}^{\mathbf{T}}(\varphi) \mid r(v) = n\}$$

for $0 \le n \le |SV(\varphi)|$.

The *n-th layer of the* $\mathbf{F}$*-truth assignments of* φ is the set

$$\mathrm{PTA}^{\mathbf{F}}_n(\varphi) = \{v \in \mathrm{PTA}^{\mathbf{F}}(\varphi) \mid r(v) = n\}$$

for $0 \le n \le |SV(\varphi)|$.

Clearly, if S is a finite set of statement variables, $v, v' \in \mathrm{PTA}_S$ and $v \le v'$, then $r(v) \le r(v')$.

(153) Prove that

$$\mathrm{PTA}^b_{n-1}(\varphi) = \{u \star v \mid u, v \in \mathrm{PTA}^b_n(\varphi) \text{ and } u \star v \text{ is defined}\}$$

for $b \in \mathbf{Bool}$ and $1 \le n \le |SV(\varphi)|$.

Solution. Supplement 152 implies that

$$\{u \star v \mid u, v \in \mathrm{PTA}^b_n(\varphi) \text{ and } u \star v \text{ is defined}\} \subseteq \mathrm{PTA}^b_{n-1}(\varphi)$$

for $b \in \mathbf{Bool}$ and $1 \le n \le |SV(\varphi)|$. Therefore, we need to prove only the reverse inclusion.

Let $v \in \mathrm{PTA}^b_{n-1}(\varphi)$. Every extension w of v to $SV(\varphi)$ belongs to $\mathrm{PTA}^b(\varphi)$. Since $r(v) < |SV(\varphi)|$, there exists a statement variable q such that $q \notin \mathrm{Dom}(v)$. Let u_0, u_1 be two partial truth assignments such that $\mathrm{Dom}(u_0) = \mathrm{Dom}(u_1) = \mathrm{Dom}(v) \cup \{q\}$. These assignments are defined by

$$u_0(p) = \begin{cases} v(p) & \text{if } p \in \mathrm{Dom}(v) \\ \mathbf{F} & \text{if } p = q \end{cases}$$

and

$$u_1(p) = \begin{cases} v(p) & \text{if } p \in \mathrm{Dom}(v) \\ \mathbf{T} & \text{if } p = q. \end{cases}$$

It is immediate that both u_0 and u_1 belong to $\mathrm{PTA}^b_n(\varphi)$, $u_0 \star u_1$ is defined and $u_0 \star u_1 = v$. This shows the reverse inclusion.

(154) Consider the formula $\varphi = ((p_0 \wedge p_1) \vee ((\neg p_0) \wedge p_2) \vee (p_1 \wedge p_2))$ whose truth table is as follows:

$v(p_0)$	$v(p_1)$	$v(p_2)$	$v(\varphi)$
F	**F**	**F**	**F**
F	**F**	**T**	**T**
F	**T**	**F**	**F**
F	**T**	**T**	**T**
T	**F**	**F**	**F**
T	**F**	**T**	**F**
T	**T**	**F**	**T**
T	**T**	**T**	**T**

Construct the Hasse diagrams of $(\text{PTA}^{\mathbf{T}}(\varphi), \leq)$ and $(\text{PTA}^{\mathbf{F}}(\varphi), \leq)$.

Solution. The Hasse diagrams are given in Figures 2.32 and 2.33.

Observe that out of the twelve partial truth assignments $v \in \text{PTA}(\varphi)$ whose domain contains two variables only six are obtained by using the $\star$ operation: $(\mathbf{F}, -, \mathbf{T})$, $(-, \mathbf{T}, \mathbf{T})$, $(\mathbf{T}, \mathbf{T}, -)$ in $\text{PTA}^{\mathbf{T}}(\varphi)$ and $(\mathbf{F}, -, \mathbf{F})$, $(-, \mathbf{F}, \mathbf{F})$, $(\mathbf{T}, \mathbf{F}, -)$ in $\text{PTA}^{\mathbf{F}}(\varphi)$. A partial truth assignment such as $(\mathbf{T}, -, \mathbf{F})$ is neither in $\text{PTA}^{\mathbf{T}}(\varphi)$ nor in $\text{PTA}^{\mathbf{F}}(\varphi)$ because its extensions u, v defined by $(\mathbf{T}, \mathbf{F}, \mathbf{F})$ and $(\mathbf{T}, \mathbf{T}, \mathbf{F})$, respectively, are such that $u(\varphi) = \mathbf{F}$ and $v(\varphi) = \mathbf{T}$.

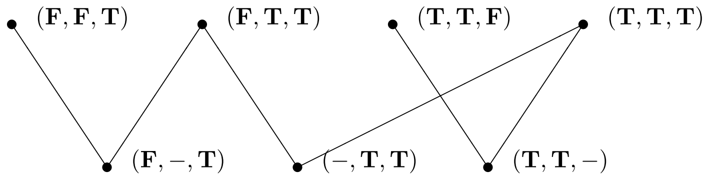


Fig. 2.32. Hasse diagram of the poset $(\text{PTA}^{\mathbf{T}}(\varphi), \leq)$.

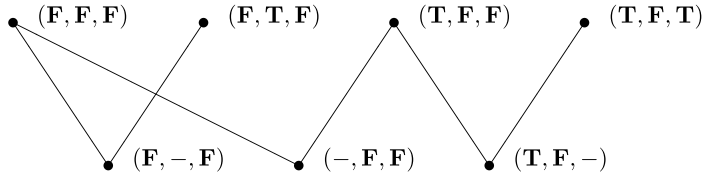


Fig. 2.33. Hasse diagram of the poset $(\text{PTA}^{\mathbf{F}}(\varphi), \leq)$.

(155) Show that if $v \in \mathrm{PTA}^{\mathbf{T}}(\varphi)$ and $v' \in \mathrm{PTA}^{\mathbf{F}}(\varphi)$, then there exists a statement variable p such that $p \in \mathrm{Dom}(v) \cap \mathrm{Dom}(v')$ and $v(p) \neq v'(p)$.
Solution. Let $v \in \mathrm{PTA}^{\mathbf{T}}(\varphi)$ and $v' \in \mathrm{PTA}^{\mathbf{F}}(\varphi)$ be two partial truth assignments. If v and v' agreed on all variables in $\mathrm{Dom}(v) \cap \mathrm{Dom}(v')$, then v and v' would have a common extension w with domain $SV(\varphi)$, which is impossible (because this would imply both $w(\varphi) = \mathbf{T}$ and $w(\varphi) = \mathbf{F}$). Thus, the desired conclusion holds.

Since $\mathrm{PTA}(\varphi)$ is a finite set for every formula $\varphi \in \mathrm{PLFORM}_F$, it is clear that for every $w \in \mathrm{PTA}^b(\varphi)$ there is a minimal truth assignment $v \in \mathrm{PTA}^b(\varphi)$ such that $w \geq v$. We will denote the set of minimal elements of $\mathrm{PTA}^b(\varphi)$ by $\mathrm{MINPTA}^b(\varphi)$ for $b \in \mathbf{Bool}$.

Recall that the domination relation between minterms was introduced in Definition 2.5.19 and the minterm μ_v was introduced in Definition 2.5.5.

(156) Let S be a nonempty set of statement variables. Define the mapping $\Psi : \mathrm{PTA}^+_S \longrightarrow \mathrm{PMINTRM}(S)$ by $\Psi(v) = \mu_v$. Prove that

(a) Ψ is a bijection,
(b) for every $v, w \in \mathrm{PTA}^+_S$, we have $v \leq w$ if and only if $\Psi(v) \geq \Psi(w)$,
(c) for every $v, w \in \mathrm{PTA}^+_S$, $v \star w$ is defined if and only if $\Psi(v) \star \Psi(w)$ is defined and, if $v \star w$ is defined, then $\Psi(v) \star \Psi(w) = \Psi(v \star w)$.

Let $F = \{f_C \mid C \in \mathbf{C}\}$ be a collection of truth functions indexed by a set of connective symbols $\mathbf{C}$. Let $C \in \mathbf{C}$ be an n-ary connective symbol. Define $\varphi_C \in \mathrm{PLFORM}_F$ by $\varphi_C = C(p_0, \ldots, p_{n-1})$, if $n > 0$, and $\varphi_C = C$, if $n = 0$.

(157) Let $S = \{p_0, \ldots, p_{n-1}\}$ and let $\psi_C \in \mathrm{PLFORM}$ be a formula using the standard connective symbols such that $SV(\psi_C) = S$ and $v(\varphi_C) = f_C(v(p_0), \ldots, v(p_{n-1})) = v(\psi_C)$, for every $v \in \mathrm{TA}_S$, where C is an n-ary connective symbol. Prove that for every $v \in \mathrm{PTA}^+_S$ and $b \in \mathbf{Bool}$, we have

(a) $v \in \mathrm{PTA}^b(\varphi_C)$ if and only if $\mu_v \in \mathrm{IMPL}(\psi^b_C)$,

(b) if f_C is not a constant function, then $v \in \text{MINPTA}^b(\varphi_C)$ if and only if μ_v is a prime implicant of ψ^b_C. (Thus, Ψ induces a bijection between $\text{MINPTA}^b(\varphi_C)$ and the prime implicants of ψ^b_C.)

A signed F-formula is a pair $(b, \varphi) \in \mathbf{Bool} \times \text{PLFORM}_F$. The set of signed F-formulas will be denoted by SPLFORM_F.

Let $\varphi = C(\varphi_0, \ldots, \varphi_{n-1})$ (or $\varphi = C$, if $n = 0$) be a formula from $\text{PLFORM}_F - SV$. If $v \in \text{PTA}(\varphi_C)$, define the set of signed formulas $K_v(\varphi) = \{v(p_i)\varphi_i \mid p_i \in \text{Dom}(v)\}$. If $b \in \mathbf{Bool}$, define a sequence of sets of signed formulas

$$\mathtt{d}(b\varphi) = (K_{v_0}(\varphi), \ldots, K_{v_{m-1}}(\varphi)),$$

where $\text{MINPTA}^b(\varphi_C) = \{v_0, \ldots, v_{m-1}\}$ and $v_0 \sqsubset \cdots \sqsubset v_{m-1}$.

Define $\mathtt{D}(b\varphi)$ as the collection of sets of signed formulas that consists of the sets that occur in the sequence $\mathtt{d}(b\varphi)$. The elements of $\mathtt{D}(b\varphi)$ are called the *constituents* of the signed formula $b\varphi$. In the special case when f_C is the constant function $f_C(a_0, \ldots, a_{n-1}) = b$, we have $\text{MINPTA}^b(\varphi_C) = \{v_\emptyset\}$ and $\text{MINPTA}^{f_\neg(b)}(\varphi_C) = \emptyset$. This gives $\mathtt{D}(b\varphi) = \{\emptyset\}$ and $\mathtt{D}(f_\neg(b)\varphi) = \emptyset$ (since $K_{v_\emptyset}(\varphi) = \emptyset$).

(158) Let $F = \{f_\neg, f_\wedge, f_\vee, f_\rightarrow, f_\leftrightarrow\}$ be the set of truth functions indexed by the standard set of connective symbols. The function Φ introduced in Example 2.10.2 can be expanded to SPLFORM by defining $\Phi(b\alpha) = b\Phi(\alpha)$ and can be further expanded to $\text{Seq}(\mathcal{P}(\text{SPLFORM}))$ by $\Phi(\Delta_0, \ldots, \Delta_{m-1}) = (\Phi(\Delta_0), \ldots, \Phi(\Delta_{m-1}))$.
Show that the constituent sequence of formulas in SPLFORM and SPLFORM_F are the same up to translation by the bijections Φ and Ψ, that is, $\mathtt{d}(b\alpha) = \Phi(\mathtt{d}(b\Psi(\alpha)))$ for every $b\alpha$ in SPLFORM_F that is not a signed variable.

(159) Prove that a truth assignment v satisfies a signed formula $b\varphi$, where $\varphi \in \text{PLFORM}_F - SV$ if and only if it satisfies at least one of the constituents of $b\varphi$.
Solution. Suppose that v is a truth assignment that satisfies $b\varphi$, where $\varphi = C(\varphi_0, \ldots, \varphi_{n-1})$, or $\varphi = C$, that is,

$$f_C(v(\varphi_0), \ldots, v(\varphi_{n-1})) = b.$$

Define the partial truth assignment u on $\{p_0, \ldots, p_{n-1}\}$ by $u(p_j) = v(\varphi_j)$ for $0 \leq j \leq n-1$. We have

$u(\varphi_C) = f_C(u(p_0), \ldots, u(p_{n-1})) = b$, so $u \in \mathrm{PTA}^b(\varphi_C)$. This implies the existence of $w \in \mathrm{MINPTA}^b(\varphi_C)$ such that $w \leq u$. If $\mathrm{Dom}(w) = \{p_{i_0}, \ldots, p_{i_{l-1}}\}$, we have $w(p_{i_j}) = u(p_{i_j}) = v(\varphi_{i_j})$ for $0 \leq j \leq l-1$. Since

$$K_w(\varphi) = \{w(p_{i_0})\varphi_{i_0}, \ldots, w(p_{i_{l-1}})\varphi_{i_{l-1}}\},$$

it follows that v satisfies every signed formula $w(p_{i_j})\varphi_{i_j}$ for $0 \leq j \leq l-1$, so v satisfies the constituent $K_w(\varphi)$ of $b\varphi$. Conversely, let v be a truth assignment that satisfies a constituent $K_w(\varphi)$ for some $w \in \mathrm{MINPTA}^b(\varphi_C)$. If

$$K_w(\varphi) = \{w(p_{i_0})\varphi_{i_0}, \ldots, w(p_{i_{l-1}})\varphi_{i_{l-1}}\},$$

we have $v(\varphi_{i_j}) = w(p_{i_j})$ for $0 \leq j \leq l-1$. Define the partial truth assignment u as an extension of w to $SV(\varphi_C) = \{p_0, \ldots, p_{n-1}\}$ as $u(p_i) = v(\varphi_i)$ for $0 \leq i \leq n-1$. By the definition of $\mathrm{MINPTA}^b(\varphi_C)$, we have $u(\varphi_C) = b$, that is, $f_C(u(p_0), \ldots, u(p_{n-1})) = b$. This, in turn, implies

$$\begin{aligned} v(\varphi) &= f_C(v(\varphi_0), \ldots, v(\varphi_{n-1})) \\ &= f_C(u(p_0), \ldots, u(p_{n-1})) \\ &= u(\varphi_C) = b, \end{aligned}$$

which shows that v satisfies $b\varphi$.

(160) Let $\varphi \in \mathrm{PLFORM}_F - SV$ and let $K \in \mathtt{D}(b\varphi)$ and $H \in \mathtt{D}(f_\neg(b)\varphi)$ for some $b \in \mathbf{Bool}$. Show that there exists an immediate subformula γ of φ such that

$$\{\mathbf{T}\gamma, \mathbf{F}\gamma\} \subseteq K \cup H.$$

Solution. This is an immediate consequence of Supplement 155.

(161) Let $\varphi = C(\varphi_0, \ldots, \varphi_{n-1})$ or $\varphi = C$ and assume that

$$\mathtt{d}(b\varphi) = (K_0, \ldots, K_{m-1}).$$

Prove that if $b_j\psi_j \in K_j$ is a signed formula for each j, $0 \leq j \leq m-1$ such that the set $L = \{b_j\psi_j \mid 0 \leq j \leq m-1\}$ does not contain both $\mathbf{T}\gamma$ and $\mathbf{F}\gamma$ for any formula γ, then there is a constituent H of $f_\neg(b)\varphi$ such that $H \subseteq \{f_\neg(b_j)\psi_j \mid 0 \leq j \leq m-1\}$.

Solution. Let $\mathrm{MINPTA}^b(\varphi_C) = \{v_0, \ldots, v_{m-1}\}$, where $v_0 \sqsubset v_1 \sqsubset \cdots \sqsubset v_{m-1}$ and $K_i = K_{v_i}(\varphi)$ for $0 \leq i \leq m-1$. For each j, $0 \leq j \leq m-1$, there is i_j with $b_j = v_j(p_{i_j})$, and $\psi_j = \varphi_{i_j}$. We claim that the set of pairs $\{(p_{i_j}, b_j) \mid 0 \leq j \leq m-1\}$ is a partial truth assignment. Indeed, if $p_{i_j} = p_{i_k}$ (that is, $i_j = i_k$) and $b_j \neq b_k$, then we have $b_j\varphi_{i_j}$ and $b_k\varphi_{i_k} = b_k\varphi_{i_j}$ in L, which is in conflict with the assumption that L does no contain both $\mathbf{T}\alpha$ and $\mathbf{F}\alpha$ for any formula α. Therefore, $w = \{(p_{i_j}, f_\neg(b_j)) \mid 0 \leq j \leq m-1\}$ is a partial truth assignment. We further claim that $w \in \mathrm{PTA}^{f_\neg(b)}(\varphi_C)$. Indeed, let z be an arbitrary extension of w to $SV(\varphi_C)$. Since $z(p_{i_j}) = w(p_{i_j}) = f_\neg(b_j) = f_\neg(v_j(p_{i_j}))$ for each $v_j \in \mathrm{MINPTA}^b(\varphi_C)$, we have $v_j \not\leq z$ for each such v_j. This implies that $z \notin \mathrm{PTA}^b(\varphi_C)$. Therefore, z satisfies $f_\neg(b)\varphi_C$ and we may conclude that $w \in \mathrm{PTA}^{f_\neg(b)}(\varphi_C)$. In turn, this means that there exists $w' \in \mathrm{MINPTA}^{f_\neg(b)}(\varphi_C)$ such that $w' \leq w$. If $H = K_{w'}(\varphi)$, we obtain the desired constituent of $f_\neg(b)\varphi$.

The notions of maximally satisfiable, $\mathbf{T}$-downward closed, $\mathbf{F}$-downward closed, downward closed, $\mathbf{T}$-upward closed, $\mathbf{F}$-upward closed, upward closed, and saturated extend obviously from sets of signed formulas to sets of signed F-formulas. Also, the notion of Hintikka set can be naturally extended to sets of signed F-formulas.

(162) Prove that every Hintikka set of signed F-formulas is satisfiable.

The reader should note the parallels between Supplements 104 and 160 and between Supplements 105 and 161. Observe, however, that in Supplement 105 we have $H = \{f_\neg(b_j)\psi_j \mid 0 \leq j \leq m-1\}$ while in Supplement 161, $H \subseteq \{f_\neg(b_j)\psi_j \mid 0 \leq j \leq m-1\}$. The latter inclusion suffices for the following exercise.

(163) Formulate and prove analogues of Supplements 101, 102, 103, 106, 107, 110, and 112 for signed F-formulas.

(164) Let F be a set of truth functions, $F = \{f_C \mid C \in \mathbf{C}\}$, and let $\varphi_C = C(p_0, \ldots, p_{m-1})$. For the signed formula $b\varphi \in \mathrm{SPLFORM}_F$ given by $\varphi = C(\varphi_0, \ldots, \varphi_{n-1})$, define $\mathtt{d}'(b\varphi) = (K_{v_0}(\varphi), \ldots, K_{v_{m-1}}(\varphi))$, where $\{v_0, \ldots, v_{m-1}\} = \mathrm{TA}^b(\varphi_C)$ and $v_0 \sqsubset \cdots \sqsubset v_{m-1}$. Also, let $\mathtt{D}'(b\varphi)$ be the collection of sets that occur in $\mathtt{d}'(b\varphi)$:

(a) Prove that a truth assignment v satisfies $b\varphi$ if and only if it satisfies an element of $\mathtt{D}'(b\varphi)$. Thus, the members of $\mathtt{D}'(b\varphi)$ can be regarded as an alternative to constituents of signed formulas.

(b) Show that the statement contained in Exercise 158 does not carry over to $\mathtt{d}'$.

(c) Prove that statement of Exercise 160 remains true when $\mathtt{D}$ is replaced by $\mathtt{D}'$.

(d) Show that Exercise 161 fails with $\mathtt{d}$ replaced by $\mathtt{d}'$, when $F = \{f_{\mathrm{maj}}\}$.

(165) Let $F = \{f_C\}$, where $f_C(x_0, x_1, x_2) = \mathbf{T}$ if and only if $x_0 = x_1$ or $x_0 = x_2$:

(a) Starting from Example 2.5.39, show that $|\mathtt{d}(\mathbf{T}\varphi)| = 6$, where $\varphi = f_C(\varphi_0, \varphi_1, \varphi_2)$.

(b) Using the same example, show how to define a "constituent sequence" of length three for $\mathbf{T}\varphi$ such that the statement of Exercise 159 remains true.

We will now show how to extend the notion of constituent of an unsigned formula to unsigned formulas involving an arbitrary set of connectives that contains negation. In other words, if $F = \{f_C \mid C \in \mathbf{C}\}$, then for some fixed $N \in \mathbf{C}$, $f_N = f_\neg$. We will write a formula $N(\varphi)$ as $(\neg\varphi)$. A formula $\varphi \in \mathrm{PLFORM}_F$ is *negative* if it has the form $N(\psi)$; otherwise, we say that φ is *positive*. We extend the notation introduced in Definition 2.5.1 to formulas in PLFORM_F for such an F, namely, $\varphi^{\mathbf{T}} = \varphi$ and $\varphi^{\mathbf{F}} = (\neg\varphi)$ for every $\varphi \in \mathrm{PLFORM}_F$. Also, if $\varphi \in \mathrm{PLFORM}_F$, we define $\overline{\varphi}$ to be $N(\varphi)$ if φ is positive and to be ψ if $\varphi = N(\psi)$.

Let $C \in \mathbf{C}$ be a connective symbol and let $\varphi = C(\varphi_0, \ldots, \varphi_{n-1})$ (or $\varphi = C$ if $n = 0$) be a formula in $\mathrm{PLFORM}_F - SV$. For $v \in \mathrm{PTA}(\varphi_C)$, define the set of formulas $H_v(\varphi) = \{\varphi_i^{v(p_i)} \mid p_i \in \mathrm{Dom}(v)\}$. If $C \neq N$, then we define

$$\mathtt{d}(\varphi) = (H_{v_{i_0}}(\varphi), \ldots, H_{v_{i_{m-1}}}(\varphi)),$$

where $\mathrm{MINPTA}^{\mathbf{T}}(\varphi_C) = \{v_{i_0}, \ldots, v_{i_{m-1}}\}$ and $v_{i_0} \sqsubset \cdots \sqsubset v_{i_{m-1}}$. If $\varphi = N(\psi)$ where $\psi = C'(\psi_0, \ldots, \psi_{m-1})$, then we define

$$\mathtt{d}(\varphi) = (H_{v_{j_0}}(\psi), \ldots, H_{v_{j_{l-1}}}(\psi)),$$

where $\text{MINPTA}^{\mathbf{F}}(\varphi_{C'}) = \{v_{j_0}, \ldots, v_{j_{l-1}}\}$ and $v_{j_0} \sqsubset \cdots \sqsubset v_{j_{l-1}}$. Note that $\mathtt{d}(\varphi)$ is not defined when $\varphi = p$ or $\varphi = (\neg p)$ for some $p \in SV$.

When $\mathtt{d}(\varphi)$ is defined, we define $\mathtt{D}(\varphi)$ to be the collection of sets that occur in $\mathtt{d}(\varphi)$. The members of $\mathtt{D}(\varphi)$ are called the constituents of φ.

(166) Verify that $\mathtt{D}(N(N(\varphi))) = \{\{\varphi\}\}$ for every formula $\varphi \in \text{PLFORM}_F$.

(167) Let $\mathbf{C} = \{\neg, \vee, \wedge, \rightarrow, \leftrightarrow\}$ and $F = \{f_C \mid C \in \mathbf{C}\}$. Show that the constituent sequence of formulas in PLFORM and PLFORM_F are the same up to translation by the bijections Φ and Ψ, that is, $\mathtt{d}(\alpha) = \Phi(\mathtt{d}(\Psi(\alpha)))$ for every α in PLFORM_F that is not a variable. Here, Φ, Ψ are the bijections mentioned in Exercise 148.

(168) Let $\mathbf{C}$ be a set of connective symbols that contains in addition to N, a symbol N' such that $f_{N'} = f_{\neg}$. Verify that $\mathtt{D}(N'(\varphi)) = \{\{N(\varphi)\}\}$.

(169) Let φ be a positive formula in $\text{PLFORM}_F - SV$ and let $K \in \mathtt{D}(\varphi)$ and $H \in \mathtt{D}(N(\varphi))$. Show that there exists an immediate subformula γ of φ such that

$$\{\gamma, N(\gamma)\} \subseteq K \cup H.$$

Solution. This is an immediate consequence of Supplement 155.

(170) Let φ be $C(\varphi_0, \ldots, \varphi_{n-1})$, or $N(C(\varphi_0, \ldots, \varphi_{n-1}))$ (or C, or $N(C)$ when $n = 0$) for some formulas $\varphi_0, \ldots, \varphi_{n-1}$ and some connective symbol $C \in \mathbf{C}$ other than N. Suppose that $\mathtt{d}(\varphi) = (H_0, \ldots, H_{m-1})$ and that $L = \{\psi_i \mid 0 \le i \le m-1\}$ is a set of formulas such that $\psi_i \in H_i$ for $0 \le i \le m-1$ and L does not contain any pair of formulas $\gamma, (\neg\gamma)$. Prove that there exist formulas θ_i such that for every i, $0 \le i \le m-1$, either $\theta_i = (\neg\psi_i)$ or $\psi_i = (\neg\theta_i)$ holds and $\{\theta_i \mid 0 \le i \le m-1\}$ contains a constituent H of $\overline{\varphi}$.

Solution. Let $b = \mathbf{T}$ and $\gamma = \varphi$ if $\varphi = C(\varphi_0, \ldots, \varphi_{n-1})$ or $\varphi = C$, and let $b = \mathbf{F}$ and $\gamma = \overline{\varphi}$ if $\varphi = N(C(\varphi_0, \ldots, \varphi_{n-1}))$ or $\varphi = N(C)$. For each of the constituents H_j in $\mathtt{d}(\varphi) = (H_0, \ldots, H_{m-1})$, we have $H_j = H_{v_j}(\gamma)$, where $\text{MINPTA}^b(\varphi_C) = \{v_0, \ldots, v_{m-1}\}$ and $v_0 \sqsubset \cdots \sqsubset v_{m-1}$. For each of the formulas ψ_j, there is k_j with $0 \le k_j \le n-1$

such that $\psi_j = \varphi_{k_j}^{v_j(p_{k_j})}$. Observe that if $k_j = k_{j'}$, then $v_j(p_{k_j}) = v_{j'}(p_{k_{j'}})$. Indeed, if $v_j(p_{k_j}) \neq v_{j'}(p_{k_{j'}})$, this would imply that the set L contains both a formula and its negation represented by ψ_j and $\psi_{j'}$. Therefore, the set $\{(p_{k_j}, v_j(p_{k_j})) \mid 0 \leq j \leq m-1\}$ is a partial truth assignment and this implies that $w = \{(p_{k_j}, f_\neg(v_j(p_{k_j}))) \mid 0 \leq j \leq m-1\}$ is a partial truth assignment. We further claim that $w \in \mathrm{PTA}^{f_\neg(b)}(\varphi_C)$. Indeed, let z be an arbitrary extension of w to $SV(\varphi_C)$. Since $z(p_{k_j}) = w(p_{k_j}) = f_\neg(v_j(p_{k_j}))$ for each j, $0 \leq j \leq m-1$, we have $v_j \not\leq z$ for each such j. This implies that $z \notin \mathrm{PTA}^b(\varphi_C)$. Therefore, $z(\varphi_C) = f_\neg(b)$ and we may conclude that $w \in \mathrm{PTA}^{f_\neg(b)}(\varphi_C)$. In turn, this means that there exists $w' \in \mathrm{MINPTA}^{f_\neg(b)}(\varphi_C)$ such that $w' \leq w$. Let

$$\begin{aligned} H = H_{w'}(\gamma) &= \{\varphi_l^{w'(p_l)} \mid p_l \in \mathrm{Dom}(w')\} \\ &= \{\varphi_{k_j}^{f_\neg(v_j(p_{k_j}))} \mid p_{k_j} \in \mathrm{Dom}(w')\}. \end{aligned}$$

Since $C \neq N$, it is clear that $H \in \mathtt{D}(\overline{\varphi})$. Define

$$\theta_j = \begin{cases} \varphi_{k_j}^{f_\neg(v_j(p_{k_j}))} & \text{if } p_{k_j} \in \mathrm{Dom}(w') \\ (\neg\psi_j) & \text{otherwise.} \end{cases}$$

We have $H \subseteq \{\theta_j \mid 0 \leq j \leq m-1\}$. Suppose that $p_{k_j} \in \mathrm{Dom}(w')$. If $v_j(p_{k_j}) = \mathbf{T}$, we have $\psi_j = \varphi_{k_j}$ and $\theta_j = (\neg\varphi_{k_j}) = (\neg\psi_j)$. If $v_j(p_{k_j}) = \mathbf{F}$, we have $\psi_j = (\neg\varphi_{k_j})$ and $\theta_j = \varphi_{k_j}$, so $\psi_j = (\neg\theta_j)$. Thus, the formulas θ_j are as desired.

(171) Formulate analogues for (unsigned) F-formulas of the concepts of truth set, Hintikka set, and the upwards and downwards closure properties and examine the possibility of extending results presented in Sections 2.7 and 2.12 from the standard connectives to arbitrary sets of connectives that contain a negation-like symbol N.

Circuits and Truth Functions

(172) Let $\mathcal{K} = (G, \varpi, q)$ be an (F, m)-circuit, where $G = (V, U)$:

(a) Prove that if $(\mathtt{v}, \mathtt{v}')$ is an edge in G, then $\mathtt{depth}(\mathtt{v}) < \mathtt{depth}(\mathtt{v}')$.

(b) Prove that for every node $\mathtt{v}$ of $\mathcal{K}$, $\mathtt{depth}(\mathtt{v})$ is the length of the longest path that joins an input to $\mathtt{v}$.

(c) Prove that the depth of a vertex of $\mathcal{K}$ can be defined inductively as follows:

i. If $d_G^-(\mathtt{v}) = 0$, then $\mathtt{depth}(\mathtt{v}) = 0$.

ii. Suppose that we have defined the depth of all vertices of $V_\mathtt{v} - \{\mathtt{v}\}$, where $d_G^-(\mathtt{v}) > 0$; define $\mathtt{depth}(\mathtt{v}) = 1 + \max\{\mathtt{depth}(y) \mid y \in V_\mathtt{v} - \{\mathtt{v}\}\}$.

(d) Prove that the last condition of the previous inductive definition can be replaced by the following:

Suppose that we have defined the depth of all vertices of $\mathtt{v}'$ with $(\mathtt{v}', \mathtt{v}) \in U$; define

$$\mathtt{depth}(\mathtt{v}) = 1 + \max\{\mathtt{depth}(y) \mid (y, \mathtt{v}) \in U\}.$$

(173) Let $\mathcal{K}$ be a single-output (F, m)-circuit that computes a function $g \in \mathtt{TF}_m$ and let $\wp : \{0, \ldots, m-1\} \longrightarrow \{0, \ldots, n-1\}$. Consider the (F, n)-circuit $\mathcal{K}^\wp$ obtained from $\mathcal{K}$ by replacing each label p_i of an input node by $p_{\wp(i)}$:

(a) Prove that for every vertex $\mathtt{v}$ of $\mathcal{K}$, we have $\Theta_{\mathcal{K}^\wp}(\mathtt{v}) = (\Theta_\mathcal{K}(\mathtt{v}))^\wp$.

(b) Prove that the function computed by $\mathcal{K}^\wp$ is $g^\wp$.

Hint. Use Exercise 135 for Part (a).

(174) Let F be a set of truth functions. Determine the Boolean functions that belong to the domain of COMB_F^0.

(175) Show that the circuits $\mathcal{K}_n$ defined in Lemma 2.11.36 can be built such that their depth is $O(\log n)$.

(176) Prove that $\mathsf{COMB}_{\mathtt{TF}_2}^\infty(f) \geq n-1$ for every nondegenerate function $f \in \mathtt{TF}_n$.

Solution. Let $\mathcal{K}$ be a fan-out ∞, $\mathtt{TF}_2$-optimal circuit for f, that is $||\mathcal{K}|| = \mathsf{COMB}_{\mathtt{TF}_2}^\infty(f)$. The statement holds trivially for $n = 0, 1$. Therefore, we may assume that $n \geq 2$. Note that for each i with $0 \leq i \leq n-1$, there is an input node labeled with p_i whose out-degree is at least 1, since otherwise, the f would be degenerate. Further, each gate except for the output gate must have fan-out at least 1 since otherwise $\mathcal{K}$ would not be optimal. Thus, the number of edges of $\mathcal{K}$ is at least $n + ||\mathcal{K}|| - 1$. Since each gate has fan-in 2, the number of edges is $2||\mathcal{K}||$. So, $2||\mathcal{K}|| \geq n + ||\mathcal{K}|| - 1$, which gives the desired inequality.

(177) For each $n \geq 1$, give examples of non-degenerate n-ary truth functions f such that $\mathsf{COMB}^{\infty}_{\mathbf{TF}_2}(f) = n-1$ (i.e., such that the lower bound of Supplement 176 is attained).

(178) Give a direct proof that the family of truth functions $\mathcal{F}_2$ of Example 2.11.42 has g-size F-circuits, where $F = \{\wedge, \vee\}$ and

$$g(n) = \begin{cases} n^2 - n - 1 & \text{if } n \geq 2 \\ 0 & \text{if } n < 2. \end{cases}$$

Hint. Start from the formula

$$\varphi_n = \bigvee_{0 \leq i < j \leq n-1} (p_i \wedge p_j).$$

(179) Prove the following extension of Theorem 2.11.40. For all $n \geq 3$ and $2 \leq k \leq n-1$, we have $\mathsf{COMB}^{\infty}_{\mathbf{TF}_2}(\mathrm{th}_{k,n}) \geq 2n - 3$.
Hint. Use induction on n and an argument similar to the one used in Theorem 2.11.40.

Let F_{nl} be the set of nonlinear binary truth functions. By Exercise 130, F_{nl} consists of 8 functions, which by Example 2.8.33 are the binary truth functions distinct from the constants, the projections and their negations, and $f_{\leftrightarrow}$ and its negation. In the following supplements, we prove that $\mathsf{COMB}^{\infty}_{F_{nl}}(f_{+,n}) = \mathsf{COMB}^{\infty}_{F_{nl}}(f_{\leftrightarrow,n}) = 3n-3$, for $n \geq 1$, where the functions $f_{+,n}$ and $f_{\leftrightarrow,n}$ are defined by

$$f_{+,n}(a_0, \ldots, a_{n-1}) = a_0 + \cdots + a_{n-1},$$
$$f_{\leftrightarrow,n}(a_0, \ldots, a_{n-1}) = a_0 \leftrightarrow \cdots \leftrightarrow a_{n-1},$$

for $a_0, \ldots, a_{n-1} \in \mathbf{Bool}$.

(180) Let $f \in \mathtt{TF}_n$ be a truth function that is not a projection or the negation of a projection. Prove that $\mathsf{COMB}^{\infty}_{F_{nl}}(f) = \mathsf{COMB}^{\infty}_{F_{nl}}(f_{\neg}f)$.
Hint. The statement follows from Exercise 133.

(181) Prove that $\mathsf{COMB}^{\infty}_{F_{nl}}(f_{+,n}) \leq 3n - 3$, for $n \geq 1$.
Solution. Let g, h be the nonlinear functions given by $g(a,b) = a \wedge \overline{b}$ and $h(a,b) = \overline{a} \wedge b$, for $a, b \in \mathbf{Bool}$. Since $f_{+,2}(a,b) = g(a,b) \vee h(a,b)$, it follows that $\mathsf{COMB}^{\infty}_{F_{nl}}(f_{+,2}) \leq 3$. Therefore, $\mathsf{COMB}^{\infty}_{F_{nl}}(f_{+,n}) \leq 3(n-1)$, for $n \geq 2$. The inequality for $n = 1$ is trivial.

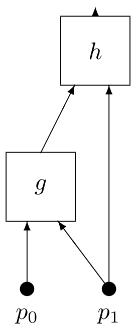


Fig. 2.34. The F_{nl}-circuit $\mathcal{K}$.

(182) Prove that $\mathsf{COMB}^{\infty}_{F_{nl}}(f_{+,2}) \geq 3$.
Solution. Since $f_{+,2}$ is linear, it is clear that $\mathsf{COMB}^{\infty}_{F_{nl}}(f_{+,2}) \geq 2$. Suppose that $\mathcal{K}$ is an F_{nl}-circuit of size 2 that computes $f_{+,2}$, say the circuit shown in Figure 2.34, which means that $f_{+,2}(a, b) = h(g(a, b), b)$ for $a, b \in \mathbf{Bool}$. Note that for every $b \in \mathbf{Bool}$, the one-argument function $f_{+,2}(x_0, b)$ depends essentially on x_0. Since h is a nonlinear function, by Exercise 132, there is $b_\star \in \mathbf{Bool}$ such that $h(x_0, b_\star)$ is a constant. Therefore, $h(g(x_0, b_\star), b_\star)$ is a constant, contradicting the fact that $f_{+,2}(x_0, b_\star)$ is not a constant. Similar arguments can be used to rule out all other F_{nl}-circuits of size 2.

(183) Prove that $\mathsf{COMB}^{\infty}_{F_{nl}}(f_{+,n}) \geq 3n - 3$ for all $n \geq 1$. (Note that in combination with Supplement 181, this statement provides a tight lower bound.)
Solution. The argument is by induction on n. The case $n = 1$ is trivial and the case $n = 2$ was dealt with in Supplement 182. Suppose $n \geq 3$ and the result is true for $n - 1$. Let $\mathcal{K}$ be an F_{nl}-circuit of minimal size that commutes $f_{+,n}$. It follows as in the proof of Theorem 2.11.40 that $\mathcal{K}$ contains a gate whose inputs are labeled by two distinct variables p_i, p_j. (Note that the functions $f_{C'_i}$ mentioned in that argument are nonlinear by Exercise 134.) Again as in the proof of Theorem 2.11.40, we claim that there is another gate $\mathtt{v}'$ that is directly connected to an input labeled by p_i. The argument this time relies on Exercise 132. Suppose that there is no such $\mathtt{v}'$. By this exercise, there is a truth value b such that $\Theta_{\mathcal{K}}(\mathtt{v})(x_0, b)$ is a constant function. By replacing all the input nodes labeled p_j by nodes

labeled by b, we obtain a circuit $\mathcal{K}'$. Since $\mathcal{K}$ computes $f_{+,n}$, $\mathcal{K}'$ computes $f_{+,n}(a_0, \ldots, a_{j-1}, b, a_{j+1}, \ldots, a_{n-1})$, which depends essentially on the ith variable. On the other hand, the function computed by $\mathcal{K}'$ cannot depend on the ith variable because no other gate than $\mathtt{v}$ is directly connected to a node labeled p_i. This contradiction shows the existence of $\mathtt{v}'$.

The gate $\mathtt{v}$ cannot be the output gate of $\mathcal{K}$ since this would imply that the function computed by the circuit depends only on x_i and x_j. We claim that there is a gate $\mathtt{v}''$ such that $\mathtt{v}'' \neq \mathtt{v}'$ and there is an edge from $\mathtt{v}$ to $\mathtt{v}''$. If not, then there would be only one edge leaving from $\mathtt{v}$ and that edge leads to $\mathtt{v}'$, meaning that the subcircuit $\mathcal{K}_{\mathtt{v}'}$ is one of the two circuits shown in Figure 2.35. Let $\mathcal{K}'$ be the circuit obtained from $\mathcal{K}_{\mathtt{v}'}$ by replacing the labels p_i and p_j by p_0 and p_1, respectively. The function $\Theta_{\mathcal{K}'}$ is different from both $f_{+,2}$ and $f_{\leftrightarrow,2}$, by the basis step and Exercise 180. Thus this function is either in F_{nl}, or is a projection, a constant function or the negation of a projection. In the first case, we could replace $\mathcal{K}_{\mathtt{v}'}$ with one F_{nl}-gate, thus contradicting the minimality of $\mathcal{K}$. In the second and third cases, $\mathcal{K}_{\mathtt{v}'}$ could be replaced by a single input node. Finally, if $\Theta_{\mathcal{K}}(\mathtt{v}')$ is the negation of a projection, then we can replace $\mathcal{K}_{\mathtt{v}}$ with a single input node labeled with a variable and modify the labels of all the gates for which $\mathtt{v}'$ is an input, using Exercise 134.

Let $\mathtt{u}, \mathtt{u}'$ be the other inputs to $\mathtt{v}'$ and $\mathtt{v}''$, respectively. A typical configuration of the nodes mentioned is shown in Figure 2.36. Note that if $\mathtt{v}'$ is the output node of $\mathcal{K}$, then $\mathtt{u}$ cannot be

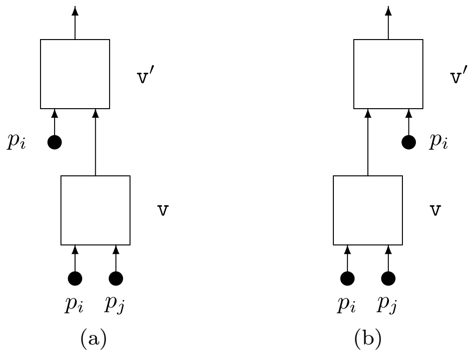


Fig. 2.35. The possible subcircuits $\mathcal{K}_{\mathtt{v}'}$.

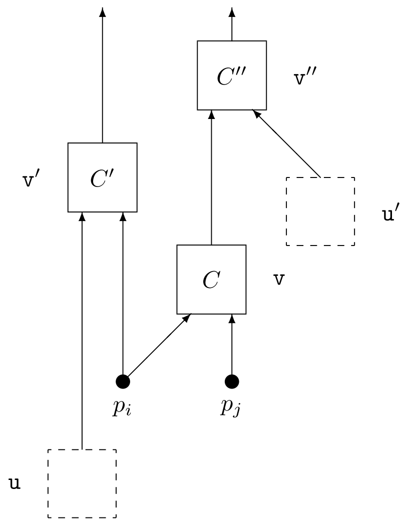


Fig. 2.36. Typical configuration of the nodes in Exercise 183.

an input node, since otherwise the function computed by the circuit would depend on only two arguments. Also, if $\mathtt{v}''$ is the output node, then $\mathtt{u}'$ cannot be an input node, since this would either imply that the output of $\mathcal{K}$ depends on only two inputs or would imply that $f_{+,3}$ is computable by two gates and therefore $f_{+,2}$ is computable by at most two gates.
Let $b \in \mathbf{Bool}$ be such that $f_C(b, a_j)$ is constant for $a_j \in \mathbf{Bool}$. The existence of b is implied by Exercise 132 since f_C is nonlinear. Let $\mathcal{H}$ be the circuit obtained from $\mathcal{K}$ by relabeling with b the input nodes labeled by p_i and relabeling each input labeled p_k with p_{k-1} for $k > i$. The circuit $\mathcal{H}$ computes either $f_{+,n-1}$ or $f_\neg f_{+,n-1}$, depending on the value of b. We claim that there is a circuit $\mathcal{H}'$ that computes the same function as $\mathcal{H}$ with three fewer gates. Since the output of $\mathtt{v}$ in $\mathcal{H}$ is constant c, one can replace $\mathtt{v}$ with an input node labeled by c, thus obtaining a circuit $\mathcal{H}_0$. As one of the inputs of $\mathtt{v}'$ in $\mathcal{H}_0$ is constant, the output of $\mathtt{v}'$ is either a constant c', $\Theta_{\mathcal{H}_0}(\mathtt{u})$ or $f_\neg \Theta_{\mathcal{H}_0}(\mathtt{u})$. In the first case, we replace $\mathtt{v}'$ by an input node labeled by c'. In the second case, if $\mathtt{v}'$ is the output node, we eliminate it and make $\mathtt{u}$ the output node. Otherwise, we eliminate $\mathtt{v}'$ and connect $\mathtt{u}$ to all former connections of $\mathtt{v}'$. In the last case, if $\mathtt{v}'$

is the output node, we make $\mathtt{u}$ the output node and replace its connective symbol $C_{\mathtt{u}}$ by $\tilde{C}_{\mathtt{u}}$, where $f_{\tilde{C}_{\mathtt{u}}}$ is the nonlinear function $f_{\neg}f_{C_{\mathtt{u}}}$. If $\mathtt{v}'$ is not the output function, the elimination of $\mathtt{v}'$ is followed by modifications of the connective symbols of all gates where $\mathtt{v}'$ is an input. This process is similar to the one in Theorem 2.11.40 and uses Exercise 134. Let $\mathcal{H}_1$ be the resulting circuit. We obtain $\mathcal{H}'$ from $\mathcal{H}_1$ by eliminating $\mathtt{v}''$ using a similar process. We have $||\mathcal{H}'|| = ||\mathcal{H}|| - 3 = ||\mathcal{K}|| - 3$ and $\mathcal{H}'$ computed either $f_{+,n-1}$ or $f_{\neg}f_{+,n-1}$. By inductive hypothesis and Exercise 180, $||\mathcal{H}'|| \geq 3(n-1) - 3$, so $||\mathcal{K}|| \geq 3n - 3$.

2.13 Bibliographical Comments

The Quine–McCluskey algorithm was introduced by Quine [31, 32] and developed by McCluskey [12]. The graphical method for simplifying Boolean expressions was created by Karnaugh [26]. Gray orderings are related to Gray codes introduced in [18].

Algorithm 2.5.65 is due to Dowling and Gallier [11].

Horn formulas were introduced in [25]. Hintikka sets were defined in [23] under the name model sets. See Smullyan [36] for truth sets and consistency properties.

Good sources for circuit complexity are the books [39,40] and the survey article [4]. Example 2.5.38 originates in [31]. Exercises 132, 181, and 183 come from [39]. The result obtained in Theorem 2.11.32 is due to Shannon and Riordan [33, 34].

Clones of binary functions were introduced and studied in [30].

The Craig Interpolation Theorem was proved in [9].

Chapter 3

Propositional Logic–Formal Systems

3.1 Introduction

This chapter concentrates on syntactical methods for proving semantic statements like $\Gamma \models \varphi$ or, equivalently, that $\Gamma \cup \{(\neg\varphi)\}$ is unsatisfiable, where Γ is a set of formulas and φ is a formula of propositional logic. As we shall see, these methods are related to a variety of formal systems. The objects manipulated by these formal systems are formulas or sets of formulas. If the formulas that appear in these objects are subformulas or negated subformulas of the formulas of $\Gamma \cup \{\varphi\}$, then the syntactic method will be called *analytical*.

As we saw in Theorem 2.3.38, using signed formulas, we can express alternatively $\Gamma \models \varphi$ as the unsatisfiability of the set $\{\mathbf{T}\psi \mid \psi \in \Gamma\} \cup \{\mathbf{F}\varphi\}$. A syntactic method using this equivalence will manipulate objects that consist of sets of signed formulas. In this case, analyticity requires that the signed formulas of the objects be signed subformulas of formulas in $\Gamma \cup \{\varphi\}$.

Analytical methods are easier to mechanize, though they may yield long proofs. Nonanalytical methods are closer to ordinary mathematical reasoning and leave room for some counterpart of “ingenuity” by allowing the use of formulas extraneous to the original collection of formulas in order to provide shorter proofs.

Some of the syntactic methods we present deal with arbitrary formulas; others are limited to formulas in conjunctive normal form. Among the general syntactic methods, we discuss both nonanalytical (Hilbert/Frege systems, tableaux with cut, sequent calculus with cut, and natural deduction) and analytical (tableaux without cut and sequent calculus without cut) ones.

3.2 A Hilbert/Frege-Style Formal System

The Hilbert[1]/Frege[2]-style formal systems are one of the earliest attempts to formally carry out mathematical reasoning.

Let Γ be a set of formulas. We would like to give a formal system whose theorems are the formulas that are logically implied by Γ. In fact, we will give a single formal system $\mathcal{HF}$ such that for each set of formulas Γ, the desired formal system will be $\mathcal{HF}_\Gamma$. (The reader should review at this point Section 1.8 of Chapter 1, where general formal systems are introduced.)

Definition 3.2.1. The formal system $\mathcal{HF}$ has PLFORM as set of objects, $\{\mathsf{R}_{mp}\}$ as set of rules, where R_{mp} is the modus ponens[3] rule

$$\frac{\varphi, (\varphi \to \psi)}{\psi}$$

for all formulas $\varphi, \psi \in$ PLFORM, and A as axiom set, where A consists of the following formulas:

(1) $(\alpha \to (\beta \to \alpha))$,
(2) $((\alpha \to (\beta \to \gamma)) \to ((\alpha \to \beta) \to (\alpha \to \gamma)))$,

[1]David Hilbert was born on January 23, 1862, in Königsberg, Germany (now Kaliningrad, in Russia), and died in Göttingen, on February 14, 1943. He studied at the University of Königsberg, where he got his Ph.D. in 1884 and taught at the University of Königsberg between 1886 and 1895 and at the University of Göttingen starting in 1895, where he remained for the rest of his life. Hilbert was one of the leading mathematicians of all times and his influence extends to almost every branch of mathematics.

[2]Friedrich Ludwig Gottlob Frege was born on November 8, 1848, in Wismar, Germany, and died on July 26, 1925 in Bad Kleinen, Germany. He studied at Jena and Göttingen and spent his entire career at the University of Jena where he became a special professor in 1879 and a regular professor in 1886. He is considered one of the founders of modern mathematical logic. Frege invented quantifiers and his work clearly separated syntactical from semantical aspects of logic.

[3]*modus ponens* is a Latin term that means "method of affirming".

(3) $(\alpha \to \alpha)$,
(4) $(\alpha \to ((\neg\alpha) \to \beta))$,
(5) $(((\neg\alpha) \to \alpha) \to \alpha)$,
(6) $((\alpha \to (\neg\alpha)) \to (\neg\alpha))$,
(7) $(\alpha \to (\alpha \vee \beta))$,
(8) $(\beta \to (\alpha \vee \beta))$,
(9) $(\alpha \to (\beta \to (\alpha \wedge \beta)))$,
(10) $((\neg\alpha) \to (\alpha \to \beta))$,
(11) $(\alpha \to (\beta \to (\alpha \leftrightarrow \beta)))$,
(12) $((\neg\alpha) \to ((\neg\beta) \to (\alpha \leftrightarrow \beta)))$,
(13) $((\neg\alpha) \to ((\neg\beta) \to (\neg(\alpha \vee \beta))))$,
(14) $((\neg\alpha) \to (\neg(\alpha \wedge \beta)))$,
(15) $((\neg\beta) \to (\neg(\alpha \wedge \beta)))$
(16) $(\alpha \to ((\neg\beta) \to (\neg(\alpha \to \beta))))$,
(17) $(\alpha \to ((\neg\beta) \to (\neg(\alpha \leftrightarrow \beta))))$,
(18) $((\neg\alpha) \to (\beta \to (\neg(\alpha \leftrightarrow \beta))))$,

for all formulas α, β, γ. □

Since R_{mp} allows us to deduce ψ from φ and $(\varphi \to \psi)$ for any formula φ (including formulas φ that are neither subformulas nor negated subformulas of ψ), it follows that $\mathcal{HF}$ is not analytical, in the sense of Section 3.1.

We have chosen our axioms in order to make the proof of completeness as simple as possible without any concern about their nonredundancy. Some of these axioms can be deduced from the others and therefore they could be dispensed with.

Example 3.2.2. Formulas in Axiom Group 3 can be inferred from Axiom Groups 1 and 2 as shown by the following proof:

(1) $(\alpha \to ((\alpha \to \alpha) \to \alpha))$	Axiom Group 1,
(2) $((\alpha \to ((\alpha \to \alpha) \to \alpha)) \to$ $((\alpha \to (\alpha \to \alpha)) \to (\alpha \to \alpha)))$	Axiom Group 2,
(3) $((\alpha \to (\alpha \to \alpha)) \to (\alpha \to \alpha))$	(1), (2) and modus ponens,
(4) $(\alpha \to (\alpha \to \alpha))$	Axiom Group 1,
(5) $(\alpha \to \alpha)$	(3) and (4) and modus ponens.

This shows that we could have omitted Axiom Group 3 without changing the theorems of $\mathcal{HF}_\Gamma$ for any Γ. There are other redundant axiom groups as well. □

Theorem 3.2.3 (Soundness of $\mathcal{HF}_\Gamma$). *Let Γ be a set of formulas and let φ be a formula. Then, $\Gamma \vdash_{\mathcal{HF}} \varphi$ implies $\Gamma \models \varphi$.*

Proof. The argument is by induction on the theorems of $\mathcal{HF}_\Gamma$. For the basis step, let φ be an axiom of $\mathcal{HF}_\Gamma$. Then, either φ is an axiom of $\mathcal{HF}$ or φ belongs to Γ. In the former case, the reader can easily verify that φ is a tautology, so $\Gamma \models \varphi$. In the latter case, we obviously have $\Gamma \models \varphi$.

The inductive step is an immediate consequence of the second part of Theorem 2.3.17. $\square$

The following theorem formalizes the standard mathematical practice in proving that a formula $(\varphi \to \psi)$ is a logical consequence of Γ. Namely, we add φ to our assumptions and we try to prove ψ from $\Gamma \cup \{\varphi\}$.

Theorem 3.2.4 (Deduction Theorem for $\mathcal{HF}$). *Let Γ be a set of formulas and let φ, ψ be formulas. Then, if $\Gamma \cup \{\varphi\} \vdash_{\mathcal{HF}} \psi$, we have $\Gamma \vdash_{\mathcal{HF}} (\varphi \to \psi)$.*

Proof. We show, by induction on $\mathrm{Thm}(\mathcal{HF}_{\Gamma\cup\{\varphi\}})$, that if ψ belongs to this set, then $\Gamma \vdash_{\mathcal{HF}} (\varphi \to \psi)$. If ψ is an axiom of $\mathcal{HF}_{\Gamma\cup\{\varphi\}}$, then either ψ is an axiom of $\mathcal{HF}_\Gamma$ or $\psi = \varphi$. In the former case, we have $\Gamma \vdash_{\mathcal{HF}} \psi$ and, since $\Gamma \vdash_{\mathcal{HF}} (\psi \to (\varphi \to \psi))$ by Axiom Group 1, we get $\Gamma \vdash_{\mathcal{HF}} (\varphi \to \psi)$ by modus ponens. In the latter case, the conclusion follows immediately from Axiom Group 3.

For the inductive step, suppose that ψ is obtained by modus ponens from θ and $(\theta \to \psi)$ and that the result holds for θ and $(\theta \to \psi)$, that is, $\Gamma \vdash_{\mathcal{HF}} (\varphi \to \theta)$ and $\Gamma \vdash_{\mathcal{HF}} (\varphi \to (\theta \to \psi))$. By Axiom Group 2, we have $\Gamma \vdash_{\mathcal{HF}} ((\varphi \to (\theta \to \psi)) \to ((\varphi \to \theta) \to (\varphi \to \psi)))$. Applying modus ponens twice, we get $\Gamma \vdash_{\mathcal{HF}} (\varphi \to \psi)$. $\square$

Corollary 3.2.5. *Let Γ be a set of formulas and let φ, ψ be formulas. Then, $\Gamma \cup \{\varphi\} \vdash_{\mathcal{HF}} \psi$ if and only if $\Gamma \vdash_{\mathcal{HF}} (\varphi \to \psi)$.*

Proof. In view of Theorem 3.2.4, we only need to show that $\Gamma \vdash_{\mathcal{HF}} (\varphi \to \psi)$ implies $\Gamma \cup \{\varphi\} \vdash_{\mathcal{HF}} \psi$. Since $\Gamma \cup \{\varphi\} \vdash_{\mathcal{HF}} \varphi$ and $\Gamma \cup \{\varphi\} \vdash_{\mathcal{HF}} (\varphi \to \psi)$, we have $\Gamma \cup \{\varphi\} \vdash_{\mathcal{HF}} \psi$ by modus ponens. $\square$

The Deduction Theorem states only that the existence of a proof of ψ in $\mathcal{HF}_{\Gamma\cup\{\varphi\}}$ implies the existence of a proof of $(\varphi \to \psi)$

in $\mathcal{HF}_\Gamma$. However, the argument of the Deduction Theorem shows more, namely, it provides an effective way of transforming a proof of ψ in $\mathcal{HF}_{\Gamma\cup\{\varphi\}}$ into a proof of $(\varphi \to \psi)$ in $\mathcal{HF}_\Gamma$.

Definition 3.2.6. A set Γ of formulas is

- *consistent* if there is no formula φ such that

$$\Gamma \vdash_{\mathcal{HF}} \varphi \text{ and } \Gamma \vdash_{\mathcal{HF}} (\neg\varphi),$$

- *inconsistent* if it is not consistent. ▯

It is easy to see that every subset of a consistent set of formulas is consistent; consequently, every superset of an inconsistent set of formulas is inconsistent. Moreover, we prove the following theorem:

Theorem 3.2.7. *Consistency is a property of finite character.*

Proof. If Γ is a consistent set of formulas, then it is clear that each of its finite subsets is consistent.

Conversely, suppose that Γ is inconsistent. Then, there is a formula φ such that $\Gamma \vdash_{\mathcal{HF}} \varphi$ and $\Gamma \vdash_{\mathcal{HF}} (\neg\varphi)$. Let $(\varphi_0, \ldots, \varphi_{n-1})$, $(\varphi'_0, \ldots, \varphi'_{m-1})$ be proofs in $\mathcal{HF}_\Gamma$ of φ and $(\neg\varphi)$, respectively. If $\Gamma_0 = (\{\varphi_i \mid 0 \leq i \leq n-1\} \cup \{\varphi'_i \mid 0 \leq i \leq m-1\}) \cap \Gamma$, then $(\varphi_0, \ldots, \varphi_{n-1})$ and $(\varphi'_0, \ldots, \varphi'_{m-1})$ are also proofs in $\mathcal{HF}_{\Gamma_0}$, so Γ_0 is a finite, inconsistent subset of Γ. □

Theorem 3.2.8. *If Γ is a satisfiable set of formulas, then Γ is consistent.*

Proof. Suppose that Γ is an inconsistent set of formulas. Then, there is a formula φ such that $\Gamma \vdash_{\mathcal{HF}} \varphi$ and $\Gamma \vdash_{\mathcal{HF}} (\neg\varphi)$. By the Soundness Theorem, we have $\Gamma \models \varphi$ and $\Gamma \models (\neg\varphi)$, which implies that Γ is unsatisfiable. Indeed, suppose that v were a truth assignment that satisfies Γ. Then, we would have $v(\varphi) = v((\neg\varphi)) = \mathbf{T}$, which is impossible. □

Theorem 3.2.9. *If Γ is a consistent set of formulas and $\Gamma \vdash_{\mathcal{HF}} \varphi$, then $\Gamma \cup \{\varphi\}$ is also consistent. Thus, if $\Gamma \cup \{\varphi\}$ is inconsistent and $\Gamma \vdash_{\mathcal{HF}} \varphi$, then Γ is inconsistent.*

Proof. Since $\Gamma \vdash_{\mathcal{HF}} \varphi$, by Corollary 1.8.6, the formal systems $\mathcal{HF}_\Gamma$ and $\mathcal{HF}_{\Gamma\cup\{\varphi\}}$ are equivalent, which gives the result. □

Corollary 3.2.10. *Let* Γ *be a maximally consistent set of formulas. Then,* $\Gamma \vdash_{\mathcal{HF}} \varphi$ *implies* $\varphi \in \Gamma$.

Proof. This follows immediately from Theorem 3.2.9. □

Theorem 3.2.11. *If* Γ *is an inconsistent set of formulas, then for every* $\psi \in$ PLFORM, $\Gamma \vdash_{\mathcal{HF}} \psi$.

Proof. Since Γ is inconsistent, there is a formula φ such that $\Gamma \vdash_{\mathcal{HF}} \varphi$ and $\Gamma \vdash_{\mathcal{HF}} (\neg\varphi)$. For every formula ψ, by Axiom Group 4, $\Gamma \vdash_{\mathcal{HF}} (\varphi \to ((\neg\varphi) \to \psi))$. Using modus ponens twice, we get $\Gamma \vdash_{\mathcal{HF}} \psi$. □

The next statement describes a method of derivation in $\mathcal{HF}$ known as *reductio ad absurdum.*

Theorem 3.2.12. *Let* Γ *be a set of formulas and* φ *be a formula. If* $\Gamma \cup \{(\neg\varphi)\}$ *is inconsistent, then* $\Gamma \vdash_{\mathcal{HF}} \varphi$.

If $\Gamma \cup \{\varphi\}$ *is inconsistent, then* $\Gamma \vdash_{\mathcal{HF}} (\neg\varphi)$.

Proof. Suppose that $\Gamma \cup \{(\neg\varphi)\}$ is inconsistent. Then, by Theorem 3.2.11, $\Gamma \cup \{(\neg\varphi)\} \vdash_{\mathcal{HF}} \varphi$ and so, by the Deduction Theorem, $\Gamma \vdash_{\mathcal{HF}} ((\neg\varphi) \to \varphi)$. By Axiom Group 5 and modus ponens, $\Gamma \vdash_{\mathcal{HF}} \varphi$.

By Theorem 3.2.11, the inconsistency of $\Gamma \cup \{\varphi\}$ means that we have $\Gamma \cup \{\varphi\} \vdash_{\mathcal{HF}} (\neg\varphi)$. By the Deduction Theorem, $\Gamma \vdash_{\mathcal{HF}} (\varphi \to (\neg\varphi))$. Since $\Gamma \vdash_{\mathcal{HF}} ((\varphi \to (\neg\varphi)) \to (\neg\varphi))$ by Axiom Group 6, an application of R_{mp} yields $\Gamma \vdash_{\mathcal{HF}^{\mathcal{L}}} (\neg\varphi)$. □

We will often use the contrapositive of this theorem, namely, if $\Gamma \nvdash_{\mathcal{HF}} \varphi$, then $\Gamma \cup \{(\neg\varphi)\}$ is consistent, and if $\Gamma \nvdash_{\mathcal{HF}} (\neg\varphi)$, then $\Gamma \cup \{\varphi\}$ is consistent.

Corollary 3.2.13. *Let* Γ *be a consistent set of formulas. Then, for each formula* φ*, at least one of* $\Gamma \cup \{\varphi\}$ *and* $\Gamma \cup \{(\neg\varphi)\}$ *is consistent. Therefore, if both* $\Gamma \cup \{\varphi\}$ *and* $\Gamma \cup \{(\neg\varphi)\}$ *are inconsistent, then so is* Γ.

If Γ *is maximally consistent and* φ *is a formula, then exactly one of the formulas* φ *and* $(\neg\varphi)$ *belongs to* Γ.

Proof. If $\Gamma \nvdash_{\mathcal{HF}} \varphi$, then, by Theorem 3.2.12, $\Gamma \cup \{(\neg\varphi)\}$ is consistent. If $\Gamma \vdash_{\mathcal{HF}} \varphi$, then, by Theorem 3.2.9, $\Gamma \cup \{\varphi\}$ is consistent. The second part is immediate. □

Next, we discuss a systematic effectivization of the previous results by introducing the notion of certificate of inconsistency.

Definition 3.2.14. Let $\mathcal{H}$ be a formal system whose objects are the formulas and let Γ be a set of formulas. An $(\mathcal{H},\Gamma)$-*certificate of inconsistency* is a pair (q,q') of proofs in $\mathcal{H}_\Gamma$ such that for some formula α, q is a proof of α and q' is a proof of $(\neg\alpha)$. ▯

Effective versions of Theorems 3.2.11 and 3.2.12 and Corollary 3.2.13 are given in the following theorem.

Theorem 3.2.15.

(1) *There is an effective, syntactic construction that starts with a formula φ and an $(\mathcal{HF},\Gamma)$-certificate of inconsistency (q,q') and produces a proof in $\mathcal{HF}_\Gamma$ for φ.*
(2) *There is an effective, syntactic construction that starts with (q,q'), an $(\mathcal{HF},\Gamma\cup\{(\neg\varphi)\})$-certificate of inconsistency, and yields a proof of φ in $\mathcal{HF}_\Gamma$.*
(3) *There is an effective, syntactic construction that starts with $(\mathcal{HF},\Gamma\cup\{\varphi\})$- and $(\mathcal{HF},\Gamma\cup\{(\neg\varphi)\})$-certificates of inconsistency and produces an $(\mathcal{HF},\Gamma)$-certificate of inconsistency.*

Proof. For Part (a), assume that q is a proof of α and q' is a proof of $(\neg\alpha)$. Then, taking into account the existence of the axiom $(\alpha\to((\neg\alpha)\to\varphi))$, we obtain the proof

$$qq'((\alpha\to((\neg\alpha)\to\varphi)),((\neg\alpha)\to\varphi),\varphi)$$

for φ.

In order to prove Part (b), let us remark that by Part (a), we can construct effectively a proof in $\mathcal{HF}_{\Gamma\cup\{(\neg\varphi)\}}$ of φ. Using the observation that follows Corollary 3.2.5, we can construct effectively a proof of $((\neg\varphi)\to\varphi)$ in $\mathcal{HF}_\Gamma$. Then, by using the axiom $(((\neg\varphi)\to\varphi)\to\varphi)$, we obtain a proof of φ in $\mathcal{HF}_\Gamma$.

For the last part, note that, by Part (b), we can effectively construct a proof r of φ in $\mathcal{HF}_\Gamma$ from the $(\mathcal{HF},\Gamma\cup\{(\neg\varphi)\})$-certificate of inconsistency. Then, if (q,q') is an $(\mathcal{HF},\Gamma\cup\{\varphi\})$-certificate of inconsistency, the pair (rq,rq') is an $(\mathcal{HF},\Gamma)$-certificate of inconsistency. □

A more explicit description of the construction outlined in Part (3) of the previous theorem starts from (q_0, q_0'), a certificate of inconsistency for $\Gamma \cup \{\varphi\}$ and (q_1, q_1') and a certificate of inconsistency for $\Gamma \cup \{(\neg\varphi)\}$. Let α and $(\neg\alpha)$ be the formulas proved by q_1 and q_1', respectively. Then, we have the proof in $\mathcal{HF}_{\Gamma\cup\{(\neg\varphi)\}}$:

$$s = q_1 q_1'((\alpha \to ((\neg\alpha) \to \varphi)), ((\neg\alpha) \to \varphi), \varphi).$$

By the effectivized version of the Deduction Theorem, we can obtain a proof s' of $((\neg\varphi) \to \varphi)$ in $\mathcal{HF}_\Gamma$ starting from s. Let $r = s'((((\neg\varphi) \to \varphi) \to \varphi), \varphi)$. The desired Γ-certificate of inconsistency is (rq_0, rq_0').

Lemma 3.2.16. *Let Γ be a set of formulas and let φ be a positive or a negated positive formula which is not a literal, that is, $\varphi = (\alpha C \beta)$ or $\varphi = (\neg(\alpha C \beta))$. Then, there is an effective, syntactic construction that, starting from $(\mathcal{HF}, \Gamma \cup K)$-certificates of inconsistency for all constituents K of φ, produces $(\mathcal{HF}, \Gamma')$-certificates of inconsistency when Γ' is $\Gamma \cup \{\varphi, \alpha, \beta\}$, $\Gamma \cup \{\varphi, (\neg\alpha), \beta\}$, $\Gamma \cup \{\varphi, \alpha, (\neg\beta)\}$, or $\Gamma \cup \{\varphi, (\neg\alpha), (\neg\beta)\}$.*

Proof. Suppose that $\varphi = (\alpha \leftrightarrow \beta)$. By hypothesis, we have $(\mathcal{HF}, \Gamma \cup \{\alpha, \beta\})$- and $(\mathcal{HF}, \Gamma \cup \{(\neg\alpha), (\neg\beta)\})$-certificates of inconsistency, which are at the same time $(\mathcal{HF}, \Gamma \cup \{\varphi, \alpha, \beta\})$- and $(\mathcal{HF}, \Gamma \cup \{\varphi, (\neg\alpha), (\neg\beta)\})$-certificates of inconsistency, respectively. Moreover, the pair (q, q'), where $q = ((\alpha \leftrightarrow \beta))$ and

$$\begin{aligned} q' = (&\alpha, (\neg\beta), (\alpha \to ((\neg\beta) \to (\neg(\alpha \leftrightarrow \beta)))), \\ &((\neg\beta) \to (\neg(\alpha \leftrightarrow \beta))), (\neg(\alpha \leftrightarrow \beta))), \end{aligned}$$

is an $(\mathcal{HF}, \Gamma \cup \{\varphi, \alpha, (\neg\beta)\})$-certificate of inconsistency because of Axiom Group 17.

In a similar manner, using Axiom Group 18, we construct an $(\mathcal{HF}, \Gamma \cup \{\varphi, (\neg\alpha), \beta\})$-certificate of inconsistency.

Let now $\varphi = (\neg(\alpha \wedge \beta))$. By hypothesis, we have the pairs (q_0, q_0') and (q_1, q_1') that are $(\mathcal{HF}, \Gamma \cup \{(\neg\alpha)\})$- and $(\mathcal{HF}, \Gamma \cup \{(\neg\beta)\})$-certificates of inconsistency, respectively. Note that (q_0, q_0') is an $(\mathcal{HF}, \Gamma \cup \{\varphi, (\neg\alpha), \beta\})$- and, also, an $(\mathcal{HF}, \Gamma \cup \{\varphi, (\neg\alpha), (\neg\beta)\})$-certificate of inconsistency and (q_1, q_1') is both an $(\mathcal{HF}, \Gamma \cup \{\varphi, (\neg\alpha), (\neg\beta)\})$- and an $(\mathcal{HF}, \Gamma \cup \{\varphi, \alpha, (\neg\beta)\})$-certificate of inconsistency.

The pair (r, r') given by $r = ((\neg(\alpha \wedge \beta)))$ and

$$r = (\alpha, \beta, (\alpha \to (\beta \to (\alpha \wedge \beta))), (\beta \to (\alpha \wedge \beta)), (\alpha \wedge \beta))$$

is an $(\mathcal{HF}, \Gamma \cup \{\varphi, \alpha, \beta\})$-certificate of inconsistency due to Axiom Group 9.

We leave to the reader consideration of the remaining cases. □

Theorem 3.2.17. *There is an effective, syntactic construction that starts with a set* Γ *of formulas, a formula* φ*, and* $(\mathcal{HF}, \Gamma \cup K)$*-certificates of inconsistency for all constituents* K *of* φ *and produces an* $(\mathcal{HF}, \Gamma \cup \{\varphi\})$*-certificate of inconsistency.*

Proof. If φ is a positive formula or a negated positive formula, apply Lemma 3.2.16 and Theorem 3.2.15, Part (3).

If $\varphi = (\neg(\neg\alpha))$, then we have an $(\mathcal{HF}, \Gamma \cup \{\alpha\})$-certificate of inconsistency, which is also an $(\mathcal{HF}, \Gamma \cup \{\varphi, \alpha\})$-certificate of inconsistency. Also, note that $(((\neg\alpha)), ((\neg(\neg\alpha))))$ is an $(\mathcal{HF}, \Gamma \cup \{\varphi, (\neg\alpha)\})$-certificate of inconsistency. Using Theorem 3.2.15, Part (3), we obtain an $(\mathcal{HF}, \Gamma \cup \{\varphi\})$-certificate of inconsistency. □

Theorem 3.2.18. *If* Γ *is a maximally consistent set of formulas, then* Γ *is a truth set.*

Proof. Let Γ be a maximally consistent set of formulas. For every formula φ, $(\neg\varphi) \in \Gamma$ if and only if $\varphi \notin \Gamma$ because of Corollary 3.2.13. By Theorem 2.7.7, in the presence of the above condition, Γ is a truth set if and only if for every formula φ that is either a positive formula, but not a variable, or the negation of such a formula, if a constituent of φ is a subset of Γ, then $\varphi \in \Gamma$. Let φ be such a formula and let K be a constituent of φ that is included in Γ. For some $\varphi_0, \ldots, \varphi_{n-1}$, we have $K = \{\varphi_0, \ldots, \varphi_{n-1}\}$ and there is an axiom of $\mathcal{HF}$ equal to

$$(\varphi_0 \to (\varphi_1 \to \cdots (\varphi_{n-1} \to \varphi) \cdots)),$$

where n is either 1 or 2. Then, by applying modus ponens n times, we obtain $\Gamma \vdash_{\mathcal{HF}} \varphi$, which, by Corollary 3.2.10, implies $\varphi \in \Gamma$. □

Theorem 3.2.19. *If* Γ *is a consistent set of formulas, then* Γ *is satisfiable.*

Proof. By Theorem 3.2.7, consistency is a property of finite character. Therefore, by Theorem 1.3.3, there is a maximally consistent set Γ' such that $\Gamma \subseteq \Gamma'$. By Theorem 3.2.18, Γ' is a truth set and, hence, by Theorem 2.7.13, Γ' is satisfiable, which implies the satisfiability of Γ. $\square$

Note that Theorem 3.2.17 shows that inconsistency of sets of formulas is an inconsistency property (and, therefore, consistency of sets of formulas is a consistency property). An alternative proof of Theorem 3.2.19 is therefore an immediate consequence of Theorem 2.7.21.

Corollary 3.2.20. *A set of formulas* Γ *is consistent if and only if it is satisfiable.*

Proof. The statement follows from Theorems 3.2.8 and 3.2.19. $\square$

Theorem 3.2.21 (Completeness of $\mathcal{HF}_\Gamma$). *Let* Γ *be a set of formulas and let* φ *be a formula. If* $\Gamma \models \varphi$*, then* $\Gamma \vdash_{\mathcal{HF}} \varphi$*.*

Proof. If $\Gamma \nvdash_{\mathcal{HF}} \varphi$, then, by Theorem 3.2.12, $\Gamma \cup \{(\neg\varphi)\}$ is consistent, so, by Theorem 3.2.19, $\Gamma \cup \{(\neg\varphi)\}$ is satisfiable. This implies that $\Gamma \not\models \varphi$, by Part 1 of Theorem 2.3.17. $\square$

Corollary 3.2.22. *Let* Γ *be a set of formulas and let* φ *be a formula. Then,* $\Gamma \models \varphi$ *if and only if* $\Gamma \vdash_{\mathcal{HF}} \varphi$*.*

Proof. This follows immediately from Theorems 3.2.3 and 3.2.21. $\square$

Example 3.2.23. The set of formulas $\{(\neg\varphi), (\psi \to \varphi), \psi\}$ is clearly inconsistent. Therefore, by Theorem 3.2.12 (reductio ad absurdum), we have $\{(\neg\varphi), (\psi \to \varphi)\} \vdash_{\mathcal{HF}} (\neg\psi)$. ▯

The next statement is a formalization of proof by cases.

Theorem 3.2.24. *Let* Γ *be a set of formulas and let* φ, ψ *be two formulas such that we have both* $\Gamma \cup \{\varphi\} \vdash_{\mathcal{HF}} \psi$ *and* $\Gamma \cup \{(\neg\varphi)\} \vdash_{\mathcal{HF}} \psi$*. Then,* $\Gamma \vdash_{\mathcal{HF}} \psi$*.*

Proof. The result will follow easily if we prove $\vdash_{\mathcal{HF}} ((\varphi \to \psi) \to (((\neg\varphi) \to \psi) \to \psi))$. To this end, we will show that $\{(\varphi \to \psi), ((\neg\varphi) \to \psi)\} \vdash_{\mathcal{HF}} \psi$ and then apply the Deduction

Theorem twice. By *reductio ad absurdum*, it suffices to show that the set $\{(\varphi \to \psi), ((\neg\varphi) \to \psi), (\neg\psi)\}$ is inconsistent. By Example 3.2.23, we have $\{(\varphi \to \psi), ((\neg\varphi) \to \psi), (\neg\psi)\} \vdash_{\mathcal{HF}} (\neg\varphi)$ and $\{(\varphi \to \psi), ((\neg\varphi) \to \psi), (\neg\psi)\} \vdash_{\mathcal{HF}} (\neg(\neg\varphi))$. □

The argument of Theorem 3.2.24 implies that the proof that shows $\Gamma \vdash_{\mathcal{HF}} \psi$ can be obtained effectively from proofs that show that $\Gamma \cup \{\varphi\} \vdash_{\mathcal{HF}} \psi$ and $\Gamma \cup \{(\neg\varphi)\} \vdash_{\mathcal{HF}} \psi$.

Lemma 3.2.25. *Let φ, ψ be two formulas. We have $(\varphi \leftrightarrow \psi) \vdash_{\mathcal{HF}} (\varphi \to \psi)$ and $(\varphi \leftrightarrow \psi) \vdash_{\mathcal{HF}} (\psi \to \varphi)$.*

Proof. Starting from the instance of the Axiom Group 17 $(\varphi \to ((\neg\psi) \to (\neg(\varphi \leftrightarrow \psi))))$, it follows that the set $\{(\varphi \leftrightarrow \psi), \varphi, (\neg\psi)\}$ is inconsistent. Therefore, by *reductio ad absurdum*, we have $\{(\varphi \leftrightarrow \psi), \varphi\} \vdash_{\mathcal{HF}} \psi$. An application of the Deduction Theorem shows that $\{(\varphi \leftrightarrow \psi)\} \vdash_{\mathcal{HF}} (\varphi \to \psi)$.

The existence of the second proof can be shown similarly. □

Lemma 3.2.26. *Let φ, ψ be two formulas. We have $\{(\varphi \to \psi), (\psi \to \varphi)\} \vdash_{\mathcal{HF}} (\varphi \leftrightarrow \psi)$.*

Proof. Note that by modus ponens, we have $\{\varphi, (\varphi \to \psi), (\psi \to \varphi)\} \vdash_{\mathcal{HF}} \psi$. The formula $(\varphi \to (\psi \to (\varphi \leftrightarrow \psi)))$ is an instance of Axiom Group 11. By using modus ponens twice, we obtain $\{\varphi, (\varphi \to \psi), (\psi \to \varphi)\} \vdash_{\mathcal{HF}} (\varphi \leftrightarrow \psi)$.

By Example 3.2.23, we have $\{(\neg\varphi), (\varphi \to \psi), (\psi \to \varphi)\} \vdash_{\mathcal{HF}} (\neg\psi)$. The formula $((\neg\varphi) \to ((\neg\psi) \to (\varphi \leftrightarrow \psi)))$ is an instance of Axiom Group 12. Again, by a double application of modus ponens, we have $\{(\neg\varphi), (\varphi \to \psi), (\psi \to \varphi)\} \vdash_{\mathcal{HF}} (\varphi \leftrightarrow \psi)$. By Theorem 3.2.24, we have finally $\{(\varphi \to \psi), (\psi \to \varphi)\} \vdash_{\mathcal{HF}} (\varphi \leftrightarrow \psi)$. □

Given the formulas φ and ψ, we can effectively find the proofs in $\mathcal{HF}$ whose existence is asserted by Lemmas 3.2.25 and 3.2.26.

Theorem 3.2.27. *Let $\varphi_0, \varphi_1, \varphi_2$ be formulas. We have $\{(\varphi_0 \leftrightarrow \varphi_1), (\varphi_1 \leftrightarrow \varphi_2)\} \vdash_{\mathcal{HF}} (\varphi_0 \leftrightarrow \varphi_2)$.*

Proof. Lemma 3.2.25 implies that $(\varphi_0 \leftrightarrow \varphi_1) \vdash_{\mathcal{HF}} (\varphi_0 \to \varphi_1)$ and $(\varphi_0 \leftrightarrow \varphi_1) \vdash_{\mathcal{HF}} (\varphi_1 \to \varphi_0)$. Similarly, we have $(\varphi_1 \leftrightarrow \varphi_2) \vdash_{\mathcal{HF}} (\varphi_1 \to \varphi_2)$ and $(\varphi_1 \leftrightarrow \varphi_2) \vdash_{\mathcal{HF}} (\varphi_2 \to \varphi_1)$. This allows us to write $\{(\varphi_0 \leftrightarrow \varphi_1), (\varphi_1 \leftrightarrow \varphi_2)\} \vdash_{\mathcal{HF}} (\varphi_0 \to \varphi_2)$ by applying the hypothetical syllogism rule introduced in Supplement 2. Similarly, we have

$\{(\varphi_0 \leftrightarrow \varphi_1), (\varphi_1 \leftrightarrow \varphi_2)\} \vdash_{\mathcal{HF}} (\varphi_2 \to \varphi_0)$. Finally, by Lemma 3.2.26, we obtain $\{(\varphi_0 \leftrightarrow \varphi_1), (\varphi_1 \leftrightarrow \varphi_2)\} \vdash_{\mathcal{HF}} (\varphi_0 \leftrightarrow \varphi_2)$. □

The last theorem shows that starting from the formulas φ_0, φ_1, and φ_2, we can produce effectively a proof of $(\varphi_0 \leftrightarrow \varphi_2)$ from proofs of $(\varphi_0 \leftrightarrow \varphi_1)$ and $(\varphi_1 \leftrightarrow \varphi_2)$.

Corollary 3.2.28. *Let $n \geq 2$ and let $\varphi_0, \ldots, \varphi_{n-1}$ be n formulas. There is a proof in $\mathcal{HF}$ that shows that $\{(\varphi_0 \leftrightarrow \varphi_1), \ldots, (\varphi_{n-2} \leftrightarrow \varphi_{n-1})\} \vdash_{\mathcal{HF}} (\varphi_0 \leftrightarrow \varphi_{n-1})$.*

Proof. The argument is by induction on $n \geq 2$. The basis step, $n = 2$, is immediate. Suppose the statement holds for n and let $\varphi_0, \ldots, \varphi_n$ be $n+1$ formulas. By inductive hypothesis, there is a proof in $\mathcal{HF}$ that shows that $\{(\varphi_0 \leftrightarrow \varphi_1), \ldots, (\varphi_{n-2} \leftrightarrow \varphi_{n-1})\} \vdash_{\mathcal{HF}} (\varphi_0 \leftrightarrow \varphi_{n-1})$. By Theorem 3.2.27, we have $\{(\varphi_0 \leftrightarrow \varphi_{n-1}), (\varphi_{n-1} \leftrightarrow \varphi_n)\} \vdash_{\mathcal{HF}} (\varphi_0 \leftrightarrow \varphi_n)$, which yields the desired conclusion. □

The argument of Corollary 3.2.28 shows that the proof of $(\varphi_0 \leftrightarrow \varphi_{n-1})$ from $\{(\varphi_0 \leftrightarrow \varphi_1), \ldots, (\varphi_{n-2} \leftrightarrow \varphi_{n-1})\}$ can be found effectively.

Corollary 3.2.29. *Let $\varphi_0, \ldots, \varphi_{n-1}$ be formulas, where $n \geq 2$, and let Γ be a set of formulas. Starting from proofs $\Gamma \vdash_{\mathcal{HF}} (\varphi_i \leftrightarrow \varphi_{i+1})$ for $0 \leq i \leq n-2$, one can produce effectively a proof for $\Gamma \vdash_{\mathcal{HF}} (\varphi_0 \leftrightarrow \varphi_{n-1})$.*

Proof. The corollary follows directly from Corollary 3.2.28. □

Definition 3.2.30. Let Γ be a set of formulas and let φ, ψ be two formulas. We say that φ and ψ are Γ-*provably equivalent* if $\Gamma \vdash_{\mathcal{HF}} (\varphi \leftrightarrow \psi)$. The formulas φ, ψ are *provably equivalent* if they are $\emptyset$-provably equivalent. ▯

We will prove now a syntactic version of the Replacement Theorem (Theorem 2.6.12) without appealing to the Completeness Theorem. This requires providing explicit proofs of several tautologies. These proofs, of course, exist by the Completeness Theorem.

Lemma 3.2.31. *The following tautologies have proofs in the formal system $\mathcal{HF}$ that can be found effectively given the formulas $\theta_0, \theta_1, \theta_0'$*

and the binary connective symbol C:

$$((\theta_0 \leftrightarrow \theta_0') \rightarrow ((\theta_0 C \theta_1) \leftrightarrow (\theta_0' C \theta_1))),$$
$$((\theta_0 \leftrightarrow \theta_0') \rightarrow ((\theta_1 C \theta_0) \leftrightarrow (\theta_1 C \theta_0'))).$$

In addition, for all θ_0, θ_0', we can find effectively a proof for $((\theta_0 \leftrightarrow \theta_0') \rightarrow ((\neg\theta_0) \leftrightarrow (\neg\theta_0')))$.

Proof. To find a proof for $((\theta_0 \leftrightarrow \theta_0') \rightarrow ((\theta_0 \vee \theta_1) \leftrightarrow (\theta_0' \vee \theta_1)))$, it suffices to find a proof for $(\theta_0 \leftrightarrow \theta_0') \vdash_{\mathcal{HF}} ((\theta_0 \vee \theta_1) \leftrightarrow (\theta_0' \vee \theta_1))$ and then apply the Deduction Theorem. In turn, it suffices to find proofs which show that

$$\{(\theta_0 \leftrightarrow \theta_0'), (\theta_0 \vee \theta_1)\} \vdash_{\mathcal{HF}} (\theta_0' \vee \theta_1), \tag{3.1}$$
$$\{(\theta_0 \leftrightarrow \theta_0'), (\theta_0' \vee \theta_1)\} \vdash_{\mathcal{HF}} (\theta_0 \vee \theta_1), \tag{3.2}$$

and then apply twice the Deduction Theorem and Lemma 3.2.26. For the proof of (3.1), using proof by cases, we will show that

$$\{(\theta_0 \leftrightarrow \theta_0'), (\theta_0 \vee \theta_1), \theta_1\} \vdash_{\mathcal{HF}} (\theta_0' \vee \theta_1), \tag{3.3}$$
$$\{(\theta_0 \leftrightarrow \theta_0'), (\theta_0 \vee \theta_1), (\neg\theta_1)\} \vdash_{\mathcal{HF}} (\theta_0' \vee \theta_1). \tag{3.4}$$

The proof for (3.3) follows directly from Axiom Group 8. For (3.4), we note that $\{(\neg\theta_1), (\neg\theta_0), (\theta_0 \vee \theta_1)\}$ is inconsistent by Axiom Group 13. By *reductio ad absurdum*, we have $\{(\theta_0 \leftrightarrow \theta_0'), (\neg\theta_1), (\theta_0 \vee \theta_1)\} \vdash_{\mathcal{HF}} \theta_0$. By Lemma 3.2.25, we have $\{(\theta_0 \leftrightarrow \theta_0'), (\neg\theta_1), (\theta_0 \vee \theta_1)\} \vdash_{\mathcal{HF}} (\theta_0 \rightarrow \theta_0')$. By modus ponens, we have $\{(\theta_0 \leftrightarrow \theta_0'), (\neg\theta_1), (\theta_0 \vee \theta_1)\} \vdash_{\mathcal{HF}} \theta_0'$. Now, by Axiom Group 7, we obtain $\{(\theta_0 \leftrightarrow \theta_0'), (\neg\theta_1), (\theta_0 \vee \theta_1)\} \vdash_{\mathcal{HF}} (\theta_0' \vee \theta_1)$. The argument for the proof of (3.2) is similar.

For a proof of $((\theta_0 \leftrightarrow \theta_0') \rightarrow ((\theta_0 \wedge \theta_1) \leftrightarrow (\theta_0' \wedge \theta_1)))$, it suffices to find a proof for $(\theta_0 \leftrightarrow \theta_0') \vdash_{\mathcal{HF}} ((\theta_0 \wedge \theta_1) \leftrightarrow (\theta_0' \wedge \theta_1))$ and then apply the Deduction Theorem. In turn, it suffices to find proofs which show that

$$\{(\theta_0 \leftrightarrow \theta_0'), (\theta_0 \wedge \theta_1)\} \vdash_{\mathcal{HF}} (\theta_0' \wedge \theta_1), \tag{3.5}$$
$$\{(\theta_0 \leftrightarrow \theta_0'), (\theta_0' \wedge \theta_1)\} \vdash_{\mathcal{HF}} (\theta_0 \wedge \theta_1), \tag{3.6}$$

and then apply twice the Deduction Theorem and Lemma 3.2.26. By Axiom Group 14, the set $\{(\theta_0 \leftrightarrow \theta_0'), (\theta_0 \wedge \theta_1), (\neg\theta_0)\}$ is inconsistent.

By *reduction ad absurdum*, we obtain $\{(\theta_0 \leftrightarrow \theta_0'), (\theta_0 \wedge \theta_1)\} \vdash_{\mathcal{HF}} \theta_0$. Similarly, we obtain $\{(\theta_0 \leftrightarrow \theta_0'), (\theta_0 \wedge \theta_1)\} \vdash_{\mathcal{HF}} \theta_1$. By Lemma 3.2.25, we obtain $\{(\theta_0 \leftrightarrow \theta_0'), (\theta_0 \wedge \theta_1)\} \vdash_{\mathcal{HF}} (\theta_0 \rightarrow \theta_0')$ and by modus ponens, we have $\{(\theta_0 \leftrightarrow \theta_0'), (\theta_0 \wedge \theta_1)\} \vdash_{\mathcal{HF}} \theta_0'$. By Axiom Group 9, we have $\{(\theta_0 \leftrightarrow \theta_0'), (\theta_0 \wedge \theta_1)\} \vdash_{\mathcal{HF}} (\theta_0' \wedge \theta_1)$. A similar argument can be used to obtain a proof of (3.6).

We leave to reader to the reader the remaining proofs, which are easier. □

Theorem 3.2.32. *Let φ be a formula. If α, β are provably equivalent formulas and (α, i) is an occurrence of α in φ, then φ is provably equivalent to $\psi = \texttt{replace}\,(\varphi, (\alpha, i), \beta)$.*

Proof. First note that in the special case when φ coincides with α, we have $\psi = \beta$ and thus φ and ψ are clearly provably equivalent.

The argument is by induction on the formula φ. If φ is a variable, we are in the special case. Suppose that $\varphi = (\varphi_0 C \varphi_1)$, where C is a binary connective symbol. If we are not in the special case, the occurrence of α may be located either in φ_0 or in φ_1. Suppose that the occurrence is located in φ_0. (The argument is similar if the occurrence is within φ_1.) Let φ_0' be the formula obtained from φ_0 by replacing the occurrence of α by β. Then, $\psi = (\varphi_0' C \varphi_1)$. By the inductive hypothesis, the formulas φ_0, φ_0' are provably equivalent. By Lemma 3.2.31, for every binary connective symbol C, there is a proof of the formula

$$((\varphi_0 \leftrightarrow \varphi_0') \rightarrow ((\varphi_0 C \varphi_1) \leftrightarrow (\varphi_0' C \varphi_1))).$$

By modus ponens, the formula $((\varphi_0 C \varphi_1) \leftrightarrow (\varphi_0' C \varphi_1))$ is also provable and so $(\varphi \leftrightarrow \psi)$ is provable.

We leave to the reader the case when $\varphi = (\neg \varphi_0)$. □

The argument of Theorem 3.2.32 shows that starting from a formula φ, two provably equivalent formulas α and β and a proof of $(\alpha \leftrightarrow \beta)$, and an occurrence of α in φ, we can effectively find a proof of $(\varphi \leftrightarrow \psi)$, where ψ is the formula obtained from φ by replacing the occurrence of α by β.

3.3 Tableaux

Tableaux constitute a systematic method of searching for truth assignments that satisfy a set of signed or unsigned formulas. They make use of the fact that a truth assignment satisfies a formula if and only if it satisfies one of its constituents and are built through the following recursive process. If $\Omega = \Omega' \cup \{\alpha\}$ is a set of signed or unsigned formulas, then Ω is satisfiable if and only if $\Omega' \cup K$ is satisfiable for some constituent K of α. We can now repeat the process for each $\Omega' \cup K$ by picking a formula α' in the set. A tree is used to keep track of the development of the algorithm. Each branch of the tree represents a possible way of satisfying the set Ω.

We begin this section with tableaux for signed formulas. We will use the sequence of constituents $\mathtt{d}(b\varphi)$ of a signed formula $b\varphi$ introduced in Section 2.7.

Observe that $\mathtt{d}(\mathbf{T}(\varphi \vee \varphi))$, $\mathtt{d}(\mathbf{F}(\varphi \wedge \varphi))$, and $\mathtt{d}(\mathbf{F}(\varphi \leftrightarrow \varphi))$ are all sequences of length two whose entries are the same. This will help us in presenting a uniform treatment of tableaux.

Recall that we refer to a labeled ordered tree as a lot. (For this and related terminology, see Section 1.7.)

Definition 3.3.1. A *signed tableau*, or just a *tableau* for brevity, is a lot whose labels are sets of signed formulas.

If $\mathtt{T}$ is a tableau and $\mathtt{P}$ is a path of $\mathtt{T}$, then we say that a signed formula $b\varphi$ *occurs* in $\mathtt{P}$ if $b\varphi \in \mathtt{T}(r)$ for some $r \in \mathtt{P}$.

A truth assignment *satisfies a node q* (*a path* $\mathtt{P}$) if it satisfies every signed formula in $\mathtt{T}(q)$ (every signed formula that occurs in $\mathtt{P}$). $\square$

If $\mathtt{P}$ is a path of a tableau $\mathtt{T}$, then $\mathtt{T}(\mathtt{P})$ is the set of signed formulas that occur in $\mathtt{P}$. (More precisely, the set of signed formulas that occur in a path $\mathtt{P}$ ought to be denoted by $\bigcup \mathtt{T}(\mathtt{P})$; however, we prefer to use this slight abuse of notation.)

When there is no risk of confusion, we may use in this section the term “formula” instead of “signed formula.”

Definition 3.3.2. A node q of a tableau $\mathtt{T}$ is *closed* if $\mathtt{T}(q)$ is closed, that is, there is an unsigned formula φ such that both $\mathbf{T}\varphi$ and $\mathbf{F}\varphi$ belong to $\mathtt{T}(q)$. A branch $\mathtt{B}$ of $\mathtt{T}$ is *closed* if the set of formulas occurring in $\mathtt{B}$ is closed. $\mathtt{B}$ is *strongly closed* if it is finite and q is closed, where q is the endpoint of $\mathtt{B}$.

A branch $\mathtt{B}$ is *complete* if the set of formulas occurring in $\mathtt{B}$ is a Hintikka set. $\square$

Since tableaux are partial functions from $\mathrm{Seq}(\mathbf{N})$ to $\mathcal{P}(\mathrm{SPLFORM})$, it makes sense to talk about one tableau being an extension of another tableau. In other words, a tableau $\mathtt{T}'$ is an extension of a tableau $\mathtt{T}$ if $\mathrm{Dom}(\mathtt{T}) \subseteq \mathrm{Dom}(\mathtt{T}')$ and for every $q \in \mathrm{Dom}(\mathtt{T})$, we have $\mathtt{T}(q) = \mathtt{T}'(q)$.

Let Δ be a set of signed formulas and let $\mathtt{T}$ be a tableau. Define $\mathtt{T}'$ as the tableau $(\mathtt{T}; \mathtt{T}(\lambda) \cup \Delta)$, that is, the $(\mathtt{T}(\lambda) \cup \Delta)$-join of $\mathtt{T}$, as introduced in Definition 1.7.17. We will denote $\mathtt{T}'$ by $\mathtt{T} \uplus \Delta$.

Definition 3.3.3. A tableau is *closed* (*strongly closed*) if every branch is closed (strongly closed). A tableau is *completed* (*strongly completed*) if every branch is either closed (strongly closed) or complete. $\square$

Observe that if $\mathtt{T}$ is (strongly) closed, then so is $\mathtt{T} \uplus \Delta$.

Definition 3.3.4. Let Δ be a set of signed formulas. A Δ-*tableau* is a tableau $\mathtt{T}$ that satisfies the following conditions:

- The root of $\mathtt{T}$ is labeled by Δ, i.e., $\mathtt{T}(\lambda) = \Delta$.
- If q is an interior node of $\mathtt{T}$, one of the following cases occurs:
  1. There is some set of signed formulas Δ' and a signed formula $b\varphi$ with $\mathtt{d}(b\varphi) = (K_0, \ldots, K_{n-1})$ such that $\mathtt{T}(q) = \Delta' \cup \{b\varphi\}$, q has n immediate descendants and $\mathtt{T}(qi) = \Delta' \cup K_i$ for $0 \leq i \leq n-1$.
  2. The node q has one immediate descendant $q0$ and $\mathtt{T}(q0) \subseteq \mathtt{T}(q)$. $\square$

If the first part of the second item of Definition 3.3.4 is applied at a node q, we say that *regular expansion* was used at q. If the second part of the second item of Definition 3.3.4 was used, we say that *thinning* was used at q.

Note that if regular expansion is used at a node q, $b\varphi$ may or may not be a member of Δ'. If $b\varphi \in \Delta'$, then $b\varphi$ and Δ' may not be uniquely determined. For example, suppose that $\mathtt{T}(q) = \{\mathbf{T}(\alpha \wedge \beta), \mathbf{T}\alpha, \mathbf{F}(\neg\beta)\}$ and $\mathtt{T}(q0) = \{\mathbf{T}(\alpha \wedge \beta), \mathbf{T}\alpha, \mathbf{T}\beta, \mathbf{F}(\neg\beta)\}$. Then, $\Delta' = \mathtt{T}(q)$ and we could have either $b\varphi = \mathbf{T}(\alpha \wedge \beta)$ or $b\varphi = \mathbf{F}(\neg\beta)$. If $b\varphi \notin \Delta'$, then $b\varphi$ is uniquely determined as the signed formula in $\mathtt{T}(q)$ that does not belong to $\mathtt{T}(qi)$ for any of the direct descendants

qi of q and Δ' is also uniquely determined as $\mathtt{T}(q) - \{b\varphi\}$. Despite the ambiguity, in the first case, we say that the formula expanded at q is *retained* at q and in the second case, we say that the formula is *removed* at q. The reader will observe that we cannot have both expansion with retention and expansion with removal at the same node. However, it is possible to have both regular expansion (either with retention or with removal) and thinning at the same node.

If the second condition of Definition 3.3.4 is met by an interior node q of an arbitrary tableau $\mathtt{T}$, then we say that $\mathtt{T}$ is *locally consistent* at q.

If regular expansion was used at the node q, we say that $\mathtt{T}$ is *locally conservative* at q. If $\mathtt{T}$ is locally conservative at all its interior nodes, then we say that $\mathtt{T}$ is *conservative*.

Definition 3.3.5. Let Δ be a set of signed formulas. A Δ-*tableau with retention* is a Δ-tableau $\mathtt{T}$ such that at every interior node of $\mathtt{T}$ where thinning is not used, the formula expanded is retained.

A Δ-*tableau with removal* is a Δ-tableau $\mathtt{T}$ such that at every interior node of $\mathtt{T}$ where thinning is not used, the formula expanded is removed. □

Example 3.3.6. Consider the formula $\alpha = (\varphi \to (\psi \to \varphi))$ and the set $\Delta = \{\mathbf{F}\alpha\}$. The Δ-tableau $\mathtt{T}$ given by

$$\begin{aligned}\mathtt{T}(\lambda) &= \{\mathbf{F}(\varphi \to (\psi \to \varphi))\},\\ \mathtt{T}(0) &= \{\mathbf{F}(\varphi \to (\psi \to \varphi)), \mathbf{T}\varphi, \mathbf{F}(\psi \to \varphi)\},\\ \mathtt{T}(00) &= \{\mathbf{F}(\varphi \to (\psi \to \varphi)), \mathbf{T}\varphi, \mathbf{F}(\psi \to \varphi), \mathbf{T}\psi, \mathbf{F}\varphi\}\end{aligned}$$

is represented in Figure 3.1. Clearly, this is a strongly closed tableau with retention. □

Example 3.3.7. Let $\Delta = \{\mathbf{T}((p_0 \wedge (\neg p_1)) \vee ((\neg p_0) \wedge p_1))\}$. A strongly completed Δ-tableau $\mathtt{T}$ with removal (represented in Figure 3.2) is given by

$$\begin{aligned}\mathtt{T}(\lambda) &= \{\mathbf{T}((p_0 \wedge (\neg p_1)) \vee ((\neg p_0) \wedge p_1))\},\\ \mathtt{T}(0) &= \{\mathbf{T}(p_0 \wedge (\neg p_1))\},\\ \mathtt{T}(1) &= \{\mathbf{T}((\neg p_0) \wedge p_1)\},\end{aligned}$$

$$\begin{tikzcd}
\boxed{\mathbf{F}(\varphi \to (\psi \to \varphi))} \arrow[d, dash, "0"'] \\
\boxed{\mathbf{F}(\varphi \to (\psi \to \varphi)), \mathbf{T}\varphi, \mathbf{F}(\psi \to \varphi)} \arrow[d, dash, "0"'] \\
\boxed{\begin{array}{l}\mathbf{F}(\varphi \to (\psi \to \varphi)), \mathbf{T}\varphi, \mathbf{F}(\psi \to \varphi), \\ \mathbf{T}\psi, \mathbf{F}\varphi\end{array}}
\end{tikzcd}$$

Fig. 3.1. Strongly closed tableau of Example 3.3.6.

$$\begin{aligned}
\mathtt{T}(00) &= \{\mathbf{T}p_0, \mathbf{T}(\neg p_1)\}, \\
\mathtt{T}(000) &= \{\mathbf{T}p_0, \mathbf{F}p_1\}, \\
\mathtt{T}(10) &= \{\mathbf{T}(\neg p_0), \mathbf{T}p_1\}, \\
\mathtt{T}(100) &= \{\mathbf{F}p_0, \mathbf{T}p_1\}.
\end{aligned}$$

$\square$

Example 3.3.8. Let $\Delta = \{\mathbf{T}(p_0 \vee p_1), \mathbf{F}(p_0 \wedge p_1)\}$. A strongly completed Δ-tableau $\mathtt{T}$ is given by

$$\begin{aligned}
\mathtt{T}(\lambda) &= \{\mathbf{T}(p_0 \vee p_1), \mathbf{F}(p_0 \wedge p_1)\}, \\
\mathtt{T}(0) &= \{\mathbf{F}(p_0 \wedge p_1), \mathbf{T}p_0\}, \\
\mathtt{T}(1) &= \{\mathbf{F}(p_0 \wedge p_1), \mathbf{T}p_1\}, \\
\mathtt{T}(00) &= \{\mathbf{F}(p_0 \wedge p_1), \mathbf{T}p_0, \mathbf{F}p_0\}, \\
\mathtt{T}(01) &= \{\mathbf{F}(p_0 \wedge p_1), \mathbf{T}p_0, \mathbf{F}p_1\}, \\
\mathtt{T}(10) &= \{\mathbf{F}(p_0 \wedge p_1), \mathbf{T}p_1, \mathbf{F}p_0\}, \\
\mathtt{T}(11) &= \{\mathbf{F}(p_0 \wedge p_1), \mathbf{T}p_1, \mathbf{F}p_1\}.
\end{aligned}$$

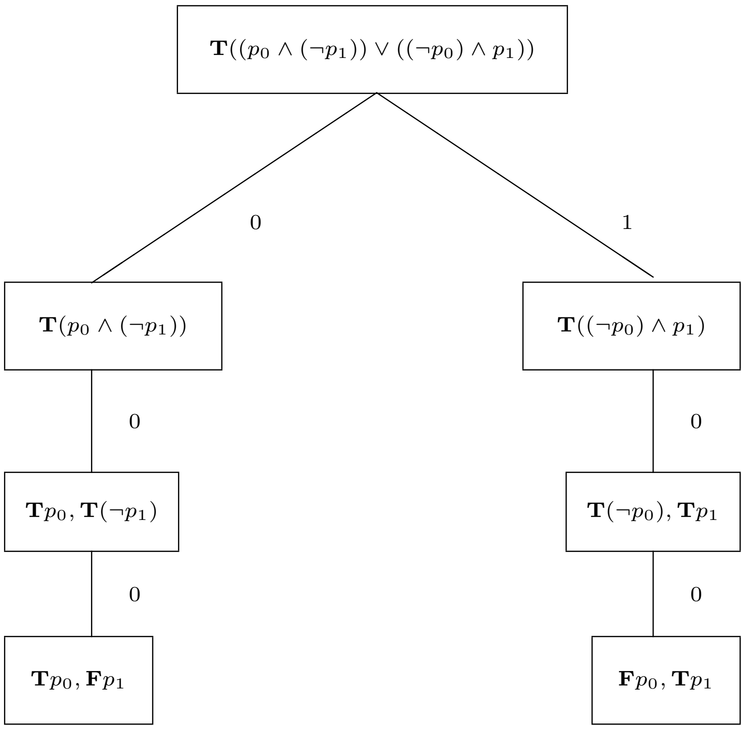

$\mathbf{T}((p_0 \wedge (\neg p_1)) \vee ((\neg p_0) \wedge p_1))$
0
1
$\mathbf{T}(p_0 \wedge (\neg p_1))$
$\mathbf{T}((\neg p_0) \wedge p_1)$
0
0
$\mathbf{T}p_0, \mathbf{T}(\neg p_1)$
$\mathbf{T}(\neg p_0), \mathbf{T}p_1$
0
0
$\mathbf{T}p_0, \mathbf{F}p_1$
$\mathbf{F}p_0, \mathbf{T}p_1$

Fig. 3.2. Strongly completed tableau for Example 3.3.7.

This tableau is represented in Figure 3.3. Note that T is neither with retention nor with removal. ▯

Note that every Δ-tableau is finitely branching. Therefore, by König's Lemma (see Theorem 1.7.7), every strongly closed Δ-tableau T is finite because every branch of T is finite.

For finite, conservative Δ-tableaux with retention, the notions "strongly completed" and "completed" coincide. In other words, if T is a finite, conservative Δ-tableau with retention, then T is strongly completed if and only if it is completed.

Lemma 3.3.9. *Let Δ, Δ' be two sets of signed formulas. If* T *is a Δ-tableau, then* $\mathtt{T} \uplus \Delta'$ *is a $(\Delta \cup \Delta')$-tableau. Moreover, if* T *is a (strongly) closed Δ-tableau, then* $\mathtt{T} \uplus \Delta'$ *is a (strongly) closed $(\Delta \cup \Delta')$-tableau.*

Proof. This straightforward argument is also left to the reader. □

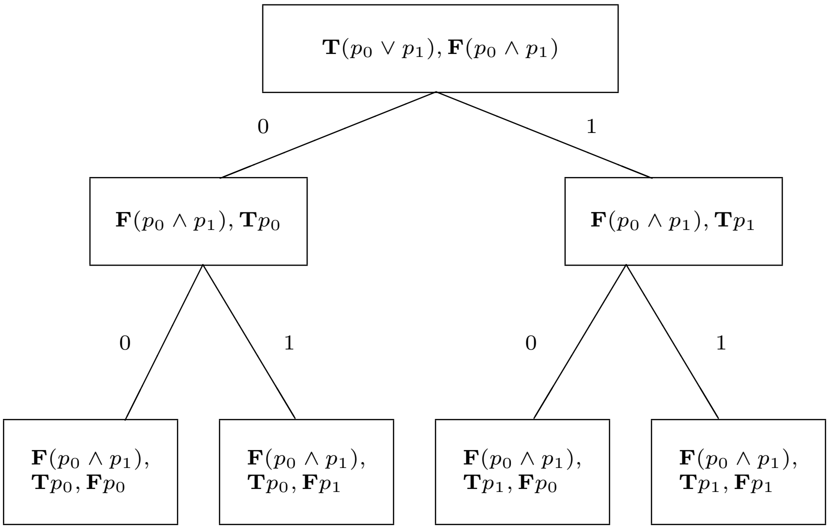

$\mathbf{T}(p_0 \vee p_1), \mathbf{F}(p_0 \wedge p_1)$
0
1
$\mathbf{F}(p_0 \wedge p_1), \mathbf{T}p_0$
$\mathbf{F}(p_0 \wedge p_1), \mathbf{T}p_1$
0
1
0
1
$\mathbf{F}(p_0 \wedge p_1), \mathbf{T}p_0, \mathbf{F}p_0$
$\mathbf{F}(p_0 \wedge p_1), \mathbf{T}p_0, \mathbf{F}p_1$
$\mathbf{F}(p_0 \wedge p_1), \mathbf{T}p_1, \mathbf{F}p_0$
$\mathbf{F}(p_0 \wedge p_1), \mathbf{T}p_1, \mathbf{F}p_1$

Fig. 3.3. Strongly completed tableau for Example 3.3.8.

Theorem 3.3.10. *Let Δ be a set of signed formulas. Suppose that $\mathtt{T}$ is a Δ-tableau, $\Delta = \Delta' \cup \{b\varphi\}$, $\mathtt{d}(b\varphi) = (K_0, \ldots, K_{n-1})$, and $\mathtt{T}(i) = \Delta' \cup K_i$ for $0 \le i \le n-1$.*

For every i, $0 \le i \le n-1$, the following hold:

(1) *$\mathtt{T}_{[i]}$ is a $(\Delta' \cup K_i)$-tableau.*
(2) *If $\mathtt{T}$ is a closed tableau with retention at the root, then $\mathtt{T}_{[i]}$ is a closed $(\Delta \cup K_i)$-tableau.*
(3) *If $\mathtt{T}$ is strongly closed, then so is $\mathtt{T}_{[i]}$.*
(4) *If $\mathtt{T}$ is closed, then $\mathtt{T}_{[i]} \uplus \{b\varphi\}$ is a closed $(\Delta \cup K_i)$-tableau.*

Proof. This straightforward argument is left to the reader. □

Theorem 3.3.11. *Let Δ be a set of signed formulas and let $b\varphi$ be such that $\mathtt{d}(b\varphi) = (K_0, \ldots, K_{n-1})$. Suppose that $\mathtt{T}_i$ is a $(\Delta \cup K_i)$-tableau for $0 \le i \le n-1$. Then, $\mathtt{T} = (\mathtt{T}_0, \ldots, \mathtt{T}_{n-1}; \Delta \cup \{b\varphi\})$ is a $(\Delta \cup \{b\varphi\})$-tableau. Further, if $\mathtt{T}_0, \ldots, \mathtt{T}_{n-1}$ are (strongly) closed, then so is $\mathtt{T}$.*

Proof. The argument is left to the reader. □

The following theorem shows the analyticity of tableaux for signed formulas.

Theorem 3.3.12. *For every node q of a Δ-tableau $\mathtt{T}$, we have*

$$\mathtt{T}(q) \subseteq \mathbf{Bool} \times \mathrm{SUBF}(\{\varphi \mid b\varphi \in \Delta \textit{ for some } b \in \mathbf{Bool}\}).$$

Proof. The argument is by induction on the level of q. If q is the root, then it is clear that

$$\mathtt{T}(q) = \Delta \subseteq \mathbf{Bool} \times \mathrm{SUBF}(\{\varphi \mid b\varphi \in \Delta \text{ for some } b \in \mathbf{Bool}\}).$$

Now suppose that the result holds for all nodes on level i and that r is on level $i+1$. Then, $r = qj$ for some node q on level i. If we used thinning at q, then $j = 0$ and $\mathtt{T}(r) \subseteq \mathtt{T}(q)$ and the inclusion holds for r by the inductive hypothesis. If regular expansion was used, then $\mathtt{T}(r) = \Delta' \cup K$, where $\Delta' \subseteq \mathtt{T}(q)$ and K is a constituent of a formula $b\theta$ from $\mathtt{T}(q)$. Observe that $K \subseteq \mathbf{Bool} \times \mathrm{SUBF}(\theta)$ and this allows us to write

$$\begin{aligned} K &\subseteq \mathbf{Bool} \times \mathrm{SUBF}(\theta) \\ &\subseteq \mathbf{Bool} \times \mathrm{SUBF}(\mathrm{SUBF}(\{\varphi \mid b\varphi \in \Delta \text{ for some } b \in \mathbf{Bool}\})) \\ &\quad \text{(by inductive hypothesis)} \\ &= \mathbf{Bool} \times \mathrm{SUBF}(\{\varphi \mid b\varphi \in \Delta \text{ for some } b \in \mathbf{Bool}\}). \end{aligned}$$

Since, by inductive hypothesis,

$$\mathtt{T}(q) \subseteq \mathbf{Bool} \times \mathrm{SUBF}(\{\varphi \mid b\varphi \in \Delta \text{ for some } b \in \mathbf{Bool}\}),$$

the desired conclusion follows immediately. □

Corollary 3.3.13. *If Δ is a finite set of signed formulas and $\mathtt{T}$ is a Δ-tableau, then $\mathtt{T}(q)$ is finite for every $q \in \mathrm{Dom}(\mathtt{T})$.*

Proof. This follows immediately from Theorem 3.3.12. □

Corollary 3.3.14. *The set of signed formulas that occur in a branch $\mathtt{B}$ of a Δ-tableau $\mathtt{T}$ is a subset of $\mathbf{Bool} \times \mathrm{SUBF}(\{\varphi \mid b\varphi \in \Delta$ for some $b \in \mathbf{Bool}\})$.*

Proof. This follows immediately from Theorem 3.3.12. □

Theorem 3.3.15. *If Δ is a set of signed formulas and* $\mathtt{T}$ *is a Δ-tableau, then for every truth assignment v, v satisfies Δ if and only if v satisfies* $\mathtt{T}(\mathtt{B})$ *for some branch* $\mathtt{B}$ *of* $\mathtt{T}$.

Proof. Let v be a truth assignment which satisfies $\Delta = \mathtt{T}(\lambda)$. We construct a sequence $\lambda = q_0, q_1, \ldots$ of nodes of $\mathtt{T}$ such that for each $k \geq 0$, v satisfies $\mathtt{T}(q_k)$, and either q_{k+1} is an immediate descendant of q_k or q_k is a leaf and $q_{k+1} = q_k$. Suppose that q_k is defined and v satisfies $\mathtt{T}(q_k)$. If q_k is a leaf, we let $q_{k+1} = q_k$. Otherwise, we need to consider two cases:

`Case 1:` Regular expansion is used at q_k. Then, there is a signed formula $b\varphi$ and a set of signed formulas Δ' with $\mathtt{T}(q_k) = \Delta' \cup \{b\varphi\}$ and $\mathtt{d}(b\varphi) = (K_0, \ldots, K_{n-1})$ such that q_k has $q_k0, \ldots, q_k(n-1)$ as immediate descendants and $\mathtt{T}(q_kj) = \Delta' \cup K_j$ for $0 \leq j \leq n-1$. Since v satisfies $b\varphi$, by Theorem 2.7.25, there is a constituent K_j that v satisfies. Let $q_{k+1} = q_kj$, where j is the least number satisfying the condition of the previous sentence. Then, v satisfies $\mathtt{T}(q_{k+1})$.

`Case 2:` Thinning is used at q_k. Then, let $q_{k+1} = q_k0$. Since $\mathtt{T}(q_k0) \subseteq \mathtt{T}(q_k)$, v satisfies $\mathtt{T}(q_{k+1})$.

The desired branch $\mathtt{B}$ is $\{q_0, \ldots, q_k, \ldots\}$.

Conversely, if v satisfies $\mathtt{T}(\mathtt{B})$ for some branch $\mathtt{B}$ of $\mathtt{T}$ then, since $\Delta \subseteq \mathtt{T}(\mathtt{B})$, it is immediate that v satisfies Δ. □

Clearly, if a branch $\mathtt{B}$ of a Δ-tableau $\mathtt{T}$ is closed, then $\mathtt{T}(\mathtt{B})$ is not satisfiable.

Theorem 3.3.16. *Let Δ be a set of signed formulas and let* $\mathtt{T}$ *be a completed Δ-tableau. Then, Δ is satisfiable if and only if* $\mathtt{T}$ *is not closed.*

Proof. Suppose first that Δ is satisfiable. Then, Theorem 3.3.15 implies that $\mathtt{T}$ has a branch that is satisfiable. Therefore, $\mathtt{T}$ is not closed.

Conversely, if $\mathtt{T}$ is not closed, then, since it is completed, there must be a complete branch $\mathtt{B}$ of $\mathtt{T}$. Since $\mathtt{T}(\mathtt{B})$ is a Hintikka set, by Corollary 2.7.29, $\mathtt{T}(\mathtt{B})$ is satisfiable and, by Theorem 3.3.15, Δ is satisfiable. □

Theorem 3.3.17 (Soundness Theorem for Tableaux of Propositional Logic). *Let Δ be a set of signed formulas. If there is a closed Δ-tableau, then Δ is unsatisfiable.*

Proof. This is an immediate consequence of Theorem 3.3.16. □

Combining Theorems 2.7.28 and 3.3.15, we see that with a completed Δ-tableau $\mathtt{T}$, we can not only determine whether Δ is satisfiable but we can actually identify those truth assignments that satisfy Δ.

Theorem 3.3.18. *Let Δ be a set of signed formulas and let $\mathtt{T}$ be a completed Δ-tableau. Then, a truth assignment v satisfies Δ if and only if there is a branch $\mathtt{B}$ of $\mathtt{T}$ such that $\mathtt{T}(\mathtt{B})$ is a Hintikka set and $v(p) = b$ for every $bp \in \mathtt{T}(\mathtt{B})$.*

Proof. This follows immediately from Theorems 2.7.28 and 3.3.15. □

Lemma 3.3.19. *Let $\mathtt{P}$ be a path of a conservative Δ-tableau $\mathtt{T}$ ending in the node q. If $b\varphi \in \mathtt{T}(\mathtt{P}) - \mathtt{T}(q)$, then there exists a constituent K of $b\varphi$ such that $K \subseteq \mathtt{T}(\mathtt{P})$.*

Proof. Let $r \in \mathtt{P}$ be the node closest to q that contains $b\varphi$. Note that $r \neq q$ and for every immediate descendant ri of r, there is a constituent K_i of $b\varphi$ included in $\mathtt{T}(ri)$ because regular expansion was used at r. One of these immediate descendants is in $\mathtt{P}$. □

The following algorithm shows that we can always construct a conservative completed Δ-tableau for a finite set of signed formulas Δ.

Algorithm 3.3.20.
Input: A finite set Δ of signed formulas.
Output: A finite conservative completed Δ-tableau.
Method: We construct a sequence $\mathtt{T}_0, \mathtt{T}_1, \ldots$ of Δ-tableaux such that each $\mathtt{T}_{i+1}$ is a leaf extension of $\mathtt{T}_i$ using the following steps:

(A) Let $\mathtt{T}_0$ be the one-node tableau with root labeled by Δ.
(B) Suppose that $\mathtt{T}_i$ has been defined. Then, if $\mathtt{T}_i$ is completed, the algorithm stops with $\mathtt{T}_i$ as output. Otherwise, $\mathtt{T}_i$ has branches that are neither closed nor complete. Select nondeterministically such a branch $\mathtt{B}$ ending in the leaf q. Then, select nondeterministically a formula $b_q\varphi_q \in \mathtt{T}_i(q)$ and a subset Δ_q of $\mathtt{T}(q)$ such that $\mathtt{T}(q) = \Delta_q \cup \{b_q\varphi_q\}$ and none of the constituents $K_0, \ldots, K_{n-1}$ of $b_q\varphi_q$ is included in $\mathtt{T}(\mathtt{B})$. (By Lemma 3.3.19, $b_q\varphi_q$ exists.) Define $\mathtt{T}_{i+1}$ by adding to $\mathrm{Dom}(\mathtt{T}_i)$ the nodes $q0, \ldots, q(n-1)$ and letting $\mathtt{T}_{i+1}(qj) = \Delta_q \cup K_j$ for $0 \leq j \leq n-1$.

Proof of Correctness: It is clear that every tree $\mathtt{T}_i$ is a Δ-tableau and that if $\mathtt{T}_{i+1}$ is defined, then it is a strict extension of $\mathtt{T}_i$. Therefore, $|\mathtt{T}_i| \geq i+1$.

If $\mathtt{T}_i$ is defined, $qj \in \mathrm{Dom}(\mathtt{T}_i)$, $\mathtt{P}$ is the path leading to q, and $\mathtt{P}'$ is the path leading to qj (that is, $\mathtt{P}' = \mathtt{P} \cup \{qj\}$), then $\mathtt{T}_i(\mathtt{P}) \subset \mathtt{T}_i(\mathtt{P}')$. This implies that if q is at level k of $\mathtt{T}_i$, then $|\mathtt{T}_i(\mathtt{P})| \geq k$, where $\mathtt{P}$ is the path leading to q. Also, observe that if $\mathtt{T}_i$ is defined and $\mathtt{P}$ is a closed path in $\mathtt{T}_i$, then $\mathtt{P}$ is a branch of $\mathtt{T}_i$.

Let $M = |\mathrm{SUBF}(\{\varphi \mid b\varphi \in \Delta \text{ for some } b \in \mathbf{Bool}\})|$. By Theorem 3.3.12, no path of a Δ-tableau can contain more than $2M$ formulas and therefore no path of a Δ-tableau $\mathtt{T}_i$ constructed as above can have length greater than $2M$.

If $\mathtt{T}_i$ is defined, then, by Exercise 90 of Chapter 1, it contains no more than N nodes, where $N = 2^{2M+1} - 1$, since for our set of connectives, a node has no more than two immediate descendants. Since every tableau $\mathtt{T}_i$ has at least $i+1$ nodes, we have $i \leq N-1$. Thus, the algorithm must halt necessarily producing a completed Δ-tableau. ▮

Example 3.3.21. One of the possible completed Δ-tableaux for the set of signed formulas $\Delta = \{\mathbf{T}(p_0 \wedge p_1), \mathbf{F}((p_0 \wedge p_1) \vee p_2)\}$ that could result from applying Algorithm 3.3.20 is shown in Figure 3.4. Since the tableau is closed, we conclude that the $\{\mathbf{T}(p_0 \wedge p_1), \mathbf{F}((p_0 \wedge p_1) \vee p_2)\}$ is unsatisfiable. Note, however, that this tableau is not strongly closed. We will return to this point later. ▯

The correctness of Algorithm 3.3.20, with its nondeterminism, implies that in the construction of a conservative completed Δ-tableau for a finite set of signed formulas Δ, any legal choice of a branch and formula in the branch to expand at each step will lead eventually to the desired tableau. Also, if the formula expanded is retained at every step, we end up with a completed tableau with retention; if the formula expanded is removed at every step, we obtain a completed tableau with removal.

To recapitulate the development discussed so far in this section, if we have a finite set of signed formulas Δ and we wish to determine whether this set of formulas is satisfiable, we first produce a completed Δ-tableau $\mathtt{T}$ (possibly using Algorithm 3.3.20) and then inspect the branches. If all the branches of $\mathtt{T}$ are closed, then Δ is not satisfiable. Otherwise, each complete branch of $\mathtt{T}$ (which is labeled

$$\begin{tikzcd}
\boxed{\mathbf{T}(p_0 \wedge p_1), \mathbf{F}((p_0 \wedge p_1) \vee p_2)} \arrow[d, dash, "0"] \\
\boxed{\mathbf{T}p_0, \mathbf{T}p_1, \mathbf{F}((p_0 \wedge p_1) \vee p_2)} \arrow[d, dash, "0"] \\
\boxed{\mathbf{T}p_0, \mathbf{T}p_1, \mathbf{F}(p_0 \wedge p_1), \mathbf{F}p_2}
\end{tikzcd}$$

Fig. 3.4. Completed tableau for $\{\mathbf{T}(p_0 \wedge p_1), \mathbf{F}((p_0 \wedge p_1) \vee p_2)\}$.

by a Hintikka set) determines a partial truth assignment and a truth assignment v satisfies Δ if and only if it extends one of these partial truth assignments.

The following theorem is a special case of the Completeness Theorem for Tableaux (Theorem 3.3.34).

Theorem 3.3.22. *Let Δ be a finite set of signed formulas. If Δ is unsatisfiable, then there exists a conservative finite closed Δ-tableau.*

Proof. The theorem follows immediately from Theorem 3.3.16 and the correctness of Algorithm 3.3.20. □

By the remark made after Algorithm 3.3.20, we could strengthen Theorem 3.3.22 to obtain a closed tableau with retention or with removal.

Corollary 3.3.23. *Let Δ be a finite set of signed formulas. Then, the following three statements are equivalent:*

(1) *Δ is unsatisfiable.*
(2) *There exists a closed Δ-tableau.*
(3) *There exists a conservative closed Δ-tableau.*

Proof. The corollary combines Theorems 3.3.17 and 3.3.22. □

Example 3.3.24. The set of signed formulas $\Delta = \{\mathbf{F}\alpha\}$, where $\alpha = (\varphi \to (\psi \to \varphi))$, considered in Example 3.3.6 is unsatisfiable because the Δ-tableau $\mathtt{T}$ given in that example is closed. Therefore, the signed formula $\mathbf{F}\alpha$ is not satisfiable, which means that for every truth assignment v, we have $v(\alpha) = \mathbf{T}$; in other words, α is a tautology. □

Example 3.3.25. The set of signed formulas $\Delta = \{\mathbf{T}(p_0 \vee p_1), \mathbf{F}(p_0 \wedge p_1)\}$ considered in Example 3.3.8 is satisfiable since the completed Δ-tableau given in that example has two branches that are labeled by Hintikka sets. We conclude that the set of truth assignments that satisfy Δ consists of those truth assignments v such that $v(p_0) = \mathbf{T}$ and $v(p_1) = \mathbf{F}$ (corresponding to the Hintikka set $\mathtt{T}(\lambda) \cup \mathtt{T}(0) \cup \mathtt{T}(01)$) and of those truth assignments v' such that $v'(p_0) = \mathbf{F}$ and $v'(p_1) = \mathbf{T}$ (corresponding to the Hintikka set $\mathtt{T}(\lambda) \cup \mathtt{T}(1) \cup \mathtt{T}(10)$). □

Example 3.3.26. The existence of the closed tableau shown in Figure 3.4 shows that the set $\{\mathbf{T}(p_0 \wedge p_1), \mathbf{F}((p_0 \wedge p_1) \vee p_2)\}$ is unsatisfiable, which gives the unsurprising logical implication $\{(p_0 \wedge p_1)\} \models ((p_0 \wedge p_1) \vee p_2)$. □

For the sake of Section 3.5, we present a variant of Algorithm 3.3.20 that generates strongly completed tableaux.

Algorithm 3.3.27.
Input: A finite set Δ of signed formulas.
Output: A conservative finite strongly completed Δ-tableau.
Method: Apply the method of Algorithm 3.3.20 with "completed" replaced by "strongly completed" and "closed" replaced by "strongly closed." Note that if a branch $\mathtt{B}$ in any of the Δ-tableaux $\mathtt{T}_i$ is neither strongly closed nor complete, the branch cannot contain both $\mathbf{T}r$ and $\mathbf{F}r$, for some variable r, because if it did, then by Lemma 3.3.19, $\mathbf{T}r$ and $\mathbf{F}r$ would belong to the leaf of $\mathtt{B}$ and thus $\mathtt{B}$ would be strongly closed.

Proof of Correctness: The proof is the same as the argument in the correctness proof of Algorithm 3.3.20. ■

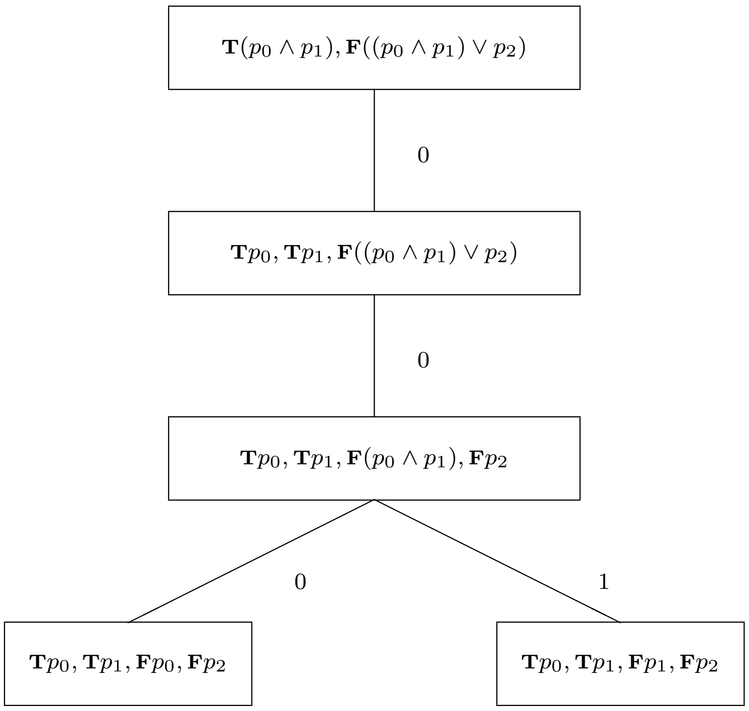

$\mathbf{T}(p_0 \wedge p_1), \mathbf{F}((p_0 \wedge p_1) \vee p_2)$
0
$\mathbf{T}p_0, \mathbf{T}p_1, \mathbf{F}((p_0 \wedge p_1) \vee p_2)$
0
$\mathbf{T}p_0, \mathbf{T}p_1, \mathbf{F}(p_0 \wedge p_1), \mathbf{F}p_2$
0
1
$\mathbf{T}p_0, \mathbf{T}p_1, \mathbf{F}p_0, \mathbf{F}p_2$
$\mathbf{T}p_0, \mathbf{T}p_1, \mathbf{F}p_1, \mathbf{F}p_2$

Fig. 3.5. A strongly completed Δ-tableau.

Again, Algorithm 3.3.27 can be used to produce conservative strongly completed tableaux with retention or with removal, as desired.

Example 3.3.28. Figure 3.5 shows one of the possible strongly completed Δ-tableaux for the set of signed formulas $\Delta = \{\mathbf{T}(p_0 \wedge p_1), \mathbf{F}((p_0 \wedge p_1) \vee p_2)\}$ that can be obtained by applying Algorithm 3.3.27. This application consists of retracing the steps followed by Algorithm 3.3.20 to produce the tableau of Example 3.3.21 and then going one step further to produce a strongly closed tableau. □

Observe that a tableau that is both strongly completed and closed is also strongly closed. This remark is used in the argument of the following theorem.

Theorem 3.3.29. *Let Δ be a finite set of signed formulas. If Δ is unsatisfiable, then there exists a strongly closed conservative Δ-tableau.*

Proof. The theorem follows from Theorem 3.3.16 and the correctness of Algorithm 3.3.27. □

Note that Theorem 3.3.29 can be strengthened by asserting the existence of conservative strongly closed tableaux with retention or with removal.

Corollary 3.3.30. *Let Δ be a finite set of signed formulas. Then, the following statements are equivalent:*

(1) Δ *is unsatisfiable.*
(2) *There exists a strongly closed Δ-tableau.*
(3) *There exists a conservative strongly closed Δ-tableau.*

Proof. The corollary combines Theorems 3.3.17 and 3.3.29. □

Corollary 3.3.30 can be rephrased in terms of formal systems.

Definition 3.3.31. $\mathcal{F}^{\text{tabl,cons}}$ is the formal system whose set of objects is the collection of all finite sets of signed formulas, set of axioms is the collection of all closed finite sets of signed formulas, and single rule of inference is

$$\frac{\Delta \cup K_0, \ldots, \Delta \cup K_{n-1}}{\Delta \cup \{b\varphi\}} \, R\,,$$

where Δ is a finite set of signed formulas, φ is not a statement variable, and $\mathtt{d}(b\varphi) = (K_0, \ldots, K_{n-1})$.

If we add the *thinning rule*

$$\frac{\Delta}{\Delta'} \, R_{\text{thin}}$$

where Δ, Δ' are finite sets of signed formulas such that $\Delta \subseteq \Delta'$, we obtain the formal system $\mathcal{F}^{\text{tabl}}$. ▯

Note that by Corollary 3.3.13, an $\mathcal{F}^{\text{tabl}}$-deduction ($\mathcal{F}^{\text{tabl,cons}}$-deduction) tree for a finite set of signed formulas Δ is the same thing as a finite (conservative) Δ-tableau and an $\mathcal{F}^{\text{tabl}}$-proof tree ($\mathcal{F}^{\text{tabl,cons}}$-proof tree) for Δ is the same thing as a (conservative) strongly closed Δ-tableau. Thus, the set $\mathcal{PT}_{\mathcal{F}^{\text{tabl}}}$ ($\mathcal{PT}_{\mathcal{F}^{\text{tabl,cons}}}$) is the same as the set of all tableaux $\mathtt{T}$ such that $\mathtt{T}$ is a (conservative) strongly closed $\mathtt{T}(\lambda)$-tableau and $\mathtt{T}(\lambda)$ is finite.

Theorem 3.3.32 (Soundness and Completeness of the Formal Systems $\mathcal{F}^{\text{tabl}}$ and $\mathcal{F}^{\text{tabl,cons}}$). *The formal systems $\mathcal{F}^{tabl}$ and $\mathcal{F}^{tabl,cons}$ are sound and complete with respect to the collection of all finite unsatisfiable sets of signed formulas.*

Proof. The result follows from Corollary 3.3.30 and from the remark which precedes the theorem. □

By the remark following Theorem 3.3.29, the formal systems $\mathcal{F}^{\text{tabl}}$ and $\mathcal{F}^{\text{tabl,cons}}$ remain complete (and, of course, sound) if in the definition of the rule R, we add one of the additional restrictions: $b\varphi \in \Delta$ or $b\varphi \notin \Delta$.

We have seen that for a finite set Δ of signed formulas, we can construct a finite conservative completed Δ-tableau. In the construction, we have complete leeway in choosing the branch and unexpanded signed formula in the branch to expand. We now give a construction which accomplishes the same thing for arbitrary sets of signed formulas (finite or not). In order for the construction to work, certain restrictions have to be made on the choice of branch and signed formula to expand. We refer to the process as a "construction" rather than an "algorithm" because for infinite sets of signed formulas, it may never halt.

Construction 3.3.33.
Input: A set Δ of signed formulas.
Output: A (finite or infinite) sequence $\mathtt{T}_0, \mathtt{T}_1, \ldots$ of finite Δ-tableaux such that each $\mathtt{T}_{i+1}$ is a leaf extension of $\mathtt{T}_i$ and $\mathtt{T} = \bigcup\{\mathtt{T}_i \mid i \geq 0\}$ is a conservative completed Δ-tableau.

Method:

(A) Let $\mathtt{T}_0$ be the one-node Δ-tableau with root labeled by Δ.
(B) Suppose that $\mathtt{T}_i$ has been defined. Then, if $\mathtt{T}_i$ is completed, the construction stops with $\mathtt{T}_i$ as the output. Otherwise, $\mathtt{T}_i$ has branches that are neither closed nor complete. Select nondeterministically among the shortest such branches a branch

B ending in the leaf q. Then, let $b_q\varphi_q \in \mathtt{T}_i(q)$ be the first signed formula in the standard order such that no constituent $K_0, \ldots, K_{n-1}$ of $b_q\varphi_q$ is in $\mathtt{T}(\mathtt{B})$. Pick a subset Δ_q of $\mathtt{T}(q)$ such that $\mathtt{T}(q) = \Delta_q \cup \{b_q\varphi_q\}$. (By Lemma 3.3.19, $b_q\varphi_q$ exists and actually is the first formula that occurs in B such that none of its constituents occurs in B.) Define $\mathtt{T}_{i+1}$ by adding to $\mathrm{Dom}(\mathtt{T}_i)$ the nodes $q0, \ldots, q(n-1)$ and defining $\mathtt{T}_{i+1}(qj) = \Delta_q \cup K_j$ for $0 \leq j \leq n-1$.

Proof of Correctness: We begin by observing that for each i such that $\mathtt{T}_{i+1}$ is defined, there is a unique node q such that q is a leaf of $\mathtt{T}_i$ but not of $\mathtt{T}_{i+1}$. We will refer to q as the node expanded at stage $i+1$ of the construction. Further, if $i < j$ and $\mathtt{T}_{j+1}$ is defined, then the node expanded at stage $j+1$ is different from the node expanded at stage $i+1$. It follows that for each n, there are only finitely many i such that the node expanded at stage i has length n.

We also observe that if $q \in \mathrm{Dom}(\mathtt{T}_i)$, q is not a leaf of $\mathtt{T}_i$, P is the path leading to q, and $b\varphi$ is the first signed formula in $\mathtt{T}_i(\mathtt{P})$ such that no constituent of $b\varphi$ is contained in $\mathtt{T}_i(\mathtt{P})$, then every immediate descendant of q in $\mathtt{T}_i$ contains a constituent of $b\varphi$.

Each $\mathtt{T}_i$ is a conservative Δ-tableau and therefore T is a conservative Δ-tableau.

Suppose that T is not completed. Then, there is a branch B that is neither closed not complete. Let $b\varphi$ be the first signed formula in the standard order contained in $\mathtt{T}(\mathtt{B})$ which is not a signed variable such that none of its constituents is in $\mathtt{T}(\mathtt{B})$. Let $q \in \mathtt{B}$ be such that $b\varphi \in \mathtt{T}(\mathtt{P})$ and for all $b'\varphi' \in \mathtt{T}(\mathtt{B})$ that occur before $b\varphi$ in the standard ordering and are not a signed variable, a constituent of $b'\varphi'$ is contained in $\mathtt{T}(\mathtt{P})$, where P is the path leading to q.

Suppose initially that B contains some immediate descendant qj of q and let k be such that $qj \in \mathrm{Dom}(\mathtt{T}_k)$. Then, $\mathtt{T}_k(\mathtt{P}) = \mathtt{T}(\mathtt{P})$, so $\mathtt{T}_k(qj)$ contains a constituent of $b\varphi$, which means that $\mathtt{T}(\mathtt{B})$ contains a constituent of $b\varphi$. We thus obtain a contradiction. If there is no such immediate descendant, B is the path P leading to q. Choose k such that $q \in \mathrm{Dom}(\mathtt{T}_k)$ and, at stage $k+1$, no node r such that $|r| \leq |q|$ is expanded. Then, q is a leaf of $\mathtt{T}_k$, $\mathtt{T}_k(\mathtt{P})$ is not closed, $b\varphi \in \mathtt{T}_k(\mathtt{P})$, and none of the constituents of $b\varphi$ is contained in $\mathtt{T}_k(\mathtt{P})$. This makes q a node eligible to be expanded at stage $k+1$, so the node expanded at stage $k+1$ must have length no greater than $|q|$, thus contradicting our choice of k. □

Again, in the previous construction, we have the leeway to produce a tableau with retention or with removal.

Theorem 3.3.34 (Completeness Theorem for Tableaux of Propositional Logic). *Let Δ be a set of signed formulas. If Δ is unsatisfiable, then there exists a finite, conservative, closed Δ-tableau.*

Proof. If Δ is unsatisfiable, then Construction 3.3.33 produces a conservative, completed Δ-tableau $\mathtt{T}$ which, by Theorem 3.3.16, is closed. Thus, the single argument to be made is that $\mathtt{T}$ is finite. Since $\mathtt{T}$ is finitely branching, by König's Lemma, it suffices to show that $\mathtt{T}$ has no infinite branch. Suppose that $\mathtt{B}$ is an infinite branch of $\mathtt{T}$ and let $\mathtt{B}_i = \mathtt{B} \cap \mathrm{Dom}(\mathtt{T}_i)$ for $i \geq 0$, where $\mathtt{T}_0, \mathtt{T}_1, \ldots$ is the sequence of finite tableaux that we built in Construction 3.3.33. By Theorem 1.7.28, $\mathtt{B} = \bigcup\{\mathtt{B}_i \mid i \geq 0\}$ and each $\mathtt{B}_i$ is a branch of $\mathtt{T}_i$. Since $\mathtt{T}$ is a closed tree, $\mathtt{B}$ is a closed branch and, hence, there must be an i_0 with $\mathtt{B}_{i_0}$ closed. It is easily seen that a closed branch of $\mathtt{T}_i$ remains a (closed) branch of $\mathtt{T}_{i+1}$ and, therefore, of $\mathtt{T}_j$ for all $j \geq i$. Thus, $\mathtt{B}_j = \mathtt{B}_{i_0}$ for $j \geq i_0$ and $\mathtt{B} = \mathtt{B}_{i_0}$, contradicting the assumption that $\mathtt{B}$ is infinite. □

Corollary 3.3.35. *Let Δ be a set of signed formulas. Then, the following conditions are equivalent:*

(1) *Δ is unsatisfiable.*
(2) *There exists a finite, closed Δ-tableau.*
(3) *There exists a closed Δ-tableau.*
(4) *There exists a finite, conservative, closed Δ-tableau.*
(5) *There exists a conservative, closed Δ-tableau.*

Proof. This follows immediately from Theorems 3.3.17 and 3.3.34. □

Example 3.3.36. Consider the set of signed formulas

$$\Delta = \{\mathbf{T}p_0\} \cup \{\mathbf{T}(p_i \to p_{i+1}) \mid i \in \mathbf{N}\}.$$

Note that if $i \leq j$, then $\mathbf{T}(p_i \to p_{i+1})$ precedes $\mathbf{T}(p_j \to p_{j+1})$ in the standard order as the reader can easily verify. Using Construction 3.3.33, we construct the sequence of conservative Δ-tableaux

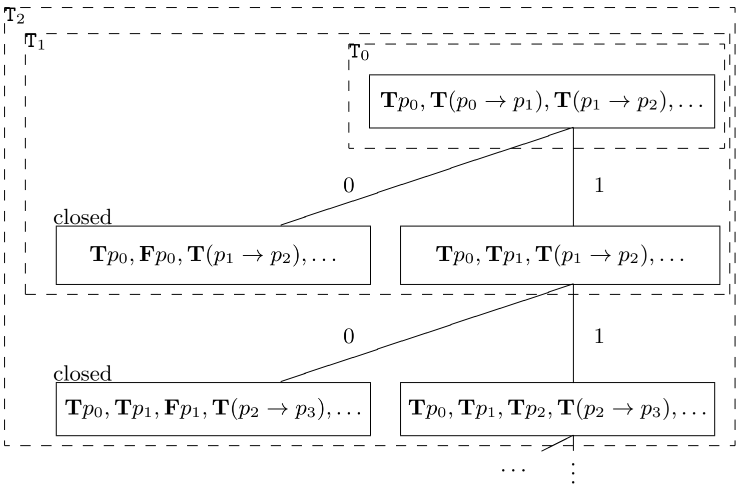

$\mathtt{T}_2$
$\mathtt{T}_1$
$\mathtt{T}_0$
$\mathbf{T}p_0, \mathbf{T}(p_0 \to p_1), \mathbf{T}(p_1 \to p_2), \dots$
0
1
closed
$\mathbf{T}p_0, \mathbf{F}p_0, \mathbf{T}(p_1 \to p_2), \dots$
$\mathbf{T}p_0, \mathbf{T}p_1, \mathbf{T}(p_1 \to p_2), \dots$
0
1
closed
$\mathbf{T}p_0, \mathbf{T}p_1, \mathbf{F}p_1, \mathbf{T}(p_2 \to p_3), \dots$
$\mathbf{T}p_0, \mathbf{T}p_1, \mathbf{T}p_2, \mathbf{T}(p_2 \to p_3), \dots$

Fig. 3.6. Tableau sequence.

$\mathtt{T}_0, \mathtt{T}_1, \mathtt{T}_2, \dots$ (see Figure 3.6). Note that in the tableau $\mathtt{T}_i$ only the node 1^i is not closed and contains a formula that can be expanded, namely, we have $b_{1^i}\varphi_{1^i} = \mathbf{T}(p_i \to p_{i+1})$. The tree $\mathtt{T}$ contains an infinite branch $\mathtt{B} = \{1^i \mid i \geq 0\}$ and $\mathtt{T}(\mathtt{B})$ is the Hintikka set $\Delta \cup \{\mathbf{T}p_i \mid i \geq 0\}$. Therefore, Δ is satisfiable and there is only one truth assignment v that satisfies Δ, given by $v(p_i) = \mathbf{T}$ for $i \geq 0$. ∎

Paralleling what we did for finite sets of signed formulas, we now give a variant of Construction 3.3.33 that produces a strongly completed Δ-tableau where Δ is an arbitrary set of signed formulas.

Construction 3.3.37.
Input: A set Δ of signed formulas.
Output: A (finite or infinite) sequence $\mathtt{T}_0, \mathtt{T}_1, \dots$ of finite Δ-tableaux such that each $\mathtt{T}_{i+1}$ is a leaf extension of $\mathtt{T}_i$ and $\mathtt{T} = \bigcup\{\mathtt{T}_i \mid i \geq 0\}$ is a conservative strongly completed Δ-tableau.
Method:

(A) Let $\mathtt{T}_0$ be the one-node Δ-tableau with root labeled by Δ.
(B) Suppose that $\mathtt{T}_i$ has been defined. Then, if $\mathtt{T}_i$ is strongly completed, the construction stops with $\mathtt{T}_i$ as the output. Otherwise, $\mathtt{T}_i$ has branches that are neither strongly closed nor complete.

Select nondeterministically among the shortest such branches a branch $\mathtt{B}$ ending in the leaf q. Then, let $b_q\varphi_q \in \mathtt{T}_i(\mathtt{B})$ be the first signed formula in the standard order such that no constituent $K_0, \ldots, K_{n-1}$ of $b_q\varphi_q$ is in the branch. (We can show the existence of such a formula as follows. Since $\mathtt{T}_i(\mathtt{B})$ is not a Hintikka set, either $\{\mathbf{T}p, \mathbf{F}p\} \subseteq \mathtt{T}_i(\mathtt{B})$ for some variable p or such a formula exists in the branch. By Lemma 3.3.19, in the first case, we would have $\{\mathbf{T}p, \mathbf{F}p\} \subseteq \mathtt{T}_i(q)$, which would contradict the fact that $\mathtt{B}$ is not strongly closed.) By Lemma 3.3.19, $b_q\varphi_q \in \mathtt{T}_i(q)$. Pick a subset Δ_q of $\mathtt{T}(q)$ such that $\mathtt{T}(q) = \Delta_q \cup \{b_q\varphi_q\}$. Define $\mathtt{T}_{i+1}$ by adding to $\mathrm{Dom}(\mathtt{T}_i)$ the nodes $q0, \ldots, q(n-1)$ and defining $\mathtt{T}_{i+1}(qj) = \Delta_q \cup K_j$ for $0 \leq j \leq n-1$.

Proof of Correctness: Note that the statements contained in the first three paragraphs of the proof of correctness of Construction 3.3.33 remain valid for the current construction. Further, if $\mathtt{T}_i(q)$ is defined, then, for some $j \leq i$, q is a leaf of $\mathtt{T}_j$. This can be shown easily by induction on i. Also, if $\mathtt{T}_i(q)$ is defined, q is a leaf of $\mathtt{T}_i$, and $\mathtt{T}_i(q)$ contains both $\mathbf{T}\varphi$ and $\mathbf{F}\varphi$ for some formula φ, then for every $k \geq i$, q is a leaf of $\mathtt{T}_k$ and, therefore, q is a leaf of $\mathtt{T}$. This can be verified by induction on k.

Suppose that $\mathtt{T}$ is not strongly completed. Then, there is a branch $\mathtt{B}$ that is neither strongly closed not complete. Observe that $\mathtt{T}(\mathtt{B})$ cannot contain both $\mathbf{T}p$ and $\mathbf{F}p$ for any statement variable p. Indeed, if r, r' are two nodes of $\mathtt{B}$ such that $\mathbf{T}p \in \mathtt{T}(r)$ and $\mathbf{F}p \in \mathtt{T}(r')$, then, by Lemma 3.3.19, q, the longer of r and r' is such that $\{\mathbf{T}p, \mathbf{F}p\} \subseteq \mathtt{T}(q)$. By the previous remark, there is an h such that q is a leaf of $\mathtt{T}_h$ and consequently, q is a leaf of $\mathtt{T}$. This implies that $\mathtt{B}$ is strongly closed, which contradicts our assumption. Thus, $\mathtt{T}(\mathtt{B})$ satisfies the first condition of the definition of a Hintikka set. The noncompleteness of $\mathtt{B}$ implies the existence of a signed formula which is not a signed variable such that none of its constituents is included in $\mathtt{T}(\mathtt{B})$. Let $b\varphi$ be the first signed formula in the standard order with this property contained in $\mathtt{T}(\mathtt{B})$. Take $q \in \mathtt{B}$ such that $b\varphi \in \mathtt{T}(\mathtt{P})$, where $\mathtt{P}$ is the path leading to q, such that for all $b'\varphi' \in \mathtt{T}(\mathtt{B})$ that are not signed variables and occur before $b\varphi$ in the standard ordering, a constituent of $b'\varphi'$ is contained in $\mathtt{T}(\mathtt{P})$.

Suppose initially that $\mathtt{B}$ contains some immediate descendant qj of q and let k be such that $qj \in \mathrm{Dom}(\mathtt{T}_k)$. Then, $\mathtt{T}_k(\mathtt{P}) = \mathtt{T}(\mathtt{P})$, so

$\mathtt{T}(qj)$ contains a constituent of $b\varphi$, which means that $\mathtt{T}(\mathtt{B})$ contains a constituent of $b\varphi$. We thus obtain a contradiction. If there is no such immediate descendant, $\mathtt{B}$ is the path $\mathtt{P}$ leading to q. Choose k such that $q \in \mathrm{Dom}(\mathtt{T}_k)$ and, at stage $k+1$, no node r such that $|r| \leq |q|$ is expanded. Then, q is a leaf of $\mathtt{T}_k$, $\mathtt{P}$ is not strongly closed, $b\varphi \in \mathtt{T}_k(\mathtt{P})$, and none of the constituents of $b\varphi$ is contained in $\mathtt{T}_k(\mathtt{P})$. This makes q a node eligible to be expanded at stage $k+1$, so the node expanded at stage $k+1$ must have length no greater than $|q|$, thus contradicting our choice of k. □

We saw earlier that a strongly closed Δ-tableau is finite.

Theorem 3.3.38 (Strong Completeness Theorem for Tableaux of Propositional Logic). *Let Δ be a set of signed formulas. If Δ is unsatisfiable, then there exists a conservative strongly closed (hence, finite) Δ-tableau.*

Proof. If Δ is unsatisfiable, then Construction 3.3.37 yields a conservative strongly completed Δ-tableau $\mathtt{T}$ which, by Theorem 3.3.16, is closed. As observed earlier, a strongly completed closed Δ-tableau is strongly closed. □

Generalizing a remark we made earlier, we note that Theorem 3.3.38 can be strengthened by asserting the existence of strongly closed tableaux with retention or with removal.

Corollary 3.3.39. *Let Δ be a set of signed formulas. Then, the following conditions are equivalent:*

(1) *Δ is unsatisfiable.*
(2) *There exists a finite, closed Δ-tableau.*
(3) *There exists a finite, conservative, closed Δ-tableau.*
(4) *There exists a closed Δ-tableau.*
(5) *There exists a conservative closed Δ-tableau.*
(6) *There exists a strongly closed Δ-tableau.*
(7) *There exists a conservative strongly closed Δ-tableau.*

Proof. This follows from Corollary 3.3.35 and Theorem 3.3.38. □

Theorem 3.3.40. *There is an effective, syntactic construction that begins with a strongly closed Δ-tableau $\mathtt{T}$ and produces a strongly closed Δ'-tableau $\mathtt{T}'$, where Δ' is a finite subset of Δ.*

Proof. We proceed recursively on the size of $\mathtt{T}$. If $\mathtt{T}$ is a one node tree, then Δ contains $\mathbf{T}\varphi, \mathbf{F}\varphi$ for some formula φ, so $\mathtt{T}'$ is the one-node tableau with $\mathtt{T}'(\lambda) = \{\mathbf{T}\varphi, \mathbf{F}\varphi\}$ and $\Delta' = \{\mathbf{T}\varphi, \mathbf{F}\varphi\}$.

Let $\mathtt{T}(\lambda) = \hat{\Delta} \cup \{b\varphi\}$, where $\mathtt{T}(i) = \hat{\Delta} \cup K_i$ and $\mathtt{d}(b\varphi) = (K_0, \ldots, K_{n-1})$. Applying the process recursively to the tableaux $\mathtt{T}_{[i]}$, we obtain strongly closed Δ'_i-tableaux $\mathtt{T}'_i$, where $\Delta'_i \subseteq \hat{\Delta} \cup K_i$, for $0 \leq i \leq n-1$. Define the finite sets of signed formulas $\Delta''_i = \Delta'_i \cap \hat{\Delta}$, for $0 \leq i \leq n-1$. Starting from each of the tableaux $\mathtt{T}'_i$, we obtain by thinning at the root, a strongly closed $(\Delta''_0 \cup \cdots \cup \Delta''_{n-1} \cup K_i)$-tableau $\mathtt{T}''_i$. Finally, the construction returns the strongly closed tableau $(\mathtt{T}''_0, \ldots, \mathtt{T}''_{n-1}; \Delta''_0 \cup \cdots \cup \Delta''_{n-1} \cup \{b\varphi\})$, whose root is labeled by the finite subset $\Delta''_0 \cup \cdots \cup \Delta''_{n-1} \cup \{b\varphi\}$ of Δ.

If thinning is applied at the root of $\mathtt{T}$, we have $\mathtt{T}(0) = \Delta_0$, where $\Delta_0 \subseteq \Delta$. We apply the process recursively to $\mathtt{T}_{[0]}$ to obtain a strongly closed Δ'_0-tableau $\mathtt{T}'$, for some finite subset Δ'_0 of Δ_0. Since Δ'_0 is a finite subset of Δ, the construction returns $\mathtt{T}'$. □

We can now reprove the Compactness Theorem (in a formulation using signed formulas), independently of our argument in Theorem 2.4.3.

Theorem 3.3.41 (Compactness Theorem for Signed Formulas of Propositional Logic). *Let Δ be a set of signed formulas. Then Δ is satisfiable if and only if every finite subset of Δ is satisfiable.*

Proof. If Δ is satisfiable, then clearly every finite subset of Δ is satisfiable.

Conversely, let Δ be unsatisfiable. By the Strong Completeness Theorem for Tableaux, there is (finite) conservative strongly closed Δ-tableau $\mathtt{T}$. By Theorem 3.3.40, there is a strongly closed Δ'-tableau for some finite subset Δ' of Δ. The Soundness Theorem for Tableaux implies that Δ' is unsatisfiable. □

It is instructive to compare the argument of Theorem 3.3.41 with Exercise 118 of Chapter 2 which uses explicitly the notion of consistency property.

The proof of the Compactness Theorem, together with the proof of Theorem 3.3.40, gives a recursive construction of a finite unsatisfiable subset Δ' of an unsatisfiable set of signed formulas Δ. An alternative, nonrecursive construction is discussed in Supplement 24.

Corollary 3.3.39 can be rephrased in terms of formal systems.

Definition 3.3.42. $\mathcal{F}^{\text{tabl},\infty,\text{cons}}$ is the formal system whose set of objects is the collection of all sets of signed formulas, set of axioms is the collection of all closed sets of signed formulas, and single rule of inference is

$$\frac{\Delta \cup K_0, \ldots, \Delta \cup K_{n-1}}{\Delta \cup \{b\varphi\}}\ {\scriptstyle R},$$

where φ is not a statement variable and $\mathtt{d}(b\varphi) = (K_0, \ldots, K_{n-1})$.

If we add the *thinning rule*

$$\frac{\Delta}{\Delta'}\ {\scriptstyle R_{\text{thin}}}$$

where Δ, Δ' are sets of signed formulas such that $\Delta \subseteq \Delta'$, we obtain the formal system $\mathcal{F}^{\text{tabl},\infty}$. ▯

A general $\mathcal{F}^{\text{tabl},\infty}$-deduction tree ($\mathcal{F}^{\text{tabl},\infty,\text{cons}}$-deduction tree) for Δ is the same thing as a Δ-tableau (conservative Δ-tableau) and an $\mathcal{F}^{\text{tabl},\infty}$-proof tree ($\mathcal{F}^{\text{tabl},\infty,\text{cons}}$-proof tree) for Δ is the same thing as a strongly closed Δ-tableau (conservative strongly closed Δ-tableau). Thus, the set $\mathcal{PT}_{\mathcal{F}^{\text{tabl},\infty}}$ ($\mathcal{PT}_{\mathcal{F}^{\text{tabl},\infty,\text{cons}}}$) equals the set of all tableaux $\mathtt{T}$ such that $\mathtt{T}$ is a strongly closed $\mathtt{T}(\lambda)$-tableau (conservative strongly closed $\mathtt{T}(\lambda)$-tableau). We will denote the sets of proof trees $\mathcal{PT}_{\mathcal{F}^{\text{tabl},\infty}}$ and $\mathcal{PT}_{\mathcal{F}^{\text{tabl},\infty,\text{cons}}}$ by SCT and $\mathsf{SCTCONS}$, respectively.

Theorem 3.3.43 (Soundness and Completeness of $\mathcal{F}^{\text{tabl},\infty}$). *The formal systems $\mathcal{F}^{tabl,\infty}$ and $\mathcal{F}^{tabl,\infty,cons}$ are sound and complete with respect to the collection of all unsatisfiable sets of signed formulas.*

Proof. The statement of the result follows from Corollary 3.3.39 because a strongly closed Δ-tableau is the same thing as an $\mathcal{F}^{\text{tabl},\infty}$-proof tree of Δ and a conservative strongly closed Δ-tableau is the same thing as an $\mathcal{F}^{\text{tabl},\infty,\text{cons}}$-proof tree of Δ. □

By the remark preceding Corollary 3.3.39, the formal systems $\mathcal{F}^{\text{tabl},\infty}$ and $\mathcal{F}^{\text{tabl},\infty,\text{cons}}$ remain complete (and, of course, sound) if in the definition of the rule R, we add one of the additional restrictions: $b\varphi \in \Delta$ or $b\varphi \notin \Delta$.

We now turn to tableaux for sets of unsigned formulas.

Definition 3.3.44. An *unsigned tableau* is a lot whose labels are subsets of PLFORM. □

The concepts introduced in Definition 3.3.1 through Definition 3.3.3 can be transferred to unsigned tableaux with obvious modifications. (Recall that a set of unsigned formulas Γ is closed if there is $\varphi \in$ PLFORM such that $\{\varphi, (\neg\varphi)\} \subseteq \Gamma$.)

Definition 3.3.45. Let Γ be a set of formulas. A Γ-*tableau* is an unsigned tableau $\mathtt{T}$ that satisfies the following conditions:

- The root of $\mathtt{T}$ is labeled by Γ, i.e., $\mathtt{T}(\lambda) = \Gamma$.
- If q is an interior node of $\mathtt{T}$, one of the following cases occurs:
  (1) There is some set of formulas Γ' and a formula φ with $\mathtt{d}(\varphi) = (K_0, \ldots, K_{n-1})$ such that $\mathtt{T}(q) = \Gamma' \cup \{\varphi\}$, q has n immediate descendants and $\mathtt{T}(qi) = \Gamma' \cup K_i$ for $0 \leq i \leq n-1$.
  (2) The node q has one immediate descendant $q0$ and $\mathtt{T}(q0) \subseteq \mathtt{T}(q)$.

□

The terminology for tableaux developed for signed formulas is carried over to tableaux for unsigned formulas.

Example 3.3.46. Consider the set of formulas

$$\Gamma = \{(p \to q), (\neg q), (\neg(\neg p))\}.$$

In Figure 3.7, we give a conservative strongly closed unsigned Γ-tableau. □

The notions of Γ-tableau with retention and with removal for a set of unsigned formulas Γ parallel the corresponding notions for Δ-tableau for a set of signed formulas Δ. Further, in Exercises 27–37, we present results for unsigned tableaux similar to the ones discussed in this section for signed tableaux. Some of these results can be obtained alternatively by applying the translation given in the following algorithm. This translation is obtained using the function $\mathtt{u}$ defined just before Exercise 96 of Section 2.12, namely, $\mathtt{u}(\mathbf{T}\varphi) = \varphi$

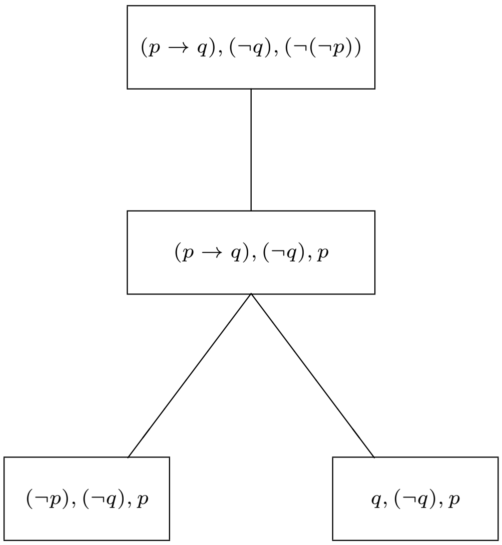


Fig. 3.7. Strongly closed unsigned Γ-tableau.

and $\mathtt{u}(\mathbf{F}\varphi) = (\neg\varphi)$ for every $\varphi \in$ PLFORM. It is not difficult to verify that if $b\varphi$ is not a signed variable and is not of the form $\mathbf{T}(\neg\psi)$ for some formula ψ, and $\mathtt{d}(b\varphi) = (K_0, \ldots, K_{n-1})$, then

$$\mathtt{d}(\mathtt{u}(b\varphi)) = (\mathtt{u}(K_0), \ldots, \mathtt{u}(K_{n-1})).$$

Construction 3.3.47.
Input: A signed tableau $\mathtt{T}$ that is a strongly closed Δ-tableau for some set Δ of signed formulas.
Output: A strongly closed unsigned $\mathtt{u}(\Delta)$-tableau.
Method: If $\mathtt{T}$ is a one-node tree, then output the one-node tree $\mathtt{T}'$ with $\mathtt{T}'(\lambda) = \mathtt{u}(\mathtt{T}(\lambda))$.
If $\mathtt{T}$ has more than one node, we need to consider two cases depending on whether thinning was used at the root.
If $\mathtt{T}(0)$ was obtained from $\mathtt{T}(\lambda)$ by thinning, apply the construction recursively to $\mathtt{T}_{[0]}$ to obtain a strongly closed $\mathtt{u}(\mathtt{T}(0))$-tableau $\mathtt{T}'$. Then, output $(\mathtt{T}'; \mathtt{u}(\mathtt{T}(\lambda))$.
If thinning was not used at the root, select a signed formula $b\varphi$ and a set of signed formulas Δ' such that $\mathtt{T}(\lambda) = \Delta' \cup \{b\varphi\}$, $\mathtt{d}(b\varphi) = (K_0, \ldots, K_{n-1})$, λ has n immediate descendents in $\mathtt{T}$, and $\mathtt{T}(i) = \Delta' \cup K_i$ for $0 \le i \le n-1$.

If $b\varphi = \mathbf{T}(\neg\psi)$ for some $\psi \in$ PLFORM, then apply the construction recursively to the subtree $\mathtt{T}_{[0]}$ and output the unsigned tableau resulting from this application.

If $b\varphi$ does not have the form mentioned above, apply the construction recursively to the subtrees $\mathtt{T}_{[i]}$, $0 \leq i \leq n-1$, to obtain the n unsigned tableaux $\mathtt{T}'_0, \ldots, \mathtt{T}'_{n-1}$ and output the unsigned tableau $(\mathtt{T}'_0, \ldots, \mathtt{T}'_{n-1}; \mathsf{u}(\Delta))$.

Proof of Correctness: By the third part of Theorem 3.3.10, recursive calls of the construction are applied to the right kind of tableaux. Also, by induction on the number of nodes of the input tableau, one can easily verify that the construction always terminates.

We prove now by induction on the number of nodes of the input tableau $\mathtt{T}$ that if $\mathtt{T}$ is a strongly closed Δ-tableau, then the output is a strongly closed $\mathsf{u}(\Delta)$-tableau. The basis step is easy and is left to the reader. For the inductive step, we distinguish two cases depending on whether thinning was used at the root.

In the first case, when thinning was used at the root, by inductive hypothesis, $\mathtt{T}'$ is a strongly closed $\mathsf{u}(\mathtt{T}(0))$-tableau, and because $\mathsf{u}(\mathtt{T}(0)) \subseteq \mathsf{u}(\mathtt{T}(\lambda))$, the output is a strongly closed $\mathsf{u}(\mathtt{T}(\lambda))$-tableau.

In the second case, when regular expansion is used, let $b\varphi, \Delta'$ be as in the construction. We distinguish two subcases.

In the first subcase, $b\varphi = \mathbf{T}(\neg\psi)$. Then, $\mathtt{T}_{[0]}$ is a strongly closed $(\Delta' \cup \{\mathbf{F}\psi\})$-tableau, so, by inductive hypothesis, the output of the construction applied to $\mathtt{T}_{[0]}$ is a strongly closed $(\mathsf{u}(\Delta' \cup \{\mathbf{F}\psi\}))$-tableau. Since $\mathsf{u}(\Delta' \cup \{\mathbf{F}\psi\}) = \mathsf{u}(\Delta') \cup \{(\neg\psi)\} = \mathsf{u}(\Delta' \cup \{\mathbf{T}(\neg\psi)\}) = \mathsf{u}(\Delta)$, this output is actually a strongly closed $\mathsf{u}(\Delta)$-tableau, as desired.

In the second subcase, $b\varphi$ does not have the form $\mathbf{T}(\neg\psi)$. By the inductive hypothesis, $\mathtt{T}'_i$ is a strongly closed $\mathsf{u}(\Delta' \cup K_i)$-tableau for $0 \leq i \leq n-1$. Since $\mathsf{u}(\Delta' \cup K_i) = \mathsf{u}(\Delta') \cup \mathsf{u}(K_i)$ and $\mathsf{d}(\mathsf{u}(b\varphi)) = (\mathsf{u}(K_0), \ldots, \mathsf{u}(K_{n-1}))$, as noted previously, the output tableau is a $(\mathsf{u}(\Delta') \cup \{\mathsf{u}(b\varphi)\})$-tableau. This is the desired tableau because

$$\mathsf{u}(\Delta) = \mathsf{u}(\Delta' \cup \{b\varphi\}).$$

Example 3.3.48. Let $\Delta = \{\mathbf{T}(p \to q), \mathbf{T}(\neg q), \mathbf{F}(\neg p)\}$. Application of the Construction 3.3.47 to the strongly closed Δ-tableau given in Figure 3.8 yields the strongly closed Γ-tableau given in Figure 3.7. □

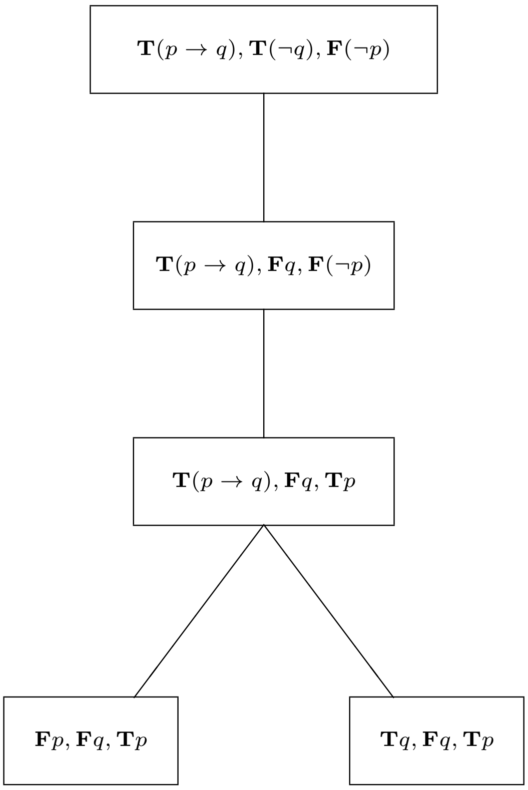

$\mathbf{T}(p \to q), \mathbf{T}(\neg q), \mathbf{F}(\neg p)$
$\mathbf{T}(p \to q), \mathbf{F}q, \mathbf{F}(\neg p)$
$\mathbf{T}(p \to q), \mathbf{F}q, \mathbf{T}p$
$\mathbf{F}p, \mathbf{F}q, \mathbf{T}p$
$\mathbf{T}q, \mathbf{F}q, \mathbf{T}p$

Fig. 3.8. Strongly closed Δ-tableau.

Supplement 26 contains the inverse translation construction from unsigned tableaux to signed tableaux.

3.4 The Cut Rule for Tableaux

In this section, we introduce a nonanalytical version of tableaux. The purpose of this extension of tableaux is to generate smaller closed tableaux for unsatisfiable sets of signed formulas.

Definition 3.4.1. Let Δ be a set of signed formulas. A Δ-*tableau with cut* is a tableau $\mathtt{T}$ that satisfies the following conditions:

(1) The root of $\mathtt{T}$ is labeled by Δ, i.e., $\mathtt{T}(\lambda) = \Delta$.
(2) If q is an interior node of $\mathtt{T}$, then one of the following holds:

(a) There is some signed formula $b\varphi$ with $\mathtt{d}(b\varphi) = (K_0, \ldots, K_{n-1})$ and a set of formulas Δ' such that $\mathtt{T}(q) = \Delta' \cup \{b\varphi\}$, q has n immediate descendants, and $\mathtt{T}(qi) = \Delta' \cup K_i$ for $0 \leq i \leq n-1$.

(b) The node q has one immediate descendant $q0$ and $\mathtt{T}(q0) \subseteq \mathtt{T}(q)$.

(c) There is a formula φ and a set of signed formulas Δ' such that q has two immediate descendants, $\mathtt{T}(q) = \Delta'$, $\mathtt{T}(q0) = \Delta' \cup \{\mathbf{T}\varphi\}$, and $\mathtt{T}(q1) = \Delta' \cup \{\mathbf{F}\varphi\}$. ▯

If regular expansion or the cut rule was used at the node q, we say that $\mathtt{T}$ is *locally conservative* at q. If $\mathtt{T}$ is locally conservative at all its interior nodes, then we say that $\mathtt{T}$ is *conservative.*

Because of Part (2c), Definition 3.4.1 is even "more" nonuniquely readable than Definition 3.3.4. Indeed, consider the Δ-tableau with cut given in Figure 3.9, where $\Delta = \{\mathbf{T}(p_0 \vee p_0), \mathbf{T}p_1, \mathbf{F}p_0\}$. Note that for $q = \lambda$, both Parts 2a and 2c of the definition of Δ-tableau with cut hold.

Whenever Part (2c) of Definition 3.4.1 holds for a node q, we say that the *cut rule* was applied at q. By the previous comment, the fact that the cut rule was applied at q does not rule out that Part (2a) also holds for q.

Note that Part (2c) makes tableaux with cut nonanalytical since the formula φ may be unrelated to the set Δ.

An application showing the usefulness of the cut rule is given in the following example.

Example 3.4.2. Consider the following statement:

> If there is a strongly closed $\{\mathbf{F}\varphi\}$-tableau $\mathtt{T}_0$ and a strongly closed $\{\mathbf{F}(\varphi \rightarrow \psi)\}$-tableau $\mathtt{T}_1$, then there is a strongly closed $\{\mathbf{F}\psi\}$-tableau $\mathtt{T}$.

A semantic argument for the existence of $\mathtt{T}$ can be made immediately. Indeed, the existence of $\mathtt{T}_0$ and $\mathtt{T}_1$ implies that φ and $(\varphi \rightarrow \psi)$ are tautologies. So, ψ is a tautology, which implies that $\{\mathbf{F}\psi\}$ is unsatisfiable and thus there is a strongly closed $\{\mathbf{F}\psi\}$-tableau $\mathtt{T}$. This argument, however, does not give us any way of constructing $\mathtt{T}$ based on the syntactic structure of $\mathtt{T}_0$ and $\mathtt{T}_1$. However, we can give

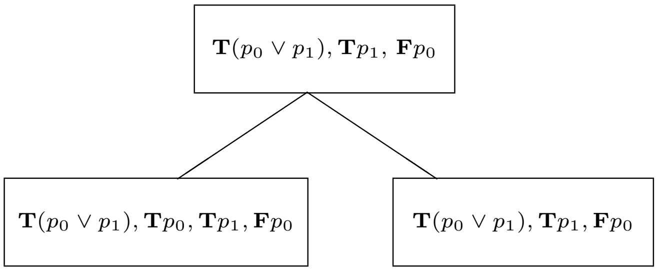

$\mathbf{T}(p_0 \vee p_1), \mathbf{T}p_1, \mathbf{F}p_0$
$\mathbf{T}(p_0 \vee p_1), \mathbf{T}p_0, \mathbf{T}p_1, \mathbf{F}p_0$
$\mathbf{T}(p_0 \vee p_1), \mathbf{T}p_1, \mathbf{F}p_0$

Fig. 3.9. Δ-tableau with cut.

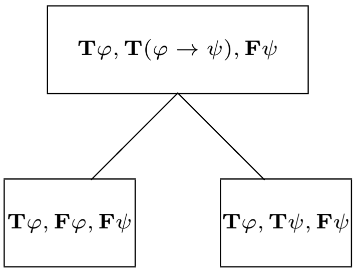

$\mathbf{T}\varphi, \mathbf{T}(\varphi \to \psi), \mathbf{F}\psi$
$\mathbf{T}\varphi, \mathbf{F}\varphi, \mathbf{F}\psi$
$\mathbf{T}\varphi, \mathbf{T}\psi, \mathbf{F}\psi$

Fig. 3.10. The tableau $\mathtt{T}_1''$.

a syntactic construction of a strongly closed $\{\mathbf{F}\psi\}$-tableau with cut $\mathtt{T}$ as follows:

(1) Define the tableau $\mathtt{T}_0' = \mathtt{T}_0 \uplus \{\mathbf{F}\psi\}$. Then, $\mathtt{T}_0'$ is a strongly closed $\{\mathbf{F}\varphi, \mathbf{F}\psi\}$-tableau.
(2) Define the tableau $\mathtt{T}_1' = \mathtt{T}_1 \uplus \{\mathbf{T}\varphi, \mathbf{F}\psi\}$. Then, $\mathtt{T}_1'$ is a strongly closed $\{\mathbf{T}\varphi, \mathbf{F}\psi, \mathbf{F}(\varphi \to \psi)\}$-tableau.
(3) Let $\mathtt{T}_1''$ be the strongly closed $\{\mathbf{T}\varphi, \mathbf{T}(\varphi \to \psi), \mathbf{F}\psi\}$-tableau shown in Figure 3.10.
(4) By applying the cut rule to $\mathtt{T}_1'$ and $\mathtt{T}_1''$, we obtain the strongly closed $\{\mathbf{T}\varphi, \mathbf{F}\psi\}$-tableau with cut $\mathtt{T}_1''' = (\mathtt{T}_1'', \mathtt{T}_1'; \{\mathbf{T}\varphi, \mathbf{F}\psi\})$.
(5) Another application of the cut rule to $\mathtt{T}_1'''$ and $\mathtt{T}_0'$ yields the strongly closed $\{\mathbf{F}\psi\}$-tableau with cut $\mathtt{T} = (\mathtt{T}_1''', \mathtt{T}_0'; \{\mathbf{F}\psi\})$ as shown in Figure 3.11.

Similarly, if we have a strongly closed $(\Delta \cup \{\mathbf{F}\varphi\})$-tableau $\mathtt{T}_0$ and a strongly closed $(\Delta \cup \{\mathbf{F}(\varphi \to \psi)\})$-tableau $\mathtt{T}_1$, then we can syntactically produce a strongly closed $(\Delta \cup \{\mathbf{F}\psi\})$-tableau with cut $\mathtt{T}$. ∎

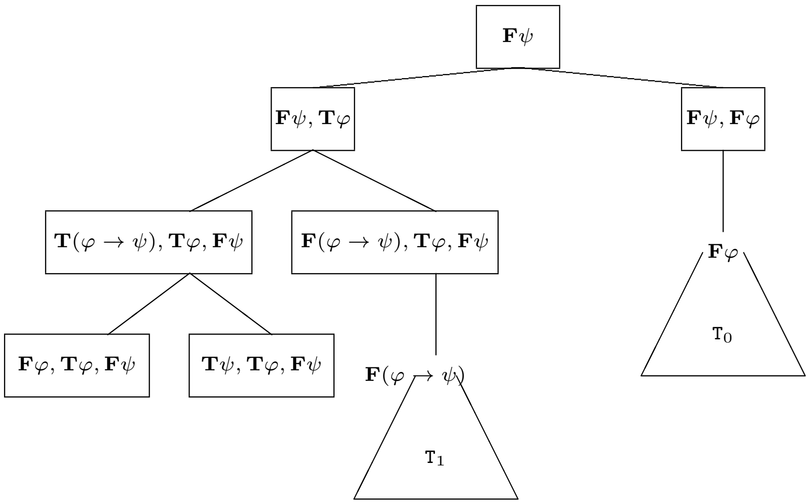

$\mathbf{F}\psi$
$\mathbf{F}\psi, \mathbf{T}\varphi$
$\mathbf{F}\psi, \mathbf{F}\varphi$
$\mathbf{T}(\varphi \to \psi), \mathbf{T}\varphi, \mathbf{F}\psi$
$\mathbf{F}(\varphi \to \psi), \mathbf{T}\varphi, \mathbf{F}\psi$
$\mathbf{F}\varphi$
$\mathtt{T}_0$
$\mathbf{F}\varphi, \mathbf{T}\varphi, \mathbf{F}\psi$
$\mathbf{T}\psi, \mathbf{T}\varphi, \mathbf{F}\psi$
$\mathbf{F}(\varphi \to \psi)$
$\mathtt{T}_1$

Fig. 3.11. Strongly closed $\{\mathbf{F}\psi\}$-tableau with cut (using strongly closed $\{\mathbf{F}\varphi\}$- and $\{\mathbf{F}(\varphi \to \psi)\}$-tableaux).

Lemma 3.4.3. *Let Δ, Δ' be two sets of signed formulas. If $\mathtt{T}$ is a Δ-tableau with cut, then the tableau $\mathtt{T} \uplus \Delta'$ is a $(\Delta \cup \Delta')$-tableau with cut.*

Proof. This proof is left to the reader. □

Lemma 3.4.4. *If $\mathtt{T}$ is a (conservative) Δ-tableau with cut such that the root has $n > 0$ immediate descendants, then $\mathtt{T}_{[i]}$ is a (conservative) $\mathtt{T}(i)$-tableau with cut for $0 \leq i \leq n-1$.*

Proof. The argument is a direct application of Definition 3.4.1. □

Theorem 3.4.5. *Let Δ be a set of signed formulas and $\mathtt{T}$ be a Δ-tableau with cut. If v is a truth assignment, then v satisfies Δ if and only if it satisfies $\mathtt{T}(\mathtt{B})$ for some branch $\mathtt{B}$ of $\mathtt{T}$.*

Proof. The proof is similar to the one used in Theorem 3.3.16. The only difference occurs when the cut rule is applied at the node q_k. In this case, q_k has two immediate descendants q_k0 and q_k1, $\mathtt{T}(q_k0) = \mathtt{T}(q_k) \cup \{\mathbf{T}\varphi\}$ and $\mathtt{T}(q_k1) = \mathtt{T}(q_k) \cup \{\mathbf{F}\varphi\}$ for some formula φ. Since v satisfies $\mathtt{T}(q_k)$, it satisfies exactly one of $\mathtt{T}(q_k0), \mathtt{T}(q_k1)$ and we let q_{k+1} be the immediate descendant of q_k such that v satisfies $\mathtt{T}(q_{k+1})$. □

Theorem 3.4.6. *Let* $\mathtt{T}$ *be a completed* Δ*-tableau with cut. A truth assignment* v *satisfies* Δ *if and only if there is a branch* $\mathtt{B}$ *of* $\mathtt{T}$ *such that* $\mathtt{T}(\mathtt{B})$ *is a Hintikka set and* $v(p) = b$ *for all* $bp \in \mathtt{T}(\mathtt{B})$*.*

Proof. This follows immediately from Theorems 2.7.28 and 3.4.5. □

Corollary 3.4.7 (Soundness and Strong Completeness for Tableaux with Cut). *A set of signed formulas* Δ *is unsatisfiable if and only if there exists a strongly closed* Δ*-tableau with cut.*

Proof. The existence of a strongly closed Δ-tableau with cut for an unsatisfiable set Δ follows from Theorem 3.3.38. Conversely, if there is a strongly closed Δ-tableau with cut, then Theorem 3.4.6 implies that Δ is unsatisfiable. □

The following theorem corresponds to Theorem 3.3.40 for tableaux without cut.

Theorem 3.4.8. *There is an effective, syntactic construction that begins with a strongly closed* Δ*-tableau* $\mathtt{T}$ *with cut and produces a strongly closed* Δ'*-tableau* $\mathtt{T}'$ *with cut, where* Δ' *is a finite subset of* Δ*.*

Proof. The argument is similar to the one used in Theorem 3.3.40 with the exception of the case when the cut rule is used at the root of $\mathtt{T}$. If this is the case, then $\mathtt{T}(0) = \Delta \cup \{\mathbf{T}\varphi\}$ and $\mathtt{T}(1) = \Delta \cup \{\mathbf{F}\varphi\}$ for some formula φ. By inductive hypothesis, we obtain the strongly closed Δ'_0-tableau with cut $\mathtt{T}'_0$ and the strongly closed Δ'_1-tableau with cut $\mathtt{T}'_1$, where Δ'_0 and Δ'_1 are finite subsets of $\Delta \cup \{\mathbf{T}\varphi\}$ and $\Delta \cup \{\mathbf{F}\varphi\}$, respectively. Let $\Delta''_0 = \Delta'_0 \cap \Delta$ and $\Delta''_1 = \Delta'_1 \cap \Delta$. By thinning, we construct the tableaux $\mathtt{T}''_0 = \mathtt{T}'_0 \uplus (\Delta''_0 \cup \Delta''_1 \cup \{\mathbf{T}\varphi\})$ and $\mathtt{T}''_1 = \mathtt{T}'_1 \uplus (\Delta''_0 \cup \Delta''_1 \cup \{\mathbf{F}\varphi\})$. By applying the cut rule, we return the strongly closed tableau $(\mathtt{T}''_0, \mathtt{T}''_1; \Delta''_0 \cup \Delta''_1)$. □

Definition 3.4.9. The formal system $\mathcal{F}^{\text{tabl,cons,cut}}$ is the formal system obtained from $\mathcal{F}^{\text{tabl,cons}}$ by adding the following "cut" rule:

$$\frac{\Delta \cup \{\mathbf{T}\varphi\}, \Delta \cup \{\mathbf{F}\varphi\}}{\Delta}$$

for all finite sets of signed formulas Δ and formulas φ.

If the thinning rule is added to $\mathcal{F}^{\text{tabl,cons,cut}}$, we obtain the formal system $\mathcal{F}^{\text{tabl,cut}}$.

The formal system $\mathcal{F}^{\text{tabl},\infty,\text{cons},\text{cut}}$ is obtained from $\mathcal{F}^{\text{tabl},\infty,\text{cons}}$ by adding the previous cut rule for arbitrary sets Δ, rather than for finite sets. Finally, the formal system $\mathcal{F}^{\text{tabl},\infty,\text{cut}}$ is obtained from $\mathcal{F}^{\text{tabl},\infty,\text{cons}}$ by adding the cut rule. ▯

Theorem 3.4.10. (Soundness and Completeness of $\mathcal{F}^{\text{tabl},\text{cut}}$ and $\mathcal{F}^{\text{tabl},\infty,\text{cut}}$). *The formal system $\mathcal{F}^{tabl,cut}$ ($\mathcal{F}^{tabl,\infty,cut}$) is sound and complete for the collection of finite unsatisfiable sets of signed formulas (collection of unsatisfiable sets of signed formulas).*

Proof. It is easy to see that the set of proof trees of $\mathcal{F}^{\text{tabl},\text{cut}}$ is the set of all tableaux $\mathtt{T}$ such that $\mathtt{T}$ is a strongly closed $\mathtt{T}(\lambda)$-tableau with cut and $\mathtt{T}(\lambda)$ is finite. Therefore, $\mathcal{F}^{\text{tabl},\text{cut}}$ is sound and complete for the collection of finite, unsatisfiable sets of signed formulas. Similarly, the set of proof trees of $\mathcal{F}^{\text{tabl},\infty,\text{cut}}$ is the set of all tableaux $\mathtt{T}$ such that $\mathtt{T}$ is a strongly closed $\mathtt{T}(\lambda)$-tableau with cut, so $\mathcal{F}^{\text{tabl},\infty,\text{cut}}$ is sound and complete for the collection of unsatisfiable sets of signed formulas. □

Soundness and completeness continue to hold for the formal systems $\mathcal{F}^{\text{tabl},\text{cons},\text{cut}}$ and $\mathcal{F}^{\text{tabl},\infty,\text{cons},\text{cut}}$.

We shall denote by SCTCUT the set of tableaux which are strongly closed Δ-tableaux with cut for some set Δ.

Because of the completeness of the formal systems $\mathcal{F}^{\text{tabl}}$ and $\mathcal{F}^{\text{tabl},\infty}$, the cut rule is superfluous in proving unsatisfiability of a set of signed formulas. Its use however can reduce the size of such proofs. Given a strongly closed Δ-tableau with cut $\mathtt{T}$, we know that there exists a strongly closed Δ-tableau $\mathtt{T}'$, which does not use the cut rule and, hence, is analytic. We could find the Δ-tableau $\mathtt{T}'$ using Construction 3.3.33 (or Algorithm 3.3.20 if Δ is finite), but this would not use $\mathtt{T}$. We will describe a syntactic process, called "cut elimination," for generating the tableau $\mathtt{T}'$ starting from $\mathtt{T}$. This process becomes an algorithm when the sets of formulas involved are finite.

We begin with a construction that computes a function cet (short for "cut elimination for tableaux") that, given strongly closed tableaux $\mathtt{T}_0, \mathtt{T}_1$, two sets Δ_0, Δ_1 of signed formulas and a formula φ such that $\mathtt{T}_0$ is a $(\Delta_0 \cup \{\mathbf{T}\varphi\})$-tableau and $\mathtt{T}_1$ is a $(\Delta_1 \cup \{\mathbf{F}\varphi\})$-tableau, produces a strongly closed $\Delta_0 \cup \Delta_1$-tableau. Actually, this is a slightly more general result than the one we need, but it has an easier proof. When Δ_0 and Δ_1 are finite, the construction becomes effective and can be regarded as being an algorithm for computing the function cet defined in the following.

Construction 3.4.11.
Input: Two strongly closed tableaux $\mathtt{T}_0, \mathtt{T}_1$, two sets Δ_0, Δ_1 of signed formulas and a formula φ such that $\mathtt{T}_0$ is a $(\Delta_0 \cup \{\mathbf{T}\varphi\})$-tableau and $\mathtt{T}_1$ is a $(\Delta_1 \cup \{\mathbf{F}\varphi\})$-tableau.
Output: A strongly closed $(\Delta_0 \cup \Delta_1)$-tableau $\mathsf{cet}(\mathtt{T}_0, \mathtt{T}_1, \Delta_0, \Delta_1, \varphi)$.
Method: Proceed according to which of the following cases holds:

`Case 1:` Either $\mathbf{T}\varphi \in \Delta_0$ or $\mathbf{F}\varphi \in \Delta_1$.

- `Case 1.1:` $\mathbf{T}\varphi \in \Delta_0$. Then, return the tableau $\mathtt{T}_0 \uplus \Delta_1$.
- `Case 1.2:` $\mathbf{F}\varphi \in \Delta_1$. Then, return the tableau $\mathtt{T}_1 \uplus \Delta_0$.

`Case 2:` Either $\Delta_0 \cup \{\mathbf{T}\varphi\}$ or $\Delta_1 \cup \{\mathbf{F}\varphi\}$ is closed and neither $\mathbf{T}\varphi \in \Delta_0$ nor $\mathbf{F}\varphi \in \Delta_1$.

- `Case 2.1:` $\Delta_0 \cup \Delta_1$ is closed. Then, return the one-node tableau labeled by $\Delta_0 \cup \Delta_1$.
- `Case 2.2:` $\Delta_0 \cup \Delta_1$ is not closed, but $\Delta_0 \cup \{\mathbf{T}\varphi\}$ is closed. Then, return $\mathtt{T}_1 \uplus \Delta_0$.
- `Case 2.3:` $\Delta_0 \cup \Delta_1$ is not closed, but $\Delta_1 \cup \{\mathbf{F}\varphi\}$ is closed. Then, return $\mathtt{T}_0 \uplus \Delta_1$.

`Case 3:` Neither $\Delta_0 \cup \{\mathbf{T}\varphi\}$ nor $\Delta_1 \cup \{\mathbf{F}\varphi\}$ is closed and neither $\mathbf{T}\varphi \in \Delta_0$ nor $\mathbf{F}\varphi \in \Delta_1$. Then, both $\mathtt{T}_0$ and $\mathtt{T}_1$ have more than one node.

- `Case 3.1:` Thinning is used at the root of $\mathtt{T}_0$ or $\mathtt{T}_1$.
  - `Case 3.1.1:` Thinning is used at the root of $\mathtt{T}_0$. Let $\Delta_0' = \mathtt{T}_0(0)$. We distinguish two subcases depending on whether $\mathbf{T}\varphi \in \Delta_0'$ or not. If $\mathbf{T}\varphi \notin \Delta_0'$, then return $(\mathtt{T}_0)_{[0]} \uplus (\Delta_0 \cup \Delta_1)$. If $\mathbf{T}\varphi \in \Delta_0'$, then let $\mathtt{T}_0' = \mathsf{cet}((\mathtt{T}_0)_{[0]}, \mathtt{T}_1, \Delta_0' - \{\mathbf{T}\varphi\}, \Delta_1, \varphi)$ and return $\mathtt{T}_0' \uplus (\Delta_0 \cup \Delta_1)$.
  - `Case 3.1.2:` Thinning is used at the root of $\mathtt{T}_1$. Proceed as in Case 2.1.1 reversing the roles of 0 and 1.
- `Case 3.2:` Thinning was used neither at the root of $\mathtt{T}_0$ nor at the root of $\mathtt{T}_1$. Then, there are

Δ_0', Δ_1', $b_0\psi_0 \in \Delta_0 \cup \{\mathbf{T}\varphi\}$ and $b_1\psi_1 \in \Delta_1 \cup \{\mathbf{F}\varphi\}$ such that $\Delta_0 \cup \{\mathbf{T}\varphi\} = \Delta_0' \cup \{b_0\psi_0\}$, $\Delta_1 \cup \{\mathbf{F}\varphi\} = \Delta_1' \cup \{b_1\psi_1\}$, $\mathtt{d}(b_0\psi_0) = (K_0, \ldots, K_{n-1})$, $\mathtt{d}(b_1\psi_1) = (H_0, \ldots, H_{m-1})$, $\mathtt{T}_0(i) = \Delta_0' \cup K_i$ for $0 \le i \le n-1$, and $\mathtt{T}_1(j) = \Delta_1' \cup H_j$ for $0 \le j \le m-1$. (There could be several such choices of $b_0\psi_0, b_1\psi_1$, any of which would work. To be definite, we could choose the first ones in the standard ordering of the signed formulas.)

`Case 3.2.1:` $b_0\psi_0 \in \Delta_0$, so $b_0\psi_0 \neq \mathbf{T}\varphi$ because $\mathbf{T}\varphi \notin \Delta_0$. Then, let $\mathtt{V}_i = \mathsf{cet}((\mathtt{T}_0)_{[i]}, \mathtt{T}_1, (\Delta_0' - \{\mathbf{T}\varphi\}) \cup K_i, \Delta_1, \varphi)$ for $0 \le i \le n-1$ and return $(\mathtt{V}_0, \ldots, \mathtt{V}_{n-1}; \Delta_0 \cup \Delta_1)$.

`Case 3.2.2:` $b_1\psi_1 \in \Delta_1$ and $b_0\psi_0 \notin \Delta_0$. Then, let $\mathtt{W}_j = \mathsf{cet}(\mathtt{T}_0, (\mathtt{T}_1)_{[j]}, \Delta_0, \Delta_1' - \{\mathbf{F}\varphi\} \cup H_j, \varphi)$ for $0 \le j \le m-1$ and return $(\mathtt{W}_0, \ldots, \mathtt{W}_{m-1}; \Delta_0 \cup \Delta_1)$.

`Case 3.2.3:` Neither $b_0\psi_0$ is in Δ_0 nor $b_1\psi_1$ is in Δ_1, so $b_0\psi_0 = \mathbf{T}\varphi$ and $b_1\psi_1 = \mathbf{F}\varphi$.

`Case 3.2.3.1:` $\varphi = (\neg\alpha)$. Then, $(\mathtt{T}_0)_{[0]}$ is a strongly closed $\Delta_0' \cup \{\mathbf{F}\alpha\}$-tableau and $(\mathtt{T}_1)_{[0]}$ is a strongly closed $\Delta_1' \cup \{\mathbf{T}\alpha\}$-tableau. Then, we are in one of the following four subcases:

`Case 3.2.3.1.1:` $\mathbf{T}(\neg\alpha) \notin \Delta_0'$ and $\mathbf{F}(\neg\alpha) \notin \Delta_1'$, so $\Delta_0' = \Delta_0$ and $\Delta_1' = \Delta_1$. Return $\mathsf{cet}((\mathtt{T}_1)_{[0]}, (\mathtt{T}_0)_{[0]}, \Delta_1', \Delta_0', \alpha)$.

`Case 3.2.3.1.2:` $\mathbf{T}(\neg\alpha) \in \Delta_0'$ and $\mathbf{F}(\neg\alpha) \notin \Delta_1'$, so $\Delta_1' = \Delta_1$. Let $\mathtt{V}_0 = \mathsf{cet}((\mathtt{T}_0)_{[0]}, \mathtt{T}_1, (\Delta_0' - \{\mathbf{T}(\neg\alpha)\}) \cup \{\mathbf{F}\alpha\}, \Delta_1, (\neg\alpha))$. Return $\mathsf{cet}((\mathtt{T}_1)_{[0]}, \mathtt{V}_0, \Delta_1', (\Delta_0' - \{\mathbf{T}(\neg\alpha)\}) \cup \Delta_1, \alpha)$.

`Case 3.2.3.1.3:` $\mathbf{T}(\neg\alpha) \notin \Delta_0'$ and $\mathbf{F}(\neg\alpha) \in \Delta_1'$, so $\Delta_0' = \Delta_0$. Let $\mathtt{W}_0 = \mathsf{cet}(\mathtt{T}_0, (\mathtt{T}_1)_{[0]}, \Delta_0, (\Delta_1' - \{\mathbf{F}(\neg\alpha)\}) \cup \{\mathbf{T}\alpha\}, (\neg\alpha))$. Return $\mathsf{cet}(\mathtt{W}_0, (\mathtt{T}_0)_{[0]}, (\Delta_1' - \{\mathbf{F}(\neg\alpha)\}) \cup \Delta_0, \Delta_0', \alpha)$.

`Case 3.2.3.1.4:` $\mathbf{T}(\neg\alpha) \in \Delta_0'$ and $\mathbf{F}(\neg\alpha) \in \Delta_1'$. Let $\mathtt{V}_0$ and $\mathtt{W}_0$ be as in the previous two subcases and return $\mathsf{cet}(\mathtt{W}_0, \mathtt{V}_0, \Delta_0 \cup (\Delta_1' - \{\mathbf{F}(\neg\alpha)\}), (\Delta_0' - \{\mathbf{T}(\neg\alpha)\}) \cup \Delta_1, \alpha)$.

`Case 3.2.3.2:` $\varphi = (\alpha C \beta)$ for some binary connective symbol C. For each $b, b' \in \mathbf{Bool}$, there is a constituent K_i or a constituent H_j contained in the set $G_{bb'} = \{b\alpha, b'\beta\}$, so define $k_{bb'}$ to be the least i such that $K_i \subseteq G_{bb'}$ if there exists such an i with $0 \leq i \leq n-1$. Otherwise, let $k_{bb'}$ be the least j, $0 \leq j \leq m-1$, such that $H_j \subseteq G_{bb'}$.

`Case 3.2.3.2.1:` $\mathbf{T}(\alpha C \beta) \notin \Delta_0'$ and $\mathbf{F}(\alpha C \beta) \notin \Delta_1'$, which implies $\Delta_0 = \Delta_0'$ and $\Delta_1 = \Delta_1'$. Let $\mathtt{V}_i = (\mathtt{T}_0)_{[i]} \uplus \Delta_1$ and $\mathtt{W}_j = (\mathtt{T}_1)_{[j]} \uplus \Delta_0$ for $0 \leq i \leq n-1$ and $0 \leq j \leq m-1$. Define $\mathtt{T}_{bb'}$ to be $\mathtt{V}_{k_{bb'}} \uplus G_{bb'}$ if there is an i such that $K_i \subseteq G_{bb'}$ and to be $\mathtt{W}_{k_{bb'}} \uplus G_{bb'}$ otherwise. Let $\mathtt{T}_{\mathbf{T}} = \mathsf{cet}(\mathtt{T}_{\mathbf{TT}}, \mathtt{T}_{\mathbf{TF}}, \Delta_0 \cup \Delta_1 \cup \{\mathbf{T}\alpha\}, \Delta_0 \cup \Delta_1 \cup \{\mathbf{T}\alpha\}, \beta)$ and $\mathtt{T}_{\mathbf{F}} = \mathsf{cet}(\mathtt{T}_{\mathbf{FT}}, \mathtt{T}_{\mathbf{FF}}, \Delta_0 \cup \Delta_1 \cup \{\mathbf{F}\alpha\}, \Delta_0 \cup \Delta_1 \cup \{\mathbf{F}\alpha\}, \beta)$. Return $\mathtt{T}' = \mathsf{cet}(\mathtt{T}_{\mathbf{T}}, \mathtt{T}_{\mathbf{F}}, \Delta_0 \cup \Delta_1, \Delta_0 \cup \Delta_1, \alpha)$.

`Case 3.2.3.2.2:` $\mathbf{T}(\alpha C \beta) \in \Delta_0'$ and $\mathbf{F}(\alpha C \beta) \notin \Delta_1'$, which implies $\Delta_0 = \Delta_0' - \{\mathbf{T}(\alpha C \beta)\}$ and $\Delta_1 = \Delta_1'$. For this case, we define $\mathtt{V}_i = \mathsf{cet}((\mathtt{T}_0)_{[i]}, \mathtt{T}_1, (\Delta_0' - \{\mathbf{T}(\varphi C \beta)\}) \cup K_i, \Delta_1, (\alpha C \beta))$ and we take $\mathtt{W}_j$ as in the previous case. Return $\mathtt{T}'$ obtained as in the previous case.

`Case 3.2.3.2.3:` $\mathbf{T}(\alpha C \beta) \notin \Delta_0'$ and $\mathbf{F}(\alpha C \beta) \in \Delta_1'$, which implies $\Delta_0 = \Delta_0'$ and $\Delta_1 = \Delta_1' - \{\mathbf{F}(\alpha C \beta)\}$. For this case, $\mathtt{V}_i$ is obtained as in Case 3.2.3.2.1 and we define $\mathtt{W}_j = \mathsf{cet}(\mathtt{T}_0, (\mathtt{T}_1)_{[j]}, \Delta_0, (\Delta_1 - \{\mathbf{F}(\alpha C \beta)\}) \cup H_j, (\alpha C \beta))$. Return $\mathtt{T}'$ obtained as in Case 3.2.3.2.1.

`Case 3.2.3.2.4:` $\mathbf{T}(\alpha C \beta) \notin \Delta_0'$ and $\mathbf{F}(\alpha C \beta) \notin \Delta_1'$, which implies $\Delta_0 = \Delta_0' - \{\mathbf{T}(\alpha C \beta)\}$ and $\Delta_1 = \Delta_1' - \{\mathbf{F}(\alpha C \beta)\}$. For this case, we define $\mathtt{V}_i$ as in Case 3.2.3.2.2 and $\mathtt{W}_j$ as in Case 3.2.3.2.3. Return $\mathtt{T}'$ obtained as in Case 3.2.3.2.1.

Proof of Correctness: We show by course-of-values induction on the length of φ that if the construction is applied to a strongly closed $(\Delta_0 \cup \{\mathbf{T}\varphi\})$-tableau $\mathtt{T}_0$ and a strongly closed $(\Delta_1 \cup \{\mathbf{F}\varphi\})$-tableau $\mathtt{T}_1$, then it halts and produces the desired strongly closed $\Delta_0 \cup \Delta_1$-tableau.

Suppose that the result is true for formulas shorter than φ. We now show the result for φ by course-of-values induction on $n = |\mathtt{T}_0| + |\mathtt{T}_1|$. The "inner" inductive hypothesis means that for all $\Delta'_0, \Delta'_1, \mathtt{T}'_0, \mathtt{T}'_1$, if $\mathtt{T}'_0$ is a strongly closed $(\Delta'_0 \cup \{\mathbf{T}\varphi\})$-tableau, $\mathtt{T}'_1$ is a strongly closed $(\Delta'_1 \cup \{\mathbf{F}\varphi\})$-tableau, and $|\mathtt{T}'_0| + |\mathtt{T}'_1| < n$, then the construction halts and produces a strongly closed $\Delta'_0 \cup \Delta'_1$-tableau. We examine each case in turn.

In Cases 1 and 2.1, there is nothing to prove. In Case 2.2, $\mathbf{F}\varphi \in \Delta_0$ because $\Delta_0 \cup \{\mathbf{T}\varphi\}$ is closed and Δ_0 is not because $\Delta_0 \cup \Delta_1$ is not closed, so $\mathtt{T}_1 \uplus \Delta_0$ is a strongly closed $(\Delta_0 \cup \Delta_1 \cup \{\mathbf{F}\varphi\})$-tableau, that is, a strongly closed $(\Delta_0 \cup \Delta_1)$-tableau. Case 2.3 is similar to Case 2.2.

In the first subcase of Case 3.1.1, $\Delta'_0 \subseteq \Delta_0$, so $(\mathtt{T}_0)_{[0]} \uplus (\Delta_0 \cup \Delta_1)$ is a strongly closed $(\Delta_0 \cup \Delta_1)$-tableau with thinning at the root. In the second subcase of Case 3.1.1, since $|(\mathtt{T}_0)_{[0]}| + |\mathtt{T}_1| < |\mathtt{T}_0| + |\mathtt{T}_1|$, by the inductive hypothesis, $\mathtt{T}'_0 = \mathsf{cet}((\mathtt{T}_0)_{[0]}, \mathtt{T}_1, \Delta'_0 - \{\mathbf{T}\varphi\}, \Delta_1, \varphi)$, is a strongly closed $(\Delta'_0 - \{\mathbf{T}\varphi\}) \cup \Delta_1$-tableau. Consequently, $\mathtt{T}'_0 \uplus (\Delta_0 \cup \Delta_1)$ is a strongly closed $(\Delta_0 \cup \Delta_1)$-tableau. Case 3.1.2 is treated similarly.

In Case 3.2.1, since $|(\mathtt{T}_0)_{[i]}| + |\mathtt{T}_1| < |\mathtt{T}_0| + |\mathtt{T}_1|$, by inductive hypothesis, $\mathtt{V}_i$ is a strongly closed $(\Delta'_0 - \{\mathbf{T}\varphi\}) \cup K_i \cup \Delta_1$-tableau. Observe that, under the assumptions of this case, we have $\Delta_0 = (\Delta'_0 - \{\mathbf{T}\varphi\}) \cup \{b_0\psi_0\}$. Therefore, $\Delta_0 \cup \Delta_1 = (\Delta'_0 - \{\mathbf{T}\varphi\}) \cup \{b_0\psi_0\} \cup \Delta_1$, and consequently, $(\mathtt{V}_0, \ldots, \mathtt{V}_{n-1}; \Delta_0 \cup \Delta_1)$ is a strongly closed $(\Delta_0 \cup \Delta_1)$-tableau. Case 3.2.2 is entirely similar.

In Case 3.2.3.1.1, since $|\alpha| < |\varphi|$, by the inductive hypothesis, the returned tableau $\mathsf{cet}((\mathtt{T}_1)_{[0]}, (\mathtt{T}_0)_{[0]}, \Delta'_1, \Delta'_0, \alpha)$ is strongly closed $(\Delta_0 \cup \Delta_1)$-tableau.

In Case 3.2.3.1.2, by the inductive hypothesis, $\mathtt{V}_0$ is a strongly closed $(\Delta_0 \cup \Delta_1 \cup \{\mathbf{F}\alpha\})$-tableau because $|(\mathtt{T}_0)_{[0]}| + |\mathtt{T}_1| < |\mathtt{T}_0| + |\mathtt{T}_1|$ and $\Delta'_0 - \{\mathbf{T}\varphi\} = \Delta_0$. A new application of the inductive hypothesis allowed by the fact that $|\alpha| < |\varphi|$ implies that $\mathsf{cet}((\mathtt{T}_1)_{[0]}, \mathtt{V}_0, \Delta'_1, (\Delta'_0 - \{\mathbf{T}(\neg\alpha)\}) \cup \Delta_1, \alpha)$ is a strongly closed $(\Delta_0 \cup \Delta_1)$-tableaux.

Case 3.2.3.1.3 is handled similar to Case 3.2.3.1.2. The last of this sequence of cases, Case 3.2.3.1.4, can be dealt with by combining the arguments used in the previous two cases.

In Case 3.2.3.2.1, note that $\mathtt{V}_i$ is a strongly closed $(\Delta_0 \cup \Delta_1 \cup K_i)$-tableau and $\mathtt{W}_j$ is a strongly closed $(\Delta_0 \cup \Delta_1 \cup H_j)$-tableau because $\Delta'_0 = \Delta_0$ and $\Delta'_1 = \Delta_1$. Therefore, $\mathtt{T}_{bb'}$ is a strongly closed $(\Delta_0 \cup \Delta_1 \cup G_{bb'})$-tableau for every $b, b' \in \mathbf{Bool}$. Since $|\beta| < |\varphi|$, by the inductive hypothesis, it follows that $\mathtt{T}_{\mathbf{T}}$ is a strongly closed $(\Delta_0 \cup \Delta_1 \cup \{\mathbf{T}\alpha\})$-tableau and $\mathtt{T}_{\mathbf{F}}$ is a strongly closed $(\Delta_0 \cup \Delta_1 \cup \{\mathbf{F}\alpha\})$-tableau. A new application of the inductive hypothesis allowed by the fact that $|\alpha| < |\varphi|$ shows that $\mathtt{T}'$ is a strongly closed $(\Delta_0 \cup \Delta_1)$-tableau.

Observe that in Case 3.2.3.2.2, $\mathtt{V}_i$ is a strongly closed $(\Delta_0 \cup \Delta_1 \cup K_i)$-tableau, which follows from the fact that $|(\mathtt{T}_0)_{[i]}| + |\mathtt{T}_1| < |\mathtt{T}_0| + |\mathtt{T}_1|$ and the inductive hypothesis. This case can now be completed using the same argument as in Case 3.2.3.2.1.

Case 3.2.3.2.3 is treated like Case 3.2.3.2.2. Finally, Case 3.2.3.2.4, is proven by combining the arguments used in the previous two cases. $\square$

In Case 3.2.3.2 of Construction 3.4.11, we sacrificed efficiency for a uniform treatment of the binary connectives, which can be generalized to arbitrary sets of connectives. Supplement 43 indicates how we can take advantage of the nature of the constituents for specific connectives to reduce the number of steps in the construction.

The following construction gives a syntactic way of transforming strongly closed tableaux that use the cut rule into ones that do not.

Construction 3.4.12.
Input: A strongly closed Δ-tableau with cut $\mathtt{T}$.
Output: A strongly closed Δ-tableau without cut $\mathsf{CET}(\mathtt{T})$.
Method: If $\mathtt{T}$ is a one-node strongly closed $\mathtt{T}(\lambda)$-tableau with cut, then $\mathsf{CET}(\mathtt{T}) = \mathtt{T}$.
If $\mathtt{T}$ has more than one node and there is a signed formula $b\varphi$ such that $\mathtt{T}(\lambda) = \Delta' \cup \{b\varphi\}$, $\mathtt{d}(b\varphi) = (K_0, \ldots, K_{n-1})$, and $\mathtt{T}(i) = \Delta' \cup K_i$ for $0 \leq i \leq n-1$, then choose $b\varphi$ to be the first such formula in the standard order and let

$$\mathsf{CET}(\mathtt{T}) = (\mathsf{CET}(\mathtt{T}_{[0]}), \ldots, \mathsf{CET}(\mathtt{T}_{[n-1]}); \mathtt{T}(\lambda)).$$

If neither of the two previous cases apply and the cut rule was not applied at the root of $\mathtt{T}$, then thinning was used at the root and we let

$$\mathsf{CET}(\mathtt{T}) = \mathsf{CET}((\mathtt{T})_{[0]}) \uplus \Delta.$$

If none of the previous cases apply, then the cut rule was applied at the root of $\mathtt{T}$ and there is a formula φ such that $\mathtt{T}(0) = \Delta \cup \{\mathbf{T}\varphi\}$ and $\mathtt{T}(1) = \Delta \cup \{\mathbf{F}\varphi\}$. There could be several possible choices for φ. We select the first formula φ in the standard order for which the decomposition can be made. Now, we can define

$$\mathsf{CET}(\mathtt{T}) = \mathsf{cet}(\mathsf{CET}(\mathtt{T}_{[0]}), \mathsf{CET}(\mathtt{T}_{[1]}), \Delta, \Delta, \varphi).$$

Proof of Correctness: One can show by induction on $|\mathtt{T}|$ that the construction halts with proper output on $\mathtt{T}$. We leave this straightforward argument to the reader. □

Note that if the sets of formulas involved in Constructions 3.4.11 and 3.4.12 are finite, then these constructions become effective and yield algorithms.

Example 3.4.13. We saw in Example 3.4.2 how to construct a strongly closed $\{\mathbf{F}\psi\}$-tableau with cut $\mathtt{T}$ from a strongly closed $\{\mathbf{F}\varphi\}$-tableau $\mathtt{T}_0$ and a strongly closed $\{\mathbf{F}(\varphi \to \psi)\}$-tableau $\mathtt{T}_1$. The previous construction allows us to obtain a strongly closed $\{\mathbf{F}\psi\}$-tableau without cut $\mathsf{CET}(\mathtt{T})$. ▯

In the following, we recapitulate the notations used for the formal systems introduced in this section:

Infinite sets	Conservative tableaux	Cut rule	Notation
No	No	No	$\mathcal{F}^{\text{tabl}}$
No	Yes	No	$\mathcal{F}^{\text{tabl,cons}}$
No	No	Yes	$\mathcal{F}^{\text{tabl,cut}}$
No	Yes	Yes	$\mathcal{F}^{\text{tabl,cons,cut}}$
Yes	No	No	$\mathcal{F}^{\text{tabl},\infty}$
Yes	Yes	No	$\mathcal{F}^{\text{tabl},\infty,\text{cons}}$
Yes	No	Yes	$\mathcal{F}^{\text{tabl},\infty,\text{cut}}$
Yes	Yes	Yes	$\mathcal{F}^{\text{tabl},\infty,\text{cons,cut}}$

3.5 Sequents

The notion of sequent and a formal system that deals with sequents were introduced by German logician Gerhard Gentzen.[4]

Definition 3.5.1. A *sequent* is a pair $\kappa = (\Gamma, \Gamma')$ of sets of formulas. We will denote the sequent (Γ, Γ') by $\Gamma \Rightarrow \Gamma'$.

Γ is the *antecedent* and Γ' is the *succeedent* of the sequent $\Gamma \Rightarrow \Gamma'$.

The sequent $\Gamma \Rightarrow \Gamma'$ is *finite* if both Γ and Γ' are finite.

We will denote the set of all sequents by SQT and the set of all finite sequents by $\mathsf{SQT}^{\text{fin}}$. □

Note that if $\Gamma \Rightarrow \Gamma'$ is a sequent, either or both of Γ and Γ' could be empty. We denote the sequents $\emptyset \Rightarrow \Gamma'$, $\Gamma \Rightarrow \emptyset$ and $\emptyset \Rightarrow \emptyset$ by $\Rightarrow \Gamma'$, $\Gamma \Rightarrow$ and $\Rightarrow$, respectively.

The finite sequent $\{\varphi_0, \ldots, \varphi_{n-1}\} \Rightarrow \{\psi_0, \ldots, \psi_{m-1}\}$ will be written as $\varphi_0, \ldots, \varphi_{n-1} \Rightarrow \psi_0, \ldots, \psi_{m-1}$. More generally, we will frequently use the simpler notation $\Gamma, \varphi_0, \ldots, \varphi_{n-1} \Rightarrow \Gamma', \psi_0, \ldots, \psi_{m-1}$ for a sequent $\Gamma \cup \{\varphi_0, \ldots, \varphi_{n-1}\} \Rightarrow \Gamma' \cup \{\psi_0, \ldots, \psi_{m-1}\}$.

Definition 3.5.2. A truth assignment v *satisfies* a sequent $\Gamma \Rightarrow \Gamma'$ if there exists a formula $\varphi \in \Gamma$ such that $v(\varphi) = \mathbf{F}$ or there exists a formula $\psi \in \Gamma'$ such that $v(\psi) = \mathbf{T}$. If v does not satisfy the sequent, then it *falsifies* the sequent.

A sequent is *valid* if it is satisfied by every truth assignment. □

Example 3.5.3. Any sequent $\Gamma \Rightarrow \Gamma'$ such that $\Gamma \cap \Gamma' \neq \emptyset$ is valid. □

Theorem 3.5.4. *Let Γ be a set of formulas and let φ be a formula. Then, we have the following:*

(1) $\Gamma \models \varphi$ *if and only if the sequent* $\Gamma \Rightarrow \varphi$ *is valid. (In particular, φ is a tautology if and only if $\Rightarrow \varphi$ is valid.)*
(2) Γ *is satisfiable if and only if* $\Gamma \Rightarrow$ *is not valid.*

[4]Gerhard Karl Erich Gentzen was born in Greifswald, Pomerania, on November 24, 1909, and died in tragic circumstances in Prague on August 4, 1945. He studied at the Universities of Greifswald and Göttingen and worked as Hilbert's assistant at the latter university. Gentzen is known for his contributions to the study of the consistency of a system of arithmetic and for his results in developing more natural methods of proof.

Proof. The argument for this simple theorem is left for the reader. □

The mappings

$$\texttt{sf} : \mathsf{SQT} \longrightarrow \mathcal{P}(\text{SPLFORM}) \text{ and } \texttt{sqt} : \mathcal{P}(\text{SPLFORM}) \longrightarrow \mathsf{SQT}$$

that we are about to introduce help us relate sequents to tableaux.

Definition 3.5.5. Let $\kappa = \Gamma \Rightarrow \Gamma'$ be a sequent. The set of signed formulas $\texttt{sf}(\kappa)$ is defined to be $\{\mathbf{T}\varphi \mid \varphi \in \Gamma\} \cup \{\mathbf{F}\varphi \mid \varphi \in \Gamma'\}$.

Let Δ be a set of signed formulas. The sequent $\texttt{sqt}(\Delta)$ is given by $\{\varphi \mid \mathbf{T}\varphi \in \Delta\} \Rightarrow \{\varphi \mid \mathbf{F}\varphi \in \Delta\}$. ▯

It is easy to see that the mappings sf and sqt are both bijections and are inverse to each other.

Let $\kappa_0 = \Gamma_0 \Rightarrow \Gamma'_0, \kappa_1 = \Gamma_1 \Rightarrow \Gamma'_1$ be two sequents. We denote by $\kappa_0 \cup \kappa_1$ the sequent $\Gamma_0 \cup \Gamma_1 \Rightarrow \Gamma'_0 \cup \Gamma'_1$. The reader can verify that

$$\texttt{sqt}(\Delta_0 \cup \Delta_1) = \texttt{sqt}(\Delta_0) \cup \texttt{sqt}(\Delta_1),$$
$$\texttt{sf}(\kappa_0 \cup \kappa_1) = \texttt{sf}(\kappa_0) \cup \texttt{sf}(\kappa_1),$$

for $\Delta_0, \Delta_1 \subseteq \text{SPLFORM}$ and $\kappa_0, \kappa_1 \in \mathsf{SQT}$.

Theorem 3.5.6. *Let κ be a sequent and v be a truth assignment. Then, the following two conditions are equivalent:*

(1) *v satisfies κ.*
(2) *v does not satisfy the set of signed formulas $\texttt{sf}(\kappa)$.*

In addition, if $\kappa = \{\varphi_0, \ldots, \varphi_{n-1}\} \Rightarrow \{\psi_0, \ldots, \psi_{m-1}\}$ and $n, m \geq 1$, then the above conditions are equivalent to the following:

(3) *v satisfies the formula $((\varphi_0 \wedge \cdots \wedge \varphi_{n-1}) \rightarrow (\psi_0 \vee \cdots \vee \psi_{m-1}))$.*

Proof. The argument is straightforward and it is left to the reader. □

Corollary 3.5.7. *Let κ be a sequent. Then the following two conditions are equivalent:*

(1) *The sequent κ is valid.*
(2) *The set of signed formulas $\texttt{sf}(\kappa)$ is unsatisfiable.*

In addition, if $\kappa = \{\varphi_0, \ldots, \varphi_{n-1}\} \Rightarrow \{\psi_0, \ldots, \psi_{m-1}\}$ *and* $n, m \geq 1$, *then the above conditions are equivalent to the following:*

(3) *The formula* $((\varphi_0 \wedge \cdots \wedge \varphi_{n-1}) \to (\psi_0 \vee \cdots \vee \psi_{m-1}))$ *is a tautology.*

Proof. This follows immediately from Theorem 3.5.6. □

Corollary 3.5.7 shows that it is possible to determine the validity of sequents using tableaux. Actually, in this section, we will introduce a formal system $\mathcal{F}^{\mathrm{seq},\infty}$ whose theorems are the valid sequents. This formal system is the counterpart of the formal system $\mathcal{F}^{\mathrm{tabl},\infty}$ which we introduced previously for tableaux. A proof tree in $\mathcal{F}^{\mathrm{seq},\infty}$ for a sequent κ will be another way of representing the construction of a strongly closed tableau that shows that the set $\mathtt{sf}(\kappa)$ is unsatisfiable. In fact, proof trees in $\mathcal{F}^{\mathrm{seq},\infty}$ will be translations under $\mathtt{sqt}$ of proof trees in $\mathcal{F}^{\mathrm{tabl},\infty}$. We will also show that a "subsystem" $\mathcal{F}^{\mathrm{seq}}$ of the formal system $\mathcal{F}^{\mathrm{seq},\infty}$ whose axioms are finite sequents is sound and complete with respect to the set of valid finite sequents.

Definition 3.5.8. Let κ be a sequent and let S be a set of sequents. We say that κ is *dominated by S* if $\mathtt{sf}(\kappa) \subseteq \mathtt{sf}(S)$. If $S = \{\kappa'\}$, then we say that κ is *dominated* by κ' (or κ' *dominates* κ) when κ is dominated by S. In other words, $\Gamma_0 \Rightarrow \Gamma_1$ is dominated by $\Gamma_0' \Rightarrow \Gamma_1'$ if $\Gamma_0 \subseteq \Gamma_0'$ and $\Gamma_1 \subseteq \Gamma_1'$. ▯

Note that if κ is dominated by κ' and κ is valid, then κ' is also valid.

Definition 3.5.9. For the *conservative formal system*

$$\mathcal{F}^{\mathrm{seq},\infty,\mathrm{cons}} = (\mathsf{SQT}, A, I),$$

the set of axioms A consists of all sequents $\Gamma \Rightarrow \Gamma'$ such that $\Gamma \cap \Gamma' \neq \emptyset$, and the set of rules of inference I consists of the rules $\mathsf{R}_{C,l}$ and $\mathsf{R}_{C,r}$, given in Figure 3.12, where $C \in \{\neg, \vee, \wedge, \to, \leftrightarrow\}$, Γ, Γ' are sets of formulas and φ, ψ are formulas.

The rules $\mathsf{R}_{C,l}$ and $\mathsf{R}_{C,r}$ are often called the C*-left* and the C*-right* rule, respectively, for each connective C. A *connective rule* is either a C-left or a C-right rule for some connective symbol C.

$$\frac{\Gamma \Rightarrow \Gamma', \varphi}{\Gamma, (\neg\varphi) \Rightarrow \Gamma'}\mathsf{R}_{\neg,l} \qquad \frac{\Gamma, \varphi \Rightarrow \Gamma'}{\Gamma \Rightarrow \Gamma', (\neg\varphi)}\mathsf{R}_{\neg,r}$$

$$\frac{\Gamma, \varphi \Rightarrow \Gamma' \quad \Gamma, \psi \Rightarrow \Gamma'}{\Gamma, (\varphi \vee \psi) \Rightarrow \Gamma'}\mathsf{R}_{\vee,l} \qquad \frac{\Gamma \Rightarrow \Gamma', \varphi, \psi}{\Gamma \Rightarrow \Gamma', (\varphi \vee \psi)}\mathsf{R}_{\vee,r}$$

$$\frac{\Gamma, \varphi, \psi \Rightarrow \Gamma'}{\Gamma, (\varphi \wedge \psi) \Rightarrow \Gamma'}\mathsf{R}_{\wedge,l} \qquad \frac{\Gamma \Rightarrow \Gamma', \varphi \quad \Gamma \Rightarrow \Gamma', \psi}{\Gamma \Rightarrow \Gamma', (\varphi \wedge \psi)}\mathsf{R}_{\wedge,r}$$

$$\frac{\Gamma \Rightarrow \Gamma', \varphi \quad \Gamma, \psi \Rightarrow \Gamma'}{\Gamma, (\varphi \to \psi) \Rightarrow \Gamma'}\mathsf{R}_{\to,l} \qquad \frac{\Gamma, \varphi \Rightarrow \Gamma', \psi}{\Gamma \Rightarrow \Gamma', (\varphi \to \psi)}\mathsf{R}_{\to,r}$$

$$\frac{\Gamma, \varphi, \psi \Rightarrow \Gamma' \quad \Gamma \Rightarrow \Gamma', \varphi, \psi}{\Gamma, (\varphi \leftrightarrow \psi) \Rightarrow \Gamma'}\mathsf{R}_{\leftrightarrow,l} \qquad \frac{\Gamma, \varphi \Rightarrow \Gamma', \psi \quad \Gamma, \psi \Rightarrow \Gamma', \varphi}{\Gamma \Rightarrow \Gamma', (\varphi \leftrightarrow \psi)}\mathsf{R}_{\leftrightarrow,r}.$$

Fig. 3.12. Rules of the formal system $\mathcal{F}^{\text{seq},\infty,\text{cons}}$.

If add the rule R_{thin} given by

$$\frac{\kappa}{\kappa'}R_{\text{thin}},$$

where κ' dominates κ, we obtain the formal system $\mathcal{F}^{\text{seq},\infty}$.

The formal system $\mathcal{F}^{\text{seq,cons}}$ is the triple $(\mathsf{SQT}^{\text{fin}}, A \cap \mathsf{SQT}^{\text{fin}}, I_{\text{fin}})$ where the definition of the rules in I_{fin} is obtained from the definition of the rules in I by requiring the sequents involved to be finite.

Finally, the formal system $\mathcal{F}^{\text{seq}}$ is obtained from $\mathcal{F}^{\text{seq,cons}}$ by adding the thinning rule restricted to finite sequents. ∎

By Theorem 1.8.4, $\mathcal{F}^{\text{seq},\infty}$ is an extension of $\mathcal{F}^{\text{seq}}$ and $\mathcal{F}^{\text{seq},\infty,\text{cons}}$ is an extension of $\mathcal{F}^{\text{seq,cons}}$. Also, for every instance of a rule of $\mathcal{F}^{\text{seq},\infty,\text{cons}}$, the conclusion is a finite sequent if and only if all of the hypotheses are finite sequents. Moreover, if the conclusion of a rule of $\mathcal{F}^{\text{seq},\infty}$ is a finite sequent, then the premises of the rule are also finite sequents.

The following results show the connection between various sequent formal systems and tableaux and will be instrumental in showing the soundness and completeness of these formal systems.

Theorem 3.5.10. *For each set of signed formulas Δ and signed formula $b\varphi$, where $\varphi \notin SV$ and $\mathtt{d}(b\varphi) = (K_0, \ldots, K_{n-1})$,*

$$\frac{\mathtt{sqt}(\Delta \cup K_0), \ldots, \mathtt{sqt}(\Delta \cup K_{n-1})}{\mathtt{sqt}(\Delta \cup \{b\varphi\})}$$

is an instance of a connective rule, and every such instance can be obtained in this way.

For all sets of signed formulas Δ, Δ' such that $\Delta \subseteq \Delta'$,

$$\frac{\mathtt{sqt}(\Delta)}{\mathtt{sqt}(\Delta')}$$

is an instance of R_{thin} and every instance of this rule has this form.

Proof. The theorem follows by inspecting the definition of $\mathtt{d}(b\varphi)$ and the form of the rules. □

Corollary 3.5.11. *For each instance*

$$\frac{\kappa_0, \ldots, \kappa_{n-1}}{\kappa}$$

of a connective rule, there is a set of signed formulas Δ and a signed formula $b\varphi$ (where φ is not a statement variable) such that $\mathtt{d}(b\varphi) = (K_0, \ldots, K_{n-1})$, $\mathtt{sf}(\kappa) = \Delta \cup \{b\varphi\}$, and $\mathtt{sf}(\kappa_i) = \Delta \cup K_i$ for $0 \leq i \leq n-1$.

Further, for each instance

$$\frac{\kappa}{\kappa'}$$

of the thinning rule, there are sets of signed formulas Δ, Δ', with $\Delta \subseteq \Delta'$, such that $\kappa = \mathtt{sqt}(\Delta)$ and $\kappa' = \mathtt{sqt}(\Delta')$.

Proof. This is an immediate consequence of Theorem 3.5.10, taking into account the fact that $\mathtt{sqt}$ and $\mathtt{sf}$ are inverse bijections. □

If

$$\frac{\Gamma_0 \Rightarrow \Gamma_0', \ldots, \Gamma_{n-1} \Rightarrow \Gamma_{n-1}'}{\Gamma \Rightarrow \Gamma'}$$

is an instance of a rule of $\mathcal{F}^{\text{seq},\infty}$, then $\Gamma_i, \Gamma_i' \subseteq \text{SUBF}(\Gamma \cup \Gamma')$, for $0 \leq i \leq n-1$. Thus, every formula that appears in a generalized deduction tree $\mathtt{T} \in \mathcal{GDT}_{\mathcal{F}^{\text{seq},\infty}}$ for a sequent κ is a subformula of a formula that occurs in κ. This shows the analyticity of the formal systems $\mathcal{F}^{\text{seq}}$ and $\mathcal{F}^{\text{seq},\infty}$.

Definition 3.5.12. Let

$$\frac{\kappa_0, \ldots, \kappa_{n-1}}{\kappa}$$

be an instance of a connective rule and let Δ and $b\varphi$ be as in Corollary 3.5.11. Then we call φ a *principal formula* of the instance. If $b\varphi \in \Delta$, we call the instance an *instance with retention*; otherwise, we refer to this instance as an *instance with removal.* $\square$

It is not difficult to see that an instance of a connective rule cannot be both an instance with retention and an instance with removal. Further, in an instance with removal, there is only one principal formula. The following example shows, however, that in an instance with retention, there could be several principal formulas.

Example 3.5.13. Consider the instance

$$\frac{(p_0 \wedge p_1), (p_0 \wedge p_2), p_0, p_1, p_2 \Rightarrow}{(p_0 \wedge p_1), (p_0 \wedge p_2), p_1, p_2 \Rightarrow}\ \mathsf{R}_{\wedge,l}\,.$$

Observe that both formulas $(p_0 \wedge p_1)$ and $(p_0 \wedge p_2)$ can serve as principal formulas of this instance.

Note that the instance

$$\frac{(p_0 \wedge p_1), p_0, p_1 \Rightarrow (\neg p_0)}{(p_0 \wedge p_1), p_1 \Rightarrow (\neg p_0)}$$

can be regarded as an instance of $\mathsf{R}_{\wedge,l}$ with principal formula $(p_0 \wedge p_1)$ and as an instance of $\mathsf{R}_{\neg,r}$ with principal formula $(\neg p_0)$. $\square$

The following theorem formalizes some of our previous remarks on the connection between deduction trees of $\mathcal{F}^{\mathrm{seq},\infty}$ and $\mathcal{F}^{\mathrm{tabl},\infty}$.

Theorem 3.5.14. *For* $\mathtt{T} \in \mathcal{GDT}_{\mathcal{F}^{seq,\infty}}$, *define* $\Phi(\mathtt{T}) = \mathtt{sf} \circ \mathtt{T}$ *and for* $\mathtt{T} \in \mathcal{GDT}_{\mathcal{F}^{tabl,\infty}}$, *define* $\Psi(\mathtt{T}) = \mathtt{sqt} \circ \mathtt{T}$. *Then,* $\Phi : \mathcal{GDT}_{\mathcal{F}^{seq,\infty}} \to \mathcal{GDT}_{\mathcal{F}^{tabl,\infty}}$ *and* $\Psi : \mathcal{GDT}_{\mathcal{F}^{tabl,\infty}} \longrightarrow \mathcal{GDT}_{\mathcal{F}^{seq,\infty}}$ *are inverse bijections and* $\Phi(\mathcal{PT}_{\mathcal{F}^{seq,\infty}}) = \mathcal{PT}_{\mathcal{F}^{tabl,\infty}}$ *and* $\Psi(\mathcal{PT}_{\mathcal{F}^{tabl,\infty}}) = \mathcal{PT}_{\mathcal{F}^{seq,\infty}}$.

Also, if $\mathtt{T} \in \mathcal{GDT}_{\mathcal{F}^{seq,\infty,cons}}$ $(\mathtt{T} \in \mathcal{PT}_{\mathcal{F}^{seq,\infty,cons}})$, *then* $\Phi(\mathtt{T}) \in \mathcal{GDT}_{\mathcal{F}^{tabl,\infty,cons}}$ $(\Phi(\mathtt{T}) \in \mathcal{PT}_{\mathcal{F}^{tabl,\infty,cons}})$ *and if* $\mathtt{T} \in \mathcal{GDT}_{\mathcal{F}^{tabl,\infty,cons}}$ $(\mathtt{T} \in \mathcal{PT}_{\mathcal{F}^{tabl,\infty,cons}})$, *then* $\Psi(\mathtt{T}) \in \mathcal{GDT}_{\mathcal{F}^{tabl,\infty,cons}}$ $(\Psi(\mathtt{T}) \in \mathcal{PT}_{\mathcal{F}^{tabl,\infty,cons}})$.

Proof. Corollary 3.5.11 implies that Φ maps $\mathcal{GDT}_{\mathcal{F}^{\mathrm{seq},\infty}}$ into $\mathcal{GDT}_{\mathcal{F}^{\mathrm{tabl},\infty}}$ and Theorem 3.5.10 implies that Ψ maps $\mathcal{GDT}_{\mathcal{F}^{\mathrm{tabl},\infty}}$ into $\mathcal{GDT}_{\mathcal{F}^{\mathrm{seq},\infty}}$. Since $\mathtt{sf}$ and $\mathtt{sqt}$ are inverse mappings, it follows immediately that Φ and Ψ are inverse mappings. The next statement follows from the fact that the mappings $\mathtt{sf}$ and $\mathtt{sqt}$ preserve the axioms of the formal systems $\mathcal{F}^{\mathrm{seq},\infty}$ and $\mathcal{F}^{\mathrm{tabl},\infty}$; in other words, if κ is an axiom of $\mathcal{F}^{\mathrm{seq},\infty}$, then $\mathtt{sf}(\kappa)$ is an axiom of $\mathcal{F}^{\mathrm{tabl},\infty}$, and if Δ is an axiom of $\mathcal{F}^{\mathrm{tabl},\infty}$, then $\mathtt{sqt}(\Delta)$ is an axiom of $\mathcal{F}^{\mathrm{seq},\infty}$. Finally, the second part of the theorem follows from the first part by omitting the usage of thinning rules. $\square$

Theorem 3.5.15 (Soundness of $\mathcal{F}^{\mathrm{seq},\infty}$). *Every theorem of* $\mathcal{F}^{seq,\infty}$ *is a valid sequent.*

Proof. Let κ be a theorem of $\mathcal{F}^{\mathrm{seq},\infty}$ and let $\mathtt{T}$ be an $\mathcal{F}^{\mathrm{seq},\infty}$-proof tree for κ. Then, $\Phi(\mathtt{T})$ is an $\mathcal{F}^{\mathrm{tabl},\infty}$-proof tree for $\mathtt{sf}(\kappa)$, so $\mathtt{sf}(\kappa)$ is an unsatisfiable set of signed formulas by Theorem 3.3.43 which, by Corollary 3.5.7, implies the validity of κ. $\square$

Corollary 3.5.16 (Soundness of $\mathcal{F}^{\mathrm{seq}}, \mathcal{F}^{\mathrm{seq,cons}}, \mathcal{F}^{\mathrm{seq},\infty,\mathrm{cons}}$). *Every theorem of* $\mathcal{F}^{seq}$ *and* $\mathcal{F}^{seq,cons}$ *is a valid finite sequent. Further, every theorem of* $\mathcal{F}^{seq,\infty,cons}$ *is a valid sequent.*

Proof. This is immediate since every theorem of any of these formal systems is a theorem of $\mathcal{F}^{\mathrm{seq},\infty}$. $\square$

Example 3.5.17. Let $\varphi = (((\neg\alpha) \to (\neg\beta)) \to (\beta \to \alpha))$. We will prove that φ is a tautology by showing that the sequent $\Rightarrow \varphi$ is valid. Figure 3.13 gives a $\mathcal{F}^{\mathrm{seq}}$-proof tree for $\Rightarrow \varphi$, so, by the Soundness Theorem, $\Rightarrow \varphi$ is valid. $\square$

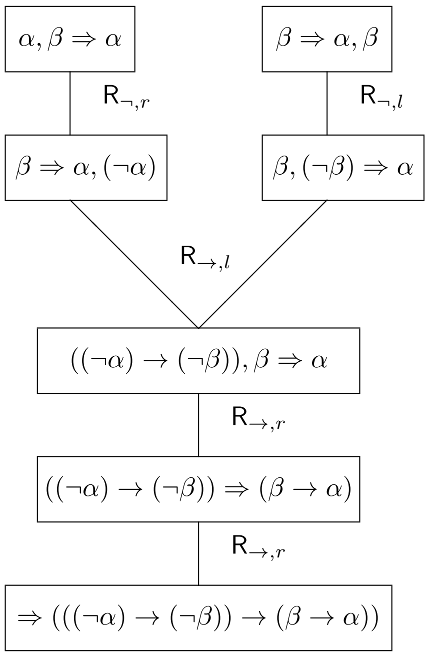

$\alpha, \beta \Rightarrow \alpha$
$\beta \Rightarrow \alpha, \beta$
$\mathsf{R}_{\neg,r}$
$\mathsf{R}_{\neg,l}$
$\beta \Rightarrow \alpha, (\neg\alpha)$
$\beta, (\neg\beta) \Rightarrow \alpha$
$\mathsf{R}_{\rightarrow,l}$
$((\neg\alpha) \rightarrow (\neg\beta)), \beta \Rightarrow \alpha$
$\mathsf{R}_{\rightarrow,r}$
$((\neg\alpha) \rightarrow (\neg\beta)) \Rightarrow (\beta \rightarrow \alpha)$
$\mathsf{R}_{\rightarrow,r}$
$\Rightarrow (((\neg\alpha) \rightarrow (\neg\beta)) \rightarrow (\beta \rightarrow \alpha))$

Fig. 3.13. Proof of validity for $\Rightarrow (((\neg\alpha) \rightarrow (\neg\beta)) \rightarrow (\beta \rightarrow \alpha))$.

Theorem 3.5.18 (Completeness of $\mathcal{F}^{\mathbf{seq},\infty,\mathbf{cons}}$). *Every valid sequent is a theorem of $\mathcal{F}^{seq,\infty,cons}$.*

Proof. Let κ be a valid sequent. Then, $\mathtt{sf}(\kappa)$ is unsatisfiable and, by Theorem 3.3.43, there is an $\mathcal{F}^{\mathrm{tabl},\infty,\mathrm{cons}}$-proof tree $\mathtt{T}$ for $\mathtt{sf}(\kappa)$. By Theorem 3.5.14, $\Psi(\mathtt{T})$ is a $\mathcal{F}^{\mathrm{seq},\infty,\mathrm{cons}}$-proof tree for $\mathtt{sqt}(\mathtt{sf}(\kappa)) = \kappa$. □

Corollary 3.5.19 (Completeness of $\mathcal{F}^{\mathbf{seq},\infty}, \mathcal{F}^{\mathbf{seq},\mathbf{cons}}, \mathcal{F}^{\mathbf{seq}}$). *Every valid sequent is a theorem of $\mathcal{F}^{seq,\infty}$. Every valid finite sequent is a theorem of $\mathcal{F}^{seq,cons}$ and $\mathcal{F}^{seq}$.*

Proof. The completeness of $\mathcal{F}^{\mathrm{seq},\infty}$ follows immediately from the completeness of $\mathcal{F}^{\mathrm{seq},\infty,\mathrm{cons}}$ because the latter formal system is extended by the former.

By the completeness of $\mathcal{F}^{\mathrm{seq},\infty,\mathrm{cons}}$, for every valid finite sequent κ, there is an $\mathcal{F}^{\mathrm{seq},\infty,\mathrm{cons}}$-proof tree $\mathtt{T}$ for κ. By the remark made after the definition of $\mathcal{F}^{\mathrm{seq},\infty}$, the nodes of this tree are finite sequents, so

T is an $\mathcal{F}^{\text{seq,cons}}$-proof tree for κ and therefore an $\mathcal{F}^{\text{seq}}$-proof tree for the same. $\square$

Theorem 3.5.20. *There is an effective, syntactic construction that starts with an $\mathcal{F}^{seq,\infty}$-proof tree* $\mathtt{T}$ *for a sequent κ and produces an $\mathcal{F}^{seq,\infty}$-proof tree* $\mathtt{T}'$ *for a finite sequent κ' that is dominated by κ.*

Proof. The argument involves the translation mappings $\Phi : \mathcal{GDT}_{\mathcal{F}^{\text{seq},\infty}} \to \mathcal{GDT}_{\mathcal{F}^{\text{tabl},\infty}}$ and $\Psi : \mathcal{GDT}_{\mathcal{F}^{\text{tabl},\infty}} \longrightarrow \mathcal{GDT}_{\mathcal{F}^{\text{seq},\infty}}$. By Theorem 3.5.14, these functions map proof trees to proof trees. Let $\mathtt{T}$ be a $\mathcal{F}^{\text{seq},\infty}$-proof tree for a sequent κ. Then, $\mathtt{T}_0 = \Phi(\mathtt{T})$ is an $\mathcal{F}^{\text{tabl},\infty}$-proof tree for the set $\mathtt{sf}(\kappa)$ that is a strongly closed $\mathtt{sf}(\kappa)$-tableau. By Theorem 3.3.40, we effectively obtain a strongly closed Δ'-tableau $\mathtt{T}_0'$, where Δ' is a finite subset of $\Delta = \mathtt{sf}(\kappa)$. Thus, there is a finite sequent $\kappa' = \mathtt{sqt}(\Delta')$ such that $\mathtt{T}' = \Psi(\mathtt{T}_0')$ is an $\mathcal{F}^{\text{seq},\infty}$-proof tree for κ'. Since $\mathtt{sf}(\kappa) = \Delta$ and $\mathtt{sf}(\kappa') = \Delta'$, κ dominates κ'. $\square$

Note that $\mathtt{T}'$ is actually a $\mathcal{F}^{\text{seq}}$-proof tree, as we observed in a previous remark.

Theorem 3.5.21 (Compactness Theorem for Sequents). *For every sequent κ, κ is valid if and only if it dominates a valid finite sequent κ'.*

Proof. If a sequent κ dominates a valid finite sequent κ', then, by the remark following the definition of domination, κ is valid.

Conversely, suppose that κ is a valid sequent. By the completeness of $\mathcal{F}^{\text{seq},\infty}$, there is an $\mathcal{F}^{\text{seq},\infty}$-proof tree for κ. Theorem 3.5.20 implies the existence of a $\mathcal{F}^{\text{seq},\infty}$-proof tree for a finite sequent κ' dominated by κ. By the soundness of $\mathcal{F}^{\text{seq},\infty}$, κ' is valid. $\square$

Definition 3.5.22. A general $\mathcal{F}^{\text{seq},\infty}$-deduction tree $\mathtt{T}$ is *finished* if $\Phi(\mathtt{T})$ is a strongly completed tableau. ▯

Definition 3.5.23. A sequent κ' is a *constituent* of the sequent $\varphi \Rightarrow$ if $\kappa' = \mathtt{sqt}(K)$ where K is a constituent of $\mathbf{T}\varphi$; κ' is a constituent of the sequent $\Rightarrow \varphi$ if $\kappa' = \mathtt{sqt}(K)$ where K is a constituent of $\mathbf{F}\varphi$.

A branch $\mathtt{B}$ of a general $\mathcal{F}^{\text{seq},\infty}$-deduction tree $\mathtt{T}$ is called *finished* if

(1) for no statement variable p does $\mathtt{T}(\mathtt{B})$ dominate both $p \Rightarrow$ and $\Rightarrow p$ and

(2) for every sequent κ of the form $\varphi \Rightarrow$ or $\Rightarrow \varphi$, with φ not a literal, if κ is dominated by $\mathtt{T}(\mathtt{B})$, then there is a constituent κ' of κ that is dominated by $\mathtt{T}(\mathtt{B})$. $\square$

Theorem 3.5.24. *A general $\mathcal{F}^{seq,\infty}$-deduction tree* $\mathtt{T}$ *is finished if and only if every branch of* $\mathtt{T}$ *is either finished or ends with a node labeled by an axiom of $\mathcal{F}^{seq,\infty}$.*

Proof. The theorem follows immediately from the observation that a branch $\mathtt{B}$ of a general $\mathcal{F}^{\text{seq},\infty}$-deduction tree $\mathtt{T}$ is finished (in the sense of Definition 3.5.23) if and only if $\mathtt{sf}(\mathtt{T}(\mathtt{B}))$ is a Hintikka set. $\square$

Example 3.5.25. Consider the sequent $(p_0 \vee p_1) \Rightarrow (p_0 \wedge p_1)$. Note that $\mathtt{sf}((p_0 \vee p_1) \Rightarrow (p_0 \wedge p_1)) = \{\mathbf{T}(p_0 \vee p_1), \mathbf{F}(p_0 \wedge p_1)\}$ and we have shown in Example 3.3.25 that this set of signed formulas is satisfiable. A finished $\mathcal{F}^{\text{seq}}$-deduction tree for the sequent $(p_0 \vee p_1) \Rightarrow (p_0 \wedge p_1)$ is shown in Figure 3.14. $\square$

Theorem 3.5.26. *There is a finished $\mathcal{F}^{seq,cons}$-deduction tree for every finite sequent κ.*

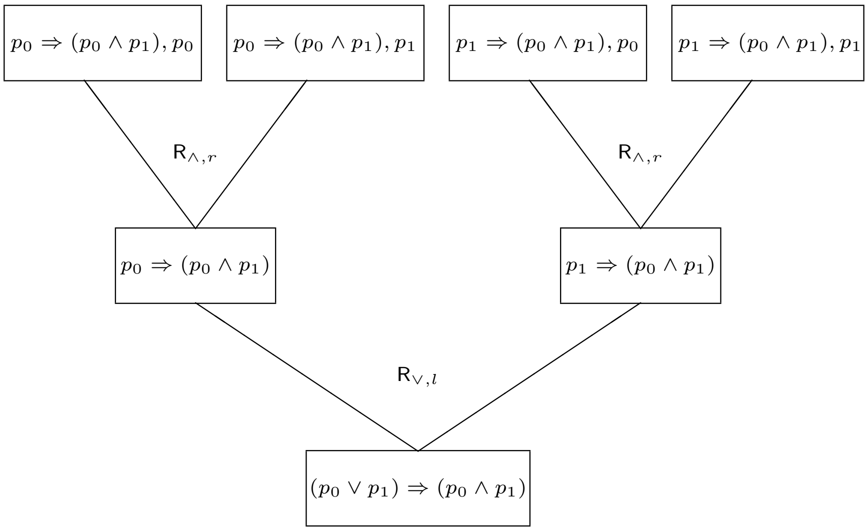

$p_0 \Rightarrow (p_0 \wedge p_1), p_0$
$p_0 \Rightarrow (p_0 \wedge p_1), p_1$
$p_1 \Rightarrow (p_0 \wedge p_1), p_0$
$p_1 \Rightarrow (p_0 \wedge p_1), p_1$
$\mathsf{R}_{\wedge,r}$
$\mathsf{R}_{\wedge,r}$
$p_0 \Rightarrow (p_0 \wedge p_1)$
$p_1 \Rightarrow (p_0 \wedge p_1)$
$\mathsf{R}_{\vee,l}$
$(p_0 \vee p_1) \Rightarrow (p_0 \wedge p_1)$

Fig. 3.14. Finished deduction tree for $(p_0 \vee p_1) \Rightarrow (p_0 \wedge p_1)$.

If κ is a sequent, there is a finished general $\mathcal{F}^{seq,\infty,cons}$-deduction tree for κ.

Proof. Using Construction 3.3.37 (or Algorithm 3.3.27 for the finite case), it is possible to construct a strongly completed (finite) $\mathtt{sf}(\kappa)$-tableau $\mathtt{T}$. Then, by Theorem 3.5.14, when κ is a finite sequent, $\mathtt{sqt} \circ \mathtt{T}$ is a finished $\mathcal{F}^{\mathrm{seq,cons}}$-deduction tree for $\mathtt{sqt}(\mathtt{sf}(\kappa)) = \kappa$; otherwise, $\mathtt{sqt} \circ \mathtt{T}$ is a finished general $\mathcal{F}^{\mathrm{seq},\infty,\mathrm{cons}}$-deduction tree for κ. □

In the finite case, the proof of Theorem 3.5.26 in fact gives an algorithm, albeit not the most direct one, for producing a finished $\mathcal{F}^{\mathrm{seq}}$-deduction tree for a given sequent κ. We can rephrase this algorithm in terms of sequents.

Algorithm 3.5.27.
Input: A finite sequent κ.
Output: A finished $\mathcal{F}^{\mathrm{seq,cons}}$-deduction tree for κ.
Method: We construct a sequence $\mathtt{T}_0, \mathtt{T}_1, \ldots$ of $\mathcal{F}^{\mathrm{seq,cons}}$-deduction trees for κ such that each $\mathtt{T}_{i+1}$ is a leaf extension of $\mathtt{T}_i$ using the following steps:

(A) Let $\mathtt{T}_0$ be the one-node tree with root labeled by κ.

(B) Suppose that $\mathtt{T}_i$ has been defined. Then, if $\mathtt{T}_i$ is finished, the algorithm stops with $\mathtt{T}_i$ as the output. Otherwise, $\mathtt{T}_i$ has branches that neither are finished nor end in an axiom. Select nondeterministically such a branch $\mathtt{B}$ ending in the leaf q. Then, select nondeterministically a sequent κ_q of one of the forms $\varphi \Rightarrow$ or $\Rightarrow \varphi$ with φ not a literal, such that κ_q is dominated by $\mathtt{T}(q)$ but no constituent of κ_q is dominated by $\mathtt{T}(\mathtt{B})$. (By an analog of Lemma 3.3.19, such a sequent exists.) Let

$$\frac{\kappa_0, \ldots, \kappa_{n-1}}{\mathtt{T}(q)}$$

be an instance of a rule of $\mathcal{F}^{\mathrm{seq,cons}}$ with φ as principal formula. (Since $\mathtt{T}(q)$ is not an axiom, there is only one rule of which this can be an instance.) Define $\mathtt{T}_{i+1}$ by adding to $\mathrm{Dom}(\mathtt{T}_i)$ the nodes $q0, \ldots, q(n-1)$ and defining $\mathtt{T}_{i+1}(qj) = \kappa_j$ for $0 \le j \le n-1$.

Proof of Correctness: It is easy to see that each $\mathtt{T}_i$ is an $\mathcal{F}^{\text{seq,cons}}$-deduction tree for κ. Further, $\mathtt{sf} \circ \mathtt{T}_0, \ldots$ is a sequence of $\mathtt{sf}(\kappa)$-tableaux that can be obtained by applying Algorithm 3.3.27 to $\mathtt{sf}(\kappa)$. Thus, the correctness of that algorithm implies the correctness of the present algorithm. □

The above algorithm can produce a finished $\mathcal{F}^{\text{seq,cons}}$-deduction tree with retention or with removal.

A similar rephrasing can be applied to Construction 3.3.37 to generate a direct construction of a finished general $\mathcal{F}^{\text{seq},\infty,\text{cons}}$-deduction tree for an arbitrary sequent κ.

Construction 3.5.28.
Input: A sequent κ.
Output: A (finite or infinite) sequence $\mathtt{T}_0, \mathtt{T}_1, \ldots$ of $\mathcal{F}^{\text{seq},\infty,\text{cons}}$-deduction trees for κ such that each $\mathtt{T}_{i+1}$ is a leaf extension of $\mathtt{T}_i$ and $\mathtt{T} = \bigcup\{\mathtt{T}_i \mid i \geq 0\}$ is a finished general $\mathcal{F}^{\text{seq},\infty,\text{cons}}$-deduction tree for κ.
Method:

(A) Let $\mathtt{T}_0$ be the one-node tree with root labeled by κ.
(B) Suppose that $\mathtt{T}_i$ has been defined. Then, if $\mathtt{T}_i$ is finished, the construction stops (so $\mathtt{T} = \mathtt{T}_i$). Otherwise, $\mathtt{T}_i$ has branches that neither are finished nor end with an axiom. Select nondeterministically among the shortest such branches a branch $\mathtt{B}$ ending in the leaf q. Then, let φ be the first formula in the standard ordering of formulas such that for κ_q chosen from among $\varphi \Rightarrow$ and $\Rightarrow \varphi$, κ_q is dominated by $\mathtt{T}(q)$ and no constituent of κ_q is dominated by $\mathtt{T}(\mathtt{B})$. (By an analog of Lemma 3.3.19, such a sequent exists and is unique because $\mathtt{T}(q)$ is not an axiom.) Let

$$\frac{\kappa_0, \ldots, \kappa_{n-1}}{\mathtt{T}(q)}$$

be an instance of a rule of $\mathcal{F}^{\text{seq},\infty,\text{cons}}$ with φ as principal formula. (Since $\mathtt{T}(q)$ is not an axiom, there is only one rule of which this can be an instance.) Define $\mathtt{T}_{i+1}$ by adding to $\text{Dom}(\mathtt{T}_i)$ the nodes $q0, \ldots, q(n-1)$ and defining $\mathtt{T}_{i+1}(qj) = \kappa_j$ for $0 \leq j \leq n-1$.

Proof of Correctness: It is clear that each $\mathtt{T}_i$, if defined, is an $\mathcal{F}^{\text{seq},\infty,\text{cons}}$-deduction tree for κ and that $\mathtt{T}_{i+1}$, if defined, is a leaf extension of $\mathtt{T}_i$.

The sequence of trees $\mathtt{sf} \circ \mathtt{T}_0, \ldots$ is a sequence of $\mathtt{sf}(\kappa)$-tableaux that could have been obtained by applying Construction 3.3.37 to $\mathtt{sf}(\kappa)$. By the correctness of that construction, $\mathtt{T}' = \bigcup\{\mathtt{sf} \circ \mathtt{T}_i \mid i \geq 0\}$ is a strongly completed $\mathtt{sf}(\kappa)$-tableau. Therefore, since $\mathtt{sf} \circ \mathtt{T} = \mathtt{T}'$, $\mathtt{T}$ is a finished general $\mathcal{F}^{\text{seq},\infty,\text{cons}}$-deduction tree for κ. $\square$

As usual, we observe that the algorithm can be used to produce a general $\mathcal{F}^{\text{seq},\infty,\text{cons}}$-deduction tree with retention or with removal.

As just shown, given a (finite) sequent κ, we can construct a finished general $\mathcal{F}^{\text{seq},\infty,\text{cons}}$-deduction (finished $\mathcal{F}^{\text{seq},\text{cons}}$-deduction) tree $\mathtt{T}$ for κ. If $\mathtt{T}$ is a proof tree, then κ is valid. Otherwise, we can determine the truth assignments that falsify κ.

Theorem 3.5.29. *Let* $\mathtt{T}$ *be a finished general* $\mathcal{F}^{seq,\infty}$*-deduction tree for a sequent* κ*. For each finished branch* $\mathtt{B}$ *of* $\mathtt{T}$*, let* $v_{\mathtt{B}}$ *be the partial truth assignment defined by*

$$v_{\mathtt{B}}(p) = \begin{cases} \mathbf{T} \text{ if } p \Rightarrow \text{ is dominated by } \mathtt{T}(\mathtt{B}) \\ \mathbf{F} \text{ if } \Rightarrow p \text{ is dominated by } \mathtt{T}(\mathtt{B}). \end{cases}$$

Then, a truth assignment v *falsifies* κ *if and only if* v *extends some partial truth assignment* $v_{\mathtt{B}}$*, where* $\mathtt{B}$ *is a finished branch of* $\mathtt{T}$*.*

Proof. By Theorem 3.5.6, v falsifies κ if and only if it satisfies $\mathtt{sf}(\kappa)$. Since $\mathtt{sf} \circ \mathtt{T}$ is a (strongly) completed $\mathtt{sf}(\kappa)$-tableau, by Theorem 3.3.18, this is equivalent to having $v(p) = b$ for all $bp \in \mathtt{sf}(\mathtt{T}(\mathtt{B}))$ for some finished branch $\mathtt{B}$ of $\mathtt{T}$. In turn, this is equivalent to v extending one of the partial truth assignments $v_{\mathtt{B}}$. $\square$

Example 3.5.30. In the deduction tree $\mathtt{T}$ shown in Figure 3.14, we have the finished branches $\mathtt{B}_1 = \{\lambda, 0, 01\}$ and $\mathtt{B}_2 = \{\lambda, 1, 10\}$, where

$$\mathtt{T}(\mathtt{B}_1) = \{(p_0 \vee p_1) \Rightarrow (p_0 \wedge p_1), p_0 \Rightarrow (p_0 \wedge p_1), p_1 \Rightarrow (p_0 \wedge p_1), p_1\},$$
$$\mathtt{T}(\mathtt{B}_2) = \{(p_0 \vee p_1) \Rightarrow (p_0 \wedge p_1), p_1 \Rightarrow (p_0 \wedge p_1), p_1 \Rightarrow (p_0 \wedge p_1), p_0\}.$$

The truth assignments associated with these two branches are

$$v_{\mathtt{B}_1}(p_0) = \mathbf{T} \; v_{\mathtt{B}_1}(p_1) = \mathbf{F},$$
$$v_{\mathtt{B}_2}(p_0) = \mathbf{F} \; v_{\mathtt{B}_2}(p_1) = \mathbf{T}.$$

A truth assignment falsifies the sequent $(p_0 \vee p_1) \Rightarrow (p_0 \wedge p_1)$ if and only if it extends one of the previous partial truth assignments. □

We will now introduce a cut rule for sequents corresponding to the cut rule for tableaux.

Definition 3.5.31. The formal systems $\mathcal{F}^{\mathrm{seq,cut}}$ and $\mathcal{F}^{\mathrm{seq,cons,cut}}$ are obtained from $\mathcal{F}^{\mathrm{seq}}$ and $\mathcal{F}^{\mathrm{seq,cons}}$, respectively, by adding the following "cut" rule:

$$\frac{\Gamma, \varphi \Rightarrow \Gamma' \quad \Gamma \Rightarrow \Gamma', \varphi}{\Gamma \Rightarrow \Gamma'}\,\mathsf{R}_{\mathrm{cut}}$$

for all finite sets of formulas Γ, Γ' and formulas φ.

The formal systems $\mathcal{F}^{\mathrm{seq},\infty,\mathrm{cut}}$ and $\mathcal{F}^{\mathrm{seq},\infty,\mathrm{cons,cut}}$are the formal systems obtained from $\mathcal{F}^{\mathrm{seq},\infty}$ and $\mathcal{F}^{\mathrm{seq},\infty,\mathrm{cons}}$ by adding a cut rule for arbitrary sequents similar to the one introduced above. □

As with tableaux with cut, the formal systems introduced above are not analytical since the formula φ that appears in an instance

$$\frac{\Gamma, \varphi \Rightarrow \Gamma' \quad \Gamma \Rightarrow \Gamma', \varphi}{\Gamma \Rightarrow \Gamma'}\,\mathsf{R}_{\mathrm{cut}}$$

of the cut rule need not be a subformula of a formula in Γ or Γ'.

The parallelism between tableaux and deduction trees for sequents is maintained between tableaux with cut and deduction trees with cut for sequents.

Theorem 3.5.32. *For* $\mathtt{T} \in \mathcal{GDT}_{\mathcal{F}^{seq,\infty,cut}}$, *let* $\Phi^{cut}(\mathtt{T}) = \mathtt{sf} \circ \mathtt{T}$ *and for* $\mathtt{T} \in \mathcal{GDT}_{\mathcal{F}^{tabl,\infty,cut}}$, *let* $\Psi^{cut}(\mathtt{T}) = \mathtt{sqt} \circ \mathtt{T}$. *The mappings*

$$\Phi^{cut} : \mathcal{GDT}_{\mathcal{F}^{seq,\infty,cut}} \to \mathcal{GDT}_{\mathcal{F}^{tabl,\infty,cut}},$$
$$\Psi^{cut} : \mathcal{GDT}_{\mathcal{F}^{tabl,\infty,cut}} \longrightarrow \mathcal{GDT}_{\mathcal{F}^{seq,\infty,cut}}$$

are inverse bijections, and we have $\Phi^{cut}(\mathcal{PT}_{\mathcal{F}^{seq,\infty,cut}}) = \mathcal{PT}_{\mathcal{F}^{tabl,\infty,cut}}$ *and* $\Psi^{cut}(\mathcal{PT}_{\mathcal{F}^{tabl,\infty,cut}}) = \mathcal{PT}_{\mathcal{F}^{seq,\infty,cut}}$.

Further, if $\mathtt{T} \in \mathcal{GDT}_{\mathcal{F}^{seq,\infty,cons,cut}}$ ($\mathtt{T} \in \mathcal{PT}_{\mathcal{F}^{seq,\infty,cons,cut}}$), *then* $\Phi(\mathtt{T})$ *belongs to* $\mathcal{GDT}_{\mathcal{F}^{tabl,\infty,cons,cut}}$ ($\Phi(\mathtt{T})$ *belongs to* $\mathcal{PT}_{\mathcal{F}^{tabl,\infty,cons,cut}}$).

Similarly, if $\mathtt{T} \in \mathcal{GDT}_{\mathcal{F}^{tabl,\infty,cons,cut}}$ ($\mathtt{T} \in \mathcal{PT}_{\mathcal{F}^{tabl,\infty,cons,cut}}$), *then* $\Psi(\mathtt{T})$ *belongs to* $\mathcal{GDT}_{\mathcal{F}^{tabl,\infty,cons,cut}}$ ($\Psi(\mathtt{T})$ *belongs to* $\mathcal{PT}_{\mathcal{F}^{tabl,\infty,cons,cut}}$).

Proof. The argument expands the argument for Theorem 3.5.14 by the following observation. If

$$\frac{\kappa_0 \quad \kappa_1}{\kappa}$$

is an instance of the cut rule of $\mathcal{F}^{\mathrm{seq},\infty,\mathrm{cut}}$, then

$$\frac{\mathtt{sf}(\kappa_0) \quad \mathtt{sf}(\kappa_1)}{\mathtt{sf}(\kappa)}$$

is an instance of the cut rule of $\mathcal{F}^{\mathrm{tabl},\infty,\mathrm{cut}}$ and, if

$$\frac{\Delta \cup \{\mathbf{T}\varphi\} \quad \Delta \cup \{\mathbf{F}\varphi\}}{\Delta}$$

is an instance of the cut rule for $\mathcal{F}^{\mathrm{tabl},\infty,\mathrm{cut}}$, then

$$\frac{\mathtt{sqt}(\Delta \cup \{\mathbf{T}\varphi\}) \quad \mathtt{sqt}(\Delta \cup \{\mathbf{F}\varphi\})}{\mathtt{sqt}(\Delta)}$$

is an instance of the cut rule of $\mathcal{F}^{\mathrm{seq},\infty,\mathrm{cut}}$. □

Theorem 3.5.33 (Soundness of $\mathcal{F}^{seq,\infty,cut}$). *Every theorem of $\mathcal{F}^{seq,\infty,cut}$ is a valid sequent.*

Proof. The argument is similar to the argument of Theorem 3.5.15, using Φ^{cut} in place of Φ and Theorem 3.4.10 in place of Theorem 3.3.43. □

Corollary 3.5.34 (Soundness of Other Sequent Systems). *Each theorem of $\mathcal{F}^{seq,cut}$ and $\mathcal{F}^{seq,cons,cut}$ is a valid finite sequent. Further, every theorem of $\mathcal{F}^{seq,\infty,cons,cut}$ is a valid sequent.*

Proof. This statement follows directly from Theorem 3.5.33 because the formal system $\mathcal{F}^{\mathrm{seq},\infty,\mathrm{cut}}$ is an extension of all formal systems mentioned in the corollary. □

Theorem 3.5.35 (Completeness of Other Sequent Systems). *Every valid sequent is a theorem of $\mathcal{F}^{seq,\infty,cons,cut}$ and $\mathcal{F}^{seq,\infty,cut}$. Every valid finite sequent is a theorem of $\mathcal{F}^{seq,cons,cut}$ and $\mathcal{F}^{seq,cut}$.*

Proof. The argument follows immediately by observing that each of these formal systems extends existing complete formal systems by adding the cut rule. □

Theorem 3.5.36. *There is an effective, syntactic construction that starts with an $\mathcal{F}^{seq,\infty,cut}$-proof tree* `T` *for a sequent κ and produces an $\mathcal{F}^{seq,\infty,cut}$-proof tree* `T′` *for a finite sequent κ' that is dominated by κ.*

Proof. The argument for this theorem is similar to that for Theorem 3.5.20, using translations between tableaux with cut and proof trees with cut for sequents, and Theorem 3.4.8. □

Definition 3.5.37. We say that a general $\mathcal{F}^{\text{seq},\infty,\text{cut}}$-deduction tree `T` is *finished* if $\Phi^{\text{cut}}(\mathtt{T})$ is a strongly completed tableau with cut. ▯

The notion of finished branch of a general $\mathcal{F}^{\text{seq},\infty,\text{cut}}$-deduction tree is defined in exactly the same way as the corresponding notion for $\mathcal{F}^{\text{seq},\infty}$-deduction trees (see Definition 3.5.23.)

Theorem 3.5.38. *A general $\mathcal{F}^{seq,\infty,cut}$-deduction tree* `T` *is finished if and only if every branch of* `T` *is either finished or ends with a node labeled by an axiom of $\mathcal{F}^{seq,\infty,cut}$.*

Proof. The argument is similar to that of Theorem 3.5.24. □

Theorem 3.5.26 implies the existence of a finished $\mathcal{F}^{\text{seq,cut}}$-deduction tree for every finite sequent κ and a finished general $\mathcal{F}^{\text{seq},\infty,\text{cut}}$-deduction tree for every sequent κ.

Theorem 3.5.29 extends to finished general $\mathcal{F}^{\text{seq},\infty,\text{cut}}$-deduction trees with cut as follows.

Theorem 3.5.39. *Let* `T` *be a finished general $\mathcal{F}^{seq,\infty,cut}$-deduction tree for a sequent κ. For each finished branch* `B` *of* `T`*, let $v_{\mathtt{B}}$ be the partial truth assignment defined by*

$$v_{\mathtt{B}}(p) = \begin{cases} \mathbf{T} \text{ if } p \Rightarrow \text{ is dominated by } \mathtt{T}(\mathtt{B}) \\ \mathbf{F} \text{ if } \Rightarrow p \text{ is dominated by } \mathtt{T}(\mathtt{B}). \end{cases}$$

Then, a truth assignment v falsifies κ if and only if v extends some partial truth assignment $v_{\mathtt{B}}$, where `B` *is a finished branch of* `T`*.*

Proof. The argument is the same as the one for Theorem 3.5.29 except that Theorem 3.3.18 is replaced by Theorem 3.4.5. □

In the following theorem, we show how cut elimination for tableaux translates into cut elimination for sequents. This gives a syntactic transformation of a nonanalytical proof of validity of a sequent into an analytical one.

Theorem 3.5.40 (Cut Elimination for Sequents). *There is a syntactic transformation $\mathsf{CETS} : \mathcal{PT}_{\mathcal{F}^{seq,\infty,cut}} \longrightarrow \mathcal{PT}_{\mathcal{F}^{seq,\infty}}$ such that $\mathsf{CETS}(\mathtt{T})(\lambda)$ and $\mathtt{T}(\lambda)$ are the same sequent.*

Proof. Define $\mathsf{CETS} = \Psi \circ \mathsf{CET} \circ \Phi^{cut}$. Since all three mappings involved in this definition are syntactic transformations, then so is CETS. Let $\mathtt{T} \in \mathcal{PT}_{\mathcal{F}^{\mathrm{seq},\infty,\mathrm{cut}}}$. Define $\mathtt{T}' = \Phi^{cut}(\mathtt{T})$, $\mathtt{T}'' = \mathsf{CET}(\mathtt{T}')$, and $\mathtt{T}_1 = \Psi(\mathtt{T}'')$. We have $\mathtt{T}' \in \mathcal{PT}_{\mathcal{F}^{\mathrm{tabl},\infty,\mathrm{cut}}}$ and $\mathtt{T}'(\lambda) = \mathtt{sf}(\mathtt{T}(\lambda))$. Next, $\mathtt{T}'' \in \mathcal{PT}_{\mathcal{F}^{\mathrm{tabl},\infty}}$ and $\mathtt{T}''(\lambda) = \mathtt{T}'(\lambda)$. Finally, $\mathtt{T}_1 \in \mathcal{PT}_{\mathcal{F}^{\mathrm{seq},\infty}}$ and $\mathtt{T}_1(\lambda) = \mathtt{sqt}(\mathtt{T}''(\lambda)) = \mathtt{sqt}(\mathtt{sf}(\mathtt{T}(\lambda))) = \mathtt{T}(\lambda)$. □

The operation $\boxplus$ that we are about to introduce is used in the following two theorems. These, in turn, will be useful, in Section 3.7, for linking sequent proofs to natural deduction trees, which we study in Section 3.6.

Let $\mathtt{T} \in \mathcal{GDT}_{\mathcal{F}^{\mathrm{seq},\infty,\mathrm{cut}}}$ and let κ be a sequent. Define $\mathtt{T}'$ as the *lot* $(\mathtt{T}; \mathtt{T}(\lambda) \cup \{\kappa\})$. We will denote $\mathtt{T}'$ by $\mathtt{T} \boxplus \kappa$.

Theorem 3.5.41. *If $\mathtt{T} \in \mathcal{GDT}_{\mathcal{F}^{tabl,\infty,cut}}$ and $\Delta_0 \subseteq$ SPLFORM, then*

$$\Psi^{\mathrm{cut}}(\mathtt{T} \uplus \Delta_0) = \Psi^{\mathrm{cut}}(\mathtt{T}) \boxplus \mathtt{sqt}(\Delta_0).$$

Also, if $\mathtt{T} \in \mathcal{GDT}_{\mathcal{F}^{seq,\infty,cut}}$ and κ is a sequent, then

$$\Phi^{\mathrm{cut}}(\mathtt{T} \boxplus \kappa) = \Phi^{\mathrm{cut}}(\mathtt{T}) \uplus \mathtt{sf}(\kappa).$$

Proof. This is immediate from the definitions of Ψ^{cut} and Φ^{cut}. □

The following theorem can be easily proven directly; however, we will give a proof making use of the operations of Theorem 3.5.41.

Theorem 3.5.42. *If $\mathtt{T} \in \mathcal{GDT}_{\mathcal{F}}, \mathcal{DT}_{\mathcal{F}}, \mathcal{PT}_{\mathcal{F}}$ and κ is a sequent (finite sequent), then $\mathtt{T} \boxplus \kappa$ is in $\mathcal{GDT}_{\mathcal{F}}, \mathcal{DT}_{\mathcal{F}}, \mathcal{PT}_{\mathcal{F}}$, for $\mathcal{F} \in \{\mathcal{F}^{seq,\infty}, \mathcal{F}^{seq,\infty,cut}\}$ (for $\mathcal{F} \in \{\mathcal{F}^{seq}, \mathcal{F}^{seq,cut}\}$), respectively.*

Proof. Let $\mathtt{T} \in \mathcal{GDT}_{\mathcal{F}^{\mathrm{seq},\infty,\mathrm{cut}}}$ and let $\kappa \in \mathsf{SQT}$. Since $\Psi^{\mathrm{cut}}, \Phi^{\mathrm{cut}}$ and $\mathtt{sqt}, \mathtt{sf}$ are pairs of inverse bijections, we can write

$$\mathtt{T} \boxplus \kappa = \Psi^{\mathrm{cut}}(\Phi^{\mathrm{cut}}(\mathtt{T})) \boxplus \mathtt{sqt}(\mathtt{sf}(\kappa)).$$

By Theorem 3.5.41, we have

$$\mathtt{T} \boxplus \kappa = \Psi^{\mathrm{cut}}(\Phi^{\mathrm{cut}}(\mathtt{T}) \uplus \mathtt{sf}(\kappa)).$$

Since $\Phi^{\text{cut}}(\mathtt{T}) \in \mathcal{GDT}_{\mathcal{F}^{\text{tabl},\infty,\text{cut}}}$ (by Theorem 3.5.32), it follows that

$$\Phi^{\text{cut}}(\mathtt{T}) \uplus \mathtt{sf}(\kappa) \in \mathcal{GDT}_{\mathcal{F}^{\text{tabl},\infty,\text{cut}}},$$

which enables us to conclude that $\mathtt{T} \boxplus \kappa \in \mathcal{GDT}_{\mathcal{F}^{\text{seq},\infty,\text{cut}}}$. We leave the remaining cases to the reader. □

3.6 Natural Deduction

Natural Deduction was introduced by Gentzen as a formalism that reflects "as accurately as possible the actual logical reasoning involved in mathematical proofs" (see [16]).

Ironically, in view of the fact that natural deduction is meant to model common mathematical reasoning faithfully, the introduction and study of the system are rather cumbersome. On the other hand, this could be anticipated since common mathematical reasoning is a very powerful intellectual tool.

We begin by introducing a formal system whose set of axioms is empty and, therefore, has no theorems. We will use deduction trees of this system to formalize natural deduction processes.

Definition 3.6.1. $\mathcal{F}_{\text{nd}}$ is the formal system $(\text{PLFORM}, \emptyset, I)$ whose set of rules I is given in Figure 3.15, where φ, ψ, θ are arbitrary formulas in PLFORM.

If $\mathtt{T}$ is an $\mathcal{F}_{\text{nd}}$-deduction tree for φ, we will refer to the labels of the leaves of $\mathtt{T}$ as the *hypotheses* of $\mathtt{T}$ and to φ as the *conclusion* of $\mathtt{T}$. ▯

We will refer to the rules of the form R_{CI}, R_{CIl}, R_{CIr} as *C-introduction rules*; rules of the form R_{CE}, R_{CEl}, R_{CEr} are called *C-elimination rules.*

It is clear that the formal system $\mathcal{F}_{\text{nd}}$ is nonanalytical. For example, in an instance

$$\frac{(\varphi \vee \psi), \theta, \theta}{\theta}$$

of the rule $\mathsf{R}_{\vee E}$, φ and ψ do not have to be subformulas of θ.

Lemma 3.6.2. *For each instance of a rule of $\mathcal{F}_{nd}$, the set of premises logically implies the conclusion.*

$\wedge$-rules:

$$\frac{\varphi, \psi}{(\varphi \wedge \psi)}\mathsf{R}_{\wedge I} \qquad \frac{(\varphi \wedge \psi)}{\varphi}\mathsf{R}_{\wedge El} \qquad \frac{(\varphi \wedge \psi)}{\psi}\mathsf{R}_{\wedge Er}$$

$\vee$-rules:

$$\frac{\varphi}{(\varphi \vee \psi)}\mathsf{R}_{\vee Il} \qquad \frac{\psi}{(\varphi \vee \psi)}\mathsf{R}_{\vee Ir} \qquad \frac{(\varphi \vee \psi), \theta, \theta}{\theta}\mathsf{R}_{\vee E}$$

$\rightarrow$-rules:

$$\frac{\psi}{(\varphi \rightarrow \psi)}\mathsf{R}_{\rightarrow I} \qquad \frac{\varphi, (\varphi \rightarrow \psi)}{\psi}\mathsf{R}_{\rightarrow E}$$

$\leftrightarrow$-rules:

$$\frac{\psi, \varphi}{(\varphi \leftrightarrow \psi)}\mathsf{R}_{\leftrightarrow I} \qquad \frac{\varphi, (\varphi \leftrightarrow \psi)}{\psi}\mathsf{R}_{\leftrightarrow El} \qquad \frac{\psi, (\varphi \leftrightarrow \psi)}{\varphi}\mathsf{R}_{\leftrightarrow Er}$$

$\neg$-rules:

$$\frac{\psi, (\neg\psi)}{(\neg\varphi)}\mathsf{R}_{\neg I} \qquad \frac{\psi, (\neg\psi)}{\varphi}\mathsf{R}_{\neg E}\,.$$

Fig. 3.15. The set of rules of $\mathcal{F}_{\text{nd}}$.

Proof. The argument is immediate by inspecting each rule. □

It follows from Lemma 3.6.2 that for any $\mathcal{F}_{\text{nd}}$-deduction tree, the hypotheses of the tree logically imply the conclusion of the tree. In certain cases, however, we can show that some of the hypotheses can be eliminated. For example, let $\mathtt{T}$ be an $\mathcal{F}_{\text{nd}}$-deduction tree for ψ whose set of hypotheses is Γ. Then, using the $\rightarrow$-introduction rule, we obtain the $\mathcal{F}_{\text{nd}}$-deduction tree $\mathtt{T}' = (\mathtt{T}; (\varphi \rightarrow \psi))$ for $(\varphi \rightarrow \psi)$ whose set of hypotheses is still Γ. As just noted, we have $\Gamma \models (\varphi \rightarrow \psi)$, but, in fact, we can make the stronger statement that $\Gamma - \{\varphi\} \models (\varphi \rightarrow \psi)$. Indeed, let v be a truth assignment that satisfies $\Gamma - \{\varphi\}$. If $v(\varphi) = \mathbf{F}$, then $v((\varphi \rightarrow \psi)) = \mathbf{T}$; if $v(\varphi) = \mathbf{T}$, then v satisfies Γ, so again $v((\varphi \rightarrow \psi)) = \mathbf{T}$. This observation will be formalized by the syntactic device of canceling or discharging hypotheses. The discharging of hypotheses of a deduction tree amounts to marking the leaves of the tree labeled with hypotheses that are no longer required for logical implication of the conclusion. In the case of the $\rightarrow$-introduction rule, the discharging of hypotheses is a formal equivalent of the most common technique for proving an implication $(\varphi \rightarrow \psi)$. Namely, if we can prove ψ using φ and some other hypotheses, then we conclude that $(\varphi \rightarrow \psi)$ follows from the other hypotheses alone.

Example 3.6.3. Let $\mathtt{T}_0, \mathtt{T}_1, \mathtt{T}_2$ be $\mathcal{F}_{\mathrm{nd}}$-deduction trees with $\mathtt{T}_0(\lambda) = (\varphi \vee \psi)$ and $\mathtt{T}_1(\lambda) = \mathtt{T}_2(\lambda) = \theta$ whose sets of hypotheses are Γ_0, Γ_1, and Γ_2, respectively. Using the $\vee$-elimination rule, we obtain the $\mathcal{F}_{\mathrm{nd}}$-deduction tree $\mathtt{T} = (\mathtt{T}_0, \mathtt{T}_1, \mathtt{T}_2; \theta)$. The set of hypotheses of $\mathtt{T}$ is $\Gamma_0 \cup \Gamma_1 \cup \Gamma_2$, so we obtain $\Gamma_0 \cup \Gamma_1 \cup \Gamma_2 \models \theta$. Note that a stronger statement holds, namely,

$$\Gamma_0 \cup (\Gamma_1 - \{\varphi\}) \cup (\Gamma_2 - \{\psi\}) \models \theta.$$

Indeed, if v is a truth assignment that satisfies $\Gamma_0 \cup (\Gamma_1 - \{\varphi\}) \cup (\Gamma_2 - \{\psi\})$, then at least one of the equalities $v(\varphi) = \mathbf{T}$ or $v(\psi) = \mathbf{T}$ holds. If $v(\varphi) = \mathbf{T}$, then, since v satisfies $\Gamma_1 - \{\varphi\}$, it follows that it satisfies Γ_1 and, therefore, it satisfies θ. A similar argument is applicable if $v(\psi) = \mathbf{T}$, so in any case, $v(\theta) = \mathbf{T}$. This observation allows us to discharge the leaves of $\mathtt{T}_1$ labeled by φ and the leaves of $\mathtt{T}_2$ labeled by ψ.

Proof by cases of θ from some hypotheses Γ, in common mathematical practice, amounts to proving θ from φ and other hypotheses Γ_1, proving θ from ψ and other hypotheses Γ_2, and then proving $(\varphi \vee \psi)$, where Γ consists of Γ_1, Γ_2 and the hypotheses used to prove $(\varphi \vee \psi)$. This corresponds in broad lines to the discharging process discussed above. $\square$

Example 3.6.4. The usual way to show the equivalence of φ and ψ is to show ψ starting from φ and then show φ starting from ψ. We can formalize this standard type of argument in $\mathcal{F}_{\mathrm{nd}}$ as follows. Let $\mathtt{T}'$ be an $\mathcal{F}_{\mathrm{nd}}$-deduction tree for ψ with hypotheses Γ' and let $\mathtt{T}''$ be an $\mathcal{F}_{\mathrm{nd}}$-deduction tree for φ with hypotheses Γ''. Then, by the $\leftrightarrow$-introduction rule, we obtain the $\mathcal{F}_{\mathrm{nd}}$-deduction tree $\mathtt{T} = (\mathtt{T}', \mathtt{T}''; (\varphi \leftrightarrow \psi))$. The existence of $\mathtt{T}$ shows that $\Gamma' \cup \Gamma'' \models (\varphi \leftrightarrow \psi)$, but, in fact, we have $(\Gamma' - \{\varphi\}) \cup (\Gamma'' - \{\psi\}) \models (\varphi \leftrightarrow \psi)$, as the reader can easily verify. This means that we can discharge the hypothesis φ from Γ' (if $\varphi \in \Gamma'$) and the hypothesis ψ from Γ'' (if $\psi \in \Gamma''$). $\square$

Example 3.6.5. A proof by contradiction of φ in mathematics consists of assuming $(\neg\varphi)$ and proving both ψ and $(\neg\psi)$ for some formula ψ. This allows us to conclude φ. Once again, following the pattern of the previous examples, let $\mathtt{T}', \mathtt{T}''$ be $\mathcal{F}_{\text{nd}}$-deduction trees for ψ and $(\neg\psi)$ which have Γ', Γ'' as their sets of hypotheses, respectively. Using the $\neg$-elimination rule, we obtain the $\mathcal{F}_{\text{nd}}$-deduction tree $\mathtt{T} = (\mathtt{T}', \mathtt{T}''; \varphi)$ which shows that $\Gamma' \cup \Gamma'' \models \varphi$. Note however that the stronger statement $(\Gamma' - \{(\neg\varphi)\}) \cup (\Gamma'' - \{(\neg\varphi)\}) \models \varphi$ holds. This allows us to discharge $(\neg\varphi)$ from both Γ' and Γ''.

Similarly, the hypothesis φ can be discharged in any proof by contradiction of $(\neg\varphi)$ that uses the $\neg$-introduction rule. $\square$

Definition 3.6.6. A *marked $\mathcal{F}_{nd}$-deduction tree* is a pair $(\mathtt{T}, M)$, where $\mathtt{T}$ is an $\mathcal{F}_{\text{nd}}$-deduction tree and $M \subseteq \text{LEAVES}(\mathtt{T})$. The leaves that belong to the set M are said to be *canceled.*

The set of *uncanceled hypotheses* of a marked $\mathcal{F}_{\text{nd}}$-deduction tree $(\mathtt{T}, M)$ is the set

$$\mathcal{UNC}(\mathtt{T}, M) = \{\mathtt{T}(q) \mid q \in \text{LEAVES}(\mathtt{T}) - M\}. \qquad \square$$

The notation introduced after Theorem 1.7.18 is used in the following definition. Also, using Definition 1.7.10, if $\mathcal{T} = (\mathtt{T}, M)$ is a marked $\mathcal{F}_{\text{nd}}$-deduction tree and φ is a formula, we denote by $L_\varphi(\mathcal{T})$ the marked $\mathcal{F}_{\text{nd}}$-deduction tree $(\mathtt{T}, M \cup \text{LEAVES}_\varphi(\mathtt{T}))$.

Definition 3.6.7. The set NDT of *natural deduction trees* is the set of marked $\mathcal{F}_{\text{nd}}$-deduction trees given inductively by the following:

(1) If $\mathtt{T}$ is a one-node $\mathcal{F}_{\text{nd}}$-deduction tree, then $(\mathtt{T}, \emptyset) \in \text{NDT}$.
(2) ($\rightarrow$-introduction) If $\mathcal{T} = (\mathtt{T}, M) \in \text{NDT}$, $\mathtt{T}(\lambda) = \psi$, and φ is a formula, then

$$(L_\varphi(\mathcal{T}); (\varphi \rightarrow \psi)) \in \text{NDT}.$$

(3) ($\vee$-elimination) If $\mathcal{T}_i = (\mathtt{T}_i, M_i) \in \text{NDT}$ for $0 \leq i \leq 2$ and $\mathtt{T}_0(\lambda) = (\varphi \vee \psi)$, $\mathtt{T}_1(\lambda) = \mathtt{T}_2(\lambda) = \theta$, then

$$(\mathcal{T}_0, L_\varphi(\mathcal{T}_1), L_\psi(\mathcal{T}_2); \theta)$$

belongs to NDT.

(4) ($\leftrightarrow$-introduction) If $\mathcal{T}_0 = (\mathtt{T}_0, M_0)$ and $\mathcal{T}_1 = (\mathtt{T}_1, M_1)$ belong to NDT, $\mathtt{T}_0(\lambda) = \psi$, and $\mathtt{T}_1(\lambda) = \varphi$, then

$$(L_\varphi(\mathcal{T}_0), L_\psi(\mathcal{T}_1); (\varphi \leftrightarrow \psi))$$

belongs to NDT.

(5) ($\neg$-introduction) If $\mathcal{T}_0 = (\mathtt{T}_0, M_0)$ and $\mathcal{T}_1 = (\mathtt{T}_1, M_1)$ belong to NDT, $\mathtt{T}_0(\lambda) = \psi$, $\mathtt{T}_1(\lambda) = (\neg\psi)$ and $\varphi \in$ PLFORM, then

$$(L_\varphi(\mathcal{T}_0), L_\varphi(\mathcal{T}_1); (\neg\varphi)) \in \text{NDT}.$$

(6) ($\neg$-elimination) If $\mathcal{T}_0 = (\mathtt{T}_0, M_0)$ and $\mathcal{T}_1 = (\mathtt{T}_1, M_1)$ belong NDT, $\mathtt{T}_0(\lambda) = \psi$, $\mathtt{T}_1(\lambda) = (\neg\psi)$ and $\varphi \in$ PLFORM, then

$$(L_{(\neg\varphi)}(\mathcal{T}_0), L_{(\neg\varphi)}(\mathcal{T}_1)); \varphi) \in \text{NDT}.$$

(7) If $\dfrac{\alpha}{\beta}$ is an instance of one of the remaining rules that have one hypothesis and $\mathcal{T} = (\mathtt{T}, M) \in$ NDT with $\mathtt{T}(\lambda) = \alpha$, then $(\mathcal{T}; \beta) \in$ NDT.

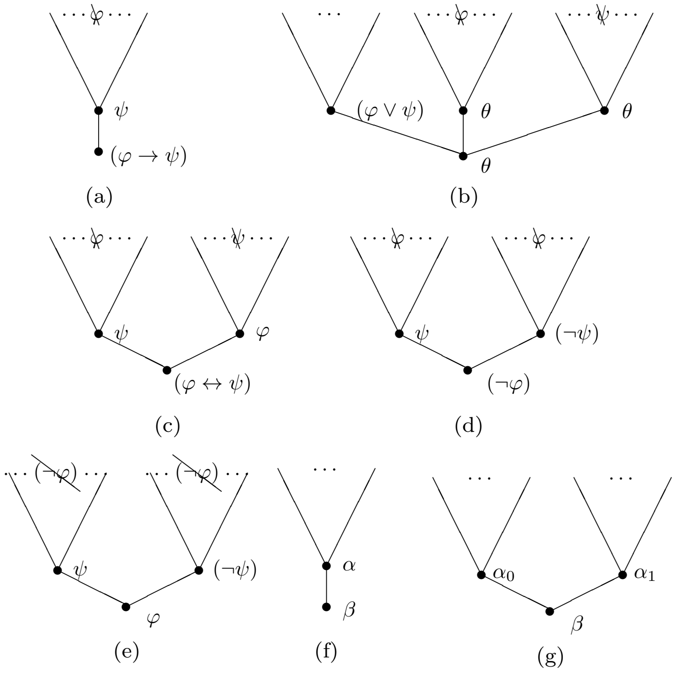
(8) If $\dfrac{\alpha_0, \alpha_1}{\beta}$ is an instance of one of the remaining rules that have two hypotheses, and $\mathcal{T}_0 = (\mathtt{T}_0, M_0)$ and $\mathcal{T}_1 = (\mathtt{T}_1, M_1)$ belong to NDT with $\mathtt{T}_0(\lambda) = \alpha_0$, and $\mathtt{T}_1(\lambda) = \alpha_1$, then $(\mathcal{T}_0, \mathcal{T}_1; \beta) \in$ NDT. $\square$

Figure 3.16 illustrates the inductive parts of Definition 3.6.7. Parts (a)–(e) show the application of the $\rightarrow$-introduction, $\vee$-elimination, $\leftrightarrow$-introduction, $\neg$-introduction, and $\neg$-elimination rules, respectively. Parts (f) and (g) show the application of the remaining parts that have one or two hypotheses, respectively.

Definition 3.6.8. Let Γ be a set of formulas and let φ be a formula. We write $\Gamma \overset{\bullet}{\vdash}_{\text{nd}} \varphi$ if there is a natural deduction tree $(\mathtt{T}, M)$ such that $\mathtt{T}(\lambda) = \varphi$ and $\mathcal{UNC}(\mathtt{T}, M) \subseteq \Gamma$. $\square$

We included the $\neg$-introduction rule in order to have an introduction and an elimination rule for each of the five connectives; however, as we show in Supplement 70, we can drop this rule without affecting the relation $\overset{\bullet}{\vdash}_{\text{nd}}$.

Natural deduction is not an analytical formalism since if $\Gamma \overset{\bullet}{\vdash}_{\text{nd}} \varphi$ is shown by the natural deduction tree $(\mathtt{T}, M)$, then formulas can



$\cdots\cancel{\varphi}\cdots$
ψ
$(\varphi \to \psi)$
(a)
$\cdots$
$(\varphi \vee \psi)$
$\cdots\cancel{\varphi}\cdots$
θ
$\cdots\cancel{\psi}\cdots$
θ
θ
(b)
$\cdots\cancel{\varphi}\cdots$
ψ
$\cdots\cancel{\psi}\cdots$
φ
$(\varphi \leftrightarrow \psi)$
(c)
$\cdots\cancel{\varphi}\cdots$
ψ
$\cdots\cancel{\varphi}\cdots$
$(\neg\psi)$
$(\neg\varphi)$
(d)
$\cdots\cancel{(\neg\varphi)}\cdots$
ψ
$\cdots\cancel{(\neg\varphi)}\cdots$
$(\neg\psi)$
φ
(e)
$\cdots$
α
β
(f)
$\cdots$
α_0
$\cdots$
α_1
β
(g)

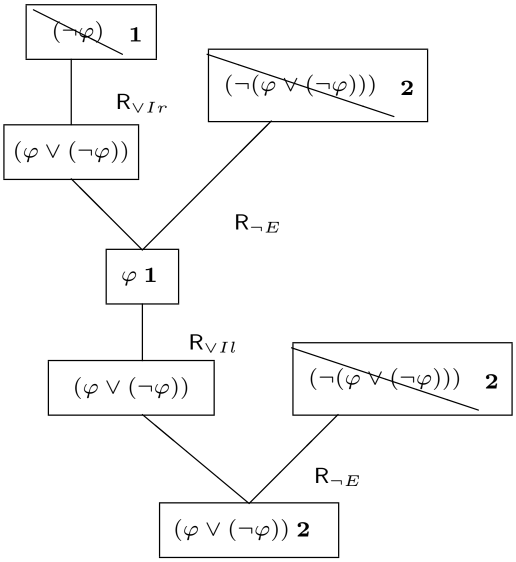
Fig. 3.16. Building natural deduction trees.

appear in T without being subformulas or negated subformulas of formulas in $\Gamma \cup \{\varphi\}$.

When drawing a natural deduction tree, we employ the following graphical conventions:

- The name of the rule applied is put near each interior node.
- A canceled leaf is indicated by a bar.
- Whenever a node is generated using a rule that causes one or more leaves to be canceled, we place a numerical mark inside that node and also inside all the leaves that have been canceled by the application of the rule.

Example 3.6.9. Figure 3.17 contains a natural deduction tree for the formula $(\varphi \vee (\neg\varphi))$. Since all the leaves of this tree are canceled, we have $\emptyset \overset{\bullet}{\vdash}_{\mathrm{nd}} (\varphi \vee (\neg\varphi))$. ∎



$(\neg\varphi)$ **1**
$\mathsf{R}_{\vee Ir}$
$(\neg(\varphi \vee (\neg\varphi)))$ **2**
$(\varphi \vee (\neg\varphi))$
$\mathsf{R}_{\neg E}$
φ **1**
$\mathsf{R}_{\vee Il}$
$(\varphi \vee (\neg\varphi))$
$(\neg(\varphi \vee (\neg\varphi)))$ **2**
$\mathsf{R}_{\neg E}$
$(\varphi \vee (\neg\varphi))$ **2**

Fig. 3.17. Natural deduction tree for $(\varphi \vee (\neg\varphi))$.

Example 3.6.10. In Figure 3.18, we give a natural deduction tree for the formula $((\varphi \vee (\alpha \wedge \beta)) \to ((\varphi \vee \alpha) \wedge (\varphi \vee \beta)))$ which shows that $\emptyset \overset{\bullet}{\vdash}_{\text{nd}} ((\varphi \vee (\alpha \wedge \beta)) \to ((\varphi \vee \alpha) \wedge (\varphi \vee \beta)))$. $\square$

Theorem 3.6.11 (Soundness Theorem for Natural Deduction). *Let Γ be a set of formulas and let θ be a formula. If $\Gamma \overset{\bullet}{\vdash}_{nd} \theta$, then $\Gamma \models \theta$.*

Proof. We must show that if $(\mathtt{T}, M)$ is a natural deduction tree such that $\mathcal{UNC}(\mathtt{T}, M) \subseteq \Gamma$ and $\mathtt{T}(\lambda) = \theta$, then $\Gamma \models \theta$. We proceed by induction on the definition of natural deduction trees (Definition 3.6.7).

For the basis step, $\mathtt{T}$ is a one-node $\mathcal{F}_{\text{nd}}$-deduction tree and $M = \emptyset$. Since $\mathcal{UNC}(\mathtt{T}, \emptyset) = \{\mathtt{T}(\lambda)\}$, the result is immediate.

Suppose that $(\mathtt{T}, M)$ is obtained from $(\mathtt{T}_0, M_0)$ using the $\to$-introduction rule (Case 2 of Definition 3.6.7). Then, there are some formulas φ, ψ such that $\theta = \mathtt{T}(\lambda) = (\varphi \to \psi)$ and $\mathtt{T}_0(\lambda) = \psi$. We have $\mathcal{UNC}(\mathtt{T}_0, M_0) \subseteq \Gamma \cup \{\varphi\}$ because $\mathcal{UNC}(\mathtt{T}, M) = \mathcal{UNC}(\mathtt{T}_0, M_0) - \{\varphi\} \subseteq \Gamma$. By the inductive hypothesis, we have $\Gamma \cup \{\varphi\} \models \psi$, so, $\Gamma \models (\varphi \to \psi)$ by the third part of Theorem 2.3.17.

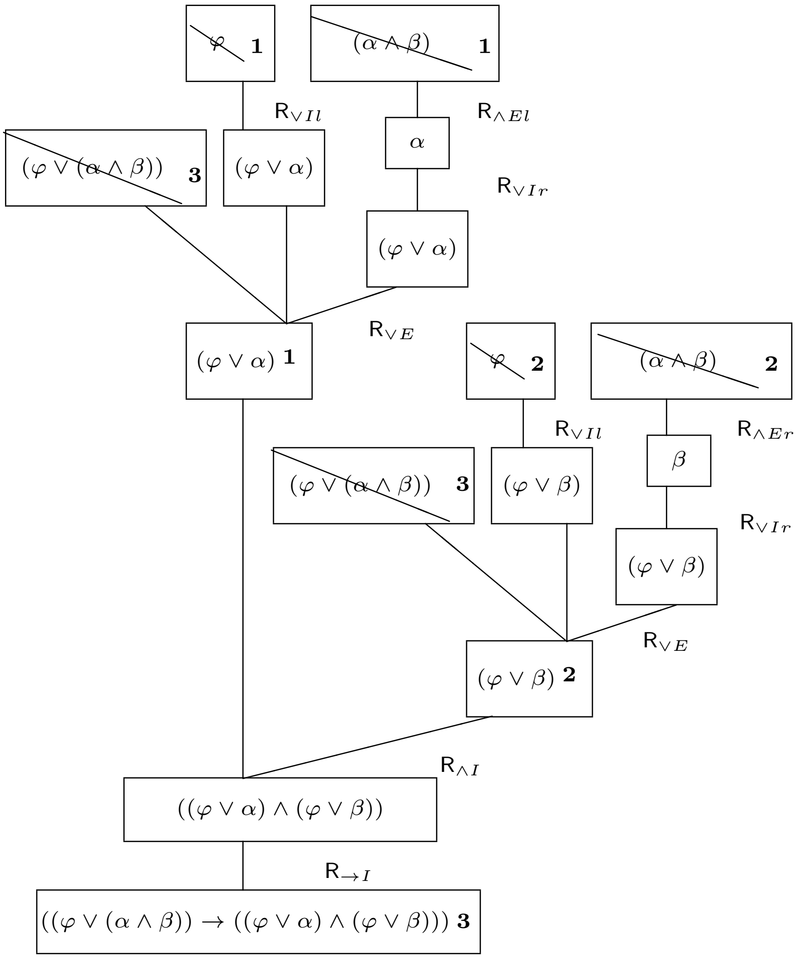


Fig. 3.18. Natural deduction tree.

Assume now that $(\mathtt{T}, M)$ is obtained from $(\mathtt{T}_0, M_0)$, $(\mathtt{T}_1, M_1)$, $(\mathtt{T}_2, M_2)$ using the $\vee$-elimination rule. Then, for some formulas φ, ψ, we have $\mathtt{T}_0(\lambda) = (\varphi \vee \psi)$ and $\mathtt{T}_1(\lambda) = \mathtt{T}_2(\lambda) = \theta$. Since

$$\begin{aligned}\mathcal{UNC}(\mathtt{T}, M) &= \mathcal{UNC}(\mathtt{T}_0, M_0) \cup (\mathcal{UNC}(\mathtt{T}_1, M_1) - \{\varphi\}) \\ &\quad \cup (\mathcal{UNC}(\mathtt{T}_2, M_2) - \{\psi\}) \subseteq \Gamma,\end{aligned}$$

it follows that

$$\begin{aligned}\mathcal{UNC}(\mathtt{T}_0, M_0) &\subseteq \Gamma,\\ \mathcal{UNC}(\mathtt{T}_1, M_1) &\subseteq \Gamma \cup \{\varphi\},\\ \mathcal{UNC}(\mathtt{T}_2, M_2) &\subseteq \Gamma \cup \{\psi\}.\end{aligned}$$

By the inductive hypothesis, we have $\Gamma \models (\varphi \vee \psi), \Gamma \cup \{\varphi\} \models \theta$, and $\Gamma \cup \{\psi\} \models \theta$. This allows us to conclude that $\Gamma \models \theta$. The reader should observe the close similarity between this argument and Example 3.6.3. Cases 4–6 of Definition 3.6.7 can be dealt with in a manner similar to Examples 3.6.4 and 3.6.5.

The last two cases of Definition 3.6.7 are handled by a direct argument using Lemma 3.6.2. □

Theorem 3.6.12 (Completeness Theorem for Natural Deduction). *Let* Γ *be a set of formulas and let* θ *be a formula. If* $\Gamma \models \theta$*, then* $\Gamma \overset{\bullet}{\vdash}_{nd} \theta$*.*

Proof. If $\Gamma \models \theta$, then, by the completeness of $\mathcal{HF}_\Gamma$ (Theorem 3.2.21), we have $\Gamma \vdash_{\mathcal{HF}} \theta$. Therefore, it suffices to prove by induction on the theorems of $\mathcal{HF}_\Gamma$ that if $\Gamma \vdash_{\mathcal{HF}} \theta$, then $\Gamma \overset{\bullet}{\vdash}_{\mathrm{nd}} \theta$. In Figures 3.19–3.25, we show that for every axiom θ of $\mathcal{HF}$, $\emptyset \overset{\bullet}{\vdash}_{\mathrm{nd}} \theta$ and, therefore, $\Gamma \overset{\bullet}{\vdash}_{\mathrm{nd}} \theta$. If $\theta \in \Gamma$, the existence of the one-node natural deduction tree $(\mathtt{T}, \emptyset)$ such that $\mathtt{T}(\lambda) = \theta$ shows that $\Gamma \overset{\bullet}{\vdash}_{\mathrm{nd}} \theta$.

Suppose now that $(\mathtt{T}_0, M_0)$ and $(\mathtt{T}_1, M_1)$ are natural deduction trees for φ and $(\varphi \to \psi)$, respectively, with $\mathcal{UNC}(\mathtt{T}_0, M_0)$, $\mathcal{UNC}(\mathtt{T}_1, M_1) \subseteq \Gamma$. Then, by the eighth rule given in Definition 3.6.7,

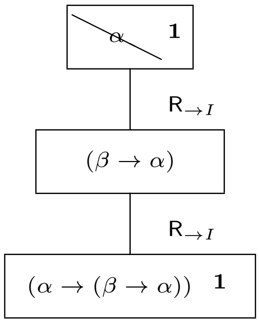
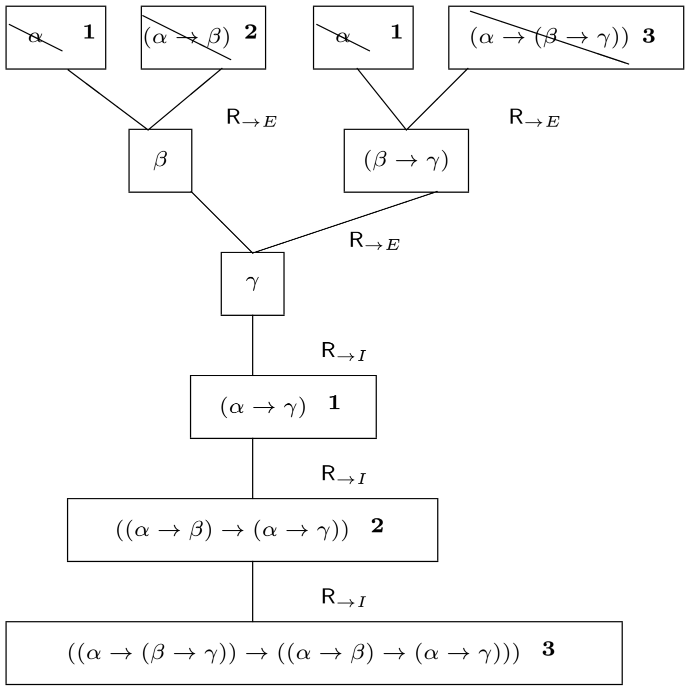


Fig. 3.19. Natural deduction tree for $(\alpha \to (\beta \to \alpha))$.



$\cancel{\alpha}$ **1**
$\cancel{(\alpha \to \beta)}$ **2**
$\cancel{\alpha}$ **1**
$\cancel{(\alpha \to (\beta \to \gamma))}$ **3**
$\mathsf{R}_{\to E}$
$\mathsf{R}_{\to E}$
β
$(\beta \to \gamma)$
$\mathsf{R}_{\to E}$
γ
$\mathsf{R}_{\to I}$
$(\alpha \to \gamma)$ **1**
$\mathsf{R}_{\to I}$
$((\alpha \to \beta) \to (\alpha \to \gamma))$ **2**
$\mathsf{R}_{\to I}$
$((\alpha \to (\beta \to \gamma)) \to ((\alpha \to \beta) \to (\alpha \to \gamma)))$ **3**

Fig. 3.20. Natural deduction tree for $((\alpha \to (\beta \to \gamma)) \to ((\alpha \to \beta) \to (\alpha \to \gamma)))$.

$(\mathtt{T}, M) = ((\mathtt{T}_0, M_0), (\mathtt{T}_1, M_1); \psi)$ is a natural deduction tree for ψ such that $\mathcal{UNC}(\mathtt{T}, M) \subseteq \Gamma$. Therefore, $\Gamma \overset{\bullet}{\vdash}_{\text{nd}} \psi$. □

Theorem 3.6.13. *There is an effective, syntactic algorithm that, starting from a proof in $\mathcal{HF}_\Gamma$ of a formula φ, yields a natural deduction tree $(\mathtt{T}, M)$ such that $\mathtt{T}(\lambda) = \varphi$ and $\mathcal{UNC}(\mathtt{T}, M) \subseteq \Gamma$.*

Proof. This was shown in the proof of the Completeness Theorem for Natural Deduction. □

Natural deduction, as we have formulated it, combines a formal system with an annotation of the proof trees of the system. As we shall show, it is possible to present natural deduction purely as a

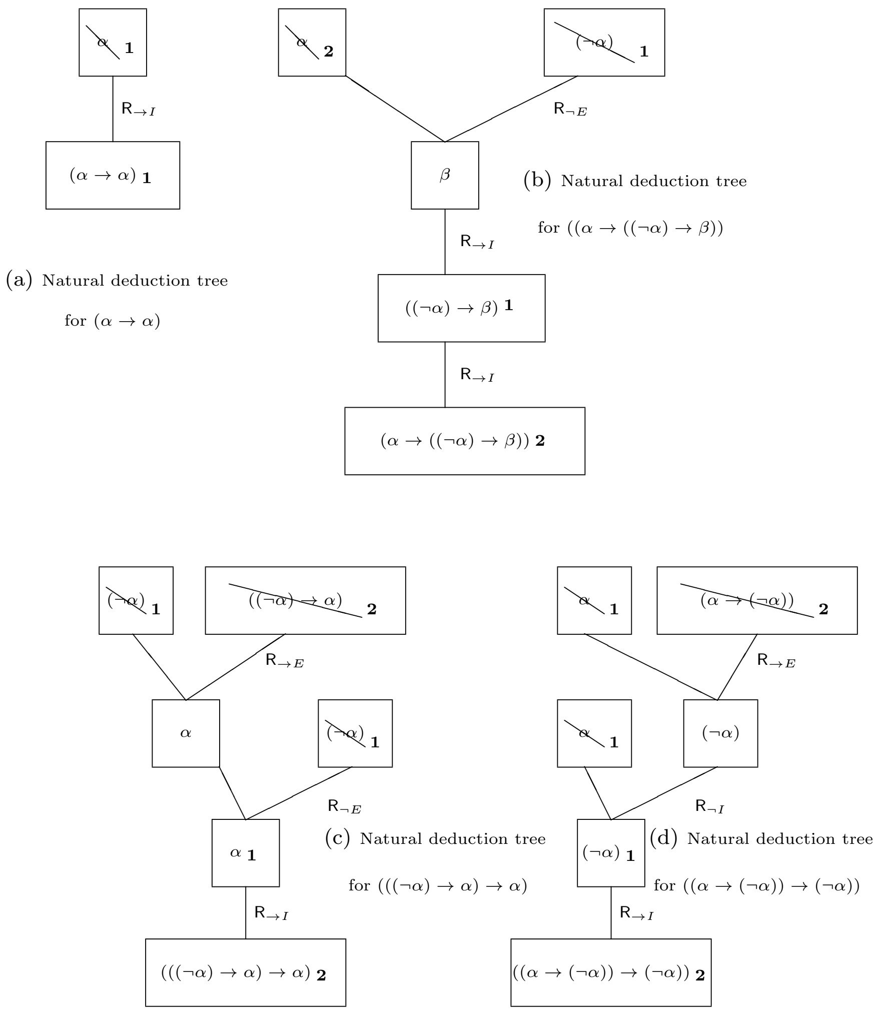
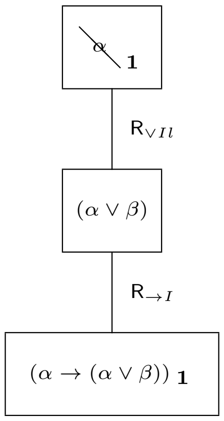

$\cancel{\alpha}$ 1
$\mathsf{R}_{\to I}$
$(\alpha \to \alpha)$ 1
(a) Natural deduction tree for $(\alpha \to \alpha)$
$\cancel{\alpha}$ 2
$\cancel{(\neg\alpha)}$ 1
$\mathsf{R}_{\neg E}$
β
(b) Natural deduction tree for $((\alpha \to ((\neg\alpha) \to \beta))$
$\mathsf{R}_{\to I}$
$((\neg\alpha) \to \beta)$ 1
$\mathsf{R}_{\to I}$
$(\alpha \to ((\neg\alpha) \to \beta))$ 2
$\cancel{(\neg\alpha)}$ 1
$\cancel{((\neg\alpha) \to \alpha)}$ 2
$\mathsf{R}_{\to E}$
α
$\cancel{(\neg\alpha)}$ 1
$\mathsf{R}_{\neg E}$
α 1
(c) Natural deduction tree for $(((\neg\alpha) \to \alpha) \to \alpha)$
$\mathsf{R}_{\to I}$
$(((\neg\alpha) \to \alpha) \to \alpha)$ 2
$\cancel{\alpha}$ 1
$\cancel{(\alpha \to (\neg\alpha))}$ 2
$\mathsf{R}_{\to E}$
$\cancel{\alpha}$ 1
$(\neg\alpha)$
$\mathsf{R}_{\neg I}$
$(\neg\alpha)$ 1
(d) Natural deduction tree for $((\alpha \to (\neg\alpha)) \to (\neg\alpha))$
$\mathsf{R}_{\to I}$
$((\alpha \to (\neg\alpha)) \to (\neg\alpha))$ 2

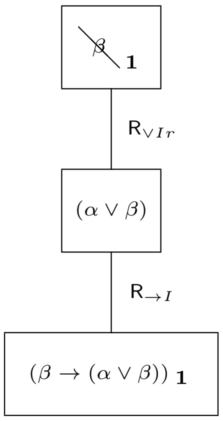
Fig. 3.21. Natural deduction trees for Axioms (3)–(6).

formal system (called $\mathcal{ND}$); however, this alternative approach is less successful in capturing the spirit of ordinary mathematical reasoning.

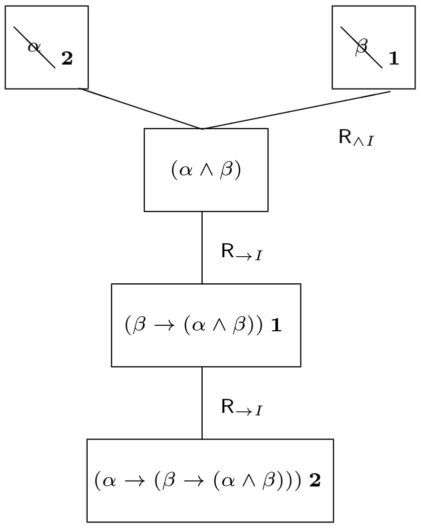
Definition 3.6.14. The set of objects of $\mathcal{ND}$ is $\mathcal{P}(\text{PLFORM}) \times \text{PLFORM}$; the set of axioms is $A = \{(\Gamma, \varphi) \mid \varphi \in \Gamma\}$.


$\cancel{\alpha}$ **1**
$\mathsf{R}_{\vee Il}$
$(\alpha \vee \beta)$
$\mathsf{R}_{\rightarrow I}$
$(\alpha \rightarrow (\alpha \vee \beta))$ **1**

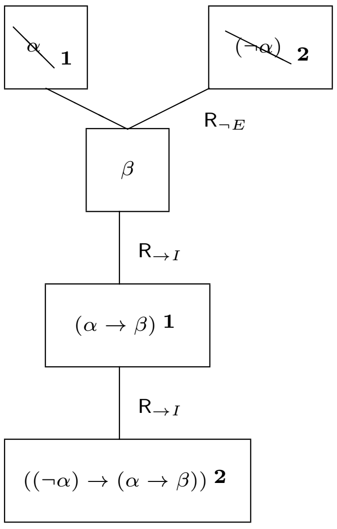
(a) Natural deduction tree for $(\alpha \rightarrow (\alpha \vee \beta))$


$\cancel{\beta}$ **1**
$\mathsf{R}_{\vee Ir}$
$(\alpha \vee \beta)$
$\mathsf{R}_{\rightarrow I}$
$(\beta \rightarrow (\alpha \vee \beta))$ **1**

(b) Natural deduction tree for $(\beta \rightarrow (\alpha \vee \beta))$

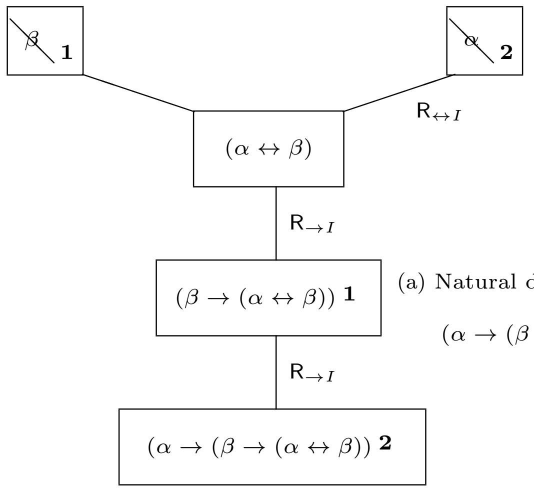

$\cancel{\alpha}$ **2**
$\cancel{\beta}$ **1**
$\mathsf{R}_{\wedge I}$
$(\alpha \wedge \beta)$
$\mathsf{R}_{\rightarrow I}$
$(\beta \rightarrow (\alpha \wedge \beta))$ **1**
$\mathsf{R}_{\rightarrow I}$
$(\alpha \rightarrow (\beta \rightarrow (\alpha \wedge \beta)))$ **2**

(c) Natural deduction tree for $(\alpha \rightarrow (\beta \rightarrow (\alpha \wedge \beta)))$


$\cancel{\alpha}$ **1**
$\cancel{(\neg\alpha)}$ **2**
$\mathsf{R}_{\neg E}$
β
$\mathsf{R}_{\rightarrow I}$
$(\alpha \rightarrow \beta)$ **1**
$\mathsf{R}_{\rightarrow I}$
$((\neg\alpha) \rightarrow (\alpha \rightarrow \beta))$ **2**

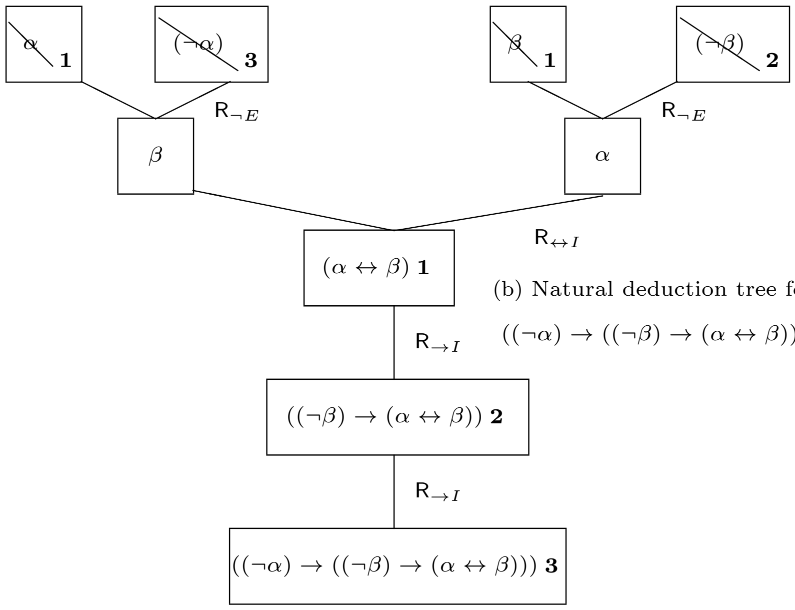
(d) Natural deduction tree $((\neg\alpha) \rightarrow (\alpha \rightarrow \beta))$

Fig. 3.22. Natural deduction trees for Axioms (7)–(10).


$\cancel{\beta}$ **1**
$\cancel{\alpha}$ **2**
$\mathsf{R}_{\leftrightarrow I}$
$(\alpha \leftrightarrow \beta)$
$\mathsf{R}_{\to I}$
$(\beta \to (\alpha \leftrightarrow \beta))$ **1**
$\mathsf{R}_{\to I}$
$(\alpha \to (\beta \to (\alpha \leftrightarrow \beta))$ **2**

(a) Natural deduction tree for

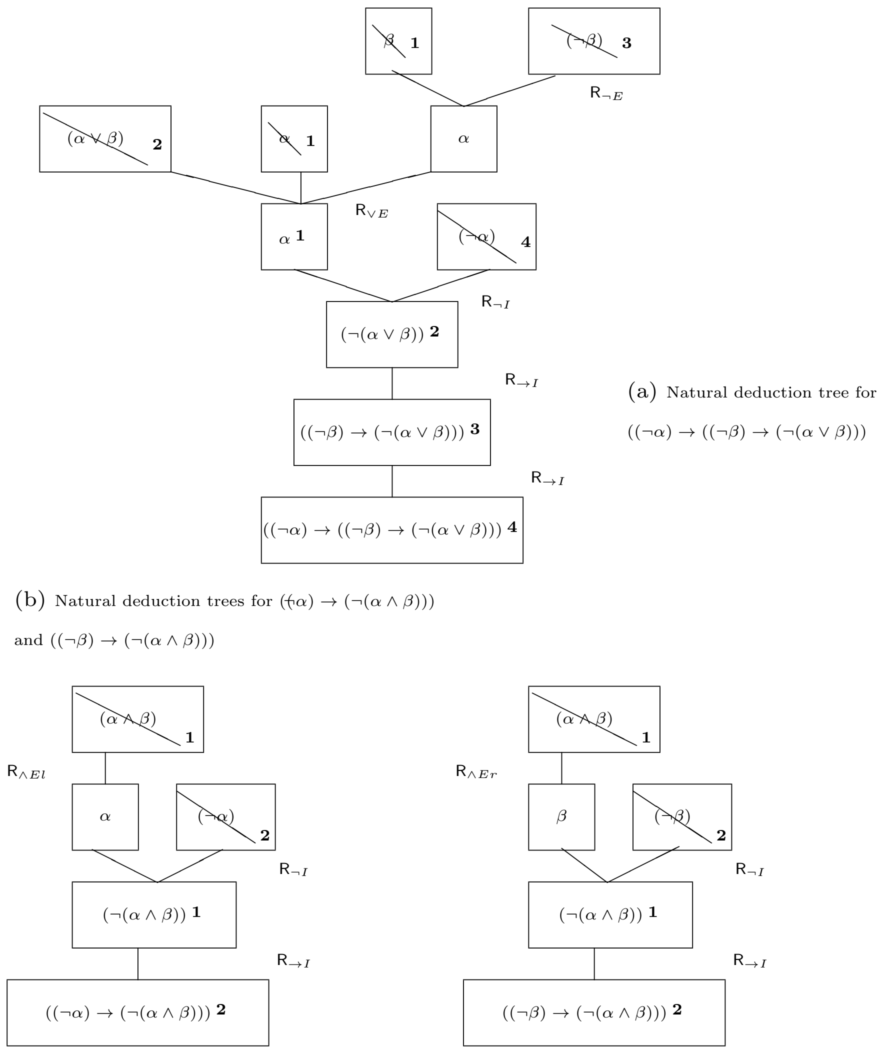
$(\alpha \to (\beta \to (\alpha \leftrightarrow \beta)))$


$\cancel{\alpha}$ **1**
$\cancel{(\neg\alpha)}$ **3**
$\cancel{\beta}$ **1**
$\cancel{(\neg\beta)}$ **2**
$\mathsf{R}_{\neg E}$
$\mathsf{R}_{\neg E}$
β
α
$\mathsf{R}_{\leftrightarrow I}$
$(\alpha \leftrightarrow \beta)$ **1**
$\mathsf{R}_{\to I}$
$((\neg\beta) \to (\alpha \leftrightarrow \beta))$ **2**
$\mathsf{R}_{\to I}$
$((\neg\alpha) \to ((\neg\beta) \to (\alpha \leftrightarrow \beta)))$ **3**

(b) Natural deduction tree for

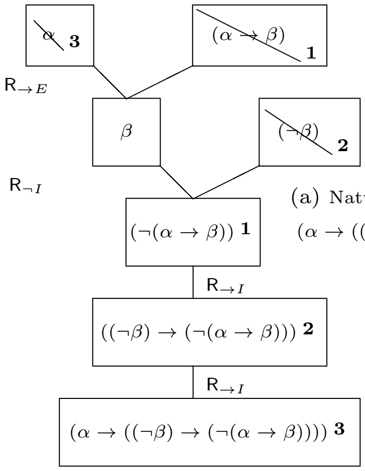
$((\neg\alpha) \to ((\neg\beta) \to (\alpha \leftrightarrow \beta)))$

Fig. 3.23. Natural deduction trees for Axioms (11) and (12).



$\cancel{\beta}$ 1
$\cancel{(\neg\beta)}$ 3
$R_{\neg E}$
$\cancel{(\alpha \vee \beta)}$ 2
$\cancel{\alpha}$ 1
α
$R_{\vee E}$
α 1
$\cancel{(\neg\alpha)}$ 4
$R_{\neg I}$
$(\neg(\alpha \vee \beta))$ 2
$R_{\to I}$
$((\neg\beta) \to (\neg(\alpha \vee \beta)))$ 3
$R_{\to I}$
$((\neg\alpha) \to ((\neg\beta) \to (\neg(\alpha \vee \beta)))$ 4
(a) Natural deduction tree for
$((\neg\alpha) \to ((\neg\beta) \to (\neg(\alpha \vee \beta)))$
(b) Natural deduction trees for $((\neg\alpha) \to (\neg(\alpha \wedge \beta)))$
and $((\neg\beta) \to (\neg(\alpha \wedge \beta)))$
$\cancel{(\alpha \wedge \beta)}$ 1
$R_{\wedge El}$
α
$\cancel{(\neg\alpha)}$ 2
$R_{\neg I}$
$(\neg(\alpha \wedge \beta))$ 1
$R_{\to I}$
$((\neg\alpha) \to (\neg(\alpha \wedge \beta)))$ 2
$\cancel{(\alpha \wedge \beta)}$ 1
$R_{\wedge Er}$
β
$\cancel{(\neg\beta)}$ 2
$R_{\neg I}$
$(\neg(\alpha \wedge \beta))$ 1
$R_{\to I}$
$((\neg\beta) \to (\neg(\alpha \wedge \beta)))$ 2

Fig. 3.24. Natural deduction trees for Axioms (13)–(15).


$\cancel{\alpha}$ 3
$\cancel{(\alpha \to \beta)}$ 1
$\mathsf{R}_{\to E}$
β
$\cancel{(\neg\beta)}$ 2
$\mathsf{R}_{\neg I}$
$(\neg(\alpha \to \beta))$ 1
$\mathsf{R}_{\to I}$
$((\neg\beta) \to (\neg(\alpha \to \beta)))$ 2
$\mathsf{R}_{\to I}$
$(\alpha \to ((\neg\beta) \to (\neg(\alpha \to \beta))))$ 3

(a) Natural deduction tree for $(\alpha \to ((\neg\beta) \to (\neg(\alpha \to \beta))))$

(b) Natural deduction trees for $(\alpha \to ((\neg\beta) \to (\neg(\alpha \leftrightarrow \beta))))$ and $((\neg\alpha) \to (\beta \to (\neg(\alpha \leftrightarrow \beta))))$

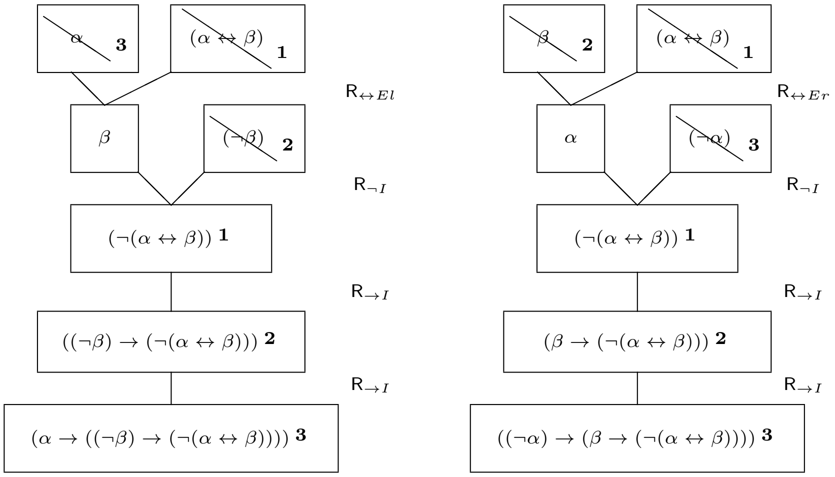

$\cancel{\alpha}$ 3
$\cancel{(\alpha \leftrightarrow \beta)}$ 1
$\mathsf{R}_{\leftrightarrow El}$
β
$\cancel{(\neg\beta)}$ 2
$\mathsf{R}_{\neg I}$
$(\neg(\alpha \leftrightarrow \beta))$ 1
$\mathsf{R}_{\to I}$
$((\neg\beta) \to (\neg(\alpha \leftrightarrow \beta)))$ 2
$\mathsf{R}_{\to I}$
$(\alpha \to ((\neg\beta) \to (\neg(\alpha \leftrightarrow \beta))))$ 3
$\cancel{\beta}$ 2
$\cancel{(\alpha \leftrightarrow \beta)}$ 1
$\mathsf{R}_{\leftrightarrow Er}$
α
$\cancel{(\neg\alpha)}$ 3
$\mathsf{R}_{\neg I}$
$(\neg(\alpha \leftrightarrow \beta))$ 1
$\mathsf{R}_{\to I}$
$(\beta \to (\neg(\alpha \leftrightarrow \beta)))$ 2
$\mathsf{R}_{\to I}$
$((\neg\alpha) \to (\beta \to (\neg(\alpha \leftrightarrow \beta))))$ 3

Fig. 3.25. Natural deduction trees for Axioms (16)–(18).

The *rules for introducing connective symbols* are as follows:

$\dfrac{(\Gamma_0,\varphi),(\Gamma_1,\psi)}{(\Gamma_0\cup\Gamma_1,(\varphi\wedge\psi))}$	$\wedge$- introduction
$\dfrac{(\Gamma,\varphi)}{(\Gamma,(\varphi\vee\psi))}$ $\quad$ $\dfrac{(\Gamma,\psi)}{(\Gamma,(\varphi\vee\psi))}$	$\vee$- introduction
$\dfrac{(\Gamma\cup\{\varphi\},\psi)}{(\Gamma,(\varphi\to\psi))}$	$\to$- introduction
$\dfrac{(\Gamma_0\cup\{\varphi\},\psi),(\Gamma_1\cup\{\psi\},\varphi)}{(\Gamma_0\cup\Gamma_1,(\varphi\leftrightarrow\psi))}$	$\leftrightarrow$- introduction
$\dfrac{(\Gamma_0\cup\{\varphi\},\psi),(\Gamma_1\cup\{\varphi\},(\neg\psi))}{(\Gamma_0\cup\Gamma_1,(\neg\varphi))}$	$\neg$- introduction

The *rules for eliminating connective symbols* are as follows:

$\dfrac{(\Gamma,(\varphi\wedge\psi))}{(\Gamma,\varphi)}$ $\quad$ $\dfrac{(\Gamma,(\varphi\wedge\psi))}{(\Gamma,\psi)}$	$\wedge$- elimination
$\dfrac{(\Gamma_0,(\varphi\vee\psi)),(\Gamma_1\cup\{\varphi\},\alpha),(\Gamma_2\cup\{\psi\},\alpha)}{(\Gamma_0\cup\Gamma_1\cup\Gamma_2,\alpha)}$	$\vee$- elimination
$\dfrac{(\Gamma_0,\varphi),(\Gamma_1,(\varphi\to\psi))}{(\Gamma_0\cup\Gamma_1,\psi)}$	$\to$- elimination
$\dfrac{(\Gamma_0,\varphi),(\Gamma_1,(\varphi\leftrightarrow\psi))}{(\Gamma_0\cup\Gamma_1,\psi)}$ $\quad$ $\dfrac{(\Gamma_0,\psi),(\Gamma_1,(\varphi\leftrightarrow\psi))}{(\Gamma_0\cup\Gamma_1,\varphi)}$	$\leftrightarrow$- elimination
$\dfrac{(\Gamma_0\cup\{(\neg\varphi)\},\psi),(\Gamma_1\cup\{(\neg\varphi))\},(\neg\psi)}{(\Gamma_0\cup\Gamma_1,\varphi)}$	$\neg$- elimination

$\square$

Example 3.6.15. We prove that $\vdash_{\mathcal{ND}} (\emptyset,((\varphi\vee(\alpha\wedge\beta))\to((\varphi\vee\alpha)\wedge(\varphi\vee\beta))))$ for all formulas $\varphi,\alpha,\beta\in$ PLFORM.

Consider the following proof, where $\theta=(\varphi\vee(\alpha\wedge\beta))$:

(1) $(\{\theta\},(\varphi\vee(\alpha\wedge\beta)))$	axiom
(2) $(\{\theta,\varphi\},\varphi)$	axiom
(3) $(\{\theta,(\alpha\wedge\beta)\},(\alpha\wedge\beta))$	axiom
(4) $(\{\theta,\varphi\},(\varphi\vee\alpha))$	(2) and $\vee$-introduction
(5) $(\{\theta,\varphi\},(\varphi\vee\beta))$	(2) and $\vee$-introduction
(6) $(\{\theta,\varphi\},((\varphi\vee\alpha)\wedge(\varphi\vee\beta)))$	(4), (5) and $\wedge$-introduction
(7) $(\{\theta,(\alpha\wedge\beta)\},\alpha)$	(3) and $\wedge$-elimination
(8) $(\{\theta,(\alpha\wedge\beta)\},\beta)$	(3) and $\wedge$-elimination
(9) $(\{\theta,(\alpha\wedge\beta)\},(\varphi\vee\alpha))$	(7) and $\vee$-introduction
(10) $(\{\theta,(\alpha\wedge\beta)\},(\varphi\vee\beta))$	(8) and $\vee$-introduction
(11) $(\{\theta,(\alpha\wedge\beta)\},((\varphi\vee\alpha)\wedge(\varphi\vee\beta)))$	(9),(10) and $\wedge$-introduction

At this point by applying the $\vee$-elimination rule to steps (1), (6), and (11), we obtain

$$(\{(\varphi \vee (\alpha \wedge \beta))\}, ((\varphi \vee \alpha) \wedge (\varphi \vee \beta))).$$

Using the $\rightarrow$-introduction rule, we have $\vdash_{\mathcal{ND}} (\emptyset, ((\varphi \vee (\alpha \wedge \beta)) \rightarrow ((\varphi \vee \alpha) \wedge (\varphi \vee \beta))))$. $\square$

Theorem 3.6.16. *Let* Γ *be a set of formulas and let* φ *be a formula. Then,* $\Gamma \overset{\bullet}{\vdash}_{nd} \varphi$ *if and only if* $\vdash_{\mathcal{ND}} (\Gamma, \varphi)$.

Proof. To prove that $\Gamma \overset{\bullet}{\vdash}_{\mathrm{nd}} \varphi$ implies $\vdash_{\mathcal{ND}} (\Gamma, \varphi)$, we use induction on natural deduction trees to show that if $(\mathtt{T}, M)$ is a natural deduction tree such that $\mathcal{UNC}(\mathtt{T}, M) \subseteq \Gamma$ and $\mathtt{T}(\lambda) = \varphi$, then $\vdash_{\mathcal{ND}} (\Gamma, \varphi)$. If $(\mathtt{T}, \emptyset)$ is a one-node natural deduction tree, we have $\varphi \in \Gamma$, so (Γ, φ) is an axiom of $\mathcal{ND}$. This establishes the basis step.

According to the definition of natural deduction trees, we need to consider seven inductive steps. We discuss here the one corresponding to $\vee$-elimination and leave the rest of them to the reader. Suppose that $(\mathtt{T}_0, M_0)$, $(\mathtt{T}_1, M_1)$, $(\mathtt{T}_2, M_2)$ are natural deduction trees such that

$$\mathtt{T}_0(\lambda) = (\varphi \vee \psi), \mathtt{T}_1(\lambda) = \mathtt{T}_2(\lambda) = \theta$$

and that $(\mathtt{T}, M)$ is

$$((\mathtt{T}_0, M_0), (\mathtt{T}_1, M_1 \cup \mathrm{LEAVES}_{\varphi}(\mathtt{T}_1)), (\mathtt{T}_2, M_2 \cup \mathrm{LEAVES}_{\psi}(\mathtt{T}_2)); \theta),$$

with $\mathcal{UNC}(\mathtt{T}, M) \subseteq \Gamma$. This implies

$$\begin{aligned}
&\mathcal{UNC}(\mathtt{T}_0, M_0) \subseteq \Gamma, \\
&\mathcal{UNC}(\mathtt{T}_1, M_1) \subseteq \Gamma \cup \{\varphi\}, \\
&\mathcal{UNC}(\mathtt{T}_2, M_2) \subseteq \Gamma \cup \{\psi\}.
\end{aligned}$$

By inductive hypothesis, $(\Gamma, (\varphi \vee \psi))$, $(\Gamma \cup \{\varphi\}, \theta)$, and $(\Gamma \cup \{\psi\}, \theta)$ are theorems of $\mathcal{ND}$. Applying the $\vee$-elimination rule of $\mathcal{ND}$, we obtain $\vdash_{\mathcal{ND}} (\Gamma, \theta)$.

Conversely, we show by induction on the theorems of $\mathcal{ND}$ that $\vdash_{\mathcal{ND}} (\Gamma, \varphi)$ implies $\Gamma \overset{\bullet}{\vdash}_{\mathrm{nd}} \varphi$. For the basis step, let (Γ, φ) be an axiom of $\mathcal{ND}$. Since $\varphi \in \Gamma$, the existence of the one node natural deduction tree $(\mathtt{T}, \emptyset)$ with $\mathtt{T}(\lambda) = \varphi$ implies $\Gamma \overset{\bullet}{\vdash}_{\mathrm{nd}} \varphi$.

There is one inductive step for each of the rules of $\mathcal{ND}$. We give here the step for $\vee$-elimination; the remaining steps are left for the reader. Suppose that (Γ, α) is obtained from $(\Gamma_0, (\varphi \vee \psi))$, $(\Gamma_1 \cup \{\varphi\}, \alpha)$, and $(\Gamma_2 \cup \{\psi\}, \alpha)$ by applying the $\vee$-elimination rule, which means that $\Gamma = \Gamma_0 \cup \Gamma_1 \cup \Gamma_2$. By the inductive hypothesis, we have $\Gamma_0 \overset{\bullet}{\vdash}_{\mathrm{nd}} (\varphi \vee \psi)$, $\Gamma_1 \cup \{\varphi\} \overset{\bullet}{\vdash}_{\mathrm{nd}} \alpha$, and $\Gamma_2 \cup \{\psi\} \overset{\bullet}{\vdash}_{\mathrm{nd}} \alpha$. Consequently, there are natural deduction trees $(\mathtt{T}_0, M_0)$, $(\mathtt{T}_1, M_1)$, $(\mathtt{T}_2, M_2)$ such that $\mathtt{T}_0(\lambda) = (\varphi \vee \psi)$, $\mathtt{T}_1(\lambda) = \mathtt{T}_2(\lambda) = \alpha$ and

$$\begin{aligned}
&\mathcal{UNC}(\mathtt{T}_0, M_0) \subseteq \Gamma_0 \subseteq \Gamma, \\
&\mathcal{UNC}(\mathtt{T}_1, M_1) \subseteq \Gamma_1 \cup \{\varphi\} \subseteq \Gamma \cup \{\varphi\}, \\
&\mathcal{UNC}(\mathtt{T}_2, M_2) \subseteq \Gamma_2 \cup \{\psi\} \subseteq \Gamma \cup \{\psi\}.
\end{aligned}$$

By Definition 3.6.7, we obtain the natural deduction tree $(\mathtt{T}, M)$ given by

$$((\mathtt{T}_0, M_0), (\mathtt{T}_1, M_1 \cup \mathrm{LEAVES}_{\varphi}(\mathtt{T}_1)), (\mathtt{T}_2, M_2 \cup \mathrm{LEAVES}_{\psi}(\mathtt{T}_2)); \alpha).$$

Since $\mathcal{UNC}(\mathtt{T}, M) \subseteq \Gamma$, we have $\Gamma \overset{\bullet}{\vdash}_{\mathrm{nd}} \alpha$. □

Corollary 3.6.17. *Let Γ be a set of formulas and φ be a formula. The following three statements are equivalent:*

(1) $\Gamma \overset{\bullet}{\vdash}_{nd} \varphi$,
(2) $\vdash_{\mathcal{ND}} (\Gamma, \varphi)$,
(3) $\Gamma \models \varphi$.

Proof. The statement follows from Theorems 3.6.16, 3.6.11, and 3.6.12. □

3.7 Translations between Formal Systems

The aim of this section is discuss syntactic transformations between proofs in different formalisms of the fact that $\Gamma \models \varphi$. The formalisms we have considered so far are

- a Hilbert–Frege system,
- tableaux with and without cut,

- sequent systems with and without cut,
- natural deduction.

Some of these transformations have already been presented. Others are introduced here.

3.7.1 *From Unsigned Tableaux to Hilbert–Frege Proofs*

Theorem 3.7.1. *There is an effective, syntactic construction that starts with a strongly closed unsigned* Γ*-tableau* $\mathtt{T}$ *and yields an* $(\mathcal{HF}, \Gamma)$*-certificate of inconsistency.*

Proof. If $\mathtt{T}$ consists of one node, then there is a formula φ such that $\{\varphi, (\neg\varphi)\} \subseteq \Gamma$. Thus, $((\varphi), ((\neg\varphi)))$ is the needed certificate of inconsistency.

If $\mathtt{T}$ consists of more than one node, we have two cases. If thinning was applied at the root and we have $\mathtt{T}(0) = \Gamma'$, then $\mathtt{T}_{[0]}$ is a strongly closed tableau. Therefore, by applying the method recursively, we obtain an $(\mathcal{HF}, \Gamma')$-certificate of inconsistency, which is also an $(\mathcal{HF}, \Gamma)$-certificate of inconsistency.

If regular expansion of the formula φ is used at the root of $\mathtt{T}$, the immediate descendents of the root are strongly closed tableaux. By applying the construction to these tableaux, we obtain certificates of inconsistency from them and then using Theorem 3.2.17, we assemble these certificates of inconsistency into an $(\mathcal{HF}, \Gamma)$-certificate of inconsistency. $\square$

The main result of the subsection is given in the following.

Theorem 3.7.2. *There is an effective, syntactic construction that starts with a strongly closed* $\Gamma \cup \{(\neg\varphi)\}$*-tableau* $\mathtt{T}$ *and produces a proof in* $\mathcal{HF}_\Gamma$ *of* φ*, for every set of formulas* Γ *and formula* φ *such that* $\Gamma \models \varphi$*.*

Proof. By Theorem 3.7.1, we can construct an $(\mathcal{HF}, \mathtt{T}(\lambda))$-certificate of inconsistency, where $\mathtt{T}(\lambda) = \Gamma \cup \{(\neg\varphi)\}$. By applying Part (2) of Theorem 3.2.15, we can construct effectively a proof of φ in $\mathcal{HF}$. $\square$

3.7.2 *From Natural Deduction Trees to Sequent Proofs*

Theorem 3.7.3. *There is a syntactic algorithm that takes as input a natural deduction tree $\mathcal{T} = (\mathtt{T}, M)$ in NDT and produces as output a $\mathcal{F}^{seq,cut}$-proof tree of the sequent $\mathcal{UNC}(\mathcal{T}) \Rightarrow \mathtt{T}(\lambda)$.*

Proof. We give a recursive algorithm, based on the inductive definition of natural deduction tree (cf. Definition 3.6.7). We proceed according to the first case of this definition that is applicable to $\mathcal{T}$.

If $\mathcal{T} = (\mathtt{T}, \emptyset)$ consists of one node, then the one node tree whose root is labeled by $\mathtt{T}(\lambda) \Rightarrow \mathtt{T}(\lambda)$ is the desired proof tree in $\mathcal{F}^{\text{seq,cut}}$ because $\mathcal{UNC}(\mathcal{T}) = \{\mathtt{T}(\lambda)\}$.

If $\mathcal{T} = (L_\varphi(\mathcal{T}_0); (\varphi \to \psi))$, where $\mathcal{T}_0 = (\mathtt{T}_0, M_0)$ and $\mathtt{T}_0(\lambda) = \psi$, we have $\mathcal{UNC}(\mathcal{T}_0) - \{\varphi\} = \mathcal{UNC}(\mathcal{T})$. Let $\mathtt{T}'_0$ be the $\mathcal{F}^{\text{seq,cut}}$-proof tree of $\mathcal{UNC}(\mathcal{T}_0) \Rightarrow \psi$ obtained by applying the algorithm recursively to $\mathcal{T}_0$. By Theorem 3.5.42, $\mathtt{T}'_1 = \mathtt{T}'_0 \boxplus (\varphi \Rightarrow \emptyset)$ is an $\mathcal{F}^{\text{seq,cut}}$-proof tree of the sequent

$$(\mathcal{UNC}(\mathcal{T}_0) - \{\varphi\}), \varphi \Rightarrow \psi.$$

Applying rule $\mathsf{R}_{\to,r}$ of $\mathcal{F}^{\text{seq,cut}}$ to $\mathtt{T}'_1$, we obtain the $\mathcal{F}^{\text{seq,cut}}$-proof tree $(\mathtt{T}'_1; \mathcal{UNC}(\mathcal{T}_0) - \{\varphi\} \Rightarrow (\varphi \to \psi))$ which is the desired output.

If $\mathcal{T} = (\mathcal{T}_0, L_\varphi(\mathcal{T}_1), L_\psi(\mathcal{T}_2); \theta)$, where $\mathcal{T}_i = (\mathtt{T}_i, M_i) \in$ NDT for $0 \le i \le 2$ and $\mathtt{T}_0(\lambda) = (\varphi \vee \psi)$, $\mathtt{T}_1(\lambda) = \mathtt{T}_2(\lambda) = \theta$, then observe that

$$\mathcal{UNC}(\mathcal{T}) = \mathcal{UNC}(\mathcal{T}_0) \cup (\mathcal{UNC}(\mathcal{T}_1) - \{\varphi\}) \cup (\mathcal{UNC}(\mathcal{T}_2) - \{\psi\}). \quad (3.7)$$

By applying the algorithm recursively to $\mathcal{T}_0, \mathcal{T}_1, \mathcal{T}_2$, we obtain the $\mathcal{F}^{\text{seq,cut}}$-proof trees $\mathtt{T}'_0, \mathtt{T}'_1, \mathtt{T}'_2$ of the sequents

$$\begin{aligned}
&\mathcal{UNC}(\mathcal{T}_0) \Rightarrow (\varphi \vee \psi),\\
&\mathcal{UNC}(\mathcal{T}_1) \Rightarrow \theta,\\
&\mathcal{UNC}(\mathcal{T}_2) \Rightarrow \theta,
\end{aligned}$$

respectively. By Theorem 3.5.42, we obtain $\mathcal{F}^{\text{seq,cut}}$-proof trees

$$\begin{aligned}
\mathtt{T}''_0 &= \mathtt{T}'_0 \boxplus (\mathcal{UNC}(\mathcal{T}_1) - \{\varphi\}) \cup (\mathcal{UNC}(\mathcal{T}_2) - \{\psi\}) \Rightarrow \theta,\\
\mathtt{T}''_1 &= \mathtt{T}'_1 \boxplus ((\mathcal{UNC}(\mathcal{T}_2) - \{\psi\}) \cup \{\varphi\} \Rightarrow \emptyset),\\
\mathtt{T}''_2 &= \mathtt{T}'_2 \boxplus ((\mathcal{UNC}(\mathcal{T}_1) - \{\varphi\}) \cup \{\psi\} \Rightarrow \emptyset)
\end{aligned}$$

of the sequents

$$\begin{aligned}&\mathcal{UNC}(\mathcal{T}_0) \cup (\mathcal{UNC}(\mathcal{T}_1) - \{\varphi\}) \cup (\mathcal{UNC}(\mathcal{T}_2) - \{\psi\}) \Rightarrow \theta, (\varphi \vee \psi),\\ &(\mathcal{UNC}(\mathcal{T}_1) - \{\varphi\}) \cup (\mathcal{UNC}(\mathcal{T}_2) - \{\psi\}), \varphi \Rightarrow \theta,\\ &(\mathcal{UNC}(\mathcal{T}_1) - \{\varphi\}) \cup (\mathcal{UNC}(\mathcal{T}_2) - \{\psi\}), \psi \Rightarrow \theta,\end{aligned}$$

respectively.

Applying Rule $\mathsf{R}_{\vee,l}$ of $\mathcal{F}^{\text{seq,cut}}$, we obtain the $\mathcal{F}^{\text{seq,cut}}$-proof tree

$$\mathtt{T}_3' = (\mathtt{T}_1'', \mathtt{T}_2''; (\mathcal{UNC}(\mathcal{T}_1) - \{\varphi\}) \cup (\mathcal{UNC}(\mathcal{T}_2) - \{\psi\}), (\varphi \vee \psi) \Rightarrow \theta).$$

Another application of Theorem 3.5.42 gives the $\mathcal{F}^{\text{seq,cut}}$-proof tree $\mathtt{T}_3'' = \mathtt{T}_3' \boxplus (\mathcal{UNC}(\mathcal{T}_0) \Rightarrow \emptyset)$ of the sequent

$$\mathcal{UNC}(\mathcal{T}_0) \cup (\mathcal{UNC}(\mathcal{T}_1) - \{\varphi\}) \cup (\mathcal{UNC}(\mathcal{T}_2) - \{\psi\}), (\varphi \vee \psi) \Rightarrow \theta.$$

Applying the cut rule to $\mathtt{T}_3''$ and $\mathtt{T}_0''$ yields the $\mathcal{F}^{\text{seq,cut}}$-proof tree

$$\mathtt{T}_4' = (\mathtt{T}_3'', \mathtt{T}_0''; \mathcal{UNC}(\mathcal{T}_0) \cup (\mathcal{UNC}(\mathcal{T}_1) - \{\varphi\}) \cup (\mathcal{UNC}(\mathcal{T}_2) - \{\psi\}) \Rightarrow \theta),$$

which, in view of Equation (3.7), is the desired output.

If $\mathcal{T} = (L_\varphi(\mathcal{T}_0), L_\psi(\mathcal{T}_1); (\varphi \leftrightarrow \psi))$, where $\mathcal{T}_0 = (\mathtt{T}_0, M_0)$ and $\mathcal{T}_1 = (\mathtt{T}_1, M_1)$ belong to NDT, $\mathtt{T}_0(\lambda) = \psi$, and $\mathtt{T}_1(\lambda) = \varphi$, then we can recursively apply the algorithm to $\mathcal{T}_0$ and $\mathcal{T}_1$ to obtain $\mathcal{F}^{\text{seq,cut}}$-proof trees $\mathtt{T}_0', \mathtt{T}_1'$ of the sequents $\mathcal{UNC}(\mathcal{T}_0) \Rightarrow \psi$ and $\mathcal{UNC}(\mathcal{T}_1) \Rightarrow \varphi$, respectively. Using Theorem 3.5.42, we construct the $\mathcal{F}^{\text{seq,cut}}$-proof trees

$$\begin{aligned}\mathtt{T}_0'' &= \mathtt{T}_0' \boxplus ((\mathcal{UNC}(\mathcal{T}_1) - \{\psi\}) \cup \{\varphi\}) \Rightarrow \emptyset,\\ \mathtt{T}_1'' &= \mathtt{T}_1' \boxplus ((\mathcal{UNC}(\mathcal{T}_0) - \{\varphi\}) \cup \{\psi\}) \Rightarrow \emptyset\end{aligned}$$

of the sequents

$$\begin{aligned}&(\mathcal{UNC}(\mathcal{T}_0) - \{\varphi\}) \cup (\mathcal{UNC}(\mathcal{T}_1) - \{\psi\}), \varphi \Rightarrow \psi,\\ &(\mathcal{UNC}(\mathcal{T}_0) - \{\varphi\}) \cup (\mathcal{UNC}(\mathcal{T}_1) - \{\psi\}), \psi \Rightarrow \varphi,\end{aligned}$$

respectively. By applying Rule $\mathsf{R}_{\leftrightarrow,r}$, we obtain the $\mathcal{F}^{\text{seq,cut}}$-proof tree

$$\mathtt{T}_2' = (\mathtt{T}_0'', \mathtt{T}_1''; (\mathcal{UNC}(\mathcal{T}_0) - \{\varphi\}) \cup (\mathcal{UNC}(\mathcal{T}_1) - \{\psi\}) \Rightarrow (\varphi \leftrightarrow \psi)),$$

which is the desired $\mathcal{F}^{\text{seq,cut}}$-proof tree because $\mathcal{UNC}(\mathcal{T}) = (\mathcal{UNC}(\mathcal{T}_0) - \{\varphi\}) \cup (\mathcal{UNC}(\mathcal{T}_1) - \{\psi\})$.

If $\mathcal{T} = (L_\varphi(\mathcal{T}_0), L_\varphi(\mathcal{T}_1); (\neg\varphi))$, where $\mathcal{T}_0 = (\mathtt{T}_0, M_0)$ and $\mathcal{T}_1 = (\mathtt{T}_1, M_1)$ belong to NDT, $\mathtt{T}_0(\lambda) = \psi$, $\mathtt{T}_1(\lambda) = (\neg\psi)$, and φ is a formula, then a recursive application of the algorithm yields $\mathcal{F}^{\mathrm{seq,cut}}$-proof trees $\mathtt{T}'_0, \mathtt{T}'_1$ of the sequents $\mathcal{UNC}(\mathcal{T}_0) \Rightarrow \psi$ and $\mathcal{UNC}(\mathcal{T}_1) \Rightarrow (\neg\psi)$, respectively. Using again Theorem 3.5.42, we obtain the $\mathcal{F}^{\mathrm{seq,cut}}$-proof trees $\mathtt{T}''_0 = \mathtt{T}'_0 \boxplus (\mathcal{UNC}(\mathcal{T}_1) \cup \{\varphi\} \Rightarrow \emptyset)$ and $\mathtt{T}''_1 = \mathtt{T}'_1 \boxplus (\mathcal{UNC}(\mathcal{T}_0) \cup \{\varphi\} \Rightarrow \emptyset)$ of $(\mathcal{UNC}(\mathcal{T}_0) - \{\varphi\}) \cup (\mathcal{UNC}(\mathcal{T}_1) - \{\varphi\}), \varphi \Rightarrow \psi$ and $(\mathcal{UNC}(\mathcal{T}_0) - \{\varphi\}) \cup (\mathcal{UNC}(\mathcal{T}_1) - \{\varphi\}), \varphi \Rightarrow (\neg\psi)$. By Rule $\mathsf{R}_{\wedge,r}$, we have the $\mathcal{F}^{\mathrm{seq,cut}}$-proof tree

$$\mathtt{T}'_2 = (\mathtt{T}''_0, \mathtt{T}''_1; (\mathcal{UNC}(\mathcal{T}_0) - \{\varphi\}) \cup (\mathcal{UNC}(\mathcal{T}_1) - \{\varphi\}), \varphi \Rightarrow (\psi \wedge (\neg\psi))).$$

By Supplement 54, Part (a), there is a $\mathcal{F}^{\mathrm{seq,cut}}$-proof tree $\mathtt{T}'_3$ of $(\psi \wedge (\neg\psi)) \Rightarrow$. Using Theorem 3.5.42, we build the $\mathcal{F}^{\mathrm{seq,cut}}$-proof tree

$$\mathtt{T}''_3 = \mathtt{T}'_3 \boxplus (\mathcal{UNC}(\mathcal{T}_0) - \{\varphi\}) \cup (\mathcal{UNC}(\mathcal{T}_1) - \{\varphi\}), \varphi \Rightarrow \emptyset.$$

Therefore, by the cut rule, we obtain the $\mathcal{F}^{\mathrm{seq,cut}}$-proof tree

$$\mathtt{T}'_4 = (\mathtt{T}''_3, \mathtt{T}'_2; (\mathcal{UNC}(\mathcal{T}_0) - \{\varphi\}) \cup (\mathcal{UNC}(\mathcal{T}_1) - \{\varphi\}), \varphi \Rightarrow).$$

Finally, an application of $\mathsf{R}_{\neg,r}$ gives the $\mathcal{F}^{\mathrm{seq,cut}}$-proof tree

$$\mathtt{T}'_5 = (\mathtt{T}'_4; (\mathcal{UNC}(\mathcal{T}_0) - \{\varphi\}) \cup (\mathcal{UNC}(\mathcal{T}_1) - \{\varphi\}) \Rightarrow (\neg\varphi)),$$

which is the needed output because

$$\mathcal{UNC}(\mathcal{T}) = (\mathcal{UNC}(\mathcal{T}_0) - \{\varphi\}) \cup (\mathcal{UNC}(\mathcal{T}_1) - \{\varphi\}).$$

If $\mathcal{T} = (L_{(\neg\varphi)}(\mathcal{T}_0), L_{(\neg\varphi)}(\mathcal{T}_1)); \varphi)$, where $\mathcal{T}_0 = (\mathtt{T}_0, M_0)$ and $\mathcal{T}_1 = (\mathtt{T}_1, M_1)$ both belong to NDT, $\mathtt{T}_0(\lambda) = \psi$, $\mathtt{T}_1(\lambda) = (\neg\psi)$, and $\varphi \in$ PLFORM, then by applying the previous construction with φ replaced by $(\neg\varphi)$, we obtain a $\mathcal{F}^{\mathrm{seq,cut}}$-proof tree $\mathtt{T}'_5$ of

$$(\mathcal{UNC}(\mathcal{T}_0) - \{(\neg\varphi)\}) \cup (\mathcal{UNC}(\mathcal{T}_1) - \{(\neg\varphi)\}) \Rightarrow (\neg(\neg\varphi))$$

and let $\mathtt{T}''_5 = \mathtt{T}'_5 \boxplus \emptyset \Rightarrow \varphi$. Supplement 54, Part (b), gives a $\mathcal{F}^{\mathrm{seq,cut}}$-proof tree $\mathtt{T}'_6$ of $(\neg(\neg\varphi)) \Rightarrow \varphi$. This yields the $\mathcal{F}^{\mathrm{seq,cut}}$-proof tree

$$\mathtt{T}''_6 = \mathtt{T}'_6 \boxplus (\mathcal{UNC}(\mathcal{T}_0) - \{(\neg\varphi)\}) \cup (\mathcal{UNC}(\mathcal{T}_1) - \{(\neg\varphi)\}) \Rightarrow \emptyset$$

of the sequent

$$(\mathcal{UNC}(\mathcal{T}_0) - \{(\neg\varphi)\}) \cup (\mathcal{UNC}(\mathcal{T}_1) - \{(\neg\varphi)\}) \cup \{(\neg(\neg\varphi))\} \Rightarrow \varphi.$$

Using the cut rule, we have the $\mathcal{F}^{\text{seq,cut}}$-proof tree

$$\mathtt{T}'_7 = (\mathtt{T}''_6, \mathtt{T}''_5; (\mathcal{UNC}(\mathcal{T}_0) - \{(\neg\varphi)\}) \cup (\mathcal{UNC}(\mathcal{T}_1) - \{(\neg\varphi)\}) \Rightarrow \varphi),$$

which is the desired output.

When $\mathcal{T} = (\mathcal{T}_0; \varphi)$, where $\mathcal{T}_0 = (\mathtt{T}_0, M_0) \in \mathrm{NDT}$ and $\mathtt{T}_0(\lambda) = (\varphi \wedge \psi)$, by recursive application of the algorithm to $\mathcal{T}_0$, we obtain an $\mathcal{F}^{\text{seq,cut}}$-proof tree $\mathtt{T}'_0$ of $\mathcal{UNC}(\mathcal{T}_0) \Rightarrow (\varphi \wedge \psi)$. Let $\mathtt{T}'_1$ be the $\mathcal{F}^{\text{seq,cut}}$-proof tree of the sequent $(\varphi \wedge \psi) \Rightarrow \varphi$ which exists by Part (c) of Supplement 54. Next, we build the $\mathcal{F}^{\text{seq,cut}}$-proof trees $\mathtt{T}''_0 = \mathtt{T}'_0 \boxplus (\emptyset \Rightarrow \varphi)$ and $\mathtt{T}''_1 = \mathtt{T}'_1 \boxplus (\mathcal{UNC}(\mathcal{T}_0) \Rightarrow \emptyset)$ of the sequents $\mathcal{UNC}(\mathcal{T}_0) \Rightarrow (\varphi \wedge \psi), \varphi$ and $\mathcal{UNC}(\mathcal{T}_0), (\varphi \wedge \psi) \Rightarrow \varphi$, respectively. Combining $\mathtt{T}''_0$ and $\mathtt{T}''_1$ using the cut rule gives $\mathtt{T}'_2 = (\mathtt{T}''_1, \mathtt{T}''_0; \mathcal{UNC}(\mathcal{T}_0) \Rightarrow \varphi)$. The companion case, when $\mathcal{T} = (\mathcal{T}_0; \psi)$, is left to the reader.

When $\mathcal{T} = (\mathcal{T}_0; (\varphi \vee \psi))$, where $\mathcal{T}_0 = (\mathtt{T}_0, M_0) \in \mathrm{NDT}$ and $\mathtt{T}_0(\lambda) = \varphi$, by recursive application of the algorithm to $\mathcal{T}_0$, we obtain an $\mathcal{F}^{\text{seq,cut}}$-proof tree $\mathtt{T}'_0$ of $\mathcal{UNC}(\mathcal{T}_0) \Rightarrow \varphi$. This allows us to construct $\mathtt{T}'_1 = \mathtt{T}'_0 \boxplus (\emptyset \Rightarrow \psi)$, a $\mathcal{F}^{\text{seq,cut}}$-proof tree of $\mathcal{UNC}(\mathcal{T}_0) \Rightarrow \varphi, \psi$. By application of $\mathsf{R}_{\vee,r}$, we obtain the $\mathcal{F}^{\text{seq,cut}}$-proof tree $\mathtt{T}'_2 = (\mathtt{T}'_1; \mathcal{UNC}(\mathcal{T}_0) \Rightarrow (\varphi \vee \psi))$. Again, we leave to the reader the case when $\mathcal{T}_0(\lambda) = \psi$.

If $\mathcal{T} = (\mathcal{T}_0, \mathcal{T}_1; (\varphi \wedge \psi))$, where $\mathcal{T}_0 = (\mathtt{T}_0, M_0), \mathcal{T}_1 = (\mathtt{T}_1, M_1) \in \mathrm{NDT}$, and $\mathtt{T}_0(\lambda) = \varphi, \mathtt{T}_1(\lambda) = \psi$, recursive application of the algorithm yields $\mathcal{F}^{\text{seq,cut}}$-proof trees $\mathtt{T}'_0, \mathtt{T}'_1$ of $\mathcal{UNC}(\mathcal{T}_0) \Rightarrow \varphi$ and $\mathcal{UNC}(\mathcal{T}_1) \Rightarrow \psi$, respectively. Thus, we have the $\mathcal{F}^{\text{seq,cut}}$-proof trees

$$\mathtt{T}''_0 = \mathtt{T}'_0 \boxplus (\mathcal{UNC}(\mathcal{T}_1) \Rightarrow \emptyset) \text{ and } \mathtt{T}''_1 = \mathtt{T}'_1 \boxplus (\mathcal{UNC}(\mathcal{T}_0) \Rightarrow \emptyset)$$

of the sequents

$$\mathcal{UNC}(\mathcal{T}_0) \cup \mathcal{UNC}(\mathcal{T}_1) \Rightarrow \varphi \text{ and } \mathcal{UNC}(\mathcal{T}_0) \cup \mathcal{UNC}(\mathcal{T}_1) \Rightarrow \psi,$$

respectively. Finally, by applying $\mathsf{R}_{\wedge,r}$, we obtain the desired $\mathcal{F}^{\text{seq,cut}}$-proof tree

$$\mathtt{T}'_2 = (\mathtt{T}''_0, \mathtt{T}''_1; \mathcal{UNC}(\mathcal{T}_0) \cup \mathcal{UNC}(\mathcal{T}_1) \Rightarrow (\varphi \wedge \psi)).$$

If $\mathcal{T} = (\mathcal{T}_0, \mathcal{T}_1; \psi)$, where $\mathcal{T}_0 = (\mathtt{T}_0, M_0), \mathcal{T}_1 = (\mathtt{T}_1, M_1) \in \mathrm{NDT}$ and $\mathtt{T}_0(\lambda) = \varphi, \mathtt{T}_1(\lambda) = (\varphi \to \psi)$, we obtain the $\mathcal{F}^{\text{seq,cut}}$-proof trees $\mathtt{T}'_0, \mathtt{T}'_1$ of $\mathcal{UNC}(\mathcal{T}_0) \Rightarrow \varphi$ and $\mathcal{UNC}(\mathcal{T}_1) \Rightarrow (\varphi \to \psi)$, respectively, by recursive application of the algorithm to $\mathcal{T}_0, \mathcal{T}_1$. These trees allow us to construct the $\mathcal{F}^{\text{seq,cut}}$-proof trees $\mathtt{T}''_0 = \mathtt{T}'_0 \boxplus (\mathcal{UNC}(\mathcal{T}_1) \Rightarrow \psi)$

and $\mathtt{T}_1'' = \mathtt{T}_1' \boxplus (\mathcal{UNC}(\mathcal{T}_0) \Rightarrow \psi)$ of $\mathcal{UNC}(\mathcal{T}_0) \cup \mathcal{UNC}(\mathcal{T}_1) \Rightarrow \varphi, \psi$ and $\mathcal{UNC}(\mathcal{T}_0) \cup \mathcal{UNC}(\mathcal{T}_1) \Rightarrow (\varphi \to \psi), \psi$, respectively. Let $\mathtt{T}_2'$ be the one-node $\mathcal{F}^{\mathrm{seq,cut}}$-proof tree whose root is labeled by $\mathcal{UNC}(\mathcal{T}_0) \cup \mathcal{UNC}(\mathcal{T}_1), \psi \Rightarrow \psi$. By applying the rule $\mathsf{R}_{\to,l}$ of $\mathcal{F}^{\mathrm{seq,cut}}$, we obtain the $\mathcal{F}^{\mathrm{seq,cut}}$-proof tree

$$\mathtt{T}_3' = (\mathtt{T}_2', \mathtt{T}_0''; \mathcal{UNC}(\mathcal{T}_0) \cup \mathcal{UNC}(\mathcal{T}_1), (\varphi \to \psi) \Rightarrow \psi).$$

Finally, an application of the cut rule gives us the desired $\mathcal{F}^{\mathrm{seq,cut}}$-proof tree

$$\mathtt{T}_4' = (\mathtt{T}_3', \mathtt{T}_1''; \mathcal{UNC}(\mathcal{T}_0) \cup \mathcal{UNC}(\mathcal{T}_1) \Rightarrow \psi).$$

Let now $\mathcal{T} = (\mathcal{T}_0, \mathcal{T}_1; \psi)$, where $\mathcal{T}_i = (\mathtt{T}_i, M_i)$ for $i = 0, 1$, and $\mathtt{T}_0(\lambda) = \varphi, \mathtt{T}_1(\lambda) = (\varphi \leftrightarrow \psi)$. Through recursive application of the algorithm, we obtain the $\mathcal{F}^{\mathrm{seq,cut}}$-proof trees $\mathtt{T}_0'$ and $\mathtt{T}_1'$ of $\mathcal{UNC}(\mathcal{T}_0) \Rightarrow \varphi$ and $\mathcal{UNC}(\mathcal{T}_1) \Rightarrow (\varphi \leftrightarrow \psi)$. Further, we have the $\mathcal{F}^{\mathrm{seq,cut}}$-proof tree $\mathtt{T}_0'' = \mathtt{T}_0' \boxplus (\mathcal{UNC}(\mathcal{T}_1) \Rightarrow \psi)$ of the sequent $\mathcal{UNC}(\mathcal{T}_0) \cup \mathcal{UNC}(\mathcal{T}_1) \Rightarrow \varphi, \psi$. Starting from the one-node $\mathcal{F}^{\mathrm{seq,cut}}$-proof tree $\mathtt{T}_2'$ of $\mathcal{UNC}(\mathcal{T}_0) \cup \mathcal{UNC}(\mathcal{T}_1), \varphi, \psi \Rightarrow \psi$, we construct the $\mathcal{F}^{\mathrm{seq,cut}}$-proof tree

$$\mathtt{T}_3' = (\mathtt{T}_2', \mathtt{T}_0''; \mathcal{UNC}(\mathcal{T}_0) \cup \mathcal{UNC}(\mathcal{T}_1), (\varphi \leftrightarrow \psi) \Rightarrow \psi)$$

using the rule $\mathsf{R}_{\leftrightarrow,l}$. From $\mathtt{T}_1'$, we build the $\mathcal{F}^{\mathrm{seq,cut}}$-proof tree $\mathtt{T}_1'' = \mathtt{T}_1' \boxplus (\mathcal{UNC}(\mathcal{T}_0) \Rightarrow \psi)$ of $\mathcal{UNC}(\mathcal{T}_0) \cup \mathcal{UNC}(\mathcal{T}_1) \Rightarrow (\varphi \leftrightarrow \psi), \psi$. Finally, by applying the cut rule, we obtain the desired $\mathcal{F}^{\mathrm{seq,cut}}$-proof tree

$$\mathtt{T}_4' = (\mathtt{T}_3', \mathtt{T}_1''; \mathcal{UNC}(\mathcal{T}_0) \cup \mathcal{UNC}(\mathcal{T}_1) \Rightarrow \psi).$$

We leave to the reader the case when $\mathcal{T} = (\mathcal{T}_0, \mathcal{T}_1; \varphi)$ and $\mathtt{T}_0(\lambda) = \psi$. □

Corollary 3.7.4. *There is a syntactic construction that given a set of formulas* Γ *and a natural deduction tree* $\mathcal{T} = (\mathtt{T}, M)$ *in NDT such that* $\mathcal{UNC}(\mathcal{T}) \subseteq \Gamma$ *produces an* $\mathcal{F}^{seq,\infty,cut}$*-proof tree for* $\Gamma \Rightarrow \mathtt{T}(\lambda)$.

Proof. The $\mathcal{F}^{\mathrm{seq},\infty,\mathrm{cut}}$-proof tree for $\Gamma \Rightarrow \mathtt{T}(\lambda)$ can be obtained from the $\mathcal{F}^{\mathrm{seq,cut}}$-proof tree obtained from the algorithm outlined in Theorem 3.7.3 by applying the thinning rule once. □

Note that the syntactic construction of Corollary 3.7.4 is effective if Γ is a finite set.

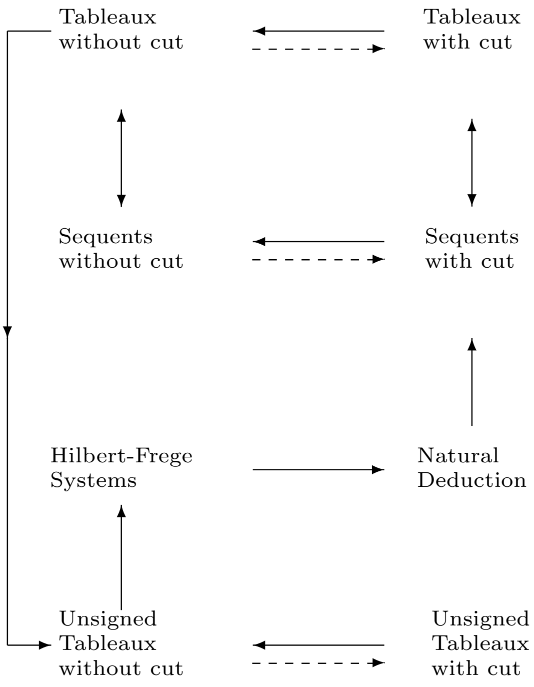


Fig. 3.26. General layout of transformations.

3.7.3 *Closing the Circle*

We now close the circle, that is, using results we have proven so far, we show how to transform a formal proof that $\Gamma \models \varphi$ in one system into a formal proof in another one. The general layout of these transformations is shown in Figure 3.26. The specific objects being transformed as well as the results involved are shown in Figure 3.27.

Note that all transformations shown in Figure 3.27 become effective if the set of formulas Γ is finite.

3.8 Resolution

Resolution is a formalism that handles satisfiability of formulas in conjunctive normal form. It uses a special representation of these formulas as collections of sets of literals referred to as *clauses* (defined in the following).

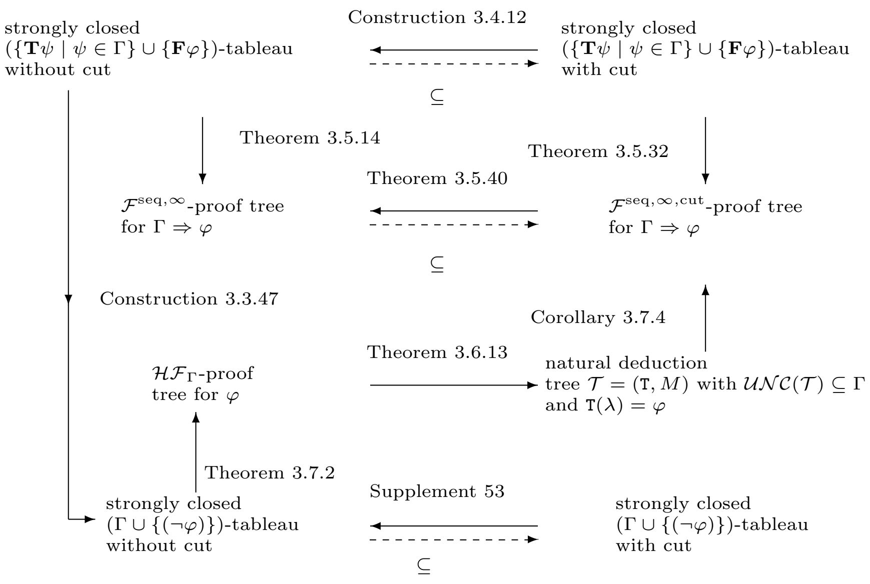

strongly closed $(\{\mathbf{T}\psi \mid \psi \in \Gamma\} \cup \{\mathbf{F}\varphi\})$-tableau without cut
Construction 3.4.12
strongly closed $(\{\mathbf{T}\psi \mid \psi \in \Gamma\} \cup \{\mathbf{F}\varphi\})$-tableau with cut
$\subseteq$
Theorem 3.5.14
Theorem 3.5.32
Theorem 3.5.40
$\mathcal{F}^{\mathrm{seq},\infty}$-proof tree for $\Gamma \Rightarrow \varphi$
$\mathcal{F}^{\mathrm{seq},\infty,\mathrm{cut}}$-proof tree for $\Gamma \Rightarrow \varphi$
$\subseteq$
Construction 3.3.47
Corollary 3.7.4
Theorem 3.6.13
$\mathcal{HF}_\Gamma$-proof tree for φ
natural deduction tree $\mathcal{T} = (\mathtt{T}, M)$ with $\mathcal{UNC}(\mathcal{T}) \subseteq \Gamma$ and $\mathtt{T}(\lambda) = \varphi$
Theorem 3.7.2
Supplement 53
strongly closed $(\Gamma \cup \{(\neg\varphi)\})$-tableau without cut
strongly closed $(\Gamma \cup \{(\neg\varphi)\})$-tableau with cut
$\subseteq$

Fig. 3.27. Objects being transformed.

Definition 3.8.1. A *clause* is a finite set of literals. The empty clause will be called “box” and denoted by □. ▯

Of course, $\Box = \emptyset$, but we use a different notation than $\emptyset$ in order to avoid confusion with other uses of $\emptyset$, such as denoting the empty set of clauses.

When we write clauses, we will systematically omit parentheses.

Example 3.8.2. The clause $\{p, (\neg q), (\neg r)\}$ will be written as $\{p, \neg q, \neg r\}$. ▯

The following definition introduces several kinds of clauses we will be using.

Definition 3.8.3. A clause is a

- *tautologous clause* if it contains both p and $\neg p$ for some statement variable p,
- *Horn clause* if it contains at most one positive literal,
- *positive clause* if it contains only positive literals,
- *negative clause* if it contains only negative literals,

- *non-positive clause* if it contains at least one negative literal,
- *unit clause* if it consists of a single literal. ▯

We use the letters C, D, E, with or without subscripts to denote clauses and $\mathcal{C}, \mathcal{D}, \mathcal{E}$ to denote sets of clauses.

If C is a clause, let $SV(C)$ be the set of all statement variables p such that p or $\neg p$ is a member of C. For a set of clauses $\mathcal{C}$, we define the sets $SV(\mathcal{C}) = \bigcup_{C \in \mathcal{C}} SV(C)$ and $\text{LIT}(\mathcal{C}) = \bigcup \mathcal{C}$. In other words, $\text{LIT}(\mathcal{C})$ is the set of literals which appear in some clause of $\mathcal{C}$.

Definition 3.8.4. Let C be a clause and v be a truth assignment; v *satisfies* C if $v(\ell) = \mathbf{T}$ for some literal $\ell \in C$; if v does not satisfy C, we say that v *falsifies* C. ▯

Note that the notion of a truth assignment satisfying a set of formulas introduced before Definition 2.3.4 is quite different from the notion of a truth assignment satisfying a clause given above. We rely on the context to differentiate between these concepts.

Definition 3.8.5. A truth assignment v satisfies a set of clauses $\mathcal{C}$ if v satisfies every clause in $\mathcal{C}$; otherwise, we say that v *falsifies* $\mathcal{C}$.

A clause is *satisfiable* if there is a truth assignment that satisfies the clause; likewise, a set of clauses is *satisfiable* if there is a truth assignment that satisfies the set of clauses. ▯

Note that every nonempty clause is satisfiable.

The following theorem gives the basic properties of satisfaction.

Theorem 3.8.6. *Let $\mathcal{C}$ and $\mathcal{D}$ be sets of clauses:*

(1) *If $\mathcal{C} \subseteq \mathcal{D}$, then every truth assignment which satisfies $\mathcal{D}$ also satisfies $\mathcal{C}$.*
(2) *If $\mathcal{C} \subseteq \mathcal{D}$, then if $\mathcal{D}$ is satisfiable, $\mathcal{C}$ is satisfiable and if $\mathcal{C}$ is unsatisfiable, $\mathcal{D}$ is unsatisfiable.*
(3) *$\square$ is an unsatisfiable clause and $\{\square\}$ is an unsatisfiable set of clauses.*
(4) *$\emptyset$ is a satisfiable set of clauses.*
(5) *If $\mathcal{C}$ contains $\square$, then $\mathcal{C}$ is unsatisfiable.*
(6) *If C is a tautologous clause, then every truth assignment satisfies C.*

Proof. We leave most of this easy proof to the reader. Note, however, that $\square$ is unsatisfiable because in order to be satisfiable, a truth assignment v and a literal ℓ in $\square$ should exist such that $v(\ell) = \mathbf{T}$; since $\square$ does not contain any literal, it follows that it cannot be satisfiable. On the other hand, the empty set of clauses is satisfiable because, otherwise for every truth assignment v, a clause in $\emptyset$ should exist that is not satisfied by v. Since no such clause exists, $\emptyset$ is satisfiable. $\square$

Definition 3.8.7. Let

$$\varphi = \bigwedge_{i=0}^{n-1} (\ell_{i0} \vee \cdots \vee \ell_{im_i-1}),$$

where each ℓ_{ij} is a literal, be a formula in conjunctive normal form. The *clause set associated with* φ is

$$\mathcal{C}_\varphi = \{\{\ell_{i0}, \ldots, \ell_{im_i-1}\} \mid 0 \leq i \leq n-1\}.$$

Let Γ be a set of formulas in conjunctive normal form. The set of clauses $\mathcal{C}_\Gamma$ is the set $\bigcup_{\varphi \in \Gamma} \mathcal{C}_\varphi$. $\square$

For every set of clauses $\mathcal{C}$ that does not contain $\square$, there is a set of formulas Γ each of which is a disjunction of literals (hence in degenerate conjunctive normal form) such that $\mathcal{C} = \mathcal{C}_\Gamma$. Note that Γ is not unique since the order of the literals is not determined uniquely.

If φ is in conjunctive normal form, then $\mathcal{C}_\varphi$ is a set of Horn clauses if and only if φ is a Horn formula. Further, if Γ is a finite set of formulas in conjunctive normal form, then $\mathcal{C}_\Gamma$ is a finite of clauses.

The following result allows us to translate satisfiability of formulas in conjunctive normal form into satisfiability of sets of clauses.

Theorem 3.8.8. *If φ is a formula in conjunctive normal form and v is a truth assignment, then v satisfies φ if and only if v satisfies $\mathcal{C}_\varphi$.*

Proof. Note that if φ is as in Definition 3.8.7, then a truth assignment v satisfies φ if and only if for every i, $0 \leq i \leq n-1$, there is a j, $0 \leq j \leq m_i - 1$ such that $v(\ell_{ij}) = \mathbf{T}$. It is easy to see that this is equivalent to v satisfying $\mathcal{C}_\varphi$. $\square$

Corollary 3.8.9. *If Γ is a set of formulas in conjunctive normal form, then Γ is satisfiable if and only if $\mathcal{C}_\Gamma$ is satisfiable.*

Proof. The proof is immediate and is left to the reader. $\square$

Example 3.8.10. Consider the formula $\psi = ((p \vee (\neg q)) \wedge ((\neg p) \vee q))$. Its corresponding set of clauses is

$$\mathcal{C}_\psi = \{\{p, \neg q\}, \{\neg p, q\}\}.$$

It is easy to see that both ψ and $\mathcal{C}_\psi$ are satisfiable since the truth assignment $v \in \mathrm{TA}_{SV(\psi)}$ given by $v(p) = v(q) = \mathbf{T}$ satisfies them both.

Let $\varphi = ((p \vee q) \wedge ((\neg p) \vee q) \wedge (p \vee (\neg q)) \wedge ((\neg p) \vee (\neg q)))$. Neither φ nor $\mathcal{C}_\varphi$ given by

$$\mathcal{C}_\varphi = \{\{p, q\}, \{\neg p, q\}, \{p, \neg q\}, \{\neg p, \neg q\}\}$$

are satisfiable. Indeed, suppose that $u \in \mathrm{TA}_{SV(\varphi)}$ satisfies $\mathcal{C}_\varphi$. Since u must satisfy $\{p, q\}$, we must have $u(p) = \mathbf{T}$ or $u(q) = \mathbf{T}$ but not both (because this would imply u falsifies $\{\neg p, \neg q\}$). Suppose that $u(p) = \mathbf{T}$ and $u(q) = \mathbf{F}$. This means that u falsifies $\{\neg p, q\}$ so u falsifies $\mathcal{C}_\varphi$. A similar conclusion can be reached if we assume that $u(p) = \mathbf{F}$ and $u(q) = \mathbf{T}$. This proves that φ and $\mathcal{C}_\varphi$ are not satisfiable. ▯

The Compactness Theorem for formulas can be transferred to a compactness result for sets of clauses.

Theorem 3.8.11 (Compactness Theorem for Clauses). *Let $\mathcal{C}$ be an unsatisfiable set of clauses. Then, there exists a finite subset $\mathcal{C}_0$ of $\mathcal{C}$ such that $\mathcal{C}_0$ is unsatisfiable.*

Proof. If $\square \in \mathcal{C}$, then we take $\mathcal{C}_0 = \{\square\}$. So, suppose that $\square \notin \mathcal{C}$. We observed that there exists a set of formulas Γ in conjunctive normal form such that $\mathcal{C} = \mathcal{C}_\Gamma$. By Corollary 3.8.9, Γ is unsatisfiable and by the Compactness Theorem for formulas, Γ contains a finite subset Γ_0 that is unsatisfiable. Again by Corollary 3.8.9, $\mathcal{C}_{\Gamma_0}$ is unsatisfiable and $\mathcal{C}_{\Gamma_0} \subseteq \mathcal{C}_\Gamma = \mathcal{C}$. $\square$

It is possible to reduce the fundamental problem we are concerned with across different formalisms, namely, the logical implication of a formula φ by a set of formulas Γ, to the satisfiability of a set of clauses. Indeed, we saw $\Gamma \models \varphi$ if and only if the set of formulas $\Gamma' = \Gamma \cup \{(\neg\varphi)\}$ is unsatisfiable. If Γ'' is obtained from Γ' by replacing each formula with an equivalent formula in conjunctive normal form, then Γ' is unsatisfiable if and only if Γ'' is unsatisfiable. Finally, we have shown that Γ'' is unsatisfiable if and only if $\mathcal{C}_{\Gamma''}$ is unsatisfiable.

Definition 3.8.12. Let C_0, C_1 and R be clauses. Then, we say that R is obtained from C_0 and C_1 by resolving on the literal ℓ and we write $R = \mathrm{res}_\ell(C_0, C_1)$, if

(1) $\ell \in C_0$,
(2) $\bar{\ell} \in C_1$,
(3) $R = (C_0 - \{\ell\}) \cup (C_1 - \{\bar{\ell}\})$.

R is called a *resolvent* of C_0 and C_1 if there is a literal ℓ such that $R = \mathrm{res}_\ell(C_0, C_1)$. The clauses C_0, C_1 are referred to as *premises* of R. ▯

For any clause R, there are infinitely many possible pairs of premises for that clause. This can be seen by defining $C_0 = R \cup \{\ell\}$ and $C_1 = R \cup \{\bar{\ell}\}$, for any literal ℓ which does not occur in R.

Theorem 3.8.13. *Let $\mathcal{C}$ be a set of clauses and let R be a resolvent of two clauses in $\mathcal{C}$. Then, for every truth assignment v, v satisfies $\mathcal{C}$ if and only if v satisfies $\mathcal{C} \cup \{R\}$.*

Proof. By the first part of Theorem 3.8.6, if v satisfies $\mathcal{C} \cup \{R\}$, then v satisfies $\mathcal{C}$.

Conversely, let v satisfy $\mathcal{C}$ and assume that $R = (C_0 - \{\ell\}) \cup (C_1 - \{\bar{\ell}\})$, where $C_0, C_1 \in \mathcal{C}$. Then, v satisfies C_0 and C_1. We consider two cases.

If $v(\ell) = \mathbf{T}$, then $v(\bar{\ell}) = \mathbf{F}$. Since v satisfies C_1, it must make some literal in C_1, other than $\bar{\ell}$, true, and this means that v satisfies $C_1 - \{\bar{\ell}\}$ and, therefore, it satisfies R.

If $v(\ell) = \mathbf{F}$, then, since v satisfies C_0, it must make some literal in C_0, other than ℓ, true, and this means that v satisfies $C_0 - \{\ell\}$ and, therefore, it satisfies R. □

Definition 3.8.14. Let $\mathcal{C}$ be a set of clauses. Then, we define

$$\mathrm{Res}(\mathcal{C}) = \mathcal{C} \cup \{R \mid R \text{ is a resolvent of two clauses in } \mathcal{C}\}.$$

▯

Resolution is analytical in the sense that every literal that occurs in $\text{Res}(\mathcal{C})$ occurs in member of $\mathcal{C}$.

Theorem 3.8.15. *Let $\mathcal{C}$ and $\mathcal{D}$ be sets of clauses and let v be a truth assignment. Then,*

(1) $\mathcal{C} \subseteq Res(\mathcal{C})$,
(2) *if* $\mathcal{C} \subseteq \mathcal{D}$*, then* $Res(\mathcal{C}) \subseteq Res(\mathcal{D})$,
(3) v *satisfies* $Res(\mathcal{C})$ *if and only if* v *satisfies* $\mathcal{C}$.

Proof. The first two parts of the theorem are immediate consequences of Definition 3.8.14.

By Part 1 of Theorem 3.8.6 and Part 1 of this theorem, if v satisfies $\text{Res}(\mathcal{C})$, then v satisfies $\mathcal{C}$. Conversely, if v satisfies $\mathcal{C}$, then, by Theorem 3.8.13, v satisfies every resolvent of two clauses in $\mathcal{C}$, and therefore, v satisfies $\text{Res}(\mathcal{C})$. □

Let $\mathcal{S}$ be the set of all clauses. Then, $\text{Res} : \mathcal{P}(\mathcal{S}) \longrightarrow \mathcal{P}(\mathcal{S})$, so we can consider its iterations Res^n for $n \in \mathbf{N}$, following the standard definition:

$$\text{Res}^0(\mathcal{C}) = \mathcal{C},$$
$$\text{Res}^{n+1}(\mathcal{C}) = \text{Res}(\text{Res}^n(\mathcal{C}))$$

for every set of clauses $\mathcal{C} \in \mathcal{P}(\mathcal{S})$.

Note that, as a consequence of the first two parts of Theorem 3.8.15, we have the increasing chain of sets

$$\mathcal{C} = \text{Res}^0(\mathcal{C}) \subseteq \text{Res}^1(\mathcal{C}) \subseteq \cdots \subseteq \text{Res}^n(\mathcal{C}) \subseteq \cdots . \tag{3.8}$$

Definition 3.8.16. Let $\mathcal{C}$ be a set of clauses. Then, we define $\text{Res}^*(\mathcal{C})$, the *resolution closure* of $\mathcal{C}$, by

$$\text{Res}^*(\mathcal{C}) = \bigcup_{n \geq 0} \text{Res}^n(\mathcal{C}).$$

▯

It is clear that every literal in $\text{Res}^*(\mathcal{C})$ occurs in a member of $\mathcal{C}$, which is another manifestation of the analyticity of resolution.

Theorem 3.8.17. *Let $\mathcal{C}$ and $\mathcal{D}$ be sets of clauses and let v be a truth assignment. Then,*

(1) $\mathcal{C} \subseteq Res^n(\mathcal{C})$ *for all* $n \in \mathbf{N}$ *and* $\mathcal{C} \subseteq Res^*(\mathcal{C})$,
(2) *for all* $n \in \mathbf{N}$, v *satisfies* $Res^n(\mathcal{C})$ *if and only if* v *satisfies* $\mathcal{C}$,
(3) v *satisfies* $Res^*(\mathcal{C})$ *if and only if* v *satisfies* $\mathcal{C}$,
(4) $\mathcal{C}$ *is satisfiable if and only if* $Res^*(\mathcal{C})$ *is satisfiable,*
(5) *if* $\mathcal{C} \subseteq \mathcal{D}$, *then* $Res^n(\mathcal{C}) \subseteq Res^n(\mathcal{D})$ *for all* $n \in \mathbf{N}$ *and* $Res^*(\mathcal{C}) \subseteq Res^*(\mathcal{D})$.

Proof. The first part of the theorem follows from Equation (3.8).

For the second part of the theorem, the argument is by induction on n. The basis step, $n = 0$, is trivial. Suppose that for every truth assignment v, v satisfies $\mathcal{C}$ if and only if v satisfies $\mathrm{Res}^n(\mathcal{C})$. Using the last part of Theorem 3.8.15, applied to $\mathrm{Res}^n(\mathcal{C})$, we obtain the desired conclusion.

The third part is an easy consequence of the second and the fourth part follows immediately from the third. The fifth part is a simple proof by induction, which uses Theorem 3.8.15, Part (2). □

Definition 3.8.18. Let $\mathcal{C}$ be a set of clauses. A *resolution proof* over $\mathcal{C}$ is a finite sequence $(C_0, C_1, \ldots, C_{n-1})$ of clauses such that $n \geq 1$ and for each i, $0 \leq i \leq n-1$ either $C_i \in \mathcal{C}$ or else $C_i \notin \mathcal{C}$ and there are $j, k < i$ such C_i is a resolvent of C_j and C_k. In the first case, i is an *input step* of the proof; in the second case, i is a *resolution step* and C_j and C_k are premises of the ith step.

A *resolution proof of a clause* C over $\mathcal{C}$ is a resolution proof over $\mathcal{C}$ whose last entry is C. ▯

Note that for a resolution step i, the premises need not be unique. If there is no risk of confusion, we will also refer to C_i as the ith step of the proof.

Theorem 3.8.19. *Let* $\mathcal{C}$ *be a set of clauses. Then,* $\mathrm{Res}^*(\mathcal{C})$ *is the set of clauses which have resolution proofs over* $\mathcal{C}$.

Proof. We first show that every clause in $\mathrm{Res}^*(\mathcal{C})$ has a resolution proof over $\mathcal{C}$ by using induction to show that every clause in $\mathrm{Res}^n(\mathcal{C})$ has a resolution proof over $\mathcal{C}$. The basis, $n = 0$, is trivial since every clause in $\mathcal{C}$ has a proof of length 1. Suppose that every clause in $\mathrm{Res}^n(\mathcal{C})$ has a proof and let $C \in \mathrm{Res}^{n+1}(\mathcal{C})$. If $C \in \mathrm{Res}^n(\mathcal{C})$, we are done by inductive hypothesis. Otherwise, C is a resolvent of two clauses C' and C'' in $\mathrm{Res}^n(\mathcal{C})$. By the inductive hypothesis, we have resolution proofs over $\mathcal{C}$

$$(C'_0, \dots, C'_{m-1}) \text{ where } C'_{m-1} = C',$$
$$(C''_0, \dots, C''_{k-1}) \text{ where } C''_{k-1} = C''.$$

Then, $(C'_0, \dots, C'_{m-1}, C''_0, \dots, C''_{k-1}, C)$ is a resolution proof over $\mathcal{C}$ for C.

Suppose that C has a resolution proof $(C_0, \dots, C_{m-1})$ over $\mathcal{C}$. We prove by course-of-values induction that $C_i \in \mathrm{Res}^*(\mathcal{C})$ for every i, $0 \le i \le m-1$. Suppose that the result is true for all $j < i$. If C_i is in $\mathcal{C}$, then by Part (1) of Theorem 3.8.17, $C_i \in \mathrm{Res}^*(\mathcal{C})$. Otherwise, C_i is a resolvent of C_j, C_k for some $j, k < i$. By inductive hypothesis, C_j, C_k are both in $\mathrm{Res}^*(\mathcal{C})$ and therefore, there are $n_j, n_k \in \mathbf{N}$ such that $C_j \in \mathrm{Res}^{n_j}(\mathcal{C})$ and $C_k \in \mathrm{Res}^{n_k}(\mathcal{C})$. Let $n' = \max\{n_j, n_k\}$. By (3.8), we have $C_j, C_k \in \mathrm{Res}^{n'}(\mathcal{C})$ and therefore, $C \in \mathrm{Res}^{n'+1}(\mathcal{C}) \subseteq \mathrm{Res}^*(\mathcal{C})$, which completes the induction. It follows that $C_{m-1} = C \in \mathrm{Res}^*(\mathcal{C})$, as desired. □

The previous theorem can be rephrased in terms of formal systems.

Definition 3.8.20. The formal system $\mathcal{FRES}$ is

$$(\mathcal{P}_{\text{fin}}(\text{LIT}), \emptyset, \{\mathsf{R}\}),$$

where the binary rule R consists of all pairs $((C, D), E)$ where E is a resolvent of C and D. ▯

Note that if $\mathcal{C}$ is a set of clauses, then a resolution proof over $\mathcal{C}$ is the same thing as a proof in the formal system $\mathcal{FRES}_{\mathcal{C}}$. Thus, Theorem 3.8.19 amounts to saying that $\mathrm{Thm}(\mathcal{FRES}_{\mathcal{C}}) = \mathrm{Res}^*(\mathcal{C})$. The introduction of a formal system allows us to make use of the idea of proof tree.

Definition 3.8.21. Let $\mathcal{C}$ be a set of clauses. A *resolution tree over* C is an $\mathcal{FRES}_{\mathcal{C}}$-proof tree. ▯

In other words, a resolution tree over $\mathcal{C}$ is a lot such that its leaves are labeled with clauses from $\mathcal{C}$ and each interior node is labeled with a clause that is a resolvent of the clauses which are labels of its two immediate descendents. Theorem 1.8.23 and our previous discussion allow us to conclude that a clause C is in $\mathrm{Res}^*(\mathcal{C})$ if and only if there is a resolution tree over $\mathcal{C}$ such that C is the label of its root.

Example 3.8.22. In Example 3.8.10 we saw that the set of clauses

$$\mathcal{C} = \{\{p, q\}, \{p, \neg q\}, \{\neg p, q\}, \{\neg p, \neg q\}\}$$

is not satisfiable.

We can reach the same conclusion by Part (4) of Theorem 3.8.17 and considering the resolution proof $(C_0, \ldots, C_6)$ given in the following, where $C_6 = \Box$.

Clause	Clause content	Derivation
C_0	$\{p, q\}$	In $\mathcal{C}$
C_1	$\{p, \neg q\}$	In $\mathcal{C}$
C_2	$\{\neg p, q\}$	In $\mathcal{C}$
C_3	$\{\neg p, \neg q\}$	In $\mathcal{C}$
C_4	$\{p\}$	$\mathrm{Res}_q(C_0, C_1)$
C_5	$\{\neg p\}$	$\mathrm{Res}_q(C_2, C_3)$
C_6	$\Box$	$\mathrm{Res}_p(C_4, C_5)$

The resolution tree of this proof is given in Figure 3.28.

∎

Theorem 3.8.23. *Let $\mathcal{C}$ be a set of clauses:*

(1) *If* $\mathrm{Res}^n(\mathcal{C}) = \mathrm{Res}^{n+1}(\mathcal{C})$ *for some* $n \in \mathbf{N}$, *then* $\mathrm{Res}^*(\mathcal{C}) = \mathrm{Res}^n(\mathcal{C})$.
(2) *If $\mathcal{C}$ is finite, then* $\mathrm{Res}^n(\mathcal{C}) = \mathrm{Res}^{n+1}(\mathcal{C})$ *for some* $n \in \mathbf{N}$.

Proof. Suppose that $\mathrm{Res}^n(\mathcal{C}) = \mathrm{Res}^{n+1}(\mathcal{C})$. We show by induction on k that $\mathrm{Res}^{n+k}(\mathcal{C}) \subseteq \mathrm{Res}^n(\mathcal{C})$. For $k = 0$, this is trivial. Suppose

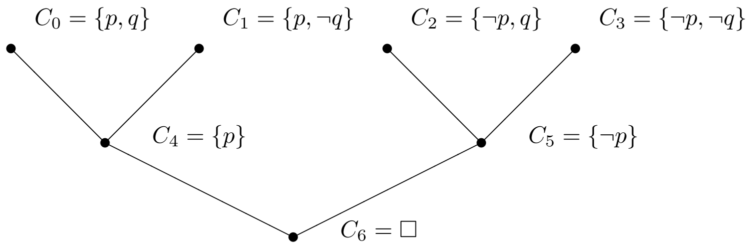


Fig. 3.28. Resolution tree of resolution proof.

that $\mathrm{Res}^{n+k}(\mathcal{C}) \subseteq \mathrm{Res}^n(\mathcal{C})$. Using the second part of Theorem 3.8.15, it follows that $\mathrm{Res}^{n+k+1}(\mathcal{C}) \subseteq \mathrm{Res}^{n+1}(\mathcal{C}) = \mathrm{Res}^n(\mathcal{C})$, by hypothesis. This shows that

$$\mathrm{Res}^*(\mathcal{C}) = \bigcup_{i \geq 0} \mathrm{Res}^i(\mathcal{C}) = \bigcup_{i \leq n} \mathrm{Res}^i(\mathcal{C}) = \mathrm{Res}^n(\mathcal{C}).$$

Note that if $\mathcal{C}$ is finite, then $\bigcup \mathcal{C}$, the set of literals that occur in clauses of $\mathcal{C}$, is finite because all clauses of $\mathcal{C}$ are finite. It is easy to verify, by induction on n, that $\bigcup \mathrm{Res}^n(\mathcal{C}) \subseteq \bigcup \mathcal{C}$ for every $n \in \mathbf{N}$. Therefore, there are only finitely many distinct members in the collection $\{\mathrm{Res}^n(\mathcal{C}) \mid n \in \mathbf{N}\}$. This implies that there is n such that $\mathrm{Res}^n(\mathcal{C}) = \mathrm{Res}^{n+1}(\mathcal{C})$ since otherwise, we would have the increasing sequence of sets

$$\mathcal{C} = \mathrm{Res}^0(\mathcal{C}) \subset \mathrm{Res}^1(\mathcal{C}) \subset \cdots \subset \mathrm{Res}^n(\mathcal{C}) \subset \cdots$$

and this would contradict the finiteness of $\{\mathrm{Res}^n(\mathcal{C}) \mid n \in \mathbf{N}\}$. □

Definition 3.8.24. Let $\mathcal{C}$ be a set of clauses and let ℓ be a literal. The set of clauses $\mathcal{C}^\ell$ is defined by

$$\mathcal{C}^\ell = \{C - \{\bar{\ell}\} \mid C \in \mathcal{C} \text{ and } \ell \notin C\}.$$

▯

Note that neither ℓ nor $\bar{\ell}$ belongs to any clause in $\mathcal{C}^\ell$.

Example 3.8.25. Let $\mathcal{C} = \{\{p, q, r\}, \{\bar{p}, \bar{q}, r\}, \{q, \bar{r}\}\}$. We have,

$$\mathcal{C}^p = \{\{\bar{q}, r\}, \{q, \bar{r}\}\},$$
$$\mathcal{C}^{\bar{p}} = \{\{q, r\}, \{q, \bar{r}\}\}.$$

Note that both $\mathcal{C}^p$ and $\mathcal{C}^{\bar{p}}$ are satisfiable. ▯

Theorem 3.8.26. *Let $\mathcal{C}$ be a set of clauses and let ℓ be a literal. Then, $\mathcal{C}$ is satisfiable if and only if at least one of $\mathcal{C}^\ell$ and $\mathcal{C}^{\bar{\ell}}$ is satisfiable.*

Proof. First, suppose that $\mathcal{C}$ is satisfiable, and let v be a truth assignment that satisfies $\mathcal{C}$. We consider two cases: $v(\ell) = \mathbf{T}$ and $v(\ell) = \mathbf{F}$. In the first case, we claim that v satisfies $\mathcal{C}^\ell$. Indeed, if $C \in \mathcal{C}$, then v satisfies C, and $v(\bar{\ell}) = \mathbf{F}$, so v satisfies $C - \{\bar{\ell}\}$, which shows that v satisfies $\mathcal{C}^{\ell}$. Similarly, it is easy to see that in the second case, v satisfies $\mathcal{C}^{\bar{\ell}}$.

Conversely, assume that at least one of the sets $\mathcal{C}^{\ell}$ and $\mathcal{C}^{\bar{\ell}}$ is satisfiable. If $\mathcal{C}^{\ell}$ is satisfiable, there exists a truth assignment v that satisfies all clauses of $\mathcal{C}^{\ell}$. We need to prove that there exists a truth assignment w that satisfies all clauses of $\mathcal{C}$.

Since v satisfies all clauses of $\mathcal{C}^{\ell}$, for every clause in $\mathcal{C}$ that does not contain ℓ, there is a literal ℓ' contained by that clause such that $v(\ell') = \mathbf{T}$ and $\ell' \neq \bar{\ell}$. Define a truth assignment w by

$$w = \begin{cases} [p \to \mathbf{T}]v \text{ if } \ell = p, \\ [p \to \mathbf{F}]v \text{ if } \ell = \neg p. \end{cases}$$

Clearly, w will satisfy all clauses of $\mathcal{C}$ that do not contain ℓ because w coincides with v on all literals with the possible exception of ℓ and $\bar{\ell}$. On the other hand, w satisfies all clauses that contain ℓ because $w(\ell) = \mathbf{T}$.

The case when $\mathcal{C}^{\bar{\ell}}$ is satisfiable can be dealt with in a similar manner. ◻

Lemma 3.8.27. *Let C, D, R be clauses such that $R = res_{\ell}(C, D)$. If E, F are clauses such that $\ell \notin E$ and $\bar{\ell} \notin F$, then*

$$R \cup E \cup F = \mathrm{res}_{\ell}(C \cup E, D \cup F).$$

Proof. Note that $\ell \in C \cup E$ and $\bar{\ell} \in D \cup F$. By elementary set-theoretical properties, we can write

$$\begin{aligned} \mathrm{res}_{\ell}(C \cup E, D \cup F) &= ((C \cup E) - \{\ell\}) \cup ((D \cup F) - \{\bar{\ell}\}) \\ &= (C - \{\ell\}) \cup (E - \{\ell\}) \cup (D - \{\bar{\ell}\}) \cup (F - \{\bar{\ell}\}) \\ &= ((C - \{\ell\}) \cup E) \cup ((D - \{\bar{\ell}\}) \cup F) \\ &= (C - \{\ell\}) \cup (D - \{\bar{\ell}\}) \cup E \cup F \\ &= R \cup E \cup F. \end{aligned}$$

◻

The following lemma says that if $\mathcal{C}$ is a set of clauses and ℓ is a literal, then a resolution proof $(C_0, \ldots, C_{n-1})$ over $\mathcal{C}^{\ell}$ can be "lifted" to a resolution proof $(C'_0, \ldots, C'_{n-1})$ over $\mathcal{C}$ such that for all i, $0 \leq i \leq n-1$, either $C'_i = C_i$ or $C'_i = C_i - \{\overline{\ell}\}$. The lemma is stated in a technical form which allows it to be applied in Section 3.9.

Lemma 3.8.28 (P-Lifting Lemma). *Let $\mathcal{C}$ be a set of clauses and ℓ be a literal. Suppose that $(C_0, \ldots, C_{n-1})$ is a resolution proof over $\mathcal{C}^\ell$. Let K be a subset of $\{0, \ldots, n-1\}$ that contains all resolution steps of this proof. Suppose that for each $k \in K$ we have $i_k, j_k < k$ and a literal ℓ_k such that $C_k = res_{\ell_k}(C_{i_k}, C_{j_k})$, then there is a resolution proof $(C'_0, \ldots, C'_{n-1})$ over $\mathcal{C}$ such that for every k such that $0 \leq k \leq n-1$, the following conditions are satisfied:*

(1) *the clause C'_k is either C_k or $C_k \cup \{\bar{\ell}\}$,*
(2) *if $k \notin K$ (so k is an input step of the proof $(C_0, \ldots, C_{n-1})$), then $C'_k \in \mathcal{C}$,*
(3) *if $k \in K$, then $C'_k = res_{\ell_k}(C'_{i_k}, C'_{j_k})$.*

Proof. The proof is by induction on n. For the basis step, $n = 1$, we must have $0 \notin K$ and $C_0 \in \mathcal{C}^\ell$. There is a clause $C'_0 \in \mathcal{C}$ such that $C_0 = C'_0 - \{\bar{\ell}\}$. Note that C'_0 is either C_0 or $C_0 \cup \{\bar{\ell}\}$.

Let $n > 1$ and suppose that the result is true for proofs of length $n-1$. Consider now a resolution proof $(C_0, \ldots, C_{n-1})$ over $\mathcal{C}^\ell$. By the inductive hypothesis, there is a resolution proof $(C'_0, \ldots, C'_{n-2})$ over $\mathcal{C}$ such that for every k, $0 \leq k \leq n-2$, the following conditions are satisfied:

(1) $C'_k = C_k$ or $C'_k = C_k \cup \{\bar{\ell}\}$,
(2) if $k \notin K$, then $C'_k \in \mathcal{C}$,
(3) if $k \in K$, then $C'_k = \text{res}_{\ell_k}(C'_{i_k}, C'_{j_k})$.

We consider two cases for $n - 1$:

(1) $n - 1 \notin K$,
(2) $n - 1 \in K$.

In the first case, there is a clause $C'_{n-1} \in \mathcal{C}$ such that $C_{n-1} = C'_{n-1} - \{\bar{\ell}\}$. Note that C'_{n-1} is either C_{n-1} or $C_{n-1} \cup \{\bar{\ell}\}$.

In the second case, we must have ℓ_{n-1} different from ℓ and $\bar{\ell}$ because neither ℓ nor $\bar{\ell}$ belongs to any clause of $\mathcal{C}^\ell$ and, therefore, neither belongs to any clause of $\text{Res}^*(\mathcal{C}^\ell)$. By the inductive hypothesis, we have $C'_{i_{n-1}} = C_{i_{n-1}} \cup E$ and $C'_{j_{n-1}} = C_{j_{n-1}} \cup F$, where E and F are either $\emptyset$ or $\{\bar{\ell}\}$. Let $C'_{n-1} = C_{n-1} \cup E \cup F$. Then, C'_{n-1} is either C_{n-1} or $C_{n-1} \cup \{\bar{\ell}\}$ and, by Lemma 3.8.27, $C'_{n-1} = \text{res}_{\ell_{n-1}}(C'_{i_{n-1}}, C'_{j_{n-1}})$.

In either case, $(C'_0, \ldots, C'_{n-1})$ is the desired resolution proof over $\mathcal{C}$. □

Theorem 3.8.29 (Soundness of Resolution). *Let $\mathcal{C}$ be a set of clauses. If $\Box \in \mathit{Res}^*(\mathcal{C})$, then $\mathcal{C}$ is unsatisfiable.*

Proof. If $\Box \in \mathrm{Res}^*(\mathcal{C})$, then, by the fifth part of Theorem 3.8.6, $\mathrm{Res}^*(\mathcal{C})$ is unsatisfiable so, by the fourth part of Theorem 3.8.17, $\mathcal{C}$ is unsatisfiable. $\Box$

In terms of the formal system $\mathcal{FRES}$, Theorem 3.8.29 means that if $\Box$ is a theorem of $\mathcal{FRES}_{\mathcal{C}}$, then $\mathcal{C}$ is unsatisfiable.

Theorem 3.8.30 (Resolution Completeness for Finite Sets). *Let $\mathcal{C}$ be a finite set of clauses. If $\mathcal{C}$ is unsatisfiable, then $\Box \in \mathit{Res}^*(\mathcal{C})$.*

Proof. We prove the statement by showing by induction on n that if $|SV(\mathcal{C})| = n$ and $\mathcal{C}$ is unsatisfiable, then $\Box \in \mathrm{Res}^*(\mathcal{C})$. This suffices to show the result, since if $\mathcal{C}$ is finite, then $SV(\mathcal{C})$ is finite.

For the basis step, suppose that $|SV(\mathcal{C})| = 0$ and that $\mathcal{C}$ is unsatisfiable. Then, $\mathcal{C} = \{\Box\}$. (The only other set of clauses without variables is $\emptyset$, but this set of clauses is satisfiable.) By the first part of Theorem 3.8.17, we have, for this $\mathcal{C}$, $\Box \in \mathcal{C} \subseteq \mathrm{Res}^*(\mathcal{C})$.

For the inductive step, suppose that $n \geq 0$ and that the result is true for all sets of clauses $\mathcal{C}$ with $|SV(\mathcal{C})| \leq n$. Suppose that $\mathcal{C}$ is an unsatisfiable set of clauses with $|SV(\mathcal{C})| = n + 1$. Let p be a member of $SV(\mathcal{C})$. Then, since neither p nor $\bar{p}$ appears in any clause of $\mathcal{C}^p$ and $\mathcal{C}^{\bar{p}}$, we have $|SV(\mathcal{C}^p)|, |SV(\mathcal{C}^{\bar{p}})| \leq n$. Also, by Theorem 3.8.26, $\mathcal{C}^p$ and $\mathcal{C}^{\bar{p}}$ are both unsatisfiable. Thus, by inductive hypothesis, $\Box$ is in both $\mathrm{Res}^*(\mathcal{C}^p)$ and $\mathrm{Res}^*(\mathcal{C}^{\bar{p}})$. By Theorem 3.8.19, we have resolution proofs $(C_0, \ldots, C_{m-1})$ of $\Box$ over $\mathcal{C}^p$ and $(D_0, \ldots, D_{k-1})$ of $\Box$ over $\mathcal{C}^{\bar{p}}$. Applying Lemma 3.8.28, we get resolution proofs $(C_0', \ldots, C_{m-1}')$ and $(D_0', \ldots, D_{k-1}')$ over $\mathcal{C}$ such that C_{m-1}' is either C_{m-1} or $C_{m-1} \cup \{\bar{p}\}$, that is, since $C_{m-1} = \Box$, C_{m-1}' is either $\Box$ or $\{\bar{p}\}$, and, similarly, D_{k-1}' is either $\Box$ or $\{p\}$. If either $C_{m-1}' = \Box$ or $D_{k-1}' = \Box$, then we have a resolution proof over $\mathcal{C}$ of $\Box$. If, on the other hand, $C_{m-1}' = \{\bar{p}\}$ and $D_{k-1}' = \{p\}$, then $(C_0', \ldots, C_{m-1}', D_0', \ldots, D_{k-1}', \Box)$ is a resolution proof of $\Box$ over $\mathcal{C}$. Thus, by Theorem 3.8.19, $\Box \in \mathrm{Res}^*(\mathcal{C})$. $\Box$

In terms of the formal system $\mathcal{FRES}$, Theorem 3.8.30 means that if $\mathcal{C}$ is unsatisfiable, then $\Box$ is a theorem of $\mathcal{FRES}_{\mathcal{C}}$.

Corollary 3.8.31. *Let $\mathcal{C}$ be a finite set of clauses. Then, $\mathcal{C}$ is unsatisfiable if and only if $\Box \in \mathit{Res}^*(\mathcal{C})$.*

Proof. This corollary follows from Theorems 3.8.29 and 3.8.30. $\square$

Theorem 3.8.32 (Resolution Completeness for All Sets). *Let $\mathcal{C}$ be a set of clauses. If $\mathcal{C}$ is unsatisfiable, then $\square \in Res^*(\mathcal{C})$.*

Proof. If $\mathcal{C}$ is unsatisfiable, by the Compactness Theorem for Clauses (Theorem 3.8.11), there is a finite unsatisfiable subset $\mathcal{C}_0$ of $\mathcal{C}$. By the Resolution Completeness Theorem for Finite Sets of Clauses, $\square \in \text{Res}^*(\mathcal{C}_0) \subseteq \text{Res}^*(\mathcal{C})$. $\square$

Corollary 3.8.33 (Resolution Soundness and Completeness). *Let $\mathcal{C}$ be a set of clauses. Then, $\mathcal{C}$ is unsatisfiable if and only if $\square \in Res^*(\mathcal{C})$.*

Proof. This statement follows from Theorems 3.8.32 and 3.8.29. $\square$

If $\mathcal{C}$ is a finite set of clauses, we now have the following algorithm to determine if $\mathcal{C}$ is satisfiable:

Algorithm 3.8.34.
Input: A finite set of clauses $\mathcal{C}$.
Output: "Yes" if $\mathcal{C}$ is satisfiable and "No" otherwise.
Method: Calculate $\text{Res}(\mathcal{C}), \text{Res}^2(\mathcal{C}), \ldots$ until a k is found with $\text{Res}^k(\mathcal{C}) = \text{Res}^{k+1}(\mathcal{C})$. Then, return "Yes" if $\square \notin \text{Res}^k(\mathcal{C})$ and return "No" otherwise.

Proof of Correctness: Since $\mathcal{C}$ is finite, by the second part of Theorem 3.8.23, a k as in the algorithm will be found, and, by the first part of this theorem, for this k, $\text{Res}^k(\mathcal{C}) = \text{Res}^*(\mathcal{C})$. Thus, by Corollary 3.8.33, the algorithm is correct. $\square$

In the previous algorithm, if it is found that $\square \in \text{Res}^k(\mathcal{C})$ for some k, then there is no need to continue on with the calculation of $\text{Res}^*(\mathcal{C})$, since it is already clear that $\square \in \text{Res}^*(\mathcal{C})$. Thus, a slightly more efficient version of the algorithm is given by the following:

Algorithm 3.8.35.
Input: A finite set of clauses $\mathcal{C}$.
Output: "Yes" if $\mathcal{C}$ is satisfiable and "No" otherwise.
Method: Calculate $\text{Res}(\mathcal{C}), \text{Res}^2(\mathcal{C}), \ldots$ until a k with either $\text{Res}^k(\mathcal{C}) = \text{Res}^{k+1}(\mathcal{C})$ or $\square \in \text{Res}^k(\mathcal{C})$ is obtained. Then, return "Yes" if $\square \notin \text{Res}^k(\mathcal{C})$ and return "No" otherwise.

Proof of Correctness: The correctness follows immediately from the above discussion. $\Box$

Example 3.8.36. We claim that the set of clauses

$$\mathcal{C} = \{\{\neg p_0, \neg p_1, p_2\}, \{p_0, p_2\}, \{p_1, p_2\}, \{\neg p_2\}\}$$

is unsatisfiable. In order to justify this claim, we will compute the sets $\mathrm{Res}^n(\mathcal{C})$ for $n \in \mathbf{N}$. Clearly, $\mathrm{Res}^0(\mathcal{C}) = \mathcal{C}$. The clauses of $\mathrm{Res}^1(\mathcal{C})$ are given in the following table:

Clause	Clause content	Derivation
C_0	$\{\neg p_0, \neg p_1, p_2\}$	in $\mathcal{C}$
C_1	$\{p_0, p_2\}$	in $\mathcal{C}$
C_2	$\{p_1, p_2\}$	in $\mathcal{C}$
C_3	$\{\neg p_2\}$	in $\mathcal{C}$
C_4	$\{\neg p_1, p_2\}$	$\mathrm{res}_{p_0}(C_1, C_0)$
C_5	$\{\neg p_0, p_2\}$	$\mathrm{res}_{p_1}(C_2, C_0)$
C_6	$\{\neg p_0, \neg p_1\}$	$\mathrm{res}_{p_2}(C_0, C_3)$
C_7	$\{p_0\}$	$\mathrm{res}_{p_2}(C_1, C_3)$
C_8	$\{p_1\}$	$\mathrm{res}_{p_2}(C_2, C_3)$

The set $\mathrm{Res}^2(\mathcal{C})$ includes the clauses C_0 to C_8 as well as the clauses included by the following table:

Clause	Clause content	Derivation
C_9	$\{p_2\}$	$\mathrm{Res}_{p_0}(C_1, C_5)$
C_{10}	$\{\neg p_1\}$	$\mathrm{Res}_{p_2}(C_4, C_3)$
C_{11}	$\{\neg p_0\}$	$\mathrm{Res}_{p_2}(C_5, C_3)$

Note that $\mathrm{Res}^3(\mathcal{C})$ contains the empty clause, since $\mathrm{res}_{p_0}(C_7, C_{11}) = \Box$, so $\mathcal{C}$ is unsatisfiable. ▯

If $\mathcal{C}$ is unsatisfiable, then this can be shown simply by exhibiting a resolution proof of $\Box$ over $\mathcal{C}$, and such a proof can sometimes be found by a trial and error process. It is sometimes much less work to find such a resolution proof of $\Box$ than it is to calculate all the $\mathrm{Res}^k(\mathcal{C})$s until one containing $\Box$ is found, since this latter calculation usually involves many resolvents which play no role in the generation of $\Box$. However, if it is not known in advance whether or not $\mathcal{C}$ is satisfiable, any time spent looking for a resolution proof of $\Box$ over $\mathcal{C}$ can be wasted (unless the search is carried out so systematically that it shows that no such resolution proof exists). A reasonable strategy

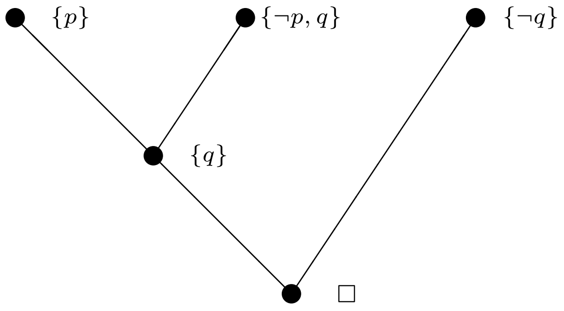


Fig. 3.29. Resolution tree for proving $\{p, (p \to q)\} \models q$.

is to first try to find a resolution proof of $\square$ and if this does not seem to be leading anywhere, to switch to the algorithm given above.

In view of previous discussion, we now have the following algorithm to determine if a finite set of formulas $\Gamma = \{\varphi_0, \ldots, \varphi_{n-1}\}$ logically implies a formula φ.

Algorithm 3.8.37.
Input: A finite set of formulas $\Gamma = \{\varphi_0, \ldots, \varphi_{n-1}\}$ and a formula φ.
Output: "Yes" if $\Gamma \models \varphi$ and "No" otherwise.
Method: Put $\varphi_0, \ldots, \varphi_{n-1}, (\neg\varphi)$ into conjunctive normal form to obtain a set of formulas Γ'. Let $\mathcal{C}_{\Gamma'}$ be as in Definition 3.8.7. Use Algorithm 3.8.34 to determine if $\mathcal{C}_{\Gamma'}$ is satisfiable. Then, return "Yes" if $\mathcal{C}_{\Gamma'}$ is not satisfiable and "No" otherwise.

Proof of Correctness: The correctness of the algorithm follows from previous discussion. $\square$

Example 3.8.38. Let p, q be two statement variables. We will show that $\{p, (p \to q)\} \models q$ using resolution. This amounts to proving that the set of formulas $\{p, (p \to q), (\neg q)\}$ is unsatisfiable. Putting these formulas into conjunctive normal form, we obtain the set of formulas $\Gamma' = \{p, ((\neg p) \vee q), (\neg q)\}$. The set of clauses $\mathcal{C}_{\Gamma'} = \{\{p\}, \{\neg p, q\}, \{\neg q\}\}$ is not satisfiable because we have the resolution tree shown in Figure 3.29. ▯

Let now $\Gamma = \{\varphi_0, \ldots, \varphi_n, \ldots\}$ be an infinite set of formulas and φ be a formula. By the Compactness Theorem for formulas, which we proved both semantically and syntactically, $\Gamma \models \varphi$ if and only if there is an n such that $\{\varphi_0, \ldots, \varphi_{n-1}\} \models \varphi$. This gives the following construction to determine if $\Gamma \models \varphi$.

Construction 3.8.39.
Input: An infinite set of formulas $\Gamma = \{\varphi_0, \ldots, \varphi_{n-1}, \ldots\}$ and a formula φ.
Output: "Yes" if $\Gamma \models \varphi$.
Method: For $n = 0, 1, \ldots,$ use Algorithm 3.8.37 to determine whether $\{\varphi_0, \ldots, \varphi_{n-1}\} \models \varphi$. If so, return "Yes", if not, increment n and repeat.

Proof of Correctness: The correctness of the construction follows from previous discussion. $\Box$

In calculating $\text{Res}^*(\mathcal{C})$, one often encounters tautologous clauses. We now show that these clauses can safely be ignored.

Theorem 3.8.40. *Let $\mathcal{C}$ be a finite set of clauses. Then, $\mathcal{C}$ is unsatisfiable if and only if there is a resolution proof of $\Box$ over $\mathcal{C}$ none of whose entries are tautologous.*

Proof. If there is a resolution proof of $\Box$ over $\mathcal{C}$ none of whose entries is tautologous, then by Theorem 3.8.19, $\Box \in \text{Res}^*(\mathcal{C})$, so by the Soundness Theorem (Theorem 3.8.29), $\mathcal{C}$ is unsatisfiable.

Conversely, we show by induction on n that if $|SV(\mathcal{C})| = n$ and $\mathcal{C}$ is unsatisfiable, then there is a resolution proof of $\Box$ over $\mathcal{C}$ none of whose entries is a tautologous. The proof exactly parallels the corresponding part of the proof of Theorem 3.8.30. If $|SV(\mathcal{C})| = 0$ and $\mathcal{C}$ is unsatisfiable, then $\mathcal{C} = \{\Box\}$, and the sequence $(\Box)$ is the desired proof.

Now suppose that $n \geq 0$ and that the result is true for all $\mathcal{C}$ with $|SV(\mathcal{C})| \leq n$. Let $\mathcal{C}$ be an unsatisfiable set of clauses with $|SV(\mathcal{C})| = n + 1$. Let p be an element of $SV(\mathcal{C})$. Then, $\mathcal{C}^p$ and $\mathcal{C}^{\bar{p}}$ are unsatisfiable, so, by inductive hypothesis, there are resolution proofs $(C_0, \ldots, C_{m-1})$ of $\Box$ over $\mathcal{C}^p$ and $(D_0, \ldots, D_{k-1})$ of $\Box$ over $\mathcal{C}^{\bar{p}}$ such that none of the C_is and D_is are tautologous.

Applying Lemma 3.8.28, we get resolution proofs $C'_0, \ldots, C'_{m-1}$ and $D'_0, \ldots, D'_{k-1}$ over $\mathcal{C}$ such that for each i, $0 \le i \le m-1$, C'_i is either C_i or $C_i \cup \{\bar{p}\}$, and, for each i, $0 \le i \le k-1$, D'_i is either D_i or $D_i \cup \{p\}$. Since clauses in $\mathcal{C}^p$ do not contain $\bar{p}$ and clauses in $\mathcal{C}^{\bar{p}}$ do not contain p, each C'_i and D'_i is also nontautologous. In the proof of Theorem 3.8.30, we showed that either $(C'_0, \ldots, C'_{m-1})$, $(D'_0, \ldots, D'_{k-1})$, or $(C'_0, \ldots, C'_{m-1}, D'_0, \ldots, D'_{k-1}, \square)$ is a resolution proof of $\square$ over $\mathcal{C}$, and in any of these cases, the proofs do not contain any tautologous clauses. $\square$

Definition 3.8.41. Let $\mathcal{C}$ be a set of clauses. Then, we define $\mathrm{NT}(\mathcal{C})$ to be $\{C \in \mathcal{C} \mid C \text{ is nontautologous}\}$ and

$$\begin{aligned}&\mathrm{NTRes}(\mathcal{C})\\&\quad = \mathcal{C} \cup \{R \mid R \text{ is a nontautologous resolvent of two clauses in } \mathcal{C}\}.\end{aligned}$$

▯

Just as with Res, we can now define $\mathrm{NTRes}^n(\mathcal{C}), \mathrm{NTRes}^*(\mathcal{C})$ for any set of clauses $\mathcal{C}$.

Theorem 3.8.42. *Let $\mathcal{C}$ be a set of clauses. Then,*

(1) *$NTRes(\mathcal{C}) \subseteq Res(\mathcal{C})$,*
(2) *$NTRes^n(\mathcal{C}) \subseteq Res^n(\mathcal{C})$ for each $n \in \mathbf{N}$, and $NTRes^*(\mathcal{C}) \subseteq Res^*(\mathcal{C})$,*
(3) *a clause C is in $NTRes^*(\mathcal{C})$ if and only if there is a resolution proof for C over $\mathcal{C}$ such that each entry in the proof is either in $\mathcal{C}$ or is a nontautologous resolvent of previous entries.*

Proof. The first part is immediate from the definition of NTRes.

We use induction to show that $\mathrm{NTRes}^n(\mathcal{C}) \subseteq \mathrm{Res}^n(\mathcal{C})$ for all $n \in \mathbf{N}$. The basis is immediate. Supposing that $\mathrm{NTRes}^n(\mathcal{C}) \subseteq \mathrm{Res}^n(\mathcal{C})$, we get, using the first part of this theorem and the second part of Theorem 3.8.15,

$$\begin{aligned}\mathrm{NTRes}^{n+1}(\mathcal{C}) &= \mathrm{NTRes}(\mathrm{NTRes}^n(\mathcal{C}))\\&\subseteq \mathrm{Res}(\mathrm{NTRes}^n(\mathcal{C}))\\&\subseteq \mathrm{Res}(\mathrm{Res}^n(\mathcal{C}))\\&= \mathrm{Res}^{n+1}(\mathcal{C}).\end{aligned}$$

This establishes the first claim in the second part of the theorem; the second claim follows immediately.

The third part of the theorem is proven in the same way that Theorem 3.8.19 is proven. $\square$

Theorem 3.8.43. *Let $\mathcal{C}$ be a finite set of clauses. Then, $\mathcal{C}$ is unsatisfiable if and only if $\square \in NTRes^*(NT(\mathcal{C}))$.*

Proof. By the second part of the previous theorem, $\mathrm{NTRes}^*(\mathrm{NT}(\mathcal{C})) \subseteq \mathrm{Res}^*(\mathrm{NT}(\mathcal{C}))$. Since $\mathrm{NT}(\mathcal{C}) \subseteq \mathcal{C}$, the last part of Theorem 3.8.17 shows that $\mathrm{Res}^*(\mathrm{NT}(\mathcal{C})) \subseteq \mathrm{Res}^*(\mathcal{C})$. Thus, if $\square \in \mathrm{NTRes}^*(\mathrm{NT}(\mathcal{C}))$, then $\square \in \mathrm{Res}^*(\mathcal{C})$, so, by Theorem 3.8.29, $\mathcal{C}$ is unsatisfiable.

Conversely, suppose that $\mathcal{C}$ is unsatisfiable. Then, by Theorem 3.8.40, there is a resolution proof $(C_0, \ldots, C_{n-1})$ of $\square$ over $\mathcal{C}$ such that no C_i is tautologous. Thus, each C_i is either in $\mathrm{NT}(\mathcal{C})$ or else is a nontautologous resolvent of previous entries. It follows from the third part of Theorem 3.8.42 that $\square \in \mathrm{NTRes}^*(\mathrm{NT}(\mathcal{C}))$. $\square$

3.9 Variations of Resolution

In Section 3.8, we saw that we can restrict resolution to nontautologous clauses without affecting its completeness. In this section, we examine other restrictions on resolution which preserve completeness.

Definition 3.9.1. Let $\mathcal{C}$ be a set of clauses. A *positive resolution proof* (*negative resolution proof*) over $\mathcal{C}$ is a resolution proof over $\mathcal{C}$ such that for every resolution step, there is a pair of premises such that one of the premises is positive (negative). ▯

Both positive resolution and negative resolution are complete as shown by the following theorem.

Theorem 3.9.2. *A finite set of clauses $\mathcal{C}$ is unsatisfiable if and only if there exists a positive resolution (negative resolution) proof of $\square$ over $\mathcal{C}$.*

Proof. Since every positive resolution (or negative resolution) proof is a resolution proof, if $\square$ has a positive resolution proof (a negative resolution proof) over $\mathcal{C}$, it is clear by soundness that $\mathcal{C}$

is unsatisfiable. Therefore, it remains to prove only that if $\mathcal{C}$ is an unsatisfiable finite set of clauses, then there is a positive resolution proof of $\square$ over $\mathcal{C}$ and, of course, a negative resolution proof of $\square$. We make the case here for the positive resolution proof. The negative resolution case is left to the reader.

The argument is by induction on $n = |SV(\mathcal{C})|$. If $n = 0$, then $\mathcal{C} = \{\square\}$ because $\mathcal{C}$ is unsatisfiable and the argument is complete.

Suppose that the statement holds for sets of clauses containing fewer than n distinct variables and let $\mathcal{C}$ be a set of clauses that contains n variables. Consider a positive literal ℓ such that either ℓ or $\bar{\ell}$ is in $\bigcup \mathcal{C}$. Both $\mathcal{C}^{\ell}$ and $\mathcal{C}^{\bar{\ell}}$ are unsatisfiable (by Theorem 3.8.26) and contain fewer than n variables. By the inductive hypothesis, there are positive resolution proofs $(C_0, \dots, C_{m-1})$ over $\mathcal{C}^{\ell}$ and $(D_0, \dots, D_{k-1})$ over $\mathcal{C}^{\bar{\ell}}$ of $\square$.

Starting from the positive resolution proof of $\square$ over $\mathcal{C}^{\bar{\ell}} = \{C - \{\ell\} \mid C \in \mathcal{C} \text{ and } \bar{\ell} \notin C\}$, by Lemma 3.8.28, there exists a resolution proof $(D'_0, \dots, D'_{k-1})$ over $\mathcal{C}$ such that for every i, $0 \leq i \leq k-1$, D'_i is either D_i or is $D_i \cup \{\ell\}$. Observe that the resolution proof $(D'_0, \dots, D'_{k-1})$ is positive because the positive clauses involved in the resolution steps of the proof $(D_0, \dots, D_{k-1})$ remain positive if ℓ is added. If $D'_{k-1} = \square$, we have obtained the desired positive resolution proof of $\square$ over $\mathcal{C}$. Otherwise, $D'_{k-1} = \{\ell\}$. Let $(C_{h_0}, \dots, C_{h_{p-1}})$ be the subsequence of the sequence $(C_0, \dots, C_{m-1})$ that contains the clauses C_i that do not belong to $\mathcal{C}$ and are not resolvents of previous clauses in the sequence $(C_0, \dots, C_{m-1})$. For each i, $0 \leq i \leq p-1$, there is a clause $C'_{h_i} \in \mathcal{C}$ such that $C'_{h_i} = C_{h_i} \cup \{\bar{\ell}\}$. Observe that $C_{h_i} = \mathrm{res}_{\ell}(\{\ell\}, C'_{h_i})$. Thus, we obtain the positive resolution proof

$$(D'_0, \dots, D'_{k-1}, C'_{h_0}, \dots, C'_{h_{p-1}}, C_0, \dots, C_{m-1})$$

of $\square$ over $\mathcal{C}^{\ell}$. $\square$

The following corollary shows that Theorem 3.9.2 can be extended to arbitrary sets of clauses (not necessarily finite).

Corollary 3.9.3. *A set of clauses $\mathcal{C}$ is unsatisfiable if and only if there exists a positive resolution (negative resolution) proof of $\square$ over $\mathcal{C}$.*

Proof. Again, by soundness, the existence of a positive (negative) resolution proof of $\square$ over $\mathcal{C}$ implies the unsatisfiability of $\mathcal{C}$.

Conversely, if $\mathcal{C}$ is unsatisfiable, then by the Compactness Theorem for Clauses (Theorem 3.8.11), there is a finite unsatisfiable subset $\mathcal{C}'$ of $\mathcal{C}$ and therefore by Theorem 3.9.2, there is a positive (negative) resolution proof of $\Box$ over $\mathcal{C}'$ which is also a positive (negative) resolution proof of $\Box$ over $\mathcal{C}$. $\Box$

Example 3.9.4. Consider the following set of clauses:

$$\mathcal{C} = \{\{p_0, p_1, \neg p_2\}, \{\neg p_0, \neg p_2\}, \{\neg p_1, \neg p_2\}, \{p_2\}\}.$$

The sequence $(C_0, \ldots, C_6)$ given in the following is a negative resolution proof of $\Box$:

Clause	Clause content	Derivation
C_0	$\{p_0, p_1, \neg p_2\}$	in $\mathcal{C}$
C_1	$\{\neg p_0, \neg p_2\}$	in $\mathcal{C}$
C_2	$\{\neg p_1, \neg p_2\}$	in $\mathcal{C}$
C_3	$\{p_2\}$	in $\mathcal{C}$
C_4	$\{p_0, \neg p_2\}$	$\text{res}_{p_1}(C_0, C_2)$
C_5	$\{\neg p_2\}$	$\text{res}_{p_0}(C_4, C_1)$
C_6	$\Box$	$\text{res}_{p_2}(C_3, C_5)$

According to Theorem 3.9.2, there exists a positive resolution proof for $\Box$. This is given by the following table:

Clause	Clause content	Derivation
C_0	$\{p_0, p_1, \neg p_2\}$	in $\mathcal{C}$
C_1	$\{\neg p_0, \neg p_2\}$	in $\mathcal{C}$
C_2	$\{\neg p_1, \neg p_2\}$	in $\mathcal{C}$
C_3	$\{p_2\}$	in $\mathcal{C}$
C_4	$\{\neg p_1\}$	$\text{res}_{p_2}(C_3, C_2)$
C_5	$\{\neg p_0\}$	$\text{res}_{p_2}(C_3, C_1)$
C_6	$\{p_0, p_1\}$	$\text{res}_{p_2}(C_3, C_0)$
C_7	$\{p_1\}$	$\text{res}_{p_0}(C_6, C_5)$
C_8	$\Box$	$\text{res}_{p_1}(C_7, C_4)$

The reader can easily verify that a positive clause is involved at every resolution step of the above proof. ▯

The soundness and completeness of positive (negative) resolution can be restated in terms of formal systems. To this end, we need the following definition.

Definition 3.9.5. The formal system $\mathcal{FRES}^{\text{pos}}$ is

$$\mathcal{FRES}^{\text{pos}} = (\mathcal{P}_{\text{fin}}(\text{LIT}), \emptyset, \{\,\mathsf{R}_{\text{pos}}\}),$$

where the binary rule R_{pos} consists of all pairs $((C, D), E)$ where E is a resolvent of C and D, and one of the clauses C, D is positive.

The formal system $\mathcal{FRES}^{\text{neg}}$ is $\mathcal{FRES}^{\text{neg}} = (\mathcal{P}_{\text{fin}}(\text{LIT}), \emptyset, \{\,\mathsf{R}_{\text{neg}}\})$, where the binary rule R_{neg} consists of all pairs $((C, D), E)$ where E is a resolvent of C and D, and one of the clauses C, D is negative. $\square$

Note that if $\mathcal{C}$ is a set of clauses, then a positive resolution proof over $\mathcal{C}$ is the same thing as a proof in the formal system $\mathcal{FRES}^{\text{pos}}_{\mathcal{C}}$ and a similar fact holds for negative resolution.

Theorem 3.9.2 can be rephrased in terms of formal systems by saying that if $\mathcal{C}$ is a set of clauses, then $\mathcal{C}$ is unsatisfiable if and only if $\square$ is a theorem of $\mathcal{FRES}^{\text{pos}}_{\mathcal{C}}$ (or a theorem of $\mathcal{FRES}^{\text{neg}}_{\mathcal{C}}$).

The introduction of a formal system allows us to make use of the idea of proof tree.

Definition 3.9.6. Let $\mathcal{C}$ be a set of clauses. A *positive (negative) resolution tree over* C is an $\mathcal{FRES}^{\text{pos}}_{\mathcal{C}}$-proof tree ($\mathcal{FRES}^{\text{neg}}_{\mathcal{C}}$-proof tree). $\square$

A positive (negative) resolution tree over $\mathcal{C}$ is a lot such that its leaves are labeled with clauses from $\mathcal{C}$ and each interior node is labeled with a clause that is a positive (negative) resolvent of the clauses which are labels of its two immediate descendents. By Theorem 1.8.23, we conclude that a clause C has a positive (negative) resolution proof over a set of clauses $\mathcal{C}$ if and only if there is a positive (negative) resolution tree over $\mathcal{C}$ such that C is the label of its root.

Example 3.9.7. In Figures 3.30 and 3.31, we give positive and negative resolution trees for $\square$ over the set of clauses $\mathcal{C}$ introduced in Example 3.9.4. $\square$

Definition 3.9.8. Let $\mathcal{C}$ be a set of clauses called the set of *input clauses*. A (C, k)*-based linear resolution proof* over $\mathcal{C}$ is a finite

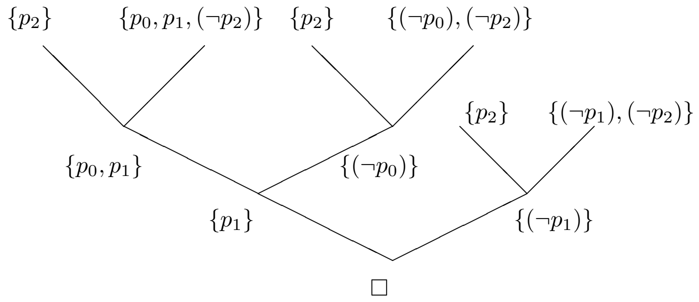


Fig. 3.30. Positive resolution tree for $\Box$.

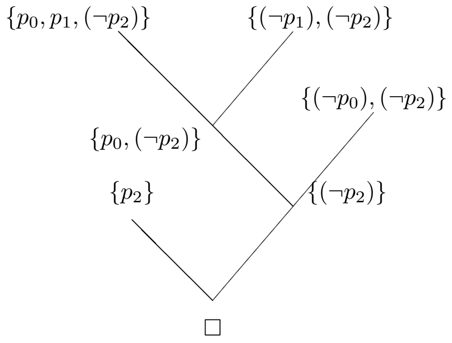

$\{p_0, p_1, (\neg p_2)\}$
$\{(\neg p_1), (\neg p_2)\}$
$\{(\neg p_0), (\neg p_2)\}$
$\{p_0, (\neg p_2)\}$
$\{p_2\}$
$\{(\neg p_2)\}$
$\Box$

Fig. 3.31. Negative resolution tree for $\Box$.

sequence of clauses $(C_0, C_1, \ldots, C_{n-1})$ such that the following conditions are satisfied:

(1) $0 \leq k \leq n-1$,
(2) $C_k = C$,
(3) for $0 \leq j \leq k$, $C_j \in \mathcal{C}$,
(4) for every h, such that $k < h \leq n-1$, C_h is a resolvent of C_{h-1} and C_m for some m, $m < h-1$.

For h with $k < h \leq n-1$, where $C_h = \text{res}_\ell(C_{h-1}, C_m)$, we shall refer to C_{h-1} as the *center clause* and to C_m as a *side clause* of that step.

A *C-based linear resolution proof of a clause D* over $\mathcal{C}$ is a (C,k)-based linear resolution proof over $\mathcal{C}$ for some k whose last entry is D. A *linear resolution proof of a clause D* is a C-based linear resolution proof of D, for some clause C. ▯

Example 3.9.9. Let $\mathcal{C}$ be the set of clauses considered in Example 3.9.4. There exists a $(\{p_2\}, 3)$-based linear resolution proof of $\square$ over the same set of clauses $\mathcal{C}$. This proof is given by the following table:

Clause	Clause content	Derivation
C_0	$\{p_0, p_1, \neg p_2\}$	in $\mathcal{C}$
C_1	$\{\neg p_0, \neg p_2\}$	in $\mathcal{C}$
C_2	$\{\neg p_1, \neg p_2\}$	in $\mathcal{C}$
C_3	$\{p_2\}$	in $\mathcal{C}$
C_4	$\{\neg p_1\}$	$\mathrm{res}_{p_2}(C_3, C_2)$
C_5	$\{p_0, \neg p_2\}$	$\mathrm{res}_{\overline{p_1}}(C_4, C_0)$
C_6	$\{\neg p_2\}$	$\mathrm{res}_{p_0}(C_5, C_1)$
C_7	$\square$	$\mathrm{res}_{\overline{p_2}}(C_6, C_3)$

The proof tree that corresponds to this proof is given in Figure 3.32. ∎

Lemma 3.9.10. *Let $(C_0, \ldots, C_k, \ldots, C_{n-1})$ be a (C, k)-based linear resolution proof over $\mathcal{C}$. If $(D_0, \ldots, D_l, \ldots, D_{m-1})$ is a (D, l)-based linear resolution proof of a clause E over a set of clauses $\mathcal{D}$ (where $E = D_{m-1}$ and $C_{n-1} = D_l$), then the sequence*

$$(D_0, \ldots, D_{l-1}, C_0, \ldots, C_k, C_{k+1}, \ldots, C_{n-1}, D_{l+1}, \ldots, D_{m-1})$$

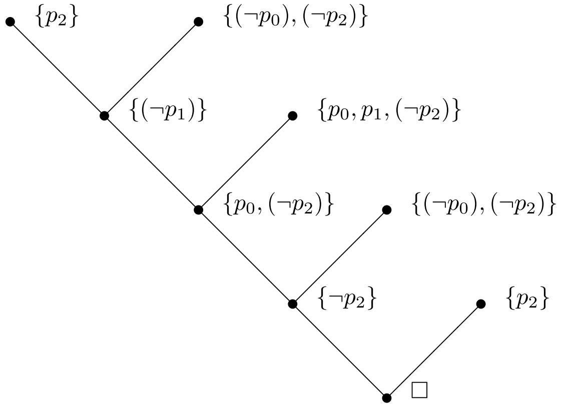

$\{p_2\}$
$\{(\neg p_0), (\neg p_2)\}$
$\{(\neg p_1)\}$
$\{p_0, p_1, (\neg p_2)\}$
$\{p_0, (\neg p_2)\}$
$\{(\neg p_0), (\neg p_2)\}$
$\{\neg p_2\}$
$\{p_2\}$
$\square$

Fig. 3.32. Resolution tree of the C_4-based linear resolution proof.

is a $(C, l+k)$-based linear resolution proof of E over $\mathcal{C} \cup \mathcal{D}$. Further, if none of the clauses D_j, $0 \leq j \leq l-1$, equals D, then the above sequence is a $(C, l+k)$-based linear resolution proof over $\mathcal{C} \cup (\mathcal{D} - \{D\})$.

Proof. The argument consists of a direct verification of the satisfaction of the requirements of Definition 3.9.8 and is left to the reader. □

Definition 3.9.11. A set of clauses $\mathcal{C}$ is *minimally unsatisfiable* if it is unsatisfiable and for every clause $C \in \mathcal{C}$, $\mathcal{C} - \{C\}$ is a satisfiable set of clauses. ▯

Note that if $\mathcal{D}$ is a finite unsatisfiable set of clauses, then $\mathcal{D}$ contains a subset of clauses which is minimally unsatisfiable. Also, observe that no minimally unsatisfiable set of clauses contains a tautologous clause.

Lemma 3.9.12. *Let $\mathcal{C}$ be a finite set of clauses and let E be a clause from $\mathcal{C}$. If $\mathcal{C}$ is unsatisfiable and $\mathcal{C} - \{E\}$ is satisfiable, then there exists an E-based linear resolution proof of $\square$ over $\mathcal{C}$.*

Proof. The argument is by course-of-values induction on the number n of variables that occur in $\mathcal{C}$.

Suppose that the statement holds for sets of clauses with fewer than n variables and let $\mathcal{C}$ be a set of clauses that contains n variables. Observe that E cannot be a tautological clause. We consider three cases: $E = \square$, $|E| = 1$, and $|E| > 1$.

The first case is trivial.

Suppose that $|E| = 1$ and let $E = \{\ell\}$. Since $\mathcal{C}$ is unsatisfiable, so is $\mathcal{C}^{\ell} = \{C - \{\bar{\ell}\} | C \in \mathcal{C} \text{ and } \ell \notin C\}$, by Theorem 3.8.26. Let $\mathcal{D}$ be a minimally unsatisfiable set of clauses contained in $\mathcal{C}^{\ell}$. Note that the set $\mathcal{C}^{\ell}$ contains fewer than n variables and so does $\mathcal{D}$.

There is a clause $D \in \mathcal{D}$ such that $D \notin \mathcal{C}$. Indeed, if this were not the case, then $\mathcal{D}$ would be a subset of $\mathcal{C} - \{\{\ell\}\} = \mathcal{C} - \{E\}$ and, thus, it would be satisfiable. For this clause D, we have $D \cup \{\bar{\ell}\} \in \mathcal{C}$. Since $\mathcal{D}$ is minimally unsatisfiable, $\mathcal{D} - \{D\}$ is satisfiable. By the inductive hypothesis, there exists a linear (D, l)-based resolution proof $\varpi = (D_0, \ldots, D_l, \ldots, D_{m-1})$ of $\square$ over $\mathcal{D}$. Since $\mathcal{D} \subseteq \mathcal{C}^{\ell}$, by Lemma 3.8.28, there exists a resolution proof $\varpi' = (D'_0, \ldots, D'_l, \ldots, D'_{m-1})$ over $\mathcal{C}$, where D'_i is either equal to D_i or $D'_i = D_i \cup \{\bar{\ell}\}$ for $0 \leq i \leq m-1$. Since D_l is an input step of ϖ, D'_l is an input step of ϖ' which implies $D'_l = D_l \cup \{\bar{\ell}\}$ because $D_l = D \notin \mathcal{C}$.

Next, we show that there is a sequence $(D''_0, \ldots, D''_{m-1})$ of clauses such that

(1) $(D''_0, \ldots, D''_l, E, D_l, D''_{l+1}, \ldots, D''_{m-1})$ is an $(E, l+1)$-based linear resolution proof over $\mathcal{C}$,
(2) $D''_k = D_k$ or $D''_k = D_k \cup \{\bar{\ell}\}$ for $0 \leq k \leq m-1$.

Observe first that by the P-Lifting Lemma, each D'_j with $0 \leq j \leq l$ is in $\mathcal{C}$ and is equal either to D_j or to $D_j \cup \{\bar{\ell}\}$. Thus, we can define $D''_k = D'_k$ for $0 \leq k \leq l$. Further, $D'_l = D_l \cup \{\bar{\ell}\} = D \cup \{\bar{\ell}\}$ and $E = \{\ell\} \in \mathcal{C}$, so $D_l = \mathrm{res}_\ell(E, D'_l)$.

We show by induction on k with $l+1 \leq k \leq m-1$ that there is a clause D''_k such that $D''_k = D_k$ or $D''_k = D_k \cup \{\bar{\ell}\}$ and the sequence $(D'_0, \ldots, D'_l, E, D_l, D''_{l+1}, \ldots, D''_k)$ is an $(E, l+1)$-based linear resolution proof.

For the basis step, $k = l+1$, we have $D_{l+1} = \mathrm{res}_{\ell_{l+1}}(D_l, D_i)$ with $0 \leq i \leq l$ and ℓ_{l+1} cannot be equal to either ℓ or $\bar{\ell}$. Define $D''_{l+1} = \mathrm{res}_{\ell_{l+1}}(D_l, D'_i)$. Note that this definition is correct because $\mathrm{res}_{\ell_{l+1}}(D_l, D_i)$ is defined and $D_i \subseteq D'_i$. Furthermore, we have $D'_i = D_i \cup F_i$, where $F_i = \emptyset$ or $F_i = \{\bar{\ell}\}$. By Lemma 3.8.27,

$$\begin{aligned} D''_{l+1} &= \mathrm{res}_{\ell_{l+1}}(D_l, D_i \cup F_i) \\ &= \mathrm{res}_{\ell_{l+1}}(D_l, D_i) \cup F_i \\ &= D_{l+1} \cup F_i \end{aligned}$$

because $\ell_{l+1} \neq \ell$.

For the inductive step, suppose that for $l+1 \leq k < m-1$, we have the desired sequence of clauses $D''_{l+1}, \ldots, D''_k$. Note that for some i, $0 \leq i \leq k$, $D_{k+1} = \mathrm{res}_{\ell_{k+1}}(D_k, D_i)$. Define $D''_{k+1} = \mathrm{res}_{\ell_{k+1}}(D''_k, D''_i)$. As before, this definition is correct because $D_k \subseteq D''_k$ and $D_i \subseteq D''_i$. Further, we have $D''_k = D_k \cup E_k$, $D''_i = D_i \cup F_i$, where $E_k, F_i \in \{\emptyset, \{\bar{\ell}\}\}$

and ℓ_{k+1} is distinct from ℓ and $\bar{\ell}$. This allows us to write

$$\begin{aligned} D''_{k+1} &= \mathrm{res}_{\ell_{k+1}}(D_k \cup E_k, D_i \cup F_i) \\ &= \mathrm{res}_{\ell_{k+1}}(D_k, D_i) \cup E_k \cup F_i \\ &= D_{l+1} \cup E_k \cup F_i \end{aligned}$$

by Lemma 3.8.27.

We thus have a resolution proof for $\square$ or for $\{\bar{\ell}\}$. In the first case, we have found the desired resolution proof. In the second, an extra step is necessary to obtain $\square$ by resolving E and D''_{m-1}.

Suppose now that $|E| > 1$ and let ℓ be a literal from E. Note that E does not contain $\bar{\ell}$ because E is not tautologous. Therefore, if $E' = E - \{\ell\}$, then E' belongs to the unsatisfiable set of clauses $\mathcal{C}^{\bar{\ell}} = \{C - \{\ell\} | C \in \mathcal{C} \text{ and } \bar{\ell} \notin C\}$.

Note also that $E' \notin \mathcal{C}$ because if E' were in $\mathcal{C}$, then $E' \in \mathcal{C} - \{E\}$ and a truth assignment satisfying $\mathcal{C} - \{E\}$ would also satisfy $\mathcal{C}$ since $E' \subseteq E$.

The set $\mathcal{C}^{\bar{\ell}} - \{E'\}$ is satisfiable. Indeed, let v be a truth assignment that satisfies $\mathcal{C} - \{E\}$. Note that v does not satisfy the clause E since otherwise v would satisfy $\mathcal{C}$. Therefore, $v(\ell) = \mathbf{F}$ because $\ell \in E$. This implies that v satisfies $\mathcal{C}^{\bar{\ell}} - \{E'\}$.

Let $\mathcal{E}$ be a minimally unsatisfiable subset of $\mathcal{C}^{\bar{\ell}}$. Clearly, we have $E' \in \mathcal{E}$ because, otherwise we would have $\mathcal{E} \subseteq \mathcal{C}^{\bar{\ell}} - \{E'\}$ and this would mean that $\mathcal{E}$ is satisfiable. By the inductive hypothesis, there is an E'-based linear resolution proof of $\square$ over $\mathcal{E}$. By applying Lemma 3.8.28, we obtain the existence of an E-based linear resolution proof of $\{\ell\}$ or $\square$ over $\mathcal{C}$. In fact, we will have an E-based linear resolution proof of $\{\ell\}$ because $\ell \in E$, and the P-Lifting Lemma preserves the literals used in the resolutions.

Observe that $(\mathcal{C} - \{E\}) \cup \{\{\ell\}\}$ is unsatisfiable and $(\mathcal{C} - \{E\})$ is satisfiable. Indeed, as we saw before, for any truth assignment v that satisfies $\mathcal{C} - \{E\}$ we have $v(\ell) = \mathbf{F}$. From the first case (dealing with clauses that contain one literal), we obtain the existence of an $\{\ell\}$-based linear resolution proof of $\square$ over $(\mathcal{C} - \{E\}) \cup \{\{\ell\}\}$, where we can assume without loss of generality that $\{\ell\}$ occurs only once as an input step. By Lemma 3.9.10 applied to these proofs, we obtain the needed E-based resolution proof of $\square$. $\square$

Theorem 3.9.13. *A finite set of clauses $\mathcal{C}$ is unsatisfiable if and only if there exists a linear resolution proof of $\Box$ over $\mathcal{C}$.*

Proof. Let $\mathcal{C}$ be a finite unsatisfiable set of clauses. If $\mathcal{D}$ is a minimally unsatisfiable subset of $\mathcal{C}$, then $\mathcal{D} \neq \emptyset$. Let D be a clause in $\mathcal{D}$. By Lemma 3.9.12, there exists a D-based linear resolution proof of $\Box$ over $\mathcal{D}$. Clearly, this is also a D-based linear resolution proof of $\Box$ over $\mathcal{C}$.

The converse implication is obvious since every linear resolution proof is a resolution proof. $\Box$

Corollary 3.9.14. *A set of clauses $\mathcal{C}$ is unsatisfiable if and only if there exists a linear resolution proof of $\Box$ over $\mathcal{C}$.*

Proof. By soundness, the existence of a linear resolution proof of $\Box$ over $\mathcal{C}$ implies the unsatisfiability of $\mathcal{C}$.

Conversely, if $\mathcal{C}$ is unsatisfiable, then by the Compactness Theorem for Clauses (Theorem 3.8.11), there is a finite unsatisfiable subset $\mathcal{C}'$ of $\mathcal{C}$ and therefore by Theorem 3.9.13, there is a linear resolution proof of $\Box$ over $\mathcal{C}'$ which is also a linear resolution proof of $\Box$ over $\mathcal{C}$. $\Box$

A further restriction can be placed on linear resolutions by demanding that all side clauses should be input clauses.

Definition 3.9.15. A (C, k)*-based input resolution proof* over a set of input clauses $\mathcal{C}$ is a (C, k)-based linear resolution proof $(C_0, C_1, \ldots, C_{n-1})$ such that the following additional condition is satisfied: for every h such that $k < h \leq n-1$, C_h is a resolvent of C_{h-1} and C_m for some m, $m \leq k$.

A (C, k)*-based input resolution proof of a clause* D over $\mathcal{C}$ is a (C, k)-based input resolution proof over $\mathcal{C}$ whose last entry is D.

An *input resolution proof of a clause* D over $\mathcal{C}$ is a (C, k)-based input resolution proof over $\mathcal{C}$ whose last entry is D for some C and k. ▯

Note that in an input resolution proof, the side clause must be an input clause. In the more general case of linear resolution, the side clause could be any predecessor of the center clause.

Input resolution is not complete as shown by the following example.

Example 3.9.16. Consider the input set

$$\mathcal{C} = \{\{p_0, p_1\}, \{p_0, \neg p_1\}, \{\neg p_0, p_1\}, \{\neg p_0, \neg p_1\}\}.$$

A linear C_3-based linear resolution proof of $\square$ is given in the following table:

Clause	Clause content	Derivation
C_0	$\{p_0, p_1\}$	in $\mathcal{C}$
C_1	$\{p_0, \neg p_1\}$	in $\mathcal{C}$
C_2	$\{\neg p_0, p_1\}$	in $\mathcal{C}$
C_3	$\{\neg p_0, \neg p_1\}$	in $\mathcal{C}$
C_4	$\{\neg p_0\}$	$\mathrm{res}_{\overline{p_1}}(C_3, C_2)$
C_5	$\{\neg p_1\}$	$\mathrm{res}_{\overline{p_0}}(C_4, C_1)$
C_6	$\{p_0\}$	$\mathrm{res}_{\overline{p_1}}(C_5, C_0)$
C_7	$\square$	$\mathrm{res}_{p_0}(C_6, C_4)$

It is clear that this is not an input resolution proof because in the last step, we use no input clause.

If $\square$ is obtained as a resolvent of two clauses, these clauses must contain one literal each. Since $\mathcal{C}$ contains no one-literal clauses, it is clear that $\square$ cannot be obtained by an input resolution proof over $\mathcal{C}$. ▯

The definition of Horn clause implies that $\square$ is a Horn clause since the defining condition of such a clause is vacuously satisfied. Moreover, $\square$ is both a positive and a negative clause.

Lemma 3.9.17. *If a set $\mathcal{C}$ of Horn clauses is unsatisfiable, then $\mathcal{C}$ must contain at least one positive unit clause or $\square$, and one negative clause.*

Proof. If v is a truth assignment such that $v(p) = \mathbf{T}$ for every $p \in SV(\mathcal{C})$, then v satisfies every clause that contains a positive literal. On the other hand, if v_1 is a truth assignment such that $v_1(p) = \mathbf{F}$ for every $p \in SV(\mathcal{C})$, then v_1 satisfies every clause that contains some negative literal. Therefore, if $\mathcal{C}$ is unsatisfiable, it may not consist exclusively of clauses that contain some positive literal and it may not consist exclusively of clauses that contain some negative literal. In other words, $\mathcal{C}$ must contain a negative clause and either a positive unit clause or $\square$. □

Theorem 3.9.18. *If $\mathcal{C}$ is a finite unsatisfiable set of Horn clauses and C is a negative clause of $\mathcal{C}$ such that $\mathcal{C} - \{C\}$ is satisfiable, then there exists a C-based input resolution proof of $\Box$ over $\mathcal{C}$.*

Proof. By Lemma 3.9.12, there exists a (C, k)-based linear resolution proof over $\mathcal{C}$ of $\Box$, $\varpi = (C_0, \ldots, C_k, C_{k+1}, \ldots, C_{n-1})$. Note that in this proof a negative clause can be resolved only against a clause that contains a positive literal and the resulting clause is going to be again a negative clause because $\mathcal{C}$ consists of Horn clauses. Since $C_k = C$ is a negative clause, all its successors in this sequence must be negative clauses and the side clauses must all contain a positive literal. Therefore, the side clauses must be input clauses and the above resolution proof is an input resolution proof. $\Box$

Corollary 3.9.19. *Let $\mathcal{C}$ be a set of Horn clauses. Then, $\mathcal{C}$ is unsatisfiable if and only if there is an input resolution proof of $\Box$ over $\mathcal{C}$.*

Proof. Suppose that $\mathcal{C}$ is an unsatisfiable set of Horn clauses. By the Compactness Theorem for Clauses (Theorem 3.8.11), there is a finite unsatisfiable subset $\mathcal{C}'$ of $\mathcal{C}$. Let $\mathcal{C}''$ be a minimally unsatisfiable subset of $\mathcal{C}'$. By Lemma 3.9.17, $\mathcal{C}''$ contains a negative clause C. By Theorem 3.9.18, there is a C-based input resolution proof of $\Box$ over $\mathcal{C}''$, thus over $\mathcal{C}$.

As usual, the converse follows by soundness. $\Box$

Definition 3.9.20. Let $\mathcal{C}$ be a set of clauses and let v be a truth assignment. A *v-semantic resolution proof over $\mathcal{C}$* is a resolution proof over $\mathcal{C}$ such that if i is a resolution step, then there are premises C_j, C_k for step i such that v falsifies at least one of the clauses C_j, C_k. $\Box$

Theorem 3.9.21. *Let $\mathcal{C}$ be a finite unsatisfiable set of clauses. For every truth assignment v, there exists a v-semantic resolution proof over $\mathcal{C}$ of $\Box$.*

Proof. The argument is by induction on $n = |SV(\mathcal{C})|$. If $n = 0$, we have $\mathcal{C} = \{\Box\}$ and $(\Box)$ is a v-semantic resolution proof of $\Box$ that has no resolution steps.

Suppose that the statement holds for finite unsatisfiable sets of clauses with fewer than $n \geq 1$ variables and let $\mathcal{C}$ be a finite unsatisfiable set of clauses with $|SV(\mathcal{C})| = n$. Let ℓ be a literal such that either ℓ or $\bar{\ell}$ occurs in $\mathcal{C}$ and $v(\ell) = \mathbf{T}$. By Theorem 3.8.26, both

$\mathcal{C}^{\ell}$ and $\mathcal{C}^{\bar{\ell}}$ are unsatisfiable and $|SV(\mathcal{C}^{\ell})| = |SV(\mathcal{C}^{\bar{\ell}})| \leq n-1$. By the inductive hypothesis, there exist v-semantic resolution proofs of $\Box$ over both $\mathcal{C}^{\ell}$ and $\mathcal{C}^{\bar{\ell}}$, namely, $(C_0, \ldots, C_{k-1})$ and $(D_0, \ldots, D_{l-1})$, respectively, where $C_{k-1} = D_{l-1} = \Box$.

Starting from the v-semantic resolution proof $(C_0, \ldots, C_{k-1})$ of $\Box$ over $\mathcal{C}^{\ell} = \{C - \{\bar{\ell}\} \mid C \in \mathcal{C} \text{ and } \ell \notin C\}$, there exists a resolution proof $(C'_0, \ldots, C'_{k-1})$ over $\mathcal{C}$ which satisfies the properties given by Lemma 3.8.28, and a similar resolution proof $(D'_0, \ldots, D'_{l-1})$ over $\mathcal{C}$ can be constructed starting from $(D_0, \ldots, D_{l-1})$. Since C'_i is either C_i or $C_i \cup \{\bar{\ell}\}$ and $v(\bar{\ell}) = \mathbf{F}$, the lifted proof $(C'_0, \ldots, C'_{k-1})$ is a v-semantic proof. Note that C'_{k-1} is either $\Box$ or $\{\bar{\ell}\}$. In the first case, we have the desired v-semantic proof of $\Box$ over $\mathcal{C}$, so we may assume that $C'_{k-1} = \{\bar{\ell}\}$.

Suppose that the input steps of the resolution proof $(D_0, \ldots, D_{l-1})$ over $\mathcal{C}^{\bar{\ell}}$ are $D_{k_0}, \ldots, D_{k_{m-1}}$ for some $m \in \mathbf{N}$, $m \geq 1$. We claim that

$$(C'_0, \ldots, C'_{k-1} = \{\bar{\ell}\}, D'_{k_0}, \ldots, D'_{k_{m-1}}, D_0, \ldots, D_{l-1}) \tag{3.9}$$

is a v-semantic resolution proof of $\Box$ over $\mathcal{C}$. Note that $D'_{k_0}, \ldots, D'_{k_{m-1}}$ are in $\mathcal{C}$ (and so are input steps) by the P-Lifting lemma because $D_{k_0}, \ldots, D_{k_{m-1}}$ are in $\mathcal{C}^{\bar{\ell}}$. Each j with $0 \leq j \leq l-1$ falls into one of the following three cases:

- If $D_j \in \mathcal{C}$, then j is an input step.
- If $D_j \in \mathcal{C}^{\bar{\ell}} - \mathcal{C}$, then j is an input step in the proof $(D_0, \ldots, D_{l-1})$ and $D'_j = D_j \cup \{\ell\}$. Thus, $D_j = \text{res}_{\ell}(D'_j, \{\bar{\ell}\})$ and both D'_j and $\{\bar{\ell}\}$ occur previously in the proof. Further, v falsifies $\{\bar{\ell}\}$.
- If D_j is neither in $\mathcal{C}$ nor in $\mathcal{C}^{\bar{\ell}}$, then j is a resolution step in the proof $(D_0, \ldots, D_{l-1})$ over $\mathcal{C}^{\bar{\ell}}$ and hence is a resolution step in the proof (3.9), and v falsifies one of the premises of this step. $\Box$

Corollary 3.9.22. *Let $\mathcal{C}$ be a set of clauses and let v be a truth assignment. There exists a v-semantic resolution proof over $\mathcal{C}$ of $\Box$ if and only if $\mathcal{C}$ is unsatisfiable.*

Proof. By soundness, the existence of a v-semantic resolution proof of $\Box$ over $\mathcal{C}$ implies the unsatisfiability of $\mathcal{C}$.

Conversely, if $\mathcal{C}$ is unsatisfiable, then by the Compactness Theorem for Clauses (Theorem 3.8.11), there is a finite unsatisfiable subset

$\mathcal{C}'$ of $\mathcal{C}$ and therefore by Theorem 3.9.21, there is a v-semantic resolution proof of $\Box$ over $\mathcal{C}'$ which is also a v-semantic resolution proof of $\Box$ over $\mathcal{C}$. $\Box$

Definition 3.9.23. Let $\mathcal{C}$ be a set of clauses. $\mathcal{D}$ is a *set-of-support* for $\mathcal{C}$ if $\mathcal{D} \subseteq \mathcal{C}$ and $\mathcal{C} - \mathcal{D}$ is satisfiable.

Let $\mathcal{D}$ be a set-of-support for $\mathcal{C}$. A *$\mathcal{D}$-set-of-support resolution proof over $\mathcal{C}$* is a resolution proof over $\mathcal{C}$ such that for every resolution step there is a pair of premises at least one of which does not belong to $\mathcal{C} - \mathcal{D}$. ▯

Theorem 3.9.24. *Let $\mathcal{C}$ be a set of clauses and $\mathcal{D}$ be a set-of-support for $\mathcal{C}$. Then, there exists a $\mathcal{D}$-set-of-support resolution proof of $\Box$ over $\mathcal{C}$ if and only if $\mathcal{C}$ is unsatisfiable.*

Proof. By soundness, the existence of a $\mathcal{D}$-set-of-support resolution proof of $\Box$ over $\mathcal{C}$ implies the unsatisfiability of $\mathcal{C}$.

Suppose that $\mathcal{C}$ is unsatisfiable. Since $\mathcal{D}$ is a set-of-support for $\mathcal{C}$, $\mathcal{C} - \mathcal{D}$ is satisfiable. Thus, there exists a truth assignment v such that v satisfies all clauses from $\mathcal{C} - \mathcal{D}$. By Corollary 3.9.22, there exists a v-semantic resolution proof of $\Box$ over $\mathcal{C}$. Note that in every resolution step of this proof, there is a premise C that is falsified by v, that is a clause not from $\mathcal{C} - \mathcal{D}$. Therefore, this proof is a $\mathcal{D}$ set-of-support proof of $\Box$ over $\mathcal{C}$. $\Box$

We introduce now a generalization of positive resolution.

Definition 3.9.25. A positive clause C is a *hyperresolvent* of a sequence of clauses $C_1, \ldots, C_n, D$, where $n \geq 1$ and $C_1, \ldots, C_n$ are positive, if there is a sequence of clauses $E_0, \ldots, E_n$ such that the following conditions are satisfied:

(1) $E_0 = C$ and $E_n = D$,
(2) for $0 \leq i \leq n-1$, E_i is a resolvent of C_{i+1} and E_{i+1}. ▯

In a positive resolution tree, as we follow the path from a nonpositive clause toward the root of the tree, we see the negative literals disappearing one at a time until we reach a positive clause. The idea of hyperresolution is to collapse this "annihilation" of negative literals into a single step.

Definition 3.9.26. Let $\mathcal{C}$ be a set of clauses. A *hyperresolution proof* over $\mathcal{C}$ is a finite sequence $(C_0, C_1, \ldots, C_{n-1})$ of clauses such that

$n \geq 1$ and for each step i, $0 \leq i \leq n-1$ either $C_i \in \mathcal{C}$ or else $C_i \notin \mathcal{C}$ and there is a sequence $j_0, \ldots, j_{p-1} < i$ such C_i is a hyperresolvent of $C_{j_0}, \ldots, C_{j_{p-1}}$. In the first case, i is an *input step* of the proof; in the second case, i is a *hyperresolution step.*

A *hyperresolution proof of a clause* C over $\mathcal{C}$ is a hyperresolution proof over $\mathcal{C}$ whose last entry is C. ▯

The next two statements will help us prove the soundness of hyperresolution.

Lemma 3.9.27. *If E is a hyperresolvent of $(C_1, \ldots, C_n, D)$, then there is a positive resolution proof of E over $\{C_1, \ldots, C_n, D\}$.*

Proof. We use the notations of Definition 3.9.25 and prove by induction on j with $0 \leq j \leq n$ that E_{n-j} has a positive resolution proof over $\{C_1, \ldots, C_n, D\}$. For $j = 0$, this is immediate because $E_n = D$. Suppose that $0 \leq j < n$ and the result is true for j. Since E_{n-j-1} is a positive resolvent of C_{n-j} and E_{n-j}, a positive proof for E_{n-j-1} over $\{C_1, \ldots, C_n, D\}$ is obtained by taking the positive proof of E_{n-j} and appending C_{n-j} and E_{n-j-1}. Since $E = E_0$, this concludes the argument. □

Lemma 3.9.28. *If there is a hyperresolution proof of a clause C over a set of clauses $\mathcal{C}$, then there is a positive resolution proof of C over $\mathcal{C}$.*

Proof. We provide an argument by course-of-values induction on the length of the hyperresolution proof. Suppose the statement holds for proofs of length less than m and C has a hyperresolution proof of length m. If $C \in \mathcal{C}$, the conclusion is immediate. Otherwise, C is a hyperresolvent of clauses $(K_1, \ldots, K_n, D)$ which have shorter hyperresolution proofs over $\mathcal{C}$. By Lemma 3.9.27, C has a positive resolution proof over $\{K_1, \ldots, K_n, D\}$. A positive resolution proof of C over $\mathcal{C}$ can be obtained by prepending the positive resolution proofs of $K_1, \ldots, K_n, D$ over $\mathcal{C}$ (which exist by inductive hypothesis) to the positive proof of C over $\{K_1, \ldots, K_n, D\}$. □

Theorem 3.9.29. *Let $\mathcal{C}$ be a set of clauses. If there is a hyperresolution proof of $\square$ over $\mathcal{C}$, then $\mathcal{C}$ is unsatisfiable.*

Proof. If there is a hyperresolution proof of $\square$ over $\mathcal{C}$, then there is a (positive) resolution proof of $\square$ over $\mathcal{C}$ by Lemma 3.9.28, which implies that $\mathcal{C}$ is unsatisfiable, by Theorem 3.8.29. $\square$

Example 3.9.30. We give a hyperresolution proof which shows that the set of clauses

$$\mathcal{C} = \{\{p_0, p_1, \neg p_2\}, \{\neg p_0, \neg p_2\}, \{\neg p_1, \neg p_2\}, \{p_2\}\}$$

is unsatisfiable. This proof was obtained by regrouping the steps of the positive resolution proof of $\square$ given in Example 3.9.4.

Clause	Clause content	Source
C_0	$\{p_0, p_1, \neg p_2\}$	In $\mathcal{C}$
C_1	$\{\neg p_0, \neg p_2\}$	In $\mathcal{C}$
C_2	$\{\neg p_1, \neg p_2\}$	In $\mathcal{C}$
C_3	$\{p_2\}$	In $\mathcal{C}$
C_4	$\{p_0, p_1\}$	Hyperresolvent of (C_3, C_0)
C_5	$\{p_1\}$	Hyperresolvent of (C_4, C_3, C_1)
C_6	$\square$	Hyperresolvent of (C_5, C_3, C_2)

$\square$

The following lemma allows us to prove the completeness of hyperresolution.

Lemma 3.9.31. *Let $\mathcal{C}$ be a set of clauses and C be a positive clause. If there is positive resolution proof of C over $\mathcal{C}$, then there is a hyperresolution proof of C over $\mathcal{C}$.*

Proof. We prove by course-of-values induction on the length of the positive resolution proof. Suppose that a positive clause C has a positive resolution proof of length n over $\mathcal{C}$ and the result holds for all positive clauses that have shorter proofs. If $C \in \mathcal{C}$, the conclusion is immediate. Otherwise, C is obtained by positive resolution from a predecessor positive clause C_1 in the proof and a predecessor clause E_1. By inductive hypothesis, there is a hyperresolution proof ϖ_1 of C_1 over $\mathcal{C}$. If $E_1 \notin \mathcal{C}$, then E_1 is obtained by positive resolution from a predecessor positive clause C_2 and a predecessor clause E_2. Continuing in this way, we obtain predecessor positive clauses $C_1, \ldots, C_n$ with hyperresolution proofs $\varpi_1, \ldots, \varpi_n$ over $\mathcal{C}$ and a clause $E_n \in \mathcal{C}$ such that C is obtained by hyperresolution from $C_1, \ldots, C_n, E_n$. Thus, $\varpi_1 \cdots \varpi_n \cdot (E_n, C)$ is a hyperresolution proof of C over $\mathcal{C}$. $\square$

Theorem 3.9.32. *Let $\mathcal{C}$ be a set of clauses. If $\mathcal{C}$ is unsatisfiable, then there is a hyperresolution proof of $\square$ over $\mathcal{C}$.*

Proof. By the completeness of positive resolution established in Corollary 3.9.3, there exists a positive proof of $\square$ over $\mathcal{C}$. Therefore, there is a hyperresolution proof of $\square$ over $\mathcal{C}$ by Lemma 3.9.31. $\square$

Next, we introduce a formal system for hyperresolution. Because the number of premises for hyperresolution is not fixed, the system will have infinitely many rules.

Definition 3.9.33. The formal system $\mathcal{FRES}^{\text{hyper}}$ is

$$\mathcal{FRES}^{\text{hyper}} = (\mathcal{P}_{\text{fin}}(\text{LIT}), \emptyset, \{\, \mathsf{R}^n_{\text{hyper}} \mid n \geq 1\}),$$

where the $(n+1)$-ary rule $\mathsf{R}^n_{\text{hyper}}$ consists of all pairs $((C_1, \ldots, C_n, D), E)$, where E is a hyperresolvent of $C_1, \ldots, C_n$ and D. ▯

Note that if $\mathcal{C}$ is a set of clauses, then a hyperresolution proof over $\mathcal{C}$ is the same thing as a proof in the formal system $\mathcal{FRES}^{\text{hyper}}_{\mathcal{C}}$.

In terms of formal systems, Theorems 3.9.29 and 3.9.32 can be rephrased by saying that if $\mathcal{C}$ is a set of clauses, then $\mathcal{C}$ is unsatisfiable if and only if $\square$ is a theorem of $\mathcal{FRES}^{\text{hyper}}_{\mathcal{C}}$.

As before, the introduction of a formal system allows us to make use of the idea of proof tree.

Definition 3.9.34. Let $\mathcal{C}$ be a set of clauses. A *hyperresolution tree over C* is an $\mathcal{FRES}^{\text{hyper}}_{\mathcal{C}}$-proof tree. ▯

A hyperresolution tree over $\mathcal{C}$ is a lot such that its leaves are labeled with clauses from $\mathcal{C}$ and each interior node is labeled with a clause that is a hyperresolvent of the clauses which are labels of its immediate descendents. By Theorem 1.8.23, we conclude that a clause C has a hyperresolution proof over a set of clauses $\mathcal{C}$ if and only if there is a hyperresolution tree over $\mathcal{C}$ such that C is the label of its root.

Example 3.9.35. Figure 3.33 gives the hyperresolution tree corresponding to the hyperresolution proof of Example 3.9.30. ▯

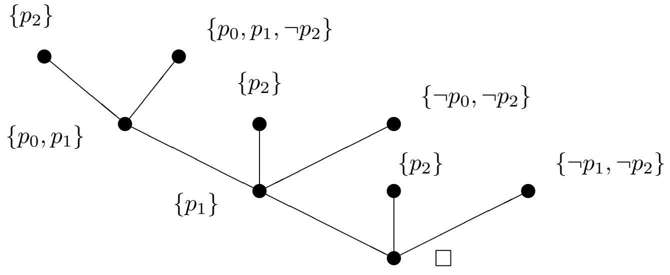


Fig. 3.33. Hyperresolution tree.

3.10 Cutting Planes

We are going to translate clauses into linear inequalities. To this end, we need to define formally the notion of linear inequality with integer coefficients, referred to here, for brevity, as inequalities. Let AVAR $= \{z_0, \ldots, z_n, \ldots\}$ be an infinite set of symbols. We call the members of AVAR *arithmetic variables*.

Definition 3.10.1. A *formal sum* is an infinite sequence a of integers such that $\mathrm{Supp}(a) = \{i \in \mathbf{N} \mid a_i \neq 0\}$ is finite. The set $\mathrm{Supp}(a)$ is called the *support* of a. $\square$

A formal sum a is denoted by $\sum a_i z_i$. An alternate notation for a is $a_{i_0} z_{i_0} + \cdots + a_{i_m} z_{i_m}$, where $i_0 < i_1 < \cdots < i_m$ and $\mathrm{Supp}(a) = \{i_0, \ldots, i_m\} \neq \emptyset$. The formal sum a with $\mathrm{Supp}(a) = \emptyset$ will be denoted $\mathsf{0}$. We will refer to a_i as the *coefficient of* z_i in a.

Example 3.10.2. The formal sum $(0, -1, 2, 0, 4, -6, 0, 0, \ldots)$ is denoted by $(-1)z_1 + 2z_2 + 4z_4 + (-6)z_5$ or, alternatively, by $-z_1 + 2z_2 + 4z_4 - 6z_5$. $\square$

Definition 3.10.3. A *linear inequality with integer coefficients* or, simply, an *inequality* is a pair (a, A), where a is a formal sum and $A \in \mathbf{Z}$. $\square$

We denote the inequality (a, A) by $a \geq A$. The set of all inequalities will be denoted INEQ.

Example 3.10.4. If $a = (0, -1, 2, 0, 4, -6, 0, 0, \ldots)$, then the inequality $(a, 3)$ is written as $-z_1 + 2z_2 + 4z_4 - 6z_5 \geq 3$. $\square$

Definition 3.10.5. A mapping $\xi : \text{AVAR} \longrightarrow \mathbf{Z}$ is called an *arithmetic assignment*. An arithmetic assignment ξ satisfies an inequality (a, A) if $\sum\{a_i\xi(z_i) \mid i \in \text{Supp}(a)\} \geq A$. □

Note that no arithmetic assignment satisfies the inequality $\mathsf{0} \geq 1$.

Example 3.10.6. The assignment $\xi(z_i) = i$ does not satisfy the inequality of Example 3.10.4. On the other hand, the assignment $\xi'(z_i) = i$ if $i \leq 2$ and $\xi'(z_i) = 0$, otherwise, satisfies that inequality. □

Definition 3.10.7. Let $\mathcal{I}$ be a set of inequalities and let I be an inequality. $\mathcal{I}$ *arithmetically implies* I if every arithmetic assignment that satisfies every inequality in $\mathcal{I}$ also satisfies I. We write $\mathcal{I} \models\!\equiv I$ if $\mathcal{I}$ arithmetically implies I. □

Example 3.10.8. It is clear that $\{z_1 \geq 5, z_2 \geq 7\} \models\!\equiv z_1 + z_2 \geq 12$. A less trivial example is $\{4z_2 + 6z_3 \geq 7\} \models\!\equiv 2z_2 + 3z_3 \geq 4$. □

Definition 3.10.9. Let v be a truth assignment. The *arithmetic assignment representing* v is ξ_v where $\xi_v(z_i) = 1$ if $v(p_i) = \mathbf{T}$ and $\xi_v(z_i) = 0$ otherwise. □

For a clause C, we denote by $n(C)$ the number of negative literals in C. If C is a nontautologous clause and p is a variable, let $S_C(p)$ be defined by

$$S_C(p) = \begin{cases} 1 \text{ if } p \in C \\ -1 \text{ if } \neg p \in C \\ 0 \text{ otherwise.} \end{cases}$$

Note that $S_C(p)$ is well defined because C is nontautologous.

Definition 3.10.10. Let C be a nontautologous clause. The *inequality I_C determined by C* is

$$\sum S_C(p_i)z_i \geq 1 - n(C).$$

□

Example 3.10.11. $I_\square$, the inequality determined by the empty clause, is $\mathsf{0} \geq 1$. □

Example 3.10.12. Let $\mathcal{C} = \{C_0, C_1, C_2, C_3\}$ be the set of clauses considered in Example 3.8.36, where $C_0 = \{\neg p_0, \neg p_1, p_2\}$, $C_1 = \{p_0, p_2\}$, $C_2 = \{p_1, p_2\}$, and $C_3 = \{\neg p_2\}$. The inequalities that are determined by these four clause are as follows:

$$\begin{array}{lrrrrr} I_{C_0}: & -z_0 & -z_1 & +z_2 & \geq & -1, \\ I_{C_1}: & z_0 & & +z_2 & \geq & 1, \\ I_{C_2}: & & z_1 & +z_2 & \geq & 1, \\ I_{C_3}: & & & -z_2 & \geq & 0. \end{array}$$

□

Theorem 3.10.13. *Let C be a nontautologous clause. A truth assignment v satisfies C if and only if ξ_v satisfies I_C.*

Proof. Observe that

$$\begin{aligned} &\sum\{S_C(p_i)\xi_v(z_i) \mid S_C(p_i) \neq 0\} \\ &\quad = \sum\{S_C(p) \mid p \in SV(C) \text{ and } v(p) = \mathbf{T}\} \\ &\quad = |\{p \in C \mid v(p) = \mathbf{T}\}| - |\{\neg p \in C \mid v(p) = \mathbf{T}\}| \\ &\quad = |\{p \in C \mid v(p) = \mathbf{T}\}| + |\{\neg p \in C \mid v(p) = \mathbf{F}\}| - n(C). \end{aligned}$$

It follows that ξ_v satisfies I_C if and only if

$$|\{p \in C \mid v(p) = \mathbf{T}\}| + |\{\neg p \in C \mid v(p) = \mathbf{F}\}| \geq 1$$

which holds if and only if v satisfies C. □

Definition 3.10.14. Let $\mathcal{C}$ be a set of clauses. The *set of inequalities $\mathcal{I}_\mathcal{C}$ determined by $\mathcal{C}$* is

$$\{I_C \mid C \in \mathrm{NT}(\mathcal{C})\} \cup \{z_i \geq 0 \mid p_i \in SV(\mathcal{C})\} \cup \{-z_i \geq -1 \mid p_i \in SV(\mathcal{C})\}.$$

(Recall that $\mathrm{NT}(\mathcal{C})$ is the set of all nontautologous clauses in $\mathcal{C}$.) □

Example 3.10.15. The set of inequalities for the set of clauses $\mathcal{C}$ of Example 3.10.12 consists of the inequalities I_{C_i} for $0 \leq i \leq 3$, together with the inequalities:

$$\begin{array}{cccc} z_0 \geq 0 & z_1 \geq 0 & z_2 \geq 0 & z_3 \geq 0, \\ -z_0 \geq -1 & -z_1 \geq -1 & -z_2 \geq -1 & -z_3 \geq -1. \end{array}$$

□

Corollary 3.10.16. *A truth assignment v satisfies a set of clauses $\mathcal{C}$ if and only if ξ_v satisfies $\mathcal{I}_{\mathcal{C}}$.*

Proof. The corollary follows immediately from Theorem 3.10.13. □

To make inferences about inequalities, we introduce a formal system $\mathcal{CP}$.

Definition 3.10.17. The formal system $\mathcal{CP}$ is $(\text{INEQ}, \emptyset, I)$, where I consists of the following rules:

$$\frac{\sum a_i z_i \geq A, \sum b_i z_i \geq B}{\sum (a_i + b_i) z_i \geq A + B} \qquad \text{Addition Rule}$$

$$\frac{\sum a_i z_i \geq A}{\sum (c a_i) z_i \geq cA} \qquad \text{Multiplication Rule}$$

for $c \in \mathbf{N}$

$$\frac{\sum (c a_i) z_i \geq A}{\sum a_i z_i \geq \lceil A/c \rceil} \qquad \text{Division Rule}$$

for $c \in \mathbf{P}$ and $\sum a_i z_i \neq \mathsf{0}$. □

Note that any application of the Multiplication Rule with $c > 0$ can be replaced by repeated applications of the Addition Rule. Proofs, however, can be shorter if this rule is included in I.

The objects manipulated by formal systems used in logic are, in general, strings of symbols. Although inequalities, as we have defined them, are not strings of symbols and therefore do not have a logical flavor, we can encode them as strings of symbols using the following technique. We can regard the notations $a_{i_0} z_{i_0} + \cdots + a_{i_m} z_{i_m} \geq A$ and $\mathsf{0} \geq A$ as strings of symbols over the infinite set $\mathbf{Z} \cup \text{AVAR} \cup \{+, -, \geq, \mathsf{0}\}$ and then regard the objects of $\mathcal{CP}$ as being these strings rather than the inequalities themselves. A further encoding step would regard inequalities as being represented by strings over the alphabet $\{0, 1, a, z, +, \geq, \mathsf{0}\}$. Namely, we can replace a_i by $as\mathtt{k}_{|a_i|}$ and z_i by $z\mathtt{k}_i$, where s is the sign of a_i and $\mathtt{k}_i$ is the notation introduced in Section 2.2.

Theorem 3.10.18. *Let $\mathcal{I}$ be a set of inequalities and let I be an inequality. If $\mathcal{I} \vdash_{\mathcal{CP}} I$, then $\mathcal{I} \models I$.*

Proof. We proceed by induction on the theorems of $\mathcal{CP}_{\mathcal{I}}$. If $I \in \mathcal{I}$, then we have immediately $\mathcal{I} \models I$.

Suppose that I is the inequality $\sum a_i z_i \geq \lceil A/c \rceil$, and that I was obtained from the inequality I' which is $\sum (ca_i) z_i \geq A$, where $\mathcal{I} \models\!\equiv I'$ and $c \in \mathbf{P}$. Let ξ be an arithmetic assignment that satisfies $\mathcal{I}$. Then, ξ satisfies I', so $\sum ca_i \xi(z_i) \geq A$. This implies $\sum a_i \xi(z_i) \geq A/c$ which gives $\sum a_i \xi(z_i) \geq \lceil A/c \rceil$ because $\sum a_i \xi(z_i)$ is an integer. Therefore, ξ satisfies I.

We leave to the reader the verification of the cases when I is obtained using the Addition or Multiplication Rules. □

Example 3.10.19. Let $\mathcal{I} = \{3z_0 + 3z_1 \geq 5, 4z_0 - z_1 \geq 10\}$. The following proof shows that $\mathcal{I} \models\!\equiv z_0 \geq 3$:

Line	Inequality	Justification
1	$3z_0 + 3z_1 \geq 5$	In $\mathcal{I}$
2	$z_0 + z_1 \geq 2$	Division Rule applied to Line 1
3	$4z_0 - z_1 \geq 10$	In $\mathcal{I}$
4	$5z_0 \geq 12$	Addition Rule applied to Lines 2 and 3
5	$z_0 \geq 3$	Division Rule applied to Line 4

The proof tree of the inequality $z_0 \geq 3$ in the formal system $\mathcal{CP}_{\mathcal{I}}$ is given in Figure 3.34. ▯

Theorem 3.10.20 (Soundness of Cutting Planes). *Let $\mathcal{C}$ be a set of clauses. If $\mathcal{I}_{\mathcal{C}} \vdash_{\mathcal{CP}} 0 \geq 1$, then $\mathcal{C}$ is unsatisfiable.*

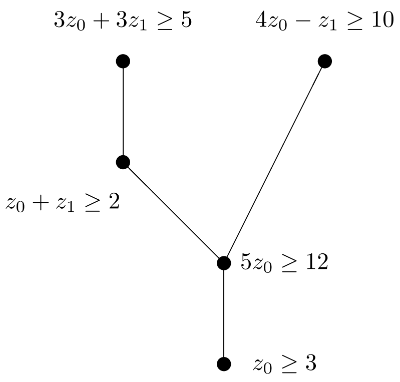


Fig. 3.34. Proof tree in the formal system $\mathcal{CP}_{\mathcal{I}}$.

Proof. If $\mathcal{I}_{\mathcal{C}} \vdash_{\mathcal{CP}} \mathsf{0} \geq 1$, then, by Theorem 3.10.18, we have $\mathcal{I}_{\mathcal{C}} \models \mathsf{0} \geq 1$. Since there is no arithmetic assignment that satisfies $\mathsf{0} \geq 1$, there is no arithmetic assignment that satisfies $\mathcal{I}_{\mathcal{C}}$. By Theorem 3.10.13, $\mathcal{C}$ is unsatisfiable. □

Example 3.10.21. The set of clauses in Example 3.10.12 was shown to be unsatisfiable in Example 3.8.36. Here we provide an alternate proof of this fact, using the soundness of $\mathcal{CP}$. In other words, we show that we can derive $\mathsf{0} \geq 1$ from the set of inequalities $\mathcal{I}_{\mathcal{C}}$ (given in Example 3.10.15) in the formal system $\mathcal{CP}$. We have the following proof:

Line	Inequality	Justification
1	$-z_0 - z_1 + z_2 \geq -1$	In $\mathcal{I}_{\mathcal{C}}$
2	$z_0 + z_2 \geq 1$	In $\mathcal{I}_{\mathcal{C}}$
3	$z_1 + z_2 \geq 1$	In $\mathcal{I}_{\mathcal{C}}$
4	$-z_2 \geq 0$	In $\mathcal{I}_{\mathcal{C}}$
5	$-z_1 + 2z_2 \geq 0$	Addition Rule applied to Lines 1 and 2
6	$3z_2 \geq 1$	Addition Rule applied to Lines 3 and 5
7	$2z_2 \geq 1$	Addition Rule applied to Lines 4 and 6
8	$z_2 \geq 1$	Division Rule applied to Line 7
9	$\mathsf{0} \geq 1$	Addition Rule applied to Lines 4 and 8

▯

Lemma 3.10.22. *Let C_0 and C_1 be nontautologous clauses. If R is a nontautologous resolvent of C_0 and C_1, then $\mathcal{I}_{\{C_0\}} \cup \mathcal{I}_{\{C_1\}} \vdash_{\mathcal{CP}} I_R$.*

Proof. The hypotheses of the lemma imply that there is a unique variable q such that q occurs in one of C_0, C_1 and $\neg q$ occurs in the other clause.

Adding the inequalities

$$I_{C_0} : \sum S_{C_0}(p_i) z_i \geq 1 - n(C_0),$$
$$I_{C_1} : \sum S_{C_1}(p_i) z_i \geq 1 - n(C_1),$$

we obtain the inequality

$$\sum a_i z_i \geq 2 - n(C_0) - n(C_1), \tag{3.10}$$

where a_i is given by

$$a_i = \begin{cases} 0 \text{ if } p_i = q \\ 2 \text{ if } p_i \in C_0 \cap C_1 \\ -2 \text{ if } \neg p_i \in C_0 \cap C_1 \\ 1 \text{ if } p_i \in C_0 \oplus C_1 \text{ and } p_i \neq q \\ -1 \text{ if } \neg p_i \in C_0 \oplus C_1 \text{ and } p_i \neq q \\ 0 \text{ if } p_i \notin SV(\{C_0, C_1\}). \end{cases}$$

We use here "$\oplus$" for the symmetric difference operation on sets. The reader can easily verify that the cases mentioned in the previous equation are exhaustive and mutually exclusive because of the hypotheses of the lemma.

Note that $C_0 \oplus C_1$ contains $n(C_0) + n(C_1) - 2n(C_0 \cap C_1) - 1$ negative literals distinct from $\neg q$. Starting with Inequality (3.10) and applying the Addition Rule repeatedly with the inequalities $\{z_i \geq 0 \mid p_i \in C_0 \oplus C_1 \text{ and } p_i \neq q\}$ and $\{-z_i \geq -1 \mid \neg p_i \in C_0 \oplus C_1 \text{ and } p_i \neq q\}$, we obtain

$$\sum b_i z_i \geq 3 - 2(n(C_0) + n(C_1) - n(C_0 \cap C_1)), \tag{3.11}$$

where

$$b_i = \begin{cases} 2 \text{ if } p_i \in C_0 \cup C_1 \text{ and } p_i \neq q \\ -2 \text{ if } \neg p_i \in C_0 \cup C_1 \text{ and } p_i \neq q \\ 0 \text{ otherwise.} \end{cases}$$

Note that for all i, $b_i = 2S_R(p_i)$.

Assume initially that $\sum b_i z_i \neq 0$. The Division Rule can be applied to Inequality (3.11) to divide by 2, and this gives

$$\sum S_R(p_i) z_i \geq 1 - (n(C_0) + n(C_1) - n(C_0 \cap C_1) - 1)$$

which concludes the argument for this case because $n(R) = n(C_0) + n(C_1) - n(C_0 \cap C_1) - 1$.

If $\sum b_i z_i = 0$, then $\{\{C_0\}, \{C_1\}\} = \{\{q\}, \{\neg q\}\}$, $R = \square$, and I_R is $0 \geq 1$. Since $n(C_0) + n(C_1) - n(C_0 \cap C_1) = 1$, I_R is Inequality (3.11) and no application of the Division Rule is required. $\square$

Theorem 3.10.23 (Completeness of Cutting Planes). *Let $\mathcal{C}$ be a finite set of clauses. If $\mathcal{C}$ is unsatisfiable, then $\mathcal{I}_\mathcal{C} \vdash_{\mathcal{CP}} 0 \geq 1$.*

Proof. We prove by course-of-values induction on n that if $(C_0, C_1, \ldots, C_n)$ is a resolution proof over $\mathcal{C}$ none of whose entries are tautologous, then $\mathcal{I}_\mathcal{C} \vdash_{\mathcal{CP}} I_{C_n}$. If $C_n \in \mathcal{C}$, then $I_{C_n} \in \mathcal{I}_\mathcal{C}$, so the conclusion is immediate. Suppose now that C_n is a resolvent of C_i and C_j with $0 \leq i, j < n$. By inductive hypothesis, we have $\mathcal{I}_\mathcal{C} \vdash_{\mathcal{CP}} I_{C_i}$ and $\mathcal{I}_\mathcal{C} \vdash_{\mathcal{CP}} I_{C_j}$. Using Exercise 81, it follows that for each inequality $I \in \mathcal{I}_{\{C_i\}} \cup \mathcal{I}_{\{C_j\}}$, $\mathcal{I}_\mathcal{C} \vdash_{\mathcal{CP}} I$. By Lemma 3.10.22, we have $\mathcal{I}_{\{C_i\}} \cup \mathcal{I}_{\{C_j\}} \vdash_{\mathcal{CP}} I_{C_n}$, so, by Corollary 1.8.8, $\mathcal{I}_\mathcal{C} \vdash_{\mathcal{CP}} I_{C_n}$.

If $\mathcal{C}$ is unsatisfiable, by Theorem 3.8.40, there is a resolution proof over $\mathcal{C}$ of $\Box$ each of whose entries is nontautologous. Therefore, by the claim proven above, $\mathcal{I}_\mathcal{C} \vdash_{\mathcal{CP}} I_\Box$, that is, $\mathcal{I}_\mathcal{C} \vdash_{\mathcal{CP}} 0 \geq 1$. $\Box$

We noted earlier that any application of the Multiplication Rule with $c > 0$ could be replaced by repeated applications of the Addition Rule. In fact, in the proofs of Lemma 3.10.22 and Theorem 3.10.23, we made no use of the Multiplication Rule (even with $c = 0$). Therefore, Theorem 3.10.23 would still be true if we had omitted the Multiplication Rule from the definition of $\mathcal{CP}$.

3.11 Exercises and Supplements

A Hilbert-Style Formal System

(1) Verify that every axiom of $\mathcal{HF}$ is a tautology.

(2) Show, without using the Completeness Theorem for $\mathcal{HF}$, that for all formulas φ, ψ, θ, we have

$$\vdash_{\mathcal{HF}} ((\varphi \to \psi) \to ((\psi \to \theta) \to (\varphi \to \theta))).$$

Conclude the *hypothetical syllogism rule*: if $\Gamma \vdash_{\mathcal{HF}} (\varphi \to \psi)$ and $\Gamma \vdash_{\mathcal{HF}} (\psi \to \theta)$, then $\Gamma \vdash_{\mathcal{HF}} (\varphi \to \theta)$. Further, argue that the $\mathcal{HF}_\Gamma$-proof of $(\varphi \to \theta)$ can be effectively obtained from the $\mathcal{HF}_\Gamma$-proofs of $(\varphi \to \psi)$ and $(\psi \to \theta)$.

Solution. Observe that by using modus ponens twice, we have

$$\{\varphi, (\varphi \to \psi), (\psi \to \theta)\} \vdash_{\mathcal{HF}} \theta,$$

so, by the Deduction Theorem applied three times, we reach the desired conclusion.

(3) Show, without using the Completeness Theorem for $\mathcal{HF}$, that for all formulas α, β, we have the following:

(a) $\vdash_{\mathcal{HF}} (((\neg\alpha) \to (\neg\beta)) \to (\beta \to \alpha))$,
(b) $\vdash_{\mathcal{HF}} ((\neg(\neg\alpha)) \to \alpha)$,
(c) $\vdash_{\mathcal{HF}} (\alpha \to (\neg(\neg\alpha)))$,
(d) $\vdash_{\mathcal{HF}} ((\alpha \to \beta) \to ((\neg\beta) \to (\neg\alpha)))$.

Solution. For the first part, let Γ be $\{((\neg\alpha) \to (\neg\beta)), \beta, (\neg\alpha)\}$. By modus ponens, we have $\Gamma \vdash_{\mathcal{HF}} (\neg\beta)$ and by Axiom Group 4, we have $\Gamma \vdash_{\mathcal{HF}} (\beta \to ((\neg\beta) \to \alpha))$. A double application of modus ponens yields successively $\Gamma \vdash_{\mathcal{HF}} ((\neg\beta) \to \alpha)$ and $\Gamma \vdash_{\mathcal{HF}} \alpha$. By the Deduction Theorem, $\{((\neg\alpha) \to (\neg\beta)), \beta\} \vdash_{\mathcal{HF}} ((\neg\alpha) \to \alpha)$. Using Axiom Group 5, we have $\{((\neg\alpha) \to (\neg\beta)), \beta\} \vdash_{\mathcal{HF}} \alpha$, and then, applying the Deduction Theorem twice gives the desired result.

To prove the second part, observe that $\{(\neg(\neg\alpha)), (\neg\alpha)\}$ is obviously inconsistent, so by reductio ad absurdum, we have $\{(\neg(\neg\alpha))\} \vdash_{\mathcal{HF}} \alpha$; an application of the Deduction Theorem yields the result.

We leave the third part to the reader, and we discuss next Part (d). Since $\vdash_{\mathcal{HF}} ((\neg(\neg\alpha)) \to \alpha)$, we have $\{(\alpha \to \beta), (\neg\beta), (\neg(\neg\alpha))\} \vdash_{\mathcal{HF}} ((\neg(\neg\alpha)) \to \alpha)$. We also have $\{(\alpha \to \beta), (\neg\beta), (\neg(\neg\alpha))\} \vdash_{\mathcal{HF}} \alpha$, by modus ponens. Another application of modus ponens gives $\{(\alpha \to \beta), (\neg\beta), (\neg(\neg\alpha))\} \vdash_{\mathcal{HF}} \beta$. Note that we also have

$$\{(\alpha \to \beta), (\neg\beta), (\neg(\neg\alpha))\} \vdash_{\mathcal{HF}} (\neg\beta).$$

Since the formula $((\beta \to ((\neg\beta) \to (\neg\alpha)))$ is an instance of the Axiom Group 4, a double application of modus ponens implies $\{(\alpha \to \beta), (\neg\beta), (\neg(\neg\alpha))\} \vdash_{\mathcal{HF}} (\neg\alpha)$. By the Deduction Theorem, we have $\{(\alpha \to \beta), (\neg\beta)\} \vdash_{\mathcal{HF}} ((\neg(\neg\alpha)) \to (\neg\alpha))$. Using the formula

$$(((\neg(\neg\alpha)) \to (\neg\alpha)) \to (\neg\alpha)),$$

an instance of Axiom Group 5, we obtain $\{(\alpha \to \beta), (\neg\beta)\} \vdash_{\mathcal{HF}} (\neg\alpha)$. A final double application of the Deduction Theorem gives the desired formula.

(4) Use the Deduction Theorem to show that if $\Gamma \vdash_{\mathcal{HF}} (\varphi \to (\alpha \to \beta))$, then $\Gamma \vdash_{\mathcal{HF}} (\varphi \to ((\neg\beta) \to (\neg\alpha)))$; show also that if $\Gamma \vdash_{\mathcal{HF}} (\alpha \to (\beta \to \gamma))$, then $\Gamma \vdash_{\mathcal{HF}} (\beta \to (\alpha \to \gamma))$.

(5) Let $\mathcal{HF}'$ be the formal system $(\text{PLFORM}, A', \{\mathsf{R}_{mp}\})$, where A' consists of the following groups of axioms:

- $(\alpha \to (\beta \to \alpha))$,
- $((\alpha \to (\beta \to \gamma)) \to ((\alpha \to \beta) \to (\alpha \to \gamma)))$,
- $(\alpha \to \alpha)$,
- $(\alpha \to ((\neg\alpha) \to \beta))$,
- $(((\neg\alpha) \to \alpha) \to \alpha)$,
- $((\alpha \to (\neg\alpha)) \to (\neg\alpha))$,
- $((\alpha \vee \beta) \to ((\neg\alpha) \to \beta))$,
- $((\alpha \wedge \beta) \to \alpha)$,
- $((\alpha \wedge \beta) \to \beta)$,
- $((\alpha \leftrightarrow \beta) \to (\alpha \to \beta))$,
- $((\alpha \leftrightarrow \beta) \to ((\neg\alpha) \to (\neg\beta)))$,
- $((\neg(\alpha \vee \beta)) \to (\neg\alpha))$,
- $((\neg(\alpha \vee \beta)) \to (\neg\beta))$,
- $((\neg(\alpha \wedge \beta)) \to (\alpha \to (\neg\beta)))$,
- $((\neg(\alpha \to \beta)) \to \alpha)$,
- $((\neg(\alpha \to \beta)) \to (\neg\beta))$,
- $((\neg(\alpha \leftrightarrow \beta)) \to (\alpha \to (\neg\beta)))$,
- $((\neg(\alpha \leftrightarrow \beta)) \to ((\neg\alpha) \to \beta))$,

for all formulas α, β, γ.

(a) Formulate and prove a version of the Soundness Theorem for the formal system $\mathcal{HF}'$.

(b) Show that the Deduction Theorem holds with $\mathcal{HF}'$ in place of $\mathcal{HF}$.

(c) Show that if the definitions of consistency and inconsistency are changed by replacing $\mathcal{HF}$ with $\mathcal{HF}'$, then the results from Theorem 3.2.7 to Corollary 3.2.13 hold with the obvious changes.

(d) Using one of the characterizations of truth sets in Theorem 2.7.7, show that Theorem 3.2.18 remains valid for the notion of maximal consistency obtained using $\mathcal{HF}'$.

(e) Prove that consistency of sets of formulas with respect to $\mathcal{HF}'$ is a consistency property.

(f) Show that either of the two previous parts can be used to show that a consistent set of formulas (with respect to $\mathcal{HF}'$) is satisfiable.

(g) Prove the Completeness Theorem for $\mathcal{HF}'_{\Gamma}$.

(6) Prove that the following statements

   (a) the Soundness Theorem,
   (b) if a set of formulas Γ is satisfiable, then Γ is consistent

   can be directly derived from each other.
   Solution. The proof of Theorem 3.2.8 shows that the second statement follows from the Soundness Theorem. Conversely, suppose that $\Gamma \not\models \varphi$. Thus, $\Gamma \cup \{(\neg\varphi)\}$ is satisfiable by the first part of Theorem 2.3.17. Assuming that satisfiability entails consistency, $\Gamma \cup \{(\neg\varphi)\}$ is consistent, which implies that $\Gamma \nvdash_{\mathcal{HF}} \varphi$. This establishes the contrapositive of the Soundness Theorem.

(7) Prove that the following statements

   (a) the Completeness Theorem,
   (b) if a set of formulas Γ is consistent, then Γ is satisfiable

   can be directly derived from each other.
   Solution. The proof of the Completeness Theorem shows that this theorem follows from the second statement. Conversely, suppose that Γ is not satisfiable. Thus, for any formula φ, we have both $\Gamma \models \varphi$ and $\Gamma \models (\neg\varphi)$. Assuming the Completeness Theorem, we have both $\Gamma \vdash_{\mathcal{HF}} \varphi$ and $\Gamma \vdash_{\mathcal{HF}} (\neg\varphi)$, which implies that Γ is not consistent. Thus, we derived the contrapositive of the second statement from the Completeness Theorem.

Tableaux

(8) Let s be a propositional substitution and let $\mathtt{T}$ be a Δ-tableau. Prove that $s \circ \mathtt{T}$ is an $s(\Delta)$-tableau. Moreover, prove that if $\mathtt{T}$ is (strongly) closed, then $s \circ \mathtt{T}$ is (strongly) closed.
   Hint. Use Exercise 81 of Chapter 2.

(9) Prove that if $\mathtt{T}$ is a Δ-tableau with retention and q, r are two nodes of $\mathtt{T}$ such that q is a prefix of r, then $\mathtt{T}(q) \subseteq \mathtt{T}(r)$.

Let $\mathtt{T}$ be a Δ-tableau and let $b\varphi$ be a signed formula. A node qi of $\mathtt{T}$ is said to be *$b\varphi$-introducing* if $b\varphi \in \mathtt{T}(qi) - \mathtt{T}(q)$.

(10) Let $\mathtt{T}$ be a Δ-tableau and let $b\varphi$ be a signed formula that is not expanded at any node of $\mathtt{T}$. Prove that the tableau $\mathtt{T}'$, where

$\mathrm{Dom}(\mathtt{T}') = \mathrm{Dom}(\mathtt{T})$ and

$$\mathtt{T}'(q) = \begin{cases} \mathtt{T}(q) - \{b\varphi\} & \text{if } q \text{ has no } b\varphi\text{-introducing ancestor} \\ \mathtt{T}(q) & \text{otherwise} \end{cases}$$

for $q \in \mathrm{Dom}(\mathtt{T}')$, is a $(\Delta - \{b\varphi\})$-tableau. (We will denote $\mathtt{T}'$ by $\mathtt{T}\dot{-}b\varphi$.) Also, show that if $\mathtt{T}$ is conservative, then $\mathtt{T}\dot{-}b\varphi$ is conservative.

Solution. Clearly, $\mathtt{T}'(\lambda) = \Delta - \{b\varphi\}$. We claim that $\mathtt{T}'$ is a $(\Delta - \{b\varphi\})$-tableau, that is, $\mathtt{T}'$ is locally consistent at every interior node q. Suppose that q is an interior node in $\mathrm{Dom}(\mathtt{T}) = \mathrm{Dom}(\mathtt{T}')$. There are two possible cases: either we used thinning at q or we expanded a formula $b_q\varphi_q$ at q. In the first case, q has one direct descendant $q0$ in $\mathtt{T}$ and $\mathtt{T}'$ and $\mathtt{T}(q0) \subseteq \mathtt{T}(q)$. If q has no $b\varphi$-introducing ancestor, then neither does $q0$, so $\mathtt{T}'(q0) = \mathtt{T}(q0) - \{b\varphi\} \subseteq \mathtt{T}(q0) - \{b\varphi\} = \mathtt{T}'(q)$. If q has a $b\varphi$-introducing ancestor, then $\mathtt{T}'(q0) = \mathtt{T}(q0) \subseteq \mathtt{T}(q) = \mathtt{T}'(q)$. Thus, in either of the above two subcases, we have thinning at q in $\mathtt{T}'$. If there is expansion at q in $\mathtt{T}$, we have $\mathtt{T}(q) = \Delta_q \cup \{b_q\varphi_q\}$, $\mathtt{d}(b_q\varphi_q) = (H_0, \ldots, H_{l-1})$, and $\mathtt{T}(qj) = \Delta_q \cup H_j$, for $0 \leq j \leq l-1$. We show that after the possible removal of $b\varphi$, the resulting tableau $\mathtt{T}'$ is locally consistent at q.

There are two subcases to consider. If $b\varphi$ is not removed from $\mathtt{T}(q)$ (because q has a $b\varphi$-introducing node as an ancestor), then $b\varphi$ is not removed from the children of q, so $\mathtt{T}'$ is locally consistent at q. Now, suppose that $b\varphi$ is removed from $\mathtt{T}(q)$. Since $b\varphi$ is not expanded in any node of $\mathtt{T}$, we have $b_q\varphi_q \neq b\varphi$. If i is such that $b\varphi \notin H_i$, then qi is not $b\varphi$-introducing, so we subtract $b\varphi$ from $\mathtt{T}(qi)$, which implies that

$$\begin{aligned} \mathtt{T}'(q) &= \mathtt{T}(q) - \{b\varphi\} = (\Delta_q \cup \{b_q\varphi_q\}) - \{b\varphi\} \\ &= (\Delta_q - \{b\varphi\}) \cup \{b_q\varphi_q\}, \\ \mathtt{T}'(qi) &= (\Delta_q \cup H_i) - \{b\varphi\} = (\Delta_q - \{b\varphi\}) \cup H_i, \end{aligned}$$

since $b\varphi \neq b_q\varphi_q$. If $b\varphi \in H_i$, then $\mathtt{T}'(qi) = \Delta_q \cup H_i = (\Delta_q - \{b\varphi\}) \cup H_i$. This shows that $\mathtt{T}'$ is locally consistent at q. Furthermore, the previous argument shows that if $\mathtt{T}$ is conservative, then so is $\mathtt{T}'$. This completes the proof.

(11) Let $\mathtt{T}$ be a $(\Delta \cup \{bp\})$-tableau, where $p \notin SV(\Delta)$. Prove that the tableau $\mathtt{T}'$ defined by $\mathrm{Dom}(\mathtt{T}') = \mathrm{Dom}(\mathtt{T})$ and $\mathtt{T}'(q) = \mathtt{T}(q) - \{bp\}$ for $q \in \mathrm{Dom}(\mathtt{T}')$ is a Δ-tableau that is (strongly) closed if $\mathtt{T}$ is (strongly) closed.

Solution. Observe that in $\mathtt{T}$, no formula containing p other than bp occurs in any node. This can be shown easily by induction on the depth of the nodes of $\mathtt{T}$. Therefore, no node of $\mathtt{T}$ has a bp-introducing ancestor and so $\mathtt{T}' = \mathtt{T} \dot{-} \bar{b}p$, which shows that $\mathtt{T}'$ is a Δ-tableau by Supplement 10. Since $\bar{b}p$ does not occur in any node of $\mathtt{T}$, if $\mathtt{T}$ is (strongly) closed, then so is $\mathtt{T}'$.

(12) Let $\mathtt{T}$ be a Δ-tableau and define a tableau $\mathtt{T}'$ with the same domain as $\mathtt{T}$ by

$$\mathtt{T}'(q) = \bigcup \{\mathtt{T}(r) \mid r \in \mathrm{PREF}(q)\}.$$

Prove the following:

(a) $\mathtt{T}'$ is a Δ-tableau with retention and for each branch $\mathtt{B}$ of $\mathtt{T}$, $\mathtt{B}$ is closed in $\mathtt{T}$ if and only if it is closed in $\mathtt{T}'$.

(b) The branch $\mathtt{B}$ is complete in $\mathtt{T}$ if and only if it is complete in $\mathtt{T}'$.

(c) $\mathtt{T}$ is closed (completed) if and only if $\mathtt{T}'$ is.

(d) If $\mathtt{T}$ is strongly completed, then so is $\mathtt{T}'$.

(e) If $\mathtt{T}$ is conservative, then so is $\mathtt{T}'$.

(13) Let $\mathtt{T}$ be a finite closed Δ-tableau with retention. Prove that there is a finite closed Δ-tableau with retention $\mathtt{T}'$ such that no formula is expanded twice in any branch of $\mathtt{T}'$ and $|\mathtt{T}'| \leq |\mathtt{T}|$. Furthermore, if $\mathtt{T}$ is conservative, then so is $\mathtt{T}'$.

Solution. The argument is by induction on $n = |\mathtt{T}|$. The basis step, $n = 1$, is trivial. Suppose that the statement holds for tableaux with fewer than n nodes. If there is no branch on which a formula is expanded more than once, then $\mathtt{T}' = \mathtt{T}$. Otherwise, let $\mathtt{B}$ be a branch of $\mathtt{T}$ where a formula $b\varphi$ is expanded at both q and r with q a proper prefix of r. Suppose that $\mathtt{d}(b\varphi) = (K_0, \ldots, K_{p-1})$ and that qi is a prefix of r. Then, $K_i \subseteq \mathtt{T}(qi) \subseteq \mathtt{T}(r)$, by Exercise 9, so $\mathtt{T}(ri) = \mathtt{T}(r) \cup K_i = \mathtt{T}(r)$. Therefore, the tableau $\mathtt{T}'' = \mathtt{T}[r \to \mathtt{T}_{[ri]}]$ obtained by inserting the tableau $\mathtt{T}_{[ri]}$ at r is also a Δ-tableau with retention and fewer than n nodes. Moreover, for every branch $\mathtt{B}''$ of $\mathtt{T}''$, there is branch $\mathtt{B}$ of $\mathtt{T}$ such that $\mathtt{T}''(\mathtt{B}'') = \mathtt{T}(\mathtt{B})$ and this implies that since $\mathtt{T}$ is a closed Δ-tableau, $\mathtt{T}''$ is also a closed Δ-tableau. Also, if $\mathtt{T}$ is conservative, then so is $\mathtt{T}''$. By the inductive hypothesis, there is a closed Δ-tableau $\mathtt{T}'$ such that no formula is expanded more than once in any of its branches and $|\mathtt{T}'| \leq |\mathtt{T}''| < |\mathtt{T}|$. Further, if $\mathtt{T}''$ is conservative, then so is $\mathtt{T}'$.

(14) Let $\mathtt{T}$ be a finite Δ-tableau such that no formula is expanded twice in any of its branches. Prove that there is a Δ-tableau with removal $\mathtt{T}'$ such that $\mathrm{Dom}(\mathtt{T}') = \mathrm{Dom}(\mathtt{T})$ and for every branch $\mathtt{B}$ of $\mathtt{T}'$, we have $\mathtt{T}'(\mathtt{B}) = \mathtt{T}(\mathtt{B})$. Conclude that if $\mathtt{T}$ is a closed Δ-tableau, then so is $\mathtt{T}'$. Also, prove that if $\mathtt{T}$ is conservative, then so is $\mathtt{T}'$.

Solution. The argument is by induction on $n = |\mathtt{T}|$. The basis step, $n = 1$, is trivial. Suppose that the statement holds for tableaux with fewer than n nodes.

Suppose that $\Delta = \mathtt{T}(\lambda) = \Delta' \cup \{b\varphi\}$ and $\mathtt{d}(b\varphi) = (K_0, \ldots, K_{p-1})$, where $b\varphi$ is expanded at the root. For $0 \leq i \leq p-1$, the tableau $\mathtt{T}_{[i]}$ is a $(\Delta' \cup K_i)$-tableau in which the formula $b\varphi$ is never expanded, due to the hypothesis we made on $\mathtt{T}$. Therefore, by Supplement 10, the tableau $\mathtt{T}_{[i]} \dot{-} b\varphi$ is a $(\Delta' - \{b\varphi\}) \cup K_i$-tableau in which no formula is expanded twice in any branch and is conservative if $\mathtt{T}$ is conservative. By the inductive hypothesis, there is a $(\Delta' - \{b\varphi\}) \cup K_i$-tableau with removal $\mathtt{T}'_i$ such that $\mathrm{Dom}(\mathtt{T}'_i) = \mathrm{Dom}(\mathtt{T}_{[i]} \dot{-} b\varphi)$ and $\mathtt{T}'_i(\mathtt{B}) = (\mathtt{T}_{[i]} \dot{-} b\varphi)(\mathtt{B})$ for every branch $\mathtt{B}$ of $\mathtt{T}'_i$. The desired tableau $\mathtt{T}'$ is $(\mathtt{T}'_0, \ldots, \mathtt{T}'_{p-1}; \Delta)$.

The case when thinning is used at the root of $\mathtt{T}$ is left to the reader.

(15) Let $\mathtt{T}$ be a finite closed Δ-tableau. Prove that there is a finite closed Δ-tableau with removal $\widehat{\mathtt{T}}$ such that $|\widehat{\mathtt{T}}| \leq |\mathtt{T}|$. Moreover, in $\widehat{\mathtt{T}}$, no formula is expanded twice in any branch and if $\mathtt{T}$ is conservative, then so is $\widehat{\mathtt{T}}$.

Solution. By Exercise 12, there is a Δ-tableau with retention $\mathtt{T}'$ such that $\mathrm{Dom}(\mathtt{T}') = \mathrm{Dom}(\mathtt{T})$ and $\mathtt{T}'$ is closed since $\mathtt{T}$ is closed. By Supplement 13, there is Δ-tableau $\mathtt{T}''$ (with retention) such that no formula is expanded twice in a branch of $\mathtt{T}''$, $|\mathtt{T}''| \leq |\mathtt{T}'|$, and $\mathtt{T}''$ is closed because $\mathtt{T}'$ is closed. Then, by Supplement 14, there is a Δ-tableau $\widehat{\mathtt{T}}$ with removal such that

$\text{Dom}(\widehat{\mathtt{T}}) = \text{Dom}(\mathtt{T}'')$; $\widehat{\mathtt{T}}$ is closed since $\mathtt{T}''$ is closed, so $\widehat{\mathtt{T}}$ is the desired Δ-tableau.

As we saw previously, the conservative property is preserved by our definition of $\widehat{\mathtt{T}}$.

(16) Effectivize the statement of Supplement 15 by giving a construction that transforms a finite closed Δ-tableau $\mathtt{T}$ into a finite closed Δ-tableau $\mathtt{T}'$ with removal such that $|\mathtt{T}'| \leq |\mathtt{T}|$.

(17) Let $\mathtt{T}$ be a Δ-tableau and let Δ' be a set of signed formulas. Define the tableau $\mathtt{T} \sqcup \Delta'$ by $\text{Dom}(\mathtt{T} \sqcup \Delta') = \text{Dom}(\mathtt{T})$ and

$$(\mathtt{T} \sqcup \Delta')(q) = \mathtt{T}(q) \cup \Delta'$$

for $q \in \text{Dom}(\mathtt{T})$. Prove that $\mathtt{T} \sqcup \Delta'$ is a $(\Delta \cup \Delta')$-tableau. Further, show that if $\mathtt{T}$ is (strongly) closed, then $\mathtt{T} \sqcup \Delta'$ is (strongly) closed. Also, show that if $\mathtt{T}$ is conservative, then so it $\mathtt{T} \sqcup \Delta'$.

(18) Give a syntactic construction that starts with a finite tableau $\mathtt{T}$ and produces a conservative $\mathtt{T}(\lambda)$-tableau $\mathtt{T}'$ such that $|\mathtt{T}'| \leq |\mathtt{T}|$ and for every branch $\mathtt{B}$ of $\mathtt{T}$, there is a branch $\mathtt{B}'$ of $\mathtt{T}'$ such that $\mathtt{T}'(\mathtt{B}') = \mathtt{T}(\mathtt{B})$. Conclude that if $\mathtt{T}$ is a (strongly) closed tableau, then so is $\mathtt{T}'$.

Solution. The construction builds the tableau $\mathtt{T}'$ recursively as follows. If $\mathtt{T}$ is a one-node tableau, then $\mathtt{T}' = \mathtt{T}$. Suppose that regular expansion is used at the root of $\mathtt{T}$, where the root has n children. In this case, we apply the construction recursively to the tableaux $\mathtt{T}_{[0]}, \ldots, \mathtt{T}_{[n-1]}$ which yields conservative tableaux $\mathtt{T}'_0, \ldots, \mathtt{T}'_{n-1}$, where $|\mathtt{T}'_i| \leq |\mathtt{T}_i|$ for $0 \leq i \leq n-1$ and the sets of the formulas of the branches are preserved. Then, $\mathtt{T}' = (\mathtt{T}'_0, \ldots, \mathtt{T}'_{n-1}; \mathtt{T}(\lambda))$. If thinning was applied at the root of $\mathtt{T}$, then the root has one descendant tableau $\mathtt{T}_{[0]}$ such that $\mathtt{T}_{[0]}(\lambda) \subseteq \mathtt{T}(\lambda)$. A recursive application of the construction to $\mathtt{T}_{[0]}$ generates a conservative tableau $\mathtt{T}'_0$ with preservation of the sets of formulas of the branches. Then, $\mathtt{T}' = \mathtt{T}'_0 \sqcup \mathtt{T}(\lambda)$ (see Exercise 17.)

(19) Let α be a formula and let $\mathtt{T}$ be a Δ-tableau. Prove that if α does not occur positively (negatively) in Δ, then α does not occur positively (negatively) in any node of $\mathtt{T}$.

Hint. Use the result stated in Exercise 94 of Chapter 2.

(20) Let $\mathtt{T}$ be a (strongly) closed $(\Delta \cup \{\mathbf{T}\varphi\})$-tableau, where φ does not occur negatively in Δ. Prove that for each constituent K of

$\mathbf{T}\varphi$ there is a (strongly) closed $(\Delta \cup K)$-tableau with no more nodes than $\mathtt{T}$.

Also, prove that if $\mathtt{T}$ is a (strongly) closed $(\Delta \cup \{\mathbf{F}\varphi\})$-tableau, where φ does not occur positively in Δ, then for each constituent K of $\mathbf{F}\varphi$ there is a (strongly) closed $(\Delta \cup K)$-tableau with no more nodes than $\mathtt{T}$.

Solution. We prove only the first statement and leave the similar proof of the second statement to the reader. The proof is by induction on $n = |\mathtt{T}|$, the number of nodes of $\mathtt{T}$.

For the basis step, $n = 1$, the existence of a one-node closed $(\Delta \cup \{\mathbf{T}\varphi\})$-tableau implies that $\Delta \cup \{\mathbf{T}\varphi\}$ is closed. Since φ does not occur negatively in Δ, $\mathbf{F}\varphi \notin \Delta$, which implies that Δ itself is closed. Thus, for any constituent K of $\mathbf{T}\varphi$, $\Delta \cup K$ is closed and thus there is a one-node strongly closed $(\Delta \cup K)$-tableau.

Suppose that the statement holds for tableaux with fewer than n nodes. If regular expansion is used at the root of $\mathtt{T}$, then let $b\theta$ be the formula expanded there, where $\mathtt{d}(b\theta) = (K_0, \ldots, K_{n-1})$. Define $\mathtt{T}' = \mathtt{T} \sqcup \{b\theta\}$. Then, by Exercise 17, $\mathtt{T}'$ is a (strongly) closed $(\Delta \cup \{\mathbf{T}\varphi\})$-tableau with retention at the root and $|\mathtt{T}'| = |\mathtt{T}|$. By Theorem 3.3.10, for $0 \leq i \leq n-1$, $\mathtt{T}'_{[i]}$ is a (strongly) closed $(\Delta \cup \{\mathbf{T}\varphi\} \cup K_i)$-tableau with fewer than n nodes. We consider two cases: $b\theta = \mathbf{T}\varphi$ and $b\theta \neq \mathbf{T}\varphi$.

In the first case, if $K = K_{i_0}$, then $\mathtt{T}'_{[i_0]}$ is a (strongly) closed $(\Delta \cup K \cup \{\mathbf{T}\varphi\})$-tableau. Note that φ does not occur negatively in K because it does not occur in K at all. Thus, by the inductive hypothesis, there is a (strongly) closed $(\Delta \cup K \cup K = \Delta \cup K)$-tableau with no more nodes than $\mathtt{T}'_{[i_0]}$ and hence with no more than n nodes.

In the second case, observe that, by Exercise 94 of Chapter 2, φ does not occur negatively in any set $\Delta \cup K_i$, for $0 \leq i \leq n-1$. Therefore, applying the inductive hypothesis to the tableaux $\mathtt{T}'_{[i]}$, we obtain (strongly) closed $(\Delta \cup K_i \cup K)$-tableaux $\mathtt{T}''_i$ with $|\mathtt{T}''_i| \leq |\mathtt{T}'_{[i]}|$. Consequently, the lot $(\mathtt{T}''_0, \ldots, \mathtt{T}''_{n-1}; \Delta \cup K)$ is a (strongly) closed $(\Delta \cup K)$-tableau with no more than n nodes.

If thinning was used at the root of $\mathtt{T}$, define $\mathtt{T}' = \mathtt{T} \sqcup (\Delta \cup \{\mathbf{T}\varphi\})$. Then, $\mathtt{T}'$ is a (strongly) closed $(\Delta \cup \{b\varphi\})$-tableau, so $\mathtt{T}'_{[0]}$ is a (strongly) closed $(\Delta \cup \{b\varphi\})$-tableau with $n-1$ nodes. Note that the fact that $\mathtt{T}'_{[0]}$ is closed if $\mathtt{T}$ is closed follows from the fact that $\mathtt{T}'(0) = \mathtt{T}'(\lambda)$. By the inductive hypothesis, there is a (strongly) closed $(\Delta \cup K)$-tableau with no more than $n-1$ nodes.

(21) Let Δ be a set of signed formulas and let $\mathtt{T}$ be a conservative Δ-tableau. Prove that if q is a leaf of $\mathtt{T}$ and v is a truth assignment that satisfies $\mathtt{T}(q)$, then v satisfies $\mathtt{T}(\mathtt{B})$, where $\mathtt{B}$ is the branch that ends in q.

We remind the reader that according to Definition 1.8.16, the size $\mathtt{size}(\mathtt{T})$ of a finite tableau $\mathtt{T}$ is the sum of the sizes of the labels of its nodes, where the size of the label of a node is the sum of the sizes of the (signed) formulas that occur in the node.

(22) Let Δ be a finite unsatisfiable set of signed formulas. Prove that there is a strongly closed Δ-tableaux with no more than $2^{2L+1} - 1$ nodes and size no more than $(2^{2L+1} - 1)K$, where $L = \sum\{|\varphi| \mid b\varphi \in \Delta \text{ for some } b \in \mathbf{Bool}\}$ and K is

$$2\sum\{\mathtt{size}(\psi) \mid \psi \in \bigcup\{\mathrm{SUBF}(\varphi) \mid b\varphi \in \Delta \text{ for some } b \in \mathbf{Bool}\}\}.$$

Hint. Use the correctness proof of Algorithm 3.3.27 and Exercise 8 of Chapter 2.

(23) (a) Let $\mathcal{I}$ ($\mathcal{I}_s$) be the collection of all sets of signed formulas Δ such that there is a finite closed (strongly closed) Δ-tableau. Show that $\mathcal{I}$ ($\mathcal{I}_s$) is an inconsistency property.

(b) Use the first part to show that if Δ is an unsatisfiable set of signed formulas, then there is a finite closed (strongly closed) Δ-tableau. (This gives a different proof of the Completeness (Strong Completeness) Theorem for Δ-tableaux.)

Hint. The proof of the first part uses Theorem 3.3.11. For the second part, suppose that there is no finite closed Δ-tableau. Then, by Part (a), Δ belongs to a consistency property, so it is satisfiable.

(24) In Theorem 3.3.40, we gave an effective, recursive construction that produces a strongly closed Δ'-tableau starting from a strongly closed Δ-tableau, where Δ' is a finite subset of Δ. Give an explicit, nonrecursive construction which, starting from a finite, conservative, closed Δ-tableau $\mathtt{T}$, produces a finite unsatisfiable subset of Δ.

Solution. For each interior node q of $\mathtt{T}$, let $b_q\varphi_q$ be a signed formula with $\mathtt{d}(b_q\varphi_q) = (K_0^q, \ldots, K_{n_q-1}^q)$ and let Δ_q be a set of signed formulas such that $\mathtt{T}(q) = \Delta_q \cup \{b_q\varphi_q\}$ and q has n immediate descendants with $\mathtt{T}(qj) = \Delta_q \cup K_j^q$ for $0 \leq j \leq n_q - 1$. Since $\mathtt{T}$ is closed, for every branch $\mathtt{B}$ of $\mathtt{T}$, there is a formula $\varphi_{\mathtt{B}}$ such that both $\mathbf{T}\varphi_{\mathtt{B}}$ and $\mathbf{F}\varphi_{\mathtt{B}}$ occur in $\mathtt{B}$.
Define a finite subset Δ_0 of Δ by

$$\begin{aligned}\Delta_0 = \Delta \cap \Big(&\{b_q\varphi_q \mid q \in \mathrm{INTN}(\mathtt{T})\} \\ &\cup \bigcup \{K_j^q \mid q \in \mathrm{INTN}(\mathtt{T}) \text{ and } 0 \leq j \leq n_q - 1\} \\ &\cup \bigcup \{\{\mathbf{T}\varphi_{\mathtt{B}}, \mathbf{F}\varphi_{\mathtt{B}}\} \mid \mathtt{B} \text{ is a branch of } \mathtt{T}\}\Big).\end{aligned}$$

Consider the tableau $\mathtt{T}'$, where $\mathrm{Dom}(\mathtt{T}') = \mathrm{Dom}(\mathtt{T})$ and $\mathtt{T}'(q) = \mathtt{T}(q) - (\Delta - \Delta_0)$. We show that $\mathtt{T}'$ is a closed Δ_0-tableau. We have $\mathtt{T}'(\lambda) = \Delta_0$ because $\Delta_0 \subseteq \Delta$. For each interior node q of $\mathtt{T}'$, let $\Delta'_q = \Delta_q - (\Delta - \Delta_0)$. Because $b_q\varphi_q \notin \Delta - \Delta_0$ and $K_j^q \cap (\Delta - \Delta_0) = \emptyset$, for $0 \leq j \leq n_q - 1$, we have $\mathtt{T}'(q) = \Delta'_q \cup \{b_q\varphi_q\}$ and $\mathtt{T}'(qj) = \Delta'_q \cup K_j^q$, which shows that $\mathtt{T}'$ is a Δ_0-tableau. Also notice that $\mathtt{T}'$ is closed because $\{\mathbf{T}\varphi_{\mathtt{B}}, \mathbf{F}\varphi_{\mathtt{B}}\} \subseteq \mathtt{T}'(\mathtt{B})$ for every branch $\mathtt{B}$ of $\mathtt{T}'$. By Theorem 3.3.16, Δ_0 is unsatisfiable.

(25) Reprove Exercise 116 of Chapter 2 using tableaux. In other words, show that if Δ is a satisfiable set of signed formulas, then Δ is contained in a Hintikka set Δ' such that

$$\Delta' \subseteq \mathbf{Bool} \times \mathrm{SUBF}(\{\varphi \mid b\varphi \in \Delta \text{ for some } b \in \mathbf{Bool}\}).$$

Solution. By Construction 3.3.33, there is a completed Δ-tableau $\mathtt{T}$. Theorem 3.3.16 implies that $\mathtt{T}$ is not closed and therefore it contains a complete branch $\mathtt{B}$. Let Δ' consist of the signed formulas that occur in $\mathtt{B}$. Then, Δ' is a Hintikka set that contains Δ and Theorem 3.3.12 implies that it has the desired property.

(26) Give a construction whose input is a strongly closed unsigned Γ-tableau $\mathtt{T}$ and whose output is a strongly closed $\mathtt{s}(\Gamma)$-tableau $\mathtt{T}'$.
Solution. Suppose initially that $\mathtt{T}$ is a one-node tree, which means that $\mathtt{T}(\lambda)$ contains both φ and $(\neg\varphi)$ for some formula φ. If φ is positive, return the one-node tableau $\mathtt{T}'$ whose root

is labeled by $\mathtt{s}(\Gamma)$. If φ is negative, say $\varphi = (\neg\psi)$, the output is the two-node signed tableau $\mathtt{T}'$ defined by $\mathtt{T}'(\lambda) = \mathtt{s}(\Gamma)$ and $\mathtt{T}'(0) = \mathtt{s}(\Gamma) \cup \{\mathbf{T}\psi\}$.

Consider now the case when $\mathtt{T}$ has more than one node. If we used thinning at the root, that is, $\mathtt{T}(\lambda) = \Gamma$ and $\mathtt{T}(0)$ is a subset Γ_0 of Γ, apply the construction recursively to $\mathtt{T}_{[0]}$ to produce a strongly closed $\mathtt{s}(\Gamma_0)$-tableau $\mathtt{T}'_0$ and return $(\mathtt{T}'_0; \mathtt{s}(\Gamma))$.

Suppose now that $\mathtt{T}(\lambda) = \Gamma_0 \cup \{(\neg(\neg\varphi))\}$ and $\mathtt{T}(0) = \Gamma_0 \cup \{\varphi\}$. Apply the construction to $\mathtt{T}_{[0]}$ to obtain $\mathtt{T}'_0$, a signed $(\mathtt{s}(\Gamma_0) \cup \{\mathbf{T}\varphi\})$-tableau if φ is positive, or a signed $(\mathtt{s}(\Gamma_0) \cup \{\mathbf{F}\psi\})$-tableau, if $\varphi = (\neg\psi)$. In the first case, return $(\mathtt{T}'_0; \mathtt{s}(\Gamma))$; in the second case, return $((\mathtt{T}'_0; \mathtt{s}(\Gamma_0) \cup \{\mathbf{T}(\neg\psi)\}); \mathtt{s}(\Gamma))$.

Next, we treat one of the remaining cases, when we expand a formula $\varphi = (\psi \wedge \alpha)$, that is, $\mathtt{T}(\lambda) = \Gamma_0 \cup \{(\psi \wedge \alpha)\}$ and $\mathtt{T}(0) = \Gamma_0 \cup \{\psi, \alpha\}$. Apply the construction recursively to $\mathtt{T}_{[0]}$ to obtain an $(\mathtt{s}(\Gamma_0) \cup \mathtt{s}(\{\psi, \alpha\})$-tableau $\mathtt{T}'_0$. There are four cases to consider depending of whether the formulas ψ and α are negated or not. We deal here only with the case when ψ is positive and $\alpha = (\neg\theta)$. Applying a similar process as above, we return the signed tableau

$$((\mathtt{T}'_0; \mathtt{s}(\Gamma_0) \cup \{\mathbf{T}\psi, \mathbf{T}(\neg\theta)\}); \mathtt{s}(\Gamma)).$$

The remaining cases are similar and are left to the reader.

(27) Show that for every node q of an unsigned Γ-tableau $\mathtt{T}$, we have

$$\mathtt{T}(q) \subseteq \mathrm{SUBF}(\Gamma) \cup \{(\neg\varphi) \mid \varphi \in \mathrm{PRSUBF}(\Gamma)\}.$$

(This is the analyticity of tableaux for unsigned formulas.)

(28) Let Γ be a set of formulas and let $\mathtt{T}$ be a Γ-tableau.

(a) Prove that for every truth assignment v, v satisfies Γ if and only if there is a branch $\mathtt{B}$ of $\mathtt{T}$ such that v satisfies $\mathtt{B}$. Conclude that if there is a closed Γ-tableau, then Γ is unsatisfiable. (This is the Soundness Theorem for Unsigned Tableaux.)

(b) Prove that if $\mathtt{T}$ is completed, then Γ is satisfiable if and only if $\mathtt{T}$ is not closed.

(29) Let Γ be a set of formulas and let $\mathtt{T}$ be a completed Γ-tableau. Prove that a truth assignment v satisfies Γ if and only if there is a branch $\mathtt{B}$ of $\mathtt{T}$ such that $\mathtt{T}(\mathtt{B})$ is a Hintikka set, $v(p) = \mathbf{T}$ if $p \in \mathtt{T}(\mathtt{B})$, and $v(p) = \mathbf{F}$ if $(\neg p) \in \mathtt{T}(\mathtt{B})$.

(30) (a) If $\mathtt{P}$ is a path in a conservative Γ-tableau $\mathtt{T}$ ending in the node q and $\varphi \in \mathtt{T}(\mathtt{P}) - \mathtt{T}(q)$, show that there is a constituent K of φ such that $K \subseteq \mathtt{T}(\mathtt{P})$.
(b) Use Part (a) to formulate an algorithm that takes as input a finite set Γ of formulas and produces a finite completed Γ-tableau and prove that the algorithm is correct.
(c) Show that for every finite unsatisfiable set of formulas Γ, there exists a finite closed Γ-tableau.

(31) Formulate an algorithm that takes as input a finite set Γ of formulas and produces a finite strongly completed Γ-tableau and prove that the algorithm is correct. Conclude that if Γ is unsatisfiable, there is a strongly closed (hence finite) Γ-tableau. **Hint.** The algorithm is similar to Algorithm 3.3.27 and the correctness proof uses the analyticity of tableaux for unsigned formulas.

(32) Construct a formal system whose objects are the finite sets of formulas such that its theorems are the finite unsatisfiable sets of formulas.

(33) Formulate a construction that starts with a set Γ of formulas (not necessarily finite) and yields a sequence $\mathtt{T}_0, \mathtt{T}_1, \ldots$ of Γ-tableaux such that each $\mathtt{T}_{i+1}$ is a leaf extension of $\mathtt{T}_i$ and $\bigcup\{\mathtt{T}_i \mid i \geq 0\}$ is a completed Γ-tableau. Using an argument similar to that of Theorem 3.3.34, conclude that if Γ is unsatisfiable, there is a finite, closed Γ-tableau.

(34) Use Exercise 33 to reprove Corollary 2.7.22. In other words, show that if Γ is a satisfiable set of formulas, then there is a Hintikka set Γ' such that $\Gamma \subseteq \Gamma'$ and every formula in Γ' is either a subformula or the negation of a subformula of a formula of Γ.

(35) Formulate a construction that takes as input a set Γ of formulas and produces a strongly completed Γ-tableau and prove that the algorithm is correct. Conclude that if Γ is unsatisfiable, there is a strongly closed (hence finite) Γ-tableau.

(36) Use Exercise 35 to obtain another proof of the Compactness Theorem for unsigned formulas.

(37) Construct a formal system whose objects are the sets of formulas such that its theorems are the unsatisfiable sets of formulas.

(38) Formulate and prove results for sets of formulas analogous to the ones discussed in Theorem 3.3.11 and Exercise 23 for sets of signed formulas.

(39) If Γ is a set of formulas in negation normal form, and $\mathtt{T}$ is a Γ-tableau, prove that $\mathtt{T}(q) \subseteq \mathrm{NNF}$ for every $q \in \mathrm{Dom}(\mathtt{T})$.

(40) Recall the following definitions of formulas from Example 2.2.8 and Exercises 11 and 12 of Chapter 2:

$$\theta_i = \begin{cases} p_0 & \text{if } i = 0 \\ (p_{2i-1} \vee p_{2i}) & \text{if } i \geq 1 \end{cases}$$

$$\varphi_n = \bigwedge_{i=0}^{n-1} \theta_i \text{ for } n \geq 1$$

$$\alpha_i = (\varphi_i \to p_{2i-1}) \text{ for } i \geq 1$$

$$\beta_i = (\varphi_i \to p_{2i}) \text{ for } i \geq 1$$

$$\gamma_i = \begin{cases} p_0 & \text{if } i = 0 \\ (\alpha_i \vee \beta_i) & \text{for } i \geq 1. \end{cases}$$

Also, recall that in Exercise 34 of Chapter 2 we proved that the set of signed formulas $\Delta_n = \{\mathbf{T}\gamma_0, \ldots \mathbf{T}\gamma_n, \mathbf{F}p_{2n-1}, \mathbf{F}p_{2n}\}$ for $n \geq 1$ is unsatisfiable, so there is a closed Δ_n-tableau.

Prove that for each $n \geq 1$ any closed Δ_n-tableau has at least 2^{n+1} nodes. Conclude that the size of any closed Δ_n-tableau is super-polynomial in the size of Δ_n.

Solution. The proof is by induction on $n \geq 1$. Assume that $\mathtt{T}$ is a closed Δ_n-tableau. By Supplements 18 and 15, we may assume that $\mathtt{T}$ is a conservative tableau with removal. This assumption will not increase the size of $\mathtt{T}$. For the basis step, $n = 1$, $\Delta_1 = \{\mathbf{T}p_0, \mathbf{T}((p_0 \to p_1) \vee (p_0 \to p_2)), \mathbf{F}p_1, \mathbf{F}p_2\}$ is not closed and there is only one formula in Δ_1 that can be expanded. Expanding this formula, $\mathbf{T}((p_0 \to p_1) \vee (p_0 \to p_2))$, does not yield a closed tableau, so further expansions are necessary. Thus, any closed Δ_1-tableau has at least four nodes.

Suppose now that the statement holds for Δ_{n-1} for $n > 1$ and consider a closed Δ_n-tableau $\mathtt{T}$. Without loss of generality, we assume that $\mathtt{T}$ is a tableau with removal (see Supplement 15.) We shall prove that there exist two closed Δ_{n-1}-tableaux, $\hat{\mathtt{T}}, \check{\mathtt{T}}$ such that $|\hat{\mathtt{T}}| \leq |\mathtt{T}_{[0]}|$ and $|\check{\mathtt{T}}| \leq |\mathtt{T}_{[1]}|$. The inductive hypotheses will then allow us to write $|\mathtt{T}| = 1 + |\mathtt{T}_{[0]}| + |\mathtt{T}_{[1]}| \geq 1 + 2 \cdot 2^n \geq 2^{n+1}$.

Note that the formula expanded at the root of $\mathtt{T}$ is one of the formulas $\mathbf{T}\gamma_1, \ldots, \mathbf{T}\gamma_n$. We distinguish two cases, depending on the formula which is expanded.

Case 1: The formula expanded at the root of $\mathtt{T}$ is $\mathbf{T}\gamma_n$. The tableau $\mathtt{T}_{[0]}$ is a $\Delta_{[0]}$-tableau and $\mathtt{T}_{[1]}$ is a $\Delta_{[1]}$-tableau, where

$$\Delta_{[0]} = \{\mathbf{T}\gamma_0, \ldots, \mathbf{T}\gamma_{n-1}, \mathbf{T}\alpha_n, \mathbf{F}p_{2n-1}, \mathbf{F}p_{2n}\},$$
$$\Delta_{[1]} = \{\mathbf{T}\gamma_0, \ldots, \mathbf{T}\gamma_{n-1}, \mathbf{T}\beta_n, \mathbf{F}p_{2n-1}, \mathbf{F}p_{2n}\},$$

respectively. We will give the proof of the existence of $\hat{\mathtt{T}}$ and leave the entirely similar argument for $\check{\mathtt{T}}$ to the reader. Observe that the formula γ_n does not occur negatively in Δ_n. Therefore, by Exercise 19, $\mathbf{F}\gamma_n$ does not occur in any node of $\mathtt{T}$, which implies that $\mathtt{T}_{[0]}$ is a closed $\Delta_{[0]}$-tableau. Since α_n does not occur negatively in $\Delta_{[0]}$, by Supplement 20, there is a closed $\Delta'_{[0]}$-tableau $\hat{\mathtt{T}}'$, where $\Delta'_{[0]} = \{\mathbf{T}\gamma_0, \ldots, \mathbf{T}\gamma_{n-1}, \mathbf{F}\varphi_n, \mathbf{F}p_{2n-1}, \mathbf{F}p_{2n}\}$ and $|\hat{\mathtt{T}}'| \leq |\mathtt{T}_{[0]}|$. Note that φ_n does not occur positively in $\Delta'_{[0]}$. By another application of Supplement 20, there is a closed $\Delta''_{[0]}$-tableau $\hat{\mathtt{T}}''$, where

$$\Delta''_{[0]} = \{\mathbf{T}\gamma_0, \ldots, \mathbf{T}\gamma_{n-1}, \mathbf{F}(p_{2n-3} \vee p_{2n-2}), \mathbf{F}p_{2n-1}, \mathbf{F}p_{2n}\}$$

and $|\hat{\mathtt{T}}''| \leq |\hat{\mathtt{T}}'|$. Since neither p_{2n-1} nor p_{2n} occur in

$$SV(\{\mathbf{T}\gamma_0, \ldots, \mathbf{T}\gamma_{n-1}, \mathbf{F}(p_{2n-3} \vee p_{2n-2})\}),$$

by a double application of Supplement 11, there is a closed $\Delta'''_{[0]}$-tableau $\hat{\mathtt{T}}'''$, where $\Delta'''_{[0]} = \{\mathbf{T}\gamma_0, \ldots, \mathbf{T}\gamma_{n-1}, \mathbf{F}(p_{2n-3} \vee p_{2n-2})\}$ and $|\hat{\mathtt{T}}'''| = |\hat{\mathtt{T}}''|$. Since the formula $(p_{2n-3} \vee p_{2n-2})$ does not occur positively in $\Delta'''_{[0]}$, a final application of Supplement 20 yields a closed Δ_{n-1}-tableau $\hat{\mathtt{T}}$ with $|\hat{\mathtt{T}}| \leq |\hat{\mathtt{T}}'''| \leq |\mathtt{T}_{[0]}|$.

Case 2: The formula expanded at the root of $\mathtt{T}$ is γ_i for some i with $1 \leq i \leq n-1$. In this case, the tableau $\mathtt{T}_{[0]}$ is a $\Delta_{[0]}$-tableau and $\mathtt{T}_{[1]}$ is a $\Delta_{[1]}$-tableau, where

$$\Delta_{[0]} = \{\mathbf{T}\gamma_0, \ldots, \mathbf{T}\gamma_{i-1}, \mathbf{T}\alpha_i, \mathbf{T}\gamma_{i+1}, \ldots, \mathbf{T}\gamma_n, \mathbf{F}p_{2n-1}, \mathbf{F}p_{2n}\},$$
$$\Delta_{[1]} = \{\mathbf{T}\gamma_0, \ldots, \mathbf{T}\gamma_{i-1}, \mathbf{T}\beta_i, \mathbf{T}\gamma_{i+1}, \ldots, \mathbf{T}\gamma_n, \mathbf{F}p_{2n-1}, \mathbf{F}p_{2n}\},$$

respectively. As in the previous case, we prove the existence of $\hat{\mathtt{T}}$ and leave the argument for $\check{\mathtt{T}}$ to the reader. Observe that the formula γ_i does not occur negatively in Δ_n. Therefore, by Exercise 19, $\mathbf{F}\gamma_i$ does not occur in any node of $\mathtt{T}$, which implies that $\mathtt{T}_{[0]}$ is a closed $\Delta_{[0]}$-tableau. Since α_i does not occur negatively in $\Delta_{[0]}$, by Supplement 20, there is a closed $\Delta'_{[0]}$-tableau $\hat{\mathtt{T}}'$, where

$$\Delta'_{[0]} = \{\mathbf{T}\gamma_0, \ldots, \mathbf{T}\gamma_{i-1}, \mathbf{T}p_{2i-1}, \mathbf{T}\gamma_{i+1}, \ldots, \mathbf{T}\gamma_n, \mathbf{F}p_{2n-1}, \mathbf{F}p_{2n}\}$$

and we have $|\hat{\mathtt{T}}'| \leq |\mathtt{T}_{[0]}|$.

Define the unary operation $\sharp$ on PLFORM by $\psi^\sharp = \mathsf{R}(\psi, \varphi_{i+1}, \varphi_i)$, where R was defined before Exercise 71 of Chapter 2.

Let

$$\begin{aligned}\Delta''_{[0]} &= (\Delta'_{[0]})^\sharp \\ &= \{\mathbf{T}\gamma_0, \ldots, \mathbf{T}\gamma_{i-1}, \mathbf{T}p_{2i-1}, \\ &\qquad \mathbf{T}\gamma^\sharp_{i+1}, \ldots, \mathbf{T}\gamma^\sharp_n, \mathbf{F}p_{2n-1}, \mathbf{F}p_{2n}\},\end{aligned}$$

where

$$\gamma^\sharp_k = \left(\left(\bigwedge_{0\leq j\leq k-1, j\neq i} \theta_j \to p_{2k-1}\right) \vee \left(\bigwedge_{0\leq j\leq k-1, j\neq i} \theta_j \to p_{2k}\right)\right)$$

for $i+1 \leq k \leq n$. We want to construct a closed $\Delta''_{[0]}$-tableau $\hat{\mathtt{T}}''$ such that $|\hat{\mathtt{T}}''| \leq |\hat{\mathtt{T}}'|$. To this end, we shall prove a stronger statement, namely, that for each $q \in \mathrm{Dom}(\hat{\mathtt{T}}')$, there is a $(\hat{\mathtt{T}}'(q))^\sharp$-tableau $\hat{\mathtt{T}}''_q$ such that for each branch $\mathtt{B}''$ of $\hat{\mathtt{T}}''_q$, there is a branch $\mathtt{B}'$ of $\hat{\mathtt{T}}'_{[q]}$ for which $\hat{\mathtt{T}}''_q(\mathtt{B}'') = (\hat{\mathtt{T}}'_{[q]}(\mathtt{B}'))^\sharp$ and $|\hat{\mathtt{T}}''_q| \leq |\hat{\mathtt{T}}'_{[q]}|$. The needed tableau $\hat{\mathtt{T}}''$ would then be obtained by taking $q = \lambda$. The argument for this stronger result is by induction on the depth d of q in $\hat{\mathtt{T}}'$.

For $d = 0$, q is a leaf of $\hat{\mathtt{T}}'$ and we can take $\hat{\mathtt{T}}''_q$ to be the one-node tableau labeled $(\hat{\mathtt{T}}'(q))^\sharp$. Suppose now that the statement is true for nodes of depth less than $d = \mathtt{depth}(\hat{\mathtt{T}}')(q) > 0$, that $\hat{\mathtt{T}}'(q) = \Delta_q \cup \{b_q\varphi_q\}$, where $b_q\varphi_q$ is the formula expanded at q and that $\mathtt{d}(b_q\varphi_q) = (K^q_0, \ldots, K^q_{p-1})$. Then, q has p immediate descendants in $\hat{\mathtt{T}}'$ and for $0 \leq j \leq p-1$, $\hat{\mathtt{T}}'(qj) = \Delta_q \cup K^q_j$. We need to consider now two cases.

Case 2a: $\varphi_q \neq \varphi_{i+1}$. By Exercises 71 and 80 of Chapter 2, we have $\mathtt{d}((b_q\varphi_q)^\sharp) = ((K_0^q)^\sharp, \ldots, (K_{p-1}^q)^\sharp)$. By the inductive hypothesis, there are $(\Delta_q^\sharp \cup (K_j^q)^\sharp)$-tableaux $\hat{\mathtt{T}}''_{qj}$ for $0 \leq j \leq p-1$ such that the set of formulas of each branch of these tableaux is the $\sharp$-image of a branch of the corresponding tableau $\hat{\mathtt{T}}'_{[qj]}$ and $|\hat{\mathtt{T}}''_{qj}| \leq |\hat{\mathtt{T}}'_{[qj]}|$. It is straightforward to verify that the tableau $\hat{\mathtt{T}}''_q = (\hat{\mathtt{T}}''_{q0}, \ldots, \hat{\mathtt{T}}''_{qp-1}; (\hat{\mathtt{T}}'(q))^\sharp)$ is the desired tableau.
Case 2b: $\varphi_q = \varphi_{i+1}$. Observe that although φ_{i+1} occurs negatively in the formulas $\gamma_{i+1}, \ldots, \gamma_n$, it does not occur positively in $\Delta'_{[0]}$ and thus $\mathbf{T}\varphi_{i+1}$ does not occur in any node of $\hat{\mathtt{T}}'$, so $b_q = \mathbf{F}$. Consequently, $\hat{\mathtt{T}}'(q) = \Delta_q \cup \{\mathbf{F}\varphi_{i+1}\}$, $p = 2$, and $\hat{\mathtt{T}}'(q0) = \Delta_q \cup \{\mathbf{F}\varphi_i\}$ because $\varphi_{i+1} = \varphi_i \wedge \theta_i$. Since $(\mathbf{F}\varphi_{i+1})^\sharp = \mathbf{F}\varphi_i$, we have

$$(\hat{\mathtt{T}}'(q))^\sharp = (\Delta_q \cup \{\mathbf{F}\varphi_{i+1}\})^\sharp = (\Delta_q \cup \{\mathbf{F}\varphi_i\})^\sharp = (\hat{\mathtt{T}}'(q0))^\sharp.$$

By the inductive hypothesis, there is a $(\hat{\mathtt{T}}'_{q0})^\sharp$-tableau $\hat{\mathtt{T}}''_{q0}$ such that the set of formulas in each branch of this tableau is the $\sharp$-image of a branch of $\hat{\mathtt{T}}'_{[q0]}$ and $|\hat{\mathtt{T}}''_{q0}| \leq |\hat{\mathtt{T}}'_{[q0]}|$. The desired tableau $\hat{\mathtt{T}}''_q$ is $\hat{\mathtt{T}}''_{q0}$.
Note that the only appearance of p_{2i-1} in $\Delta''_{[0]}$ is in $\mathbf{T}p_{2i-1}$. Therefore, by Supplement 11, there is a closed $\Delta'''_{[0]}$-tableau $\hat{\mathtt{T}}'''$, where

$$\Delta'''_{[0]} = \{\mathbf{T}\gamma_0, \ldots, \mathbf{T}\gamma_{i-1}, \mathbf{T}\gamma^\sharp_{i+1}, \ldots, \mathbf{T}\gamma^\sharp_n, \mathbf{F}p_{2n-1}, \mathbf{F}p_{2n}\}$$

and $|\hat{\mathtt{T}}'''| = |\hat{\mathtt{T}}''|$.
Define the substitution s by

$$s(p_j) = \begin{cases} p_j & \text{if } j \leq 2i \\ p_{j-2} & \text{if } j > 2i. \end{cases}$$

Since

$$s(\theta_k) = \begin{cases} \theta_k & \text{if } k \leq i \\ \theta_{k-1} & \text{if } k > i, \end{cases}$$

we have $s(\varphi_k) = \varphi_k$ and $s(\gamma_k) = \gamma_k$, when $k \leq i$. Also, for $k > i$,

$$s(\varphi_k^\sharp) = \varphi_{k-1},$$
$$s(\alpha_k^\sharp) = \alpha_{k-1},$$
$$s(\beta_k^\sharp) = \beta_{k-1},$$
$$s(\gamma_k^\sharp) = \gamma_{k-1},$$

and this implies that $s(\Delta'''_{[0]}) = \Delta_{n-1}$. Thus, by Supplement 8, $\hat{\mathtt{T}} = s \circ \hat{\mathtt{T}}'''$ is a closed Δ_{n-1}-tableau with $|\hat{\mathtt{T}}| = |\hat{\mathtt{T}}'''| \leq |\mathtt{T}_{[0]}|$. This completes the proof of the desired lower bound for the number of nodes.

Observe that there is a constant c such that $\mathtt{size}(\Delta_n) \leq cn^2 \log n$ for sufficiently large n, by Exercise 12 of Chapter 2. Therefore, for sufficiently large n, we have $\mathtt{size}(\Delta_n) \leq n^3$, which in turn implies for every closed Δ_n-tableau $\mathtt{T}_n$,

$$\mathtt{size}(\mathtt{T}_n) \geq |\mathtt{T}_n| \geq 2^{n+1} \geq 2^{\sqrt[3]{\mathtt{size}(\Delta_n)}},$$

a super-polynomial lower bound on the size of $\mathtt{T}_n$.

(41) Let $\Delta_0 = \{\mathbf{T}\theta, \mathbf{T}((\theta \to \varphi) \vee (\theta \to \psi)), \mathbf{F}\varphi, \mathbf{F}\psi\}$ and let $\Delta_1 = \{\mathbf{T}\theta, \mathbf{T}((\theta \to \varphi) \vee (\theta \to \psi)), \mathbf{F}(\theta \wedge (\varphi \vee \psi))\}$, where $\theta, \varphi, \psi \in$ PLFORM. Show that there is a closed Δ_0-tableaux $\mathtt{T}_0(\theta, \varphi, \psi)$ and a closed Δ_1-tableau $\mathtt{T}_1(\theta, \varphi, \psi)$, such that $\mathtt{T}_0(\theta, \varphi, \psi)$ has seven nodes, $\mathtt{T}_1(\theta, \varphi, \psi)$ has ten nodes, $\mathtt{size}(\mathtt{T}_0(\theta, \varphi, \psi)) = 13 \cdot \mathtt{size}(\theta) + 10 \cdot \mathtt{size}(\varphi) + 10 \cdot \mathtt{size}(\psi) + 15$, and $\mathtt{size}(\mathtt{T}_1(\theta, \varphi, \psi)) = 24 \cdot \mathtt{size}(\theta) + 15 \cdot \mathtt{size}(\varphi) + 15 \cdot \mathtt{size}(\psi) + 51$.

Solution. The tableaux $\mathtt{T}_0(\theta, \varphi, \psi)$ and $\mathtt{T}_1(\theta, \varphi, \psi)$ are given in Figures 3.35 and 3.36, respectively.

(42) Let φ_n, γ_n be the formulas introduced in Exercises 11 and 12 of Chapter 2.

(a) Prove that for $n \geq 1$, there is a closed Δ_n^*-tableau, where $\Delta_n^* = \{\mathbf{T}\varphi_n, \mathbf{T}\gamma_n, \mathbf{F}\varphi_{n+1}\}$, that has 10 nodes and size $\Theta(n \log n)$.

(b) Prove that for $n \geq 1$, there is a closed $\Delta_n^\diamond$-tableau, where $\Delta_n^\diamond = \{\mathbf{T}\varphi_n, \mathbf{T}\gamma_n, \mathbf{F}p_{2n-1}, \mathbf{F}p_{2n}\}$, that has seven nodes and whose size is $\Theta(n \log n)$.

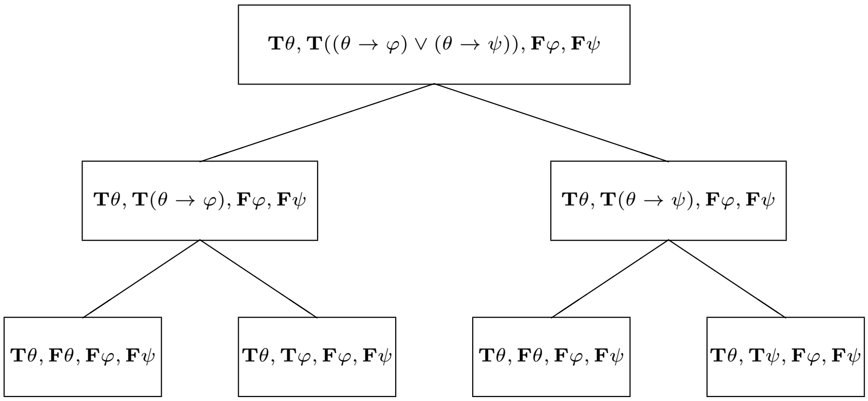

$\mathbf{T}\theta, \mathbf{T}((\theta \to \varphi) \vee (\theta \to \psi)), \mathbf{F}\varphi, \mathbf{F}\psi$
$\mathbf{T}\theta, \mathbf{T}(\theta \to \varphi), \mathbf{F}\varphi, \mathbf{F}\psi$
$\mathbf{T}\theta, \mathbf{T}(\theta \to \psi), \mathbf{F}\varphi, \mathbf{F}\psi$
$\mathbf{T}\theta, \mathbf{F}\theta, \mathbf{F}\varphi, \mathbf{F}\psi$
$\mathbf{T}\theta, \mathbf{T}\varphi, \mathbf{F}\varphi, \mathbf{F}\psi$
$\mathbf{T}\theta, \mathbf{F}\theta, \mathbf{F}\varphi, \mathbf{F}\psi$
$\mathbf{T}\theta, \mathbf{T}\psi, \mathbf{F}\varphi, \mathbf{F}\psi$

Fig. 3.35. The tableau $\mathtt{T}_0(\theta, \varphi, \psi)$.

Solution. In the notation of Exercise 41, the desired closed Δ_n^*-tableau is $\mathtt{T}_1(\varphi_n, p_{2n-1}, p_{2n})$, while the desired closed $\Delta_n^\diamond$-tableau is $\mathtt{T}_0(\varphi_n, p_{2n-1}, p_{2n})$. (The sizes of the formulas involved are computed in Exercise 11 of Chapter 2.)

The Cut Rule for Tableaux

(43) Make Case 3.2.3.2 of the Construction 3.4.11 more efficient by considering each binary connective C as a separate case.
Solution. Suppose C is $\vee$. In the analog of Case 3.2.3.2.1 of Construction 3.4.11, we have $(\mathtt{T}_0)_{[0]}$, a strongly closed $(\Delta_0 \cup \{\mathbf{T}\alpha\})$-tableau, $(\mathtt{T}_0)_{[1]}$, a strongly closed $(\Delta_0 \cup \{\mathbf{T}\beta\})$-tableau, and $(\mathtt{T}_1)_{[0]}$, a strongly closed $(\Delta_1 \cup \{\mathbf{F}\alpha, \mathbf{F}\beta\})$-tableau. Using the notation introduced in Construction 3.4.11, we notice that we can define the tableau $\mathtt{T}_{\mathbf{T}}$ as $(\mathtt{T}_0)_{[0]}$. Also, the strongly closed $(\Delta_0 \cup \{\mathbf{F}\alpha, \mathbf{T}\beta\})$-tableau $\mathtt{T}_{\mathbf{FT}}$ can be obtained as $(\mathtt{T}_0)_{[1]} \uplus \{\mathbf{F}\alpha\}$; the strongly closed $(\Delta_1 \cup \{\mathbf{F}\alpha, \mathbf{F}\beta\})$-tableau $\mathtt{T}_{\mathbf{FF}}$ can be obtained as $(\mathtt{T}_1)_{[0]}$. Now we can define the strongly closed $(\Delta_0 \cup \Delta_1 \cup \{\mathbf{F}\alpha\})$-tableau $\mathtt{T}_{\mathbf{F}}$ as $\mathtt{T}_{\mathbf{F}} = \mathsf{cet}(\mathtt{T}_{\mathbf{FT}}, \mathtt{T}_{\mathbf{FF}}, \Delta_0 \cup \{\mathbf{F}\alpha\}, \Delta_1 \cup \{\mathbf{F}\alpha\}, \beta)$. Finally, we return $\mathsf{cet}(\mathtt{T}_{\mathbf{T}}, \mathtt{T}_{\mathbf{F}}, \Delta_0, \Delta_0 \cup \Delta_1, \alpha)$. (Note that in this case, cet was called recursively only twice instead of three times.) The remaining cases for the $\vee$ connective as well as the cases for the other connectives can be treated similarly.

(44) Let Δ be a set of signed formulas, $\mathtt{T}$ be a conservative Δ-tableau with cut, and let q be a leaf of $\mathtt{T}$. Prove that if a truth assignment

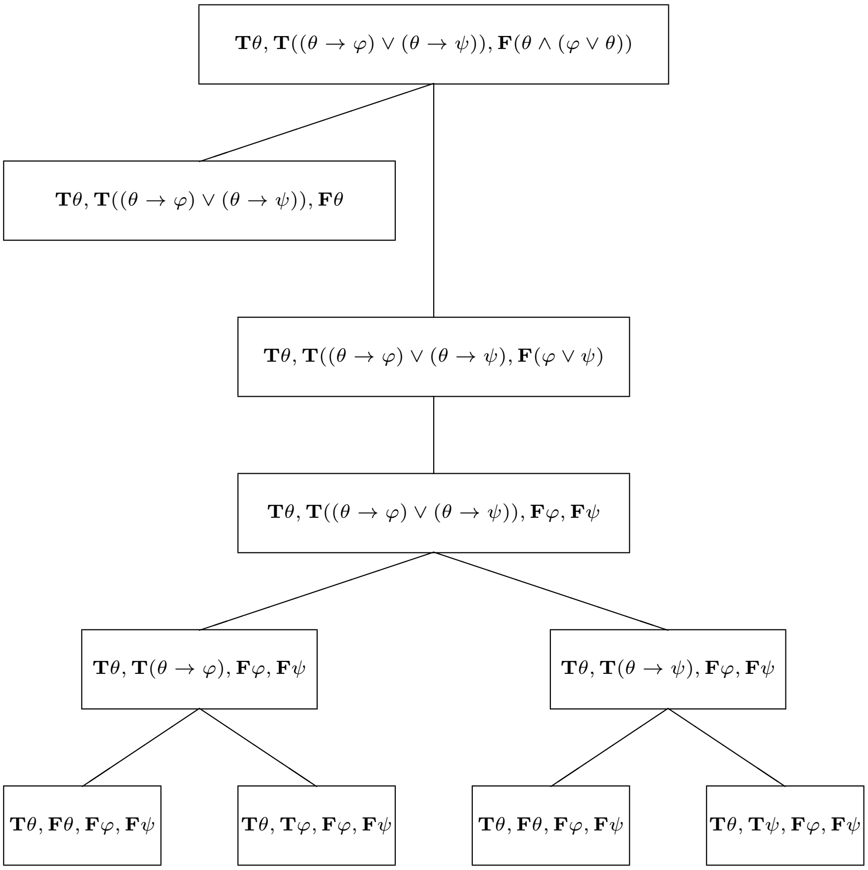

$\mathbf{T}\theta, \mathbf{T}((\theta \to \varphi) \vee (\theta \to \psi)), \mathbf{F}(\theta \wedge (\varphi \vee \theta))$
$\mathbf{T}\theta, \mathbf{T}((\theta \to \varphi) \vee (\theta \to \psi)), \mathbf{F}\theta$
$\mathbf{T}\theta, \mathbf{T}((\theta \to \varphi) \vee (\theta \to \psi), \mathbf{F}(\varphi \vee \psi)$
$\mathbf{T}\theta, \mathbf{T}((\theta \to \varphi) \vee (\theta \to \psi)), \mathbf{F}\varphi, \mathbf{F}\psi$
$\mathbf{T}\theta, \mathbf{T}(\theta \to \varphi), \mathbf{F}\varphi, \mathbf{F}\psi$
$\mathbf{T}\theta, \mathbf{T}(\theta \to \psi), \mathbf{F}\varphi, \mathbf{F}\psi$
$\mathbf{T}\theta, \mathbf{F}\theta, \mathbf{F}\varphi, \mathbf{F}\psi$
$\mathbf{T}\theta, \mathbf{T}\varphi, \mathbf{F}\varphi, \mathbf{F}\psi$
$\mathbf{T}\theta, \mathbf{F}\theta, \mathbf{F}\varphi, \mathbf{F}\psi$
$\mathbf{T}\theta, \mathbf{T}\psi, \mathbf{F}\varphi, \mathbf{F}\psi$

Fig. 3.36. The tableau $\mathtt{T}_1(\theta, \varphi, \psi)$.

v satisfies $\mathtt{T}(q)$, then it satisfies $\mathtt{T}(\mathtt{B})$, where $\mathtt{B}$ is the branch that ends in q.

It follows from Corollaries 2.7.26 and 3.3.35 that if there is a closed $(\Delta \cup \{b\varphi\})$-tableau $\mathtt{T}$, then there is a closed $(\Delta \cup K)$-tableau $\mathtt{T}_K$ for every constituent K of $b\varphi$. However, without the cut rule, we have no obvious way of obtaining $\mathtt{T}_K$ from $\mathtt{T}$. The next exercise shows that the cut rule can be used to produce $\mathtt{T}_K$.

(45) Let K be a constituent of a signed formula $b\varphi$.

(a) Prove that, using the cut rule, there is an effective way to obtain a (strongly) closed $(\Delta \cup K)$-tableau with cut from a (strongly) closed $(\Delta \cup \{b\varphi\})$-tableau with cut.

(b) Show how to obtain a strongly closed $(\Delta \cup K)$-tableau (without cut) from a strongly closed $(\Delta \cup \{b\varphi\})$-tableau (without cut), using cut elimination and Part (a).

Solution. For the first part, starting from the (strongly) closed $(\Delta \cup \{b\varphi\})$-tableau with cut $\mathtt{T}$, we construct the (strongly) closed $(\Delta \cup K \cup \{b\varphi\})$-tableau with cut $\mathtt{T}_0 = \mathtt{T} \uplus K$. Let $\mathtt{d}(\bar{b}\varphi) = (H_0, \ldots, H_{n-1})$. Define the $(\Delta \cup K \cup \{\bar{b}\varphi\})$-tableau $\mathtt{T}_1$ by $\mathrm{Dom}(\mathtt{T}_1) = \{\lambda, 0, \ldots, n-1\}$, $\mathtt{T}_1(\lambda) = \Delta \cup K \cup \{\bar{b}\varphi\}$ and $\mathtt{T}_1(i) = \Delta \cup K \cup H_i$ for $0 \le i \le n-1$. By Exercise 104 of Chapter 2, the tableau $\mathtt{T}_1$ is strongly closed. By applying the cut rule, we get the desired tableau $\mathtt{T}_K$ as $(\mathtt{T}_0, \mathtt{T}_1; \Delta \cup K)$ or as $(\mathtt{T}_1, \mathtt{T}_0; \Delta \cup K)$ depending on whether $b = \mathbf{T}$ or $b = \mathbf{F}$, respectively.

For the second part, we start with a strongly closed $(\Delta \cup \{b\varphi\})$-tableau without cut and use the construction of the first part to obtain a strongly closed $(\Delta \cup K)$-tableau with cut $\mathtt{T}'$. Then, we apply cut elimination to $\mathtt{T}'$ (which is effective).

We write $f = O(g)$, where $f, g : \mathbf{N} \longrightarrow \mathbf{R}$ if there is a positive real number c and a natural number n_0 such that $n \ge n_0$ implies $0 \le f(n) \le cg(n)$.

(46) For $n \ge 0$, let $\Delta_n^\dagger = \{\mathbf{T}\gamma_0, \ldots, \mathbf{T}\gamma_n, \mathbf{F}\varphi_{n+1}\}$, where the formulas γ_n, φ_n are defined in Exercises 11 and 12 of Chapter 2. Prove that for each n, there is a closed $\Delta_n^\dagger$-tableau with cut $\mathtt{T}_{\Delta_n^\dagger}$ having $11 \cdot n + 1$ nodes, such that $\mathtt{size}(\mathtt{T}_{\Delta_n^\dagger}) = O(n^3 \log n)$

Solution. We give a recursive construction of the tableaux $\mathtt{T}_{\Delta_n^\dagger}$.

For $n = 0$, $\mathtt{T}_{\Delta_0^\dagger}$ is the one-node tree labeled by $\Delta_0^\dagger$, which is closed.

Suppose that we have constructed $\mathtt{T}_{\Delta_n^\dagger}$ with $11 \cdot n + 1$ nodes. Consider the tableau

$$\mathtt{T}_1' = \mathtt{T}_1(\varphi_{n+1}, p_{2n+1}, p_{2n+2}) \sqcup \{\mathbf{T}\gamma_0, \ldots, \mathbf{T}\gamma_n\},$$

where $\mathtt{T}_1(\varphi_{n+1}, p_{2n+1}, p_{2n+2})$ uses the notation introduced in Exercise 41. $\mathtt{T}_1'$ is a closed $(\Delta_{n+1}^\dagger \cup \{\mathbf{T}\varphi_{n+1}\})$-tableau which has 10 nodes and size $O(n^2 \log n)$. Starting from $\mathtt{T}_{\Delta_n^\dagger}$, we construct $\mathtt{T}'_{\Delta_n^\dagger} = \mathtt{T}_{\Delta_n^\dagger} \sqcup \{\mathbf{T}\gamma_{n+1}, \mathbf{F}\varphi_{n+2}\}$ which is a closed $(\Delta_{n+1}^\dagger \cup \{\mathbf{F}\varphi_{n+1}\})$-tableau with the same number of nodes as $\mathtt{T}_{\Delta_n^\dagger}$. Finally, applying the cut rule for tableaux to the tableaux $\mathtt{T}'_1$ and $\mathtt{T}'_{\Delta_n^\dagger}$, we obtain the closed $\Delta_{n+1}^\dagger$-tableau $\mathtt{T}_{\Delta_{n+1}^\dagger} = (\mathtt{T}'_1, \mathtt{T}'_{\Delta_n^\dagger}; \Delta_{n+1}^\dagger)$ with $11(n+1)+1$ nodes. It is easy to check that $\mathtt{size}(\mathtt{T}_{\Delta_{n+1}^\dagger}) \leq \mathtt{size}(\mathtt{T}_{\Delta_n^\dagger}) + \Theta(n^2 \log n)$, which gives the desired limitation for the size.

In the next exercise, we illustrate the power of the cut rule by proving the existence of closed Δ_n-tableaux with cut having number of nodes linear in n and size polynomial in n, where Δ_n was defined in Exercise 34 of Chapter 2. In other words, by Exercise 12 of Chapter 2, we prove the existence of closed Δ_n-tableaux with cut of size polynomial in the size of Δ_n. Contrast this with the super-polynomial lower bound on the number of nodes for tableaux without cut for the same family of sets that we have shown in Supplement 40.

(47) Let Δ_n be as defined in Exercise 34 of Chapter 2. Prove that for each $n \geq 1$, there is a closed Δ_n-tableau with cut $\mathtt{T}_{\Delta_n}$ having $11 \cdot n - 2$ nodes, such that $\mathtt{size}(\mathtt{T}_{\Delta_n}) = O(n^3 \log n)$.

Solution. Let $n \geq 1$. Starting from the tableau $\mathtt{T}_0(\varphi_n, p_{2n-1}, p_{2n})$ introduced in Exercise 41 construct the closed $(\Delta_n \cup \{\mathbf{T}\varphi_n\})$-tableau

$$\mathtt{T} = \mathtt{T}_0(\varphi_n, p_{2n-1}, p_{2n}) \sqcup \{\mathbf{T}\gamma_0, \ldots, \mathbf{T}\gamma_{n-1}\}.$$

$\mathtt{T}$ has seven nodes and size $O(n^2 \log n)$. Starting from the closed $\Delta_{n-1}^\dagger$-tableau with cut introduced in Exercise 46, we construct the closed $(\Delta_n \cup \{\mathbf{F}\varphi_n\})$-tableau with cut

$$\mathtt{T}' = \mathtt{T}_{\Delta_{n-1}^\dagger} \sqcup \{\mathbf{T}\gamma_n, \mathbf{F}p_{2n-1}, \mathbf{F}p_{2n}\}$$

with $11 \cdot n - 10$ nodes and size $O(n^3 \log n)$. A final application of the cut rule gives the closed Δ_n-tableau with cut $\mathtt{T}_{\Delta_n} = (\mathtt{T}, \mathtt{T}'; \Delta_n)$ which has $11 \cdot n - 2$ nodes and size $O(n^3 \log n)$.

Let Γ be a set of formulas. A Γ-*tableau with cut* is an unsigned tableau $\mathtt{T}$ that satisfies the following conditions:

- The root of $\mathtt{T}$ is labeled by Γ, i.e., $\mathtt{T}(\lambda) = \Gamma$.
- If q is an interior node of $\mathtt{T}$, one of the following cases occurs:
  (1) There is some set of formulas Γ' and a formula φ with $\mathtt{d}(\varphi) = (K_0, \ldots, K_{n-1})$ such that $\mathtt{T}(q) = \Gamma' \cup \{\varphi\}$, q has n immediate descendants and $\mathtt{T}(qi) = \Gamma' \cup K_i$ for $0 \leq i \leq n-1$.
  (2) The node q has one immediate descendant $q0$ and $\mathtt{T}(q0) \subseteq \mathtt{T}(q)$.
  (3) There is a formula φ and a set of formulas Γ' such that q has two immediate descendants, $\mathtt{T}(q) = \Gamma'$, $\mathtt{T}(q0) = \Gamma' \cup \{\varphi\}$, and $\mathtt{T}(q1) = \Gamma' \cup \{(\neg\varphi)\}$.

(48) Let Γ be a set of formulas and let $\mathtt{T}$ be a Γ-tableau with cut. Prove that if v is a truth assignment, then v satisfies Γ if and only if it satisfies $\mathtt{T}(\mathtt{B})$ for some branch $\mathtt{B}$ of $\mathtt{T}$.

(49) Let $\mathtt{T}$ be a completed Γ-tableau with cut, where Γ is a set of unsigned formulas. Prove that a truth assignment v satisfies Γ if and only if there is a branch $\mathtt{B}$ of $\mathtt{T}$ such that $\mathtt{T}(\mathtt{B})$ is a Hintikka set and

$$v(p) = \begin{cases} \mathbf{T} & \text{if } p \in \mathtt{T}(\mathtt{B}), \\ \mathbf{F} & \text{if } (\neg p) \in \mathtt{T}(\mathtt{B}). \end{cases}$$

(50) Prove that a set of formulas Γ is unsatisfiable if and only if there exists a strongly closed Γ-tableau with cut.
Hint. The argument follows similar lines to the argument used for signed tableaux.

(51) Give a construction that starts with a strongly closed Δ-tableau with cut $\mathtt{T}$ and produces a strongly closed unsigned $\mathtt{u}(\Delta)$-tableau with cut.
Solution. The construction proceeds along the lines of Construction 3.3.47 for tableaux without cut. We need only to consider the extra step when cut is used at the root, that is, $\mathtt{T}(0) = \Delta \cup \{\mathbf{T}\varphi\}$ and $\mathtt{T}(1) = \Delta \cup \{\mathbf{F}\varphi\}$. By applying the construction recursively to the tableaux $\mathtt{T}_{[0]}$ and $\mathtt{T}_{[1]}$, we obtain the unsigned $\mathtt{u}(\Delta) \cup \{\varphi\}$-tableau $\mathtt{T}'_0$ and the unsigned $\mathtt{u}(\Delta) \cup \{(\neg\varphi)\}$-tableau $\mathtt{T}'_1$. By applying the cut rule, we return $(\mathtt{T}'_0, \mathtt{T}'_1; \mathtt{u}(\Delta))$.

(52) Give a construction that starts with a strongly closed unsigned Γ-tableau with cut $\mathtt{T}$ and produces a strongly closed signed $\mathsf{s}(\Gamma)$-tableau with cut.
Solution. The construction is similar to the construction discussed in Supplement 26. As in the previous supplement, we need only to consider the extra step when cut is used at the root. Suppose that $\mathtt{T}(0) = \Gamma \cup \{\varphi\}$ and $\mathtt{T}(1) = \Gamma \cup \{(\neg\varphi)\}$. We apply the construction recursively to $\mathtt{T}_{[0]}$ and $\mathtt{T}_{[1]}$ to obtain strongly closed tableaux with cut $\mathtt{T}'_0$ and $\mathtt{T}'_1$. We need to consider two cases based on whether φ is a nonnegated or a negated formula. In the first case, $\mathtt{T}'_0$ is an $(\mathsf{s}(\Gamma) \cup \{\mathbf{T}\varphi\})$-tableau and $\mathtt{T}_1$ is an $(\mathsf{s}(\Gamma) \cup \{\mathbf{F}\varphi\})$-tableau; by applying the cut rule, we return $(\mathtt{T}'_0, \mathtt{T}'_1; \mathsf{s}(\Gamma))$. In the second case, $\varphi = (\neg\psi)$, $\mathtt{T}'_0$ is an $(\mathsf{s}(\Gamma) \cup \{\mathbf{F}\psi\})$-tableau, $\mathtt{T}'_1$ is an $(\mathsf{s}(\Gamma) \cup \{\mathbf{F}\varphi\})$-tableau, and we return $((\mathtt{T}'_0; \mathsf{s}(\Gamma) \cup \{\mathbf{T}\varphi\}), \mathtt{T}'_1; \mathsf{s}(\Gamma))$.

(53) Give a construction that starts with a strongly closed unsigned Γ-tableau with cut $\mathtt{T}$ and produces a strongly closed unsigned Γ-tableau (without cut).
Solution. Begin by applying the construction contained in Supplement 52 to $\mathtt{T}$ to obtained a signed $\mathsf{s}(\Gamma)$-tableau $\mathtt{T}_0$ with cut. Then, apply Construction 3.4.12 to $\mathtt{T}_0$ to obtain a strongly closed $\mathsf{s}(\Gamma)$-tableau $\mathtt{T}_1$ without cut. An application of Construction 3.3.47 will yield a strongly closed unsigned $\mathsf{u}(\mathsf{s}(\Gamma))$-tableau $\mathtt{T}_2$. Taking into account the fact that $\mathsf{u}(\mathsf{s}(\Gamma)) = \Gamma$ as we saw in Exercise 96 of Chapter 2, we have obtained the desired tableau.

Sequent Systems

(54) Give proofs in $\mathcal{F}^{\text{seq}}$ for the following sequents (where φ is an arbitrary formula):

(a) $(\varphi \wedge (\neg\varphi)) \Rightarrow \emptyset$,
(b) $(\neg(\neg\varphi)) \Rightarrow \varphi$,
(c) $(\varphi \wedge \psi) \Rightarrow \varphi$.

Solution. The desired $\mathcal{F}^{\text{seq}}$-proofs are as follows:

(1)	$\varphi \Rightarrow \varphi$	Axiom
(2)	$\varphi, (\neg\varphi) \Rightarrow$	(1) and $\mathsf{R}_{\neg,l}$
(3)	$(\varphi \wedge (\neg\varphi)) \Rightarrow$	(2) and $\mathsf{R}_{\wedge,l}$

$$\frac{(\varphi \vee \alpha_0 \vee \cdots \vee \alpha_{n-1})}{((\neg(\neg\varphi)) \vee \alpha_0 \vee \cdots \vee \alpha_{n-1})}\mathsf{R}_{\neg} \qquad \frac{\begin{array}{l}((\neg\varphi) \vee \alpha_0 \vee \cdots \vee \alpha_{n-1})\\ ((\neg\psi) \vee \alpha_0 \vee \cdots \vee \alpha_{n-1})\end{array}}{((\neg(\varphi \vee \psi)) \vee \alpha_0 \vee \cdots \vee \alpha_{n-1})}\mathsf{R}_{\vee,n}$$

$$\frac{\begin{array}{l}(\varphi \vee \alpha_0 \vee \cdots \vee \alpha_{n-1})\\ (\psi \vee \alpha_0 \vee \cdots \vee \alpha_{n-1})\end{array}}{((\varphi \wedge \psi) \vee \alpha_0 \vee \cdots \vee \alpha_{n-1})}\mathsf{R}_{\wedge,p} \qquad \frac{((\neg\varphi) \vee (\neg\psi) \vee \alpha_0 \vee \cdots \vee \alpha_{n-1})}{((\neg(\varphi \wedge \psi)) \vee \alpha_0 \vee \cdots \vee \alpha_{n-1})}\mathsf{R}_{\wedge,n}$$

$$\frac{((\neg\varphi) \vee \psi \vee \alpha_0 \vee \cdots \vee \alpha_{n-1})}{((\varphi \to \psi) \vee \alpha_0 \vee \cdots \vee \alpha_{n-1})}\mathsf{R}_{\to,p} \qquad \frac{\begin{array}{l}(\varphi \vee \alpha_0 \vee \cdots \vee \alpha_{n-1})\\ ((\neg\psi) \vee \alpha_0 \vee \cdots \vee \alpha_{n-1})\end{array}}{((\neg(\varphi \to \psi)) \vee \alpha_0 \vee \cdots \vee \alpha_{n-1})}\mathsf{R}_{\to,n}$$

$$\frac{\begin{array}{l}((\neg\varphi) \vee \psi \vee \alpha_0 \vee \cdots \vee \alpha_{n-1})\\ (\varphi \vee (\neg\psi) \vee \alpha_0 \vee \cdots \vee \alpha_{n-1})\end{array}}{((\varphi \leftrightarrow \psi) \vee \alpha_0 \vee \cdots \vee \alpha_{n-1})}\mathsf{R}_{\leftrightarrow,p} \qquad \frac{\begin{array}{l}((\neg\varphi) \vee (\neg\psi) \vee \alpha_0 \vee \cdots \vee \alpha_{n-1})\\ (\varphi \vee \psi \vee \alpha_0 \vee \cdots \vee \alpha_{n-1})\end{array}}{((\neg(\varphi \leftrightarrow \psi)) \vee \alpha_0 \vee \cdots \vee \alpha_{n-1})}\mathsf{R}_{\leftrightarrow,n}$$

$$\frac{(\alpha_0 \vee \cdots \vee \alpha_i \vee \alpha_{i+1} \vee \cdots \vee \alpha_{n-1})}{(\alpha_0 \vee \cdots \vee \alpha_{i+1} \vee \alpha_i \vee \cdots \vee \alpha_{n-1})}\mathsf{R}_{\text{intch}} \qquad \frac{(\varphi_0 \vee \cdots \vee \varphi_{n-1} \vee \varphi_{n-1})}{(\varphi_0 \vee \cdots \vee \varphi_{n-1})}\mathsf{R}_{\text{cont}}$$

$$\frac{(\varphi_0 \vee \cdots \vee \varphi_{n-1})}{(\varphi_0 \vee \cdots \vee \varphi_{n-1} \vee \psi)}\mathsf{R}_{\text{thin}} \qquad \frac{\begin{array}{l}(\varphi \vee \alpha_0 \vee \cdots \vee \alpha_{n-1})\\ ((\neg\varphi) \vee \alpha_0 \vee \cdots \vee \alpha_{n-1})\end{array}}{(\alpha_0 \vee \cdots \vee \alpha_{n-1})}\mathsf{R}_{mp'} .$$

Fig. 3.37. Rules of the formal system $\mathcal{HG}$.

$$\begin{array}{lll}
(1) & \varphi \Rightarrow \varphi & \text{Axiom}\\
(2) & \Rightarrow (\neg\varphi), \varphi & (1) \text{ and } \mathsf{R}_{\neg,r}\\
(3) & (\neg(\neg\varphi)) \Rightarrow \varphi & (2) \text{ and } \mathsf{R}_{\neg,l}
\end{array}$$

and

$$\begin{array}{lll}
(1) & \varphi, \psi \Rightarrow \varphi & \text{Axiom}\\
(2) & (\varphi \wedge \psi) \Rightarrow \varphi & (1) \text{ and } \mathsf{R}_{\wedge,l}.
\end{array}$$

(55) Let κ_0, κ_1 be two sequents. Show that κ_1 dominates κ_0 if and only if there is a sequent κ such that $\kappa_0 \cup \kappa = \kappa_1$.
We introduce the formal system $\mathcal{HG}$ whose set of objects is PLFORM, set of axioms is $\{(\varphi \vee (\neg\varphi)) \mid \varphi \in \text{PLFORM}\}$, and set of rules is given in Figure 3.37:
In these rules, $\varphi, \psi, \alpha_0, \ldots, \alpha_{n-1}$ represent arbitrary formulas, in $\mathsf{R}_{\text{intch}}$, we assume $n \geq 2$ and $0 \leq i < n-1$, and in R_{cont}, R_{thin}, and $\mathsf{R}_{mp'}$, we assume $n \geq 1$. The genesis of the

rules of $\mathcal{HG}$ (except for the last four) is in the set of rules of the formal system $\mathcal{F}^{\rm seq}$; informally, we replaced sequents of the form $\varphi_0, \ldots, \varphi_{n-1} \Rightarrow \psi_0, \ldots, \psi_{m-1}$ with the formula $((\neg\varphi_0) \vee \cdots \vee (\neg\varphi_{n-1}) \vee \psi_0 \vee \cdots \vee \psi_{m-1})$. Note that there is no rule $\mathsf{R}_{\vee,p}$ since such a rule would have the same formula as both its hypothesis and conclusion. Similarly, a rule $\mathsf{R}_{\neg,p}$ would be superfluous. The rules $\mathsf{R}_{\rm intch}$, $\mathsf{R}_{\rm thin}$, and $\mathsf{R}_{\rm cont}$ are called the interchange, thinning, and contraction rules, respectively. They allow us to derive one formula which represents a sequent from another formula representing the same sequent (see Supplement 56 below.) Assuming that we can prove the formula $((\neg\alpha_0) \vee \cdots \vee (\neg\alpha_{n-1}) \vee \varphi)$, the rule $\mathsf{R}_{mp'}$ (which is a variant of the modus ponens rule R_{mp}) allows proving φ, starting from formulas $\alpha_0, \ldots, \alpha_{n-1}$.

(56) Show that the following *structural rule*

$$\frac{(\varphi_0 \vee \cdots \vee \varphi_{n-1})}{(\psi_0 \vee \cdots \vee \psi_{m-1})}\,\mathsf{R}_{\rm struc},$$

where $\{\varphi_0, \ldots, \varphi_{n-1}\} \subseteq \{\psi_0, \ldots, \psi_{m-1}\}$, is a derived rule of $\mathcal{HG}$.

Solution. The core of the argument consists in showing that if $(\varphi_0, \ldots, \varphi_{n-1}) \vdash_{\mathcal{F}_{\rm seq,PLFORM}} (\psi_0, \ldots, \psi_{m-1})$, then we also have $(\varphi_0 \vee \cdots \vee \varphi_{n-1}) \vdash_{\mathcal{HG}} (\psi_0 \vee \cdots \vee \psi_{m-1})$, where $\mathcal{F}_{{\rm seq},M}$ is the formal system introduced in Example 1.8.11. This can be seen by observing the correspondence between the rules $\mathsf{R}_{\rm intch}$, $\mathsf{R}_{\rm exp}$, and $\mathsf{R}_{\rm cont}$ of $\mathcal{F}_{seq,\rm PLFORM}$ and the rules $\mathsf{R}_{\rm intch}$, $\mathsf{R}_{\rm thin}$, and $\mathsf{R}_{\rm cont}$ of $\mathcal{HG}$ (see Example 1.8.15.)

(57) Show that the rule

$$\frac{\varphi, ((\neg\varphi) \vee \alpha_0 \vee \cdots \vee \alpha_{n-1})}{(\alpha_0 \vee \cdots \vee \alpha_{n-1})}\,\mathsf{R}_{mp''}$$

is a derived rule of the formal system $\mathcal{HG}$.

Hint. Use R_{thin} and $\mathsf{R}_{mp'}$.

We define the mapping

$$\delta : (\text{SPLFORM} - (\mathbf{Bool} \times SV)) \longrightarrow \text{Seq}(\text{Seq}(\text{SPLFORM}))$$

which is similar to the mapping d introduced in Definition 2.7.24 by the following table:

Signed Formula $b\alpha$	$\delta(b\alpha)$
$\mathbf{T}(\neg\varphi)$	$((\mathbf{F}\varphi))$
$\mathbf{F}(\neg\varphi)$	$((\mathbf{T}\varphi))$
$\mathbf{T}(\varphi \wedge \psi)$	$((\mathbf{T}\varphi, \mathbf{T}\psi))$
$\mathbf{F}(\varphi \wedge \psi)$	$((\mathbf{F}\varphi), (\mathbf{F}\psi))$
$\mathbf{T}(\varphi \vee \psi)$	$((\mathbf{T}\varphi), (\mathbf{T}\psi))$
$\mathbf{F}(\varphi \vee \psi)$	$((\mathbf{F}\varphi, \mathbf{F}\psi))$
$\mathbf{T}(\varphi \rightarrow \psi)$	$((\mathbf{F}\varphi), (\mathbf{T}\psi))$
$\mathbf{F}(\varphi \rightarrow \psi)$	$((\mathbf{T}\varphi, \mathbf{F}\psi))$
$\mathbf{T}(\varphi \leftrightarrow \psi)$	$((\mathbf{T}\varphi, \mathbf{T}\psi), (\mathbf{F}\varphi, \mathbf{F}\psi))$
$\mathbf{F}(\varphi \leftrightarrow \psi)$	$((\mathbf{T}\varphi, \mathbf{F}\psi), (\mathbf{F}\varphi, \mathbf{T}\psi))$

(58) Let $b\varphi \in \text{SPLFORM} - (\mathbf{Bool} \times SV)$, $\delta(b\varphi) = (k_0, \ldots, k_{m-1})$. Prove that if $L = \{b_j\psi_j \mid 0 \leq j \leq m-1\}$, where $b_j\psi_j$ occurs in k_j for each j, $0 \leq j \leq m-1$ and L does not contain both $\mathbf{T}\gamma$ and $\mathbf{F}\gamma$ for any formula γ, then there is a constituent H of $\overline{b\varphi}$ such that $H = \{\overline{b_j\psi_j} \mid 0 \leq j \leq m-1\}$.

Solution. The argument is identical to the one of Supplement 105 of Chapter 2.

(59) Show that the first eight rules of the formal system $\mathcal{HG}$ have the following form: for some formulas $\theta, \alpha_0, \ldots, \alpha_{n-1}$, the conclusion of the rule is $(\theta \vee \alpha_0 \vee \cdots \vee \alpha_{n-1})$, while the hypotheses are

$$(\bigvee \mathsf{u}(\overline{k_0}) \vee \alpha_0 \vee \cdots \vee \alpha_{n-1})$$
$$\vdots$$
$$(\bigvee \mathsf{u}(\overline{k_{m-1}}) \vee \alpha_0 \vee \cdots \vee \alpha_{n-1}),$$

where $\delta(\overline{\mathsf{s}(\theta)}) = (k_0, \ldots, k_{m-1})$. (Here u, s are the functions introduced on page 267.)

Conversely, if θ is not a literal and does not have the form $(\alpha \vee \psi)$, then

$$\frac{\begin{array}{l}(\bigvee \mathsf{u}(\overline{k_0}) \vee \alpha_0 \vee \cdots \vee \alpha_{n-1})\\ \vdots \\ (\bigvee \mathsf{u}(\overline{k_{m-1}}) \vee \alpha_0 \vee \cdots \vee \alpha_{n-1})\end{array}}{(\theta \vee \alpha_0 \cdots \vee \alpha_{n-1})}$$

is an instance of one of the first eight rules of $\mathcal{HG}$.

(60) Show the soundness of $\mathcal{HG}_\Gamma$; in other words, show that for every formula φ and set of formulas Γ, we have $\Gamma \vdash_{\mathcal{HG}} \varphi$ implies $\Gamma \models \varphi$.

Solution. It is clear that the axioms of $\mathcal{HG}$ are tautologies. Therefore, it suffices to show that every truth assignment that satisfies the hypotheses of a rule of $\mathcal{HG}$ also satisfies its conclusion. For the first eight rules, this can be shown using individual arguments for each rule. We give here a uniform argument based on Exercise 59. Let v satisfy

$$(\bigvee \mathsf{u}(\overline{k_0}) \vee \alpha_0 \vee \cdots \vee \alpha_{n-1})$$
$$\vdots$$
$$(\bigvee \mathsf{u}(\overline{k_{m-1}}) \vee \alpha_0 \vee \cdots \vee \alpha_{n-1}),$$

where $\delta(\overline{\mathsf{s}(\theta)}) = (k_0, \ldots, k_{m-1})$. Denote by K_i the set of formulas that appear in the sequence k_i, for $0 \leq i \leq m-1$. If v satisfies any of the formulas α_i, then it clearly satisfies the conclusion. Otherwise, v must satisfy every disjunction $\bigvee \mathsf{u}(\overline{k_i})$ for $0 \leq i \leq m-1$, that is, for every i, $0 \leq i \leq m-1$, there is a signed formula $b_i\psi_i \in K_i$ such that v satisfies $\mathsf{u}(\overline{b_i\psi_i})$. By Exercise 97 of Chapter 2, v satisfies $\overline{b_i\psi_i}$. This shows that the set $\{b_i\psi_i \mid 0 \leq i \leq m-1\}$ does not contain both $\mathbf{T}\alpha$ and $\mathbf{F}\alpha$ for any $\alpha \in$ PLFORM. By Exercise 58 of Chapter 2, $\{\overline{b_i\psi_i} \mid 0 \leq i \leq m-1\}$ is a constituent of $\overline{\overline{\mathsf{s}(\theta)}} = \mathsf{s}(\theta)$ which allows us to conclude that v satisfies $\mathsf{s}(\theta)$, and thus, it satisfies θ. We leave to the reader the argument for the last four rules.

(61) Consider the mapping $\Lambda : \mathsf{SQT} \longrightarrow \mathcal{P}(\text{PLFORM})$ defined by

$$\Lambda(\kappa) = \mathsf{u}(\overline{\mathtt{sf}(\kappa)})$$

for every $\kappa \in \mathsf{SQT}$. In other words, we have

$$\Lambda(\Gamma \Rightarrow \Gamma') = \{(\neg\varphi) \mid \varphi \in \Gamma\} \cup \Gamma'.$$

Show, by induction on the definition of proof trees, that for every proof tree $\mathtt{T} \in \mathcal{PT}_{\mathcal{F}^{\text{seq},\infty}}$, if $\mathtt{T}(\lambda) = \kappa$, then there exist $\alpha_0, \ldots, \alpha_{n-1} \in \Lambda(\kappa)$ with $n > 0$ such that

$$\vdash_{\mathcal{HG}} (\alpha_0 \vee \cdots \vee \alpha_{n-1})$$

by a proof that does not use the $\mathsf{R}_{mp'}$ rule.

Solution. The case when $\mathtt{T}$ is a one-node proof tree is straightforward. Let $\mathtt{T}_0, \dots, \mathtt{T}_{n-1}$ be $\mathcal{F}^{\mathrm{seq},\infty}$-proof trees that satisfy the condition and let $((\mathtt{T}_0(\lambda), \dots, \mathtt{T}_{n-1}(\lambda)), \kappa)$ be an instance of a rule of $\mathcal{F}^{\mathrm{seq},\infty}$ for some sequent κ. We intend to show that the proof tree $\mathtt{T} = (\mathtt{T}_0, \dots, \mathtt{T}_{n-1}; \kappa)$ also satisfies the condition. If thinning was not used at the root of $\mathtt{T}$, by Theorem 3.5.10, there is a signed formula $b\varphi$ and a set of signed formulas Δ such that $\mathtt{d}(b\varphi) = (K_0, \dots, K_{n-1})$, $\mathtt{T}_i(\lambda) = \mathtt{sqt}(\Delta \cup K_i)$ for $0 \leq i \leq n-1$ and $\kappa = \mathtt{sqt}(\Delta \cup \{b\varphi\})$. Observe that $\Lambda(\mathtt{sqt}(\Delta \cup K_i)) = \mathsf{u}(\overline{\Delta}) \cup \mathsf{u}(\overline{K_i})$. Let $\delta(b\varphi) = (k_0, \dots, k_{n-1})$. Applying the inductive hypothesis and the structural rule, there are formulas $\alpha_0, \dots, \alpha_{m-1} \in \mathsf{u}(\overline{\Delta})$ such that for every i, $0 \leq i \leq n-1$,

$$\vdash_{\mathcal{HG}} (\bigvee \mathsf{u}(\overline{k_i}) \vee \alpha_0 \vee \cdots \vee \alpha_{m-1}).$$

Suppose initially that $b\varphi$ has neither of the forms $\mathbf{F}(\alpha \vee \beta)$ and $\mathbf{F}(\neg\alpha)$. For $\theta = \mathsf{u}(\overline{b\varphi})$, we have, by Exercises 96 of Chapter 2 and 59, $\delta(\overline{\mathsf{s}(\theta)}) = (k_0, \dots, k_{m-1})$, so

$$\vdash_{\mathcal{HG}} (\mathsf{u}(\overline{b\varphi}) \vee \alpha_0 \vee \cdots \vee \alpha_{m-1})$$

and the conclusion follows immediately, observing that $\Lambda(\mathtt{sqt}(\Delta \cup \{b\varphi\})) = \mathsf{u}(\overline{\Delta}) \cup \mathsf{u}(\{\overline{b\varphi}\})$.
If $b\varphi = \mathbf{F}(\alpha \vee \beta)$, then $\delta(b\varphi) = ((\mathbf{F}\alpha, \mathbf{F}\beta))$, so $n = 1$ and $k_0 = (\mathbf{F}\alpha, \mathbf{F}\beta)$. Thus, $\bigvee \mathsf{u}(\overline{k_0}) = \alpha \vee \beta = \mathsf{u}(\overline{b\varphi})$, so the inductive hypothesis and the structural rule give the conclusion immediately, without applying any rules of $\mathcal{HG}$.
If $b\varphi = \mathbf{F}(\neg\alpha)$, then $n = 1$ and $K_0 = \{\mathbf{T}\alpha\}$. Thus,

$$\begin{aligned}\Lambda(\mathtt{T}_0(\lambda)) &= \Lambda(\mathtt{sqt}(\Delta \cup K_0)) = \Lambda(\mathtt{sqt}(\Delta \cup \{\mathbf{T}\alpha\}))\\ &= \mathsf{u}(\overline{\Delta \cup \{\mathbf{T}\alpha\}}) \qquad = \mathsf{u}(\overline{\Delta}) \cup \{(\neg\alpha)\}\\ &= \mathsf{u}(\overline{\Delta \cup \{\mathbf{F}(\neg\alpha)\}}) = \Lambda(\mathtt{sqt}(\Delta \cup \{\mathbf{F}(\neg\alpha)\}))\\ &= \Lambda(\kappa),\end{aligned}$$

so the inductive hypothesis implies the conclusion.
We leave the case when thinning was used at the root to the reader.

(62) Show that if φ is a tautology, then $\vdash_{\mathcal{HG}} \varphi$ by a proof that does not use $\mathsf{R}_{mp'}$.
Solution. The sequent $\Rightarrow \varphi$ is valid because φ is a tautology. Therefore, by Theorem 3.5.18, there is an $\mathcal{F}^{\text{seq},\infty}$-proof tree $\mathtt{T}$ for $\Rightarrow \varphi$. By Supplement 61, we have

$$\vdash_{\mathcal{HG}} (\underbrace{\varphi \vee \cdots \vee \varphi}_{n \geq 1})$$

by a proof that does not use $\mathsf{R}_{mp'}$. If $n > 1$, by the structural rule, we obtain $\vdash_{\mathcal{HG}} \varphi$.

(63) Starting from the $\mathcal{F}^{\text{seq},\infty}$-proof tree in Example 3.5.17, use the technique introduced in Supplement 61 to give a proof in $\mathcal{HG}$ of the formula $(((\neg\alpha) \to (\neg\beta)) \to (\beta \to \alpha))$ without using $\mathsf{R}_{mp'}$.

(64) Show that for every formula φ and set of formulas Γ, if $\Gamma \models \varphi$, then $\Gamma \vdash_{\mathcal{HG}} \varphi$.
Solution. Since $\Gamma \models \varphi$, the sequent $\Gamma \Rightarrow \varphi$ is valid. Therefore, by Theorem 3.5.18, there is an $\mathcal{F}^{\text{seq},\infty}$-proof tree $\mathtt{T}$ for $\Gamma \Rightarrow \varphi$. Since $\Lambda(\Gamma \Rightarrow \varphi) = \{(\neg\alpha) \mid \alpha \in \Gamma\} \cup \{\varphi\}$, by Supplement 61, and possibly an application of the structural rule, we have $\vdash_{\mathcal{HG}} ((\neg\alpha_0) \vee \cdots \vee (\neg\alpha_{n-1}) \vee \varphi)$ for some $\alpha_0, \ldots, \alpha_{n-1} \in \Gamma$. By n applications of the derived rule $\mathsf{R}_{mp''}$, we obtain $\Gamma \vdash_{\mathcal{HG}} \varphi$.

(65) Let Γ be a set of formulas and let Γ-$\mathcal{HG}$ be the formal system obtained from $\mathcal{HG}$ by replacing the rule $\mathsf{R}_{mp'}$ by the rule $\mathsf{R}_{\Gamma\text{-}mp}$ given by

$$\frac{((\neg\psi) \vee \alpha_0 \vee \cdots \vee \alpha_{n-1})}{\alpha_0 \vee \cdots \vee \alpha_{n-1}},$$

where $\psi \in \Gamma$. Prove that $\Gamma \models \varphi$ if and only if $\vdash_{\Gamma\text{-}\mathcal{HG}} \varphi$. Explain why the formal system Γ-$\mathcal{HG}$ is analytic.

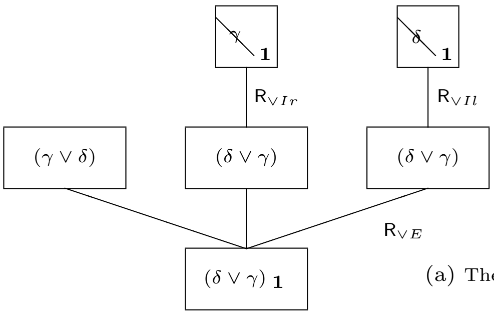
(66) Give a direct (i.e., not using tableaux), syntactic transformation of a $\mathcal{F}^{\text{seq},\infty,\text{cut}}$-proof tree for a sequent κ into a $\mathcal{F}^{\text{seq},\infty}$-proof tree for the same sequent.

Natural Deduction

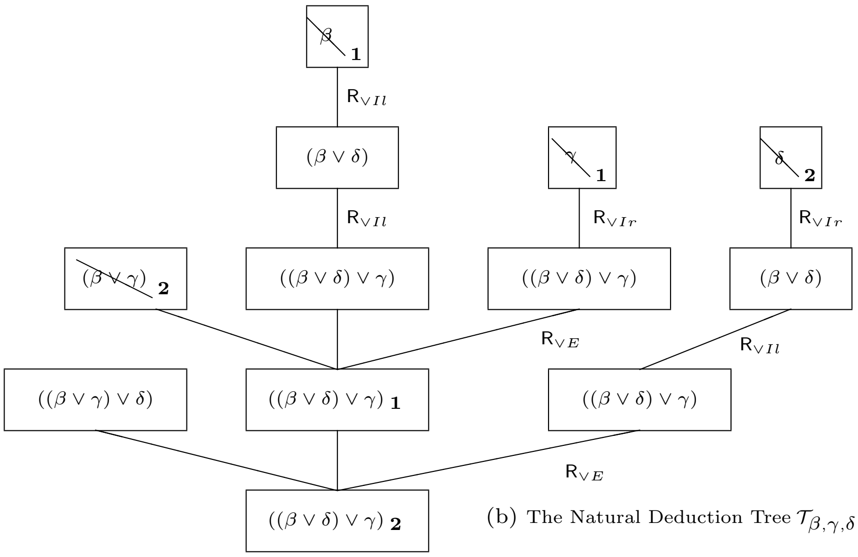
(67) Let γ, δ, β be formulas.

(a) Show that $(\gamma \vee \delta) \overset{\bullet}{\vdash}_{\text{nd}} (\delta \vee \gamma)$.

(b) Show that $((\beta \vee \gamma) \vee \delta) \overset{\bullet}{\vdash}_{\text{nd}} ((\beta \vee \delta) \vee \gamma)$.



(a) The Natural Deduction Tree $\mathcal{T}_{\gamma,\delta}$



(b) The Natural Deduction Tree $\mathcal{T}_{\beta,\gamma,\delta}$

Fig. 3.38. The natural deduction trees $\mathcal{T}_{\gamma,\delta}$ and $\mathcal{T}_{\beta,\gamma,\delta}$.

Solution. The required natural deduction trees $\mathcal{T}_{\gamma,\delta}$ and $\mathcal{T}_{\beta,\gamma,\delta}$ are given in Figure 3.38.

A marked lot $\mathcal{T}' = (\mathtt{T}, M')$ is *at least as marked* as a marked lot $\mathcal{T} = (\mathtt{T}, M)$ if $M \subseteq M'$.

(68) Let $\mathcal{T} = (\mathtt{T}, M)$, $\mathcal{T}' = (\mathtt{T}', M')$ be two natural deduction trees. Show that if r is a node of $\mathcal{T}$ such that $\mathtt{T}(r) = \mathtt{T}'(\lambda)$, then one can effectively find a marked lot $\mathcal{T}''$ that is at least as marked as $\mathcal{T}'$ such that $\mathcal{T}[r \to \mathcal{T}'']$ is a natural deduction tree.

Solution. If $r = \lambda$, then $\mathcal{T}[r \to \mathcal{T}''] = \mathcal{T}''$, so we can take $\mathcal{T}'' = \mathcal{T}'$. We now proceed by induction on the definition of natural deduction trees, specifically on $\mathcal{T}$. The basis step is immediate from the initial observation. We carry out the inductive step for the case when $\mathcal{T}$ is obtained by the $\vee$-elimination rule, that is,

$$\mathcal{T} = (\mathcal{T}_0, L_\varphi(\mathcal{T}_1), L_\psi(\mathcal{T}_2); \theta),$$

where $\mathtt{T}_0(\lambda) = (\varphi \vee \psi)$, $\mathtt{T}_1(\lambda) = \mathtt{T}_2(\lambda) = \theta$, and $\mathcal{T}_i = (\mathtt{T}_i, M_i)$ for $0 \leq i \leq 2$. The case $r = \lambda$ is covered by the initial observation; therefore, we assume that $r = ir'$, where $i \in \{0, 1, 2\}$. For $i = 0$, $\mathtt{T}_0(r') = \mathtt{T}(0r') = \mathtt{T}'(\lambda)$, so, by the inductive hypothesis, there is a marked lot $\mathcal{T}''$ at least as marked as $\mathcal{T}'$ such that $\mathcal{T}_0[r' \to \mathcal{T}'']$ is a natural deduction tree and $\mathcal{T}_0[r' \to \mathcal{T}''](\lambda) = (\varphi \vee \psi)$. Using Exercise 102 of Chapter 1, we can write

$$\mathcal{T}[r \to \mathcal{T}''] = (\mathcal{T}_0[r' \to \mathcal{T}''], L_\varphi(\mathcal{T}_1), L_\psi(\mathcal{T}_2); \theta).$$

This proves that $\mathcal{T}[r \to \mathcal{T}'']$ is a natural deduction tree.
If $r = 1r'$, then there is a marked lot $\mathcal{T}'''$ that is at least as marked as $\mathcal{T}'$ such that $\mathcal{T}_1[r' \to \mathcal{T}''']$ is a natural deduction tree. Let $\mathcal{T}'' = L_\varphi(\mathcal{T}''')$. Clearly, $\mathcal{T}''$ is at least as marked as $\mathcal{T}'''$ and, therefore, it is at least as marked as $\mathcal{T}'$. Using Exercises 102 and 101 of Chapter 1, we have

$$\begin{aligned}\mathcal{T}[r \to \mathcal{T}''] &= \mathcal{T}[1r' \to L_\varphi(\mathcal{T}''')] \\ &= (\mathcal{T}_0, L_\varphi(\mathcal{T}_1)[r' \to L_\varphi(\mathcal{T}''')], L_\psi(\mathcal{T}_2); \theta) \\ &= (\mathcal{T}_0, L_\varphi(\mathcal{T}_1[r' \to \mathcal{T}''']), L_\psi(\mathcal{T}_2); \theta).\end{aligned}$$

Again, this shows that $\mathcal{T}[r \to \mathcal{T}'']$ is a natural deduction tree.
The case when $r = 2r'$ is similar.
We leave to the reader the remaining inductive steps.
Let NDT$'$ be the subset of NDT obtained by removing Part 5 of Definition 3.6.7, that is, the $\neg$-introduction rule.

(69) Show that Supplement 68 remains valid when all the natural deduction trees mentioned in the statement belong to NDT$'$.
Let Γ be a set of formulas and let φ be a formula. We write $\Gamma \overset{\bullet}{\vdash}_{\mathrm{nd}'} \varphi$ if there is a natural deduction tree $(\mathtt{T}, M) \in \mathrm{NDT}'$ such that $\mathtt{T}(\lambda) = \varphi$ and $\mathcal{UNC}(\mathtt{T}, M) \subseteq \Gamma$.

(70) Prove that for any set of formulas Γ and formula φ we have $\Gamma \overset{\bullet}{\vdash}_{\mathrm{nd}} \varphi$ if and only if $\Gamma \overset{\bullet}{\vdash}_{\mathrm{nd}'} \varphi$.

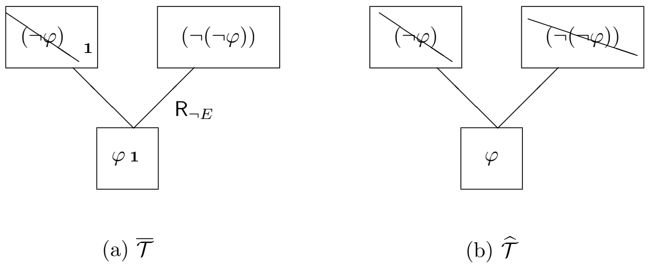


Fig. 3.39. Labeled ordered trees $\overline{\mathcal{T}}, \widehat{\mathcal{T}}$.

Solution. It is clear that $\Gamma \overset{\bullet}{\vdash}_{\text{nd}'} \varphi$ implies $\Gamma \overset{\bullet}{\vdash}_{\text{nd}} \varphi$. To prove the reverse implication, it suffices to show that for every natural deduction tree $\mathcal{T} = (\mathtt{T}, M) \in \text{NDT}$ there is a natural deduction tree $\mathcal{T}' = (\mathtt{T}', M') \in \text{NDT}'$ such that $\mathtt{T}(\lambda) = \mathtt{T}'(\lambda)$ and $\mathcal{UNC}(\mathcal{T}') \subseteq \mathcal{UNC}(\mathcal{T})$. The argument is by induction on the definition of NDT. The basis step is trivial. We discuss the only nontrivial inductive step, namely, where $\mathcal{T} = (L_\varphi(\mathcal{T}_0), L_\varphi(\mathcal{T}_1); (\neg\varphi))$ is obtained by the negation introduction rule and we can apply the inductive hypothesis to $\mathcal{T}_0 = (\mathtt{T}_0, M_0)$ and $\mathcal{T}_1 = (\mathtt{T}_1, M_1)$ to obtain $\mathcal{T}_0' = (\mathtt{T}_0', M_0')$ and $\mathcal{T}_1' = (\mathtt{T}_1', M_1')$ in NDT$'$, with $\mathtt{T}_i'(\lambda) = \mathtt{T}_i(\lambda)$ and $\mathcal{UNC}(\mathcal{T}_i') \subseteq \mathcal{UNC}(\mathcal{T}_i)$ for $i = 0, 1$. Let $\overline{\mathcal{T}}$ be the three node marked lot in NDT$'$ shown in Figure 3.39(a). Note that there is only one lot $\widehat{\mathcal{T}}$ (shown in Figure 3.39(b)) that is at least as marked as $\overline{\mathcal{T}}$ but different from it. Starting from $\mathcal{T}_0' \in \text{NDT}'$ and using the result of Exercise 69, we replace successively the leaves labeled φ by either $\overline{\mathcal{T}}$ or $\widehat{\mathcal{T}}$. The resulting sequence of marked lots consists of elements of NDT$'$ whose roots have the same label as $\mathcal{T}_0'$. If $\mathcal{T}_0''$ is the last member of the sequence, then $\mathcal{UNC}(\mathcal{T}_0'') \subseteq (\mathcal{UNC}(\mathcal{T}_0') \cup \{(\neg(\neg\varphi))\}) - \{\varphi\} \subseteq (\mathcal{UNC}(\mathcal{T}_0) \cup \{(\neg(\neg\varphi))\}) - \{\varphi\}$. A similar construction, which starts from $\mathcal{T}_1'$ yields a member $\mathcal{T}_1''$ of NDT$'$ whose root has the same label as $\mathcal{T}_1'$ and for which $\mathcal{UNC}(\mathcal{T}_1'') \subseteq (\mathcal{UNC}(\mathcal{T}_1') \cup \{(\neg(\neg\varphi))\}) - \{\varphi\} \subseteq (\mathcal{UNC}(\mathcal{T}_1) \cup \{(\neg(\neg\varphi))\}) - \{\varphi\}$. Let

$$\mathcal{T}' = (L_{(\neg(\neg\varphi))}(\mathcal{T}_0''), L_{(\neg(\neg\varphi))}(\mathcal{T}_1''); (\neg\varphi)).$$

Since $\mathcal{T}'$ was obtained by $\mathsf{R}_{\neg E}$, it is clear that $\mathcal{T}' \in \text{NDT}'$. Further, its root has the same label as the root of $\mathcal{T}$ and we

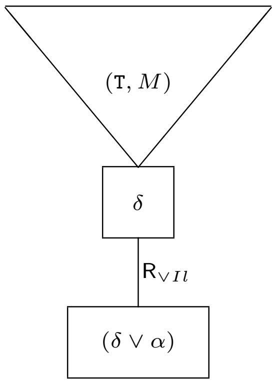


Fig. 3.40. The natural deduction tree $\Omega_0((\mathtt{T}, M), \vec{\gamma}, \delta, \alpha, m, 0)$.

have

$$\begin{aligned}\mathcal{UNC}(\mathcal{T}') &= (\mathcal{UNC}(\mathcal{T}_0'') - \{(\neg(\neg\varphi))\}) \cup (\mathcal{UNC}(\mathcal{T}_1'') - \{(\neg(\neg\varphi))\})\\ &\subseteq (\mathcal{UNC}(\mathcal{T}_0) - \{\varphi\}) \cup (\mathcal{UNC}(\mathcal{T}_1) - \{\varphi\})\\ &= \mathcal{UNC}(\mathcal{T}).\end{aligned}$$

(71) Give a recursive definition of a partial function Ω_0 from the set

$$\text{NDT} \times \text{Seq(PLFORM)} \times \text{PLFORM} \times \text{PLFORM} \times \mathbf{N} \times \mathbf{N}$$

to NDT such that $\Omega_0((\mathtt{T}, M), \vec{\gamma}, \delta, \alpha, m, i)$ is defined if and only if $\vec{\gamma} = (\gamma_0, \dots, \gamma_{m-1})$, $0 \leq i \leq m$, $\mathcal{UNC}(\mathtt{T}, M) \subseteq \{\gamma_0, \dots, \gamma_{m-1}\}$, and $\mathtt{T}(\lambda) = \delta$, and, when defined, $\Omega_0((\mathtt{T}, M), \vec{\gamma}, \delta, \alpha, m, i) = (\mathtt{T}', M')$, where $\mathcal{UNC}(\mathtt{T}', M') \subseteq \{(\gamma_0 \vee \alpha), \dots, (\gamma_{i-1} \vee \alpha), \gamma_i, \dots, \gamma_{m-1}\}$ and $\mathtt{T}'(\lambda) = (\delta \vee \alpha)$.
Solution. For $i = 0$, let $\Omega_0((\mathtt{T}, M), \vec{\gamma}, \delta, \alpha, m, 0)$ be the natural deduction tree obtained from $(\mathtt{T}, M)$ by applying the rule $\mathsf{R}_{\vee Il}$ as shown in Figure 3.40. Suppose we have defined

$$(\mathtt{T}', M') = \Omega_0((\mathtt{T}, M), \vec{\gamma}, \delta, \alpha, m, i),$$

where $\mathcal{UNC}(\mathtt{T}', M') \subseteq \{(\gamma_0 \vee \alpha), \dots, (\gamma_{i-i} \vee \alpha), \gamma_i, \dots, \gamma_{m-1}\}$ and $\mathtt{T}'(\lambda) = (\delta \vee \alpha)$. Then, $\Omega_0((\mathtt{T}, M), \vec{\gamma}, \delta, \alpha, m, i+1)$ is obtained from $(\mathtt{T}', M')$ by applying the rule $\mathsf{R}_{\vee E}$ a shown in Figure 3.41.

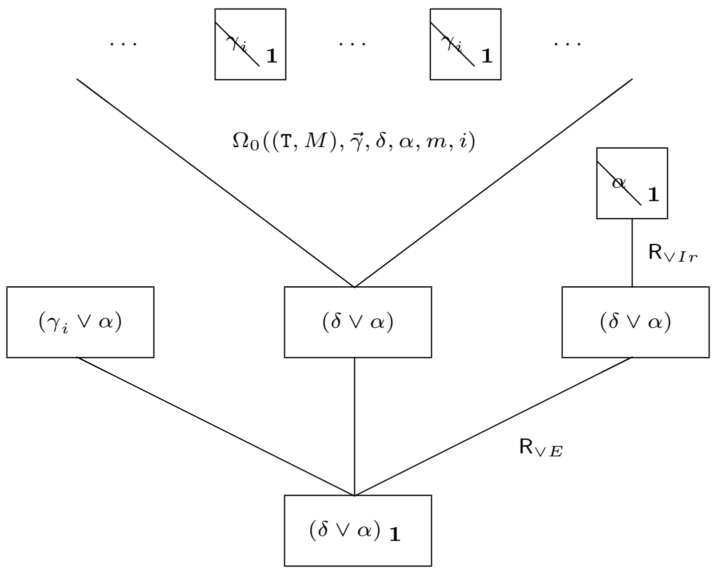


Fig. 3.41. The natural deduction tree $\Omega_0((\mathtt{T}, M), \vec{\gamma}, \delta, \alpha, m, i+1)$.

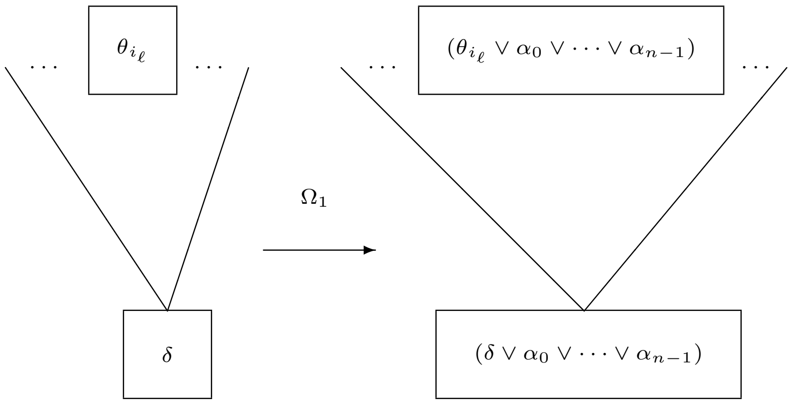
(72) Using the partial function Ω_0 from Supplement 71, give a recursive definition of a partial function Ω_1 from the set

$$\text{NDT} \times \text{Seq(PLFORM)} \times \text{PLFORM} \times \text{Seq(PLFORM)} \times \mathbf{N} \times \mathbf{N}$$

to NDT such that $\Omega_1((\mathtt{T}, M), \vec{\theta}, \delta, \vec{\alpha}, m, n)$ is defined if and only if we have $\vec{\theta} = (\theta_0, \ldots, \theta_{m-1})$, $\vec{\alpha} = (\alpha_0, \ldots, \alpha_{n-1})$, $\mathcal{UNC}(\mathtt{T}, M) \subseteq \{\theta_0, \ldots, \theta_{m-1}\}$ and $\mathtt{T}(\lambda) = \delta$, (so that $\{\theta_0, \ldots, \theta_{m-1}\} \overset{\bullet}{\vdash}_{\text{nd}} \delta$). If defined, $\Omega_1((\mathtt{T}, M), \vec{\theta}, \delta, \vec{\alpha}, m, n) = (\mathtt{T}', M')$, where

$$\mathcal{UNC}(\mathtt{T}', M') \subseteq \{(\theta_0 \vee \alpha_0 \vee \cdots \vee \alpha_{n-1}), \ldots, (\theta_{m-1} \vee \alpha_0 \vee \cdots \vee \alpha_{n-1})\}$$

and $\mathtt{T}'(\lambda) = (\delta \vee \alpha_0 \vee \cdots \vee \alpha_{n-1})$ (which means that $\{(\theta_0 \vee \alpha_0 \vee \cdots \vee \alpha_{n-1}), \ldots, (\theta_{m-1} \vee \alpha_0 \vee \cdots \vee \alpha_{n-1})\} \overset{\bullet}{\vdash}_{\text{nd}} (\delta \vee \alpha_0 \vee \cdots \vee \alpha_{n-1})$). See Figure 3.42.



θ_{i_ℓ}
Ω_1
δ
$(\theta_{i_\ell} \vee \alpha_0 \vee \cdots \vee \alpha_{n-1})$
$(\delta \vee \alpha_0 \vee \cdots \vee \alpha_{n-1})$

Fig. 3.42. Schematic representation of the mapping Ω_1.

Solution. Let $\Omega_1((\mathtt{T}, M), \vec{\theta}, \delta, \vec{\alpha}, m, 0) = (\mathtt{T}, M)$ and let

$$\begin{aligned}&\Omega_1((\mathtt{T}, M), \vec{\theta}, \delta, \vec{\alpha}, m, n+1)\\ &\quad = \Omega_0(\Omega_1((\mathtt{T}, M), \vec{\theta}, \delta, \vec{\alpha'}, m, n), \vec{\theta'}, \delta', \alpha_n, m, m),\end{aligned}$$

where

$$\begin{aligned}\vec{\alpha'} &= (\alpha_0, \ldots, \alpha_{n-1}),\\ \vec{\theta'} &= ((\theta_0 \vee \alpha_0 \vee \cdots \vee \alpha_{n-1}), \ldots, (\theta_{m-1} \vee \alpha_0 \vee \cdots \vee \alpha_{n-1})),\\ \delta' &= (\delta \vee \alpha_0 \vee \cdots \vee \alpha_{n-1}).\end{aligned}$$

(73) Prove that

$$\frac{(\Gamma, \varphi)}{(\Gamma', \varphi)},$$

where Γ, Γ' are sets of formulas such that $\Gamma \subseteq \Gamma'$ and φ is a formula is a derived rule of the formal system $\mathcal{ND}$.

(74) The formal system $\mathcal{ND}'$ has the same set of objects and set of axioms as the formal system $\mathcal{ND}$ introduced in Definition 3.6.14. Its set of rules includes the following groups:

The rules for introducing connective symbols are as follows:

$\dfrac{(\Gamma, \varphi), (\Gamma, \psi)}{(\Gamma, (\varphi \wedge \psi))}$	$\wedge$- introduction
$\dfrac{(\Gamma, \varphi)}{(\Gamma, (\varphi \vee \psi))}$ $\quad$ $\dfrac{(\Gamma, \psi)}{(\Gamma, (\varphi \vee \psi))}$	$\vee$- introduction
$\dfrac{(\Gamma \cup \{\varphi\}, \psi)}{(\Gamma, (\varphi \to \psi))}$	$\to$- introduction
$\dfrac{(\Gamma \cup \{\varphi\}, \psi), (\Gamma \cup \{\psi\}, \varphi)}{(\Gamma, (\varphi \leftrightarrow \psi))}$	$\leftrightarrow$- introduction
$\dfrac{(\Gamma \cup \{\varphi\}, \psi), (\Gamma \cup \{\varphi\}, (\neg\psi))}{(\Gamma, (\neg\varphi))}$	$\neg$- introduction

The *rules for eliminating connective symbols* are as follows:

$\dfrac{(\Gamma, (\varphi \wedge \psi))}{(\Gamma, \varphi)}$ $\quad$ $\dfrac{(\Gamma, (\varphi \wedge \psi))}{(\Gamma, \psi)}$	$\wedge$- elimination
$\dfrac{(\Gamma, (\varphi \vee \psi)), (\Gamma \cup \{\varphi\}, \alpha), (\Gamma \cup \{\psi\}, \alpha)}{(\Gamma, \alpha)}$	$\vee$- elimination
$\dfrac{(\Gamma, \varphi), (\Gamma, (\varphi \to \psi))}{(\Gamma, \psi)}$	$\to$ - elimination
$\dfrac{(\Gamma, \varphi), (\Gamma, (\varphi \leftrightarrow \psi))}{(\Gamma, \psi)}$ $\quad$ $\dfrac{(\Gamma, \psi), (\Gamma, (\varphi \leftrightarrow \psi))}{(\Gamma, \varphi)}$	$\leftrightarrow$- elimination
$\dfrac{(\Gamma \cup \{(\neg\varphi)\}, \psi), (\Gamma \cup \{(\neg\varphi))\}, (\neg\psi)}{(\Gamma, \varphi)}$	$\neg$- elimination

(a) Prove that

$$\frac{(\Gamma, \varphi)}{(\Gamma', \varphi)},$$

where Γ, Γ' are sets of formulas such that $\Gamma \subseteq \Gamma'$ and φ is a formula is a derived rule of the formal system $\mathcal{ND}'$.

(b) Prove that the theorems of $\mathcal{ND}'$ and $\mathcal{ND}$ are the same.

Translations between Formal Systems

(75) Show the following effective versions of Theorems 3.2.11 and 3.2.12, and Corollary 3.2.13, respectively, for the formal system $\mathcal{HF}'$.

(a) Suppose that the pair (q, q') is an $(\mathcal{HF}', \Gamma)$-certificate of inconsistency. Show how one can construct a proof in $\mathcal{HF}'_\Gamma$ of φ for every formula φ.
(b) Show that if (q, q') is an $(\mathcal{HF}', \Gamma \cup \{(\neg\varphi)\})$-certificate of inconsistency, then it is possible to construct effectively a proof of φ in $\mathcal{HF}'_\Gamma$.
(c) Show that one can construct effectively an $(\mathcal{HF}', \Gamma)$-certificate of inconsistency starting from $(\mathcal{HF}', \Gamma \cup \{\varphi\})$- and $(\mathcal{HF}', \Gamma \cup \{(\neg\varphi)\})$-certificates of inconsistency.

(76) Show the following effectivized version of Part (e) of Exercise 5. Let Γ be a set of formulas and let $\varphi \in \Gamma$. Show that if one is given $(\mathcal{HF}', \Gamma \cup K)$-certificates of inconsistency for all constituents K of φ, one can construct effectively an $(\mathcal{HF}', \Gamma)$-certificate of inconsistency.
Solution. There are nine cases, depending on the form of φ. We will do four cases and leave the rest to the reader. First, suppose that $\varphi = (\neg(\neg\alpha))$ and we are given an $(\mathcal{HF}', \Gamma \cup \{\alpha\})$-certificate of inconsistency. Since $((\neg(\neg\alpha)) \in \Gamma$, $((\neg\alpha), ((\neg(\neg\alpha))))$ is an $(\mathcal{HF}', \Gamma \cup \{(\neg\alpha)\})$-certificate of inconsistency, so by Part (c) of Exercise 75 we can obtain an $(\mathcal{HF}', \Gamma)$-certificate of inconsistency.
Next, suppose that $\varphi = (\alpha \wedge \beta)$ and that (q, q') is an $(\mathcal{HF}', \Gamma \cup \{\alpha, \beta\})$-certificate of inconsistency. Let $r = ((\alpha \wedge \beta), ((\alpha \wedge \beta) \to \alpha), ((\alpha \wedge \beta) \to \beta))$. Then, (rq, rq') is an $(\mathcal{HF}', \Gamma)$-certificate of inconsistency.
Let now $\varphi = (\neg(\alpha \wedge \beta))$ and suppose that we have $(\mathcal{HF}', \Gamma \cup \{(\neg\alpha)\})$- and $(\mathcal{HF}', \Gamma \cup \{(\neg\beta)\})$-certificates of inconsistency. By Part (b) of Exercise 75, we can construct effectively a proof r in $\mathcal{HF}'_\Gamma$ of α. The sequence r_1 given by

$$(((\neg(\alpha \wedge \beta)) \to (\alpha \to (\neg\beta))), (\neg(\alpha \wedge \beta)), (\alpha \to (\neg\beta)))\ r\ (\neg\beta)$$

is a proof in $\mathcal{HF}'_\Gamma$ of $(\neg\beta)$. Then, if (q, q') is an $(\mathcal{HF}', \Gamma \cup \{(\neg\beta)\})$-certificate of inconsistency, $(r_1 q, r_1 q')$ is an $(\mathcal{HF}', \Gamma)$-certificate of inconsistency.
Finally, suppose that $\varphi = (\alpha \vee \beta)$ and that we are given $(\mathcal{HF}', \Gamma \cup \{\alpha\})$- and $(\mathcal{HF}', \Gamma \cup \{\beta\})$-certificates of inconsistency.

Let r be the sequence:

$$(((\alpha \vee \beta) \rightarrow ((\neg\alpha) \rightarrow \beta)), (\alpha \vee \beta), ((\neg\alpha) \rightarrow \beta), (\neg\alpha), \beta).$$

If (q, q') is an $(\mathcal{HF}', \Gamma \cup \{\beta\})$-certificate¡ of inconsistency, then (rq, rq') is an $(\mathcal{HF}', \Gamma \cup \{(\neg\alpha)\})$-certificate of inconsistency. By Part (c) of Exercise 75, we can construct an $(\mathcal{HF}', \Gamma)$-certificate of inconsistency.

(77) Let Γ be a set of formulas and let φ be a formula such that $\Gamma \models \varphi$. Then, $\Gamma \cup \{(\neg\varphi)\}$ is unsatisfiable, so there is a strongly closed $\Gamma \cup \{(\neg\varphi)\}$-tableau $\mathtt{T}$. Show how to construct effectively a proof in $\mathcal{HF}'_\Gamma$ of φ using $\mathtt{T}$.
Solution. Since $\mathtt{T}$ is strongly closed, we have $(\mathcal{HF}', \mathtt{T}(q))$-certificates of inconsistency for each of its leaves q. If $\mathtt{T}$ is conservative, then using Supplement 76 repeatedly, we can construct an $(\mathcal{HF}', \mathtt{T}(\lambda))$-certificate of inconsistency, where $\mathtt{T}(\lambda) = \Gamma \cup \{(\neg\varphi)\}$. By applying Part (b) of Supplement 75, we can construct effectively a proof of φ in $\mathcal{HF}'_\Gamma$. The case when $\mathtt{T}$ is not conservative is left to the reader.

(78) Show that there is an effective, syntactic algorithm that transforms an $\mathcal{HG}$-deduction tree $\mathtt{T}$ into a natural deduction tree $\widehat{\mathcal{T}} = (\widehat{\mathtt{T}}, \widehat{M})$ in a way such that for all Γ, if $\mathtt{T}$ is an $\mathcal{HG}_\Gamma$-proof tree for φ, then $\mathcal{UNC}(\widehat{\mathtt{T}}, \widehat{M}) \subseteq \Gamma$ and $\widehat{\mathtt{T}}(\lambda) = \varphi$.
Solution. First, suppose that $\mathtt{T}$ is a one-node tree. If $\mathtt{T}(\lambda) = (\varphi \vee (\neg\varphi))$, for some formula φ, then we output the natural deduction tree shown in Figure 3.17. Otherwise, we output $(\mathtt{T}, \emptyset)$.
Now suppose that $|\mathtt{T}| > 1$. How the algorithm proceeds depends on which rule was used to produce the label of the root. We consider a few representative cases. Suppose that the rule used to produce the label of the root was $\mathsf{R}_{\leftrightarrow p}$. If the specific instance of $\mathsf{R}_{\leftrightarrow p}$ used at the root was

$$\frac{\begin{array}{c}((\neg\varphi) \vee \psi \vee \alpha_0 \vee \cdots \vee \alpha_{n-1})\\ (\varphi \vee (\neg\psi) \vee \alpha_0 \vee \cdots \vee \alpha_{n-1})\end{array}}{((\varphi \leftrightarrow \psi) \vee \alpha_0 \vee \cdots \vee \alpha_{n-1})}$$

let $\mathcal{T}_0 = (\mathtt{T}_0, M_0), \mathcal{T}_1 = (\mathtt{T}_1, M_1)$ be the natural deduction trees that are obtained by applying the algorithm recursively to the

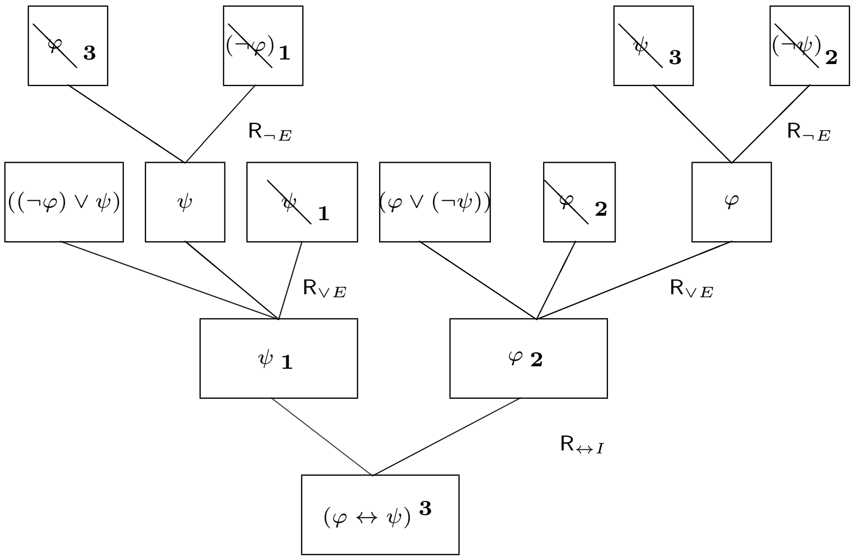


Fig. 3.43. Natural deduction tree showing $\{((\neg\varphi)\vee\psi),(\varphi\vee(\neg\psi))\} \overset{\bullet}{\vdash}_{\text{nd}} (\varphi \leftrightarrow \psi)$.

subtrees $\mathtt{T}_{[0]}, \mathtt{T}_{[1]}$ of the $\mathcal{HG}$-deduction tree $\mathtt{T}$. Then, we have

$$\mathtt{T}_0(\lambda) = ((\neg\varphi) \vee \psi \vee \alpha_0 \vee \cdots \vee \alpha_{n-1}),$$
$$\mathtt{T}_1(\lambda) = (\varphi \vee (\neg\psi) \vee \alpha_0 \vee \cdots \vee \alpha_{n-1}),$$

and if $\mathtt{T}$ is an $\mathcal{HG}_\Gamma$-proof tree, then $\mathcal{UNC}(\mathcal{T}_0), \mathcal{UNC}(\mathcal{T}_1) \subseteq \Gamma$. Let $\mathcal{T}'$ be the natural deduction tree shown in Figure 3.43 and let $\mathcal{T}'' = \Omega_1(\mathcal{T}', (((\neg\varphi) \vee \psi), (\varphi \vee (\neg\psi))), (\varphi \leftrightarrow \psi), (\alpha_0, \ldots, \alpha_{n-1}), 2, n)$, where Ω_1 is the mapping introduced in Supplement 72. If $\mathcal{T}'' = (\mathtt{T}'', M'')$, then $\mathtt{T}''(\lambda) = ((\varphi \leftrightarrow \psi) \vee \alpha_0 \vee \cdots \vee \alpha_{n-1})$ and $\mathcal{UNC}(\mathcal{T}'') \subseteq \{((\neg\varphi) \vee \psi \vee \alpha_0 \vee \cdots \vee \alpha_{n-1}), (\varphi \vee (\neg\psi) \vee \alpha_0 \vee \cdots \vee \alpha_{n-1})\}$. Let r be an unmarked leaf of $\mathcal{T}''$ such that $\mathtt{T}''(r) = ((\neg\varphi) \vee \psi \vee \alpha_0 \vee \cdots \vee \alpha_{n-1})$ or $\mathtt{T}''(r) = (\varphi \vee (\neg\psi) \vee \alpha_0 \vee \cdots \vee \alpha_{n-1})$. By Supplement 68, we can find a marked lot $\mathcal{T}_r$ that is either at least as marked as $\mathcal{T}_0$ or at least as marked as $\mathcal{T}_1$ such that $\mathcal{T}''[r \to \mathcal{T}_r]$ is a natural deduction tree. By repeating this process for all the unmarked leaves of $\mathcal{T}''$, we obtain the desired natural deduction tree $\widehat{\mathcal{T}}$. The arguments for the remaining first eight rules of $\mathcal{HG}$ are similar to the one presented above. The main difference consists of the natural deduction tree $\mathcal{T}'$. For example, the argument

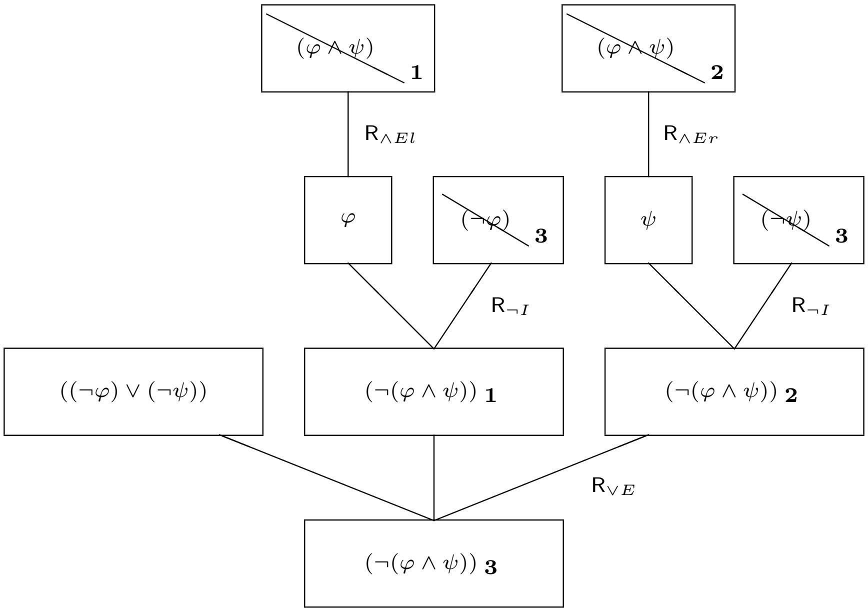

$\cancel{(\varphi \wedge \psi)}$ **1**
$\cancel{(\varphi \wedge \psi)}$ **2**
$\mathsf{R}_{\wedge El}$
$\mathsf{R}_{\wedge Er}$
φ
$\cancel{(\neg\varphi)}$ **3**
ψ
$\cancel{(\neg\psi)}$ **3**
$\mathsf{R}_{\neg I}$
$\mathsf{R}_{\neg I}$
$((\neg\varphi) \vee (\neg\psi))$
$(\neg(\varphi \wedge \psi))$ **1**
$(\neg(\varphi \wedge \psi))$ **2**
$\mathsf{R}_{\vee E}$
$(\neg(\varphi \wedge \psi))$ **3**

Fig. 3.44. Natural deduction tree showing $\{((\neg\varphi) \vee (\neg\psi))\} \overset{\bullet}{\vdash}_{\text{nd}} (\neg(\varphi \wedge \psi))$.

for $\mathsf{R}_{\wedge,n}$ makes use of the natural deduction tree shown in Figure 3.44. We leave providing the necessary details to the reader.

Suppose now that the root of the tree $\mathtt{T}$ is produced by using $\mathsf{R}_{mp'}$ and let $\mathcal{T}_0 = (\mathtt{T}_0, M_0), \mathcal{T}_1 = (\mathtt{T}_1, M_1)$ be the natural deduction trees that are obtained by applying the algorithm recursively to the subtrees $\mathtt{T}_{[0]}, \mathtt{T}_{[1]}$ of the $\mathcal{HG}$-deduction tree $\mathtt{T}$. Then, we have $\mathtt{T}_0(\lambda) = (\varphi \vee \alpha_0 \vee \cdots \alpha_{n-1})$ and $\mathtt{T}_1(\lambda) = ((\neg\varphi) \vee \alpha_0 \vee \cdots \vee \alpha_{n-1})$ and if $\mathtt{T}$ is an $\mathcal{HG}_\Gamma$-proof tree, then $\mathcal{UNC}(\mathcal{T}_0), \mathcal{UNC}(\mathcal{T}_1) \subseteq \Gamma$. Let $\mathcal{T}'$ be the natural deduction tree shown in Figure 3.45 and let

$$\mathtt{T}'' = \Omega_1(\mathtt{T}', ((\varphi \vee \alpha_0), ((\neg\varphi) \vee \alpha_0)), (\alpha_1, \ldots, \alpha_{n-1}), 2, n-1).$$

If $\mathtt{T}'' = (T'', M'')$, then $\mathtt{T}''(\lambda) = (\alpha_0 \vee \cdots \alpha_{n-1})$ and $\mathcal{UNC}(\mathtt{T}'') \subseteq \{(\varphi \vee \alpha_0 \vee \cdots \vee \alpha_{n-1}), ((\neg\varphi) \vee \alpha_0 \vee \cdots \vee \alpha_{n-1})\}$.
Let r be an unmarked leaf of $\mathcal{T}''$ such that $\mathtt{T}''(r) = (\varphi \vee \alpha_0 \vee \cdots \vee \alpha_{n-1})$ or $\mathtt{T}''(r) = ((\neg\varphi) \vee \alpha_0 \vee \cdots \vee \alpha_{n-1})$. By Supplement 68, we can find a marked lot $\mathcal{T}_r$ that is either at least as marked as

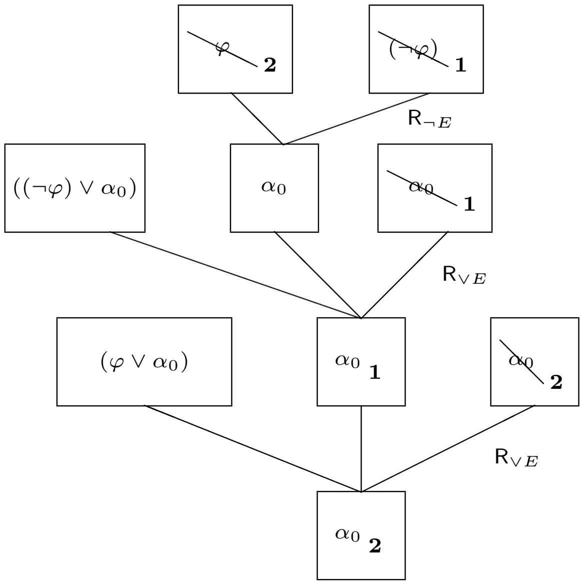


Fig. 3.45. Natural deduction tree showing $\{(\varphi \vee \alpha_0), ((\neg\varphi) \vee \alpha_0)\} \overset{\bullet}{\vdash}_{\mathrm{nd}} \alpha_0$.

$\mathcal{T}_0$ or at least as marked as $\mathcal{T}_1$ such that $\mathcal{T}''[r \to \mathcal{T}_r]$ is a natural deduction tree. By repeating this process for all the unmarked leaves of $\mathcal{T}''$, we obtain the desired natural deduction tree $\widehat{\mathcal{T}}$. Suppose now that the instance

$$\frac{(\alpha_0 \vee \cdots \vee \alpha_i \vee \alpha_{i+1} \vee \cdots \vee \alpha_{n-1})}{(\alpha_0 \vee \cdots \vee \alpha_{i+1} \vee \alpha_i \vee \cdots \vee \alpha_{n-1})}$$

of $\mathsf{R}_{\mathrm{intch}}$ was used to produce the label of the root, where $i > 0$. Let $\mathcal{T}_0$ be the natural deduction tree that results from applying the algorithm to $\mathtt{T}_{[0]}$ and let $\mathcal{T}' = \mathcal{T}_{(\alpha_0 \vee \cdots \vee \alpha_{i-1}), \alpha_i, \alpha_{i+1}}$, where $\mathcal{T}_{\beta,\gamma,\delta}$ is the natural deduction tree introduced in Supplement 67. The natural deduction tree $\mathcal{T}'' = (\mathtt{T}'', M'')$ is given by

$$\begin{aligned}\mathcal{T}'' = \Omega_1(\mathcal{T}', ((\alpha_0 \vee \cdots \vee \alpha_i \vee \alpha_{i+1})), (\alpha_0 \vee \cdots \vee \alpha_{i+1} \vee \alpha_i),\\ (\alpha_{i+2}, \ldots, \alpha_{n-1}), 1, n - i - 2).\end{aligned}$$

Then, $\mathcal{UNC}(\mathcal{T}'') \subseteq \{(\alpha_0 \vee \cdots \vee \alpha_i \vee \alpha_{i+1} \vee \cdots \vee \alpha_{n-1})\}$ and $\mathtt{T}''(\lambda) = (\alpha_0 \vee \cdots \vee \alpha_{i+1} \vee \alpha_i \vee \cdots \vee \alpha_{n-1})$. For each unmarked leaf r of $\mathcal{T}''$, there is a natural deduction tree $\mathcal{T}_r$ that is at least as marked as $\mathcal{T}_0$ such that $\mathcal{T}''[r \to \mathcal{T}_r]$ is a natural deduction tree. The natural deduction tree $\widehat{\mathcal{T}}$ is obtained by repeated application of this process. We leave to the reader consideration of the case when $i = 0$ which requires redefining $\mathcal{T}'$ as $\mathcal{T}_{\alpha_0,\alpha_1}$, where $\mathcal{T}_{\gamma,\delta}$ was introduced in Supplement 67.

Suppose now that the instance

$$\frac{(\alpha_0 \vee \cdots \vee \alpha_{n-1} \vee \alpha_{n-1})}{(\alpha_0 \vee \cdots \vee \alpha_{n-1})}$$

of R_{cont} was used to produce the label of the root, where $n > 1$. Let $\mathcal{T}_0$ be the natural deduction tree that results from applying the algorithm to $\mathtt{T}_{[0]}$ and let $\mathcal{T}'$ be the natural deduction tree given in Figure 3.46(b). Then, we can define $\widehat{\mathcal{T}} = \mathcal{T}'[0 \to \mathcal{T}_0]$. The case when $n = 1$ is similar, but uses the natural deduction tree $\mathcal{T}'$ given in Figure 3.46(a). We leave to the reader the final case when rule R_{thin} is used to produce the root.

(79) Use Supplements 64 and 78 to give an alternative proof of the completeness of natural deduction (Theorem 3.6.12).

Resolution

(80) Give an example of an infinite set Γ of formulas in conjunctive normal form such that $\mathcal{C}_\Gamma$ is finite.

(81) If $(C_0, \ldots, C_n)$ is a resolution proof over $\mathcal{C}$, show that $\mathrm{LIT}(C_n) \subseteq \mathrm{LIT}(\mathcal{C})$.
Hint. Use course-of-values induction on n.

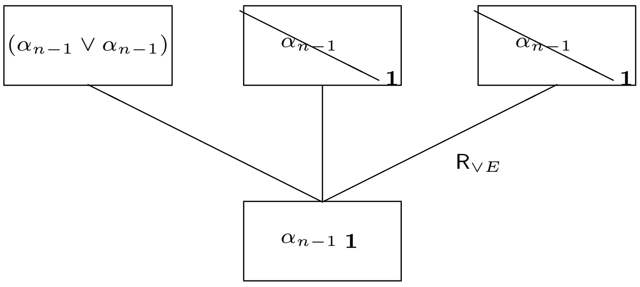
(82) Let $\mathcal{C}$ be an unsatisfiable set of clauses that does not contain $\Box$.

(a) Prove that there are two clauses $C, D \in \mathcal{C}$ and a literal ℓ such that $\ell \in C$ and $\bar{\ell} \in D$.

(b) Prove that there exists no truth assignment v which falsifies all clauses of $\mathcal{C}$.

(83) If C, D are two clauses that can be resolved with respect to two distinct literals ℓ, ℓ', show that their resolvent with respect to any literal is a tautologous clause.

(84) Let $\mathcal{C}$ be a set of clauses such that no clause in $\mathcal{C}$ contains the literal $\bar{\ell}$. Prove that $\mathcal{C}^{\ell} \subseteq \mathcal{C}^{\bar{\ell}}$; furthermore, $\mathcal{C}$ is satisfiable if and only if $\mathcal{C}^{\ell}$ is satisfiable.



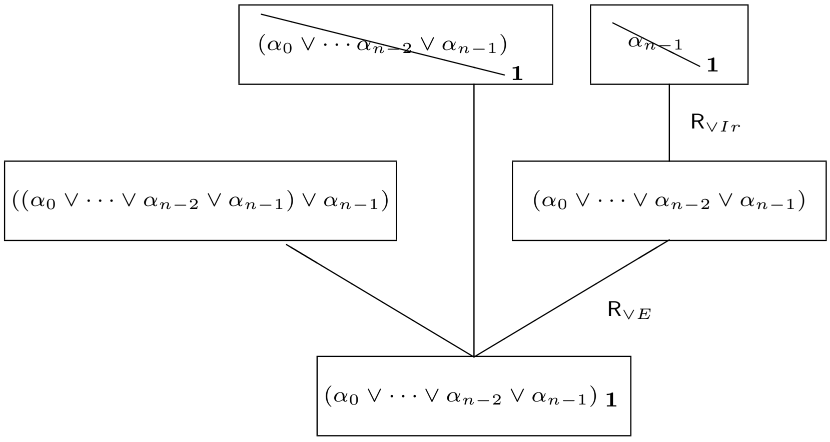

$(\alpha_0 \vee \cdots \alpha_{n-2} \vee \alpha_{n-1})$ **1**
α_{n-1} **1**
$\mathsf{R}_{\vee Ir}$
$((\alpha_0 \vee \cdots \vee \alpha_{n-2} \vee \alpha_{n-1}) \vee \alpha_{n-1})$
$(\alpha_0 \vee \cdots \vee \alpha_{n-2} \vee \alpha_{n-1})$
$\mathsf{R}_{\vee E}$
$(\alpha_0 \vee \cdots \vee \alpha_{n-2} \vee \alpha_{n-1})$ **1**
(b)

Fig. 3.46. Natural deduction trees showing (a) $\{(\alpha_{n-1} \vee \alpha_{n-1})\} \overset{\bullet}{\vdash}_{\mathrm{nd}} \alpha_{n-1}$ and (b) $\{((\alpha_0 \vee \cdots \vee \alpha_{n-2} \vee \alpha_{n-1}) \vee \alpha_{n-1})\} \overset{\bullet}{\vdash}_{\mathrm{nd}} (\alpha_0 \vee \cdots \vee \alpha_{n-2} \vee \alpha_{n-1})$.

Solution. If no clause of $\mathcal{C}$ contains $\bar{\ell}$, we have

$$\begin{aligned}\mathcal{C}^{\ell} &= \{C - \{\bar{\ell}\} | C \in \mathcal{C} \text{ and } \ell \not\in C\}\\ &= \{C \in \mathcal{C} | \ell \not\in C\},\\ \mathcal{C}^{\bar{\ell}} &= \{C - \{\ell\} | C \in \mathcal{C}\}.\end{aligned}$$

Therefore, we have $\mathcal{C}^{\ell} \subseteq \mathcal{C}^{\bar{\ell}}$. Suppose that $\mathcal{C}$ is satisfiable. Since $\mathcal{C}^{\ell} \subseteq \mathcal{C}$ it follows immediately that $\mathcal{C}^{\ell}$ is satisfiable. The converse follows from Theorem 3.8.26.

(85) Prove that if $(C_0, \ldots, C_{n-1})$ is a resolution proof over a set of clauses $\mathcal{C}$ and ℓ is a literal such that $\ell \not\in \bigcup_{i=0}^{n-1} C_i$, then

$(C_0 \cup \{\ell\}, \ldots, C_{n-1} \cup \{\ell\})$ is also a resolution proof over $\mathcal{C}_\ell = \{C \cup \{\ell\} \mid C \in \mathcal{C}\}$.

(86) If the unit clause $\{\ell\}$ is among the clauses of the set of clauses $\mathcal{C}$, then prove that $\mathcal{C}$ is satisfiable if and only if $\mathcal{C}^\ell$ is satisfiable.
Solution. If $\{\ell\} \in \mathcal{C}$, then $\Box \in \mathcal{C}^{\bar{\ell}}$, so $\mathcal{C}^{\bar{\ell}}$ is unsatisfiable. Therefore, if $\mathcal{C}$ is satisfiable, then $\mathcal{C}^\ell$ must be satisfiable by Theorem 3.8.26. The same theorem implies that if $\mathcal{C}^\ell$ is satisfiable, then so is $\mathcal{C}$.

We now present an alternative proof of the completeness of resolution (Theorem 3.8.32) based on the notion of semantic tree. This proof is simpler than the one in the main text but does not lend itself to generalizations.

For $q \in \{0,1\}^*$, define $v_q : \{p_0, \ldots, p_{|q|-1}\} \longrightarrow \{\mathbf{T}, \mathbf{F}\}$ by

$$v_q(p_i) = \begin{cases} \mathbf{T} & \text{if } q(i) = 1 \\ \mathbf{F} & \text{if } q(i) = 0 \end{cases}$$

for $0 \leq i \leq |q| - 1$.

Define the *semantic lot* $\mathtt{T}^{\text{sem}}$ by $\text{Dom}(\mathtt{T}^{\text{sem}}) = \{0,1\}^*$ and $\mathtt{T}^{\text{sem}}(q) = v_q$ for $q \in \{0,1\}^*$. If q is a prefix of r, then $v_q \subseteq v_r$, so if $\mathtt{B}$ is a branch of $\mathtt{T}^{\text{sem}}$, then define $v_{\mathtt{B}}$ as $\bigcup_{q \in \mathtt{B}} v_q$.

For a set of clauses $\mathcal{C}$, a node q of $\mathtt{T}^{\text{sem}}$ is a *failure node for* $\mathcal{C}$ if for some clause $C \in \mathcal{C}$, $v_q(C) = \mathbf{F}$ and we define $\mathtt{T}^*_{\mathcal{C}}$ as the sublot of $\mathtt{T}^{\text{sem}}$ having as domain $\{q \in \{0,1\}^* \mid$ no proper prefix of q is a failure node of $\mathcal{C}\}$.

(87) Verify that $\text{Dom}(\mathtt{T}^*_{\mathcal{C}})$ is a tree domain.

(88) Prove that v is a truth assignment if and only if $v = v_{\mathtt{B}}$ for some branch $\mathtt{B}$ of the semantic propositional lot $\mathtt{T}^{\text{sem}}$.

(89) Prove that a set of clauses $\mathcal{C}$ is satisfiable if and only if there is a branch $\mathtt{B}$ of $\mathtt{T}^{\text{sem}}$ that does not contain a failure node for $\mathcal{C}$.

(90) Let $\mathcal{C}$ be a set of clauses. Prove that every leaf of $\mathtt{T}^*_{\mathcal{C}}$ is a failure node for $\mathcal{C}$.

(91) Let $\mathcal{C}, \mathcal{C}'$ be two sets of clauses such that $\mathcal{C} \subseteq \mathcal{C}'$. Prove that $\mathtt{T}^*_{\mathcal{C}'}$ is a sublot of $\mathtt{T}^*_{\mathcal{C}}$.

(92) Let $\mathcal{C}$ be a set of clauses and $q \in \{0,1\}^*$. Prove that either both of $q0$ and $q1$ are in $\text{Dom}(\mathtt{T}^*_{\mathcal{C}})$ or neither are. Conclude that every interior node of $\mathtt{T}^*_{\mathcal{C}}$ has two immediate descendants.

(93) Let $\mathcal{C}$ be a set of clauses and let $q \in \{0,1\}^*$ with $|q| = k$ be a node which is not a failure node for $\mathcal{C}$. Prove that if $q0$ is a failure node for $\mathcal{C}$, then $\mathcal{C}$ contains a clause C such that the propositional variable $p_k \in \mathcal{C}$ and $v_q(C - \{p_k\}) = \mathbf{F}$. Similarly, if $q1$ is a failure node for $\mathcal{C}$, then $\mathcal{C}$ contains a clause C such that $(\neg p_k) \in \mathcal{C}$ and $v_q(C - \{(\neg p_k)\}) = \mathbf{F}$.

(94) Let $\mathcal{C}$ be an unsatisfiable set of clauses. Prove that $\mathtt{T}^*_{\mathcal{C}}$ does not contain an infinite branch. Conclude by König's Lemma that $\mathtt{T}^*_{\mathcal{C}}$ is finite.

In the following supplement, we reprove the completeness theorem for propositional resolution using the semantic tree. This approach will be useful in Chapter 5 in the study of a technique known as paramodulation.

(95) Prove that if $\mathcal{C}$ is an unsatisfiable set of clauses, then $\square \in \text{Res}^*(\mathcal{C})$.

Solution. Note that by Exercise 94, $\mathtt{T}^*_{\mathcal{C}}$ is finite, so we can prove the result by induction on $|\mathtt{T}^*_{\mathcal{C}}|$.

For the basis step, $|\mathtt{T}^*_{\mathcal{C}}| = 1$, we must have $\square \in \mathcal{C}$, so the result is immediate.

For the inductive step, suppose that $|\mathtt{T}^*_{\mathcal{C}}| > 1$ and the result is true for all unsatisfiable set of clauses $\mathcal{C}'$ with $|\mathtt{T}^*_{\mathcal{C}'}| < |\mathtt{T}^*_{\mathcal{C}}|$. Since $|\mathtt{T}^*_{\mathcal{C}}| > 1$, there is node q of $\mathtt{T}^*_{\mathcal{C}}$ with $\mathtt{depth}(\mathtt{T}^*_{\mathcal{C}})(q) = 1$, say $|q| = k$. By Exercise 92, q has two children in $\mathtt{T}^*_{\mathcal{C}}$ which are both failure nodes for $\mathcal{C}$ by Exercise 90. By Exercise 93, $\mathcal{C}$ contains a clause C_0 such that $p_k \in C_0$ and $v_q(C_0 - \{p_k\}) = \mathbf{F}$ and a clause C_1 such that $(\neg p_k) \in C_1$ and $v_q(C_1 - \{(\neg p_k)\}) = \mathbf{F}$. Then, $R = (C_0 - \{p_k\}) \cup (C_1 - \{(\neg p_k)\})$ is a resolvent of C_0 and C_1 and $v_q(R) = \mathbf{F}$. Thus, q is a failure node of $\mathcal{C} \cup \{R\}$. Hence, $q0, q1 \notin \text{Dom}(\mathtt{T}^*_{\mathcal{C}\cup\{R\}})$, so by Exercise 91, $|\mathtt{T}^*_{\mathcal{C}\cup\{R\}}| < |\mathtt{T}^*_{\mathcal{C}}|$. By inductive hypothesis, we have $\square \in \text{Res}^*(\mathcal{C} \cup \{R\}) = \text{Res}^*(\mathcal{C})$.

Variations of Resolution

(96) Show that Corollary 3.9.3 is a consequence of Corollary 3.9.22.

(97) Prove that if $\mathcal{D}$ is a set-of-support for a finite unsatisfiable set of clauses $\mathcal{C}$ and $\mathcal{D} \subseteq \mathcal{E} \subseteq \mathcal{C}$, then $\mathcal{E}$ is also a set of support for $\mathcal{C}$.

Prove also that, if $\mathcal{C}$ is a finite set of clauses, then every set-of-support $\mathcal{D}$ contains a minimal set-of-support $\mathcal{D}_0$.

Cutting Planes

(98) Let $\mathcal{CP}'$ be the formal system obtained from $\mathcal{CP}$ by removing the multiplication rule. Prove that if $A > 0$ and

$$\{z_0 \geq A\} \vdash_{\mathcal{CP}'} \sum a_i z_i \geq B,$$

then $B > 0$. Conclude that $\{z_0 \geq 1\} \not\vdash_{\mathcal{CP}'} \mathsf{0} \geq 0$, although $\{z_0 \geq 1\} \vdash_{\mathcal{CP}} \mathsf{0} \geq 0$.

(99) Give an example of two nontautologous clauses C_0, C_1 and a nontautologous resolvent R of C_0, C_1 such that $\{I_{C_0}, I_{C_1}\} \not\models I_R$. Conclude from the soundness of $\mathcal{CP}$ that $\{I_{C_0}\} \cup \{I_{C_1}\} \not\vdash_{\mathcal{CP}} I_R$ and therefore, the statement obtained from Lemma 3.10.22 by replacing $\mathcal{I}_{\{C_i\}}$ with $\{I_{C_i}\}$ for $i = 0, 1$ would be false.
Hint. One possibility is $C_0 = \{p_0, \neg p_1, \neg p_2\}$ and $C_1 = \{\neg p_0, \neg p_1, p_3\}$.

3.12 Bibliographical Comments

Tableaux were introduced by Beth [2] and applied by Raymond Smullyan [36].

The sequent system was introduced by Gentzen in [15]. The natural deduction system was introduced in [16]. Both papers are included in [38].

The cut rule and cut elimination were introduced by Gentzen in [16].

Statman [37] showed that proofs without the cut rule may have to be super-polynomially larger than proofs with this rule. Our proof, contained in Supplements 40 and 47 is a reworking of the proof in Buss [5] in terms of tableaux.

Propositional resolution was introduced by Davis and Putnam in [10].

Cutting planes originate in operations research. Their use in refutation propositional logic is based on the idea that unsatisfiable propositional logic formulas in conjunctive normal form are recognized by showing the non-existence of Boolean solutions to associated families of linear inequalities. This approach was proposed in [24] and further developed in [6, 7].

Bibliography

[1] Beth, E. W. (1953). Some consequences of the theorem of Lövenheim-Skolem-Gödel-Malcev, *Indag. Math.* **15**, pp. 66–71.

[2] Beth, E. W. (1955). Semantic entailment and formal derivability, *Mededelingen van de Koninklijke Nederlandse Akademie van Wetenschappen, Afdeling Letterkunde* **18**, pp. 309–342.

[3] Blum, N. (1984). A Boolean function requiring $3n$ network size, *Theoretical Computer Science* **28**, pp. 337–345.

[4] Boppana, R. B. and Sipser, M. (1994). The complexity of finite functions, in J. van Leeuwen (ed.), *Handbook of Theoretical Computer Science – Volume A: Algorithms and Complexity* (Elsevier, Amsterdam), pp. 759–804.

[5] Buss, S. (1988). Weak formal systems and connections to computational complexity, Lecture Notes for a Topics Course, University of California, Berkeley.

[6] Buss, S. R. and Clote, P. (1996). Cutting planes, connectivity, and threshold logic, *Archive for Mathematical Logic* **35**, 1, pp. 33–62.

[7] Clote, P. (1995). Cutting plane and Frege proofs, *Information and Computation* **121**, 1, pp. 103–122.

[8] Cohn, P. M. (1981). *Universal Algebra* (D. Reidel, Dordrecht).

[9] Craig, W. (1957). Three uses of the Herbrand-Gentzen theorem in relating model theory and proof theory, *Journal of Symbolic Logic* **22**, pp. 269–285.

[10] Davis, M. and Putnam, H. (1960). A computing procedure for quantification theory, *Journal of the ACM* **7**, pp. 201–215.

[11] Dowling, W. F. and Gallier, J. H. (1984). Linear-time algorithms for testing the satisfiability of propositional horn formulae, *Journal of Logic Programming* **1**, pp. 267–284.

[12] McCluskey, Jr. E. J. (1956). Minimization of Boolean functions, *Bell System Technical Journal* **35**, pp. 1417–1444.

[13] Fejer, P. A. and Simovici, D. A. (1991). *Mathematical Foundations of Computer Science. Volume I: Sets, Relations, and Induction* (Springer, New York).

[14] Frege, F. L. G. (1884). *Die Grundlagen der Arithmetik: eine logisch mathematische Untersuchung über den Begriff der Zahl* (W. Koebner, Breslau).

[15] Gentzen, G. (1932). Über die existenz unabhängiger axiomensysteme zu unedlichen satzsystemen, *Mathematische Annalen* **107**, pp. 329–350.

[16] Gentzen, G. (1935). Untersuchungen über das logische schliessen, *Mathematische Zeitschrift* **39**, pp. 176–210, 405–431.

[17] Gorn, S. (1965). Explicit definitions and linguistic dominoes, in J. Hart and S. Takasu (eds.), *Systems and Computer Science* (University of Toronto Press, Toronto), pp. 77–115.

[18] Gray, F. (1953). Pulse code communication, US Patent 2,632,058.

[19] Hilbert, D. (1923). Die logischen Grundlagen der Mathematik, *Mathematische Annalen* **88**, pp. 151–165.

[20] Hilbert, D. (1934). *Grundlagen der Mathematik*, Vol. 1 (Springer, Berlin).

[21] Hilbert, D. (1939). *Grundlagen der Mathematik*, Vol. 2 (Springer, Berlin).

[22] Hilbert, D. and Ackermann, W. (1928). *Grundzüge der theoretischen Logik* (Springer, Berlin).

[23] Hintikka, J. (1955). Form and content in quantification theory, *Acta Philosophica Fennica* **8**, pp. 7–55.

[24] Hooker, J. N. (1988). Generalized resolution and cutting planes, *Annals of Operations Research* **12**, pp. 217–239.

[25] Horn, A. (1951). On sentences which are true of direct unions of algebras, *Journal of Symbolic Logic* **16**, pp. 14–21.

[26] Karnaugh, M. (1953). The map method for synthesis of combinational logic circuits, *Communications and Electronics*, pp. 593–599.

[27] Knuth, D. E. (1973). *The Art of Computer Programming — Fundamental Algorithms*, Vol. 1 (Addison–Wesley, Reading, MA).

[28] Lupanov, O. B. (1958). A method of circuit synthesis, *Izv. VUZ Radiofiz.* **1**, pp. 120–140, (In Russian).

[29] de Bruijn, N. G. and Erdös, P. (1951). A colour problem for infinite graphs and a problem in the theory of relations, *Indag. Math.* **54**, pp. 371–373.

[30] Post, E. L. (1941). *The Two-Valued Iterative Systems of Mathematical Logic*, Annals of mathematics studies (Princeton University Press, Princeton).

[31] Quine, W. V. O. (1952). The problem of simplifying truth functions, *American Mathematical Monthly* **59**, pp. 521–531.

[32] Quine, W. V. O. (1955). A way to simplify truth functions, *American Mathematical Monthly* **62**, pp. 627–631.

[33] Riordan, J. and Shannon, C. (1942). The number of two-terminal series-parallel networks, *Journal of Mathematics and Physics* **21**, pp. 83–93.

[34] Shannon, C. (1949). The synthesis of two-terminal switching circuits, *Bell System Technical Journal* **28**, pp. 59–98.

[35] Smullyan, R. M. (1961). *Theory of Formal Systems* (Princeton University Press, Princeton).

[36] Smullyan, R. M. (1995). *First-Order Logic* (Dover Publications, New York).

[37] Statman, G. R. (1978). Bounds for proof-search and speed-up in the predicate calculus, *Annals of Mathematical Logic* **15**, pp. 225–287.

[38] Szabo, M. E. (1969). The collected papers of Gerhard Gentzen, (North-Holland, Amsterdam).

[39] Vollmer, H. (1999). *Introduction to Circuit Complexity – A Uniform Approach* (Springer, Berlin).

[40] Wegener, I. (1987). *The Complexity of Boolean Functions* (John Wiley & Sons, Chichester).

List of Notations

$L_\theta(\mathcal{T})$ marked lot obtained from $\mathcal{T}$ by adding to the marked leaves all leaves labeled with θ, 3
$(\mathtt{T}_0, \ldots, \mathtt{T}_{n-1}; \theta)$ lot obtained by joining the roots of $\mathtt{T}_0, \ldots, \mathtt{T}_{n-1}$ with a new root labeled θ, 3
$\bar{a}$ alternative notation for $f_\neg(a)$, 4
Bool the set of truth values, 4
$\Box$ empty clause, 4
$\mathtt{C}(t)$ the set of constant symbols that occur in the term t, 4
$\mathtt{C}_{S,V}(t)$ the set of constant symbols of S that occur in the term t, 4
$\mathcal{C}, \mathcal{D}, \mathcal{E}$ sets of clauses, 4
$\mathcal{F}_G$ formal system obtained from $\mathcal{F}$ by adding G to the axioms of $\mathcal{F}$, 4
$\mathcal{F}_L$ principal filter generated by L, 4
$\mathcal{F}^{\mathrm{seq,cut}}$ the formal system obtained from $\mathcal{F}^{\mathrm{seq}}$ by adding the cut rule, 5
$\mathcal{F}^{\mathrm{seq},\infty,\mathrm{cut}}$ the formal system obtained from $\mathcal{F}^{\mathrm{seq},\infty}$ by adding the cut rule, 5
$\mathcal{F}^{\mathrm{seq,cons,cut}}$ the formal system obtained from $\mathcal{F}^{\mathrm{seq,cons}}$ by adding the cut rule, 5
$\mathcal{F}^{\mathrm{seq},\infty,\mathrm{cons,cut}}$ the formal system obtained from $\mathcal{F}^{\mathrm{seq},\infty,\mathrm{cons}}$ by adding the cut rule, 5
$\mathcal{I}_\mathcal{C}$ set of inequalities determined by the set of clauses $\mathcal{C}$, 5
$\mathrm{PREF}(q)$ the set of all prefixes of the sequence q, 5
$\mathcal{K}^\wp$ the circuit obtained from the circuit $\mathcal{K}$ by rearranging the inputs according to $\wp$, 7
$\mathcal{K}_{\varphi,m}$ the formula circuit corresponding to the formula φ, 19

$\mathcal{F}^{\text{tabl,cons,cut}}$ the tableau formal system with cut of sets of signed propositional formulas, 95

$\mathcal{F}^{\text{tabl,cut}}$ the formal system obtained from $\mathcal{F}^{\text{tabl,cons,cut}}$ by adding the thinning rule, 102

$\mathcal{F}^{\text{tabl},\infty}$ the tableau formal system of finite sets of propositional signed formulas, 102

$\mathcal{F}^{\text{tabl},\infty,\text{cons}}$ the tableau formal system of sets of propositional signed formulas, 102

$\mathcal{F}^{\text{tabl},\infty,\text{cons,cut}}$ the formal system obtained from $\mathcal{F}^{\text{tabl},\infty,\text{cons}}$ by adding the cut rule, 103

$\Gamma \models \varphi$ logical implication of a formula by a set of formulas, 104

$\Gamma \overset{\bullet}{\vdash}_{\text{nd}} \varphi$ φ is *nd*-derivable from Γ, 107

$\Gamma \overset{\bullet}{\vdash}_{\text{nd}'} \varphi$ φ is *nd'*-derivable from Γ, 107

$\mathcal{GDT}_{\mathcal{F}}$ the set of general deduction trees in the formal system $\mathcal{F}$, 107

q_v sequence over $\mathbf{Bool} \cup \{-\}$ representing the partial truth assignment v, 107

GTERM_S the set of ground terms of signature S, 107

$\mathcal{HF}$ Hilbert/Frege-style system for propositional logic, 110

$\text{IMPL}(\varphi)$ set of implicants of a formula φ, 110

$\vdash_{\mathcal{F}} \theta$ θ is a theorem of $\mathcal{F}$, 112

$\text{ISeq}(D)$ the set of infinite sequences over D, 112

$\kappa = (\Gamma, \Gamma')$ sequent, 112

$\kappa_0 \cup \kappa_1$ the union of the sequents κ_0, κ_1, 113

$\text{LIT}(\mu)$ the set of literals of a minterm μ, 120

$\text{MAXTRM}(S)$ the set of maxterms over S, 123

$\text{MINTRM}(S)$ the set of minterms over S, 129

$\models \varphi$ denotes that φ is a tautology, 129

$\text{MINPTA}^b(\varphi)$ the set of all minimal elements of $\text{PTA}^b(\varphi)$, 129

μ_S minterm determined by S, 129

μ_T projection of the minterm μ on T, 129

$\text{nd}(n)$ the number of nondegenerate n-ary truth functions, 129

NDT the set of natural deduction trees, 129

$\text{NT}(\mathcal{C})$ nontautologous clauses of a set of clauses $\mathcal{C}$, 129

$\text{NTRes}(\mathcal{C})$ $\mathcal{C}$ augmented by the nontautologous resolvents of clauses of $\mathcal{C}$, 129

ν arity function of a signature, 134

ν_S maxterm determined by S, 136

TA the set of truth assignments, 209
τ_S^f truth table defined by the truth function f and the set S, 213
$\tau_{\varphi,S}$ truth table of φ over S, 213
τ_φ truth table of formula $\varphi \in$ PLFORM, 214
$\text{TERM}_S(V)$ set of terms of signature S over the set of variables V, 217
$\mathtt{TF}$ the set of truth functions having at least one argument, 224
$\mathtt{TF}_*$ the set of truth functions, 227
$\mathtt{TF}_n$ the set of truth functions of n arguments, 237
$\Theta_\mathcal{K}(\mathtt{v})$ the function computed by the circuit $\mathcal{K}$ at the vertex $\mathtt{v}$, 237
$\mathtt{D}(b\varphi)$ collection of constituents of the signed formula $b\varphi \in \text{SPLFORM}_F$, 237
$\mathtt{d}(b\varphi)$ sequence of constituents of the signed formula $b\varphi \in \text{SPLFORM}_F$, 237
S signature, 238
$\mathtt{T}\dot{-}b\varphi$ tableau obtained from $\mathtt{T}$ by removing $b\varphi$ from nodes without $b\varphi$-introducing ancestors, 238
$\mathtt{T}(\mathtt{P})$ the set of signed formulas that occur in the path $\mathtt{P}$ of the tableau $\mathtt{T}$, 239
$\widehat{\widehat{F}}$ the extended clone generated by the set F of truth functions, 243
$\mathtt{V}_{S,V}(t)$ set of variables of V that occur in the term t, 244
$\vec{a}^{\vec{b}}$ for $a_0^{b_0} \wedge \cdots \wedge a_{n-1}{}^{b_{n-1}}$, 244
$\widehat{F}$ the clone generated by the set F of truth functions, 245
$A \equiv_m B$ m-equivalence of the sets A and B, 245
$A \leq_m B$ A is many-one reducible to B, 246
a^b alternative notation for $a \leftrightarrow b$, 249
C, D, E clauses, 258
$c_\mathbf{F}^m$, m-ary constant function equal to $\mathbf{F}$, 268
$c_\mathbf{T}^m$, m-ary constant function equal to $\mathbf{T}$, 302
D^n the set of sequences of length n over D, 303
$D^{\leq n}$ the set of sequences of length less than or equal to n over D, 308
$F(f)$ the set of all sequences $(b_0, \ldots, b_{n-1})$ in $\mathbf{Bool}^n$ such that $f(b_0, \ldots, b_{n-1}) = \mathbf{F}$, 308
$f(g_0, \ldots, g_{m-1})$ composition of $f, g_0, \ldots, g_{m-1}$, 308
$f = \Omega(g)$ f grows no slower than g, 308
$f = \Theta(g)$ f grows as fast as g, 309
$f = O(g)$ f grows no faster than g, 310

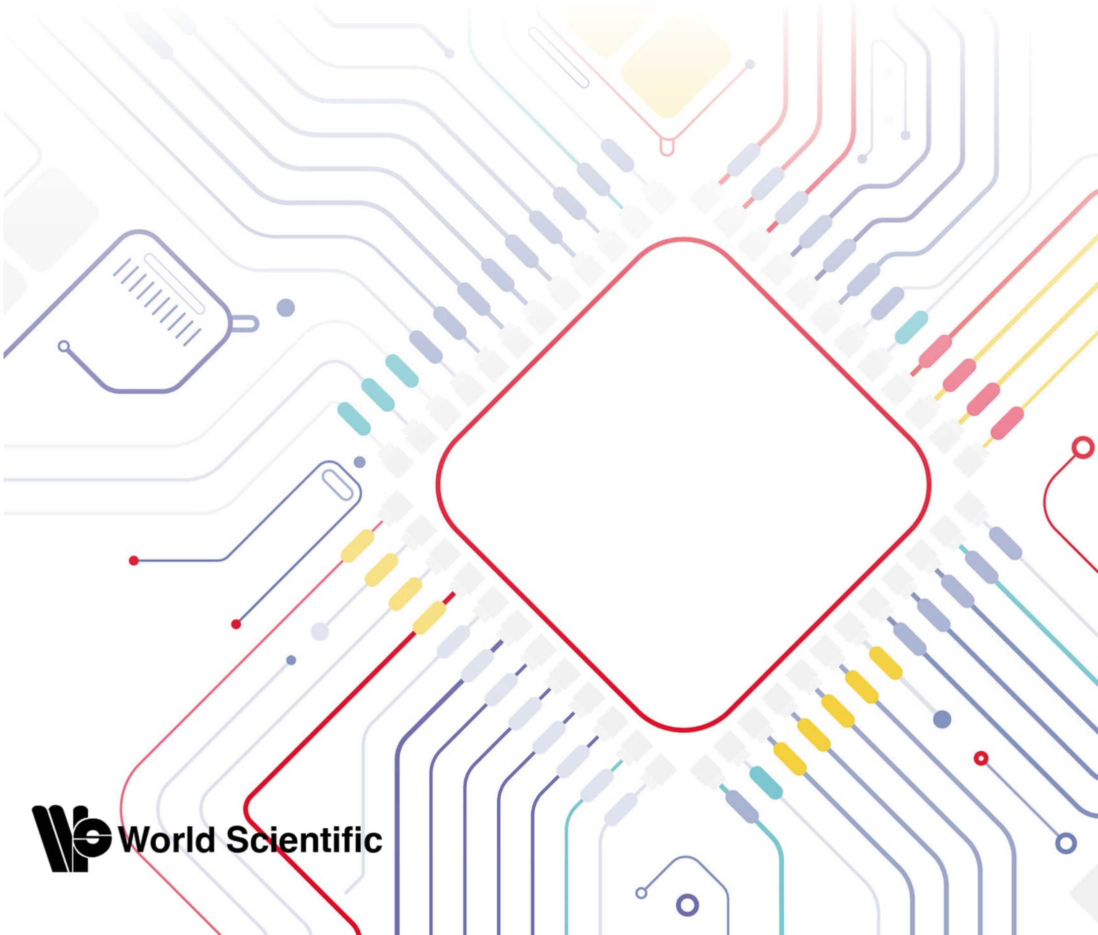
List of Results

Index

Logical Foundations of Computer Science

Vol 2: Predicate Logic

Peter A. Fejer & Dan A. Simovici



World Scientific

Logical Foundations of Computer Science

Vol 2: Predicate Logic

Logical Foundations of Computer Science

Vol 2: Predicate Logic

Peter A. Fejer & Dan A. Simovici

University of Massachusetts Boston, USA

World Scientific

NEW JERSEY • LONDON • SINGAPORE • BEIJING • SHANGHAI • HONG KONG • TAIPEI • CHENNAI • TOKYO

Published by

World Scientific Publishing Co. Pte. Ltd.
5 Toh Tuck Link, Singapore 596224
USA office: 27 Warren Street, Suite 401-402, Hackensack, NJ 07601
UK office: 57 Shelton Street, Covent Garden, London WC2H 9HE

Library of Congress Control Number: 2024016854

British Library Cataloguing-in-Publication Data
A catalogue record for this book is available from the British Library.

LOGICAL FOUNDATIONS OF COMPUTER SCIENCE
(In 2 volumes)
Volume 1: Propositional Logic
Volume 2: Predicate Logic

ISBN 978-981-12-8936-1 (set_hardcover)
ISBN 978-981-12-8934-7 (set_ebook for institutions)
ISBN 978-981-12-8935-4 (set_ebook for individuals)
ISBN 978-981-12-8765-7 (vol. 1_hardcover)
ISBN 978-981-12-8766-4 (vol. 2_hardcover)

For any available supplementary material, please visit
https://www.worldscientific.com/worldscibooks/10.1142/13714#t=suppl

Desk Editors: Balasubramanian Shanmugam/Steven Patt

Typeset by Stallion Press
Email: enquiries@stallionpress.com

Printed in Singapore

To our spouses
Elisabeth and Doina

Preface

In scientific reasoning, one starts with a collection of statements, the premises, in order to justify another statement, via a process of inference. Therefore, the study of logic is essential for students of computer science, mathematics, and all who use mathematical proofs.

Many of the fundamental computing concepts were created by logicians. The most famous such concept is the idea of a general-purpose computer, the Turing Machine. Computer programs are written in symbolic languages, e.g., Python, Java, and Lisp, that contain features of logical notations and symbolisms. Through such connections, the study of logic helps in the design of programs. Logic also has a role in the design of new programming languages, and it is essential for work in artificial intelligence.

An introductory chapter presents a set of theoretical and algebraic tools used throughout this book.

The syntactic and semantic concepts of propositional logic are discussed in the second chapter: formulas, truth assignments, truth tables, normal forms, clones of truth functions, and functional completeness. Some parts of logic are used by engineers in circuit design, a topic discussed extensively in the same chapter.

The object of the third chapter is to introduce a variety of propositional formal methods: Hilbert/Frege formal systems, tableaux, sequents, and natural deduction, and to examine transformation methods between these formalisms. Additionally, we present several variants of propositional resolution and the method of cutting planes.

The second part of the work deals with predicate logic also known as first-order logic. The development of this part parallels broadly the presentation of propositional logic. The first chapter of this part presents the syntax and semantics of predicate logic starting with the first-order formulas and structures. Various syntactic aspects specific to predicate logic are presented and then the focus shifts to semantics. We discuss normal forms for formulas and present certain special sets of formulas such as Hintikka sets and first-order theories. Also, the reduction of first-order logic to propositional logic is examined. This chapter concludes with a study of decidability in first-order logic.

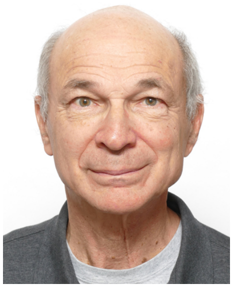
In the following chapter, several important corresponding formalisms for first-order logic are examined: Hilbert/Frege formal systems, tableaux, sequents, and natural deduction. Resolution which forms the basis for logic programming is discussed in various forms and special attention is paid to the method of paramodulation for languages with equality.

The last chapter includes the logical and mathematical analysis of programs, which allows proof of program correctness and analysis of the performance of programs. We discuss the use of logic for proving a variety of assertions concerning the correctness of programs and their performance.

The work contains more than 770 exercises and supplements that can be used to deepen the understanding of the material. We give detailed proofs and we do not shy away from technical difficulties. It is hoped that the readers would enjoy this introduction to logic and make good use of it in their own research.

Lexington and Brookline
Massachusetts
July 2023

About the Authors



Peter Fejer received his BA in Mathematics from Reed College and his SM and PhD degrees in Mathematics from the University of Chicago. He has held positions at Cornell University in the Mathematics Department and at the University of Massachusetts Boston in the Computer Science Department. He was also a Visiting Professor at Heidelberg University on several occasions. At UMass Boston, he served as Chair for 18 years and is now an Emeritus Professor. Professor Fejer's published research is in the area of Computability Theory.

Dan Simovici obtained his PhD in Mathematics from the University of Bucharest, Romania. His main research interests are in machine learning, data mining, and multi-valued logic. Dr. Simovici is a Computer Science Professor and Graduate Program Director at the University of Massachusetts in Boston, was a Visiting Professor in France and Japan, and serves as Editor-in-Chief of the *Journal of Multiple-Valued Logic and Soft Computing*. He is the author or co-author of more than 200 research publications and of several books. Dr. Simovici directed so far 13 PhD dissertations.

Contents

This page intentionally left blank

Volume 2

Predicate Logic

This page intentionally left blank

Chapter 4

First-Order Logic–Syntax and Semantics

4.1 Introduction

First-order logic builds on, and transcends the limitations of, propositional logic. Its capabilities are sufficient to handle the most common arguments of mathematics and computer science, by offering tools for reasoning about programs, representing knowledge, handling constraints and queries in databases, etc.

In this chapter we study the syntax and semantics of first-order logic. The simplest formulas, called atomic formulas, are built starting from variables over individuals and first-order languages conceived as sets of relation and function symbols. Then, the formulas of first-order logic are built in order to express properties of individuals. Their construction begins with atomic formulas and uses connective symbols (familiar from the construction of propositional logic formulas) together with quantifier symbols (over individuals). Special consideration is given to the role played by the equality symbol.

A structure for a first-order language consists of a non-empty set and interpretations of the function and relation symbols of the language as functions and relations on that set. The semantics of formulas of first-order logic is defined using structures and assignments of individuals to variables. Given a first-order language, a structure for the language, and an assignment in the structure, we assign a truth value to each formula of the language.

The term "first-order" refers to the fact that quantification can only be applied to individuals. In higher-order logic, quantification may be applied to sets of individuals as in second-order logic, or to collections of sets of individuals for third-order logic, etc.

We deal with two types of normal form for first-order logic formulas: prenex normal form and Skolem normal form. Using Skolem normal form and a special type of structure, called a Herbrand structure, we reduce first-order logic to propositional logic and thereby provide a means for transferring results, such as the Compactness Theorem, from propositional logic to first-order logic.

As in propositional logic, we define Hintikka sets, special sets of formulas which by their nature show how they can be satisfied. These will be useful in the next chapter for showing the completeness of certain formal systems of first-order logic.

4.2 First-Order Languages

We begin by specifying the symbols of first-order logic. These are:

- A countably infinite set VAR $= \{x_0, x_1, \ldots\}$ whose elements are called *variables*.
- Five *connective symbols* $\neg, \vee, \wedge, \rightarrow, \leftrightarrow$.
- The *quantifier symbols* $\forall$ and $\exists$.
- Three punctuation symbols denoted by the left parenthesis, right parenthesis, and comma.
- For each $n \in \mathbf{N}$, a countably infinite collection $\{R^n_k \mid k \in \mathbf{N}\}$ of *n-ary relation symbols*.
- For each $n \in \mathbf{N}$, a countably infinite collection $\{f^n_k \mid k \in \mathbf{N}\}$ of *n-ary function symbols*.
- The *equality symbol* $=$, which is R^2_0.

We assume that all of the symbols specified above are distinct, but otherwise do not care what mathematical objects these symbols are. (We will return to this point later.)

When we refer to the "symbols" of first-order logic, we are using standard terminology from logic. Since there are infinitely many such symbols, this use of the term is different from that common in computer science where symbols are elements of an alphabet which by definition must be a finite set.

An object which is an n-ary relation symbol for some n is called a *relation symbol*. If R is a relation symbol then the unique natural number n such that R is an n-ary relation symbol is called the *arity* of R. We define the phrases *function symbol* and *arity* of a function symbol similarly. Relation symbols are sometimes called *predicate symbols*. As usual, we use the words "unary" and "binary" in place of 1-ary and 2-ary. 0-ary relation symbols are called *propositional constants* and 0-ary function symbols are called *individual constant symbols* or just *constant symbols*. In order to simplify notation, we will denote the constant symbol f_i^0 by c_i.

The variables, connective symbols, quantifier symbols, punctuation symbols, and the equality symbol are called *logical symbols*. The other symbols of first-order logic are called *extra-logical symbols*. (The difference between logical and extra-logical symbols is that logical symbols have one fixed meaning while extra-logical symbols have a meaning that can vary in different situations.) Extra-logical symbols are often called non-logical symbols.

We will use the letters x, y, and z to denote arbitrary variables, c to denote constant symbols, f, g and h to denote function symbols which are not constant symbols, and R with various adornments to denote relation symbols.

Definition 4.2.1. The *signature of first-order logic* is the pair $S_{\text{FOL}} = (F, \nu)$ where $F = \{f_k^n \mid n, k \in \mathbf{N}\}$ and $\nu(f_k^n) = n$ for every $n, k \in \mathbf{N}$. $\square$

Definition 4.2.2. A *first-order language* $\mathcal{L}$ is a set of relation and function symbols such that $\mathcal{L}$ contains at least one relation symbol and, for every $n \in \mathbf{N}$, there are infinitely many n-ary function symbols and infinitely many n-ary relation symbols that are not elements of $\mathcal{L}$.[1] The elements of $\mathcal{L}$ that are function symbols are called the *function symbols* of $\mathcal{L}$. The set of all function symbols of $\mathcal{L}$ is denoted by $F_{\mathcal{L}}$. The elements of $\mathcal{L}$ that are relation symbols are the *relation symbols* of $\mathcal{L}$. If $\mathcal{L}$ contains the symbol $=$, then we refer to $\mathcal{L}$ as a *language with equality*.

[1]The restriction that a first-order language omits infinitely many n-ary function symbols and infinitely many n-ary relation symbols for every $n \in \mathbf{N}$ is a technical one, which can be safely ignored for the most part. Its role will become apparent later.

If $\mathcal{L}, \mathcal{L}'$ are two first-order languages, then $\mathcal{L}'$ is an *extension* of $\mathcal{L}$ if $\mathcal{L} \subseteq \mathcal{L}'$. $\mathcal{L}'$ is a *finite extension* of $\mathcal{L}$ if it is an extension of $\mathcal{L}$ and $\mathcal{L}' - \mathcal{L}$ is a finite set. $\square$

Example 4.2.3. The set

$$\mathcal{L}_{ar} = \{=, <, 0, s, +, \cdot\},$$

is a first-order language, where $<$ is a binary relation symbol, 0 is a constant symbol, s is a unary function symbol, and $+$, $\cdot$ are binary function symbols. We identify various subsets of this language that will play a role in subsequent developments as:

$$\begin{aligned} \mathcal{L}_s &= \{=, 0, s\} \\ \mathcal{L}_{s,<} &= \{=, <, 0, s\} \\ \mathcal{L}_{pra} &= \{=, <, 0, s, +\}. \end{aligned}$$

The languages $\mathcal{L}_{ar}$ and $\mathcal{L}_{pra}$ are known as the language of arithmetic and the language of Presburger[2] arithmetic, respectively. $\square$

Definition 4.2.4. Let $S_{\text{FOL}} = (F, \nu)$ be the signature of first-order logic and let $\mathcal{L}$ be a first-order language. The *signature of* $\mathcal{L}$ is the pair $S_{\mathcal{L}} = (F_{\mathcal{L}}, \nu_{\mathcal{L}})$, where $F_{\mathcal{L}} = F \cap \mathcal{L}$ and $\nu_{\mathcal{L}} = \nu \restriction F_{\mathcal{L}}$. $\square$

Observe that for any first-order language $\mathcal{L}$, we have $S_{\mathcal{L}} \preceq S_{\text{FOL}}$ (see Definition 1.5.6).

The union of two first-order languages is *not*, in general, a first-order language. Indeed, let K be the set of constant symbols and assume that $K = K' \cup K''$, where K' and K'' are disjoint infinite sets of constant symbols. $K' \cup \{=\}$ and $K'' \cup \{=\}$ are first-order languages. However, their union $K \cup \{=\}$ is not (since $K \cup \{=\}$ omits no constant symbol). Observe also that if $\mathcal{L}'$ and $\mathcal{L}''$ are finite first-order languages, then $\mathcal{L}' \cup \mathcal{L}''$ is a first-order language.

[2]Mojżesz Presburger, a Polish logician, was born in Warsaw on December 27, 1904 and died in 1943, a victim of Nazi persecution. His main contribution is contained in his master's thesis entitled "Über die Vollständigkeit eines gewissen Systems der Arithmetik ganzer Zahlen," presented at the University of Warsaw, which is regarded as one of the earliest contributions to model theory.

Definition 4.2.5. Let $\mathcal{L}$ be a first-order language. The *extension by constants of* $\mathcal{L}$ is the first-order language $\mathcal{L}^c$ given by

$$\mathcal{L}^c = \mathcal{L} \cup \{c_{i_0}, c_{i_2}, c_{i_4}, \ldots\},$$

where $c_{i_0}, c_{i_1}, c_{i_2}, \ldots$ (with $i_0 < i_1 < i_2 < \cdots$) is the list of all constant symbols that do not belong to $\mathcal{L}$.

If Γ is a set of $\mathcal{L}^c$-formulas that contains only finitely many constant symbols from $\mathcal{L}^c - \mathcal{L}$, then we say that Γ is a set with *limited constant symbols.* □

In the phrase "first-order language", we use the term "language" as it is commonly used in logic. This usage developed independently of the use of the term "language" in computer science as a set of words over an alphabet, but the two usages can be united if the "symbols" of first-order logic are themselves identified as being words over an alphabet. For instance, suppose that V is the alphabet containing the following symbols: $x, r, f, \star, 0, 1$ and for each $n \in \mathbf{N}$ let κ_n be the usual binary representation for n (so $\kappa_0 = 0$ and for $n > 0$, κ_n has no leading zeros). We can then specify that the "symbols" of first-order logic (which we have left undefined earlier) are in fact words over V as given in the following table.

Symbol of first-order logic	Definition
x_n	$x\kappa_n$
R^n_k	$r\kappa_n \star \kappa_k$
f^n_k	$f\kappa_n \star \kappa_k$

Every other symbol of first-order logic is encoded as itself. We denote the substitution that takes a symbol of first-order logic to its encoding as a word over V by code. Note that if a, b are two distinct symbols of first-order logic, then $\mathsf{code}(a)$ is not a suffix of $\mathsf{code}(b)$. Therefore, by Supplement 13 of Chapter 1, if S is the set of symbols of first-order logic, the mapping $\mathsf{code} : \mathrm{Seq}(S) \longrightarrow V^*$ is injective.

With this definition of the symbols of first-order logic, a first-order language is a language in the computer science sense, in fact, a language over the alphabet V. For our purposes, there is no particular advantage in making this definition except to eliminate an ambiguity in terminology, but as we will discuss later, in other contexts such an identification of the symbols of first-order logic is important.

4.3 Terms and Formulas

In this section, we present the syntax of the main objects dealt with by first-order logic: terms and formulas. The intuition is that terms represent individuals, while formulas make assertions about individuals. Once we have defined terms and formulas, we will examine certain syntactic properties of formulas, as well as their syntactic behavior with respect to substitutions.

4.3.1 *Terms of First-Order Logic*

We now undertake a detailed study of the terms of the signature of first-order logic and the signature of a first-order language.

Definition 4.3.1. Let $\mathcal{L}$ be a first-order language and let $S_{\mathcal{L}}$ be its signature. $\mathrm{TERM}_{\mathcal{L}}$, the set of *terms* of the first-order language $\mathcal{L}$, is the set $\mathrm{TERM}_{S_{\mathcal{L}}}(\mathrm{VAR})$.

If $V \subseteq \mathrm{VAR}$, then $\mathrm{TERM}_{\mathcal{L}}(V)$ denotes the set $\mathrm{TERM}_{S_{\mathcal{L}}}(V)$. We refer to the members of $\mathrm{TERM}_{\mathcal{L}}(V)$ as $(\mathcal{L}, V)$-*terms*.

A set of variables V is called $\mathcal{L}$-*suitable* if $\mathrm{TERM}_{\mathcal{L}}(V) \neq \emptyset$. □

Note that if $\mathcal{L}$ contains a constant symbol, then every set of variables is $\mathcal{L}$-suitable; if $\mathcal{L}$ does not contain any constant symbol, then V is $\mathcal{L}$-suitable if and only if $V \neq \emptyset$.

Example 4.3.2. Let $\mathcal{L}_{ar}$ be the language of arithmetic as defined in Example 4.2.3. Since variables and constant symbols of $\mathcal{L}_{ar}$ are terms of $\mathcal{L}_{ar}$, we have $x_0, x_1, 0 \in \mathrm{TERM}_{\mathcal{L}_{ar}}$. Therefore, by Definition 4.3.1, we obtain the terms $t_1 = s(0)$, $t_2 = s(s(0))$ and $t_3 = s(x_1)$. The following are further terms of the language:

$$\begin{aligned} \cdot(t_1, t_2) &= \cdot(s(0), s(s(0))) \\ \cdot(t_1, x_0) &= \cdot(s(0), x_0), \\ +(t_2, t_3) &= +(s(s(0)), s(x_1)). \end{aligned}$$

We prefer to use the notation $(t + t')$ and $(t \cdot t')$ instead of $+(t, t')$ and $\cdot(t, t')$ or even $t + t'$ and $t \cdot t'$, when there is no risk of confusion.

For a term $u = s(s(\cdots(s(t))\cdots))$ of $\mathcal{L}_{ar}$, where t is a term of the same language and s occurs n times, we will use the notation $s^n(t)$.

Note that the set $\mathrm{TERM}_{\mathcal{L}_s}$ consists of strings of the form $s^n(u)$, where u is either the constant symbol 0 or a variable. □

Definition 4.3.3. Let $\mathcal{L}$ be a first-order language. The set of *ground terms* of $\mathcal{L}$, $\mathrm{GTERM}_{\mathcal{L}}$, is the set of ground terms of the signature $S_{\mathcal{L}}$. □

Theorem 1.5.9 implies that $\mathrm{GTERM}_{\mathcal{L}}$ consists of those terms of $\mathcal{L}$ that do not contain any variables.

Note that the terms t_1, t_2, and $f_1^2(t_1, t_2)$ from Example 4.3.2 are ground terms of $\mathcal{L}$, while $t_3, f_1^2(t_1, x_0)$ and $f_1^2(t_2, t_3)$ are not.

Example 4.3.4. The observation in Example 1.5.5 implies that if $\mathcal{L}$ is a language without function symbols, then $\mathrm{TERM}_{\mathcal{L}} = \{x_0, x_1, \ldots\}$. □

It is easy to verify that if $\mathcal{L}_1 \subseteq \mathcal{L}_2$, then $\mathrm{TERM}_{\mathcal{L}_1} \subseteq \mathrm{TERM}_{\mathcal{L}_2}$.

Definition 4.3.5. A sequence of symbols t is a *term of first-order logic*, or simply a term, if it belongs to the set $\mathrm{TERM}_{S_{\mathrm{FOL}}}(\mathrm{VAR})$. We use TERM to denote the set of all terms. □

Theorem 4.3.6. *A sequence t is a term of first-order logic if and only if there is a first-order language $\mathcal{L}$ such that t is a term of $\mathcal{L}$.*

Proof. Since for every first-order language $\mathcal{L}$, $S_{\mathcal{L}} \preceq S_{\mathrm{FOL}}$, it follows by Theorem 1.5.7 that every term of a first-order language is a term of first-order logic.

Conversely, we show that for every $t \in \mathrm{TERM}$, there is a finite first-order language $\mathcal{L}$ such that $t \in \mathrm{TERM}_{\mathcal{L}}$.

If t is a variable, then $t \in \mathrm{TERM}_{\emptyset}$. If t is a constant symbol, then t is a term of the first-order language $\{t\}$. Now suppose that f is an n-ary function symbol with $n > 0$, $t_0, \ldots, t_{n-1}$ are members of $\mathrm{TERM}_{S_{\mathrm{FOL}}}(\mathrm{VAR})$, and $t = f(t_0, \ldots, t_{n-1})$. If, as inductive hypothesis, we assume that there are finite languages $\mathcal{L}_i$ such that $t_i \in \mathrm{TERM}_{\mathcal{L}_i}$ for $0 \leq i \leq n-1$, then $t_i \in \mathrm{TERM}_{\mathcal{L}}$ where $\mathcal{L}$ is the finite language $(\bigcup_{0 \leq i \leq n-1} \mathcal{L}_i) \cup \{f\}$. Therefore, $t \in \mathrm{TERM}_{\mathcal{L}}$. □

Definition 4.3.7. A sequence of symbols t is a *ground term of first-order logic* or simply a ground term, if it is a ground term of S_{FOL}. The set of ground terms is denoted by GTERM. □

Theorem 4.3.8. *A sequence t is a ground term of first-order logic if and only if there is a first-order language $\mathcal{L}$ such that t is a ground term of $\mathcal{L}$.*

Proof. The argument is similar to that of Theorem 4.3.6 and it is left to the reader. □

4.3.2 *Formulas of First-Order Logic*

In this subsection, we define inductively the formulas of a first-order language and of first-order logic and we prove the unique readability of these definitions. This unique readability is essential for justifying the correctness of recursive definitions of functions whose domains are various sets of first-order formulas.

Definition 4.3.9. Let $\mathcal{L}$ be a first-order language. We define $\mathrm{FORM}_{\mathcal{L}}$, the set of *formulas* of $\mathcal{L}$ (or the set of $\mathcal{L}$-*formulas*), as the set of sequences of logical symbols, function symbols of $\mathcal{L}$, and relation symbols of $\mathcal{L}$ given by the following inductive definition:

(1) Every propositional constant of $\mathcal{L}$ is a formula of $\mathcal{L}$.
(2) If R is a relation symbol of $\mathcal{L}$ of arity $n > 0$ and $t_0, \ldots, t_{n-1}$ are all terms of $\mathcal{L}$, then $R(t_0, \ldots, t_{n-1})$ is a formula of $\mathcal{L}$.
(3) If φ is a formula of $\mathcal{L}$, then so is $(\neg\varphi)$.
(4) If φ and ψ are both formulas of $\mathcal{L}$, then so are $(\varphi \wedge \psi)$, $(\varphi \vee \psi)$, $(\varphi \to \psi)$, and $(\varphi \leftrightarrow \psi)$.
(5) If x is a variable and φ is a formula of $\mathcal{L}$, then $(\forall x)\varphi$ and $(\exists x)\varphi$ are both formulas of $\mathcal{L}$.

▯

Elements of $\mathrm{FORM}_{\mathcal{L}}$ which are put in by one of the first two rules are called *atomic formulas* of $\mathcal{L}$. We denote the set of all atomic formulas of $\mathcal{L}$ by $\mathrm{AFORM}_{\mathcal{L}}$. If the relation symbol of an atomic formula is $=$, then the atomic formula is called an $\mathcal{L}$-*equality.* We denote the set of all $\mathcal{L}$-equalities as $\mathrm{EQ}_{\mathcal{L}}$.

If the relation symbol in an atomic formula is distinct from $=$, we refer to the formula as a *non-equality $\mathcal{L}$-atomic formula.* The set of such formulas is denoted as $\mathrm{AFORMNE}_{\mathcal{L}}$. Note that if $\mathcal{L}$ contains no other relation symbol than $=$, then $\mathrm{AFORMNE}_{\mathcal{L}} = \emptyset$.

Also, if V is a set of variables, $\mathrm{AFORM}_{\mathcal{L}}(V)$ denotes the set of atomic formulas of $\mathcal{L}$ such that all variables that occur in the formula belong to V. We refer to such formulas as $(\mathcal{L}, V)$-*atomic formulas*. Similarly, we define $\mathrm{EQ}_{\mathcal{L}}(V)$ and $\mathrm{AFORMNE}_{\mathcal{L}}(V)$.

The *literals* of $\mathcal{L}$ are the atomic formulas of $\mathcal{L}$ and their negations. We denote the set of literals of $\mathcal{L}$ by $\mathrm{LIT}_{\mathcal{L}}$. The set of all literals LIT is $\bigcup_{\mathcal{L}} \mathrm{LIT}_{\mathcal{L}}$. Also, if V is a set of variables, $\mathrm{LIT}_{\mathcal{L}}(V)$ denotes the set of literals of $\mathcal{L}$ such that all variables that occur in the literal belong to V. Similarly, we denote by $\mathrm{LIT}(V)$ the set of all literals constructed using variables of the set V.

The *ground atomic formulas* of $\mathcal{L}$ are the atomic formulas of $\mathcal{L}$ that contain no variables. We denote the set of ground atomic formulas of $\mathcal{L}$ by $\mathrm{GAFORM}_{\mathcal{L}}$.

The set of $\mathcal{L}$-equalities that are ground atomic formulas is denoted as $\mathrm{GEQ}_{\mathcal{L}}$ and the set of non-equality ground $\mathcal{L}$-atomic formulas is denotes as $\mathrm{GAFORMNE}_{\mathcal{L}}$.

The *prime* formulas of $\mathcal{L}$ are the atomic formulas of $\mathcal{L}$ and the formulas introduced by the fifth rule of Definition 4.3.9.

If φ and ψ are two $\mathcal{L}$-formulas, then $(\varphi \wedge \psi)$ and $(\varphi \vee \psi)$ are the *conjunction* and *disjunction* of φ and ψ, respectively.

The *quantifier-free formulas* of $\mathcal{L}$ are those formulas of $\mathcal{L}$ that do not contain occurrences of quantifier symbols. It is easily seen that an inductive definition of the set of quantifier-free formulas of $\mathcal{L}$ can be obtained from Definition 4.3.9 by omitting the fifth rule.

A *universal formula* of $\mathcal{L}$ is an $\mathcal{L}$-formula $(\forall y_0)\cdots(\forall y_{n-1})\varphi$, where φ is quantifier-free, $n \in \mathbf{N}$ and $y_0, \ldots, y_{n-1}$ are distinct variables.

If ψ is a universal formula, $\psi = (\forall y_0)\cdots(\forall y_{n-1})\varphi$, we denote by $\psi^{-\forall}$ the quantifier-free formula φ. When Γ is a set of universal formulas, the previous notation is extended by writing $\Gamma^{-\forall}$ for the set $\{\psi^{-\forall} \mid \psi \in \Gamma\}$.

An *existential formula* of $\mathcal{L}$ is an $\mathcal{L}$-formula $(\exists y_0)\cdots(\exists y_{n-1})\varphi$, where φ is quantifier-free, $n \in \mathbf{N}$ and $y_0, \ldots, y_{n-1}$ are distinct variables.

Note that every quantifier-free formula of $\mathcal{L}$ is both a universal and an existential formula of $\mathcal{L}$.

An $\mathcal{L}$-*negative formula* is a formula φ such that $\varphi = (\neg\psi)$ for some formula $\psi \in \mathrm{FORM}_{\mathcal{L}}$. An $\mathcal{L}$-formula is $\mathcal{L}$-*positive* if it is not negative.

Using terminology of Smullyan (see [31]), we will refer to formulas of the forms $(\forall x)\varphi$ and $(\neg(\exists x)\varphi)$ as $\boldsymbol{\gamma}$-formulas and to formulas of the forms $(\exists x)\varphi$ and $(\neg(\forall x)\varphi)$ as $\boldsymbol{\delta}$-formulas.

The $\mathcal{L}$-formula $(\forall y_0)\cdots(\forall y_{n-1})\varphi$ is a *generalization* of the $\mathcal{L}$-formula φ, where $y_0, \ldots, y_{n-1}$ are variables. Note that φ is a generalization of itself (obtained by taking $n = 0$).

The notion of "optimized negation" introduced in propositional logic by Definition 2.2.5 can be adapted to first-order logic.

Definition 4.3.10. Let φ be a formula. Then $\overline{\varphi}$, the *complement* of φ, is defined by:

$$\overline{\varphi} = \begin{cases} (\neg\varphi) & \text{if } \varphi \text{ is a positive formula,} \\ \psi & \text{if } \varphi = (\neg\psi) \text{ is a negative formula.} \end{cases}$$

□

Note that the complement of a literal is a literal.

Example 4.3.11. Let $\mathcal{L} = \{R, P, c\}$ be a first-order language, where R is a unary relation symbol, P is a binary relation symbol, and c is a constant symbol. The formula $R(c)$ is a ground atomic $\mathcal{L}$-formula, $R(x)$ is an atomic $\mathcal{L}$-formula but not a ground atomic formula, $R(x), R(c)$ and $(\neg R(x))$ are literals of $\mathcal{L}$, $(R(x) \wedge R(c))$ is a quantifier-free $\mathcal{L}$-formula, and $(\forall x)(R(x) \wedge R(c))$ is a prime formula of $\mathcal{L}$.

All of the formulas mentioned so far in this example are universal $\mathcal{L}$-formulas, and all but the last are also existential $\mathcal{L}$-formulas. Finally, the formula $(\exists x)(\exists y)(P(x,y) \wedge R(x))$ is an existential $\mathcal{L}$-formula. □

Note that Rule 4 actually consists of four rules, which we denote by $4_\wedge, 4_\vee, 4_\rightarrow, 4_\leftrightarrow$. For example, $4_\wedge$ is the rule

$4_\wedge$. If φ and ψ are both formulas of $\mathcal{L}$, then so is $(\varphi \wedge \psi)$.

More importantly, Rule 5 actually consists of an infinite set of rules, two for each variable x, which we denote by $5_{(\forall x)}$ and $5_{(\exists x)}$:

$5_{(\forall x)}$. If φ is a formula of $\mathcal{L}$, then $(\forall x)\varphi$ is a formula of $\mathcal{L}$.
$5_{(\exists x)}$. If φ is a formula of $\mathcal{L}$, then $(\exists x)\varphi$ is a formula of $\mathcal{L}$.

We will use the letters $\varphi, \psi, \theta, \alpha, \beta, \gamma, \delta$ to stand for formulas.

If $\mathcal{L}$ is a first-order language with equality, the atomic formula $= (t_0, t_1)$ (or $R_0^2(t_0, t_1)$) will be denoted by $t_0 = t_1$, or by $(t_0 = t_1)$ when this formula is part of a larger formula. We stress that this is only a notational device; even when we write $t_0 = t_1$, the "official" formula still remains $= (t_0, t_1)$. We will also write $t_0 \neq t_1$ or $(t_0 \neq t_1)$ to denote the formula $(\neg(t_0 = t_1))$. Similarly, when R is a binary relation symbol, we may use the notation $t_0 R t_1$ instead of $R(t_0, t_1)$. For example, we may write $t_0 < t_1$ instead of $< (t_0, t_1)$, when this does not cause any confusion.

Example 4.3.12. The $\mathcal{L}_{ar}$-formula $((u = v) \vee (u < v))$ will be alternatively written as $u \leq v$. ◻

Note that $\text{FORM}_{\mathcal{L}}$ is non-empty because $\mathcal{L}$ contains at least one relation symbol. Indeed, if R is an n-ary relation symbol of $\mathcal{L}$ with $n > 0$, then $R(x_0, \ldots, x_{n-1}) \in \text{FORM}_{\mathcal{L}}$; if R is a propositional constant of $\mathcal{L}$, then $R \in \text{FORM}_{\mathcal{L}}$.

If, as discussed above, we identify symbols of logic as being words over some basic alphabet W, then it is natural to think of terms and formulas of a first-order language as being words over W rather than as being sequences of symbols of first-order logic. For instance, using the possible definition given earlier of the symbols of first-order logic as words over $W = \{x, \mathtt{t}, \mathtt{f}, r, f, \star, 0, 1\}$, the atomic formula $R_1^1(x_2)$ becomes the word $r1 \star 1(x10)$ over W. If this is done, then the collection of all terms over a first-order language and the set of all formulas over a first-order language are themselves languages (in the computer science sense) over W.

Our remarks from Section 2.2 regarding how the size of a formula is measured when considering an algorithm which applies to formulas remain valid here. However, if we were to define formulas directly in their encoded form, certain technical results, such as unique readability of formulas, would be more difficult to show. For this reason, we regard terms and formulas in the way we have defined them; namely, as sequences of logical symbols. To emphasize this point, for us, if R is a unary relation symbol and x is any variable, then $R(x)$ is a formula of length 4; whereas, if a formula is a word over an alphabet, then the length of $R(x)$ can be arbitrarily long depending on exactly which relation symbol R is and which variable x is. As before, we denote the size of the formula φ by $\mathtt{size}(\varphi)$.

Example 4.3.13. The formula $\varphi = R_0^1(x_7)$ is encoded as $r1 \star 0(x111)$. Therefore, $\mathtt{size}(\varphi) = 10$. □

Definition 4.3.14. φ is *first-order formula* if there is a first-order language $\mathcal{L}$ such that $\varphi \in \mathrm{FORM}_{\mathcal{L}}$. We denote the set of first-order formulas by FORM.

Similarly, we say that φ is an *atomic formula* (a *literal*, a *ground atomic formula*, a *prime formula*, a *quantifier-free formula*, *negative formula*) if there is a first-order language $\mathcal{L}$ such that φ is an atomic formula (a literal, a ground atomic formula, a prime formula, a quantifier-free formula, negative formula) of $\mathcal{L}$. □

If we order the symbols of W as previously listed, this generates a lexicographic order on W^*. As we did for propositional logic, we can define a well-ordering "$\preceq$" on FORM as follows: $\varphi \preceq \psi$ if $|\mathsf{code}(\varphi)| < |\mathsf{code}(\psi)|$ or, if $|\mathsf{code}(\varphi)| = |\mathsf{code}(\psi)|$, then $\mathsf{code}(\varphi)$ precedes $\mathsf{code}(\psi)$ in the lexicographic order on W^*. We will refer to $\preceq$ as the *standard ordering* of the formulas of first-order logic. The similarly obtained order on TERM will be referred to as the *standard ordering of terms.* These orderings can be particularized to $\mathrm{FORM}_{\mathcal{L}}$ and $\mathrm{TERM}_{\mathcal{L}}$, respectively. If $\varphi_0, \varphi_1, \ldots$ is the standard ordering of the $\mathcal{L}$-formulas, we will single out the formulas $(\varphi_0 \wedge (\neg\varphi_0))$ and $(\varphi_0 \vee (\neg\varphi_0))$ as $\mathsf{false}^{\mathcal{L}}$ and $\mathsf{true}^{\mathcal{L}}$.

As noted earlier, code is an injective function on the set of sequences of symbols of first-order logic. Since both TERM and FORM are subsets of this set, it follows that both are countably infinite.

Theorem 4.3.15. FORM *equals the set F given by the following inductive definition*:

(1) *Every propositional constant is in F.*
(2) *If R is a relation symbol of arity $n > 0$ and $t_0, \ldots, t_{n-1}$ are terms, then $R(t_0, \ldots, t_{n-1})$ is in F.*
(3) *If $\varphi \in F$, then $(\neg\varphi)$ is in F.*
(4) *If $\varphi, \psi \in F$, then $(\varphi \wedge \psi)$, $(\varphi \vee \psi)$, $(\varphi \to \psi)$, and $(\varphi \leftrightarrow \psi)$ are in F.*
(5) *If x is a variable and $\varphi \in F$, then $(\forall x)\varphi$ and $(\exists x)\varphi$ are in F.*

Proof. The proof is similar to the proof of Theorem 4.3.6, and it is left to the reader. □

Definition 4.3.16. Let φ be a first-order formula. The first-order language $\mathcal{L}_\varphi$ consists of the function and relation symbols that occur in φ. ▯

Note that $\mathcal{L}_\varphi$ is a finite set, so is a first-order language according to Definition 4.2.2.

Theorem 4.3.17. *If φ is a first-order formula, then φ is a $\mathcal{L}_\varphi$-formula. Moreover, if φ is an $\mathcal{L}$-formula, then $\mathcal{L}_\varphi \subseteq \mathcal{L}$.*

Proof. We leave it to the reader to show by induction on terms that if t is a term that occurs in φ, then t is $\mathcal{L}_\varphi$-term. An induction on formulas ψ now shows that every subformula ψ of φ is an $\mathcal{L}_\varphi$-formula. Therefore, in particular, φ is an $\mathcal{L}_\varphi$-formula.

The second part is obvious and is left to the reader. □

Our definition of formulas of a first-order language satisfies the unique readability condition. In order to show this, we need to show the analogue of Lemma 1.5.11 for formulas.

Lemma 4.3.18. *No proper prefix of a formula is a formula. Furthermore, no suffix of a formula is a proper prefix of a formula.*

Proof. We extend the function K introduced in Definition 1.5.10 to the set of all symbols of first-order logic as follows:

symbol s	$\vee, \wedge, \rightarrow, \leftrightarrow$	$\neg$	$\forall, \exists$	n-ary relation symbol R
$K(s)$	-1	0	-1	$1-n$

As before, we extend K to the set of all sequences of symbols of first-order logic by letting

$$K((s_0, \ldots, s_{n-1})) = \Sigma_{i=0}^{n-1} K(s_i).$$

We now prove that for every formula φ, $K(\varphi) = 1$. If φ is a propositional constant, we have $K(\varphi) = 1$ because a propositional constant is a 0-ary relation symbol. Now let R be an n-ary relation symbol with $n \geq 1$ and let $t_0, \ldots, t_{n-1}$ be n terms. As we proved in Lemma 1.5.11, $K(t_i) = 1$ for $0 \leq i \leq n-1$. Thus

$$K(R(t_0, \ldots, t_{n-1})) = (1-n) + (-1) + \underbrace{1 + \cdots + 1}_{n} + 1 = 1.$$

Suppose that φ and ψ are formulas such that $K(\varphi) = K(\psi) = 1$. Observe that we have

$$\begin{aligned}
K\left((\neg\varphi)\right) &= -1 + 0 + 1 + 1 = 1 \\
K\left((\varphi \vee \psi)\right) &= -1 + 1 - 1 + 1 + 1 = 1 \\
K\left((\varphi \wedge \psi)\right) &= -1 + 1 - 1 + 1 + 1 = 1 \\
K\left((\varphi \rightarrow \psi)\right) &= -1 + 1 - 1 + 1 + 1 = 1 \\
K\left((\varphi \leftrightarrow \psi)\right) &= -1 + 1 - 1 + 1 + 1 = 1 \\
K\left((\forall x)\varphi\right) &= -1 - 1 + 1 + 1 + 1 = 1 \\
K\left((\exists x)\varphi\right) &= -1 - 1 + 1 + 1 + 1 = 1
\end{aligned}$$

Next, we prove that if φ is a formula, then for every proper prefix u of φ, $K(u) < 1$. This implies that a proper prefix of formula cannot be a formula.

Note that if φ is a propositional constant, then φ has no proper prefixes. We leave for the reader the argument for the prefixes of the formula $R(t_0, \ldots, t_{n-1})$, which works like the similar argument of Lemma 1.5.11, where R is an n-ary relation symbol, $n \geq 1$, and $t_0, \ldots, t_{n-1}$ are terms.

Now suppose that φ and ψ are formulas and that $K(u), K(v) < 1$, for every proper prefix u of φ and v of ψ. Let C be a binary connective symbol, Q be a quantifier symbol and x be a variable. The following tables present the cases which may arise for prefixes of formulas which we can build starting from φ and ψ.

Proper prefixes of $(\neg\varphi)$

Proper prefix w	$K(w)$
$($	-1
$(\neg$	$-1 + 0 = -1$
$(\neg u$	$-1 + 0 + K(u) < 0$
$(\neg\varphi$	$-1 + 0 + 1 = 0$

Proper prefixes of $(\varphi C\psi)$

Proper prefix w	$K(w)$
$($	-1
$(u$	$-1 + K(u) < 0$
$(\varphi$	$-1 + 1 = 0$
$(\varphi C$	$-1 + 1 - 1 = -1$
$(\varphi C v$	$-1 + 1 - 1 + K(v) < 0$
$(\varphi C \psi$	$-1 + 1 - 1 + 1 = 0$

Proper prefixes of $(Qx)\varphi$

Proper prefix w	$K(w)$
$($	-1
$(Q$	$-1 - 1 = -2$
$(Qx$	$-1 - 1 + 1 = -1$
(Qx)	$-1 - 1 + 1 + 1 = 0$
$(Qx)u$	$-1 - 1 + 1 + 1 + K(u) < 1$

For the second part of the lemma, let v be a suffix of a formula φ, say $\varphi = uv$. If $v = \lambda$, then, clearly, v cannot be a proper prefix of a formula. Suppose that $v \neq \lambda$. Then, $u \neq \varphi$, so $K(u) < 1$, by the first part of the argument when $u \neq \lambda$, or by definition when $u = \lambda$. Since $K(v) = K(\varphi) - K(u) = 1 - K(u) > 0$, it follows that $K(v) \geq 1$, so v cannot be a proper prefix of a formula. □

Theorem 4.3.19. *For every first-order language $\mathcal{L}$, the definition of formulas of $\mathcal{L}$ (Definition* 4.3.9) *meets the unique readability condition.*

Proof. Note that a formula must begin with either (, if it is put in by the last three rules, or a relation symbol, if it is put in by first or second rule.

If α is put in by one of the first two rules, then the rule by which it is put in is uniquely determined by the first symbol of α.

Suppose now that α is put in by the third rule. Then the second symbol of α is $\neg$. Therefore, α cannot be put in by Rule 4, because no formula can start with $\neg$, and cannot be put in by Rule 5, because the second symbol of a formula put in by Rule 5 is a quantifier symbol. Moreover, if $\alpha = (\neg\varphi) = (\neg\varphi')$, then $\varphi = \varphi'$.

If α is put in by Rule 4_C, where C is one of the binary connectives, then α cannot be put in by any of the other parts of Rule 4. Indeed, if $\alpha = (\varphi C \psi) = (\varphi' C' \psi')$, then, since φ cannot be a proper prefix of φ' and vice-versa, we obtain $\varphi = \varphi'$, $C = C'$ and $\psi = \psi'$. Also, α cannot be put in by any of the rules $5_{(Qx)}$ because no formula starts with a quantifier symbol. Note also that the preceding argument also shows that the hypotheses φ and ψ used to construct α are unique.

Finally, assume that α is put in by Rule $5_{(Qx)}$ where Q is a quantifier symbol. If $\alpha = (Qx)\varphi = (Q'x')\varphi'$, then $Q = Q', x = x'$ and $\varphi = \varphi'$. This shows both that the rule used is uniquely determined and that the hypothesis φ is unique. □

Note that if φ is an atomic formula of the form $R(t_0, \ldots, t_{n-1})$, then R and $t_0, \ldots, t_{n-1}$ are uniquely determined. Indeed, if $R(t_0, \ldots, t_{n-1}) = R'(t'_0, \ldots, t'_{m-1})$, then $R = R'$ since they are both the first symbol of the formula. Therefore, $n = m$. Suppose that for some i, $0 \leq i \leq n-1$ we have $t_i \neq t'_i$. Let i_0 be the least i with this property. Of course, for $j < i_0$, we have $t_j = t'_j$ and this gives

$$t_{i_0}, \ldots, t_{n-1}) = t'_{i_0}, \ldots, t'_{n-1}).$$

Since t_{i_0} cannot be a proper prefix of t'_{i_0} nor vice-versa, $t_{i_0} = t'_{i_0}$, which contradicts the definition of i_0. Therefore, $t_i = t'_i$ for each i, $0 \leq i \leq n-1$.

If $(\varphi_0, \ldots, \varphi_{n-1})$ is a nonempty, finite sequence of formulas, we introduce the notations $(\varphi_0 \vee \cdots \vee \varphi_{n-1})$, $\bigvee_{0 \leq i \leq n-1} \varphi_i$, $(\varphi_0 \wedge \cdots \wedge \varphi_{n-1})$, and $\bigwedge_{0 \leq i \leq n-1} \varphi_i$, in the same way as we did for propositional logic. The formula denoted by $(\varphi_0 \vee \cdots \vee \varphi_{n-1})$, and $\bigvee_{0 \leq i \leq n-1} \varphi_i$ is called the *disjunction* of the sequence of formulas $(\varphi_0, \ldots, \varphi_{n-1})$; the formula denoted by $(\varphi_0 \wedge \cdots \wedge \varphi_{n-1})$, and $\bigwedge_{0 \leq i \leq n-1} \varphi_i$ is called the *conjunction* of the sequence of formulas $(\varphi_0, \ldots, \varphi_{n-1})$.

If Γ is a nonempty set of formulas, and $(\varphi_0, \ldots, \varphi_{n-1})$ is the list of the formulas of Γ in standard order, we write $\bigvee \Gamma$ and $\bigwedge \Gamma$ for the formulas $\bigvee_{0 \leq i \leq n-1} \varphi_i$ and $\bigwedge_{0 \leq i \leq n-1} \varphi_i$, respectively.

4.3.3 *Occurrences in Formulas*

This subsection contains various technical results about occurrences of variables and formulas in other formulas. We define syntactic concepts (such as free and bound occurrences of variables) in a manner as close to the intuition as possible and then produce equivalent recursive characterizations of the same concepts. While we could have taken these recursive characterizations as definitions and thereby reduced some of the tedium associated with this material, we feel that the nonrecursive definitions are more natural.

Let φ be a formula of first-order logic and let s be a symbol of first-order logic. According to Definition 1.2.3, an occurrence of s in φ is a pair (s, p) where $0 \leq p \leq |\varphi| - 1$ and $\varphi(p) = s$, and the set of all occurrences of s in φ is denoted by $\mathtt{OCC}_s(\varphi)$.

The set of constant symbols that occur in a formula φ is denoted by $\mathtt{C}(\varphi)$.

If φ and ψ are formulas, again, by Definition 1.2.3, an occurrence of ψ in φ is a pair (ψ, p) such that $0 \leq p \leq |\varphi| - |\psi|$ and $\psi(\ell) = \varphi(p+\ell)$ for $0 \leq \ell \leq |\psi| - 1$. If there is an occurrence of ψ in φ, then we refer to ψ as a *subformula* of φ.

Example 4.3.20. Consider the first-order formula

$$\varphi = (R(c, f(x, c)) \vee (\forall x) R'(g(x), y)).$$

We assume that c is a constant symbol, x, y are variables, R, R' are binary relation symbols, f is a binary and g is a unary function symbol. The reader can easily verify that $|\varphi| = 27$. Note that we have three occurrences of x in φ, namely, $(x, 7), (x, 15)$ and $(x, 21)$.

Also, there is one occurrence of the formula $\alpha = R'(g(x), y)$ in φ, namely $(\alpha, 17)$ and one occurrence, $(\forall, 14)$ of the quantifier symbol $\forall$.

□

Definition 4.3.21. Let φ be a first-order formula. Its *norm* $\| \varphi \|$ is

$$|\mathtt{OCC}_{\forall}(\varphi)| + |\mathtt{OCC}_{\exists}(\varphi)| + 2 \left(\sum_{C \in \{\vee, \wedge, \rightarrow, \leftrightarrow\}} |\mathtt{OCC}_C(\varphi)| \right) + |\mathtt{OCC}_{\neg}(\varphi)|.$$

□

In other words, $\|\varphi\|$ is the total number of occurrences of quantifier symbols and negation symbols in φ plus twice the number of occurrences of binary connective symbols in φ.

Example 4.3.22. The norm of the formula $(\forall y)(\forall z)((\exists x)(R(x) \wedge P(y)) \wedge (\neg Q(y, z)))$ is 8. □

Theorem 4.3.23 (Occurrence Theorem for First-Order Logic). *Let φ, ψ, α be formulas.*

If φ is an atomic formula and $\alpha \neq \varphi$, then α does not occur in φ.

If $\alpha \neq (\neg\varphi)$, then every occurrence of α in $(\neg\varphi)$ is part of φ. (More exactly, each occurrence of α in $(\neg\varphi)$ is part of the occurrence $(\varphi, 2)$.)

Let C be a binary connective symbol. If $\alpha \neq (\varphi C \psi)$, then every occurrence of α in $(\varphi C \psi)$ is either part of φ or part of ψ. (More exactly, each occurrence of α in $(\varphi C \psi)$ is either a part of $(\varphi, 1)$ or a part of $(\psi, |\varphi| + 2)$.)

If $\alpha \neq (Qx)\varphi$, where Q is a quantifier symbol and x is a variable, then every occurrence of α in $(Qx)\varphi$ is part of φ. (Specifically, each occurrence of α in $(Qx)\varphi$ is a part of the occurrence $(\varphi, 4)$.)

Proof. If φ is a propositional constant and $\alpha \neq \varphi$, it is immediate that α does not occur in φ. Suppose now that $n > 0$ and φ is the atomic formula $R(t_0, \ldots, t_{n-1})$. Since every formula contains at least one relation symbol and terms do not contain relation symbols, if α occurs in φ, it must start with R and so by Lemma 4.3.18, $\alpha = \varphi$.

Let (α, j) be an occurrence of α in $(\varphi C \psi)$. We have $j \neq 0$ because $\alpha \neq (\varphi C \psi)$ and no formula can be a proper prefix of another formula. Since no formula starts with a connective symbol or a close parenthesis, the occurrence of α in $(\varphi C \psi)$ must begin either within φ or ψ, i.e., $1 \leq j \leq |\varphi|$ or $|\varphi|+2 \leq j \leq |\varphi|+|\psi|+1$. If the occurrence extends beyond the end of φ, in the first case, or the end of ψ, in the second case, this would imply that a suffix of a formula is a proper prefix of another formula, thus contradicting Lemma 4.3.18.

The other two cases are left to the reader. □

Theorem 4.3.24. *Let φ, ψ, α be formulas. If (ψ, i) is an occurrence of ψ in φ, then* `replace` $(\varphi, (\psi, i), \alpha)$ *is a formula.*

Proof. The argument is by induction on the definition of formulas. For the basis step, let φ be an atomic formula. By the Occurrence Theorem for First-Order Logic, $\psi = \varphi$, so $i = 0$ and `replace` $(\varphi, (\psi, 0), \alpha) = \alpha$, which is a formula.

The arguments for the inductive steps depend on the structure of φ. We discuss here the case when $\varphi = (Qx)\varphi_1$, where Q is a quantifier symbol and x is a variable. The remaining cases are similar to the cases considered in Theorem 2.2.15 for propositional logic and are left to the reader.

If $\psi = \varphi$, then $i = 0$ and the result is immediate. Otherwise, by the Occurrence Theorem, (ψ, i) is part of the occurrence $(\varphi_1, 4)$. We have

$$\texttt{replace}\,(\varphi, (\psi, i), \alpha) = (Qx)\texttt{replace}\,(\varphi_1, (\psi, i-4), \alpha).$$

By the inductive hypothesis $\texttt{replace}\,(\varphi_1, (\psi, i-4), \alpha)$ is a formula, so $\texttt{replace}\,(\varphi, (\psi, i), \alpha)$ is a formula. □

Let S be a set of symbols. For $k \in \mathbf{Z}$ define a bijection $T_k : S \times \mathbf{Z} \longrightarrow S \times \mathbf{Z}$ by $T_k(s, p) = (s, p + k)$. It is easy to see that $T_k^{-1} = T_{-k}$ for every $k \in \mathbf{Z}$.

Lemma 4.3.25. *Let φ, ψ be formulas, C be a binary connective symbol, x be a variable, and Q be a quantifier symbol. Then, for every variable or quantifier symbol s, we have*

$$\begin{aligned}
\mathtt{OCC}_s((\neg\varphi)) &= T_2(\mathtt{OCC}_s(\varphi))\\
\mathtt{OCC}_s((\varphi C \psi)) &= T_1(\mathtt{OCC}_s(\varphi)) \cup T_{2+|\varphi|}(\mathtt{OCC}_s(\psi))\\
\mathtt{OCC}_s((Qx)\varphi) &= T_4(\mathtt{OCC}_s(\varphi))\\
&\quad (\textit{if } s \notin \{x, Q\})\\
\mathtt{OCC}_x((Qx)\varphi) &= \{(x, 2)\} \cup T_4(\mathtt{OCC}_x(\varphi))\\
\mathtt{OCC}_Q((Qx)\varphi) &= \{(Q, 1)\} \cup T_4(\mathtt{OCC}_Q(\varphi)).
\end{aligned}$$

Proof. We prove only the second equality of the lemma. Let s be a variable or quantifier symbol and let $(s, i) \in \mathtt{OCC}_s((\varphi C \psi))$. We must have either $1 \leq i \leq |\varphi|$ or $|\varphi| + 2 \leq i \leq |\varphi| + |\psi| + 1$. In the first case, $s = \varphi(i-1)$, so $(s, i-1) \in \mathtt{OCC}_s(\varphi)$, which gives $(s, i) \in T_1(\mathtt{OCC}_s(\varphi))$. In the second, $s = \psi(i - 2 - |\varphi|)$, so $(s, i - 2 - |\varphi|) \in \mathtt{OCC}_s(\psi)$, which implies $(s, i) \in T_{2+|\varphi|}(\mathtt{OCC}_s(\psi))$.

Conversely, let $(s, j) \in T_1(\mathtt{OCC}_s(\varphi))$. We have $(s, j-1) \in \mathtt{OCC}_s(\varphi)$ and $1 \leq j \leq |\varphi|$. This gives $s = \varphi(j-1)$, so $(s, j) \in \mathtt{OCC}_s((\varphi C \psi))$. If $(s, j) \in T_{2+|\varphi|}(\mathtt{OCC}_s(\psi))$, then a similar argument shows that $(s, j) \in \mathtt{OCC}_s((\varphi C \psi))$. □

The next syntactic result will allow us to attach to each occurrence of a quantifier symbol in a formula φ a subformula of φ called the scope of the quantifier symbol occurrence.

Theorem 4.3.26. *Let φ be a formula of first-order logic. If Q is a quantifier symbol and $(Q, i) \in \mathtt{OCC}_Q(\varphi)$, then there is a unique formula α such that $(\alpha, i+3)$ is an occurrence of α in φ.*

Proof. Let $P(\varphi)$ be the property that for each occurrence of a quantifier symbol (Q, i) in φ there is an α with the desired properties. We shall prove by structural induction on formulas that $P(\varphi)$ is true for every formula $\varphi \in$ FORM. There is nothing to show for atomic formulas because they don't contain quantifier symbols.

Suppose that $P(\varphi)$ is true and let $\theta = (\neg\varphi)$. If (Q, i) is an occurrence of the quantifier symbol Q in θ, then $2 \leq i \leq |\varphi| + 1$. Therefore, $(Q, i') = (Q, i-2)$ is an occurrence of Q in φ. By the inductive hypothesis, there is an α such that $(\alpha, i'+3) = (\alpha, i+1)$ is an occurrence of α in φ. Since $(\varphi, 2) \in \mathtt{OCC}_\varphi(\theta)$, we have $(\alpha, i+3) \in \mathtt{OCC}_\alpha(\theta)$ by Theorem 1.2.6.

Assume now that $P(\varphi)$ and $P(\psi)$ are true and consider $\theta = (\varphi C \psi)$, where C is one of the binary connective symbols. Let (Q, i) be an occurrence of a quantifier symbol Q in α. We consider two cases depending on whether the occurrence is part of φ or part of ψ, that is, whether $1 \leq i \leq |\varphi|$ or $|\varphi| + 2 \leq i \leq |\varphi| + |\psi| + 1$.

Case 1. Let $i' = i - 1$. Then (Q, i') is an occurrence of Q in φ. By the inductive hypothesis, there is a formula α such that $(\alpha, i'+3) = (\alpha, i+2) \in \mathtt{OCC}_\alpha(\varphi)$. Then, $(\alpha, i+3) \in \mathtt{OCC}_\alpha(\theta)$.

Case 2. Let $i' = i - |\varphi| - 2$. Then (Q, i') is an occurrence of Q in ψ. Again, by the inductive hypothesis, there is a formula α such that $(\alpha, i'+3) = (\alpha, i - |\varphi| + 1) \in \mathtt{OCC}_\alpha(\psi)$. Then, $(\alpha, i - |\varphi| + 1 + 2 + |\varphi|) = (\alpha, i+3) \in \mathtt{OCC}_\varphi(\theta)$.

Finally, assume that $P(\varphi)$ is true and that $\theta = (\hat{Q}x)\varphi$ for some quantifier symbol $\hat{Q}$ and variable x. Let (Q, i) be an occurrence of a quantifier symbol in θ. We can have $i = 1$ or $4 \leq i \leq |\varphi| + 3$. In the first case, $Q = \hat{Q}$ and $\alpha = \varphi$ satisfies the conditions. In the second, (Q, i') is an occurrence of Q in φ, where $i' = i - 4$. By the inductive hypothesis, there is a formula α such that $(\alpha, i'+3) = (\alpha, i-1) \in \mathtt{OCC}_\alpha(\varphi)$. So, $(\alpha, i+3) \in \mathtt{OCC}_\alpha(\theta)$.

The uniqueness of α follows immediately from the fact that no formula can be a proper prefix of another formula (see Lemma 4.3.18). □

Definition 4.3.27. Let (Q, i) be an occurrence of a quantifier symbol Q in a formula φ and let $(\alpha, i+3)$ be the occurrence of a subformula α whose existence and uniqueness are guaranteed by Theorem 4.3.26.

The *scope* of the occurrence (Q, i) in φ is the occurrence $(\alpha, i+3)$ of α in φ, denoted by $\mathsf{scope}_{\varphi}((Q, i))$.

If $\varphi(i+1) = x$, then we say that (Q, i) is an *occurrence of the quantifier symbol Q using the variable x*.

If y is a variable and (y, j) is part of the scope of (Q, i), then we say that (y, j) is *in the scope of* (Q, i). □

Example 4.3.28. For the occurrence $(\forall, 14)$ of the quantifier symbol $\forall$ in the first-order formula

$$\varphi = (R(c, f(x, c)) \vee (\forall x) R'(g(x), y))$$

considered in Example 4.3.20, the scope is $(\alpha, 17)$, where α is the formula $R'(g(x), y)$; $(y, 24)$ is in the scope of $(\forall, 14)$. □

An occurrence of a variable x in a formula is bound if it immediately follows an occurrence of a quantifier symbol or it is located in the scope of an occurrence of a quantifier symbol using x. This is formalized in the following definition.

Definition 4.3.29. An occurrence (x, j) of a variable x in a formula φ is *bound* if there exists an occurrence of a quantifier symbol $(Q, i) \in \mathtt{OCC}_Q(\varphi)$ such that one of the following cases holds:

(1) $j = i + 1$, or
(2) $\varphi(i+1) = x$ and (x, j) is in the scope of (Q, i).

An occurrence of a variable in a formula is *free* if it is not bound.

The set of all bound occurrences of variables in a formula φ is denoted by $\mathtt{BO}(\varphi)$; the set of all free occurrences of variables in φ is denoted by $\mathtt{FO}(\varphi)$. Also, the set of all bound occurrences of a variable x in φ is denoted by $\mathtt{BO}_x(\varphi)$, while the set of free occurrences of x in φ is denoted by $\mathtt{FO}_x(\varphi)$.

Bound occurrences of variables which are next to quantifier occurrences (the first case above) are said to be *active*. Bound occurrences that are not active are called *passive*. We denote by $\texttt{ABO}(\varphi)$ and $\texttt{PBO}(\varphi)$, respectively, the sets of active and passive bound occurrences in φ. □

Clearly, $\texttt{BO}(\varphi) \cap \texttt{FO}(\varphi) = \emptyset$.

Definition 4.3.30. Let φ be a formula. The *set of bound variables of* φ is the set $\texttt{BV}(\varphi)$ that consists of those variables x such that $(x, j) \in \texttt{BO}(\varphi)$ for some $j \in \mathbf{N}$.

The *set of free variables of* φ is the set $\texttt{FV}(\varphi)$ that consists of all those variables x such that $(x, j) \in \texttt{FO}(\varphi)$ for some $j \in \mathbf{N}$.

If $\mathcal{L}$ is a first-order language, we denote by $\mathrm{FORM}_{\mathcal{L}}(V)$ the set of $\mathcal{L}$-formulas φ such that $\texttt{FV}(\varphi) \subseteq V$. The formulas in $\mathrm{FORM}_{\mathcal{L}}(V)$ will be referred to as *$(\mathcal{L}, V)$-formulas.* □

As usual, if Γ is a set of formulas, we denote by $\texttt{FV}(\Gamma)$ the set $\bigcup\{\texttt{FV}(\varphi) \mid \varphi \in \Gamma\}$, and similarly for $\texttt{BV}(\Gamma)$.

Note that a variable may occur both as a free and as a bound variable in a formula, that is, we may have $\texttt{FV}(\varphi) \cap \texttt{BV}(\varphi) \neq \emptyset$.

The set of all variables that occur in a formula φ is denoted by $\texttt{V}(\varphi)$. Clearly, $\texttt{V}(\varphi) = \texttt{BV}(\varphi) \cup \texttt{FV}(\varphi)$.

Example 4.3.31. Let φ be the formula $(\forall y)((\forall x)R(x, y) \wedge R'(x))$. Since the scope of the occurrence $(\forall, 6)$ is $(R(x, y), 9)$, the occurrences $(x, 7)$ and $(x, 11)$ are bound; $(x, 7)$ is an active bound occurrence, while $(x, 11)$ is a passive bound occurrence of x. Since there are no other occurrences of quantifier symbols using x, the occurrence $(x, 18)$ is free. The occurrence $(y, 2)$ is active bound because it follows immediately the occurrence $(\forall, 1)$ and $(y, 13)$ is passive bound because it is located in the scope of the occurrence $(\forall, 1)$. Thus $\texttt{BV}(\varphi) = \{x, y\}$ and $\texttt{FV}(\varphi) = \{x\}$. □

Example 4.3.32. Consider the formula

$$\psi = ((\exists x)R(x, y) \vee (\forall x)((\forall x)R'(x, y, y) \wedge R(x, y))).$$

There are three occurrences of quantifier symbols in ψ, namely, $(\exists, 2)$, $(\forall, 13)$, and $(\forall, 18)$. The scopes of these quantifier symbols are

$$(R(x, y), 5), (((\forall x)R'(x, y, y) \wedge R(x, y)), 16), (R'(x, y, y), 21),$$

respectively. For reasons shown in the following table, all occurrences of x are bound.

$(x, 3)$	follows $(\exists, 2)$	active
$(x, 7)$	located in the scope of $(\exists, 2)$	passive
$(x, 14)$	follows $(\forall, 13)$	active
$(x, 19)$	follows $(\forall, 18)$	active
$(x, 23)$	located in the scopes of $(\forall, 13)$ and $(\forall, 18)$	passive
$(x, 32)$	located in the scope of $(\forall, 13)$	passive

□

The passive occurrence $(x, 23)$ in the formula ψ of Example 4.3.32 is in the scope of two quantifier occurrences, $(\forall, 13)$ and $(\forall, 18)$. However, intuitively, this occurrence is linked to quantifier occurrence $(\forall, 18)$. Next, we formalize this intuition.

Definition 4.3.33. Let φ be a formula. The function $\mathsf{binding}_\varphi : \mathtt{PBO}(\varphi) \to \mathtt{ABO}(\varphi)$ is defined by $\mathsf{binding}_\varphi((x, j)) = (x, i)$, where

$$i = \max\{k \mid (x, k) \in \mathtt{ABO}(\varphi) \text{ and } (x, j) \text{ is in the scope of } (Q, k-1)\}.$$

If $\mathsf{binding}_\varphi((x, j)) = (x, i)$, we say that (x, j) is a passive bound occurrence of x associated to (x, i). □

Example 4.3.34. For the formula ψ of Example 4.3.32, we have

$$\mathsf{binding}_\psi((x, 7)) = (x, 3), \quad \mathsf{binding}_\psi((x, 23)) = (x, 19),$$
$$\mathsf{binding}_\psi((x, 32)) = (x, 14).$$

□

Definition 4.3.35. A *sentence* or a *closed formula* is a formula φ such that $\mathtt{FV}(\varphi) = \emptyset$. For a first-order language $\mathcal{L}$, the set of sentences that are $\mathcal{L}$-formulas is denoted by $\mathrm{SENT}_{\mathcal{L}}$.

The set of all sentences is denoted by SENT. □

Example 4.3.36. The formula $\varphi = (\forall y)((\forall x)R(x, y) \wedge R'(x))$ considered in Example 4.3.31 is not a sentence because x is a free variable. On the other hand, formula $\alpha = (\exists x)\varphi$ is a sentence. □

Theorem 4.3.37. *If φ is an atomic formula, then*

$$\begin{array}{ll} \mathtt{FO}(\varphi) = \bigcup_{x \in VAR} \mathtt{OCC}_x(\varphi), & \mathtt{BO}(\varphi) = \emptyset, \\ \mathtt{FV}(\varphi) = \{x \in VAR \mid \mathtt{OCC}_x(\varphi) \neq \emptyset\}, & \mathtt{BV}(\varphi) = \emptyset. \end{array}$$

Further, if φ is a propositional constant, then $\mathtt{FV}(\varphi) = \emptyset$. Also, if $\varphi = R(t_0, \ldots, t_{n-1})$, then $\mathtt{FV}(\varphi) = \bigcup\{\mathtt{V}(t_i) \mid 0 \leq i \leq n-1\}$.

Proof. We leave this simple argument to the reader. □

Theorem 4.3.38. *Let $\varphi = (\neg\psi)$, where ψ is a formula. If (Q, i) is an occurrence of a quantifier symbol Q in φ, then the scope of (Q, i) in φ is $(\alpha, i+3)$, where $(\alpha, i+1)$ is the scope of the occurrence $(Q, i-2)$ in ψ. Furthermore,*

$$\begin{array}{ll} \mathtt{FO}(\varphi) = T_2(\mathtt{FO}(\psi)), & \mathtt{BO}(\varphi) = T_2(\mathtt{BO}(\psi)), \\ \mathtt{FV}(\varphi) = \mathtt{FV}(\psi), & \mathtt{BV}(\varphi) = \mathtt{BV}(\psi). \end{array}$$

Proof. We leave this argument for the reader. □

Theorem 4.3.39. *Let $\varphi = (\alpha C \beta)$, where α and β are formulas and C is a binary connective symbol. Let (Q, i) be an occurrence of a quantifier symbol in φ. If $(Q, i) \in T_1(\mathtt{OCC}_Q(\alpha))$, then the scope of (Q, i) in φ is $(\gamma, i+3)$, where $(\gamma, i+2)$ is the scope of the occurrence $(Q, i-1)$ in α. If $(Q, i) \in T_{2+|\alpha|}(\mathtt{OCC}_Q(\beta))$, then the scope of (Q, i) in φ is $(\gamma, i+3)$ where $(\gamma, i+1-|\alpha|)$ is the scope of the occurrence $(Q, i-2-|\alpha|)$ in β. Furthermore,*

$$\begin{aligned} \mathtt{FO}(\varphi) &= T_1(\mathtt{FO}(\alpha)) \cup T_{2+|\alpha|}(\mathtt{FO}(\beta)), \\ \mathtt{BO}(\varphi) &= T_1(\mathtt{BO}(\alpha)) \cup T_{2+|\alpha|}(\mathtt{BO}(\beta)), \\ \mathtt{FV}(\varphi) &= \mathtt{FV}(\alpha) \cup \mathtt{FV}(\beta), \\ \mathtt{BV}(\varphi) &= \mathtt{BV}(\alpha) \cup \mathtt{BV}(\beta). \end{aligned}$$

Proof. We give this elementary but tedious proof in order to illustrate the technique needed to prove Theorems 4.3.37–4.3.40.

If Q is a quantifier symbol and $(Q, i) \in T_1(\mathtt{OCC}_Q(\alpha))$, then $(Q, i-1) \in \mathtt{OCC}_Q(\alpha)$ and therefore there is a subformula γ of α such that $(\gamma, i+2)$ is the scope of $(Q, i-1)$ in α. Clearly, $(\gamma, i+3)$ is an occurrence of γ in φ and, by Definition 4.3.27, $(\gamma, i+3)$ is the scope of the occurrence (Q, i). A similar argument works when $(Q, i) \in T_{2+|\alpha|}(\mathtt{OCC}_Q(\beta))$.

We prove the second of the last four equalities of the lemma; the rest are immediate consequences of this equality.

Let $(x, i) \in T_1(\mathtt{BO}(\alpha))$. We have $(x, i-1) \in \mathtt{BO}(\alpha)$. There is an occurrence of a quantifier symbol $(\tilde{Q}, j)$ in α such that $i - 1 = j + 1$ (that is, $i = j + 2$) or $(x, i-1)$ is in the scope of the occurrence $(\tilde{Q}, j)$ using x in α. Because we have the occurrence $(\tilde{Q}, j)$ in α, we have the occurrence $(\tilde{Q}, j+1)$ of $\tilde{Q}$ in φ. If $i = j + 2$, then it is clear that $(x, i) \in \mathtt{BO}(\varphi)$. Suppose now that $(x, i-1)$ is in the scope $(\gamma, j+3)$ of $(\tilde{Q}, j)$ in α. Since $(\gamma, j+4)$ is an occurrence of γ in φ it follows that $(x, i-2-|\alpha|)$ is located in the scope of the occurrence $(\tilde{Q}, j+1)$ in φ, so $(x, i) \in \mathtt{BO}(\varphi)$.

If $(x, i) \in T_{2+|\alpha|}(\mathtt{BO}(\beta))$, then $(x, i-2-|\alpha|) \in \mathtt{BO}(\beta)$. There is an occurrence (Q', k) of a quantifier symbol Q' in β such that $i-2-|\alpha| = k+1$ (that is, $i = k+3+|\alpha|$) or $(x, i-2-|\alpha|)$ is located in the scope $(\gamma, k+3)$ of (Q', k) in β. Note that $(Q', k+2+|\alpha|)$ is an occurrence of Q' in φ. Therefore, in the first case, (x, i) is the occurrence of x that follows immediately the occurrence $(Q', k+2+|\alpha|)$ in φ. In the second case, $(\gamma, k+5+|\alpha|)$ is an occurrence of γ in φ and (x, i) is located in this occurrence of γ. We conclude that

$$T_1(\mathtt{BO}(\alpha)) \cup T_{2+|\alpha|}(\mathtt{BO}(\beta)) \subseteq \mathtt{BO}(\varphi).$$

Conversely, let $(x, i) \in \mathtt{BO}(\varphi)$. There exists an occurrence (Q, h) of a quantifier symbol Q using x in φ such that $i = h + 1$ or (x, i) is located in the scope of (Q, h) in φ. By the second equality of Lemma 4.3.25 we have $(Q, h) \in T_1(\mathtt{OCC}_Q(\alpha))$ or $(Q, h) \in T_{2+|\alpha|}(\mathtt{OCC}_Q(\beta))$, which gives $(Q, h-1) \in \mathtt{OCC}_Q(\alpha)$ or $(Q, h-2-|\alpha|) \in \mathtt{OCC}_Q(\beta)$, respectively. Four cases need to be considered:

(1) $i = h + 1$ and $(Q, h-1) \in \mathtt{OCC}_Q(\alpha)$;
(2) (x, i) is located in the scope $(\gamma, h+3)$ of the occurrence of (Q, h) in φ and $(Q, h-1) \in \mathtt{OCC}_Q(\alpha)$;
(3) $i = h + 1$ and $(Q, h-2-|\alpha|) \in \mathtt{OCC}_Q(\beta)$, and
(4) (x, i) is located in the scope of (Q, h) in φ and $(Q, h-2-|\alpha|) \in \mathtt{OCC}_Q(\beta)$.

In the first case $(x, i-1) \in \mathtt{OCC}_x(\alpha)$ and this occurrence of x is the one which follows immediately the occurrence $(Q, h-1)$ in α. Therefore, $(x, i-1) \in \mathtt{BO}(\alpha)$, so $(x, i) \in T_1(\mathtt{BO}(\alpha))$.

In the second case the scope of the occurrence $(Q, h-1)$ in α is $(\gamma, h+2)$ and $(x, i-1)$ is located in this scope. Again, this implies $(x, i) \in T_1(\mathtt{BO}(\alpha))$.

Both the third and the fourth cases imply $(x, i) \in T_{2+|\alpha|}(\mathtt{BO}(\beta))$ as the reader can verify following a similar argument. Therefore,

$$\mathtt{BO}(\varphi) \subseteq T_1(\mathtt{BO}(\alpha)) \cup T_{2+|\alpha|}(\mathtt{BO}(\beta))$$

and this concludes the proof of the second of the last group of equalities. □

Theorem 4.3.40. *Let $\varphi = (Q'x)\psi$, where Q' is a quantifier symbol, x is a variable and ψ is a formula. If (Q, i) is an occurrence of a quantifier symbol Q in φ, then the scope of (Q, i) in φ is $(\alpha, i+3)$, where $(\alpha, i-1)$ is the scope of the occurrence $(Q, i-4)$ in ψ, unless $(Q, i) = (Q', 1)$, in which case the scope is $(\psi, 4)$. Furthermore,*

$$\begin{aligned}
&\mathtt{FO}(\varphi) = T_4(\mathtt{FO}(\psi) - \{(x, j) | (x, j) \in \mathtt{FO}(\psi)\}),\\
&\mathtt{BO}(\varphi) = T_4(\mathtt{BO}(\psi)) \cup \{(x, 2)\} \cup T_4(\{(x, j) | (x, j) \in \mathtt{FO}(\psi)\}),\\
&\mathtt{FV}(\varphi) = \mathtt{FV}(\psi) - \{x\},\\
&\mathtt{BV}(\varphi) = \mathtt{BV}(\psi) \cup \{x\}.
\end{aligned}$$

Proof. The argument is left to the reader. □

Theorem 4.3.41. *Let α, β be two formulas such that $\mathtt{FV}(\alpha) = \mathtt{FV}(\beta)$. Then, if (α, i) is an occurrence of α in φ, then $\mathtt{FV}(\mathtt{replace}\,(\varphi, (\alpha, i), \beta)) = \mathtt{FV}(\varphi)$.*

Proof. The argument is by induction on φ and is left to the reader. □

Definition 4.3.42. Let φ be a first-order formula whose set of free variables is $\{x_{i_0}, \ldots, x_{i_{n-1}}\}$ where $i_0 < \cdots < i_{n-1}$.

The *universal closure* of φ is the generalization $\varphi^\forall$ of φ given by:

$$(\forall x_{i_0}) \cdots (\forall x_{i_{n-1}})\varphi.$$

If Γ is a set of first-order formulas, we use the notation $\Gamma^\forall$ for the set $\{\varphi^\forall \mid \varphi \in \Gamma\}$ and we refer to this set as the *universal closure of* Γ.

The *existential closure* of φ is the formula $\varphi^\exists$ given by:

$$(\exists x_{i_0}) \cdots (\exists x_{i_{n-1}})\varphi.$$

If Γ is a set of first-order formulas, we use the notation $\Gamma^{\exists}$ for the set $\{\varphi^{\exists} \mid \varphi \in \Gamma\}$ and we refer to this set as the *existential closure of* Γ. ▯

Note that if φ is a closed formula, then $\varphi^{\forall} = \varphi^{\exists} = \varphi$. Repeated application of Theorem 4.3.40, shows that both the universal closure and the existential closure of a formula are closed formulas.

Theorem 4.3.43. *We have* $((\Gamma^{\forall})^{-\forall})^{\forall} = \Gamma^{\forall}$ *if* Γ *is a set of quantifier-free formulas.*

Proof. We leave to the reader the proof of this simple result. □

Next, we develop a classification of occurrences of atomic formulas in first-order formulas that do not contain the $\leftrightarrow$ connective symbol. This classification is needed in Chapter 5. Using induction on φ, we define positive and negative occurrences of atomic formulas in φ.

Definition 4.3.44. Let φ be a formula that does not contain the connective symbol $\leftrightarrow$, and let $\zeta = (\alpha, i)$ be an occurrence of an atomic formula α in φ.

(1) If φ is atomic, then $\alpha = \varphi$ and $i = 0$ and we say that ζ is a positive occurrence in φ.
(2) If $\varphi = (\neg\psi)$, then $\zeta' = (\varphi, i - 2)$ is an occurrence of α in ψ and we say that ζ is a positive (negative) occurrence in φ if ζ' is a negative (positive) occurrence in ψ.
(3) If $\varphi = (\psi_0 C \psi_1)$, where $C \in \{\vee, \wedge, \rightarrow\}$, then either ζ is part of $(\psi_0, 1)$ and $\zeta' = (\varphi, i - 1)$ is an occurrence in ψ_0, or ζ is part of $(\psi_1, |\psi_0| + 2)$ and $\zeta' = (\alpha, i - |\psi_0| - 2)$ is an occurrence in ψ_1. Two cases may occur.
    (a) If $C \in \{\vee, \wedge\}$, ζ has the same sign (positive or negative) as ζ' in the corresponding subformula of φ.
    (b) If C is $\rightarrow$, ζ has the same sign as ζ' if ζ' is an occurrence in ψ_1 and the opposite sign if ζ' is an occurrence in ψ_0.
(4) If $\varphi = (Qx)\psi$ and $\zeta' = (\alpha, i - 4)$ is an occurrence in ψ, then ζ has the same sign in φ as ζ' has in ψ.

▯

Example 4.3.45. Let $\varphi = (\forall x)(\neg(P(a,b) \rightarrow R(x)))$. Note that $(P(a,b), 7)$ is a positive occurrence in φ, while $(R(x), 14)$ is a negative occurrence in the same formula. ▯

4.3.4 *Signed Formulas*

As we did for propositional logic, we introduce first-order signed formulas and for similar reasons: to facilitate the presentation of certain formal systems of first-order logic.

Definition 4.3.46. A *signed formula* is a pair $(b, \varphi) \in \mathbf{Bool} \times \text{FORM}$. A signed formula (b, φ) will be denoted by $b\varphi$.

The *size* $\texttt{size}(b\varphi)$ of the signed formula $b\varphi$ is $\texttt{size}(\varphi)$.

The *norm* of $b\varphi$ is $\|\varphi\|$ and will be denoted by $\|b\varphi\|$. □

The set of signed formulas $\mathbf{Bool} \times \text{FORM}$ will be denoted by SFORM; similarly, the set $\mathbf{Bool} \times \text{FORM}_{\mathcal{L}}(V)$ will be denoted by $\text{SFORM}_{\mathcal{L}}(V)$, when $\mathcal{L}$ is a first-order language and V is a set of variables. The *standard order* of signed formulas is

$$\mathbf{T}\varphi_0, \mathbf{F}\varphi_0, \mathbf{T}\varphi_1, \mathbf{F}\varphi_1, \ldots,$$

where $\varphi_0, \varphi_1, \ldots$ is the standard order on FORM.

Signed formulas of the forms $\mathbf{T}(\forall x)\varphi$ and $\mathbf{F}(\exists x)\varphi$ will be called $\boldsymbol{\gamma}$*-signed formulas.* Signed formulas of the forms $\mathbf{T}(\exists x)\varphi$ and $\mathbf{F}(\forall x)\varphi$ will be referred to as $\boldsymbol{\delta}$*-formulas.*

If $b\varphi$ is a signed formula, then $\texttt{FV}(b\varphi)$ is the set $\texttt{FV}(\varphi)$. As usual, if Δ is a set of signed formulas, then $\texttt{FV}(\Delta) = \bigcup\{\texttt{FV}(b\varphi) \mid b\varphi \in \Delta\}$. Similarly, $\texttt{BV}(b\varphi) = \texttt{BV}(\varphi)$ and $\texttt{V}(b\varphi) = \texttt{V}(\varphi)$. These notations are extended to sets of signed formulas by defining $\texttt{BV}(\Delta) = \bigcup\{\texttt{BV}(b\varphi) \mid b\varphi \in \Delta\}$ and $\texttt{V}(\Delta) = \bigcup\{\texttt{V}(b\varphi) \mid b\varphi \in \Delta\}$.

Example 4.3.47. Following the above convention, we denote the signed formula $(\mathbf{T}, (\exists x_0)R_3^1(x_0))$ by $\mathbf{T}(\exists x_0)R_3^1(x_0)$. □

Definition 4.3.48. A set Δ of signed formulas is *closed* if there is a formula φ such that both $\mathbf{T}\varphi$ and $\mathbf{F}\varphi$ belong to Δ. □

Note that the term "closed" is overloaded because we already used this term in Definition 4.3.35 in reference to formulas that contain no free variables. The context will make it clear which meaning is applicable.

4.3.5 *Substitutions and Formulas*

In this subsection, we examine the effects of several types of substitutions on formulas; namely, replacing one relation symbol by another relation symbol with the same arity, replacing free occurrences of variables with terms, and replacing constant symbols with terms. The application of substitutions is extended to signed formulas by defining $s(b\varphi)$ as $bs(\varphi)$ for any substitution s and signed formula $b\varphi$.

Theorem 4.3.49. *Let φ be a formula and let R, R' be relation symbols of the same arity. Then, the sequence $\mathsf{s}^R_{R'}(\varphi)$ obtained by substituting R' for R in φ is a formula.*

Proof. The argument is by induction on formulas. For the basis step, let us first observe that $\mathsf{s}^R_{R'}(t) = t$ for every term t. Let P be an m-ary relation symbol with $m > 0$ and $t_0, \ldots, t_{m-1}$ be terms. For the atomic formula $\varphi = P(t_0, \ldots, t_{m-1})$, we have

$$\mathsf{s}^R_{R'}(\varphi) = \begin{cases} \varphi & \text{if } P \neq R \\ R'(t_0, \ldots, t_{n-1}) & \text{if } P = R, \end{cases}$$

which shows that $\mathsf{s}^R_{R'}(\varphi)$ is a formula. If P is a propositional constant, and $\varphi = P$, then $\mathsf{s}^R_{R'}(\varphi)$ is either P or R' depending whether $P = R$ or not. In either case, $\mathsf{s}^R_{R'}(\varphi)$ is a formula.

We leave to the reader the inductive steps. □

For the remainder of this section, we refer to $(S_{\text{FOL}}, \text{VAR})$-substitutions simply as substitutions; further, if $\mathcal{L}$ is a first-order language, $(S_{\mathcal{L}}, \text{VAR})$-substitutions are referred to as *$\mathcal{L}$-substitutions.*

Definition 4.3.50. Let φ be a formula and let $\{(y_0, i_0), \ldots, (y_{\ell-1}, i_{\ell-1})\}$ be the set $\mathtt{FO}(\varphi)$ of all free occurrences of variables in φ, where $i_0 < \cdots < i_{\ell-1}$, so that

$$\varphi = u_0 y_0 u_1 y_1 \ldots y_{\ell-1} u_\ell,$$

where

$$\begin{aligned} u_0 &= \varphi(0) \ldots \varphi(i_0 - 1) \\ u_1 &= \varphi(i_0 + 1) \ldots \varphi(i_1 - 1) \\ &\vdots \\ u_{\ell-1} &= \varphi(i_{\ell-2} + 1) \ldots \varphi(i_{\ell-1} - 1) \\ u_\ell &= \varphi(i_{\ell-1} + 1) \ldots \varphi(|\varphi| - 1) \end{aligned}$$

Let s be a substitution. The *result of applying s to the free occurrences of variables in φ* is the sequence

$$\text{FVSubst}(s, \varphi) = u_0 s(y_0) u_1 s(y_1) \cdots s(y_{\ell-1}) u_\ell.$$

For a signed formula $b\varphi$, we write $\text{FVSubst}(s, b\varphi) = b\text{FVSubst}(s, \varphi)$. ▯

If $y_0, \ldots, y_{n-1}$ are distinct variables and $t_0, \ldots, t_{n-1}$ are terms, we denote $\text{FVSubst}(\mathsf{s}^{y_0 \cdots y_{n-1}}_{t_0 \cdots t_{n-1}}, \varphi)$ by $(\varphi)_{y_0,\ldots,y_{n-1}:=t_0,\ldots,t_{n-1}}$. In particular, for $n = 1$, we use the notation $(\varphi)_{x:=t}$.

We have adopted the notation $(\varphi)_{x:=t}$ instead of $\varphi(x := t)$ in order to avoid ambiguous expressions such as $(\forall x)\varphi(x := t)$ created by the fact, mentioned in Exercise 10(b), that a proper suffix of a formula can be a formula. Further, the formula

$$(\cdots(((\varphi)_{y_0:=t_0})_{y_1:=t_1})\cdots)_{y_{n-1}:=t_{n-1}}$$

will be denoted by $(\varphi)_{y_0:=t_0,\ldots,y_{n-1}:=t_{n-1}}$ for $n \geq 1$ (where the y_is need not be distinct).

Theorem 4.3.51. *Let s, s' be substitutions and φ be a formula such that $s(x) = s'(x)$ for every $x \in \mathtt{FV}(\varphi)$. Then,* $\text{FVSubst}(s, \varphi) = \text{FVSubst}(s', \varphi)$.

Proof. This statement follows immediately from Definition 4.3.50. □

Lemma 4.3.52. *Let s be a substitution. If φ is an atomic formula, then $s(\varphi)$ is an atomic formula. Further, if φ is an atomic $\mathcal{L}$-formula and s is an $\mathcal{L}$-substitution, then $s(\varphi)$ is an atomic $\mathcal{L}$-formula.*

Proof. If φ is a propositional constant, then $s(\varphi) = \varphi$. If φ is the atomic formula $R(u_0, \ldots, u_{m-1})$, where R is an m-ary relation symbol, then

$$s(\varphi) = R(s(u_0), \ldots, s(u_{m-1}))$$

and this is an atomic formula because of Theorem 1.5.20. The second part follows immediately from the same theorem. □

If f is a function, then $[a \to b]f$ is the function obtained from f by removing the pair, if any, whose first component is a and adding the pair (a, b).

Lemma 4.3.53. *Let φ and ψ be formulas, s be a substitution, C be a binary connective symbol and Q be a quantifier symbol. Then,*

$$\mathrm{FVSubst}(s, \varphi) = s(\varphi),$$

if φ is atomic. Also,

$$\begin{aligned}\mathrm{FVSubst}(s, (\neg\varphi)) &= (\neg\mathrm{FVSubst}(s, \varphi))\\ \mathrm{FVSubst}(s, (\varphi C \psi)) &= (\mathrm{FVSubst}(s, \varphi)\ C\ \mathrm{FVSubst}(s, \psi))\\ \mathrm{FVSubst}(s, (Qy)\varphi) &= (Qy)\mathrm{FVSubst}([y \to y]s, \varphi).\end{aligned}$$

Proof. This follows immediately from Theorems 4.3.37–4.3.40. (Note that in the last case, we apply the substitution $[y \to y]s$ instead of s, because this guarantees that the free occurrences of y in φ will be unaffected by the substitution.) □

The next lemma contains a special case of Lemma 4.3.53.

Lemma 4.3.54. *Let φ and ψ be formulas, $y_0, \ldots, y_{n-1}$ be distinct variables, $t_0, \ldots, t_{n-1}$ be terms, C be a binary connective and Q be a quantifier symbol. Then,*

$$(\varphi)_{y_0,\ldots,y_{n-1}:=t_0,\ldots,t_{n-1}} = s^{y_0\cdots y_{n-1}}_{t_0\cdots t_{n-1}}(\varphi),$$

if φ is atomic. Also,

$$((\neg\varphi))_{y_0,\ldots,y_{n-1}:=t_0,\ldots,t_{n-1}} = (\neg(\varphi)_{y_0,\ldots,y_{n-1}:=t_0,\ldots,t_{n-1}}),$$

$$\begin{aligned}&((\varphi C \psi))_{y_0,\ldots,y_{n-1}:=t_0,\ldots,t_{n-1}}\\ &\quad = ((\varphi)_{y_0,\ldots,y_{n-1}:=t_0,\ldots,t_{n-1}} C (\psi)_{y_0,\ldots,y_{n-1}:=t_0,\ldots,t_{n-1}})\end{aligned}$$

and

$$\begin{aligned}&((Qy)\varphi)_{y_0,\ldots,y_{n-1}:=t_0,\ldots,t_{n-1}}\\ &= \begin{cases}(Qy)(\varphi)_{y_0,\ldots,y_{n-1}:=t_0,\ldots,t_{n-1}} & \text{if } y \notin \{y_0, \ldots, y_{n-1}\}\\ (Qy)(\varphi)_{y_0,\ldots,y_{i-1},y_{i+1},\ldots,y_{n-1}:=t_0,\ldots,t_{i-1},t_{i+1},\ldots,t_{n-1}} & \text{if } y = y_i.\end{cases}\end{aligned}$$

Proof. The statement follows immediately from Lemma 4.3.53 by taking $s = \mathsf{s}^{y_0 \cdots y_{n-1}}_{t_0 \cdots t_{n-1}}$. □

Theorem 4.3.55. *Let φ be a formula and s be a substitution. The sequence* $\mathrm{FVSubst}(s, \varphi)$ *is a formula. Moreover, if φ is an $\mathcal{L}$-formula and s is an $\mathcal{L}$-substitution, then* $\mathrm{FVSubst}(s, \varphi)$ *is an $\mathcal{L}$-formula.*

Proof. The theorem follows immediately from Lemmas 4.3.52 and 4.3.53 by structural induction on φ. □

Corollary 4.3.56. *Let φ be a formula, $y_0, \ldots, y_{n-1}$ be distinct variables and $t_0, \ldots, t_{n-1}$ be terms. Then, $(\varphi)_{y_0,\ldots,y_{n-1}:=t_0,\ldots,t_{n-1}}$ is a formula. In addition, $(\varphi)_{y_0,\ldots,y_{n-1}:=t_0,\ldots,t_{n-1}}$ is an $\mathcal{L}$-formula when φ is an $\mathcal{L}$-formula and $t_0, \ldots, t_{n-1} \in \mathrm{TERM}_{\mathcal{L}}$.*

Proof. This follows immediately from Theorem 4.3.55. □

A notion which will be useful in Section 4.10 is introduced below.

Definition 4.3.57. Let $\psi = (\forall y_0) \cdots (\forall y_{n-1})\varphi$ be a universal formula with φ quantifier-free.

A formula of the form $\theta = (\varphi)_{y_0,\ldots,y_{n-1}:=t_0,\ldots,t_{n-1}}$ is an *instance* of ψ, where $t_0, \ldots, t_{n-1}$ are terms. If $t_0, \ldots, t_{n-1} \in \mathrm{TERM}_{\mathcal{L}}$, then θ is called an *$\mathcal{L}$-instance of ψ*, where $\mathcal{L}$ is a first-order language. More specifically, if $t_0, \ldots, t_{n-1}$ are $(\mathcal{L}, V)$-terms, where $V \subseteq \mathrm{VAR}$, then we refer to θ as an *$(\mathcal{L}, V)$-instance of ψ*.

If Γ is a set of universal formulas, we denote by $\mathrm{INST}_{\mathcal{L},V}(\Gamma)$ the set of all $(\mathcal{L}, V)$-instances of the formulas of Γ. When $V = \emptyset$, we denote $\mathrm{INST}_{\mathcal{L},\emptyset}(\Gamma)$ by $\mathrm{GINST}_{\mathcal{L}}(\Gamma)$ and we refer to its members as *ground instances* of Γ. When $V = \mathrm{VAR}$, we denote the set $\mathrm{INST}_{\mathcal{L},\mathrm{VAR}}(\Gamma)$ simply by $\mathrm{INST}_{\mathcal{L},}(\Gamma)$. □

Definition 4.3.58. Let $\mathcal{L}$ be a first-order language and φ be a quantifier free $\mathcal{L}$-formula. An *$\mathcal{L}$-instantiation via an $\mathcal{L}$-substitution s* of φ is a formula φ' of the form $s(\varphi)$. □

Note that an $\mathcal{L}$-instantiation of a quantifier-free $\mathcal{L}$-formula is the same thing as an $\mathcal{L}$-instance of $\varphi^{\forall}$, the universal closure of φ.

Theorem 4.3.59. *Let $\mathcal{L}$ be a first-order language, φ be a quantifier-free $\mathcal{L}$-formula and let s, s' be two $\mathcal{L}$-substitutions. If $s(\varphi) = s'(\varphi)$, then we have $s(x) = s'(x)$ for all $x \in VAR(\varphi)$.*

Proof. Let $\varphi = q_0 x_{i_0} q_1 x_{i_1} \cdots q_{n-1} x_{i_{n-1}} q_n$, where $x_{i_0}, \ldots, x_{i_{n-1}}$ are variables and the sequences $q_0, \ldots, q_{n-1}$ contain no variables. Then,

$$s(\varphi) = q_0 s(x_{i_0}) q_1 s(x_{i_1}) \cdots q_{n-1} s(x_{i_{n-1}}) q_n$$
$$s'(\varphi) = q_0 s'(x_{i_0}) q_1 s'(x_{i_1}) \cdots q_{n-1} s'(x_{i_{n-1}}) q_n.$$

Since one term cannot be a proper prefix of another term, we have $s(x_{i_0}) = s'(x_{i_0})$. Then, for the same reason, $s(x_{i_1}) = s'(x_{i_1})$, etc. □

The above theorem shows that if φ' is an instantiation of φ, then the substitution s such that $\varphi' = s(\varphi)$ is uniquely determined on the variables that occur in φ.

Definition 4.3.60. Let $\mathcal{L}$ be a first-order language, φ be a quantifier-free $\mathcal{L}$-formula and φ' be an $\mathcal{L}$-instantiation of φ via the $\mathcal{L}$-substitution s. An occurrence of an term (t', i') in φ' is *visible* in φ if there is an occurrence of a term (t, i) in φ such that if we write $\varphi = q_0 t q_1$ with $|q| = i$ and $\varphi' = q_0' t' q_1'$ with $|q_0'| = i'$, we have

$$s(q_0) = q_0', s(t) = t', \quad \text{and} \quad s(q_1) = q_1'.$$

□

Observe that in the previous definition $s(t) = t'$ and $s(q_1) = q_1'$ follow from $s(q_0) = q_0'$ because no term can be a proper prefix of another term.

Example 4.3.61. Let f be a unary function symbol, R be a unary relation symbol, and a be a constant symbol. The occurrence $(f(a), 2)$ in the formula $\varphi' = R(f(a))$ is visible in $\varphi = R(x)$. Note that the occurrence $(a, 4)$ in φ' is not visible in φ. □

Theorem 4.3.62. *Let $\mathcal{L}$ be a first-order language, φ be a quantifier-free $\mathcal{L}$-formula and let φ' be an instantiation of φ via $s_0 * s_1$, where s_0 and s_1 are $\mathcal{L}$-substitutions, that is $\varphi' = s_0 * s_1(\varphi)$. If an occurrence of a term (t', i') in φ' is visible in φ, then this occurrence is visible in $s_1(\varphi)$.*

Proof. Note that $\varphi' = s_0 * s_1(\varphi) = s_0(s_1(\varphi))$, so φ' is an instantiation of $s_1(\varphi)$. If an occurrence (t', i') in φ' is visible in φ, then there is an occurrence of a term (t, i) in φ such that if we write $\varphi = q_0 t q_1$ with $|q_0| = i$ and $\varphi' = q_0' t' q_1'$ with $|q_0'| = i'$, we have $s_0 * s_1(q_0) = q_0'$,

$s_0 * s_1(t) = t'$, and $s_0 * s_1(q_1) = q_1'$. Since $s_1(\varphi) = s_1(q_1)s_1(t)s_1(q_1)$, the pair $(s_1(t), |s_1(q_0)|)$ is an occurrence of a term $s_1(t)$ is $s_1(\varphi)$ and we have $s_0(s_1(q_0)) = s_0 * s_1(q_0) = q_0'$, so (t', i') is visible in $s_1(\varphi)$. □

Lemma 4.3.63. *Let $\mathcal{L}$ be a first-order language, φ be a quantifier-free $\mathcal{L}$-formula and φ' be an $\mathcal{L}$-instantiation of φ via an $\mathcal{L}$-substitution s. Then, an occurrence (t', i') of a term in φ' is visible in φ if and only if there is a prefix q_0 of φ such that $|s(q_0)| = i'$.*

Proof. If (t', i') is visible in φ, then the desired q_0 exists by definition.

Conversely, suppose that q_0 is a prefix of φ and $|s(q_0)| = i'$. Since $s(q_0)$ is not the entire formula $\varphi' = s(\varphi)$, q_0 must be a proper prefix of φ. If the symbol following q_0 in φ is "(", ")", ",", a relation symbol or a connective symbol, then t' would start with this symbol, which is impossible, so the symbol following q_0 in φ must be a variable or a function symbol, which means that there is a term immediately following q_0 in φ, say $\varphi = q_0tq_1$. If we write $\varphi' = q_0't'q_1'$ with $|q_0'| = i'$, then $|s(q_0)| = i'$ implies $s(q_0) = q_0'$ which, as noted previously, implies that (t', i') is visible in φ. □

Theorem 4.3.64. *Let $\mathcal{L}$ be a first-order language, φ be a quantifier-free $\mathcal{L}$-formula, s be an $\mathcal{L}$-substitution, and let (t', i') be an occurrence of a term in the formula $\varphi' = s(\varphi)$. If we write φ as*

$$q_0y_0q_1y_1 \cdots q_{n-1}y_{n-1}q_n,$$

where $n \geq 0$, $y_0, \ldots, y_{n-1}$ are variables (not necessarily distinct), $q_0, \ldots, q_n$ do not contain variables, and $s(y_j) = u_j$ for $0 \leq j \leq n-1$ (so $\varphi' = q_0u_0q_1u_1 \cdots q_{n-1}u_{n-1}q_n$), then the occurrence (t', i') is visible in φ if and only if either (t', i') starts in one of the q_j sequences or is the occurrence (u_j, i_j) corresponding to one of the y_js.

Proof. Suppose that (t', i') does not start in one of the q_js and is not one of the (u_j, i_j)s. Then, (t', i') starts in one of the (u_j, i_j)s. If t' starts with the first symbol of u_j, then since no term can be a proper prefix of another term, we have $(t', i') = (u_j, i_j)$, contradicting the assumption, so (t', i') starts properly inside u_j. But then, $|s(q_0y_0q_1 \cdots q_j)| < i'$ and $|s(q_0y_0q_1 \cdots q_jy_j)| > i'$, so there is not prefix q of φ with $|s(q)| = i'$. By Lemma 4.3.63, (t', i') is not visible in φ.

Conversely, suppose that (t', i') starts in q_j, say $\hat{q}$ is the prefix of q_j preceeding the start of t'. Then, $|s(q_0 y_0 \cdots q_{j-1} y_{j-1} \hat{q})| = i'$, and by the same lemma, (t', i') is visible in φ.

If $(t', i') = (u_j, i_j)$, then $|s(q_0 y_0 \cdots q_j)| = i'$ and again by the same lemma, (t', i') is visible in φ. □

Next, we present a result involving substituting one relation symbol for another.

Theorem 4.3.65. *Let R and R' be n-ary relation symbols, s be a substitution and φ be a formula. Then,*

$$\mathsf{s}^{R}_{R'}(\text{FVSubst}(s, \varphi)) = \text{FVSubst}(s, \mathsf{s}^{R}_{R'}(\varphi)).$$

Proof. We shall prove by induction on the formula φ that the equality of the theorem holds for every substitution s. For the basis step, assume that φ is atomic. Then, we have:

$$\begin{aligned}
&\mathsf{s}^{R}_{R'}(\text{FVSubst}(s, \varphi))\\
&\quad = \mathsf{s}^{R}_{R'}(s(\varphi))\\
&\qquad \text{(by the first equality of Lemma 4.3.53)}\\
&\quad = s^{R}_{R'} * s(\varphi)\\
&\quad = s * s^{R}_{R'}(\varphi)\\
&\qquad \text{(by Theorem 1.2.23)}\\
&\quad = s(s^{R}_{R'}(\varphi))\\
&\quad = \text{FVSubst}(s, \mathsf{s}^{R}_{R'}(\varphi))\\
&\qquad \text{(by Lemma 4.3.53, since } \mathsf{s}^{R}_{R'}(\varphi) \text{ is atomic).}
\end{aligned}$$

We discuss only one of the inductive steps, namely, when $\varphi = (Qy)\psi$ and the result holds for ψ. In this case, we obtain

$$\begin{aligned}
&\mathsf{s}^{R}_{R'}(\text{FVSubst}(s, \varphi))\\
&\quad = \mathsf{s}^{R}_{R'}(\text{FVSubst}(s, (Qy)\psi))\\
&\quad = \mathsf{s}^{R}_{R'}((Qy)\text{FVSubst}([y \to y]s, \psi))\\
&\qquad \text{(by the last equality of Lemma 4.3.53)}
\end{aligned}$$

$$
\begin{aligned}
&= (Qy)\mathsf{s}^R_{R'}(\mathrm{FVSubst}([y \to y]s, \psi))\\
&= (Qy)\mathrm{FVSubst}([y \to y]s, \mathsf{s}^R_{R'}(\psi))\\
&\quad \text{(by inductive hypothesis)}\\
&= \mathrm{FVSubst}(s, (Qy)\mathsf{s}^R_{R'}(\psi))\\
&\quad \text{(by the last equality of Lemma 4.3.53)}\\
&= \mathrm{FVSubst}(s, \mathsf{s}^R_{R'}((Qy)\psi))\\
&= \mathrm{FVSubst}(s, \mathsf{s}^R_{R'}(\varphi)).
\end{aligned}
$$

□

Corollary 4.3.66. *If φ is a formula, $y_0, \ldots, y_{n-1}$ are distinct variables, and $t_0, \ldots, t_{n-1}$ are terms, then*

$$
\begin{aligned}
&\mathsf{s}^R_{R'}((\varphi)_{y_0,\ldots,y_{n-1}:=t_0,\ldots,t_{n-1}})\\
&\qquad = (\mathsf{s}^R_{R'}(\varphi))_{y_0,\ldots,y_{n-1}:=t_0,\ldots,t_{n-1}}.
\end{aligned}
$$

Proof. The corollary follows immediately from Theorem 4.3.65 by taking $s = \mathsf{s}^{y_0\cdots y_{n-1}}_{t_0\cdots t_{n-1}}$. □

The formulation of the next corollary requires the following observation. If Γ is a set of universal formulas and R, R' are relation symbols of the same arity, then $\mathsf{s}^R_{R'}(\Gamma)$ is also a set of universal formulas.

Corollary 4.3.67. *Let $\mathcal{L}$ be a first-order language, R be an n-ary relation symbol in $\mathcal{L}$ and R' be an n-ary relation symbol such that $R' \notin \mathcal{L}$. Define $\mathcal{L}' = (\mathcal{L} - \{R\}) \cup \{R'\}$. If Γ is a set of universal $\mathcal{L}$-formulas and $\mathtt{FV}(\Gamma) \subseteq V$, then*

$$\mathsf{s}^R_{R'}(\mathrm{INST}_{\mathcal{L},V}(\Gamma)) = \mathrm{INST}_{\mathcal{L}',V}(\mathsf{s}^R_{R'}(\Gamma)).$$

Proof. The statement follows from Corollary 4.3.66. □

Now we investigate replacing constant symbols in terms and formulas with terms.

Theorem 4.3.68. *Let φ be a formula, u, t be terms and a be a constant symbol. Then, $\mathsf{s}^a_t(u)$ is a term and $\mathsf{s}^a_t(\varphi)$ is a formula.*

Proof. The argument is by induction on terms and formulas, respectively, and is left to the reader. □

Theorem 4.3.69. *Let φ be a formula, u, t be terms and a be a constant symbol. We have*:

$$\mathtt{V}(u) \subseteq \mathtt{V}(\mathsf{s}^a_t(u)) \subseteq \mathtt{V}(u) \cup \mathtt{V}(t)$$
$$\mathtt{FV}(\varphi) \subseteq \mathtt{FV}(\mathsf{s}^a_t(\varphi)) \subseteq \mathtt{FV}(\varphi) \cup \mathtt{V}(t).$$

Proof. The argument for the first statement is by induction on the term u; the argument for the second statement is by induction on φ. Both are left for the reader. $\square$

Definition 4.3.70. Let s be a substitution, a be a constant symbol and let t be a term. The substitution ${}^a_t s$ is given by ${}^a_t s(x) = \mathsf{s}^a_t(s(x))$ for every variable x. $\square$

Lemma 4.3.71. *Let t be a term, a be a constant symbol and let y be a variable. We have ${}^a_t([y \to y]s) = [y \to y]{}^a_t s$ for all substitutions s.*

Proof. We demonstrate the equality of the two substitutions by verifying that they are equal on every variable. Suppose initially that z is a variable such that $z \neq y$. We have

$$\begin{aligned}{}^a_t([y \to y]s)(z) &= \mathsf{s}^a_t([y \to y]s(z))\\ &= \mathsf{s}^a_t(s(z))\\ &= {}^a_t s(z) = [y \to y]{}^a_t s(z).\end{aligned}$$

If $z = y$, then

$$\begin{aligned}{}^a_t([y \to y]s)(y) &= \mathsf{s}^a_t([y \to y]s(y))\\ &= \mathsf{s}^a_t(y) = y = [y \to y]{}^a_t s(y).\end{aligned}$$

$\square$

Lemma 4.3.72. *Let a be a constant symbol and let u, t be terms. For every substitution s, if ${}^a_t s(t) = t$, then*

$$\mathsf{s}^a_t(s(u)) = {}^a_t s(\mathsf{s}^a_t(u)).$$

Proof. The argument is by induction on the term u. If u is a variable, say $u = x$, then both sides of the desired equality are $\mathsf{s}^a_t(s(x))$. If u is a constant symbol $c \neq a$, then $\mathsf{s}^a_t(s(u)) = {}^a_t s(\mathsf{s}^a_t(u)) = c$. The final basis step, when $u = a$, yields $\mathsf{s}^a_t(s(u)) = t$ and ${}^a_t s(\mathsf{s}^a_t(u)) = {}^a_t s(t)$, which implies the equality of the theorem due to the assumption made on t and ${}^a_t s$.

We leave the inductive step to the reader. $\square$

The condition ${}^a_t s(t) = t$ of Lemma 4.3.72 can be satisfied, for example, if $s(x) = x$ for every $x \in \mathtt{V}(t)$, or if $s(x) = a$ and $t = x$.

Theorem 4.3.73. *Let a be a constant symbol, t be a term and φ be a formula. For every substitution s, if $s(x) = x$ for all variables $x \in \mathtt{V}(t)$, then $\mathsf{s}^a_t(\mathrm{FVSubst}(s, \varphi)) = \mathrm{FVSubst}({}^a_t s, \mathsf{s}^a_t(\varphi))$.*

Proof. The argument is by induction on φ. For the basis step, let φ be the atomic formula $R(u_0, \ldots, u_{n-1})$ and s be a substitution such that $s(x) = x$ for all $x \in \mathtt{V}(t)$. We have:

$$\begin{aligned}
&\mathsf{s}^a_t(\mathrm{FVSubst}(s, R(u_0, \ldots, u_{n-1})))\\
&\quad = \mathsf{s}^a_t(R(s(u_0), \ldots, s(u_{n-1})))\\
&\qquad \text{(by Lemma 4.3.53)}\\
&\quad = R(\mathsf{s}^a_t(s(u_0)), \ldots, \mathsf{s}^a_t(s(u_{n-1})))\\
&\quad = R({}^a_t s(\mathsf{s}^a_t(u_0)), \ldots, {}^a_t s(\mathsf{s}^a_t(u_{n-1})))\\
&\qquad \text{(by Lemma 4.3.71)}\\
&\quad = {}^a_t s(R(\mathsf{s}^a_t(u_0), \ldots, \mathsf{s}^a_t(u_{n-1})))\\
&\quad = {}^a_t s(\mathsf{s}^a_t(R(u_0, \ldots, u_{n-1})))\\
&\quad = \mathrm{FVSubst}({}^a_t s, \mathsf{s}^a_t(\varphi)).
\end{aligned}$$

We discuss the only nontrivial inductive step, when $\varphi = (Qy)\psi$ and the result is true for ψ. In this case we can write

$$\begin{aligned}
\mathsf{s}^a_t(\mathrm{FVSubst}(s, \varphi)) &= \mathsf{s}^a_t(\mathrm{FVSubst}(s, (Qy)\psi))\\
&= \mathsf{s}^a_t((Qy)\mathrm{FVSubst}([y \to y]s, \psi))\\
&\quad \text{(by Lemma 4.3.53)}\\
&= (Qy)\mathsf{s}^a_t(\mathrm{FVSubst}([y \to y]s, \psi))\\
&= (Qy)\mathrm{FVSubst}({}^a_t([y \to y]s), \mathsf{s}^a_t(\psi))\\
&\quad \text{(by inductive hypothesis)}\\
&= (Qy)\mathrm{FVSubst}([y \to y]{}^a_t s, \mathsf{s}^a_t(\psi))\\
&\quad \text{(by Lemma 4.3.71)}
\end{aligned}$$

$$\begin{aligned} &= \mathrm{FVSubst}({}^{a}_{t}s, (Qy)\mathsf{s}^{a}_{t}(\psi)) \\ &\quad \text{(by Lemma 4.3.53)} \\ &= \mathrm{FVSubst}({}^{a}_{t}s, \mathsf{s}^{a}_{t}((Qy)\psi)) \\ &= \mathrm{FVSubst}({}^{a}_{t}s, \mathsf{s}^{a}_{t}(\varphi)). \end{aligned}$$

□

The counterpart of the equality of Theorem 4.3.73 for signed formulas is

$$\mathsf{s}^{a}_{t}(\mathrm{FVSubst}(s, b\varphi)) = \mathrm{FVSubst}({}^{a}_{t}s, \mathsf{s}^{a}_{t}(b\varphi)).$$

Corollary 4.3.74. *Let a be a constant symbol, $t, u_0, \ldots, u_{n-1}$ be terms, φ be a formula and let $y_0, \ldots, y_{n-1}$ be variables that do not occur in t. Then, if $u'_i = \mathsf{s}^{a}_{t}(u_i)$ for $0 \leq i \leq n-1$, we have*

$$\mathsf{s}^{a}_{t}((\varphi)_{y_0,\ldots,y_{n-1}:=u_0,\ldots,u_{n-1}}) = (\mathsf{s}^{a}_{t}(\varphi))_{y_0,\ldots,y_{n-1}:=u'_0,\ldots,u'_{n-1}}.$$

Proof. This statement follows by applying Theorem 4.3.73 to the substitution $s = \mathsf{s}^{y_0\cdots y_{n-1}}_{u_0\cdots u_{n-1}}$ and observing that ${}^{a}_{t}s = \mathsf{s}^{y_0\cdots y_{n-1}}_{u'_0\cdots u'_{n-1}}$. □

For the special case $n = 1$ the equality of Corollary 4.3.74 becomes:

$$\mathsf{s}^{a}_{t}((\varphi)_{x:=u}) = (\mathsf{s}^{a}_{t}(\varphi))_{x:=u'}, \tag{4.1}$$

where x is a variable that does not occur in t and $u' = \mathsf{s}^{a}_{t}(u)$. When, in addition, a does not occur in u, this becomes

$$\mathsf{s}^{a}_{t}((\varphi)_{x:=u}) = (\mathsf{s}^{a}_{t}(\varphi))_{x:=u}. \tag{4.2}$$

Further, taking $u = a$ in Equality (4.1), we obtain

$$\mathsf{s}^{a}_{t}((\varphi)_{x:=a}) = (\varphi)_{x:=t} \tag{4.3}$$

assuming that x does not occur in t and a does not occur in φ. Observe that Equality (4.3) holds even without assuming that x does not occur in t due to the definition of substitution of free variables. (Exercise 25 asks for an inductive proof of this statement.)

4.3.6 *Substitutability of Terms*

When we substitute a term for the free occurrences of a variable in a formula we would like the new formula to make the same statement about the term as the old formula made about the variable. Unless we take certain precautions involving the term, this need not be the case, as the next example shows.

Example 4.3.75. Let $\varphi = (\exists y)R(x, y)$ be a formula. The variable x occurs free in φ and, intuitively, φ makes an assertion about x. The formulas $(\varphi)_{x:=z} = (\exists y)R(z, y)$ and $(\varphi)_{x:=c} = (\exists y)R(c, y)$ say the same thing about the variable z and the constant symbol c, respectively, as φ says about x. However, the formula $(\varphi)_{x:=y} = (\exists y)R(y, y)$ has no free variables and makes an assertion only about R. $\square$

As the previous example shows, arbitrary substitutions can create new, unintended bound occurrences of variables. We need to examine conditions that allow "safe" substitutions of terms in a formula in order to avoid such occurrences.

Definition 4.3.76. A term t is *substitutable for a free occurrence* (x, i) of a variable x in formula φ if t contains no variable y such that the occurrence (x, i) lies within the scope of an occurrence of a quantifier symbol (Q, j) using y.

A term t is *substitutable for a variable* x in a formula φ if it is substitutable for every free occurrence of x in φ.

A substitution s is *admissible* for a formula φ if for every variable x, the term $s(x)$ is substitutable for x in φ. $\square$

We can give now a recursive characterization of the substitutability of terms.

Theorem 4.3.77. *Let t be a term, x be a variable and α, β be formulas.*

(1) *If α is an atomic formula, then t is substitutable for x in α.*
(2) *The term t is substitutable for x in $(\neg\alpha)$ if and only if t is substitutable for x in α.*
(3) *For every binary connective symbol C, t is substitutable for x in $(\alpha C \beta)$ if and only if t is substitutable for x in both α and β.*
(4) *For every quantifier symbol Q and variable y, t is substitutable for x in $(Qy)\alpha$ if and only if either*

(a) *x does not occur free in $(Qy)\alpha$ or,*
(b) *t is substitutable for x in α, and y does not occur in t.*

Proof. The first part is immediate since atomic formulas have no quantifier symbols.

For the second part, suppose that t is not substitutable for x in α. Then, there is a free occurrence (x, i) in α which is in the scope of an occurrence (Q, j) in α of a quantifier symbol using a variable y which occurs in t. By Theorem 4.3.38, $(Q, j+2)$ is an occurrence of the quantifier symbol Q using y in $(\neg\alpha)$ and $(x, i+2)$ is a free occurrence of x in $(\neg\alpha)$ which lies in the scope of $(Q, j+2)$. Therefore, t is not substitutable for x in $(\neg\alpha)$.

Conversely, suppose that t is not substitutable for x in $(\neg\alpha)$. This means that there is a free occurrence (x, i) in $(\neg\alpha)$ which is located in the scope of an occurrence (Q, j) of a quantifier symbol using a variable y which appears in t. Again, by Theorem 4.3.38, $(Q, j-2)$ is an occurrence of the quantifier symbol Q using y in α and $(x, i-2)$ is a free occurrence of x in α which falls in the scope of $(Q, j-2)$. This shows that t is not substitutable for x in α.

The proof of the third part of the theorem proceeds in a similar manner to that of the previous part, using Theorem 4.3.39.

Finally, we consider the fourth part of the theorem. Suppose that t is not substitutable for x in $(Qy)\alpha$. Then, there is a free occurrence (x, i) of x in $(Qy)\alpha$ (which shows that condition (a) is violated) and an occurrence of a quantifier symbol $(\hat{Q}, j)$ using a variable z located in t such that (x, i) lies in the scope of $(\hat{Q}, j)$. We must show that condition (b) is violated, i.e., that either t is not substitutable for x in α or y occurs in t. We consider two cases: $j > 3$ and $j = 1$. In the first case, by Theorem 4.3.40, $(x, i-4)$ is a free occurrence of x in α and $(\hat{Q}, j-4)$ is an occurrence of the quantifier symbol $\hat{Q}$ on z whose scope in α includes the occurrence $(x, i-4)$. Therefore, t is not substitutable for x in α. In the second case, $z = y$, so y occurs in t.

Conversely, suppose that both conditions (a) and (b) fail, that is, there is a free occurrence (x, i) of x in $(Qy)\alpha$ and either t is not substitutable for x in α or y occurs in t. Note that $i > 3$, so if y occurs in t, we have immediately that t is not substitutable for x in $(Qy)\alpha$. If, on the other hand, t is not substitutable for x in α, then there must be a free occurrence (x, i') of x in α and a quantifier symbol occurrence $(\tilde{Q}, j)$ in α using a variable z such that z occurs in t and (x, i') is in the scope of $(\tilde{Q}, j)$ in α. Observe that $x \neq y$ (because x occurs free in $(Qy)\alpha$). Therefore, by Theorem 4.3.40, $(x, i' + 4)$ is a free occurrence of x in $(Qy)\alpha$, $(\tilde{Q}, j + 4)$ is an occurrence of a quantifier symbol using z in $(Qy)\alpha$ whose scope contains $(x, i' + 4)$ so t is not substitutable for x in $(Qy)\alpha$. □

In Exercise 20, we present another, more intuitive, equivalent condition for substitutability of terms. The proof of this exercise makes repeated use of Theorem 4.3.77.

The following corollaries will be useful in Section 4.6.3.

Corollary 4.3.78. *Let θ be a formula and let x, y be variables. If $y \notin \mathtt{FV}(\theta)$ and y is substitutable for x in θ, then x is substitutable for y in $(\theta)_{x:=y}$.*

Proof. The proof is by induction on the formula θ. The basis step, when θ is atomic, is immediate. We discuss only one of the inductive steps, namely, when $\theta = (Qz)\beta$ and the statement holds for β. We need to consider two cases.

`Case 1:` $x \notin \mathtt{FV}(\theta)$. Then, $(\theta)_{x:=y} = \theta$ and, since $y \notin \mathtt{FV}(\theta)$, we can conclude that x is substitutable for y in $(\theta)_{x:=y} = \theta$.

`Case 2:` $x \in \mathtt{FV}(\theta)$. Note that this implies that $x \neq z$. We consider two subcases.

`Case 2.1:` $y \notin \mathtt{FV}((\theta)_{x:=y})$. Then, it is immediate that x is substitutable for y in $(\theta)_{x:=y}$.

`Case 2.2:` $y \in \mathtt{FV}((\theta)_{x:=y})$. Since $(\theta)_{x:=y} = (Qz)(\beta)_{x:=y}$, this implies that $y \neq z$, so, since y does not occur free in θ, it does not occur free in β. Moreover, since y is substitutable for x in θ and x occurs free in θ, it follows that y is substitutable for x in β. By the inductive hypothesis, we obtain the substitutability of x for y in $(\beta)_{x:=y}$ and since $z \neq x$, we get the substitutability of x for y in $(Qz)(\beta)_{x:=y} = (\theta)_{x:=y}$.

□

Corollary 4.3.79. *Let φ be a formula, t and u be terms, x be a variable, and c be a constant symbol. If t and u are substitutable for x in φ and $x \notin \mathtt{V}(u)$, then $\mathsf{s}_u^c(t)$ is substitutable for x in $\mathsf{s}_u^c(\varphi)$.*

Proof. The argument is by induction on formulas φ. The basis step, when φ is atomic, clearly holds. For the inductive step, we consider only the case when $\varphi = (Qy)\psi$ and the result is true for ψ. We distinguish two cases, taking into account the characterization of substitutable terms provided by Theorem 4.3.77.

`Case 1:` x does not occur free in φ. Since $\mathtt{FV}(\mathsf{s}_u^c(\varphi)) \subseteq \mathtt{FV}(\varphi) \cup \mathtt{V}(u)$ (Theorem 4.3.69), $x \notin \mathtt{FV}(\mathsf{s}_u^c(\varphi))$ because $x \notin \mathtt{V}(u)$. Therefore, $\mathsf{s}_u^c(t)$ is substitutable for x in $\mathsf{s}_u^c(\varphi)$.

`Case 2:` x occurs free in φ. Then, t and u are substitutable for x in ψ and y does not occur in t or u. By the inductive hypothesis, $\mathsf{s}_u^c(t)$ is substitutable for x in $\mathsf{s}_u^c(\psi)$. Thus, $\mathsf{s}_u^c(t)$ is substitutable for x in $\mathsf{s}_u^c(\varphi)$, because $\mathsf{s}_u^c(\varphi) = (Qy)\mathsf{s}_u^c(\psi)$ and y does not occur in $\mathsf{s}_u^c(t)$ (since y does not occur in either t or u).

$\square$

Corollary 4.3.80. *Let φ be a formula, t be a term, x, y be distinct variables such that y does not occur in φ, and c be a constant symbol. If t is substitutable for x in φ, then $\mathsf{s}_y^c(t)$ is substitutable for x in $\mathsf{s}_y^c(\varphi)$.*

Proof. This statement follows from Corollary 4.3.79 by taking $u = y$ since a variable which does not occur in a formula is substitutable for any variable in that formula. $\square$

Corollary 4.3.81. *Let s be a substitution and φ, ψ be formulas.*

(1) *s is admissible for φ if φ is atomic;*
(2) *s is admissible for $(\neg\varphi)$ if and only if s is admissible for φ;*
(3) *s is admissible for $(\varphi C \psi)$, where C is a binary connective symbol, if and only if s is admissible for both φ and ψ;*
(4) *s is admissible for $(Qx)\varphi$ if and only if $[x \to x]s$ is admissible for φ and for every variable z that occurs free in $(Qx)\varphi$, x does not occur in $s(z)$.*

Proof. The arguments for the first three parts follow immediately from the corresponding parts of Theorem 4.3.77 and the definition of admissibility of substitutions.

From the definition of admissibility of substitutions and the fourth part of Theorem 4.3.77, it follows that a substitution s is admissible for a formula $(Qx)\varphi$ if and only if for every variable y either y does not occur free in $(Qx)\varphi$ or $s(y)$ is substitutable for y in φ and x does not occur in $s(y)$.

We shall prove initially that this implies the condition contained in the fourth part of the Corollary. Let $s' = [x \to x]s$. We have $s'(x) = x$, so $s'(x)$ is substitutable for x in φ. If z does not occur free in φ, then $s'(z)$ is substitutable for z in φ. If $z \neq x$ and $z \in \mathtt{FV}(\varphi)$, then, by hypothesis, $s'(z) = s(z)$ is substitutable for z in φ. Hence, s' is admissible for φ. If $z \in \mathtt{FV}((Qx)\varphi)$, then the hypothesis immediately implies that x does not occur in $s(z)$.

Conversely, suppose that the fourth condition of the corollary holds and let $z \in \mathtt{FV}((Qx)\varphi)$. Clearly, $z \neq x$, so, $s'(z) = s(z)$. By the admissibility of s' for φ, it follows that $s(z)$ is substitutable for z in φ and, by hypothesis, x does not occur in $s(z)$. □

Theorem 4.3.82. *Let s be a substitution that is admissible for the formula φ. We have*

$$\mathtt{FV}(\text{FVSubst}(s,\varphi)) = \bigcup\{\mathtt{V}(s(y)) \mid y \in \mathtt{FV}(\varphi)\}.$$

Proof. We will show by induction on φ that the result holds for every substitution s. For the basis step, suppose that φ is an atomic formula. If φ is a propositional constant, then the result is immediate. Suppose, therefore, that $\varphi = R(t_0, \ldots, t_{n-1})$, where R is an n-ary relation symbol with $n \geq 1$. Then, we can write

$$\begin{aligned}
&\mathtt{FV}(\text{FVSubst}(s,\varphi))\\
&\quad = \mathtt{FV}(s(\varphi))\\
&\quad\quad \text{(by the first part of Lemma 4.3.53)}\\
&\quad = \mathtt{FV}(R(s(t_0), \ldots, s(t_{n-1})))\\
&\quad = \bigcup\{\mathtt{V}(s(t_i)) \mid 0 \leq i \leq n-1\}\\
&\quad\quad \text{(by Theorem 4.3.37)}\\
&\quad = \bigcup\{\bigcup\{\mathtt{V}(s(x)) \mid x \in \mathtt{V}(t_i)\} \mid 0 \leq i \leq n-1\}\\
&\quad\quad \text{(by Theorem 1.5.23)}
\end{aligned}$$

$$= \bigcup\{\mathtt{V}(s(x)) \mid x \in \bigcup\{\mathtt{V}(t_i) \mid 0 \le i \le n-1\}\}$$
$$= \bigcup\{\mathtt{V}(s(x)) \mid x \in \mathtt{FV}(\varphi)\}$$
(by Theorem 4.3.37).

Among the several inductive steps, we consider only the one when $\varphi = (Qx)\psi$ and the result is assumed for ψ. We have

$$\begin{aligned}&\mathtt{FV}(\mathrm{FVSubst}(s, (Qx)\psi))\\ &\quad = \mathtt{FV}((Qx)\mathrm{FVSubst}([x \to x]s, \psi))\\ &\qquad \text{(by Lemma 4.3.53)}\\ &\quad = \mathtt{FV}(\mathrm{FVSubst}([x \to x]s, \psi)) - \{x\}\\ &\qquad \text{(by Theorem 4.3.40).}\end{aligned}$$

The admissibility of s for φ implies the admissibility of $[x \to x]s$ for ψ, by the last part of Corollary 4.3.81. This allows us to apply the inductive hypothesis, so we have

$$\begin{aligned}&\mathtt{FV}(\mathrm{FVSubst}([x \to x]s, \psi)) - \{x\}\\ &\quad = \bigcup\{\mathtt{V}([x \to x]s(y)) \mid y \in \mathtt{FV}(\psi)\} - \{x\}.\end{aligned}$$

Note that

$$\mathtt{V}([x \to x]s(y)) - \{x\} = \begin{cases}\emptyset & \text{if } y = x\\ \mathtt{V}(s(y)) - \{x\} & \text{if } y \ne x.\end{cases}$$

If $y \ne x$ and $y \in \mathtt{FV}(\psi)$, then $y \in \mathtt{FV}(\varphi)$, so by the last part of Corollary 4.3.81, $x \notin \mathtt{V}(s(y))$. Therefore,

$$\begin{aligned}&\mathtt{FV}(\mathrm{FVSubst}([x \to x]s, \psi)) - \{x\}\\ &\quad = \bigcup\{\mathtt{V}(s(y)) \mid y \in \mathtt{FV}(\psi) - \{x\}\}\\ &\quad = \bigcup\{\mathtt{V}(s(y)) \mid y \in \mathtt{FV}(\varphi)\}.\end{aligned}$$

□

Theorem 4.3.83. *Let φ be a formula, $t_0, \ldots, t_{n-1}$ be terms and $Y = \{y_0, \ldots, y_{n-1}\}$ be a set of n distinct variables such that t_i is substitutable for y_i in φ for $0 \le i \le n-1$. We have*:

$$\mathtt{FV}((\varphi)_{y_0,\ldots,y_{n-1}:=t_0,\ldots,t_{n-1}}) = (\mathtt{FV}(\varphi) - Y) \cup \bigcup\{\mathtt{V}(t_j) \mid y_j \in \mathtt{FV}(\varphi)\}.$$

Proof. This follows immediately from Theorem 4.3.82 by taking $s = \mathsf{s}^{y_0 \cdots y_{n-1}}_{t_0 \cdots t_{n-1}}$. □

Corollary 4.3.84. *Let φ be a formula, x be a variable and t be a term that is substitutable for x in φ. Then,*

$$\mathtt{FV}(\varphi) - \{x\} \subseteq \mathtt{FV}((\varphi)_{x:=t}) \subseteq (\mathtt{FV}(\varphi) - \{x\}) \cup \mathtt{V}(t).$$

Proof. This follows from Theorem 4.3.83 by taking $n = 1$. □

Note that by Exercises 32 and 35 the inclusions of Corollary 4.3.84 hold even if the term t is not substitutable for x in φ.

Corollary 4.3.85. *If ψ is the universal $(\mathcal{L}, V)$-formula $(\forall y_0) \cdots (\forall y_{n-1})\varphi$, where φ is quantifier-free, then, every $(\mathcal{L}, V)$-instance of ψ is an $(\mathcal{L}, V)$-formula.*

Proof. If $\theta = (\varphi)_{y_0,\ldots,y_{n-1}:=t_0,\ldots,t_{n-1}}$ is an $(\mathcal{L}, V)$-instance of ψ, since $\mathtt{FV}(\varphi) \subseteq V \cup \{y_0, \ldots, y_{n-1}\}$, we have $\mathtt{FV}(\theta) \subseteq V$ by Theorem 4.3.83. This concludes the argument. □

In the following theorem, we use again the notation $s' * s$ to denote the composition of substitutions s' and s as introduced on page 8.

Theorem 4.3.86. *Let s, s' be substitutions and φ be a formula such that s is admissible for φ. Then,*

$$\mathrm{FVSubst}(s' * s, \varphi) = \mathrm{FVSubst}(s', \mathrm{FVSubst}(s, \varphi)).$$

Proof. The argument is by induction on the formula φ. If φ is atomic, then we have:

$$\begin{aligned}
\mathrm{FVSubst}(s', \mathrm{FVSubst}(s, \varphi)) &= \mathrm{FVSubst}(s', s(\varphi)) \\
&\quad \text{(by Lemma 4.3.53)} \\
&= s'(s(\varphi)) \\
&\quad \text{(by Lemmas 4.3.52 and 4.3.53)} \\
&= (s' * s)(\varphi) \\
&\quad \text{(by Theorem 1.2.19)} \\
&= \mathrm{FVSubst}(s' * s, \varphi) \\
&\quad \text{(by Lemma 4.3.53).}
\end{aligned}$$

We leave to the reader the simple cases when $\varphi = (\neg\alpha)$ and $\varphi = (\alpha C \beta)$, and consider only the case when $\varphi = (Qy)\psi$. Recall that by Corollary 4.3.81 the admissibility of s for φ means that $[y \to y]s$ is admissible for ψ and for all $z \in \mathtt{FV}(\varphi)$, we have $y \notin \mathtt{V}(s(z))$. Now we can write:

$$\begin{aligned}
&\mathrm{FVSubst}(s', \mathrm{FVSubst}(s, (Qy)\psi)) \\
&\quad = \mathrm{FVSubst}(s', (Qy)\mathrm{FVSubst}([y \to y]s, \psi)) \\
&\qquad \text{(by Lemma 4.3.53)} \\
&\quad = (Qy)\mathrm{FVSubst}([y \to y]s', \mathrm{FVSubst}([y \to y]s, \psi)) \\
&\qquad \text{(by Lemma 4.3.53)} \\
&\quad = (Qy)\mathrm{FVSubst}(([y \to y]s') * ([y \to y]s), \psi) \\
&\qquad \text{(by inductive hypothesis, since } [y \to y]s \text{ is admissible for } \psi\text{).}
\end{aligned}$$

We claim that for every $z \in \mathtt{FV}(\psi)$ we have

$$([y \to y]s') * ([y \to y]s)(z) = [y \to y](s' * s)(z).$$

If $z = y$, then both sides of the above equality are equal to y. If $z \neq y$, then $z \in \mathtt{FV}(\varphi)$, so $y \notin \mathtt{V}(s(z))$. This allows us to write

$$\begin{aligned}
([y \to y]s') * ([y \to y]s)(z) &= [y \to y]s'([y \to y]s(z)) \\
&= [y \to y]s'(s(z)) \\
&\qquad \text{(because } y \neq z\text{)} \\
&= s'(s(z)) \\
&\qquad \text{(because } y \notin \mathtt{V}(s(z))\text{)} \\
&= s' * s(z) \\
&= ([y \to y](s' * s))(z).
\end{aligned}$$

By Theorem 4.3.51, we have

$$\begin{aligned}
&(Qy)\mathrm{FVSubst}(([y \to y]s') * ([y \to y]s), \psi) \\
&\quad = (Qy)\mathrm{FVSubst}([y \to y](s' * s), \psi) \\
&\quad = \mathrm{FVSubst}(s' * s, (Qy)\psi).
\end{aligned}$$

□

Corollary 4.3.87. *Let φ be a formula and x, y be variables such that y is substitutable for x in φ and $y \notin \mathtt{FV}(\varphi)$. Then, $((\varphi)_{x:=y})_{y:=x} = \varphi$.*

Proof. The statement clearly holds when $y = x$. Therefore, we assume that $y \neq x$.

The substitutability of y for x in φ means that s_y^x is an admissible substitution for φ. Since $((\varphi)_{x:=y})_{y:=x}$ is $\mathrm{FVSubst}(s_x^y, \mathrm{FVSubst}(s_y^x, \varphi))$, we have:

$$\begin{aligned}
&\mathrm{FVSubst}(s_x^y, \mathrm{FVSubst}(s_y^x, \varphi))\\
&\quad = \mathrm{FVSubst}(s_x^y * s_y^x, \varphi)\\
&\qquad \text{(by Theorem 4.3.86, since } s_y^x \text{ is admissible for } \varphi\text{)}\\
&\quad = \mathrm{FVSubst}(s^{x}_{s_x^y(y)}{}^{y}_{x}, \varphi)\\
&\qquad \text{(by Theorem 1.2.21)}\\
&\quad = \mathrm{FVSubst}(s_x^y, \varphi)\\
&\quad = \varphi\\
&\qquad \text{(because } y \notin \mathtt{FV}(\varphi)\text{).}
\end{aligned}$$

□

4.4 Structures

In order to define the semantics of formulas of first-order logic, we need to introduce structures. They play a role similar to the one played by truth assignments in propositional logic.

Definition 4.4.1. Let $\mathcal{L}$ be a first-order language. An $\mathcal{L}$-*structure* is a pair $\mathcal{A} = (A, \mathcal{I})$ where A is a nonempty set (called the *domain* or *universe* of the structure) and $\mathcal{I}$ is a function with domain $\mathcal{L}$ (called the *interpretation function* of the structure) such that

(1) for each function symbol $f \in \mathcal{L}$, $\mathcal{I}(f)$ is an n-ary function on A (i.e., $\mathcal{I}(f) : A^n \longrightarrow A$) where n is the arity of f;
(2) for each relation symbol $R \in \mathcal{L}$, $\mathcal{I}(R)$ is an n-ary relation on A (i.e., $\mathcal{I}(R) \subseteq A^n$) where n is the arity of R;
(3) if $= \in \mathcal{L}$, then $\mathcal{I}(=)$ is the equality relation on A (i.e., $\mathcal{I}(=) = \{(a, a) \mid a \in A\}$).

▯

If $\mathcal{A} = (A, \mathcal{I})$ is a structure for a first-order language $\mathcal{L}$, then we write $|\mathcal{A}|$ for A, and for each function and relation symbol s

of $\mathcal{L}$ we write $s^{\mathcal{A}}$ for $\mathcal{I}(s)$. If $|\mathcal{A}|$ is finite, countably infinite, countable, or infinite, We will say that $\mathcal{A}$ is finite, countably infinite, countable, infinite, respectively.

If we identify 0-ary functions on a set with elements of the set in the usual way, then the meaning given to an individual constant symbol of a language by the interpretation function of a structure for the language is just an element of the domain of the structure. Similarly, for any set A there are only two 0-ary relations on A, namely $\emptyset$ and $\{\lambda\}$, and if we identify these with $\mathbf{F}$ and $\mathbf{T}$ respectively, then we can consider the value given to a propositional constant by the interpretation function of a structure to be a truth value.

We will use the letters $\mathcal{A}, \mathcal{B}, \mathcal{C}$ to stand for structures.

Example 4.4.2. Let $\mathcal{L} = \{R\}$ be a first-order language that consists of one binary relation symbol R. If (A, ρ) is a partially ordered set, then there is an associated $\mathcal{L}$-structure $\mathcal{A} = (A, \mathcal{I})$, where $\mathcal{I}(R) = \rho$.

To consider structures that correspond to partially ordered sets with greatest and least element, it is natural to use a larger language $\mathcal{L}' = \{c, d, R\}$, where c and d are constant symbols and R is a binary relational symbol. A partially ordered set (A, ρ) with least element 0 and greatest element 1 corresponds to an $\mathcal{L}'$-structure $(A, \mathcal{I}')$, where $\mathcal{I}'(R) = \rho$, $\mathcal{I}'(c) = 0$, and $\mathcal{I}'(d) = 1$. $\square$

Example 4.4.3. The language $\mathcal{L} = \{R, =\}$, where R is a binary relation symbol, is the natural setting for making statements about directed graphs. If $G = (V, E)$ is a directed graph, let $\mathcal{A}_G = (V, \mathcal{I})$, where $\mathcal{I}(R) = E$. Then, $\mathcal{A}$ is an $\mathcal{L}$-structure if and only if there is a directed graph G with $\mathcal{A} = \mathcal{A}_G$. $\square$

Example 4.4.4. Consider the language of arithmetic, $\mathcal{L}_{ar}$, introduced in Example 4.2.3. Let $\mathcal{A}_{ar} = (\mathbf{N}, \mathcal{I})$ be the $\mathcal{L}$-structure defined by:

- $\mathcal{I}(0) = 0$;
- $\mathcal{I}(s) : \mathbf{N} \longrightarrow \mathbf{N}$ is given by $\mathcal{I}(s)(n) = n + 1$ for $n \in \mathbf{N}$.
- $\mathcal{I}(+)(m, n) = m + n$, $\mathcal{I}(\cdot)(m, n) = mn$;
- $\mathcal{I}(=), \mathcal{I}(<) \subseteq \mathbf{N}^2$ are the relations defined by $\mathcal{I}(=) = \{(n, n) | n \in \mathbf{N}\}$, $\mathcal{I}(<) = \{(m, n) | m, n \in \mathbf{N}, m < n\}$.

Note that the symbols $0, +, <$ have a double meaning in the definitions given above. For example, when we write $\mathcal{I}(+)(m, n)$, $+$ is a binary function symbol in $\mathcal{L}_{ar}$, while when we write $m + n$, $+$ represents the addition operation on natural numbers. We shall refer to the structure $\mathcal{A}_{ar}$ as the *standard model of arithmetic*, or simply, *arithmetic*. This structure is useful for expressing properties of natural numbers.

The names used for the symbols of $\mathcal{L}_{ar}$ suggest their intended usage for arithmetic. However, very different $\mathcal{L}_{ar}$-structures exist. Indeed, let M be a set and let $\mathcal{B} = (\mathcal{P}(M), \mathcal{J})$ be the $\mathcal{L}_{ar}$-structure, where $\mathcal{P}(M)$ is the set of subsets of M and $\mathcal{J}$ is the interpretation defined by:

- $\mathcal{J}(0) = \emptyset$;
- $\mathcal{J}(s) : \mathcal{P}(M) \longrightarrow \mathcal{P}(M)$ is given by $\mathcal{J}(s)(K) = M - K$ for $K \in \mathcal{P}(M)$;
- for $K, H \in \mathcal{P}(M)$, we have

$$\mathcal{J}(+)(K, H) = K \cup H, \mathcal{J}(\cdot)(K, H) = K \cap H;$$

- $\mathcal{J}(=)$ and $\mathcal{J}(<)$ are the relations given by

$$\begin{aligned}\mathcal{J}(=) &= \{(K, K) | K \in \mathcal{P}(M)\},\\ \mathcal{J}(<) &= \{(K, H) | K, H \in \mathcal{P}(M) \text{ and } K \subseteq H\}.\end{aligned}$$

□

Example 4.4.5. Let $\mathcal{L}$ be the first-order language that contains two binary function symbols lcm and gcd. We can define $\mathcal{A}$ as the $\mathcal{L}$-structure whose universe is $\mathbf{P}$ with the interpretation $\mathcal{I}$ defined by

$$\begin{aligned}\mathcal{I}(\text{lcm})(n, m) = &\text{ the least common multiple of } n, m,\\ \mathcal{I}(\text{gcd})(n, m) = &\text{ the greatest common divisor of } n, m.\end{aligned}$$

□

Example 4.4.6. Let $\mathcal{L} = \{=, R_1^2\}$. We can define an $\mathcal{L}$-structure $\mathcal{A}$ whose universe is a collection of sets $\mathcal{C}$ by considering the interpretation $\mathcal{I}$, where $\mathcal{I}(=) = \{(K, K) | K \text{ is a member of } \mathcal{C}\}$ and $\mathcal{I}(R_1^2)$ consists of those pairs of sets (K, H) of $\mathcal{C}$, where K is an element of H. □

Definition 4.4.7. Let $\mathcal{L}$ be a first-order language and let $(\mathcal{A}_0, \mathcal{A}_1, \ldots)$ be an infinite sequence of $\mathcal{L}$-structures, all with the same universe A, such that for every symbol $s \in \mathcal{L}$ there is a number $n_s \in \mathbf{N}$ such that for all $n \geq n_s$ we have $s^{\mathcal{A}_n} = s^{\mathcal{A}_{n_s}}$. The *limit* of the above sequence of structures is the $\mathcal{L}$-structure $\mathcal{B}$, denoted $\lim_n \mathcal{A}_n$, given by $|\mathcal{B}| = A$ and $s^{\mathcal{B}} = s^{\mathcal{A}_{n_s}}$, for every $s \in \mathcal{L}$. □

Example 4.4.8. Let $\mathcal{L} = \{c_0, c_1, \ldots, =\}$ be a first-order language and let $\mathcal{A}_n$ be defined by $|\mathcal{A}_n| = \mathbf{N}$ and $c_k^{\mathcal{A}_n} = \min\{n, k\}$. The limit of the sequence $(\mathcal{A}_0, \mathcal{A}_1, \ldots)$ is the $\mathcal{L}$-structure $\mathcal{B}$ defined by $|\mathcal{B}| = \mathbf{N}$ and $c_k^{\mathcal{B}} = k$, for $k \in \mathbf{N}$. □

Definition 4.4.9. Let $\mathcal{A}$ be an $\mathcal{L}$-structure and let R, R' be two n-ary relation symbols such that $R \in \mathcal{L}$ and $R' \notin \mathcal{L}$. Further, assume that if R' is $=$, then $R^{\mathcal{A}} = \{(a, a) \mid a \in |\mathcal{A}|\}$. The $\mathcal{L}'$-structure $\mathcal{A}'$, where $\mathcal{L}' = (\mathcal{L} - \{R\}) \cup \{R'\}$, given by $|\mathcal{A}'| = |\mathcal{A}|$, $s^{\mathcal{A}'} = s^{\mathcal{A}}$ for every $s \in \mathcal{L}' - \{R'\}$ and $R'^{\mathcal{A}'} = R^{\mathcal{A}}$ will be denoted by $\mathcal{A}_{R \to R'}$. □

Example 4.4.10. Let $\mathcal{L}, \mathcal{A}$ be as in Example 4.4.6. The structure $\mathcal{B} = \mathcal{A}_{= \to R_2^2}$ is an $\{R_2^2, R_1^2\}$-structure with the same domain as $\mathcal{A}$ and is defined by $(R_2^2)^{\mathcal{B}} = \{(K, K) | K$ is a member of $\mathcal{C}\}$ and $(R_1^2)^{\mathcal{B}} = (R_1^2)^{\mathcal{A}}$. □

Definition 4.4.11. A first-order language $\mathcal{L}$ is *algebraic* if it has $=$ as its unique relation symbol.

If $\mathcal{L}$ is an algebraic language, then $\mathcal{L}$-structures are called *$\mathcal{L}$-algebras.* □

Let $\mathcal{A} = (A, \mathcal{I})$, $\mathcal{B} = (B, \mathcal{J})$ be two $\mathcal{L}$-structures and let $h : A \longrightarrow B$ be a mapping. If $a = (a_0, \ldots, a_{n-1})$ is an n-tuple of elements in A, then we denote the tuple $(h(a_0), \ldots, h(a_{n-1}))$ by $h \circ a$. This notation is justified by the fact that a is a mapping $a : \{0, \ldots, n-1\} \longrightarrow A$. Furthermore, if $R \subseteq A^n$ is an n-ary relation on A we shall denote the set $\{h \circ a | a \in R\}$ by $h \circ R$.

Definition 4.4.12. Let $\mathcal{A} = (A, \mathcal{I})$, $\mathcal{B} = (B, \mathcal{J})$ be two $\mathcal{L}$-structures.

A *morphism* (sometimes called a *homomorphism*) from $\mathcal{A}$ to $\mathcal{B}$ is a mapping $h : A \longrightarrow B$ such that

(1) for each n-ary function symbol $f \in \mathcal{L}$,

$$h(f^{\mathcal{A}}(a_0, \dots, a_{n-1})) = f^{\mathcal{B}}(h(a_0), \dots, h(a_{n-1}))$$

for all $a_0, \dots, a_{n-1} \in A$;

(2) for each n-ary relation symbol $R \in \mathcal{L}$ different from $=$, we have $a = (a_0, \dots, a_{n-1}) \in R^{\mathcal{A}}$ if and only if $h \circ a = (h(a_0), \dots, h(a_{n-1})) \in R^{\mathcal{B}}$.[3]

A morphism from $\mathcal{A}$ to itself is called an *endomorphism of* $\mathcal{A}$. ▯

If we take $n = 0$ in the previous definition, it follows that for every constant symbol $c \in \mathcal{L}$, $h(c^{\mathcal{A}}) = c^{\mathcal{B}}$ and for every propositional constant $R \in \mathcal{L}$, $R^{\mathcal{A}} = \mathbf{T}$ if and only if $R^{\mathcal{B}} = \mathbf{T}$.

Theorem 4.4.13. *Let $\mathcal{A}, \mathcal{B}, \mathcal{C}$ be $\mathcal{L}$-structures. If $h : |\mathcal{A}| \longrightarrow |\mathcal{B}|$ and $h' : |\mathcal{B}| \longrightarrow |\mathcal{C}|$ are morphisms, then $h' \circ h$ is a morphism from $\mathcal{A}$ to $\mathcal{C}$.*

Proof. The argument is a straightforward application of the definition of morphism and is left to the reader. □

Definition 4.4.14. Let $\mathcal{A} = (A, \mathcal{I})$, $\mathcal{B} = (B, \mathcal{J})$ be two $\mathcal{L}$-structures.

A *monomorphism*, also known as an *embedding*, from $\mathcal{A}$ to $\mathcal{B}$ is an injective morphism $h : A \longrightarrow B$.

An *epimorphism* from $\mathcal{A}$ to $\mathcal{B}$ is a surjective morphism $h : A \longrightarrow B$.

An *isomorphism* from $\mathcal{A}$ to $\mathcal{B}$ is a bijective morphism $h : A \longrightarrow B$.

An *automorphism* of $\mathcal{A}$ is an isomorphism $h : A \longrightarrow A$. ▯

Theorem 4.4.15. *Let $\mathcal{A}, \mathcal{B}$ be two $\mathcal{L}$-structures. If $h : |\mathcal{A}| \longrightarrow |\mathcal{B}|$ is an isomorphism, then $h^{-1} : |\mathcal{B}| \longrightarrow |\mathcal{A}|$ is also an isomorphism.*

Proof. The argument is a straightforward application of the definition of isomorphism and is left to the reader. □

[3]Some authors use a weaker condition in defining morphism; namely, they only require that $a \in R^{\mathcal{A}}$ implies $h \circ a \in R^{\mathcal{B}}$.

Example 4.4.16. Let $\mathcal{L} = \{f_0^2, =, R_1^2\}$ and let $\mathcal{A} = (\mathbf{Z}, \mathcal{I})$, $\mathcal{B} = (\mathbf{N}, \mathcal{J})$ be the $\mathcal{L}$-structures defined by

$$\mathcal{I}(f_0^2)(m,n) = mn \quad \text{for } m,n \in \mathbf{Z}$$
$$\mathcal{J}(f_0^2)(m,n) = mn \quad \text{for } m,n \in \mathbf{N}$$

and

$$\mathcal{I}(R_1^2) = \{(m,n) \mid m,n \in \mathbf{Z} \text{ and } m \text{ divides } n\}$$
$$\mathcal{J}(R_1^2) = \{(m,n) \mid m,n \in \mathbf{N} \text{ and } m \text{ divides } n\}.$$

The mapping $h : \mathbf{Z} \longrightarrow \mathbf{N}$ given by $h(n) = |n|$ is a morphism because $|mn| = |m||n|$ and m divides n if and only if $|m|$ divides $|n|$. Note that h is an epimorphism but not a monomorphism. □

Example 4.4.17. Consider the language $\mathcal{L} = \{f_0^2, =, R_1^2\}$ and let $\mathcal{A} = (\mathbf{P}, \mathcal{I})$ be the $\mathcal{L}$-structure defined by:

- $\mathbf{P}$ is the set of positive natural numbers;
- $\mathcal{I}(f_0^2)(m,n) = \gcd(m,n)$ for $m,n \in \mathbf{P}$;
- $\mathcal{I}(=), \mathcal{I}(R_1^2) \subseteq \mathbf{P}^2$ are the relations defined by

$$\mathcal{I}(=) = \{(n,n) | n \in \mathbf{P}\},$$
$$\mathcal{I}(R_1^2) = \{(m,n) | m,n \in \mathbf{P}, m \text{ divides } n\}.$$

Let $\mathcal{B} = (\mathcal{P}(\mathbf{P}), \mathcal{J})$ be the $\mathcal{L}$-structure defined below.

- $\mathcal{J}(f_0^2)(P,Q) = P \cap Q$ for $P, Q \in \mathcal{P}(\mathbf{P})$;
- $\mathcal{J}(=), \mathcal{I}(R_1^2) \subseteq \mathbf{P}^2$ are the relations defined by

$$\mathcal{J}(=) = \{(P,P) | P \in \mathcal{P}(\mathbf{P})\},$$
$$\mathcal{J}(R_1^2) = \{(P,Q) | P,Q \in \mathcal{P}(\mathbf{P}), P \subseteq Q\}.$$

The mapping $h : \mathbf{P} \longrightarrow \mathcal{P}(\mathbf{P})$ given by $h(n) = \mathsf{DV}(n)$ where $\mathsf{DV}(n)$ is the set of divisors of n for $n \in \mathbf{P}$ is an embedding of $\mathcal{A}$ into $\mathcal{B}$. □

Example 4.4.18. Let S be the subset of $\text{ISeq}(\mathbf{N})$ that consists of those sequences that contain only finitely many nonzero members and let $q = (q_0, q_1, \ldots)$ and $r = (r_0, r_1, \ldots)$ be two sequences in S. Define the sequences $\max(q,r)$ and $\min(q,r)$ as

$$\max(q,r) = (\max(q_0,r_0), \max(q_1,r_1), \ldots)$$
$$\min(q,r) = (\min(q_0,r_0), \min(q_1,r_1), \ldots)$$

Further, define $q \sqsubseteq r$ if $q_i \leq r_i$ for $i \in \mathbf{N}$.

Consider the language $\mathcal{L} = \{f_0^2, f_1^2, =, R_1^2\}$ and let $\mathcal{A} = (\mathbf{P}, \mathcal{I})$ be the $\mathcal{L}$-structure defined by:

- $\mathcal{I}(f_i^2) : \mathbf{P}^2 \longrightarrow \mathbf{P}$, for $i = 0, 1$ are given by

$$\mathcal{I}(f_0^2)(m,n) = \gcd(m,n),$$
$$\mathcal{I}(f_1^2)(m,n) = \operatorname{lcm}(m,n),$$

  for $m, n \in \mathbf{P}$;
- $\mathcal{I}(=), \mathcal{I}(R_1^2) \subseteq \mathbf{P}^2$ are the relations defined by

$$\mathcal{I}(=) = \{(n,n) | n \in \mathbf{P}\},$$
$$\mathcal{I}(R_1^2) = \{(m,n) | m, n \in \mathbf{P}, m \text{ divides } n\}.$$

  Define another $\mathcal{L}$-structure $\mathcal{B} = (S, \mathcal{J})$ by
- $\mathcal{J}(f_i^2) : S^2 \longrightarrow S$, for $i = 0, 1$ are given by

$$\mathcal{J}(f_0^2)(q,r) = \min(q,r)$$
$$\mathcal{J}(f_1^2)(q,r) = \max(q,r)$$

- $\mathcal{J}(=), \mathcal{J}(R_1^2) \subseteq S^2$ are the relations defined by

$$\mathcal{J}(=) = \{(q,q) \mid q \in S\}$$
$$\mathcal{J}(R_1^2) = \{(q,r) \mid q \sqsubseteq r\}$$

The mapping $h : \mathbf{P} \longrightarrow S$ given by

$$h(1) = (0, 0, \ldots) \quad \text{and} \quad h(n) = (m_0, \ldots, m_{k-1}, 0, 0, \ldots)$$

if $n > 1$ and $n = p_0^{m_0} \cdots p_{k-1}^{m_{k-1}}$, where $p_0, p_1, \ldots$ is the sequence of prime numbers in increasing order and p_{k-1} is the largest prime that divides n, is an isomorphism. $\square$

Example 4.4.19. Let $\mathcal{L}$ be $\{R_1^2\}$. Define the $\mathcal{L}$-structure $\mathcal{N} = (\mathbf{N}, \mathcal{I})$, where $\mathcal{I}(R_1^2) = \{(m,n) \mid m, n \in \mathbf{N}, m < n\}$. Suppose that $h : \mathbf{N} \longrightarrow \mathbf{N}$ is an automorphism. We claim that $h(n) = n$ for all $n \in \mathbf{N}$; in other words, the structure $\mathcal{N}$ has a unique automorphism, the identity mapping. A structure whose only automorphism is the identity mapping is called *rigid*.

We prove by induction on n that $h(n) = n$. For the basis step, let p be such that $h(p) = 0$; p exists because h is onto. If $p \neq 0$, we have

$0 < p$, so $h(0) < h(p) = 0$, which is impossible. Thus, $h(0) = 0$. Now suppose that $h(n) = n$ and let q be such that $h(q) = n + 1$. Since $h(n) < h(q)$, it follows that $n < q$. Suppose $q \neq n+1$. Then, we must have $n + 1 < q$ which implies $n = h(n) < h(n + 1) < h(q) = n + 1$, a contradiction. Therefore, $q = n + 1$, so $h(n + 1) = n + 1$.

By contrast, the $\mathcal{L}$-structure $\mathcal{Z} = (\mathbf{Z}, \mathcal{J})$, where $\mathcal{J}(R_1^2) = \{(m, n) \mid m, n \in \mathbf{Z}, m < n\}$ is not rigid because for each $k \in \mathbf{Z}$, the mapping $h_k : \mathbf{Z} \longrightarrow \mathbf{Z}$ given by $h_k(p) = p + k$ for every $p \in \mathbf{Z}$ is an automorphism, as the reader can easily verify. Moreover, every automorphism of $\mathcal{Z}$ has this form. Indeed, let h be an automorphism of $\mathcal{Z}$. We claim that

$$h(n + 1) = h(n) + 1 \quad \text{and} \quad h(n - 1) = h(n) - 1 \tag{4.4}$$

for $n \in \mathbf{Z}$. We will prove only the first equality. Fix $n \in \mathbf{Z}$. Since h is onto, there is $p \in \mathbf{Z}$ such that $h(p) = h(n) + 1$. Then, $h(n) < h(p)$ implies $n < p$. If p were different from $n + 1$, we would have $n < n + 1 < p$ and this would give $h(n) < h(n + 1) < h(p) = h(n) + 1$. This contradiction shows that $h(n + 1) = h(n) + 1$.

The equalities (4.4) imply that $h(n) = n + h(0)$ for $n \in \mathbf{Z}$, which shows that h has the desired form. $\square$

Example 4.4.20. Let $\mathcal{L} = \{f_0^2, =\}$ be a first-order language and let V be an alphabet. Define the $\mathcal{L}$-structure $\mathcal{A}_V = (V^*, \mathcal{I})$, where $\mathcal{I}(f_0^2)$ is the concatenation operation on V^*. If V and U are two alphabets, a morphism from $\mathcal{A}_V$ to $\mathcal{A}_U$ is a mapping $h : V^* \longrightarrow U^*$ such that $h(xy) = h(x)h(y)$ for all words $x, y \in V^*$. An endomorphism of $\mathcal{A}_V$ is a mapping $h : V^* \longrightarrow V^*$ with the above property. For instance, if $V = \{a, b, c\}$ and $h(x)$ is obtained from the word x by erasing every symbol different from a, then h is an endomorphism of $\mathcal{A}_V$. $\square$

Definition 4.4.21. Let $\mathcal{L}$ be a first-order language. A *substructure* of an $\mathcal{L}$-structure $\mathcal{A} = (A, \mathcal{I})$ is an $\mathcal{L}$-structure $\mathcal{B} = (B, \mathcal{J})$, such that

(1) $B \subseteq A$;
(2) for each n-ary function symbol $f \in \mathcal{L}$, $f^{\mathcal{B}} = f^{\mathcal{A}} \restriction B^n$;
(3) for each n-ary relation symbol $R \in \mathcal{L}$, $R^{\mathcal{B}} = R^{\mathcal{A}} \cap B^n$.

$\square$

Note that according to Definition 4.4.21, a structure $\mathcal{A}$ is always a substructure of itself. In this capacity, it is referred to as the *trivial substructure* of $\mathcal{A}$. Any other substructure is called *nontrivial*.

Theorem 4.4.22. *Let $\mathcal{A} = (A, \mathcal{I}), \mathcal{B} = (B, \mathcal{J})$ be two $\mathcal{L}$-structures. Then, $\mathcal{B}$ is a substructure of $\mathcal{A}$ if and only if $\iota_B = \{(b, b) \mid b \in B\}$ is a morphism from $\mathcal{B}$ to $\mathcal{A}$.*

Proof. Suppose that ι_B is a morphism from $\mathcal{B}$ to $\mathcal{A}$. Then, $\iota_B : B \longrightarrow A$, so $B \subseteq A$. Since ι_B is a morphism, for each n-ary function symbol $f \in \mathcal{L}$ and $b_0, \ldots, b_{n-1} \in B$, we have:

$$\begin{aligned} f^{\mathcal{B}}(b_0, \ldots, b_{n-1}) = \iota_B(f^{\mathcal{B}}(b_0, \ldots, b_{n-1}) \\ & f^{\mathcal{A}}(\iota_B(b_0), \ldots, \iota_B(b_{n-1})) \\ & f^{\mathcal{A}}(b_0, \ldots, b_{n-1}), \end{aligned}$$

so $f^{\mathcal{B}} = f^{\mathcal{A}} \restriction B^n$. For the same reason, for every n-ary relation symbol $R \in \mathcal{L}$ and $b_0, \ldots, b_{n-1} \in B$, we have $(b_0, \ldots, b_{n-1}) \in R^{\mathcal{B}}$ if and only if $(\iota_B(b_0), \ldots, \iota_B(b_{n-1})) \in R^{\mathcal{A}}$ if and only if $(b_0, \ldots, b_{n-1}) \in R^{\mathcal{A}}$. This shows that $R^{\mathcal{B}} = R^{\mathcal{A}} \cap B^n$. So, $\mathcal{B}$ is a substructure of $\mathcal{A}$.

The similar argument for the reverse implication is left to the reader. □

Not every subset of the domain of a structure is the domain of a substructure. The following theorem characterizes domains of substructures.

Theorem 4.4.23. *Let $\mathcal{A} = (A, \mathcal{I})$ be an $\mathcal{L}$-structure and let B be a subset of A. Then, there is a substructure $\mathcal{B} = (B, \mathcal{J})$ of $\mathcal{A}$ if and only if $B \neq \emptyset$ and B is $\{f^{\mathcal{A}} \mid f \in F_{\mathcal{L}}\}$-closed. Further, for such a set B, the interpretation function $\mathcal{J}$ is uniquely determined.*

Proof. The necessity of the conditions involving B follows immediately from the definition of substructure. Conversely, suppose that $B \neq \emptyset$ and B is $\{f^{\mathcal{A}} \mid f \in F_{\mathcal{L}}\}$-closed. We obtain a substructure $\mathcal{B} = (B, \mathcal{J})$ by defining $\mathcal{J}(f) = f^{\mathcal{A}} \restriction B^n$ for every n-ary function symbol f of $\mathcal{L}$ and $\mathcal{J}(R) = R^{\mathcal{A}} \cap B^n$ for every n-ary relation symbol R of $\mathcal{L}$. Note that this definition of $\mathcal{J}$ is correct because B is $f^{\mathcal{A}}$-closed for every function symbol f of $\mathcal{L}$. Suppose that $\mathcal{B}' = (B, \mathcal{J}')$ is another substructure of $\mathcal{A}$ that has domain B. By the definition of substructure, $\mathcal{J}'(f) = f^{\mathcal{A}} \restriction B^n = \mathcal{J}(f)$ for every n-ary function symbol f of $\mathcal{L}$ and, for the same reason, $\mathcal{J}'(R) = R^{\mathcal{A}} \cap B^n = \mathcal{J}(R)$ for every n-ary relation symbol R of $\mathcal{L}$. Therefore, $\mathcal{J}' = \mathcal{J}$. $\square$

Example 4.4.24. For the structure introduced in Example 4.4.5 whose universe is $\mathbf{P}$, the set E of positive even natural numbers defines a substructure of $\mathcal{A}$ because the greatest common divisor and the least common multiple of two even numbers are again even. $\square$

Example 4.4.25. The structure $\mathcal{A}_{ar}$ defined in Example 4.4.4 has no nontrivial substructures because any set of natural numbers that contains 0 and is closed under the successor function coincides with $\mathbf{N}$. $\square$

Theorem 4.4.26. *Let $\mathcal{A}, \mathcal{B}$ be two $\mathcal{L}$-structures and let $h : |\mathcal{A}| \longrightarrow |\mathcal{B}|$ be a morphism. Then, $h(|\mathcal{A}|)$ is the domain of a substructure of $\mathcal{B}$.*

Proof. Since $|\mathcal{A}|$ is nonempty, it is clear that $h(|\mathcal{A}|)$ is nonempty. Let f be an n-ary function symbol of $\mathcal{L}$ and let $b_0, \ldots, b_{n-1} \in h(|\mathcal{A}|)$. There are $a_0, \ldots, a_{n-1} \in |\mathcal{A}|$ such that $b_i = h(a_i)$ for $0 \leq i \leq n-1$. If $b = f^{\mathcal{B}}(b_0, \ldots, b_{n-1})$, then we can write

$$b = f^{\mathcal{B}}(h(a_0), \ldots, h(a_{n-1})) = h(f^{\mathcal{A}}(a_0, \ldots, a_{n-1})) \in h(|\mathcal{A}|),$$

so the result follows by Theorem 4.4.23. $\square$

If h is a morphism from $\mathcal{A}$ to $\mathcal{B}$, by Theorems 4.4.23 and 4.4.26, there is a unique substructure of $\mathcal{B}$ with domain $h(|\mathcal{A}|)$. We will denote this substructure by $h(\mathcal{A})$.

Corollary 4.4.27. *Let $\mathcal{L}$ be a first-order language. An $\mathcal{L}$-structure $\mathcal{C}$ is a substructure of an $\mathcal{L}$-structure $\mathcal{B}$ if and only if there is an $\mathcal{L}$-structure $\mathcal{A}$ and a morphism h from $\mathcal{A}$ to $\mathcal{B}$ such that $\mathcal{C} = h(\mathcal{A})$.*

Proof. If $\mathcal{C}$ is substructure of $\mathcal{B}$, then, by Theorem 4.4.22, the mapping $\iota_{|\mathcal{C}|}$ is a morphism from $\mathcal{C}$ to $\mathcal{B}$ and we have $\iota_{|\mathcal{C}|}(\mathcal{C}) = \mathcal{C}$.

The converse follows immediately from Theorem 4.4.26. $\square$

Definition 4.4.28. Let $\mathcal{A}$ be an $\mathcal{L}$-structure. An equivalence relation ρ on $|\mathcal{A}|$ is a *congruence of* $\mathcal{A}$ if we have:

- The following $f^{\mathcal{A}}$-*compatibility condition* holds for each n-ary function symbol $f \in \mathcal{L}$: for all n-tuples $a = (a_0, \dots, a_{n-1})$, and $b = (b_0, \dots, b_{n-1}) \in |\mathcal{A}|^n$, if $a_i \rho b_i$ for $0 \leq i \leq n-1$, then $f^{\mathcal{A}}(a) \rho f^{\mathcal{A}}(b)$.
- The following $R^{\mathcal{A}}$-*compatibility condition* holds for each n-ary relation symbol $R \in \mathcal{L} - \{=\}$: for all n-tuples $a = (a_0, \dots, a_{n-1})$, $b = (b_0, \dots, b_{n-1}) \in |\mathcal{A}|^n$, if $a_i \rho b_i$ for $0 \leq i \leq n-1$, then $a \in R^{\mathcal{A}}$ if and only if $b \in R^{\mathcal{A}}$.

□

It is clear that in every structure $\mathcal{A}$, the equality relation $\iota_{|\mathcal{A}|}$ on $|\mathcal{A}|$ and the relation $|\mathcal{A}| \times |\mathcal{A}|$ are congruences of $\mathcal{A}$. Next, we consider some less trivial examples.

Example 4.4.29. Let $\mathcal{A}$ be the structure with domain $\mathbf{Z}$ introduced in Example 4.4.16. The relation $\rho = \{(m, n) \in \mathbf{Z} \times \mathbf{Z} \mid m = n \text{ or } m = -n\}$ is a congruence of $\mathcal{A}$. Indeed, as the reader can easily verify, if $a_0 = \pm b_0$ and $a_1 = \pm b_1$, then $a_0 a_1 = \pm b_0 b_1$, and a_0 divides a_1 if and only if b_0 divides b_1. □

Definition 4.4.30. Let ρ be a congruence of an $\mathcal{L}$-structure $\mathcal{A}$. The *quotient of* $\mathcal{A}$ *by* ρ is the structure $\mathcal{A}/\rho$ whose universe is $|\mathcal{A}|/\rho$ and whose interpretation function is given by

- for every n-ary function symbol f in $\mathcal{L}$ and $a_0, \dots, a_{n-1} \in |\mathcal{A}|$, we have
$$f^{\mathcal{A}/\rho}([a_0]_\rho, \dots, [a_{n-1}]_\rho) = [f^{\mathcal{A}}(a_0, \dots, a_{n-1})]_\rho$$
- for every n-ary relation symbol R in $\mathcal{L} - \{=\}$, we have
$$R^{\mathcal{A}/\rho} = \{([a_0]_\rho, \dots, [a_{n-1}]_\rho) \mid (a_0, \dots, a_{n-1}) \in R^{\mathcal{A}}\}$$
- if $= \in \mathcal{L}$, then $=^{\mathcal{A}/\rho} = \{([a]_\rho, [a]_\rho) \mid a \in |\mathcal{A}|\}$.

□

The functions $f^{\mathcal{A}/\rho}$ introduced above are well-defined because ρ is a congruence.

Example 4.4.31. The quotient structure $\mathcal{A}/\rho$, where $\mathcal{A}$ and ρ were considered in Example 4.4.29, has universe $|\mathcal{A}/\rho| = \{\{n, -n\} \mid n \in \mathbf{Z}\}$

and its interpretation function is given by

$$\begin{aligned}(f_0^2)^{\mathcal{A}/\rho}(\{n,-n\},\{m,-m\}) &= \{nm,-nm\}\\ (R_1^2)^{\mathcal{A}/\rho} &= \{(\{n,-n\},\{m,-m\}) \mid n \text{ divides } m\}\\ =^{\mathcal{A}/\rho} &= \{(\{n,-n\},\{n,-n\}) \mid n \in \mathbf{Z}\}.\end{aligned}$$

□

Theorem 4.4.32. *Let $\mathcal{A}$ be an $\mathcal{L}$-structure and let R be a binary relation symbol in $\mathcal{L}$ such that $R^{\mathcal{A}}$ is a congruence of $\mathcal{A}$. Then, $R^{\mathcal{A}/R^{\mathcal{A}}}$ is the equality relation on $|\mathcal{A}/R^{\mathcal{A}}| = |\mathcal{A}|/R^{\mathcal{A}}$.*

Proof. The argument is immediate and is left to the reader. □

Definition 4.4.33. Let $\mathcal{A}$ be an $\mathcal{L}$-structure and let ρ be a congruence of $\mathcal{A}$. The *canonical morphism of* ρ is the mapping $h_\rho : |\mathcal{A}| \longrightarrow |\mathcal{A}/\rho|$ given by $h_\rho(a) = [a]_\rho$ for every $a \in |\mathcal{A}|$. □

We leave to the reader the verification that h_ρ is indeed a morphism.

If $f : A \longrightarrow B$, then we define the *kernel* of f, denoted by $\mathbf{ker}(f)$, as the equivalence relation $\{(a,a') \in A \times A \mid f(a) = f(a')\}$.

Theorem 4.4.34. *Let $\mathcal{A}$ be an $\mathcal{L}$-structure. A binary relation ρ on $|\mathcal{A}|$ is a congruence of $\mathcal{A}$ if and only if there is an $\mathcal{L}$-structure $\mathcal{B}$ and a morphism h from $\mathcal{A}$ to $\mathcal{B}$ such that $\rho = \mathbf{ker}(h)$.*

Proof. Suppose first that ρ is a congruence of $\mathcal{A}$ and let h_ρ be the canonical morphism from $\mathcal{A}$ to $\mathcal{A}/\rho$. Observe that

$$\begin{aligned}\mathbf{ker}(h_\rho) &= \{(a,b) \in |\mathcal{A}| \times |\mathcal{A}| \mid h_\rho(a) = h_\rho(b)\}\\ &= \{(a,b) \in |\mathcal{A}| \times |\mathcal{A}| \mid [a]_\rho = [b]_\rho\}\\ &= \rho.\end{aligned}$$

Conversely, suppose that h is a morphism from $\mathcal{A}$ to $\mathcal{B}$ and $\rho = \mathbf{ker}(h)$. It is clear that ρ is an equivalence relation. Assume that $(a_i,b_i) \in \mathbf{ker}(h)$, that is, $h(a_i) = h(b_i)$ for $0 \le i \le n-1$. Let f be an n-ary symbol of $\mathcal{L}$. We have:

$$\begin{aligned}h(f^{\mathcal{A}}(a_0,\dots,a_{n-1})) &= f^{\mathcal{B}}(h(a_0),\dots,h(a_{n-1}))\\ &= f^{\mathcal{B}}(h(b_0),\dots,h(b_{n-1}))\\ &= h(f^{\mathcal{A}}(b_0,\dots,b_{n-1})),\end{aligned}$$

so $(f^{\mathcal{A}}(a_0,\dots,a_{n-1}), f^{\mathcal{A}}(b_0,\dots,b_{n-1})) \in \mathbf{ker}(h)$.

If R is an n-ary relation symbol in $\mathcal{L} - \{=\}$, the following statements are equivalent.

$$\begin{gathered}(a_0, \ldots, a_{n-1}) \in R^{\mathcal{A}};\\ (h(a_0), \ldots, h(a_{n-1})) \in R^{\mathcal{B}};\\ (h(b_0), \ldots, h(b_{n-1})) \in R^{\mathcal{B}};\\ (b_0, \ldots, b_{n-1}) \in R^{\mathcal{A}}.\end{gathered}$$

Thus, $\rho = \mathbf{ker}(h)$ is a congruence. □

Theorem 4.4.35. *If $\mathcal{A}, \mathcal{B}$ are $\mathcal{L}$-structures and h is a morphism from $\mathcal{A}$ to $\mathcal{B}$, then the mapping $k : |\mathcal{A}|/\mathbf{ker}(h) \longrightarrow h(|\mathcal{A}|)$ given by $k([a]_{\mathbf{ker}(h)}) = h(a)$ is an isomorphism between the structures $\mathcal{A}/\mathbf{ker}(h)$ and $h(\mathcal{A})$.*

Proof. It is easy to check that k is well-defined and is a bijection. Thus, it remains only to show that k is a morphism. To simplify notation, we write $[a]$ instead of $[a]_{\mathbf{ker}(h)}$.

Let f be an n-ary function symbol of $\mathcal{L}$ and let $a_0, \ldots, a_{n-1}$ be n elements of $\mathcal{A}$. We have

$$\begin{aligned} k(f^{\mathcal{A}/\mathbf{ker}(h)}([a_0], \ldots, [a_{n-1}])) &= k([f^{\mathcal{A}}(a_0, \ldots, a_{n-1})])\\ &= h(f^{\mathcal{A}}(a_0, \ldots, a_{n-1}))\\ &= f^{\mathcal{B}}(h(a_0), \ldots, h(a_{n-1}))\\ &= f^{h(\mathcal{A})}(h(a_0), \ldots, h(a_{n-1}))\\ &= f^{h(\mathcal{A})}(k([a_0]), \ldots, k([a_{n-1}])).\end{aligned}$$

If R is an n-ary relation symbol of $\mathcal{L}$, then we have

$$\begin{aligned} &([a_0], \ldots, [a_{n-1}]) \in R^{\mathcal{A}/\mathbf{ker}(h)}\\ &\quad \text{if and only if } (a_0, \ldots, a_{n-1}) \in R^{\mathcal{A}}\\ &\quad \text{if and only if } (h(a_0), \ldots, h(a_{n-1})) \in R^{\mathcal{B}}\\ &\quad \text{if and only if } (h(a_0), \ldots, h(a_{n-1})) \in R^{h(\mathcal{A})}\\ &\quad \text{if and only if } (k([a_0]), \ldots, k([a_{n-1}])) \in R^{h(\mathcal{A})}\end{aligned}$$

for all $a_0, \ldots, a_{n-1} \in |\mathcal{A}|$, which shows that k is a morphism. □

The notions we are about to introduce are useful when we deal with more than one first-order language.

Definition 4.4.36. Let $\mathcal{L}, \mathcal{L}'$ be two first-order languages such that $\mathcal{L} \subseteq \mathcal{L}'$, $\mathcal{A}$ be an $\mathcal{L}$-structure, and $\mathcal{B}$ be an $\mathcal{L}'$-structure. $\mathcal{A}$ is *the reduct of $\mathcal{B}$ to $\mathcal{L}$* and $\mathcal{B}$ is *an expansion of $\mathcal{A}$ to $\mathcal{L}'$* if $|\mathcal{A}| = |\mathcal{B}|$ and for all $s \in \mathcal{L}$, $s^{\mathcal{A}} = s^{\mathcal{B}}$. ▯

The next theorem justifies the use of the word "the" in the definition of "the reduct of a structure to a first-order language."

Theorem 4.4.37. *Let $\mathcal{L}, \mathcal{L}'$ be two first-order languages such that $\mathcal{L} \subseteq \mathcal{L}'$. Given an $\mathcal{L}$-structure $\mathcal{A}$, there is at least one $\mathcal{L}'$-structure $\mathcal{B}$ such that $\mathcal{B}$ is an expansion of $\mathcal{A}$ to $\mathcal{L}'$. Conversely, given an $\mathcal{L}'$-structure $\mathcal{B}$, there is a unique $\mathcal{L}$-structure $\mathcal{A}$ such that $\mathcal{A}$ is the reduct of $\mathcal{B}$ to $\mathcal{L}$.*

Proof. The argument is immediate and it is left to the reader. □

We will denote the reduct of a structure $\mathcal{B}$ to a first-order language $\mathcal{L}$ by $\mathrm{RED}_{\mathcal{L}}(\mathcal{B})$.

Theorem 4.4.38. *Let $\mathcal{L}, \mathcal{L}', \mathcal{L}''$ be first-order languages such that $\mathcal{L} \subseteq \mathcal{L}' \subseteq \mathcal{L}''$. If $\mathcal{A}, \mathcal{A}', \mathcal{A}''$ are an $\mathcal{L}$-structure, an $\mathcal{L}'$-structure, and an $\mathcal{L}''$-structure, respectively, then $\mathcal{A}' = RED_{\mathcal{L}'}(\mathcal{A}'')$ and $\mathcal{A} = RED_{\mathcal{L}}(\mathcal{A}')$ implies $\mathcal{A} = RED_{\mathcal{L}}(\mathcal{A}'')$.*

Proof. The argument is immediate and it is left to the reader. □

Example 4.4.39. The chain of inclusions $\mathcal{L}_s \subseteq \mathcal{L}_{s,<} \subseteq \mathcal{L}_{pra} \subseteq \mathcal{L}_{ar}$ of the first-order languages defined in Example 4.2.3 allows us to consider the following reducts of the standard model of arithmetic $\mathcal{A}_{ar}$ (introduced in Example 4.4.4):

$$\mathcal{A}_s = \mathrm{RED}_{\mathcal{L}_s}(\mathcal{A}_{ar})$$
$$\mathcal{A}_{s,<} = \mathrm{RED}_{\mathcal{L}_{s,<}}(\mathcal{A}_{ar})$$
$$\mathcal{A}_{pra} = \mathrm{RED}_{\mathcal{L}_{pra}}(\mathcal{A}_{ar}).$$

▯

4.5 Semantics of First-Order Logic

In this section, we use structures to define a semantics for formulas. The meaning of a formula is a certain Boolean-valued function whose arguments are structures and assignments; the latter are devices for assigning elements of structures to the free variables of the formula. After giving the semantics of first-order formulas, we introduce the concept of validity which is for first-order logic what tautology is for propositional logic. At the end of the section, we examine the possibility of using first-order formulas for defining certain subsets of structures.

4.5.1 *Assignments in Structures*

Definition 4.5.1. Let $\mathcal{L}$ be a first-order language and let $\mathcal{A}$ be an $\mathcal{L}$-structure. An *assignment* over $\mathcal{A}$ is a function from VAR to $|\mathcal{A}|$. $\square$

We will denote the set VAR $\longrightarrow |\mathcal{A}|$ of all assignments in a given structure $\mathcal{A}$ by ASSIGN$_{\mathcal{A}}$. The letters σ, ρ, τ will stand for assignments. Observe that for every structure $\mathcal{A}$, the set ASSIGN$_{\mathcal{A}}$ is not empty, since $|\mathcal{A}| \neq \emptyset$.

Definition 4.5.2. Let $\mathcal{L}$ be a first-order language, $\mathcal{A}$ be an $\mathcal{L}$-structure, and σ be an assignment over $\mathcal{A}$. We define a function $\sigma^{\mathcal{A}} : \text{TERM}_{\mathcal{L}} \longrightarrow |\mathcal{A}|$, which is an extension of σ, by the following recursive definition:

(1) For every constant symbol c of $\mathcal{L}$

$$\sigma^{\mathcal{A}}(c) = c^{\mathcal{A}}.$$

(2) For all variables x,

$$\sigma^{\mathcal{A}}(x) = \sigma(x).$$

(3) For all n-ary function symbols f of $\mathcal{L}$ of positive arity and $\mathcal{L}$-terms $t_0, \ldots, t_{n-1}$,

$$\sigma^{\mathcal{A}}(f(t_0, \ldots, t_{n-1})) = f^{\mathcal{A}}(\sigma^{\mathcal{A}}(t_0), \ldots, \sigma^{\mathcal{A}}(t_{n-1})).$$

$\square$

The value $\sigma^{\mathcal{A}}(t)$ is the value of the term t in $|\mathcal{A}|$ when the assignment σ is used to evaluate the variables.

Theorem 4.5.3. *Let $\mathcal{L}$ be a first-order language, $\mathcal{A}$ be an $\mathcal{L}$-structure, t be an $\mathcal{L}$-term, and let σ, τ be two assignments over $\mathcal{A}$ such that $\sigma(x) = \tau(x)$ for every variable x that occurs in t. Then, we have $\sigma^{\mathcal{A}}(t) = \tau^{\mathcal{A}}(t)$.*

Proof. The argument is by structural induction on the term t. If t is a variable x, then $\sigma(x) = \tau(x)$ and

$$\sigma^{\mathcal{A}}(x) = \sigma(x) = \tau(x) = \tau^{\mathcal{A}}(x).$$

If t is a constant symbol, the desired conclusion follows immediately from Definition 4.5.2.

Suppose that $t = f(t_0, \ldots, t_{n-1})$, where f is an n-ary function symbol with $n > 0$ and that

$$\sigma^{\mathcal{A}}(t_0) = \tau^{\mathcal{A}}(t_0), \ldots, \sigma^{\mathcal{A}}(t_{n-1}) = \tau^{\mathcal{A}}(t_{n-1}).$$

This allows us to write

$$\begin{aligned}
\sigma^{\mathcal{A}}(t) &= \sigma^{\mathcal{A}}(f(t_0, \ldots, t_{n-1})) \\
&= f^{\mathcal{A}}(\sigma^{\mathcal{A}}(t_0), \ldots, \sigma^{\mathcal{A}}(t_{n-1})) \\
&= f^{\mathcal{A}}(\tau^{\mathcal{A}}(t_0), \ldots, \tau^{\mathcal{A}}(t_{n-1})) \\
&= \tau^{\mathcal{A}}(f(t_0, \ldots, t_{n-1})) = \tau^{\mathcal{A}}(t).
\end{aligned}$$

□

Corollary 4.5.4. *Let $\mathcal{L}$ be a first-order language, $\mathcal{A}$ be an $\mathcal{L}$-structure, t be a ground $\mathcal{L}$-term. For every $\sigma, \tau \in \mathrm{ASSIGN}_{\mathcal{A}}$, we have $\sigma^{\mathcal{A}}(t) = \tau^{\mathcal{A}}(t)$.*

Proof. Since t is a ground term, $\mathtt{V}(t) = \emptyset$, so σ and τ agree on all the variables of t. The statement now follows immediately from Theorem 4.5.3. □

If t is a ground $\mathcal{L}$-term and $\mathcal{A} = (A, \mathcal{I})$ is an $\mathcal{L}$-structure, then we denote by $t^{\mathcal{A}}$ the common value of $\sigma^{\mathcal{A}}(t)$ for $\sigma \in \mathrm{ASSIGN}_{\mathcal{A}}$. Observe that if t is a constant symbol c, then the notation $c^{\mathcal{A}}$ just introduced is consistent with the notation already in use for $\mathcal{I}(c)$.

Theorem 4.5.5. *Let $\mathcal{A}$ be an $\mathcal{L}$-structure, $t_0, \ldots, t_{n-1}$ be ground terms of $\mathcal{L}$, and let f be an n-ary function symbol in $\mathcal{L}$ with $n \geq 1$. Then,*

$$(f(t_0, \ldots, t_{n-1}))^{\mathcal{A}} = f^{\mathcal{A}}(t_0^{\mathcal{A}}, \ldots, t_{n-1}^{\mathcal{A}}).$$

Proof. Let $\sigma \in \mathrm{ASSIGN}_{\mathcal{A}}$. We have:

$$\begin{aligned}(f(t_0, \ldots, t_{n-1}))^{\mathcal{A}} &= \sigma^{\mathcal{A}}(f(t_0, \ldots, t_{n-1}))\\ &= f^{\mathcal{A}}(\sigma^{\mathcal{A}}(t_0), \ldots, \sigma^{\mathcal{A}}(t_{n-1}))\\ &= f^{\mathcal{A}}(t_0^{\mathcal{A}}, \ldots, t_{n-1}^{\mathcal{A}}).\end{aligned}$$

$\square$

Theorem 4.5.6. *Let $\mathcal{L}$ be a first-order language and let $\mathcal{A}, \mathcal{B}$ be two $\mathcal{L}$-structures. If $h : |\mathcal{A}| \longrightarrow |\mathcal{B}|$ is a morphism, $\sigma \in \mathrm{ASSIGN}_{\mathcal{A}}$, and $t \in \mathrm{TERM}_{\mathcal{L}}$, then $h(\sigma^{\mathcal{A}}(t)) = (h \circ \sigma)^{\mathcal{B}}(t)$.*

Proof. The argument is by induction on the term t and is left to the reader. $\square$

Definition 4.5.7. Let $\mathcal{L}$ be a first-order language, t, u be $\mathcal{L}$-terms and $\mathcal{A}$ be an $\mathcal{L}$-structure. The terms t and u are $\mathcal{A}$-*equivalent*, written $t \equiv_{\mathcal{A}} u$, if $\sigma^{\mathcal{A}}(t) = \sigma^{\mathcal{A}}(u)$ for all $\sigma \in \mathrm{ASSIGN}_{\mathcal{A}}$. ▯

Theorem 4.5.8. *Let $\mathcal{L}$ be a first-order language, t, t', u' be $\mathcal{L}$-terms, and let $\mathcal{A}$ be an $\mathcal{L}$-structure. If $t' \equiv_{\mathcal{A}} u'$ and u is obtained from t by replacing an occurrence of t' by u', then $t \equiv_{\mathcal{A}} u$.*

Proof. If $t' = t$, then $u = u'$ and the result is immediate. The proof is by induction on t.

If t is a constant symbol, then we are in the initial case.

Suppose now that $t = f(t_0, \ldots, t_{n-1})$ and the result holds for $t_0, \ldots, t_{n-1}$. If we are not in the initial case, then by Theorem 1.5.27, the occurrence of t' in t is part of a t_i for some i with $0 \leq i \leq n-1$. Thus, we have $u = f(t_0, \ldots, t_{i-1}, u_i, t_{i+1}, \ldots, t_{n-1})$, where u_i is obtained from t_i by replacing an occurrence of t' by u'. By the inductive hypothesis, $t_i \equiv_{\mathcal{A}} u_i$. Thus, for any $\sigma \in \mathrm{ASSIGN}_{\mathcal{A}}$, we have

$$\begin{aligned}\sigma^{\mathcal{A}}(t) &= f^{\mathcal{A}}(\sigma^{\mathcal{A}}(t_0), \ldots, \sigma^{\mathcal{A}}(t_{n-1}))\\ &= f^{\mathcal{A}}(\sigma^{\mathcal{A}}(t_0), \ldots, \sigma^{\mathcal{A}}(t_{i-1}), \sigma^{\mathcal{A}}(u_i), \sigma^{\mathcal{A}}(t_{i+1}), \ldots, \sigma^{\mathcal{A}}(t_{n-1}))\\ &= \sigma^{\mathcal{A}}(f(t_0, \ldots, t_{i-1}, u_i, t_{i+1}, \ldots, t_{n-1}))\\ &= \sigma^{\mathcal{A}}(u).\end{aligned}$$

Thus, $t \equiv_{\mathcal{A}} u$. □

4.5.2 *Tarski's Definition of Truth*

The "definition of truth", due to Tarski[4], was an important milestone in the development of logic. By present day standards, it seems to be a formalization of an intuitively clear idea; at the time, however, it clarified the distinction between the syntactic and semantic aspects of first-order logic.

It is easy to verify that if $a_1 \neq a_2$, then

$$[a_2 \to b_2][a_1 \to b_1]f = [a_1 \to b_1][a_2 \to b_2]f.$$

When f is the empty function, we write $[a \to b]$ instead of $[a \to b]f$. This notation will be used in Chapter 6.

Definition 4.5.9. Let $\mathcal{L}$ be a first-order language and let $\mathcal{A}$ be an $\mathcal{L}$-structure. We define a function $\mathcal{S}_{\mathcal{A}} : \mathrm{FORM}_{\mathcal{L}} \longrightarrow (\mathrm{ASSIGN}_{\mathcal{A}} \longrightarrow$ **Bool**) by the following recursive definition, where σ is an arbitrary assignment over $\mathcal{A}$.

(1) For every propositional constant R of $\mathcal{L}$,

$$\mathcal{S}_{\mathcal{A}}\mathbf{(}R\mathbf{)}(\sigma) = R^{\mathcal{A}}.$$

[4]Alfred Tarski was born on January 14, 1902 in Warsaw, Poland and died on October 26, 1983 in Berkeley, California. Tarski received his doctorate in mathematics from the University of Warsaw in 1924 and was named docent at the same university in 1926. In 1939 he immigrated in the United States. Tarski was appointed a lecturer in mathematics at the University of California at Berkeley in 1942 and taught there as a professor of mathematics after 1946. His contributions are in logic, set theory, and algebra.

(2) For each relation symbol R of $\mathcal{L}$ of positive arity and all $\mathcal{L}$-terms $t_0, \ldots, t_{n-1}$ (where n is the arity of R),

$$\mathcal{S}_{\mathcal{A}}\boldsymbol{(}R(t_0, \ldots, t_{n-1})\boldsymbol{)}(\sigma) = \begin{cases} \mathbf{T} & \text{if } (\sigma^{\mathcal{A}}(t_0), \ldots, \sigma^{\mathcal{A}}(t_{n-1})) \in R^{\mathcal{A}} \\ \mathbf{F} & \text{otherwise.} \end{cases}$$

(3) For all formulas φ of $\mathcal{L}$,

$$\mathcal{S}_{\mathcal{A}}\boldsymbol{(}(\neg\varphi)\boldsymbol{)}(\sigma) = f_{\neg}(\mathcal{S}_{\mathcal{A}}\boldsymbol{(}\varphi\boldsymbol{)}(\sigma)).$$

(4) For all formulas φ, ψ of $\mathcal{L}$ and binary connective symbols C,

$$\mathcal{S}_{\mathcal{A}}\boldsymbol{(}(\varphi C \psi)\boldsymbol{)}(\sigma) = f_C(\mathcal{S}_{\mathcal{A}}\boldsymbol{(}\varphi\boldsymbol{)}(\sigma), \mathcal{S}_{\mathcal{A}}\boldsymbol{(}\psi\boldsymbol{)}(\sigma)).$$

(5) For every formula φ of $\mathcal{A}$ and variable x,

$$\mathcal{S}_{\mathcal{A}}\boldsymbol{(}(\forall x)\varphi\boldsymbol{)}(\sigma) = \begin{cases} \mathbf{T} & \text{if } \mathcal{S}_{\mathcal{A}}\boldsymbol{(}\varphi\boldsymbol{)}([x \to a]\sigma) = \mathbf{T} \text{ for every } a \in |\mathcal{A}| \\ \mathbf{F} & \text{otherwise.} \end{cases}$$

(6) For every formula φ of $\mathcal{A}$ and variable x,

$$\mathcal{S}_{\mathcal{A}}\boldsymbol{(}(\exists x)\varphi\boldsymbol{)}(\sigma) = \begin{cases} \mathbf{T} & \text{if } \mathcal{S}_{\mathcal{A}}\boldsymbol{(}\varphi\boldsymbol{)}([x \to a]\sigma) = \mathbf{T} \text{ for some } a \in |\mathcal{A}| \\ \mathbf{F} & \text{otherwise.} \end{cases}$$

If $\mathcal{S}_{\mathcal{A}}\boldsymbol{(}\varphi\boldsymbol{)}(\sigma) = \mathbf{T}$, then we say that the pair $(\mathcal{A},\sigma)$ *satisfies* φ and write $(\mathcal{A}, \sigma) \models \varphi$. If $(\mathcal{A}, \sigma)$ does not satisfy φ, we write $(\mathcal{A}, \sigma) \not\models \varphi$.

Let Γ be a set of $\mathcal{L}$-formulas, $\mathcal{A}$ be an $\mathcal{L}$-structure, and σ be an assignment over $\mathcal{A}$. We say that $(\mathcal{A}, \sigma)$ *satisfies* Γ and write $(\mathcal{A}, \sigma) \models \Gamma$ if $(\mathcal{A}, \sigma) \models \varphi$ for every $\varphi \in \Gamma$. If $(\mathcal{A}, \sigma)$ does not satisfy Γ, we write $(\mathcal{A}, \sigma) \not\models \Gamma$. ▯

Using the notation introduced at the end of Definition 4.5.9, we can restate that definition as follows for every first-order language $\mathcal{L}$, $\mathcal{L}$-structure $\mathcal{A}$ and assignment σ over $\mathcal{A}$:

(1) For every propositional constant R of $\mathcal{L}$, $(\mathcal{A}, \sigma) \models R$ if and only if $R^{\mathcal{A}} = \mathbf{T}$.
(2) For each relation symbol R of $\mathcal{L}$ of positive arity and all $\mathcal{L}$-terms $t_0, \ldots, t_{n-1}$ (where n is the arity of R), $(\mathcal{A}, \sigma) \models R(t_0, \ldots, t_{n-1})$ if and only if $(\sigma^{\mathcal{A}}(t_0), \ldots, \sigma^{\mathcal{A}}(t_{n-1})) \in R^{\mathcal{A}}$.

(3) For all formulas φ of $\mathcal{L}$, $(\mathcal{A},\sigma) \models (\neg\varphi)$ if and only if $(\mathcal{A},\sigma) \not\models \varphi$.
(4) For all formulas φ, ψ of $\mathcal{L}$, we have:
   (a) $(\mathcal{A},\sigma) \models (\varphi \vee \psi)$ if and only if $(\mathcal{A},\sigma) \models \varphi$ or $(\mathcal{A},\sigma) \models \psi$.
   (b) $(\mathcal{A},\sigma) \models (\varphi \wedge \psi)$ if and only if $(\mathcal{A},\sigma) \models \varphi$ and $(\mathcal{A},\sigma) \models \psi$.
   (c) $(\mathcal{A},\sigma) \models (\varphi \to \psi)$ if and only if $(\mathcal{A},\sigma) \not\models \varphi$ or $(\mathcal{A},\sigma) \models \psi$.
   (d) $(\mathcal{A},\sigma) \models (\varphi \leftrightarrow \psi)$ if and only if either $(\mathcal{A},\sigma) \models \varphi$ and $(\mathcal{A},\sigma) \models \psi$ or $(\mathcal{A},\sigma) \not\models \varphi$ and $(\mathcal{A},\sigma) \not\models \psi$.
(5) For every formula φ of $\mathcal{A}$ and variable x, $(\mathcal{A},\sigma) \models (\forall x)\varphi$ if and only if $(\mathcal{A},[x \to a]\sigma) \models \varphi$ for every $a \in |\mathcal{A}|$.
(6) For every formula φ of $\mathcal{A}$ and variable x, $(\mathcal{A},\sigma) \models (\exists x)\varphi$ if and only if $(\mathcal{A},[x \to a]\sigma) \models \varphi$ for some $a \in |\mathcal{A}|$.

Theorem 4.5.10. *Let $\mathcal{L}$ be a first-order language, $(\varphi_0, \ldots, \varphi_{n-1})$ be a nonempty sequence of $\mathcal{L}$-formulas, $\mathcal{A}$ be an $\mathcal{L}$-structure and σ be an assignment in* $\mathrm{ASSIGN}_{\mathcal{A}}$*. We have $(\mathcal{A},\sigma) \models \bigvee_{0\le i\le n-1} \varphi_i$ if and only $(\mathcal{A},\sigma) \models \varphi_i$ for some i, $0 \le i \le n-1$. Also, $(\mathcal{A},\sigma) \models \bigwedge_{0\le i\le n-1} \varphi_i$ if and only $(\mathcal{A},\sigma) \models \varphi_i$ for every i, $0 \le i \le n-1$.*

Proof. The argument is by induction on n, and is left to the reader. $\square$

Example 4.5.11. Let $\mathcal{L}_{ar}$ be the first-order language introduced in Example 4.2.3 and let $\varphi = (\exists x_0)(f_1^2(x_0,x_0) = x_1)$, where f_1^2 denotes formally multiplication instead of $\cdot$. Note that $\mathtt{FV}(\varphi) = \{x_1\}$. Let $\mathcal{A}$ be the $\mathcal{L}_{ar}$-structure $(\mathbf{N},\mathcal{I})$ introduced in Example 4.4.4 and let $\sigma \in \mathrm{ASSIGN}_{\mathcal{A}}$. We give this time only the full details of the computation of $\mathcal{S}_{\mathcal{A}}\boldsymbol{(}\varphi\boldsymbol{)}(\sigma)$. In this computation, the atomic formula $(f_1^2(x_0,x_0) = x_1)$ is denoted by ψ. We have:

$$\begin{aligned}
&\mathcal{S}_{\mathcal{A}}\boldsymbol{(}\varphi\boldsymbol{)}(\sigma)\\
&\quad= \begin{cases} \mathbf{T} & \text{if } \mathcal{S}_{\mathcal{A}}\boldsymbol{(}\psi\boldsymbol{)}([x_0 \to n]\sigma) = \mathbf{T} \text{ for some } n \in \mathbf{N} \\ \mathbf{F} & \text{otherwise} \end{cases}\\
&\quad= \begin{cases} \mathbf{T} & \text{if } (([x_0 \to n]\sigma)^{\mathcal{A}}(f_1^2(x_0,x_0)), ([x_0 \to n]\sigma)^{\mathcal{A}}(x_1)) \in =^{\mathcal{A}} \\ & \text{for some } n \in \mathbf{N} \\ \mathbf{F} & \text{otherwise} \end{cases}\\
&\quad= \begin{cases} \mathbf{T} & \text{if } ([x_0 \to n]\sigma)^{\mathcal{A}}(f_1^2(x_0,x_0)) = ([x_0 \to n]\sigma)^{\mathcal{A}}(x_1) \\ & \text{for some } n \in \mathbf{N} \\ \mathbf{F} & \text{otherwise} \end{cases}
\end{aligned}$$

Applying Definition 4.5.2, we have

$$\begin{aligned}
&([x_0 \to n]\sigma)^{\mathcal{A}}(f_1^2(x_0, x_0))\\
&\quad = (f_1^2)^{\mathcal{A}}(([x_0 \to n]\sigma)^{\mathcal{A}}(x_0), ([x_0 \to n]\sigma)^{\mathcal{A}}(x_0))\\
&\quad = (f_1^2)^{\mathcal{A}}(([x_0 \to n]\sigma)(x_0), ([x_0 \to n]\sigma)(x_0))\\
&\quad = n \cdot n.
\end{aligned}$$

The same definition yields

$$([x_0 \to n]\sigma)^{\mathcal{A}}(x_1) = ([x_0 \to n]\sigma)(x_1) = \sigma(x_1).$$

This allows us to conclude that

$$\mathcal{S}_{\mathcal{A}}\boldsymbol{(}\varphi\boldsymbol{)}(\sigma) = \begin{cases} \mathbf{T} & \text{if } n \cdot n = \sigma(x_1) \text{ for some } n \in \mathbf{N} \\ \mathbf{F} & \text{otherwise.} \end{cases}$$

In other words, $(\mathcal{A}, \sigma) \models \varphi$ if and only if $\sigma(x_1)$ is a perfect square.

For the structure $\mathcal{B} = (\mathcal{P}(M), \mathcal{J})$ considered in the same example, a similar analysis shows that

$$\mathcal{S}_{\mathcal{B}}\boldsymbol{(}\varphi\boldsymbol{)}(\sigma) = \begin{cases} \mathbf{T} & \text{if } K \cap K = \sigma(x_1) \text{ for some } K \in \mathcal{P}(M) \\ \mathbf{F} & \text{otherwise.} \end{cases}$$

Since $\sigma(x_1) = \sigma(x_1) \cap \sigma(x_1)$ for $\sigma \in \text{ASSIGN}_{\mathcal{B}}$, we conclude that $(\mathcal{B}, \sigma) \models \varphi$ for every $\sigma \in \text{ASSIGN}_{\mathcal{B}}$. □

Note that in the previous example, the value of $\mathcal{S}_{\mathcal{A}}\boldsymbol{(}\varphi\boldsymbol{)}(\sigma)$ depends only on the value of $\sigma(x_1)$, that is, on the value of σ on the unique free variable of φ. This is formalized in the next theorem.

Theorem 4.5.12 (Agreement Theorem for First-Order Logic). *Let $\mathcal{L}$ be a first-order language, $\mathcal{A}$ be an $\mathcal{L}$-structure, φ be a formula, and let σ, τ be two assignments such that $\sigma\restriction\mathtt{FV}(\varphi) = \tau\restriction\mathtt{FV}(\varphi)$. Then, $\mathcal{S}_{\mathcal{A}}\boldsymbol{(}\varphi\boldsymbol{)}(\sigma) = \mathcal{S}_{\mathcal{A}}\boldsymbol{(}\varphi\boldsymbol{)}(\tau)$.*

Proof. The proof is by induction on the formula φ. If φ is a propositional constant, then the truth value of both $\mathcal{S}_{\mathcal{A}}\boldsymbol{(}\varphi\boldsymbol{)}(\sigma)$ and $\mathcal{S}_{\mathcal{A}}\boldsymbol{(}\varphi\boldsymbol{)}(\tau)$ does not depend on the assignments σ and τ, respectively, and we have trivially $\mathcal{S}_{\mathcal{A}}\boldsymbol{(}\varphi\boldsymbol{)}(\sigma) = \mathcal{S}_{\mathcal{A}}\boldsymbol{(}\varphi\boldsymbol{)}(\tau)$.

Suppose that φ is $R(t_0, \ldots, t_{n-1})$, where R is a relation symbol of $\mathcal{L}$ of positive arity (say of arity n) and $t_0, \ldots, t_{n-1}$ are all terms of $\mathcal{L}$. Since $\mathtt{FV}(R(t_0, \ldots, t_{n-1}))$ consists of all variables that occur in $t_0, \ldots, t_{n-1}$, σ and τ coincide on all variables that occur in every term t_i for $0 \leq i \leq n-1$. Theorem 4.5.3 implies that $\sigma^{\mathcal{A}}(t_i) = \tau^{\mathcal{A}}(t_i)$ for $0 \leq i \leq n-1$. Therefore,

$$\begin{aligned}
\mathcal{S}_{\mathcal{A}}\big(R(t_0, \ldots, t_{n-1})\big)(\sigma) &= \begin{cases} \mathbf{T} & \text{if } (\sigma^{\mathcal{A}}(t_0), \ldots, \sigma^{\mathcal{A}}(t_{n-1})) \in R^{\mathcal{A}} \\ \mathbf{F} & \text{otherwise} \end{cases} \\
&= \begin{cases} \mathbf{T} & \text{if } (\tau^{\mathcal{A}}(t_0), \ldots, \tau^{\mathcal{A}}(t_{n-1})) \in R^{\mathcal{A}} \\ \mathbf{F} & \text{otherwise} \end{cases} \\
&= \mathcal{S}_{\mathcal{A}}\big(R(t_0, \ldots, t_{n-1})\big)(\tau)
\end{aligned}$$

Let φ be the formula $(\neg\psi)$ and assume that $\sigma \restriction \mathtt{FV}(\varphi) = \tau \restriction \mathtt{FV}(\varphi)$. Since $\mathtt{FV}(\psi) = \mathtt{FV}(\varphi)$ we have $\sigma \restriction \mathtt{FV}(\psi) = \tau \restriction \mathtt{FV}(\psi)$, so $\mathcal{S}_{\mathcal{A}}(\psi)(\sigma) = \mathcal{S}_{\mathcal{A}}(\psi)(\tau)$. This allows us to write

$$\begin{aligned}
\mathcal{S}_{\mathcal{A}}(\varphi)(\sigma) &= f_{\neg}(\mathcal{S}_{\mathcal{A}}(\psi)(\sigma)) \\
&= f_{\neg}(\mathcal{S}_{\mathcal{A}}(\psi)(\tau)) \text{ (by the inductive hypothesis)} \\
&= \mathcal{S}_{\mathcal{A}}(\varphi)(\tau).
\end{aligned}$$

The cases when φ has one of the forms $(\psi \wedge \psi')$, $(\psi \vee \psi')$, $(\psi \rightarrow \psi')$, and $(\psi \leftrightarrow \psi')$ are similar to the previous case and, therefore, are left to the reader.

Assume now that φ is $(\forall x)\psi$ and that $\sigma \restriction \mathtt{FV}(\varphi) = \tau \restriction \mathtt{FV}(\varphi)$. Suppose that $\mathcal{S}_{\mathcal{A}}(\varphi)(\sigma) = \mathcal{S}_{\mathcal{A}}((\forall x)\psi)(\sigma) = \mathbf{T}$. Then $\mathcal{S}_{\mathcal{A}}(\psi)([x \rightarrow a]\sigma) = \mathbf{T}$ for every $a \in |\mathcal{A}|$. The assignments σ and τ coincide on all free variables of φ, that is, on all free variables of ψ with the possible exception of x. This implies that $[x \rightarrow a]\sigma \restriction \mathtt{FV}(\psi) = [x \rightarrow a]\tau \restriction \mathtt{FV}(\psi)$ and, by the inductive hypothesis, we have

$$\mathcal{S}_{\mathcal{A}}(\psi)([x \rightarrow a]\sigma) = \mathcal{S}_{\mathcal{A}}(\psi)([x \rightarrow a]\tau) = \mathbf{T}$$

for every $a \in A$. Thus, we have $\mathcal{S}_{\mathcal{A}}(\varphi)(\tau) = \mathbf{T}$. By symmetry, $\mathcal{S}_{\mathcal{A}}(\varphi)(\tau) = \mathbf{T}$ implies $\mathcal{S}_{\mathcal{A}}(\varphi)(\sigma) = \mathbf{T}$, so $\mathcal{S}_{\mathcal{A}}(\varphi)(\sigma) = \mathcal{S}_{\mathcal{A}}(\varphi)(\tau)$. The case when φ is $(\exists x)\psi$ has a similar treatment and it is left to the reader. □

Corollary 4.5.13. *Let $\mathcal{L}$ be a first-order language and let $\mathcal{A}$ be an $\mathcal{L}$-structure.*

If $\varphi \in \mathrm{SENT}_{\mathcal{L}}$*, then either* $\mathcal{S}_{\mathcal{A}}\boldsymbol{(}\varphi\boldsymbol{)}(\sigma) = \mathbf{T}$ *for every* $\sigma \in \mathrm{ASSIGN}_{\mathcal{A}}$ *or* $\mathcal{S}_{\mathcal{A}}\boldsymbol{(}\varphi\boldsymbol{)}(\sigma) = \mathbf{F}$ *for every* $\sigma \in \mathrm{ASSIGN}_{\mathcal{A}}$*; in other words, either* $(\mathcal{A}, \sigma) \models \varphi$ *for every* $\sigma \in \mathrm{ASSIGN}_{\mathcal{A}}$ *or* $(\mathcal{A}, \sigma) \not\models \varphi$ *for every* $\sigma \in \mathrm{ASSIGN}_{\mathcal{A}}$*.*

Proof. This is an immediate consequence of Theorem 4.5.12. □

Let φ be an $\mathcal{L}$-sentence and let $\mathcal{A}$ be an $\mathcal{L}$-structure. By Corollary 4.5.13, the value of $\mathcal{S}_{\mathcal{A}}\boldsymbol{(}\varphi\boldsymbol{)}(\sigma)$ is independent of the assignment σ. Therefore, we are justified in denoting this truth value by $\varphi^{\mathcal{A}}$.

4.5.3 *Validity*

Definition 4.5.14. Let $\mathcal{L}$ be a first-order language, $\mathcal{A}$ be an $\mathcal{L}$-structure and φ be a formula of $\mathcal{L}$. If $(\mathcal{A}, \sigma) \models \varphi$ for every assignment σ over $\mathcal{A}$, then we say that φ is *valid* in $\mathcal{A}$, or that $\mathcal{A}$ is a *model* of φ, and write $\mathcal{A} \models \varphi$.

An $\mathcal{L}$-structure $\mathcal{A}$ is a *model* of Γ if $\mathcal{A}$ is a model of every formula in Γ. This is denoted by $\mathcal{A} \models \Gamma$.

If φ is an $\mathcal{L}$-formula which is valid in all $\mathcal{L}$-structures, then we call φ *logically valid* and write $\models \varphi$. ▯

Example 4.5.15. Example 4.5.11 shows that $(\mathcal{B}, \sigma) \models \varphi$ for every $\sigma \in \mathrm{ASSIGN}_{\mathcal{B}}$, so $\mathcal{B} \models \varphi$, that is, φ is valid in $\mathcal{B}$. However, φ is not logically valid because it is not valid in $\mathcal{A}$. Indeed, if $\sigma \in \mathrm{ASSIGN}_{\mathcal{A}}$ is such that $\sigma(x_1)$ is not a perfect square, then $(\mathcal{A}, \sigma) \not\models \varphi$. ▯

If φ is an $\mathcal{L}$-sentence and $\mathcal{A}$ is an $\mathcal{L}$-structure, then, by Corollary 4.5.13, either $\mathcal{A} \models \varphi$ or $\mathcal{A} \models (\neg\varphi)$. It is easy to see that if φ is not a sentence, then we could have an $\mathcal{L}$-structure $\mathcal{A}$ such that neither $\mathcal{A} \models \varphi$ nor $\mathcal{A} \models (\neg\varphi)$. Indeed, we saw an illustration of this remark in Example 4.5.11.

Theorem 4.5.16. *Let* φ, ψ *be in* $\mathrm{SENT}_{\mathcal{L}}$*,* $t_0, \ldots, t_{n-1}$ *be ground terms of* $\mathcal{L}$*,* R *be an* n*-ary relation symbol in* $\mathcal{L}$ *with* $n > 0$*, and* $\mathcal{A}$ *be an* $\mathcal{L}$*-structure. Then,*

(1) $\mathcal{A} \models R(t_0, \ldots, t_{n-1})$ *if and only if* $(t_0^{\mathcal{A}}, \ldots, t_{n-1}^{\mathcal{A}}) \in R^{\mathcal{A}}$*;*
(2) $(\neg\varphi)^{\mathcal{A}} = f_{\neg}(\varphi^{\mathcal{A}})$*;*
(3) $(\varphi C \psi)^{\mathcal{A}} = f_C(\varphi^{\mathcal{A}}, \psi^{\mathcal{A}})$ *for every binary connective symbol* C*.*

Proof. We show only the third part of the theorem. Let $\sigma \in \text{ASSIGN}_{\mathcal{A}}$ be a fixed but arbitrary assignment. By applying Definition 4.5.9, we have

$$\begin{aligned}(\varphi C \psi)^{\mathcal{A}} &= \mathcal{S}_{\mathcal{A}}\boldsymbol{(}(\varphi C \psi)\boldsymbol{)}(\sigma)\\ &= f_C(\mathcal{S}_{\mathcal{A}}\boldsymbol{(}\varphi\boldsymbol{)}(\sigma), \mathcal{S}_{\mathcal{A}}\boldsymbol{(}\psi\boldsymbol{)}(\sigma))\\ &= f_C(\varphi^{\mathcal{A}}, \psi^{\mathcal{A}}).\end{aligned}$$

□

We can restate explicitly the last two parts of Theorem 4.5.16 as follows:

(1) $\mathcal{A} \models (\neg\varphi)$ if and only if $\mathcal{A} \not\models \varphi$;
(2) $\mathcal{A} \models (\varphi \vee \psi)$ if and only if $\mathcal{A} \models \varphi$ or $\mathcal{A} \models \psi$;
(3) $\mathcal{A} \models (\varphi \wedge \psi)$ if and only if $\mathcal{A} \models \varphi$ and $\mathcal{A} \models \psi$;
(4) $\mathcal{A} \models (\varphi \rightarrow \psi)$ if and only if $\mathcal{A} \models (\neg\varphi)$ or $\mathcal{A} \models \psi$;
(5) $\mathcal{A} \models (\varphi \leftrightarrow \psi)$ if and only if either $\mathcal{A} \models \varphi$ and $\mathcal{A} \models \psi$ or $\mathcal{A} \models (\neg\varphi)$ and $\mathcal{A} \models (\neg\psi)$.

Example 4.5.17. We will show that the formulas

$$\alpha = ((\exists x)(\varphi \vee \psi) \leftrightarrow ((\exists x)\varphi \vee (\exists x)\psi))$$
$$\beta = ((\forall x)(\varphi \wedge \psi) \leftrightarrow ((\forall x)\varphi \wedge (\forall x)\psi))$$

are logically valid for all formulas φ, ψ and all variables x.

For the first part, let $\mathcal{A}$ be a structure and let $\sigma \in \text{ASSIGN}_{\mathcal{A}}$. To show that $(\mathcal{A}, \sigma) \models \alpha$, we need to prove that $(\mathcal{A}, \sigma) \models (\exists x)(\varphi \vee \psi)$ if and only if $(\mathcal{A}, \sigma) \models ((\exists x)\varphi \vee (\exists x)\psi)$.

Suppose that $(\mathcal{A}, \sigma) \models (\exists x)(\varphi \vee \psi)$. This implies that $(\mathcal{A}, [x \rightarrow a]\sigma) \models (\varphi \vee \psi)$ for some $a \in |\mathcal{A}|$. Therefore, we have $(\mathcal{A}, [x \rightarrow a]\sigma) \models \varphi$ or $(\mathcal{A}, [x \rightarrow a]\sigma) \models \psi$, which amounts to $(\mathcal{A}, \sigma) \models (\exists x)\varphi$ or $(\mathcal{A}, \sigma) \models (\exists x)\psi$. Therefore, $(\mathcal{A}, \sigma) \models ((\exists x)\varphi \vee (\exists x)\psi)$.

Conversely, suppose that $(\mathcal{A}, \sigma) \models ((\exists x)\varphi \vee (\exists x)\psi)$. This means that we have either $(\mathcal{A}, \sigma) \models (\exists x)\varphi$ or $(\mathcal{A}, \sigma) \models (\exists x)\psi$. In the first case, there is $a \in |\mathcal{A}|$ such that $(\mathcal{A}, [x \rightarrow a]\sigma) \models \varphi$ and this implies that $(\mathcal{A}, [x \rightarrow a]\sigma) \models (\varphi \vee \psi)$. Therefore, $(\mathcal{A}, \sigma) \models (\exists x)(\varphi \vee \psi)$. The second case is entirely similar.

For the second part, suppose that $(\mathcal{A}, \sigma) \models (\forall x)(\varphi \wedge \psi)$. This means that for every $a \in |\mathcal{A}|$, $(\mathcal{A}, [x \rightarrow a]\sigma) \models (\varphi \wedge \psi)$, which is equivalent to saying that $(\mathcal{A}, [x \rightarrow a]\sigma) \models \varphi$ and $(\mathcal{A}, [x \rightarrow a]\sigma) \models \psi$.

This shows that $(\mathcal{A}, \sigma) \models (\forall x)\varphi$ and $(\mathcal{A}, \sigma) \models (\forall x)\psi$, so $(\mathcal{A}, \sigma) \models ((\forall x)\varphi \wedge (\forall x)\psi))$.

Conversely, suppose that $(\mathcal{A}, \sigma) \models ((\forall x)\varphi \wedge (\forall x)\psi))$. Then, $(\mathcal{A}, \sigma) \models (\forall x)\varphi$ and $(\mathcal{A}, \sigma) \models (\forall x)\psi$, so for all $a, b \in |\mathcal{A}|$, we have $(\mathcal{A}, [x \to a]\sigma) \models \varphi$ and $(\mathcal{A}, [x \to b]\sigma) \models \psi$. Choosing $a = b$, we have that $(\mathcal{A}, [x \to a]\sigma) \models (\varphi \wedge \psi)$ for every $a \in |\mathcal{A}|$, so $(\mathcal{A}, \sigma) \models (\forall x)(\varphi \wedge \psi)$. ∎

Example 4.5.18. let $\mathcal{L}$ be a first-order language that contains the binary relation symbol R and the unary function symbols f and g. We claim that the formula $((\forall x_0)(\forall x_1)R(x_0, x_1) \to R(f(x_0), g(x_1)))$ is logically valid. Let $(\mathcal{A}, \sigma)$ be such that $\sigma \in \text{ASSIGN}_{\mathcal{A}}$. We need to show that if $(\mathcal{A}, \sigma) \models (\forall x_0)(\forall x_1)R(x_0, x_1)$, then $(\mathcal{A}, \sigma) \models R(f(x_0), g(x_1))$. This follows from the following chain of statements, where each statement implies its successor.

(1) $(\mathcal{A}, \sigma) \models (\forall x_0)(\forall x_1)R(x_0, x_1)$;
(2) $(\mathcal{A}, [x_0 \to a_0]\sigma) \models (\forall x_1)R(x_0, x_1)$, for all $a_0 \in |\mathcal{A}|$;
(3) $(\mathcal{A}, [x_1 \to a_1][x_0 \to a_0]\sigma) \models R(x_0, x_1)$, for all $a_0, a_1 \in |\mathcal{A}|$;
(4) $(([x_1 \to a_1][x_0 \to a_0]\sigma)^{\mathcal{A}}(x_0), ([x_1 \to a_1][x_0 \to a_0]\sigma)^{\mathcal{A}}(x_1)) \in R^{\mathcal{A}}$, for all $a_0, a_1 \in |\mathcal{A}|$;
(5) $(a_0, a_1) \in R^{\mathcal{A}}$, for all $a_0, a_1 \in |\mathcal{A}|$;
(6) $(f^{\mathcal{A}}(\sigma^{\mathcal{A}}(x_0)), g^{\mathcal{A}}(\sigma^{\mathcal{A}}(x_1))) \in R^{\mathcal{A}}$;
(7) $(\mathcal{A}, \sigma) \models R(f(x_0), g(x_1))$.

Similarly, the formula $((\forall x_0)(\forall x_1)R(x_0, x_1) \to R(f(x_1), g(x_0)))$ is logically valid. ∎

Example 4.5.19. Let φ be a formula of a first-order language $\mathcal{L}$. We claim that the formula $\psi = ((\forall x)\varphi \to (\exists x)\varphi)$ is logically valid for every variable x. Indeed, let $\mathcal{A}$ be an $\mathcal{L}$-structure and let $\sigma \in \text{ASSIGN}_{\mathcal{A}}$. To show that $(\mathcal{A}, \sigma) \models \psi$, it suffices to show that if $(\mathcal{A}, \sigma) \models (\forall x)\varphi$, then $(\mathcal{A}, \sigma) \models (\exists x)\varphi$. $(\mathcal{A}, \sigma) \models (\forall x)\varphi$ means that for all $a \in |\mathcal{A}|$, $(\mathcal{A}, [x \to a]\sigma) \models \varphi$. By the definition of structure, $|\mathcal{A}| \neq \emptyset$, so there is $a_0 \in |\mathcal{A}|$ such that $(\mathcal{A}, [x \to a_0]\sigma) \models \varphi$, which shows that $(\mathcal{A}, \sigma) \models (\exists x)\varphi$. ∎

Example 4.5.20. Let $\mathcal{L} = \{f, =\}$, where f is a binary function symbol. Clearly, $\mathcal{L}$ is an algebraic language and an $\mathcal{L}$-algebra is commonly called a *groupoid*. The formula

$$\varphi = (\forall x)(\forall y)(((\forall z)(f(x,z) = z) \wedge (\forall w)(f(w,y) = w)) \to (x = y)),$$

where x, y, z are distinct variables, expresses the fact that in a groupoid every left identity is equal to every right identity.

We prove that φ is logically valid. To this end, consider the following sequence of equivalent statements which involve an $\mathcal{L}$-structure $\mathcal{A}$ and an assignment $\sigma \in \text{ASSIGN}_{\mathcal{A}}$:

(1) $(\mathcal{A}, \sigma) \models \varphi$;
(2) $(\mathcal{A}, [x \to a]\sigma) \models (\forall y)(((\forall z)(f(x,z) = z) \wedge (\forall w)(f(w,y) = w)) \to (x = y))$ for all $a \in |\mathcal{A}|$;
(3) $(\mathcal{A}, [y \to b][x \to a]\sigma) \models (((\forall z)(f(x,z) = z) \wedge (\forall w)(f(w,y) = w)) \to (x = y))$ for all $a, b \in |\mathcal{A}|$;
(4) if $(\mathcal{A}, [y \to b][x \to a]\sigma) \models ((\forall z)(f(x,z) = z) \wedge (\forall w)(f(w,y) = w))$, then $(\mathcal{A}, [y \to b][x \to a]\sigma) \models (x = y)$, for all $a, b \in |\mathcal{A}|$;
(5) if $(\mathcal{A}, [y \to b][x \to a]\sigma) \models (\forall z)(f(x,z) = z)$ and $(\mathcal{A}, [y \to b][x \to a]\sigma) \models (\forall w)(f(w,y) = w)$, then $a = b$, for all $a, b \in |\mathcal{A}|$;
(6) if $(\mathcal{A}, [z \to c][y \to b][x \to a]\sigma) \models (f(x,z) = z)$ for all $c \in |\mathcal{A}|$ and $(\mathcal{A}, [w \to d][y \to b][x \to a]\sigma) \models (f(w,y) = w)$ for all $d \in |\mathcal{A}|$, then $a = b$, for all $a, b \in |\mathcal{A}|$;
(7) if $f^{\mathcal{A}}(a,c) = c)$ for all $c \in |\mathcal{A}|$ and $f^{\mathcal{A}}(d,b) = d$ for all $d \in |\mathcal{A}|$, then $a = b$, for all $a, b \in |\mathcal{A}|$.

Assume that $f^{\mathcal{A}}(a,c) = c)$ for all $c \in |\mathcal{A}|$ and $f^{\mathcal{A}}(d,b) = d$ for all $d \in |\mathcal{A}|$. Choosing $c = b$ in the first equality and $d = a$ in the second, we obtain $f^{\mathcal{A}}(a,b) = b$ and $f^{\mathcal{A}}(a,b) = a$, respectively, so $a = b$. □

Example 4.5.21. Let $\mathcal{L} = \{P\}$, where P is a unary relation symbol. We prove the validity of the formula $\varphi = (\exists x)(P(x) \to (\forall x)P(x))$.

Consider the following sequence of equivalent statements which involve an $\mathcal{L}$-structure $\mathcal{A}$ and an assignment $\sigma \in \text{ASSIGN}_{\mathcal{A}}$:

(1) $(\mathcal{A}, \sigma) \models \varphi$;
(2) for some $a \in |\mathcal{A}|$, $(\mathcal{A}, [x \to a]\sigma) \models (P(x) \to (\forall x)P(x))$;
(3) for some $a \in |\mathcal{A}|$, either $(\mathcal{A}, [x \to a]\sigma) \not\models P(x)$ or $(\mathcal{A}, [x \to a]\sigma) \models (\forall x)P(x)$;
(4) for some $a \in |\mathcal{A}|$, either $(a) \notin P^{\mathcal{A}}$ or for all $b \in |\mathcal{A}|$, $(\mathcal{A}, [x \to b][x \to a]\sigma) \models P(x)$;

(5) for some $a \in |\mathcal{A}|$, either $(a) \notin P^{\mathcal{A}}$ or for all $b \in |\mathcal{A}|$, $(b) \in P^{\mathcal{A}}$,

which obviously holds. □

Example 4.5.22. Let $\mathcal{L}$ be a first-order language that contains a binary relation symbol R. We show that the formula $\varphi = ((\exists x)(\forall y)R(x,y) \to (\forall y)(\exists x)R(x,y))$ is logically valid, where x and y are distinct variables. Observe that we have the following four equivalent statements involving an $\mathcal{L}$-structure $\mathcal{A}$ and $\sigma \in \mathrm{ASSIGN}_{\mathcal{A}}$:

(1) $(\mathcal{A}, \sigma) \models (\exists x)(\forall y)R(x,y)$;
(2) for some $a \in |\mathcal{A}|$, $(\mathcal{A}, [x \to a]\sigma) \models (\forall y)R(x,y)$;
(3) for some $a \in |\mathcal{A}|$, for all $b \in |\mathcal{A}|$, $(\mathcal{A}, [y \to b][x \to a]\sigma) \models R(x,y)$;
(4) for some $a \in |\mathcal{A}|$, for all $b \in |\mathcal{A}|$, $(a,b) \in R^{\mathcal{A}}$.

The last statement implies that for all $b \in |\mathcal{A}|$, for some $a \in |\mathcal{A}|$, $(a,b) \in R^{\mathcal{A}}$. This, in turn, is equivalent to saying that $(\mathcal{A}, \sigma) \models (\forall y)(\exists x)R(x,y)$, as can be shown following an argument similar to the first part. Thus, we have shown that $(\mathcal{A}, \sigma) \models \varphi$. □

Example 4.5.23. We show that the $\mathcal{L}$-formula $((\forall x)(\varphi \to \psi) \to ((\forall x)\varphi \to (\forall x)\psi))$ is logically valid for all $\mathcal{L}$-formulas φ, ψ. To this end, we need to prove that if $(\mathcal{A}, \sigma) \models (\forall x)(\varphi \to \psi)$, then $(\mathcal{A}, \sigma) \models ((\forall x)\varphi \to (\forall x)\psi)$. The hypothesis means that for all $a \in |\mathcal{A}|$, if $(\mathcal{A}, [x \to a]\sigma) \models \varphi$, then $(\mathcal{A}, [x \to a]\sigma) \models \psi$. It follows from this that if $(\mathcal{A}, [x \to a]\sigma) \models \varphi$ for every $a \in |\mathcal{A}|$, then $(\mathcal{A}, [x \to a]\sigma) \models \psi$ for every $a \in |\mathcal{A}|$. Thus, if $(\mathcal{A}, \sigma) \models (\forall x)\varphi$, then $(\mathcal{A}, \sigma) \models (\forall x)\psi$ and this, in turn, implies that $(\mathcal{A}, \sigma) \models ((\forall x)\varphi \to (\forall x)\psi)$. □

A first-order formula φ is an $\mathcal{L}$-formula for more than one first-order language $\mathcal{L}$. This raises the possibility that φ could be logically valid with respect to a language $\mathcal{L}_0$ but not logically valid with respect to another language $\mathcal{L}_1$ (where φ is both an $\mathcal{L}_0$- and an $\mathcal{L}_1$-formula) since the notion of logical validity introduced in Definition 4.5.14 apparently depends on the first-order language $\mathcal{L}$. We will show that this is not the case.

Theorem 4.5.24. *Let $\mathcal{L}, \mathcal{L}'$ be two first-order languages such that $\mathcal{L} \subseteq \mathcal{L}'$. Suppose that $\mathcal{B}$ is an $\mathcal{L}'$-structure and that $\mathcal{A}$ is $RED_{\mathcal{L}}(\mathcal{B})$, the reduct of $\mathcal{B}$ to $\mathcal{L}$. Then, for every $\varphi \in \mathrm{FORM}_{\mathcal{L}}$ and $\sigma \in \mathrm{ASSIGN}_{\mathcal{A}} = \mathrm{ASSIGN}_{\mathcal{B}}$, we have $(\mathcal{A}, \sigma) \models \varphi$ if and only if $(\mathcal{B}, \sigma) \models \varphi$.*

Proof. The reader can show by induction on the terms t of $\mathcal{L}$ that $\sigma^{\mathcal{A}}(t) = \sigma^{\mathcal{B}}(t)$ for every $\sigma \in \text{ASSIGN}_{\mathcal{A}}$. Then, an argument by induction on formulas φ (also left to the reader) shows that $\mathcal{S}_{\mathcal{A}}\boldsymbol{(}\varphi\boldsymbol{)}(\sigma) = \mathcal{S}_{\mathcal{B}}\boldsymbol{(}\varphi\boldsymbol{)}(\sigma)$ for every $\sigma \in \text{ASSIGN}_{\mathcal{A}}$. $\square$

Corollary 4.5.25. *Let $\mathcal{L}, \mathcal{L}'$ be two first-order languages such that $\mathcal{L} \subseteq \mathcal{L}'$. If $\mathcal{B}$ is an $\mathcal{L}'$-structure and $\mathcal{A}$ is RED$_{\mathcal{L}}(\mathcal{B})$, then for every $\mathcal{L}$-sentence φ, we have $\mathcal{A} \models \varphi$ if and only if $\mathcal{B} \models \varphi$.*

Proof. The statement follows immediately from Theorem 4.5.24. $\square$

Corollary 4.5.26. *Let $\mathcal{L}$ be a first-order language, $\mathcal{A}$ and $\mathcal{A}'$ be two $\mathcal{L}$-structures with $|\mathcal{A}| = |\mathcal{A}'|$, φ be an $\mathcal{L}$-formula and σ be an assignment over $\mathcal{A}$. If $s^{\mathcal{A}} = s^{\mathcal{A}'}$ for all $s \in \mathcal{L}_{\varphi}$, then $(\mathcal{A}, \sigma) \models \varphi$ if and only if $(\mathcal{A}', \sigma) \models \varphi$.*

Proof. Let $\mathcal{B} = \text{RED}_{\mathcal{L}_{\varphi}}(\mathcal{A})$. We also have $\mathcal{B} = \text{RED}_{\mathcal{L}_{\varphi}}(\mathcal{A}')$ by hypothesis. Therefore, Theorem 4.5.24 implies that $(\mathcal{A}, \sigma) \models \varphi$ if and only if $(\mathcal{B}, \sigma) \models \varphi$ if and only if $(\mathcal{A}', \sigma) \models \varphi$. $\square$

Theorem 4.5.27. *Let $\mathcal{L}, \mathcal{L}'$ be two first-order languages such that $\mathcal{L} \subseteq \mathcal{L}'$. If $\varphi \in \text{FORM}_{\mathcal{L}}$, then φ is logically valid as an $\mathcal{L}$-formula if and only if φ is logically valid as an $\mathcal{L}'$-formula.*

Proof. Suppose that φ is not logically valid as an $\mathcal{L}'$ formula. Then, there is an $\mathcal{L}'$-structure $\mathcal{B}$ and an assignment $\sigma \in \text{ASSIGN}_{\mathcal{B}}$ such that $(\mathcal{B}, \sigma) \not\models \varphi$. If $\mathcal{A} = \text{RED}_{\mathcal{L}}(\mathcal{B})$, then, by Theorem 4.5.24, $(\mathcal{A}, \sigma) \not\models \varphi$, which implies that φ is not logically valid as an $\mathcal{L}$-formula.

Conversely, suppose that φ is not logically valid as an $\mathcal{L}$-formula. Then, there is an $\mathcal{L}$-structure $\mathcal{A}$ and an assignment $\sigma \in \text{ASSIGN}_{\mathcal{A}}$ such that $(\mathcal{A}, \sigma) \not\models \varphi$. If $\mathcal{B}$ is an extension of $\mathcal{A}$ to $\mathcal{L}'$, the same theorem implies that $(\mathcal{B}, \sigma) \not\models \varphi$, so φ is not logically valid as an $\mathcal{L}'$-formula. $\square$

Corollary 4.5.28. *Let $\mathcal{L}, \mathcal{L}'$ be two first-order languages and let φ be a formula in $\text{FORM}_{\mathcal{L}} \cap \text{FORM}_{\mathcal{L}'}$. Then, φ is logically valid as an $\mathcal{L}$-formula if and only if φ is logically valid as an $\mathcal{L}'$-formula.*

Proof. Observe that by a double application of Theorem 4.5.27, the following statements are equivalent.

- φ is logically valid as an $\mathcal{L}$-formula.
- φ is logically valid as an $\mathcal{L} \cap \mathcal{L}'$-formula.
- φ is logically valid as an $\mathcal{L}'$-formula.

□

Definition 4.5.29. Let $\mathcal{L}$ be a first-order language, let φ and ψ be two $\mathcal{L}$-formulas and let Γ be a set of $\mathcal{L}$-formulas.

- φ is *satisfiable in* $\mathcal{A}$, where $\mathcal{A}$ is an $\mathcal{L}$-structure, if there is $\sigma \in \mathrm{ASSIGN}_{\mathcal{A}}$ such that $(\mathcal{A}, \sigma) \models \varphi$. φ is *satisfiable* if it is satisfiable in some $\mathcal{L}$-structure; otherwise, we say that φ is *unsatisfiable*.
- φ $\mathcal{A}$*-implies* ψ (written $\varphi \models_{\mathcal{A}} \psi$) where $\mathcal{A}$ is an $\mathcal{L}$-structure, if every assignment σ over $\mathcal{A}$ which satisfies φ, also satisfies ψ.
- φ *logically implies* ψ (written $\varphi \models \psi$) if for every $\mathcal{L}$-structure $\mathcal{A}$ and assignment σ over $\mathcal{A}$ which satisfy φ, $\mathcal{A}$ and σ also satisfy ψ.
- φ is $\mathcal{A}$-equivalent to ψ (written $\varphi \equiv_{\mathcal{A}} \psi$), where $\mathcal{A}$ is an $\mathcal{L}$-structure, if for every $\sigma \in \mathrm{ASSIGN}_{\mathcal{A}}$, we have $(\mathcal{A}, \sigma) \models \varphi$ if and only if $(\mathcal{A}, \sigma) \models \psi$.
- φ is *logically equivalent* to ψ (written $\varphi \equiv \psi$) if for every $\mathcal{L}$-structure $\mathcal{A}$, φ is $\mathcal{A}$-equivalent to ψ; in other words, for every $\mathcal{L}$-structure $\mathcal{A}$ and assignment $\sigma \in \mathrm{ASSIGN}_{\mathcal{A}}$, we have $(\mathcal{A}, \sigma) \models \varphi$ if and only if $(\mathcal{A}, \sigma) \models \psi$.
- φ and ψ are *equisatisfiable* if either they are both satisfiable or neither of them is satisfiable.
- Γ is *satisfiable* in $\mathcal{A}$, where $\mathcal{A}$ is an $\mathcal{L}$-structure, if there is $\sigma \in \mathrm{ASSIGN}_{\mathcal{A}}$ such that $(\mathcal{A}, \sigma)$ satisfies Γ; Γ is *satisfiable* if it is satisfiable in some $\mathcal{L}$-structure; Γ is *unsatisfiable* if it is not satisfiable; Γ is *finitely satisfiable* if every finite subset of Γ is satisfiable.
- Γ *logically implies* ψ (written $\Gamma \models \psi$) if every $\mathcal{L}$-structure $\mathcal{A}$ and assignment σ over $\mathcal{A}$ which satisfy Γ also satisfy ψ.

▯

The symbol "$\models$" now has two meanings: when written as $(\mathcal{A}, \sigma) \models \varphi$, it means that φ is satisfied by $(\mathcal{A}, \sigma)$; in $\Gamma \models \psi$ or in $\varphi \models \psi$, it means that Γ or φ, respectively, logically implies ψ. The context will differentiate clearly between these meanings.

Example 4.5.30. Let $\mathcal{L}$ be a first-order language that contains a binary relation symbol R and let t_0, t_1 be two $\mathcal{L}$-terms. If w is a variable that does not occur in either t_0 or t_1, we have

$$R(t_0, t_1) \equiv_{\mathcal{A}} (\forall w)(R(t_0, w) \leftrightarrow R(t_1, w))$$

for every $\mathcal{L}$-structure $\mathcal{A}$ where $R^{\mathcal{A}}$ is an equivalence relation on $|\mathcal{A}|$.

Indeed, we have $(\mathcal{A}, \sigma) \models R(t_0, t_1)$ if and only if $(\sigma^{\mathcal{A}}(t_0), \sigma^{\mathcal{A}}(t_1)) \in R^{\mathcal{A}}$. By elementary properties of equivalence relations, this amounts to saying that we have

$$\text{for all } a \in |\mathcal{A}|, (\sigma^{\mathcal{A}}(t_0), a) \in \mathbf{R}^{\mathcal{A}} \text{ if and only if } (\sigma^{\mathcal{A}}(t_1), a) \in \mathbf{R}^{\mathcal{A}}. \tag{4.5}$$

Note that for all $a \in |\mathcal{A}|$, σ and $[w \to a]\sigma$ agree on all variables that occur in t_0 and t_1 because w occurs in neither t_0 nor t_1. Thus, (4.5) is equivalent to

$$\begin{aligned} &\text{for all } a \in |\mathcal{A}|, (([w \to a]\sigma)^{\mathcal{A}}(t_0), a) \in \mathbf{R}^{\mathcal{A}} \\ &\quad\text{if and only if } \ (([w \to a]\sigma)^{\mathcal{A}}(t_1), a) \in \mathbf{R}^{\mathcal{A}}. \end{aligned} \tag{4.6}$$

Finally, (4.6) is equivalent to $(\mathcal{A}, \sigma) \models (\forall w)(R(t_0, w) \leftrightarrow R(t_1, w))$. □

An alternative definition of logical implication which is frequently used is given next.

Definition 4.5.31. Let $\mathcal{L}$ be a first-order language, φ be an $\mathcal{L}$-formula, and let Γ be a set of $\mathcal{L}$-formulas. We write $\Gamma \mathrel{|\!\approx} \varphi$ if every $\mathcal{L}$-structure that is a model of Γ is also a model of φ. We refer to the relation $\mathrel{|\!\approx}$ as the *weak logical implication.* □

In Supplements 59 to 62, we discuss properties of weak logical implication and its relation to logical implication.

Using the notations and terminology introduced in Definitions 4.5.14 and 4.5.29, we can expand Corollary 4.5.13 as follows.

Corollary 4.5.32. *Let $\mathcal{L}$ be a first-order language and let $\mathcal{A}$ be an $\mathcal{L}$-structure. If $\varphi \in \mathrm{SENT}_{\mathcal{L}}$, then φ is satisfiable in $\mathcal{A}$ if and only if $\mathcal{A} \models \varphi$. Further, if $\Gamma \subseteq \mathrm{SENT}_{\mathcal{L}}$, then Γ is satisfiable in $\mathcal{A}$ if and only if $\mathcal{A}$ is a model of Γ and thus, Γ is satisfiable if and only if Γ has a model.*

Proof. The first part is a restatement of Corollary 4.5.13. The second part follows immediately from the first. □

Like the notion of logical validity, the notions introduced in Definition 4.5.29, do not depend on the particular first-order language considered. This is stated in the following results.

Theorem 4.5.33. *Let $\mathcal{L}, \mathcal{L}'$ be two first-order languages such that $\mathcal{L} \subseteq \mathcal{L}'$. Let $\varphi, \psi \in \mathrm{FORM}_{\mathcal{L}}$ and let Γ be a set of $\mathcal{L}$-formulas.*

(1) *φ is satisfiable as an $\mathcal{L}$-formula if and only if φ is satisfiable as an $\mathcal{L}'$-formula.*
(2) *φ logically implies ψ as $\mathcal{L}$-formulas if and only if φ logically implies ψ as $\mathcal{L}'$-formulas.*
(3) *φ is logically equivalent to ψ as $\mathcal{L}$-formulas if and only if they are logically equivalent as $\mathcal{L}'$-formulas.*
(4) *Γ is satisfiable as a set of $\mathcal{L}$-formulas if and only if it is satisfiable as a set of $\mathcal{L}'$-formulas.*
(5) *Γ logically implies ψ as a set of $\mathcal{L}$-formulas and an $\mathcal{L}$-formula, respectively, if and only if this implication holds between them considered as a set of $\mathcal{L}'$-formulas and an $\mathcal{L}'$-formula respectively.*

Proof. The proofs of all parts of this theorem are similar to the proof of Theorem 4.5.27. □

Corollary 4.5.34. *Let $\mathcal{L}, \mathcal{L}'$ be two first-order languages, φ, ψ be two formulas in $\mathrm{FORM}_{\mathcal{L}} \cap \mathrm{FORM}_{\mathcal{L}'}$ and let $\Gamma \subseteq \mathrm{FORM}_{\mathcal{L}} \cap \mathrm{FORM}_{\mathcal{L}'}$.*

(1) *φ is satisfiable as an $\mathcal{L}$-formula if and only if φ is satisfiable as an $\mathcal{L}'$-formula.*
(2) *φ logically implies ψ as $\mathcal{L}$-formulas if and only if φ logically implies ψ as $\mathcal{L}'$-formulas.*
(3) *φ is logically equivalent to ψ as $\mathcal{L}$-formulas if and only if they are logically equivalent as $\mathcal{L}'$-formulas.*
(4) *Γ is satisfiable as a set of $\mathcal{L}$-formulas if and only if it is satisfiable as a set of $\mathcal{L}'$-formulas.*
(5) *Γ logically implies ψ as a set of $\mathcal{L}$-formulas and an $\mathcal{L}$-formula, respectively, if and only if this implication holds between them considered as a set of $\mathcal{L}'$-formulas and an $\mathcal{L}'$-formula respectively.*

Proof. These statements follow from Theorem 4.5.33 in the same manner as Corollary 4.5.28 follows from Theorem 4.5.27. □

Example 4.5.35. Let $\mathcal{L}$ be a first-order language with equality. Recall that if x, y are variables, $x \neq y$ stands for the formula $(\neg(x = y))$. Consider the formula

$$\varphi = ((x \neq y \wedge y \neq z) \wedge x \neq z),$$

where x, y, z are distinct variables. Then, for every $\mathcal{L}$-structure $\mathcal{A}$, φ is satisfiable in $\mathcal{A}$ if and only if $|\mathcal{A}| \geq 3$.

Suppose that $|\mathcal{A}| \geq 3$. Then, there is $\sigma \in \mathrm{ASSIGN}_{\mathcal{A}}$ such that the elements $\sigma(x), \sigma(y), \sigma(z)$ are all distinct. It is immediate that $(\mathcal{A}, \sigma) \models \varphi$. Conversely, suppose that φ is satisfiable in $\mathcal{A}$, say $(\mathcal{A}, \sigma) \models \varphi$. Then, it is clear that $\sigma(x), \sigma(y), \sigma(z)$ are all distinct and this implies $|\mathcal{A}| \geq 3$. ▯

Example 4.5.36. The formula φ considered in the previous example is not valid in any $\mathcal{L}$-structure $\mathcal{A}$ because for an assignment $\sigma \in \mathrm{ASSIGN}_{\mathcal{A}}$ such that $\sigma(x) = \sigma(y) = \sigma(z)$ we have $(\mathcal{A}, \sigma) \not\models \varphi$. Thus, φ is an example of a formula that is satisfiable, but is not valid in any structure. ▯

Theorem 4.5.37. *Let φ be a formula. We have $(\forall x)\varphi \models \varphi$ for every variable x. If $x \notin \mathtt{FV}(\varphi)$, then $\varphi \models (\forall x)\varphi$.*

Proof. Suppose that φ is an $\mathcal{L}$-formula, $\mathcal{A}$ is an $\mathcal{L}$-structure and $\sigma \in \mathrm{ASSIGN}_{\mathcal{A}}$. If $(\mathcal{A}, \sigma) \models (\forall x)\varphi$, this means that for every $a \in |\mathcal{A}|$, we have $(\mathcal{A}, [x \to a]\sigma) \models \varphi$. In particular, $(\mathcal{A}, [x \to \sigma(x)]\sigma) \models \varphi$, so $(\mathcal{A}, \sigma) \models \varphi$, which gives the desired logical implication.

Suppose now that $x \notin \mathtt{FV}(\varphi)$ and that $(\mathcal{A}, \sigma) \models \varphi$. Note that for every $a \in |\mathcal{A}|$, $\sigma(z) = [x \to a]\sigma(z)$ for every $z \in \mathtt{FV}(\varphi)$, so, by the Agreement Theorem, $(\mathcal{A}, [x \to a]\sigma) \models \varphi$. Thus, $(\mathcal{A}, \sigma) \models (\forall x)\varphi$. □

Corollary 4.5.38. *For every formula φ, $\varphi^{\forall} \models \varphi$.*

Proof. This follows by repeated application of Theorem 4.5.37. □

Example 4.5.39. We show that the condition $x \notin \mathtt{FV}(\varphi)$ is essential for the second part of the theorem. Indeed, let φ be the formula $x = c$ where x is a variable and c is a constant symbol and let $\mathcal{A}$ be an $\{=, c\}$-structure such that $|\mathcal{A}| = \{a, b\}$ and $c^{\mathcal{A}} = a$. If $\sigma \in \mathrm{ASSIGN}_{\mathcal{A}}$ is such that $\sigma(x) = a$, then $(\mathcal{A}, \sigma) \models \varphi$, but $(\mathcal{A}, \sigma) \not\models (\forall x)\varphi$. ▯

Theorem 4.5.40. *Let φ be a formula and x be a variable. We have $\varphi \models (\exists x)\varphi$. If $x \notin \mathtt{FV}(\varphi)$, then $(\exists x)\varphi \models \varphi$.*

Proof. Suppose that φ is an $\mathcal{L}$-formula, $\mathcal{A}$ is an $\mathcal{L}$-structure and $\sigma \in \mathrm{ASSIGN}_{\mathcal{A}}$. If $(\mathcal{A}, \sigma) \models \varphi$, this means that $(\mathcal{A}, [x \to \sigma(x)]\sigma) \models \varphi$, so $(\mathcal{A}, \sigma) \models (\exists x)\varphi$, which gives the desired logical implication.

We leave to the reader the argument for the last part of the theorem. $\square$

Corollary 4.5.41. *For every formula φ, $\varphi \models \varphi^{\exists}$.*

Proof. This follows by repeated application of Theorem 4.5.40. $\square$

Example 4.5.42. The condition $x \notin \mathtt{FV}(\varphi)$ is essential for the second part of the theorem. Indeed, let φ be the formula $x = c$ where x is a variable and c is a constant symbol and let $\mathcal{A}$ be an $\{=, c\}$-structure such that $|\mathcal{A}| = \{a, b\}$ and $c^{\mathcal{A}} = a$. If $\sigma \in \mathrm{ASSIGN}_{\mathcal{A}}$ is such that $\sigma(x) = b$, then $(\mathcal{A}, \sigma) \models (\exists x)\varphi$, but $(\mathcal{A}, \sigma) \not\models \varphi$. ▯

Corollary 4.5.43. *Let φ be a formula and x be a variable such that $x \notin \mathtt{FV}(\varphi)$. Then, $\varphi \equiv (Qx)\varphi$, for $Q \in \{\forall, \exists\}$.*

Proof. The result follows immediately from Theorems 4.5.37 and 4.5.40. $\square$

Example 4.5.44. Let φ, ψ be formulas and x be a variable such that $x \notin \mathtt{FV}(\varphi)$. We will show that the formulas $(\varphi \to (\exists x)\psi)$ and $(\exists x)(\varphi \to \psi)$ are logically equivalent.

Suppose that φ, ψ are $\mathcal{L}$-formulas, that $\mathcal{A}$ is an $\mathcal{L}$-structure and that $\sigma \in \mathrm{ASSIGN}_{\mathcal{A}}$. If $(\mathcal{A}, \sigma) \models (\varphi \to (\exists x)\psi)$ two cases may occur.

`Case 1:` $(\mathcal{A}, \sigma) \not\models \varphi$. Then, $(\mathcal{A}, \sigma) \models (\varphi \to \psi)$, that is, $(\mathcal{A}, [x \to \sigma(x)]\sigma) \models (\varphi \to \psi)$, which means that $(\mathcal{A}, \sigma) \models (\exists x)(\varphi \to \psi)$.

`Case 2:` $(\mathcal{A}, \sigma) \models (\exists x)\psi$. Then, there is $a \in |\mathcal{A}|$ such that $(\mathcal{A}, [x \to a]\sigma) \models \psi$, which implies $(\mathcal{A}, [x \to a]\sigma) \models (\varphi \to \psi)$. This in turn gives $(\mathcal{A}, \sigma) \models (\exists x)(\varphi \to \psi)$.

Conversely, suppose that $(\mathcal{A}, \sigma) \models (\exists x)(\varphi \to \psi)$, so there is $a \in |\mathcal{A}|$ such that $(\mathcal{A}, [x \to a]\sigma) \models (\varphi \to \psi)$. Again, we need to consider two cases.

`Case 1:` $(\mathcal{A}, [x \to a]\sigma) \not\models \varphi$. Since $x \notin \mathtt{FV}(\varphi)$, the assignments σ and $[x \to a]\sigma$ agree on all free variables of φ. Therefore, by the Agreement Theorem, we have $(\mathcal{A}, \sigma) \not\models \varphi$, so $(\mathcal{A}, \sigma) \models (\varphi \to (\exists x)\psi)$.

`Case 2:` $(\mathcal{A}, [x \to a]\sigma) \models \psi$. This means that $(\mathcal{A}, \sigma) \models (\exists x)\psi$, which implies $(\mathcal{A}, \sigma) \models (\varphi \to (\exists x)\psi)$.

$\square$

Theorem 4.5.45. *Let φ, ψ and θ be formulas. Then,*

(1) (*a*) $\varphi \models \varphi$;
    (*b*) *if* $\varphi \models \psi$ *and* $\psi \models \theta$, *then* $\varphi \models \theta$;
(2) (*a*) $\varphi \equiv \varphi$;
    (*b*) *if* $\varphi \equiv \psi$, *then* $\psi \equiv \varphi$;
    (*c*) *if* $\varphi \equiv \psi$ *and* $\psi \equiv \theta$, *then* $\varphi \equiv \theta$;
(3) $\varphi \equiv \psi$ *if and only if* $\varphi \models \psi$ *and* $\psi \models \varphi$.

Proof. The arguments for all the parts are straightforward and are left to the reader. $\square$

Corollary 4.5.46. *Let φ be a formula and let x be a variable. If $x \notin$ $\mathtt{FV}(\varphi)$, then the formulas $\varphi, (\exists x)\varphi$ and $(\forall x)\varphi$ are logically equivalent.*

Proof. The corollary is a direct consequence of Theorems 4.5.37, 4.5.40 and 4.5.45. $\square$

Lemma 4.5.47. *Let $\mathcal{L}$ be a first-order language. If $\varphi_0, \varphi_1, \psi_0, \psi_1$ are $\mathcal{L}$-formulas and $\mathcal{A}$ is an $\mathcal{L}$-structure such that $\varphi_0 \models_{\mathcal{A}} \varphi_1$ and $\psi_0 \models_{\mathcal{A}} \psi_1$, and x is a variable, then*

$$(\neg\varphi_1) \models_{\mathcal{A}} (\neg\varphi_0)$$
$$(\varphi_0 \vee \psi_0) \models_{\mathcal{A}} (\varphi_1 \vee \psi_1)$$
$$(\varphi_0 \wedge \psi_0) \models_{\mathcal{A}} (\varphi_1 \wedge \psi_1)$$
$$(\varphi_1 \to \psi_0) \models_{\mathcal{A}} (\varphi_0 \to \psi_1)$$
$$(\forall x)\varphi_0 \models_{\mathcal{A}} (\forall x)\varphi_1$$
$$(\exists x)\varphi_0 \models_{\mathcal{A}} (\exists x)\varphi_1$$

Proof. We give the argument only for the last part of the theorem. Suppose that σ is an assignment, $\sigma \in \mathrm{ASSIGN}_{\mathcal{A}}$ such that $(\mathcal{A}, \sigma) \models (\exists x)\varphi_0$. Then, there is $a \in |\mathcal{A}|$ such that $(\mathcal{A}, [x \to a]\sigma) \models \varphi_0$. Since $\varphi_0 \models \varphi_1$, the previous statement implies the existence of an $a \in |\mathcal{A}|$ such that $(\mathcal{A}, [x \to a]\sigma) \models \varphi_1$. This amounts to $(\mathcal{A}, \sigma) \models (\exists x)\varphi_1$, so $(\exists x)\varphi_0 \models_{\mathcal{A}} (\exists x)\varphi_1$. $\square$

Theorem 4.5.48. *If $\varphi_0, \varphi_1, \psi_0, \psi_1$ are formulas such that $\varphi_0 \models \varphi_1$ and $\psi_0 \models \psi_1$, and x is a variable, then*

$$(\neg\varphi_1) \models (\neg\varphi_0)$$
$$(\varphi_0 \vee \psi_0) \models (\varphi_1 \vee \psi_1)$$
$$(\varphi_0 \wedge \psi_0) \models (\varphi_1 \wedge \psi_1)$$
$$(\varphi_1 \to \psi_0) \models (\varphi_0 \to \psi_1)$$
$$(\forall x)\varphi_0 \models (\forall x)\varphi_1$$
$$(\exists x)\varphi_0 \models (\exists x)\varphi_1$$

Proof. The statement follows immediately from Lemma 4.5.47. □

Lemma 4.5.49. *Let $\mathcal{L}$ be a first-order language and $\mathcal{A}$ be an $\mathcal{L}$-structure. If $\varphi_0, \varphi_1, \psi_0, \psi_1$ are $\mathcal{L}$-formulas such that $\varphi_0 \equiv_{\mathcal{A}} \varphi_1$ and $\psi_0 \equiv_{\mathcal{A}} \psi_1$, and x is a variable, then*

$$(\neg\varphi_0) \equiv_{\mathcal{A}} (\neg\varphi_1),$$
$$(\varphi_0 \vee \psi_0) \equiv_{\mathcal{A}} (\varphi_1 \vee \psi_1),$$
$$(\varphi_0 \wedge \psi_0) \equiv_{\mathcal{A}} (\varphi_1 \wedge \psi_1),$$
$$(\varphi_0 \to \psi_0) \equiv_{\mathcal{A}} (\varphi_1 \to \psi_1),$$
$$(\varphi_0 \leftrightarrow \psi_0) \equiv_{\mathcal{A}} (\varphi_1 \leftrightarrow \psi_1),$$
$$(\forall x)\varphi_0 \equiv_{\mathcal{A}} (\forall x)\varphi_1,$$
$$(\exists x)\varphi_0 \equiv_{\mathcal{A}} (\exists x)\varphi_1.$$

Proof. All statements, except for the fifth, are direct consequences of Lemma 4.5.47. We leave the fifth statement, involving $\leftrightarrow$, to the reader. □

Theorem 4.5.50. *If $\varphi, \varphi', \psi, \psi'$ are formulas such that $\varphi \equiv \varphi'$ and $\psi \equiv \psi'$, and x is a variable, then*

$$(\neg\varphi) \equiv (\neg\varphi'),$$
$$(\varphi \vee \psi) \equiv (\varphi' \vee \psi'),$$
$$(\varphi \wedge \psi) \equiv (\varphi' \wedge \psi'),$$
$$(\varphi \to \psi) \equiv (\varphi' \to \psi'),$$

$$(\varphi \leftrightarrow \psi) \equiv (\varphi' \leftrightarrow \psi'),$$
$$(\forall x)\varphi \equiv (\forall x)\varphi',$$
$$(\exists x)\varphi \equiv (\exists x)\varphi'.$$

Proof. The statement follows immediately from Lemma 4.5.49. □

Theorem 4.5.51. *Let φ, ψ and θ be formulas and let Γ, Γ' be sets of formulas. Then,*

(1) $\emptyset \models \varphi$ *if and only if* $\models \varphi$;
(2) $\{\varphi\} \models \psi$ *if and only if* $\varphi \models \psi$;
(3) *If* Γ' *is satisfiable and* $\Gamma \subseteq \Gamma'$, *then* Γ *is satisfiable*;
(4) $\{\varphi\}$ *is satisfiable if and only if* φ *is satisfiable*;
(5) *if* $\Gamma \models \varphi$ *and* $\Gamma \subseteq \Gamma'$, *then* $\Gamma' \models \varphi$.

Proof. We leave the simple arguments of the theorem to the reader. □

Theorem 4.5.52. *Let φ and ψ be formulas and let Γ be a set of formulas. Then,*

(1) $\Gamma \models \varphi$ *if and only if* $\Gamma \cup \{\neg\varphi\}$ *is unsatisfiable*;
(2) *if* $\Gamma \models \varphi$ *and* $\Gamma \models (\varphi \rightarrow \psi)$, *then* $\Gamma \models \psi$;
(3) $\Gamma \cup \{\varphi\} \models \psi$ *if and only if* $\Gamma \models (\varphi \rightarrow \psi)$.

Proof. The argument is left to the reader. □

Theorem 4.5.53. *Let Γ be a set of formulas. The following statements are equivalent:*

(1) Γ *is unsatisfiable*;
(2) $\Gamma \models \varphi$ *for every formula* φ;
(3) $\Gamma \models \varphi$ *for every contradiction* φ;
(4) $\Gamma \models \varphi$ *for some contradiction* φ.

Proof. The argument is similar to the one used in Theorem 2.3.18 and it is left to the reader. □

Theorem 4.5.54. *Let $\Gamma = \{\varphi_0, \ldots, \varphi_{n-1}\}$ be a nonempty, finite set of formulas. Then, Γ is unsatisfiable if and only if $((\neg\varphi_0) \vee \cdots \vee (\neg\varphi_{n-1}))$ is logically valid.*

Proof. The argument is straightforward and is left to the reader. □

Theorem 4.5.55. *Let φ, ψ be two formulas. Then, we have:*

(1) *φ is satisfiable if and only if $(\neg\varphi)$ is not logically valid.*
(2) *φ is logically valid if and only if $(\neg\varphi)$ is unsatisfiable.*
(3) *$\varphi \models \psi$ if and only if $(\varphi \to \psi)$ is logically valid.*
(4) *$\varphi \equiv \psi$ if and only if $(\varphi \leftrightarrow \psi)$ is logically valid.*

Proof. We will prove the third part of the theorem and leave the other three for the reader.

Let φ, ψ be two $\mathcal{L}$-formulas. Suppose that $\varphi \models \psi$ and let $\mathcal{A}$ be an $\mathcal{L}$-structure and σ be an assignment in $\text{ASSIGN}_{\mathcal{A}}$. There are two cases to consider. If $(\mathcal{A}, \sigma) \not\models \varphi$, then clearly $(\mathcal{A}, \sigma) \models (\varphi \to \psi)$. Otherwise, that is, if $(\mathcal{A}, \sigma) \models \varphi$, then $(\mathcal{A}, \sigma) \models \psi$, which implies again that $(\mathcal{A}, \sigma) \models (\varphi \to \psi)$. Therefore, $(\varphi \to \psi)$ is logically valid.

Conversely, suppose that $(\varphi \to \psi)$ is logically valid and let $\mathcal{A}$ be an $\mathcal{L}$-structure and $\sigma \in \text{ASSIGN}_{\mathcal{A}}$ such that $(\mathcal{A}, \sigma) \models \varphi$. The logical validity of $(\varphi \to \psi)$ implies that $(\mathcal{A}, \sigma) \models (\varphi \to \psi)$ from which it follows that $(\mathcal{A}, \sigma) \models \psi$. This shows that $\varphi \models \psi$. □

Corollary 4.5.56. *Let φ, ψ be two formulas and x be a variable. Then, we have*

$$(\exists x)(\varphi \vee \psi) \equiv ((\exists x)\varphi \vee (\exists x)\psi),$$
$$(\forall x)(\varphi \wedge \psi) \equiv ((\forall x)\varphi \wedge (\forall x)\psi).$$

Proof. These logical equivalences follow from the fourth part of Theorem 4.5.55 and Example 4.5.17. □

Theorem 4.5.57. *Let φ be a formula and x be a variable. Then, we have:*

$$(\exists x)\varphi \equiv (\neg(\forall x)(\neg\varphi))$$
$$(\forall x)\varphi \equiv (\neg(\exists x)(\neg\varphi))$$

Proof. Let φ be an $\mathcal{L}$-formula, $\mathcal{A}$ be an $\mathcal{L}$-structure and $\sigma \in \text{ASSIGN}_{\mathcal{A}}$. We have the following obviously equivalent statements:

- $(\mathcal{A}, \sigma) \models (\neg(\forall x)(\neg\varphi))$;
- $(\mathcal{A}, \sigma) \not\models (\forall x)(\neg\varphi)$;
- there is $a \in |\mathcal{A}|$ such that $(\mathcal{A}, [x \to a]\sigma) \not\models (\neg\varphi)$;

- there is $a \in |\mathcal{A}|$ such that $(\mathcal{A}, [x \to a]\sigma) \models \varphi$;
- $(\mathcal{A}, \sigma) \models (\exists x)\varphi$.

Thus, the first logical equivalence is shown. The second logical equivalence has a similar argument and is left to the reader. □

Theorem 4.5.58. *Let $\mathcal{L}$ be a first-order language, φ be an $\mathcal{L}$-formula, and $\mathcal{A}$ be an $\mathcal{L}$-structure. Then, the following hold.*

(1) *For every variable x, $\mathcal{A} \models \varphi$ if and only if $\mathcal{A} \models (\forall x)\varphi$.*
(2) *$\mathcal{A} \models \varphi$ if and only if $\mathcal{A} \models \varphi^{\forall}$; thus, $\models \varphi$ if and only if $\models \varphi^{\forall}$.*
(3) *For every variable x, φ is satisfiable in $\mathcal{A}$ if and only if $(\exists x)\varphi$ is satisfiable in $\mathcal{A}$.*
(4) *φ is satisfiable in $\mathcal{A}$ if and only if $\varphi^{\exists}$ is satisfiable in $\mathcal{A}$; thus, φ is satisfiable if and only if $\varphi^{\exists}$ is satisfiable.*

Proof. We showed in Theorem 4.5.37 that $(\forall x)\varphi \models \varphi$. Therefore, if $\mathcal{A} \models (\forall x)\varphi$, then $\mathcal{A} \models \varphi$. Conversely, suppose that $\mathcal{A} \models \varphi$. This means that for every assignment $\sigma \in \text{ASSIGN}_{\mathcal{A}}$, $(\mathcal{A}, \sigma) \models \varphi$, so for every σ and every $a \in |\mathcal{A}|$, $(\mathcal{A}, [x \to a]\sigma) \models \varphi$. This implies that $(\mathcal{A}, \sigma) \models (\forall x)\varphi$. This concludes the argument for Part (1). Part (2) is obtained by repeated application of Part (1).

In Theorem 4.5.40, we proved that $\varphi \models (\exists x)\varphi$. Therefore, if $(\mathcal{A}, \sigma) \models \varphi$, it is clear that $(\mathcal{A}, \sigma) \models (\exists x)\varphi$. Conversely, suppose that $(\mathcal{A}, \sigma) \models (\exists x)\varphi$. Then, there is $a \in |\mathcal{A}|$ such that $(\mathcal{A}, [x \to a]\sigma) \models \varphi$. This shows that φ is satisfiable in $\mathcal{A}$. This establishes Part (3). Finally, Part (4) follows by repeated application of Part (3). □

Corollary 4.5.59. *Let $\mathcal{L}$ be a first-order language, φ be an $\mathcal{L}$-formula, and ψ be a generalization of φ. Then, φ is logically valid if and only if ψ is logically valid.*

Proof. This follows immediately from the first part of Theorem 4.5.58. □

Corollary 4.5.60. *Let $\mathcal{L}$ be a first-order language, Γ be a set of $\mathcal{L}$-formulas and $\mathcal{A}$ be an $\mathcal{L}$-structure. Then, $\mathcal{A}$ is a model of Γ if and only if $\mathcal{A}$ is a model of $\Gamma^{\forall}$; thus, Γ has a model if and only if $\Gamma^{\forall}$ has a model.*

Proof. This follows immediately from Theorem 4.5.58. □

Another way of obtaining Corollary 4.5.59 is by showing the following statement that has independent interest.

Theorem 4.5.61. *Let $\mathcal{L}$ be a first-order language and Γ be a set of $\mathcal{L}$-formulas that contain no free occurrence of a variable x. Then, for all $\mathcal{L}$-formulas φ, $\Gamma \models \varphi$ if and only if $\Gamma \models (\forall x)\varphi$.*

Proof. By Theorem 4.5.37, it is clear that if $\Gamma \models (\forall x)\varphi$, then $\Gamma \models \varphi$. To prove the reverse direction, suppose that $\Gamma \models \varphi$. To show that $\Gamma \models (\forall x)\varphi$, suppose that $(\mathcal{A}, \sigma) \models \Gamma$. Then, by the Agreement Theorem, since x does not occur free in Γ, for all $a \in |\mathcal{A}|$, $(\mathcal{A}, [x \to a]\sigma) \models \Gamma$, so $(\mathcal{A}, [x \to a]\sigma) \models \varphi$, which implies that $(\mathcal{A}, \sigma) \models (\forall x)\varphi$.

$\square$

4.5.4 *Specification of Congruences*

In this subsection, we give a collection of closed formulas involving a binary relation symbol R such that the interpretation of R is a congruence in every model of these formulas.

Definition 4.5.62. Let $\mathcal{L}$ be a first-order language and let R be a binary relation symbol in $\mathcal{L}$. Let $\mathrm{MEq}_{R,\mathcal{L}}$ be the set of formulas consisting of:

- $R(x_0, x_0)$ (reflexivity);
- $(R(x_0, x_1) \to R(x_1, x_0))$ (symmetry);
- $((R(x_0, x_1) \wedge R(x_1, x_2)) \to R(x_0, x_2))$ (transitivity);
- for every n-ary function symbol $f \in \mathcal{L}$ with $n > 0$,

$$\begin{aligned}&((R(x_0, x_n) \wedge \cdots \wedge R(x_{n-1}, x_{2n-1}))\\ &\quad \to R(f(x_0, \ldots, x_{n-1}), f(x_n, \ldots, x_{2n-1})))\\ &\qquad \text{(function compatibility);}\end{aligned}$$

- for every n-ary relation symbol $P \in \mathcal{L}$ with $P \notin \{R, =\}$ and $n > 0$,

$$\begin{aligned}&((R(x_0, x_n) \wedge \cdots \wedge R(x_{n-1}, x_{2n-1}))\\ &\quad \to (P(x_0, \ldots, x_{n-1}) \leftrightarrow P(x_n, \ldots, x_{2n-1})))\\ &\qquad \text{(relation compatibility).}\end{aligned}$$

The set $\mathrm{Eq}_{R,\mathcal{L}}$ of $(R, \mathcal{L})$-*congruence axioms* consists of the universal closures of the formulas in $\mathrm{MEq}_{R,\mathcal{L}}$. The names of the axioms are indicated near their corresponding matrices. ▯

Observe that if $\mathcal{L}$ is finite, the set $\mathrm{MEq}_{R,\mathcal{L}}$ is finite.

Theorem 4.5.63. *Let $\mathcal{A}$ be an $\mathcal{L}$-structure and let R be a binary relation symbol in $\mathcal{L}$. Then, $\mathcal{A} \models Eq_{R,\mathcal{L}}$ if and only if $R^{\mathcal{A}}$ is a congruence of $\mathcal{A}$.*

Proof. The validity of the formulas in $\mathrm{Eq}_{R,\mathcal{L}}$ in $\mathcal{A}$ implies that $R^{\mathcal{A}}$ is an equivalence on $|\mathcal{A}|$, $R^{\mathcal{A}}$ is $f^{\mathcal{A}}$-compatible for every n-ary function symbol $f \in \mathcal{L}$ that is not a constant symbol, and $R^{\mathcal{A}}$ is $P^{\mathcal{A}}$-compatible for every n-ary relation symbol $P \in \mathcal{L} - \{R, =\}$ that is not a propositional constant. The $c^{\mathcal{A}}$-compatibility of $R^{\mathcal{A}}$ for each constant symbol $c \in \mathcal{L}$ follows from the fact that $R^{\mathcal{A}}$ is reflexive. The $P^{\mathcal{A}}$-compatibility of $R^{\mathcal{A}}$ for each propositional constant $P \in \mathcal{L}$ is immediate. Thus, assuming that R is not $=$, we need to prove that $R^{\mathcal{A}}$ is compatible with itself. Suppose that $(a_0, b_0), (a_1, b_1) \in R^{\mathcal{A}}$. If $(a_0, a_1) \in R^{\mathcal{A}}$, then, since $(b_0, a_0) \in R^{\mathcal{A}}$ by symmetry, we obtain $(b_0, b_1) \in R^{\mathcal{A}}$ by applying transitivity twice to the pairs $(b_0, a_0), (a_0, a_1), (a_1, b_1) \in R^{\mathcal{A}}$. Similarly, $(b_0, b_1) \in R^{\mathcal{A}}$ implies $(a_0, a_1) \in R^{\mathcal{A}}$.

If $R^{\mathcal{A}}$ is a congruence, it is immediate that $\mathcal{A} \models \mathrm{Eq}_{R,\mathcal{L}}$. □

In Section 5.8, we will need an equivalent form of the matrices of the congruence axioms, which we introduce next.

Definition 4.5.64. Let $\mathcal{L}$ be a first-order language and let R be a binary relation symbol in $\mathcal{L}$. The set of formulas ${\mathrm{MEq}^{\dagger}}_{R,\mathcal{L}}$ is the set of formulas consisting of:

- $R(x_0, x_0)$;
- $((\neg R(x_0, x_1)) \vee R(x_1, x_0))$;
- $((\neg R(x_0, x_1)) \vee (\neg R(x_1, x_2)) \vee R(x_0, x_2))$;
- for every n-ary function symbol $f \in \mathcal{L}$ with $n > 0$,

$$((\neg R(x_0, x_n)) \vee \cdots \vee (\neg R(x_{n-1}, x_{2n-1}))$$
$$\vee R(f(x_0, \ldots, x_{n-1}), f(x_n, \ldots, x_{2n-1})));$$

- for every n-ary relation symbol $P \in \mathcal{L}$ with $P \notin \{R, =\}$ and $n > 0$,

$$((\neg R(x_0, x_n)) \vee \cdots \vee (\neg R(x_{n-1}, x_{2n-1})) \\ \vee(\neg P(x_0, \ldots, x_{n-1})) \vee P(x_n, \ldots, x_{2n-1})),$$

  and

$$((\neg R(x_0, x_n)) \vee \cdots \vee (\neg R(x_{n-1}, x_{2n-1})) \\ \vee(\neg P(x_n, \ldots, x_{2n-1})) \vee P(x_0, \ldots, x_{n-1})).$$

∎

Theorem 4.5.65. *Let $\mathcal{L}$ be a first-order language and R be a binary relation symbol of $\mathcal{L}$. If $\mathcal{A}$ is an $\mathcal{L}$-structure, then the following statements are equivalent.*

(1) $\mathcal{A} \models Eq_{R,\mathcal{L}}$;
(2) $\mathcal{A} \models MEq_{R,\mathcal{L}}$;
(3) $\mathcal{A} \models MEq^{\dagger}_{R,\mathcal{L}}$.

Proof. We leave this simple proof to the reader. □

The set of $(=, \mathcal{L})$-congruence axioms is called the set of $\mathcal{L}$-*equality axioms*. Specifically, the $\mathcal{L}$-equality axioms are the universal closures of the following $\mathcal{L}$-formulas:

- $x_0 = x_0$;
- $(x_0 = x_1 \to x_1 = x_0)$;
- $((x_0 = x_1 \wedge x_1 = x_2) \to x_0 = x_2)$;
- for every n-ary function symbol $f \in \mathcal{L}$ with $n > 0$,

$$((x_0 = x_n \wedge \cdots \wedge x_{n-1} = x_{2n-1}) \\ \to f(x_0, \ldots, x_{n-1}) = f(x_n, \ldots, x_{2n-1}));$$

- for every n-ary relation symbol $P \in \mathcal{L}$ with $P \notin \{=\}$ and $n > 0$,

$$((x_0 = x_n \wedge \cdots \wedge x_{n-1} = x_{2n-1}) \\ \to (P(x_0, \ldots, x_{n-1}) \leftrightarrow P(x_n, \ldots, x_{2n-1}))).$$

Corollary 4.5.66. *Every $\mathcal{L}$-equality axiom is logically valid.*

Proof. Let $\mathcal{A}$ be an $\mathcal{L}$-structure. Since $=^{\mathcal{A}}$ is the equality relation on $|\mathcal{A}|$, which is a congruence, it follows by Theorem 4.5.63 that $\mathcal{A} \models \mathrm{Eq}_{=,\mathcal{L}}$. □

We conclude this subsection with a technical result that will be needed later.

Theorem 4.5.67. *Let $\mathcal{L}$ be a first-order language, R be a binary relation symbol in $\mathcal{L}$, and let $\mathcal{L}' = (\mathcal{L} - \{R\}) \cup \{R'\}$, where R' is a binary relation symbol such that $R' \notin \mathcal{L}$. Then,*

$$\mathsf{s}^{R}_{R'}(Eq_{R,\mathcal{L}}) = Eq_{R',\mathcal{L}'}.$$

Proof. This straightforward argument is left to the reader. □

4.5.5 *The Morphism Theorem*

Theorem 4.5.68 (The Morphism Theorem). *Let $\mathcal{L}$ be a first-order language, $\mathcal{A}, \mathcal{B}$ be two $\mathcal{L}$-structures and $h : |\mathcal{A}| \longrightarrow |\mathcal{B}|$ be a morphism.*

(1) *If φ is a quantifier-free $\mathcal{L}$-formula that has no occurrence of $=$, then, for every $\sigma \in \mathrm{ASSIGN}_{\mathcal{A}}$, we have $(\mathcal{A}, \sigma) \models \varphi$ if and only if $(\mathcal{B}, h \circ \sigma) \models \varphi$.*
(2) *If h is also an epimorphism, then Part (1) holds for all $\mathcal{L}$-formulas φ not containing $=$. Moreover, for each such φ, $\mathcal{A} \models \varphi$ if and only if $\mathcal{B} \models \varphi$.*
(3) *If h is an embedding, then Part (1) holds for all quantifier-free $\mathcal{L}$-formulas φ.*
(4) *If h is an isomorphism, then Part (1) holds for all $\mathcal{L}$-formulas φ. Further, we have $\mathcal{A} \models \varphi$ if and only if $\mathcal{B} \models \varphi$.*

Proof. We show Part (1) by induction on φ. If φ is a propositional constant R, then $(\mathcal{A}, \sigma) \models R$ if and only if $R^{\mathcal{A}} = \mathbf{T}$ which is equivalent to $R^{\mathcal{B}} = \mathbf{T}$. This in turn is equivalent to $(\mathcal{B}, h \circ \sigma) \models R$.

Suppose now that $\varphi = R(t_0, \ldots, t_{n-1})$ with $n > 0$ and R distinct from $=$. Then,

$$\begin{aligned}(\mathcal{A}, \sigma) \models \varphi \ \text{ if and only if } & \ (\sigma^{\mathcal{A}}(t_0), \ldots, \sigma^{\mathcal{A}}(t_{n-1})) \in R^{\mathcal{A}} \\ \text{if and only if } & \ (h(\sigma^{\mathcal{A}}(t_0)), \ldots, h(\sigma^{\mathcal{A}}(t_{n-1}))) \in R^{\mathcal{B}} \\ & \ (\text{since } h \text{ is a morphism})\end{aligned}$$

$$\begin{aligned}
&\text{if and only if} \quad ((h\circ\sigma)^{\mathcal{B}}(t_0),\dots,(h\circ\sigma)^{\mathcal{B}}(t_{n-1}))\in R^{\mathcal{B}}\\
&\qquad\qquad\qquad \text{(by Theorem 4.5.6)}\\
&\text{if and only if} \quad (\mathcal{B},h\circ\sigma)\models\varphi.
\end{aligned}$$

Suppose that the statement holds for the formula φ_0 and let $\varphi = (\neg\varphi_0)$. We have

$$\begin{aligned}
(\mathcal{A},\sigma)\models\varphi \quad &\text{if and only if} \quad (\mathcal{A},\sigma)\not\models\varphi_0\\
&\text{if and only if} \quad (\mathcal{B},h\circ\sigma)\not\models\varphi_0\\
&\text{if and only if} \quad (\mathcal{B},h\circ\sigma)\models\varphi.
\end{aligned}$$

We leave to the reader the remaining inductive steps when $\varphi = (\varphi_0 C \varphi_1)$ and the result holds for φ_0, φ_1.

The proof of the first assertion of Part (2) is the same as the proof of Part (1) with the following additional inductive step. Assume that the result holds for φ_0 and let $\varphi = (Qx)\varphi_0$ where Q is a quantifier symbol. We discuss only the case when $Q = \forall$. We have

$$\begin{aligned}
&(\mathcal{A},\sigma)\models(\forall x)\varphi_0\\
&\quad\text{if and only if} \quad (\mathcal{A},[x\to a]\sigma)\models\varphi_0 \text{ for all } a\in|\mathcal{A}|\\
&\quad\text{if and only if} \quad (\mathcal{B},h\circ[x\to a]\sigma)\models\varphi_0 \text{ for all } a\in|\mathcal{A}|\\
&\qquad\qquad\qquad\quad \text{(by inductive hypothesis)}\\
&\quad\text{if and only if} \quad (\mathcal{B},[x\to h(a)](h\circ\sigma))\models\varphi_0 \text{ for all } a\in|\mathcal{A}|\\
&\quad\text{if and only if} \quad (\mathcal{B},[x\to b](h\circ\sigma))\models\varphi_0 \text{ for all } b\in|\mathcal{B}|\\
&\qquad\qquad\qquad\quad \text{(since } h \text{ is an epimorphism)}\\
&\quad\text{if and only if} \quad (\mathcal{B},h\circ\sigma)\models(\forall x)\varphi_0.
\end{aligned}$$

We used above the easily verifiable fact that $h\circ[x\to a]\sigma = [x\to h(a)](h\circ\sigma)$ for every $a\in|\mathcal{A}|$.

For the second assertion of Part (2), let $\tau\in \mathrm{ASSIGN}_{\mathcal{B}}$. Using the Axiom of Choice, for every $x\in \mathrm{VAR}$, we can select $a_x\in|\mathcal{A}|$ such that $h(a_x)=\tau(x)$, since h is a surjection. Thus, if σ is the assignment over $\mathcal{A}$ defined by $\sigma(x)=a_x$ for $x\in\mathrm{VAR}$, then $\tau = h\circ\sigma$. Suppose that $\mathcal{A}\models\varphi$. Let $\tau\in\mathrm{ASSIGN}_{\mathcal{B}}$. By the previous argument, there is $\sigma\in\mathrm{ASSIGN}_{\mathcal{A}}$ such that $\tau = h\circ\sigma$. Since $(\mathcal{A},\sigma)\models\varphi$, we have

$(\mathcal{B}, \tau) \models \varphi$, so $\mathcal{B} \models \varphi$. The converse implication follows immediately from the first assertion.

To prove Part (3), we need to add to the argument of Part (1) the basis step when φ is $t_0 = t_1$. We have

$$\begin{aligned}
(\mathcal{A}, \sigma) \models \varphi \quad & \text{if and only if} \quad \sigma^{\mathcal{A}}(t_0) = \sigma^{\mathcal{A}}(t_1) \\
& \text{if and only if} \quad h(\sigma^{\mathcal{A}}(t_0)) = h(\sigma^{\mathcal{A}}(t_1)) \\
& \qquad\qquad\qquad\quad (\text{since } h \text{ is injective}) \\
& \text{if and only if} \quad (h \circ \sigma)^{\mathcal{B}}(t_0) = (h \circ \sigma)^{\mathcal{B}}(t_1) \\
& \qquad\qquad\qquad\quad (\text{by Theorem 4.5.6}) \\
& \text{if and only if} \quad (\mathcal{B}, h \circ \sigma) \models \varphi.
\end{aligned}$$

The argument for the last part follows from the combination of the arguments of the previous parts. □

4.5.6 *Semantics of Signed Formulas*

The definition of the semantics of signed formulas proceeds along similar lines as the corresponding development for propositional signed formulas.

Definition 4.5.69. Let $b\varphi$ be a signed $\mathcal{L}$-formula, $\mathcal{A}$ be an $\mathcal{L}$-structure, and $\sigma \in \mathrm{ASSIGN}_{\mathcal{A}}$. We say that $(\mathcal{A}, \sigma)$ *satisfies* and write $(\mathcal{A}, \sigma) \models b\varphi$, if $\mathcal{S}_{\mathcal{A}}(\varphi)(\sigma) = b$, where $\mathcal{S}_{\mathcal{A}}$ is the function introduced in Definition 4.5.9. In other words, $(\mathcal{A}, \sigma) \models \mathbf{T}\varphi$ if $(\mathcal{A}, \sigma) \models \varphi$ and $(\mathcal{A}, \sigma) \models \mathbf{F}\varphi$ if $(\mathcal{A}, \sigma) \not\models \varphi$.

Let Δ be a set of signed $\mathcal{L}$-formulas. The pair $(\mathcal{A}, \sigma)$ *satisfies* Δ if $(\mathcal{A}, \sigma) \models b\varphi$ for every $b\varphi \in \Delta$. In this case, we write $(\mathcal{A}, \sigma) \models \Delta$.

A set Δ of signed formulas is *satisfiable* if $(\mathcal{A}, \sigma) \models \Delta$ for some $\mathcal{L}$-structure $\mathcal{A}$ and assignment σ.

The set Δ *logically implies* a signed formula $b\varphi$ (written $\Delta \models b\varphi$) if for every pair $(\mathcal{A}, \sigma)$, where $\mathcal{A}$ is an $\mathcal{L}$-structure and $\sigma \in \mathrm{ASSIGN}_{\mathcal{A}}$, $(\mathcal{A}, \sigma) \models b\varphi$ whenever $(\mathcal{A}, \sigma) \models \Delta$. ▯

Theorem 4.5.70. *Let $\mathcal{L}, \mathcal{L}'$ be two first-order languages such that $\mathcal{L} \subseteq \mathcal{L}'$. Suppose that $\mathcal{B}$ is an $\mathcal{L}'$-structure and that $\mathcal{A}$ is $RED_{\mathcal{L}}(\mathcal{B})$, the reduct of $\mathcal{B}$ to $\mathcal{L}$. Then, for every $b\varphi \in \mathrm{SFORM}_{\mathcal{L}}$ and*

$\sigma \in$ ASSIGN$_{\mathcal{A}}$ = ASSIGN$_{\mathcal{B}}$, we have $(\mathcal{A}, \sigma) \models b\varphi$ if and only if $(\mathcal{B}, \sigma) \models b\varphi$.

Proof. This statement follows immediately from Theorem 4.5.24. □

If Δ is both a set of signed $\mathcal{L}$-formulas and a set of signed $\mathcal{L}'$-formulas for two first-order languages $\mathcal{L}, \mathcal{L}'$, then Δ is satisfiable as a set of signed $\mathcal{L}$-formulas if and only if it is satisfiable as a set of signed $\mathcal{L}'$-formulas. The argument for this is similar to the one of Corollary 4.5.34 and makes use of Theorem 4.5.70. Since logical implication can be characterized in terms of satisfiability, it follows that logical implication is also language-independent.

As in the propositional case, a closed set of signed formulas is unsatisfiable. Also, a set may be unsatisfiable without being closed.

Theorem 4.5.71. *Let $\mathcal{L}$ be first-order language, Γ be a set of $\mathcal{L}$-formulas and let φ be an $\mathcal{L}$-formula. Then, $\Gamma \models \varphi$ if and only if the set of signed formulas $\{\mathbf{T}\psi \mid \psi \in \Gamma\} \cup \{\mathbf{F}\varphi\}$ is unsatisfiable.*

Proof. We leave this simple proof to the reader. □

Theorem 4.5.72. *Let $\mathcal{L}$ be a first-order language, Δ be a set of signed $\mathcal{L}$-formulas, and let $b\varphi$ be a signed $\mathcal{L}$-formula. Then, $\Delta \cup \{b\varphi\}$ is unsatisfiable if and only if $\Delta \models \overline{b}\varphi$.*

Proof. The argument is left to the reader. □

The next result explains the similarity between the two types of signed $\boldsymbol{\gamma}$-formulas and the two types of signed $\boldsymbol{\delta}$-formulas.

Theorem 4.5.73. *Let $\mathcal{L}$ be a first-order language, $\mathcal{A}$ be an $\mathcal{L}$-structure and $\sigma \in$ ASSIGN$_{\mathcal{A}}$.*

If $b(Qx)\varphi$ is a $\boldsymbol{\gamma}$-formula of $\mathcal{L}$, then $(\mathcal{A}, \sigma) \models b(Qx)\varphi$ if and only if $(\mathcal{A}, [x \to a]\sigma) \models b\varphi$ for every $a \in |\mathcal{A}|$.

If $b(Qx)\varphi$ is a $\boldsymbol{\delta}$-formula of $\mathcal{L}$, then $(\mathcal{A}, \sigma) \models b(Qx)\varphi$ if and only if $(\mathcal{A}, [x \to a]\sigma) \models b\varphi$ for some $a \in |\mathcal{A}|$.

Proof. The statement follows immediately from the definition of satisfaction of signed formulas. □

4.6 Semantics of Substitutions and Replacements

In this section, we will show some technical results involving the semantics of replacements and of several types of substitutions. The types of substitutions considered may involve relation symbols, or variables and terms.

We begin with a result concerning substituting one relation symbol for another.

Theorem 4.6.1. *Let $\mathcal{A}$ be an $\mathcal{L}$-structure and let R, R' be two n-ary relation symbols such that $R \in \mathcal{L}$ and $R' \notin \mathcal{L}$. Further, assume that if R' is $=$, then $R^{\mathcal{A}} = \{(a,a) \mid a \in |\mathcal{A}|\}$. If $\sigma \in \mathrm{ASSIGN}_{\mathcal{A}}$, then $(\mathcal{A}, \sigma) \models \varphi$ if and only if $(\mathcal{A}_{R \to R'}, \sigma) \models \mathsf{s}^{R}_{R'}(\varphi)$ for every $\mathcal{L}$-formula φ.*

Proof. The reader can easily prove that $\sigma^{\mathcal{A}}(t) = \sigma^{\mathcal{A}_{R\to R'}}(t)$ for every $\mathcal{L}$-term t. We use this fact in the main argument which is by induction on $\mathcal{L}$-formulas. For the basis step, let φ be the atomic formula. If φ is a propositional constant, $\varphi = P$, we distinguish two cases depending on whether or not $P = R$. In the former case, we have:

$$\begin{aligned}(\mathcal{A}_{R\to R'}, \sigma) \models \mathsf{s}^{R}_{R'}(\varphi) \;&\text{ if and only if }\; (R')^{\mathcal{A}_{R\to R'}} = \mathbf{T}\\ &\text{ if and only if }\; R^{\mathcal{A}} = \mathbf{T}\\ &\text{ if and only if }\; (\mathcal{A}, \sigma) \models \varphi.\end{aligned}$$

In the latter case, that is, when $P \neq R$, we have:

$$\begin{aligned}(\mathcal{A}_{R\to R'}, \sigma) \models \mathsf{s}^{R}_{R'}(\varphi) \;&\text{ if and only if }\; P^{\mathcal{A}_{R\to R'}} = \mathbf{T}\\ &\text{ if and only if }\; P^{\mathcal{A}} = \mathbf{T}\\ &\text{ if and only if }\; (\mathcal{A}, \sigma) \models \varphi.\end{aligned}$$

Suppose now that $\varphi = P(t_0, \ldots, t_{m-1})$. Again, we distinguish two cases depending on whether or not $P = R$. If $P = R$, then $m = n$ and we have the following equivalent statements:

$$\begin{aligned}&(\mathcal{A}_{R\to R'}, \sigma) \models \mathsf{s}^{R}_{R'}(\varphi)\\ &(\sigma^{\mathcal{A}_{R\to R'}}(t_0), \ldots, \sigma^{\mathcal{A}_{R\to R'}}(t_{n-1})) \in (R')^{\mathcal{A}_{R\to R'}}\\ &(\sigma^{\mathcal{A}}(t_0), \ldots, \sigma^{\mathcal{A}}(t_{n-1})) \in R^{\mathcal{A}}\\ &(\mathcal{A}, \sigma) \models \varphi.\end{aligned}$$

If $P \neq R$, then we have the following equivalent statements:

$$\begin{aligned}
&(\mathcal{A}_{R\to R'}, \sigma) \models \mathsf{s}^R_{R'}(\varphi)\\
&(\sigma^{\mathcal{A}_{R\to R'}}(t_0), \ldots, \sigma^{\mathcal{A}_{R\to R'}}(t_{m-1})) \in P^{\mathcal{A}_{R\to R'}}\\
&(\sigma^{\mathcal{A}}(t_0), \ldots, \sigma^{\mathcal{A}}(t_{m-1})) \in P^{\mathcal{A}}\\
&(\mathcal{A}, \sigma) \models \varphi.
\end{aligned}$$

We leave the inductive steps to the reader. □

4.6.1 *The Substitution Theorem*

Now we concentrate on $\mathcal{L}$-substitutions as defined in Section 4.3. The main point of this subsection is to show that the satisfaction of a formula obtained by a substitution from a formula φ relative to a structure and an assignment amounts to the satisfaction of φ relative to the same structure and a modified assignment.

Lemma 4.6.2. *Let $\mathcal{L}$ be a first-order language. If $\mathcal{A}$ is an $\mathcal{L}$-structure and s is an $\mathcal{L}$-substitution, then $\sigma^{\mathcal{A}} \circ \overline{s} = (\sigma^{\mathcal{A}} \circ s)^{\mathcal{A}}$, for every $\sigma \in$ ASSIGN$_{\mathcal{A}}$.*

Proof. We need to prove that $\sigma^{\mathcal{A}} \circ \overline{s}(t) = (\sigma^{\mathcal{A}} \circ s)^{\mathcal{A}}(t)$ for every $t \in \text{TERM}_{\mathcal{L}}$ and every $\sigma \in \text{ASSIGN}_{\mathcal{A}}$. We proceed by induction on the term t. If $t = x$, then both sides of the equality of the lemma are $\sigma^{\mathcal{A}}(s(x))$. When t is a constant symbol c, then both sides equal $c^{\mathcal{A}}$. This concludes the basis steps.

Now, assume that $t = f(t_0, \ldots, t_{n-1})$ for $n > 0$ and that

$$\sigma^{\mathcal{A}} \circ \overline{s}(t_i) = (\sigma^{\mathcal{A}} \circ s)^{\mathcal{A}}(t_i)$$

for $0 \leq i \leq n-1$. We have

$$\begin{aligned}
\sigma^{\mathcal{A}} \circ \overline{s}(t) &= \sigma^{\mathcal{A}}(f(\overline{s}(t_0), \ldots, \overline{s}(t_{n-1})))\\
&= f^{\mathcal{A}}(\sigma^{\mathcal{A}}(\overline{s}(t_0)), \ldots, \sigma^{\mathcal{A}}(\overline{s}(t_{n-1})))\\
&= f^{\mathcal{A}}((\sigma^{\mathcal{A}} \circ s)^{\mathcal{A}}(t_0), \ldots, (\sigma^{\mathcal{A}} \circ s)^{\mathcal{A}}(t_{n-1}))\\
&= (\sigma^{\mathcal{A}} \circ s)^{\mathcal{A}}(f(t_0, \ldots, t_{n-1})).
\end{aligned}$$

□

Lemma 4.6.3. *Let $\mathcal{L}$ be a first-order language, $y_0, \ldots, y_{n-1}$ be distinct variables, and $t, t_0, \ldots, t_{n-1}$ be terms. If $\mathcal{A}$ is an $\mathcal{L}$-structure, then*

$$\sigma^{\mathcal{A}}(\mathsf{s}^{y_0 \cdots y_{n-1}}_{t_0 \cdots t_{n-1}}(t)) = \sigma'^{\mathcal{A}}(t)$$

where $\sigma' = [y_0 \to \sigma^{\mathcal{A}}(t_0)] \cdots [y_{n-1} \to \sigma^{\mathcal{A}}(t_{n-1})]\sigma$ for every assignment $\sigma \in \mathrm{ASSIGN}_{\mathcal{A}}$.

Proof. The proof consists of applying Lemma 4.6.2 with $s = \mathsf{s}^{y_0 \cdots y_{n-1}}_{t_0 \cdots t_{n-1}}$ observing that

$$\sigma^{\mathcal{A}} \circ \mathsf{s}^{y_0 \cdots y_{n-1}}_{t_0 \cdots t_{n-1}} = [y_0 \to \sigma^{\mathcal{A}}(t_0)] \cdots [y_{n-1} \to \sigma^{\mathcal{A}}(t_{n-1})]\sigma.$$

□

The next theorem shows that in order to evaluate the truth value of a formula obtained by applying a substitution to the free variables of another formula φ, we can evaluate the truth value of the original formula φ using an assignment modified by the substitution.

Theorem 4.6.4 (The Substitution Theorem). *If $\mathcal{L}$ is a first-order language, $\mathcal{A}$ is an $\mathcal{L}$-structure, φ is an $\mathcal{L}$-formula, and s is a substitution admissible for φ, then*

$$\mathcal{S}_{\mathcal{A}}\boldsymbol{(}\mathrm{FVSubst}(s, \varphi)\boldsymbol{)}(\sigma) = \mathcal{S}_{\mathcal{A}}\boldsymbol{(}\varphi\boldsymbol{)}(\sigma^{\mathcal{A}} \circ s),$$

for every $\sigma \in \mathrm{ASSIGN}_{\mathcal{A}}$.

Proof. The argument is by induction on the formula φ. If φ is a propositional constant, then $\mathrm{FVSubst}(s, \varphi) = \varphi$ and σ and $\sigma^{\mathcal{A}} \circ s$ agree on $\mathtt{FV}(\varphi) = \emptyset$. Therefore, by Theorem 4.5.12, we obtain the desired equality.

Suppose now that φ is $R(t_0, \ldots, t_{n-1})$, where R is an n-ary relation symbol and $t_0, \ldots, t_{n-1}$ are terms. We have

$$\mathrm{FVSubst}(s, \varphi) = R(\overline{s}(t_0), \ldots, \overline{s}(t_{n-1}))$$

and

$$\begin{aligned}&\mathcal{S}_{\mathcal{A}}\boldsymbol{(}\mathrm{FVSubst}(s, \varphi)\boldsymbol{)}(\sigma)\\ &= \begin{cases}\mathbf{T} & \text{if } (\sigma^{\mathcal{A}}(\overline{s}(t_0)), \ldots, \sigma^{\mathcal{A}}(\overline{s}(t_{n-1}))) \in R^{\mathcal{A}},\\ \mathbf{F} & \text{otherwise.}\end{cases}\end{aligned}$$

By Lemma 4.6.2, we have

$$\sigma^{\mathcal{A}}(\overline{s}(t_i)) = (\sigma^{\mathcal{A}} \circ s)^{\mathcal{A}}(t_i)$$

for $0 \leq i \leq n-1$, hence

$$\begin{aligned}
&\mathcal{S}_{\mathcal{A}}\boldsymbol{(}\text{FVSubst}(s,\varphi)\boldsymbol{)}(\sigma) \\
&\quad = \begin{cases} \mathbf{T} & \text{if } ((\sigma^{\mathcal{A}} \circ s)^{\mathcal{A}}(t_0), \ldots, (\sigma^{\mathcal{A}} \circ s)^{\mathcal{A}}(t_{n-1})) \in R^{\mathcal{A}} \\ \mathbf{F} & \text{otherwise} \end{cases} \\
&\quad = \mathcal{S}_{\mathcal{A}}\boldsymbol{(}\varphi\boldsymbol{)}((\sigma^{\mathcal{A}} \circ s)).
\end{aligned}$$

Assume now that $\varphi = (\alpha C \beta)$, where C is a binary connective symbol, and that the result holds for α and β. Suppose that s is admissible for φ. By Corollary 4.3.81, s is also admissible for both α and β, so by inductive hypothesis,

$$\begin{aligned}
\mathcal{S}_{\mathcal{A}}\boldsymbol{(}\text{FVSubst}(s,\alpha)\boldsymbol{)}(\sigma) &= \mathcal{S}_{\mathcal{A}}\boldsymbol{(}\alpha\boldsymbol{)}(\sigma^{\mathcal{A}} \circ s), \\
\mathcal{S}_{\mathcal{A}}\boldsymbol{(}\text{FVSubst}(s,\beta)\boldsymbol{)}(\sigma) &= \mathcal{S}_{\mathcal{A}}\boldsymbol{(}\beta\boldsymbol{)}(\sigma^{\mathcal{A}} \circ s).
\end{aligned}$$

We have $\text{FVSubst}(s,\varphi) = (\text{FVSubst}(s,\alpha)\ C\ \text{FVSubst}(s,\beta))$, by Lemma 4.3.53 and

$$\begin{aligned}
&\mathcal{S}_{\mathcal{A}}\boldsymbol{(}\text{FVSubst}(s,\varphi)\boldsymbol{)}(\sigma) \\
&\quad = \mathcal{S}_{\mathcal{A}}\boldsymbol{(}\text{FVSubst}(s,\alpha)\ C\ \text{FVSubst}(s,\beta)\boldsymbol{)}(\sigma) \\
&\quad = f_C(\mathcal{S}_{\mathcal{A}}\boldsymbol{(}\text{FVSubst}(s,\alpha)\boldsymbol{)}(\sigma), \mathcal{S}_{\mathcal{A}}\boldsymbol{(}\text{FVSubst}(s,\beta)\boldsymbol{)}(\sigma)) \\
&\qquad \text{(by the inductive hypothesis)} \\
&\quad = f_C(\mathcal{S}_{\mathcal{A}}\boldsymbol{(}\alpha\boldsymbol{)}(\sigma^{\mathcal{A}} \circ s), \mathcal{S}_{\mathcal{A}}\boldsymbol{(}\beta\boldsymbol{)}(\sigma^{\mathcal{A}} \circ s)) \\
&\quad = \mathcal{S}_{\mathcal{A}}\boldsymbol{(}\alpha C \beta\boldsymbol{)}(\sigma^{\mathcal{A}} \circ s) \\
&\quad = \mathcal{S}_{\mathcal{A}}\boldsymbol{(}\varphi\boldsymbol{)}(\sigma^{\mathcal{A}} \circ s).
\end{aligned}$$

We leave to the reader the case when $\varphi = (\neg\alpha)$.

Assume now that $\varphi = (Qx)\alpha$, where Q is a quantifier and that the result holds for α. Suppose that s is admissible for φ. Then, by Corollary 4.3.81,

(1) $[x \to x]s$ is admissible for α, and

(2) for every variable y that occurs free in $(Qx)\alpha$, x does not occur in $s(y)$.

We need to show first that the assignments $[x \to a]\sigma^{\mathcal{A}} \circ ([x \to x]s)$ and $[x \to a](\sigma^{\mathcal{A}} \circ s)$ agree on $\mathtt{FV}(\alpha)$ for every $a \in |\mathcal{A}|$. It is clear that both of these assignments map x to a. Assume that $y \neq x$ and $y \in \mathtt{FV}(\alpha)$. Then, $y \in \mathtt{FV}(\varphi)$, so x does not occur in $s(y)$. Thus, we have

$$\begin{aligned}
&[x \to a]\sigma^{\mathcal{A}} \circ ([x \to x]s)(y) \\
&\qquad = [x \to a]\sigma^{\mathcal{A}}(s(y)) && \text{(because } y \neq x\text{)} \\
&\qquad = \sigma^{\mathcal{A}}(s(y)) && \text{(by Theorem 4.5.3)} \\
&\qquad = ([x \to a](\sigma^{\mathcal{A}} \circ s))(y) && \text{(because } y \neq x\text{)}
\end{aligned}$$

so the assignments agree on y.

By inductive hypothesis, since $[x \to x]s$ is admissible for α, we have

$$\begin{aligned}
&\mathcal{S}_{\mathcal{A}}\mathbf{(}\text{FVSubst}([x \to x]s, \alpha)\mathbf{)}([x \to a]\sigma) \\
&\quad = \mathcal{S}_{\mathcal{A}}\mathbf{(}\alpha\mathbf{)}(([x \to a]\sigma)^{\mathcal{A}} \circ ([x \to x]s)),
\end{aligned}$$

for all $a \in |\mathcal{A}|$, which allows us to write, when $Q = \forall$,

$$\begin{aligned}
&\mathcal{S}_{\mathcal{A}}\mathbf{(}\text{FVSubst}(s, (\forall x)\alpha)\mathbf{)}(\sigma) \\
&\quad = \mathcal{S}_{\mathcal{A}}\mathbf{(}(\forall x)\text{FVSubst}([x \to x]s, \alpha)\mathbf{)}(\sigma) \\
&\quad = \begin{cases} \mathbf{T} & \text{if } \mathcal{S}_{\mathcal{A}}\mathbf{(}\text{FVSubst}([x \to x]s, \alpha)\mathbf{)}([x \to a]\sigma) = \mathbf{T} \\ & \text{for every } a \in |\mathcal{A}| \\ \mathbf{F} & \text{otherwise.} \end{cases} \\
&\quad = \begin{cases} \mathbf{T} & \text{if } \mathcal{S}_{\mathcal{A}}\mathbf{(}\alpha\mathbf{)}([x \to a]\sigma^{\mathcal{A}} \circ ([x \to x]s)) = \mathbf{T} \text{ for every } a \in |\mathcal{A}| \\ \mathbf{F} & \text{otherwise.} \end{cases} \\
&\quad = \begin{cases} \mathbf{T} & \text{if } \mathcal{S}_{\mathcal{A}}\mathbf{(}\alpha\mathbf{)}([x \to a](\sigma^{\mathcal{A}} \circ s)) = \mathbf{T} \text{ for every } a \in |\mathcal{A}| \\ \mathbf{F} & \text{otherwise.} \end{cases} \\
&\quad = \mathcal{S}_{\mathcal{A}}\mathbf{(}(\forall x)\alpha\mathbf{)}(\sigma^{\mathcal{A}} \circ s).
\end{aligned}$$

We leave to the reader the case when Q is the existential quantifier. $\square$

Corollary 4.6.5. *If $\mathcal{L}$ is a first-order language, $\mathcal{A}$ is an $\mathcal{L}$-structure, φ is an $\mathcal{L}$-formula, and s is a substitution admissible for φ, then $(\mathcal{A}, \sigma) \models \mathrm{FVSubst}(s, \varphi)$ if and only if $(\mathcal{A}, \sigma^{\mathcal{A}} \circ s) \models \varphi$.*

Proof. This corollary follows immediately from Definition 4.5.9 and Theorem 4.6.4. □

Corollary 4.6.6 (The Substitution Corollary). *If $\mathcal{L}$ is a first-order language, $\mathcal{A}$ is an $\mathcal{L}$-structure, φ is an $\mathcal{L}$-formula, $y_0, \ldots, y_{n-1}$ are distinct variables and $t_0, \ldots, t_{n-1}$ are $\mathcal{L}$-terms such that t_i is substitutable for y_i in φ, then we have $(\mathcal{A}, \sigma) \models (\varphi)_{y_0,\ldots,y_{n-1}:=t_0,\ldots,t_{n-1}}$ if and only if $(\mathcal{A}, [y_0 \to \sigma^{\mathcal{A}}(t_0)] \cdots [y_{n-1} \to \sigma^{\mathcal{A}}(t_{n-1})]\sigma) \models \varphi$.*

Proof. Taking $s = \mathsf{s}^{y_0 \cdots y_{n-1}}_{t_0 \cdots t_{n-1}}$ in Corollary 4.6.5 gives the result. □

The following generalization of parts of Theorems 4.5.37 and 4.5.40 is an application of the Substitution Corollary.

Theorem 4.6.7. *If φ is a formula, $y_0, \ldots, y_{n-1}$ are distinct variables and $t_0, \ldots, t_{n-1}$ are terms such that t_i is substitutable for y_i in φ, then we have*

$$(\forall y_0) \cdots (\forall y_{n-1})\varphi \models (\varphi)_{y_0,\ldots,y_{n-1}:=t_0,\ldots,t_{n-1}}$$

and

$$(\varphi)_{y_0,\ldots,y_{n-1}:=t_0,\ldots,t_{n-1}} \models (\exists y_0) \cdots (\exists y_{n-1})\varphi.$$

Proof. Let $\mathcal{L}$ be a first-order language such that φ is an $\mathcal{L}$-formula and $t_0, \ldots, t_{n-1}$ are $\mathcal{L}$-terms. Assume that $(\mathcal{A}, \sigma) \models (\forall y_0) \cdots (\forall y_{n-1})\varphi$, where $\mathcal{A}$ is an $\mathcal{L}$-structure and $\sigma \in \mathrm{ASSIGN}_{\mathcal{A}}$. Since

$$(\mathcal{A}, [y_{n-1} \to a_{n-1}] \cdots [y_0 \to a_0]\sigma) \models \varphi$$

for all $a_0, \ldots, a_{n-1} \in |\mathcal{A}|$, we have

$$(\mathcal{A}, [y_0 \to a_0] \cdots [y_{n-1} \to a_{n-1}]\sigma) \models \varphi$$

for all $a_0, \ldots, a_{n-1} \in |\mathcal{A}|$ because the variables $y_0, \ldots, y_{n-1}$ are pairwise distinct. Choosing $a_i = \sigma^{\mathcal{A}}(t_i)$ for $0 \leq i \leq n-1$, we obtain $(\mathcal{A}, [y_0 \to \sigma^{\mathcal{A}}(t_0)] \cdots [y_{n-1} \to \sigma^{\mathcal{A}}(t_{n-1})]\sigma) \models \varphi$ which implies, by the Substitution Corollary, $(\mathcal{A}, \sigma) \models (\varphi)_{y_0,\ldots,y_{n-1}:=t_0,\ldots,t_{n-1}}$. We leave to the reader the argument for the second part of the theorem. □

If in the above theorem we take $t_i = y_i$, for $0 \leq i \leq n-1$, then it is easy to see that each t_i is substitutable for y_i in φ and $(\varphi)_{y_0,\dots,y_{n-1}:=t_0,\dots,t_{n-1}} = \varphi$. This gives the logical implications $(\forall y_0)\cdots(\forall y_{n-1})\varphi \models \varphi$ and $\varphi \models (\exists y_0)\cdots(\exists y_{n-1})\varphi$. (The cases when $n = 1$ were previously discussed in Theorems 4.5.37 and 4.5.40.)

Example 4.6.8. We will show that the substitutability of t_i for y_i in φ is essential in Theorem 4.6.7. Indeed, let $\varphi = (\exists y)R(y_0, y)$ and let $t_0 = y$, where $y \neq y_0$. Observe that t_0 is not substitutable for y_0 in φ. If $\mathcal{A}$ is the $\{R\}$-structure given by $|\mathcal{A}| = \mathbf{N}$ and $R^{\mathcal{A}} = \{(m,n) \mid m < n\}$, then $\mathcal{A} \models (\forall y_0)(\exists y)R(y_0, y)$ (because for every natural number m, there is a natural number n larger than m), but $\mathcal{A} \not\models (\exists y)R(y,y)$. Thus $(\forall y_0)\varphi \not\models (\varphi)_{y_0:=t_0}$. ▯

Theorem 4.6.9. *If θ is an instance of a universal formula ψ, then $\psi \models \theta$.*

Proof. This statement follows immediately from Definition 4.3.57 and Theorem 4.6.7. □

Corollary 4.6.10. *Let $\mathcal{L}$ be a first-order language. Then, every instance of a formula in $Eq_{=,\mathcal{L}}$ is logically valid.*

Proof. This statement follows from Theorem 4.6.9 and from the fact that all formulas in $\mathrm{Eq}_{=,\mathcal{L}}$ are logically valid (see Corollary 4.5.66). □

As another illustration of the use of the Substitution Corollary, we prove the following result on renaming bound variables in formulas.

Theorem 4.6.11. *Let φ be a formula, and x, y be variables such that y does not occur free in φ and y is substitutable for x in φ. Then,*

$$(\forall x)\varphi \equiv (\forall y)(\varphi)_{x:=y} \quad \textit{and} \quad \mathtt{FV}((\forall x)\varphi) = \mathtt{FV}((\forall y)(\varphi)_{x:=y})$$
$$(\exists x)\varphi \equiv (\exists y)(\varphi)_{x:=y} \quad \textit{and} \quad \mathtt{FV}((\exists x)\varphi) = \mathtt{FV}((\exists y)(\varphi)_{x:=y}).$$

Proof. Let φ be an $\mathcal{L}$-formula, $\mathcal{A}$ be an $\mathcal{L}$-structure, and let $\sigma \in \mathrm{ASSIGN}_{\mathcal{A}}$, where $\mathcal{L}$ is a first-order language. The following statements are equivalent:

(1) $(\mathcal{A}, \sigma) \models (\forall y)(\varphi)_{x:=y}$;
(2) for all $a \in |\mathcal{A}|$, $(\mathcal{A}, [y \to a]\sigma) \models (\varphi)_{x:=y}$;

(3) for all $a \in |\mathcal{A}|$, $(\mathcal{A}, [x \to ([y \to a]\sigma)^{\mathcal{A}}(y)][y \to a]\sigma) \models \varphi$;
(4) for all $a \in |\mathcal{A}|$, $(\mathcal{A}, [x \to a][y \to a]\sigma) \models \varphi$;
(5) for all $a \in |\mathcal{A}|$, $(\mathcal{A}, [x \to a]\sigma) \models \varphi$;
(6) $(\mathcal{A}, \sigma) \models (\forall x)\varphi$.

The equivalences of Parts (1) and (2), (3) and (4), and (5) and (6) follow immediately from the definitions. The equivalence of (2) and (3) follows from the Substitution Corollary since y is substitutable for x in φ. Finally, the equivalence of (4) and (5) follows from the fact that the assignments $[x \to a][y \to a]\sigma$ and $[x \to a]\sigma$ agree on the free variables of φ.

By Corollary 4.3.84, we have the inclusions

$$\mathtt{FV}(\varphi) - \{x\} \subseteq \mathtt{FV}((\varphi)_{x:=y}) \subseteq (\mathtt{FV}(\varphi) - \{x\}) \cup \{y\}.$$

Subtracting $\{y\}$ from the three sets involved in the previous inclusions and using the fact that y does not occur free in φ, we have $\mathtt{FV}(\varphi) - \{x\} = \mathtt{FV}((\varphi)_{x:=y}) - \{y\}$, that is $\mathtt{FV}((\forall x)\varphi) = \mathtt{FV}((\forall y)(\varphi)_{x:=y})$.

The argument for the second part is virtually the same as the argument for the first part. □

Note that if y does not occur in φ, then y and φ meet the conditions of Theorem 4.6.11.

Yet one more application of the Substitution Corollary is the following result.

Theorem 4.6.12. *Let Γ be a set of formulas and let c be a constant symbol that does not occur in any formula of Γ or in φ.*

(1) *If $\Gamma \cup \{(\exists x)\varphi\}$ is satisfiable, then $\Gamma \cup \{(\varphi)_{x:=c}\}$ is satisfiable.*
(2) *If $\Gamma \cup \{(\neg(\forall x)\varphi)\}$ is satisfiable, then $\Gamma \cup \{((\neg\varphi))_{x:=c}\}$ is satisfiable.*

Proof. To prove the first part, let $\mathcal{L}$ be a first-order language such that $\Gamma \cup \{(\exists x)\varphi\} \subseteq \mathrm{FORM}_{\mathcal{L}}$ and $c \notin \mathcal{L}$ and let $\mathcal{A}$ be an $\mathcal{L}$-structure such that $(\mathcal{A}, \sigma) \models \Gamma \cup \{(\exists x)\varphi\}$ for some $\sigma \in \mathrm{ASSIGN}_{\mathcal{A}}$. Since $(\mathcal{A}, \sigma) \models (\exists x)\varphi$, we have $(\mathcal{A}, [x \to a_0]\sigma) \models \varphi$ for some $a_0 \in |\mathcal{A}|$. Define the language $\mathcal{L}' = \mathcal{L} \cup \{c\}$ and the expansion $\mathcal{A}'$ of $\mathcal{A}$ to $\mathcal{L}'$ by $c^{\mathcal{A}'} = a_0$.

By Theorem 4.5.24, we have $(\mathcal{A}', \sigma) \models \Gamma$. Thus, we need to show only that $(\mathcal{A}', \sigma) \models (\varphi)_{x:=c}$. Since $(\mathcal{A}, [x \to a_0]\sigma) \models \varphi$, we have $(\mathcal{A}', [x \to a_0]\sigma) \models \varphi$, again by Theorem 4.5.24. Therefore, $(\mathcal{A}', [x \to$

$\sigma^{\mathcal{A}'}(c)]\sigma) \models \varphi$, which implies $(\mathcal{A}', \sigma) \models (\varphi)_{x:=c}$, by the Substitution Corollary, in view of the fact that c is substitutable for x in φ.

For the second part, observe that $\Gamma \cup \{(\exists x)(\neg\varphi)\}$ is satisfiable because $(\neg(\forall x)\varphi) \equiv (\exists x)(\neg\varphi)$. Thus, by applying the first part, we have that $\Gamma \cup \{((\neg\varphi))_{x:=c}\}$ is satisfiable. □

Theorem 4.6.13. *Suppose that* Δ *is a set of signed formulas,* $b(Qx)\varphi$ *is a* $\boldsymbol{\delta}$*-formula, and* c *is a constant symbol that does not occur in any formula of* $\Delta \cup \{b(Qx)\varphi\}$*. If* $\Delta \cup \{b(Qx)\varphi\}$ *is satisfiable, then* $\Delta \cup \{b(\varphi)_{x:=c}\}$ *is satisfiable.*

Proof. The argument follows along the same lines as the one for Theorem 4.6.12, but using Theorem 4.5.70 instead of Theorem 4.5.24. □

The next statement shows that we can reduce the problem of logical implication $\Gamma \models \varphi$ for arbitrary Γ and φ to the problem of logical implication $\Gamma' \models \varphi'$, where Γ' is a set of closed formulas and φ' is a closed formula, obtained from Γ and φ effectively, by suitably extending the first-order language involved. This theorem plays an important role in presenting resolution in first-order logic.

Theorem 4.6.14. *Let* $\mathcal{L}$ *be a first-order language,* Γ *be a set of* $\mathcal{L}$*-formulas, and* φ *be an* $\mathcal{L}$*-formula. Let* s *be the* $\mathcal{L}^c$*-substitution defined by* $s(x_i) = d_i$*, where* $\mathcal{L}^c - \mathcal{L} = \{d_0, d_1, \ldots\}$ *and the constant symbols* d_i *are listed in the standard order.*

Then, $\mathrm{FVSubst}(s, \Gamma)$ *is a set of closed* $\mathcal{L}^c$*-formulas and* $\mathrm{FVSubst}(s, \varphi)$ *is a closed* $\mathcal{L}^c$*-formula, and*

$$\Gamma \models \varphi \textit{ if and only if } \mathrm{FVSubst}(s, \Gamma) \models \mathrm{FVSubst}(s, \varphi).$$

Proof. Since s replaces variables by constant symbols, it is clear that s is admissible for any formula and by Theorem 4.3.82, $\mathrm{FVSubst}(s, \Gamma)$ is a set of closed $\mathcal{L}^c$-formulas and $\mathrm{FVSubst}(s, \varphi)$ is a closed $\mathcal{L}^c$-formula.

Assume that $\Gamma \models \varphi$, $\mathcal{A}$ is an $\mathcal{L}^c$-structure, $\sigma \in \mathrm{ASSIGN}_{\mathcal{A}}$, and $(\mathcal{A}, \sigma) \models \mathrm{FVSubst}(s, \Gamma)$. By Corollary 4.6.5, the admissibility of s makes $(\mathcal{A}, \sigma) \models \mathrm{FVSubst}(s, \Gamma)$ equivalent to $(\mathcal{A}, \sigma^{\mathcal{A}} \circ s) \models \Gamma$, so, by the assumption that $\Gamma \models \varphi$, we have $(\mathcal{A}, \sigma^{\mathcal{A}} \circ s) \models \varphi$. Another application of Corollary 4.6.5 implies that $(\mathcal{A}, \sigma) \models \mathrm{FVSubst}(s, \varphi)$.

Conversely, suppose that $\mathrm{FVSubst}(s,\Gamma) \models \mathrm{FVSubst}(s,\varphi)$ and let $\mathcal{A}$ be an $\mathcal{L}$-structure and $\sigma \in \mathrm{ASSIGN}_{\mathcal{A}}$ be such that $(\mathcal{A},\sigma) \models \Gamma$. We must show that $(\mathcal{A},\sigma) \models \varphi$.

Define $\mathcal{A}'$ to be the expansion of $\mathcal{A}$ to $\mathcal{L}^c$ given by $d_i^{\mathcal{A}'} = \sigma(x_i)$ for $i \in \mathbf{N}$. Note that $\sigma^{\mathcal{A}'} \circ s(x_i) = \sigma^{\mathcal{A}'}(s(x_i)) = \sigma^{\mathcal{A}'}(d_i) = d_i^{\mathcal{A}'} = \sigma(x_i)$ for $i \in \mathbf{N}$, so $\sigma^{\mathcal{A}'} \circ s = \sigma$. We have the following four equivalent statements:

- $(\mathcal{A}',\sigma) \models \mathrm{FVSubst}(s,\Gamma)$;
- $(\mathcal{A}',\sigma^{\mathcal{A}'} \circ s) \models \Gamma$;
- $(\mathcal{A},\sigma^{\mathcal{A}'} \circ s) \models \Gamma$;
- $(\mathcal{A},\sigma) \models \Gamma$.

The equivalence of the first and second statements follows from the admissibility of s and Corollary 4.6.5. The second statement is equivalent to the third by Theorem 4.5.24, while the equivalence of the last two statements follows from the equality shown above. Since $(\mathcal{A},\sigma) \models \Gamma$, we have $(\mathcal{A}',\sigma) \models \mathrm{FVSubst}(s,\Gamma)$, so $(\mathcal{A}',\sigma) \models \mathrm{FVSubst}(s,\varphi)$. The admissibility of σ implies that $(\mathcal{A}',\sigma^{\mathcal{A}'} \circ s) \models \varphi$. Theorem 4.5.24 implies $(\mathcal{A},\sigma^{\mathcal{A}'} \circ s) \models \varphi$, which gives $(\mathcal{A},\sigma) \models \varphi$. Therefore, $\Gamma \models \varphi$. $\square$

4.6.2 *The Replacement Theorem*

Let $\mathcal{L}$ be a first-order language and $\mathcal{A}$ be an $\mathcal{L}$-structure. Replacing a subformula of a formula φ with an $\mathcal{A}$-equivalent formula does not change the meaning of φ in $\mathcal{A}$. This is formally stated in the next lemma.

Lemma 4.6.15 (Replacement Lemma for First-Order Logic). *Let φ be a formula and $\mathcal{A}$ be an $\mathcal{L}$-structure. If α, β are $\mathcal{A}$-equivalent formulas and (α, i) is an occurrence of α in φ, then* `replace` $(\varphi,(\alpha,i),\beta) \equiv_{\mathcal{A}} \varphi$.

Proof. Note that by Theorem 4.3.24, `replace` $(\varphi,(\alpha,i),\beta)$ is a formula. If $\varphi = \alpha$, then $i = 0$ and `replace` $(\varphi,(\alpha,i),\beta) = \beta$, so the result follows immediately.

The argument is by induction on φ. If φ is an atomic formula, then $\varphi = \alpha$ by Theorem 4.3.23 and we are done by the previous remark.

We consider here only the case when $\varphi = (Qx)\varphi_1$. The remaining cases are left to the reader. Suppose that the statement holds for the formula φ_1. If $\varphi = \alpha$, the conclusion is immediate. If $\varphi \neq \alpha$, we have shown in the proof of Theorem 4.3.24 that

$$\texttt{replace}\,(\varphi, (\alpha, i), \beta) = (Qx)\texttt{replace}\,(\varphi_1, (\alpha, i-4), \beta).$$

The formulas φ_1 and $\texttt{replace}\,(\varphi_1, (\alpha, i-4), \beta)$ are $\mathcal{A}$-equivalent by the inductive hypothesis. Therefore, by Lema 4.5.49, we have

$$\begin{aligned}\varphi = (Qx)\varphi_1 &\equiv_{\mathcal{A}} (Qx)\texttt{replace}\,(\varphi_1, (\alpha, i-4), \beta)\\ &= \texttt{replace}\,(\varphi, (\alpha, i), \beta).\end{aligned}$$

□

Theorem 4.6.16 (Replacement Theorem for First-Order Logic). *Let φ be a formula. If α, β are logically equivalent formulas and (α, i) is an occurrence of α in φ, then* $\texttt{replace}\,(\varphi, (\alpha, i), \beta)$ *is logically equivalent to the formula φ.*

Proof. The statement follows immediately from Lemma 4.6.15. □

Theorem 4.6.17. *Let $\mathcal{L}$ be a first-order language, t, u be $\mathcal{L}$-terms and φ be an $\mathcal{L}$-formula. If $t \equiv_{\mathcal{A}} u$ and ψ is obtained from φ by replacing an occurrence of t by u, then $\varphi \equiv_{\mathcal{A}} \psi$.*

Proof. The argument is by induction on the formula φ.

In the basis step, $\varphi = R(t_0, \ldots, t_{n-1})$. Then, for some i, $0 \leq i \leq n-1$, $\psi = R(t_0, \ldots, t_{i-1}, u_i, t_{i+1}, \ldots, t_{n-1})$, where u_i is obtained from t_i by replacing an occurrence of t by u. By Theorem 4.5.8, $t_i \equiv_{\mathcal{A}} u_i$. Thus, for all $\sigma \in \text{ASSIGN}_{\mathcal{A}}$ we have

$$\begin{aligned}&(\mathcal{A}, \sigma) \models \varphi\\ &\quad\text{if and only if } (\sigma^{\mathcal{A}}(t_0), \ldots, \sigma^{\mathcal{A}}(t_{n-1})) \in R^{\mathcal{A}}\\ &\quad\text{if and only if } (\sigma^{\mathcal{A}}(t_0), \ldots, \sigma^{\mathcal{A}}(t_{i-1}), \sigma^{\mathcal{A}}(u_i),\\ &\qquad \sigma^{\mathcal{A}}(t_{i+1}), \ldots, \sigma^{\mathcal{A}}(t_{n-1})) \in R^{\mathcal{A}}\\ &\quad\text{if and only if } (\mathcal{A}, \sigma) \models R(t_0, \ldots, t_{i-1}, u_i, t_{i+1}, \ldots, t_{n-1})\\ &\quad\text{if and only if} (\mathcal{A}, \sigma) \models \psi,\end{aligned}$$

so $\varphi \equiv_{\mathcal{A}} \psi$.

We discuss only one inductive step and leave the remaining steps for the reader.

Suppose that $\varphi = (\neg\varphi')$ and the result holds for φ'. Then, ψ has the form $(\neg\psi')$, where ψ' is obtained from φ' by replacing an occurrence of t by u. By inductive hypothesis, $\varphi' \equiv_{\mathcal{A}} \psi'$, so by Lemma 4.6.15, we have $\varphi \equiv_{\mathcal{A}} \psi$. □

Theorem 4.6.18. *Let φ be a formula, and $y_0, \ldots, y_{n-1}, z_0, \ldots, z_{n-1}$ be variables such that $z_0, \ldots, z_{n-1}$ are distinct variables that do not occur in $(Q_0 y_0) \cdots (Q_{n-1} y_{n-1})\varphi$ where $Q_0, \ldots, Q_{n-1}$ are quantifier symbols. Then,*

$$(Q_0 y_0) \cdots (Q_{n-1} y_{n-1})\varphi \equiv (Q_0 z_0) \cdots (Q_{n-1} z_{n-1})(\varphi)_{y_{n-1}:=z_{n-1},\ldots,y_0:=z_0}.$$

Moreover, the two equivalent formulas given above have the same sets of free variables.

Proof. The proof is by induction on $n \geq 1$. The basis step is implied by Theorem 4.6.11. For the inductive step, suppose that the statement holds for $n \geq 1$ and let $\psi = (Q_0 y_0) \cdots (Q_n y_n)\varphi$. By the inductive hypothesis, we have

$$(Q_1 y_1) \cdots (Q_n y_n)\varphi \equiv (Q_1 z_1) \cdots (Q_n z_n)(\varphi)_{y_n:=z_n,\ldots,y_1:=z_1}$$
$$\mathtt{FV}((Q_1 y_1) \cdots (Q_n y_n)\varphi) = \mathtt{FV}((Q_1 z_1) \cdots (Q_n z_n)(\varphi)_{y_n:=z_n,\ldots,y_1:=z_1}).$$

Using the Replacement Theorem, Theorem 4.6.11, and the last equality of Lemma 4.3.54 for $n = 1$, successively, we obtain

$$\begin{aligned}\psi &\equiv (Q_0 y_0)(Q_1 z_1) \cdots (Q_n z_n)(\varphi)_{y_n:=z_n,\ldots,y_1:=z_1}\\ &\equiv (Q_0 z_0)((Q_1 z_1) \cdots (Q_n z_n)(\varphi)_{y_n:=z_n,\ldots,y_1:=z_1})_{y_0:=z_0}\\ &= (Q_0 z_0) \cdots (Q_n z_n)(\varphi)_{y_n:=z_n,\ldots,y_0:=z_0}.\end{aligned}$$

Further, we have:

$$\begin{aligned}\mathtt{FV}(\psi) &= \mathtt{FV}((Q_0 y_0) \cdots (Q_n y_n)\varphi)\\ &\quad \text{(definition of } \psi\text{)}\\ &= \mathtt{FV}((Q_1 y_1) \cdots (Q_n y_n)\varphi) - \{y_0\}\\ &\quad \text{(Theorem 4.3.40)}\end{aligned}$$

$$= \mathtt{FV}((Q_1 z_1) \cdots (Q_n z_n)(\varphi)_{y_n := z_n, \ldots, y_1 := z_1}) - \{y_0\}$$
(Inductive Hypothesis)
$$= \mathtt{FV}((Q_0 y_0)(Q_1 z_1) \cdots (Q_n z_n)(\varphi)_{y_n := z_n, \ldots, y_1 := z_1})$$
(Theorem 4.3.40)
$$= \mathtt{FV}((Q_0 z_0)((Q_1 z_1) \cdots (Q_n z_n)(\varphi)_{y_n := z_n, \ldots, y_1 := z_1})_{y_0 := z_0})$$
(Theorem 4.6.11)
$$= \mathtt{FV}((Q_0 z_0) \cdots (Q_n z_n)(\varphi)_{y_n := z_n, \ldots, y_0 := z_0})$$
(Lemma 4.3.54).

□

4.6.3 *Variants of Formulas*

The notion of a variant of a formula is useful in facilitating the application of substitutions. The technical difficulties of this section will pay off whenever the normal application of substitutions would be impossible.

Definition 4.6.19. Let φ, ψ be two formulas.

We say that ψ is an *immediate variant* of φ if $\psi = \varphi$ or ψ is obtained from φ by replacing an occurrence of a subformula $(Qx)\alpha$ of φ by a formula $(Qy)(\alpha)_{x:=y}$, where y does not occur free in α and y is substitutable for x in α.

The formula ψ is a *variant* of φ if there is a sequence $(\theta_0, \ldots, \theta_{n-1})$ of formulas with $n \geq 1$ such that $\varphi = \theta_0$, $\psi = \theta_{n-1}$ and θ_i is an immediate variant of θ_{i-1} for $1 \leq i \leq n-1$. □

Note that by taking $n = 1$ in the above definition, it follows that every formula is a variant of itself. Furthermore, if θ is a variant of ψ which is variant of φ, then, in turn, θ is a variant of φ (*transitivity of variants*).

If $b\varphi$ is a signed formula, then $b\varphi'$ is an (immediate) variant of $b\varphi$ if φ' is an (immediate) variant of φ.

Given two formulas φ and ψ that are immediate variants, we can decide effectively whether or not φ is equal to ψ, and if they are not equal, we can find effectively the occurrence of $(Qx)\alpha$ in φ that is replaced by $(Qy)(\alpha)_{x:=y}$.

Lemma 4.6.20. *Let φ, φ' be formulas. Then:*

(1) *If ψ is an immediate variant of φ, then $(\neg\psi)$ is an immediate variant of $(\neg\varphi)$.*
(2) *If ψ and ψ' are f of φ and φ', respectively, then $(\psi C\varphi')$ and $(\varphi C\psi')$ are immediate variants of $(\varphi C\varphi')$.*
(3) *If ψ is an immediate variant of φ, then $(Qx)\psi$ is an immediate variant of $(Qx)\varphi$.*

Proof. These statements follow from the local character of the definition of immediate variant. □

Theorem 4.6.21. *Let φ, φ' be two formulas. Then:*

(1) *If ψ is a variant of φ, then $(\neg\psi)$ is a variant of $(\neg\varphi)$.*
(2) *If ψ and ψ' are variants of φ and φ', respectively, then $(\psi C\psi')$ is a variant of $(\varphi C\varphi')$.*
(3) *If ψ is a variant of φ, then $(Qx)\psi$ is a variant of $(Qx)\varphi$.*

Proof. We prove only the second part. The first and third parts are similar, but easier. Suppose that $\theta_0, \ldots, \theta_{n-1}$ is a sequence of immediate variants such that $\varphi = \theta_0$ and $\theta_{n-1} = \psi$ and $\theta'_0, \ldots, \theta'_{m-1}$ is a sequence of immediate variants such that $\varphi' = \theta'_0$ and $\theta'_{m-1} = \psi'$. Then, by Lemma 4.6.20, the following sequence of immediate variants

$$(\theta_0 C\theta'_0), (\theta_1 C\theta'_0), \ldots, (\theta_{n-1} C\theta'_0)$$
$$(\theta_{n-1} C\theta'_1), \ldots, (\theta_{n-1} C\theta'_{m-1})$$

shows that $(\psi C\psi') = (\theta_{n-1} C\theta'_{m-1})$ is a variant of $(\varphi C\varphi') = (\theta_0 C\theta'_0)$. □

Theorem 4.6.22. *If ψ is a variant of φ, then $\mathtt{FV}(\varphi) = \mathtt{FV}(\psi)$.*

Proof. It is clear that we need to show this statement only for immediate variants. Suppose that ψ is an immediate variant of φ obtained by replacing an occurrence of $(Qx)\alpha$ by $(Qy)(\alpha)_{x:=y}$, where y does not occur free in α and y is substitutable for x in α. By Theorem 4.6.11, $\mathtt{FV}((Qy)(\alpha)_{x:=y}) = \mathtt{FV}((Qx)\alpha)$, hence by Theorem 4.3.41, $\mathtt{FV}(\varphi) = \mathtt{FV}(\psi)$. □

Theorem 4.6.23. *Every variant of formula is logically equivalent to the formula.*

Proof. The statement follows from Theorems 4.6.11 and 4.6.16. □

Example 4.6.24. Consider the formulas:

$$\begin{aligned}\varphi &= (\forall x)(R(x) \to (\exists y)S(x,y))\\ \varphi' &= (\forall x)(R(x) \to (\exists y')S(x,y'))\\ \varphi'' &= (\forall x')(R(x') \to (\exists y')S(x',y')).\end{aligned}$$

Note that φ' is an immediate variant of φ and φ'' is an immediate variant of φ', which means that φ'' is a variant of φ. □

Example 4.6.25. Let $\varphi = (\forall x)((\forall x)R(x) \wedge Q(x))$ and $\psi = (\forall y)((\forall x)R(x) \wedge Q(y))$. Clearly, ψ is an immediate variant of φ. In the other direction, since x does not occur free in $((\forall x)R(x) \wedge Q(y))$ and x is substitutable for y in the same formula, it follows that φ is also an immediate variant of ψ. □

Our next goal is to show that the property of being a variant is symmetric, that is, if ψ is a variant of φ, then φ is a variant of ψ.

Theorem 4.6.26. *Let φ, ψ be two formulas. If ψ is an immediate variant of φ, then φ is an immediate variant of ψ.*

Proof. Suppose that ψ is obtained from φ by replacing an occurrence of a subformula $(Qx)\alpha$ with $(Qy)(\alpha)_{x:=y}$, where y does not occur free in α and y is substitutable for x in α. The statement is trivially true when $y = x$. Therefore, we assume $y \neq x$. By Corollary 4.3.84, $x \notin \mathtt{FV}((\alpha)_{x:=y})$. By Corollary 4.3.78, x is substitutable for y in $(\alpha)_{x:=y}$. Since $((\alpha)_{x:=y})_{y:=x} = \alpha$ by Corollary 4.3.87, it follows that φ is an immediate variant of ψ because we can replace the occurrence of $(Qy)(\alpha)_{x:=y}$ in ψ with $(Qx)\alpha$ and thus revert back to φ. □

Corollary 4.6.27. *Let φ, ψ be formulas. If ψ is a variant of φ, then φ is a variant of ψ.*

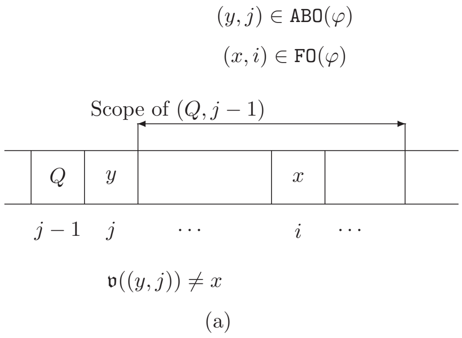
Proof. This statement follows immediately from Theorem 4.6.26. □

An alternative characterization of variants can be obtained by introducing a function that formalizes the renaming of variables which occurs in a variant.

Definition 4.6.28. A *renaming function for a formula* φ, or just a *renaming* for φ if there is no risk of confusion, is a mapping $\mathfrak{v}$: $\mathtt{ABO}(\varphi) \to \mathrm{VAR}$ that satisfies the following conditions:

(1) if $(x, i) \in \mathtt{FO}(\varphi)$, $(y, j) \in \mathtt{ABO}(\varphi)$ and (x, i) is in the scope of the quantifier occurrence $(Q, j - 1)$, then $\mathfrak{v}((y, j)) \neq x$;
(2) if $(x, i) \in \mathtt{PBO}(\varphi)$, $\mathsf{binding}_\varphi((x, i)) = (x, j)$, $(y, k) \in \mathtt{ABO}(\varphi)$ with $j < k < i$, and (x, i) occurs in the scope of the quantifier occurrence $(Q, k - 1)$, then $\mathfrak{v}((x, j)) \neq \mathfrak{v}((y, k))$.

These conditions are illustrated in Figure 4.1.


$(y, j) \in \mathtt{ABO}(\varphi)$
$(x, i) \in \mathtt{FO}(\varphi)$
Scope of $(Q, j - 1)$
Q
y
x
$j - 1$
j
$\cdots$
i
$\cdots$
$\mathfrak{v}((y, j)) \neq x$
(a)

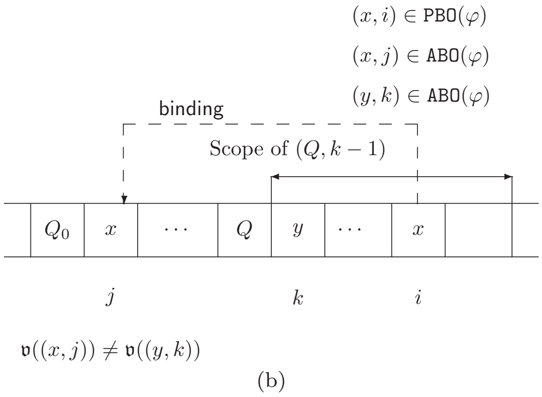

$(x, i) \in \mathtt{PBO}(\varphi)$
$(x, j) \in \mathtt{ABO}(\varphi)$
$(y, k) \in \mathtt{ABO}(\varphi)$
binding
Scope of $(Q, k - 1)$
Q_0
x
$\cdots$
Q
y
$\cdots$
x
j
k
i
$\mathfrak{v}((x, j)) \neq \mathfrak{v}((y, k))$
(b)

Fig. 4.1. Definition of renaming function for a formula.

The formula obtained from φ by applying the renaming $\mathfrak{v}$ is $\varphi^{\mathfrak{v}}$ obtained by replacing each active bound occurrence (x, i) and its associated passive bound occurrences of x by $\mathfrak{v}((x, i))$.

We use the notation $\mathfrak{V}[\varphi, i_0 \to y_0, \ldots, i_{k-1} \to y_{k-1}]$ for a renaming $\mathfrak{v}$ for a formula φ such that the following conditions are satisfied:

(1) $i_0, \ldots, i_{k-1}$ are pairwise distinct numbers such that $0 \leq i_l \leq |\varphi| - 1$ for $0 \leq l \leq k-1$;
(2) $(x_j, i_j) \in \mathtt{ABO}(\varphi)$, for $0 \leq j < k$;
(3) $\mathfrak{v}((x_j, i_j)) = y_j$, for $0 \leq j < k$;
(4) if $(x, h) \in \mathtt{ABO}(\varphi)$ and $h \notin \{i_0, \ldots, i_{k-1}\}$, then $\mathfrak{v}((x, h)) = x$.

We refer to a renaming of the form $\mathfrak{V}[\varphi, i \to y]$ as a *unit renaming.* □

Note that every renaming $\mathfrak{v}$ for a formula φ can be written as $\mathfrak{v} = \mathfrak{V}[\varphi, i_0 \to y_0, \ldots, i_{k-1} \to y_{k-1}]$. Moreover, we can assume that $i_0 < \cdots < i_{k-1}$.

Given formulas φ, ψ such that $\psi = \varphi^{\mathfrak{v}}$, for some renaming $\mathfrak{v}$, we can effectively find $\mathfrak{v}$ as follows. For each active bound occurrence (y, j) in $\mathtt{ABO}(\varphi)$ examine the corresponding active bound occurrence (y', j) in ψ. Then, $\mathfrak{v}((y, j)) = y'$.

Example 4.6.29. Let $\varphi = (\forall x)(\forall y)R(x, y)$. The set of active bound occurrences in φ is $\mathtt{ABO}(\varphi) = \{(x, 2), (y, 6)\}$. Since the occurrence $(x, 10)$ is situated in the scope of $(\forall, 5)$, it follows that $\mathfrak{v}((x, 2)$ must be distinct from $\mathfrak{v}((y, 6))$, for any renaming function $\mathfrak{v}$ for φ. Since φ contains no free occurrences of variables, it follows that any function $\mathfrak{v} : \mathtt{ABO}(\varphi) \longrightarrow \mathrm{VAR}$ that meets the above requirement is renaming function for φ. For instance, we can define $\mathfrak{v}$ by $\mathfrak{v}((x, 2)) = y$ and $\mathfrak{v}((y, 6)) = x$. The resulting formula $\varphi^{\mathfrak{v}}$ is $(\forall y)(\forall x)R(y, x)$. Clearly, we can write $\mathfrak{v} = \mathfrak{V}[\varphi, 2 \to y, 6 \to x]$. □

Example 4.6.30. Let $\psi = (\forall x)(R(x, y) \wedge (\forall z)P(z, x))$. The set of active bound occurrences of ψ is $\{(x, 2), (z, 14)\}$. Since $(y, 10)$ is a free occurrence that falls in the scope of $(\forall, 1)$, we must have $\mathfrak{v}((x, 2)) \neq y$ for any renaming function $\mathfrak{v}$ for ψ. By the second condition of Definition 4.6.28, we must also have $\mathfrak{v}((x, 2)) \neq \mathfrak{v}((z, 14))$ and these are the only restrictions that a renaming function for ψ must obey. Thus, the mapping $\mathfrak{v} = \mathfrak{V}[\varphi, 2 \to w, 14 \to y]$ is a renaming and we have $\psi^{\mathfrak{v}} = (\forall w)(R(w, y) \wedge (\forall y)P(y, w))$. □

We leave to the reader to prove the following statements involving a formula φ and a renaming function $\mathfrak{v}$ for φ:

(RN0) $\varphi^{\mathfrak{v}}$ is a formula;
(RN1) $|\varphi^{\mathfrak{v}}| = |\varphi|$;
(RN2) if $(s, i) \in \mathtt{OCC}(\varphi)$ and $s \notin \text{VAR}$, then $(s, i) \in \mathtt{OCC}(\varphi^{\mathfrak{v}})$;
(RN3) if $(x, i) \in \mathtt{OCC}(\varphi)$ and $x \in \text{VAR}$, then there is $y \in \text{VAR}$ such that $(y, i) \in \mathtt{OCC}(\varphi^{\mathfrak{v}})$;
(RN4) if Q is a quantifier symbol and $(Q, i) \in \mathtt{OCC}(\varphi)$, $\mathsf{scope}_{\varphi}((Q, i)) = (\alpha, i+3)$, and $\mathsf{scope}_{\varphi^{\mathfrak{v}}}((Q, i)) = (\alpha', i+3)$, then $|\alpha| = |\alpha'|$;
(RN5) if $(x, i) \in \mathtt{ABO}(\varphi)$, $y = \mathfrak{v}((x, i))$, then $(y, i) \in \mathtt{ABO}(\varphi^{\mathfrak{v}})$.

Theorem 4.6.31. *Let φ be a formula and $\mathfrak{v}$ be a renaming for φ. The following statements hold*:

(1) *If $(x, i) \in \mathtt{FO}(\varphi)$, then $(x, i) \in \mathtt{FO}(\varphi^{\mathfrak{v}})$.*
(2) *If $(x, i) \in \mathtt{PBO}(\varphi)$, $\mathsf{binding}_{\varphi}((x, i)) = (x, j)$ and $\mathfrak{v}((x, j)) = w$, then $(w, i) \in \mathtt{PBO}(\varphi^{\mathfrak{v}})$ and $\mathsf{binding}_{\varphi^{\mathfrak{v}}}((w, i)) = (w, j)$.*

Proof. For the first part, note that (x, i) is an occurrence in $\varphi^{\mathfrak{v}}$. Suppose that (x, i) were bound in $\varphi^{\mathfrak{v}}$. Then, (x, i) would occur in the scope of a quantifier occurrence (Q, h) followed by an active bound occurrence $(x, h+1)$ in $\varphi^{\mathfrak{v}}$. By the preliminary observations, (Q, h) is also an occurrence in φ and the scopes of (Q, h) in φ and $\varphi^{\mathfrak{v}}$ have the same lengths. Suppose that (Q, h) is followed by $(y, h+1)$ in φ, where $\mathfrak{v}((y, h + 1)) = x$. Since the free occurrence (x, i) occurs in the scope of (Q, h) in φ, it follows by the first condition of Definition 4.6.28 that $\mathfrak{v}((y, h + 1)) \neq x$. This contradiction shows that $(x, i) \in \mathtt{FO}(\varphi^{\mathfrak{v}})$.

For the second part, the preservation of the length of the scope of a quantifier in renaming implies that $(w, i) \in \mathtt{PBO}(\varphi^{\mathfrak{v}})$.

Suppose $\mathsf{binding}_{\varphi^{\mathfrak{v}}}((w, i)) = (w, k)$, where $j < k < i$. This means that there is an active bound occurrence $(y, k) \in \mathtt{ABO}(\varphi)$ such that $\mathfrak{v}((y, k)) = w$. However, this contradicts the second condition of Definition 4.6.28, so $\mathsf{binding}_{\varphi^{\mathfrak{v}}}((w, i)) = (w, j)$. □

Lemma 4.6.32. *A formula ψ is an immediate variant of a formula φ if and only if $\psi = \varphi^{\mathfrak{v}}$, where $\mathfrak{v}$ is a unit renaming for φ.*

Proof. Observe that if (Q, r) is a quantifier occurrence in φ, x is the variable following this quantifier occurrence, and $\mathsf{scope}_{\varphi}((Q, r)) = (\alpha, r + 3)$, then we have $\mathsf{binding}_{\varphi}((x, q)) = (x, r + 1)$ if and only if $(x, q - r - 3) \in \mathtt{FO}(\alpha)$.

Suppose that ψ is an immediate variant of φ obtained by replacing the occurrence of the subformula $((Qx)\alpha, i-1)$ by $(Qy)(\alpha)_{x:=y}$, where the variable y does not occur free in α, and y is substitutable for x in α. Let $\mathfrak{v} = \mathfrak{V}[\varphi, i+1 \rightarrow y]$. We need to prove that $\mathfrak{v}$ is a unit renaming and $\varphi^{\mathfrak{v}} = \psi$. In other words, we need to verify that $\mathfrak{v}$ satisfies both conditions of Definition 4.6.28.

Let $(z,k) \in \mathtt{FO}(\varphi)$, $(w,j) \in \mathtt{ABO}(\varphi)$, where (z,k) is in the scope of $(Q', j-1)$. We need to show that $\mathfrak{v}((w,j)) \neq z$.

If $j \neq i+1$, then $\mathfrak{v}((w,j)) = w$ and we have $z \neq w$ because (z,k) is a free occurrence in φ located in the scope of $(Q', j-1)$.

If $j = i+1$, then (z,k) is part of the occurrence $(\alpha, i+3)$ and, since z occurs free in α, we have $z \neq y = \mathfrak{v}((w,j))$, because y does not occur free in α. Thus, the first condition of Definition 4.6.28 is satisfied.

To verify the second condition, suppose that we have $(z,k) \in \mathtt{PBO}(\varphi)$, $\mathsf{binding}_{\varphi}((z,k)) = (z,j)$, $(w,p) \in \mathtt{ABO}(\varphi)$, where $j < p < k$, and (z,k) is in the scope of $(Q', p-1)$ (see Figure 4.2). We need to prove that $\mathfrak{v}((z,j)) \neq \mathfrak{v}((w,p))$.

We need to consider three cases:

(1) If $j \neq i+1$ and $p \neq i+1$, then $\mathfrak{v}((z,j)) = z$ and $\mathfrak{v}((w,p)) = w$. We have $z \neq w$ because $\mathsf{binding}_{\varphi}((z,k)) = (z,j)$, which is distinct from (w,p).

(2) If $j = i+1$ we have $z = x$, $\mathfrak{v}((z,j)) = y$, and $\mathfrak{v}((w,p)) = w$. Also, $(x, k-i-3) \in \mathtt{FO}(\alpha)$ and the same occurrence belongs to

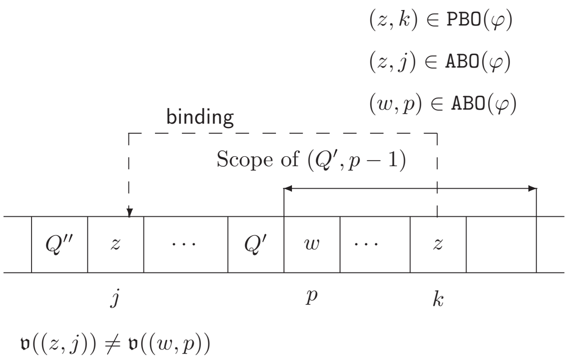

$(z,k) \in \mathtt{PBO}(\varphi)$
$(z,j) \in \mathtt{ABO}(\varphi)$
$(w,p) \in \mathtt{ABO}(\varphi)$
binding
Scope of $(Q', p-1)$
Q''
z
$\cdots$
Q'
w
$\cdots$
z
j
p
k
$\mathfrak{v}((z,j)) \neq \mathfrak{v}((w,p))$

Fig. 4.2. Verification of second condition of definition 4.6.28.

the scope of $(Q', p-i-3)$ in α. Since y is substitutable for x in α, we have $y \neq w$.

(3) Let now $p = i+1$, so $x = w$, $\mathfrak{v}((z,j)) = z$ and $\mathfrak{v}((w,p)) = y$. Since $(z, k-i-3) \in \mathtt{FO}(\alpha)$ and y does not occur free in α, it follows that $z \neq y$.

Thus, we may conclude that $\mathfrak{v}$ is indeed a unit renaming. Our initial observation immediately implies that $\psi = \varphi^{\mathfrak{v}}$.

Suppose that $\psi = \varphi^{\mathfrak{v}}$, where $\mathfrak{v}$ is a unit renaming, $\mathfrak{v} = \mathfrak{V}[\varphi, i+1 \rightarrow y]$. We claim that ψ is an immediate variant of φ. The definition of a unit renaming implies that we have an occurrence (Q,i) in φ, where Q is a quantifier symbol. Let $(\alpha, i+3)$ be the scope of this quantifier occurrence. Then, $\varphi^{\mathfrak{v}}$ is obtained from φ by replacing the occurrence $((Qx)\alpha), i-1)$ by $(Qy)(\alpha)_{x:=y}$. If $y = x$, then $\psi = \varphi$, so we can assume that $y \neq x$. In order for $\varphi^{\mathfrak{v}}$ to be an immediate variant of φ, we need to prove that y does not occur free in α and y is substitutable for x in α.

Suppose that y occurs free in α, that is, $(y,p) \in \mathtt{FO}(\alpha)$. The corresponding occurrence $(y, p+i+3)$ in φ may be free in φ or bound in this formula.

If $(y, p+i+3) \in \mathtt{FO}(\varphi)$, this would violate the first condition of Definition 4.6.28. If, on the other hand, $(y, p+i+3) \in \mathtt{PBO}(\varphi)$ and $\mathsf{binding}_{\varphi}((y, p+i+3)) = (y,q)$, then $q < i+1$, which would violate the second condition of the same definition. Thus, y does not occur free in α.

Suppose that y were not substitutable for x in α, which means that an occurrence $(x,p) \in \mathtt{FO}(\alpha)$ is in the scope of a quantifier symbol occurrence $(Q_1, r) \in \mathtt{OCC}_{Q_1}(\alpha)$ and $(y, r+1) \in \mathtt{ABO}(\alpha)$. Since $(x, p+i+3) \in \mathtt{PBO}(\alpha)$ and $\mathsf{binding}_{\alpha}((x, p+i+3)) = i+1$ and $(x, p+i+3) \in \mathsf{scope}_{\varphi}((Q_1, r+i+3))$, this contradicts the second condition of Definition 4.6.28. Thus, y is substitutable for x in α. □

Theorem 4.6.33. *Let $\mathfrak{v}$ be a renaming for the formula φ and let $\mathfrak{w}$ be a renaming for the formula $\varphi^{\mathfrak{v}}$. The mapping $\mathfrak{w} * \mathfrak{v} : \mathtt{ABO}(\varphi) \longrightarrow VAR$ given by*

$$\mathfrak{w} * \mathfrak{v}((x,p)) = \mathfrak{w}((\mathfrak{v}(x,p),p))$$

for every $(x,p) \in \mathtt{ABO}(\varphi)$ is a renaming for φ and $(\varphi^{\mathfrak{v}})^{\mathfrak{w}} = \varphi^{\mathfrak{w}*\mathfrak{v}}$.

Proof. The definition of $\mathfrak{w} * \mathfrak{v}$ is correct because $(\mathfrak{v}(x,p),p) \in \texttt{ABO}(\varphi^{\mathfrak{v}})$ for every $(x,p) \in \texttt{ABO}(\varphi)$ as we stated in **(RN5)**. We leave to the reader to verify that $\mathfrak{w} * \mathfrak{v}$ satisfies the conditions of Definition 4.6.28 and the straightforward verification of the last formula. □

It is easy to see that if $\mathfrak{w} = \mathfrak{V}[\varphi, i_0 \to y_0, \ldots, i_{k-1} \to y_{k-1}]$, then

$$\begin{aligned}&\mathfrak{V}[\varphi^{\mathfrak{w}}, j_0 \to z_0, \ldots, j_{l-1} \to z_{l-1}]\\ &\quad = \mathfrak{V}[\varphi, j_0 \to z_0, \ldots, j_{l-1} \to z_{l-1}, i_{h_0} \to y_{h_0}, \ldots, i_{h_{r-1}} \to y_{h_{r-1}}],\end{aligned} \tag{4.7}$$

where $\{i_{h_0}, \ldots, i_{h_{r-1}}\} = \{i_0, \ldots, i_{k-1}\} - \{j_0, \ldots, j_{l-1}\}$.

The next theorem gives us an algorithm for decomposing a renaming into a composition of unit renamings.

Theorem 4.6.34. *Let $\mathfrak{v}$ be a renaming for a formula φ, $\mathfrak{v} = \mathfrak{V}[\varphi, i_0 \to y_0, \ldots, i_{k-1} \to y_{k-1}]$, where $\{i_0, \ldots, i_{k-1}\}$ consists all indices of active bound occurrences in φ. Then, there is a sequence of formulas $(\alpha_0, \ldots, \alpha_{2k})$ such that $\alpha_0 = \varphi$ and $\alpha_{2k} = \varphi^{\mathfrak{v}}$, and a sequence of mappings $(\mathfrak{v}_1, \ldots, \mathfrak{v}_{2k})$ such that $\mathfrak{v}_i$ is a unit renaming for α_{i-1}, and $\alpha_i = \alpha_{i-1}^{\mathfrak{v}_i}$, for $1 \le i \le 2k$.*

Proof. Without loss of generality, we assume that $i_0 < \cdots < i_{k-1}$ and $(x_l, i_l) \in \texttt{OCC}(\varphi)$ for $0 \le l \le k-1$. Let $z_0, \ldots, z_{k-1}$ be k variables that occur neither in φ nor in the set $\{y_0, \ldots, y_{k-1}\}$. We define recursively the formulas $\alpha_0, \ldots, \alpha_k$ as follows: $\alpha_0 = \varphi$. Suppose that we have defined α_r for $0 \le r \le k-1$. Then, it easy to see that $\mathfrak{v}_{r+1} = \mathfrak{V}[\alpha_r, i_r \to z_r]$ is a renaming for α_r because of the definition of the variable z_r and we can define $\alpha_{r+1} = \alpha_r^{\mathfrak{v}_{r+1}}$. By Equality (4.7), the formula α_k can be written as $\alpha_k = \varphi^{\mathfrak{w}}$, where $\mathfrak{w} = \mathfrak{V}[\varphi, i_0 \to z_0, \ldots, i_{k-1} \to z_{k-1}]$. The second part of the sequence $\alpha_0, \ldots, \alpha_{2k}$ is also defined recursively. Suppose that α_q is defined, where $k \le q \le 2k-1$. We claim that $\mathfrak{v}_{q+1} = \mathfrak{V}[\alpha_q, i_{q-k} \to y_{q-k}]$ is a renaming for α_q. After justifying this claim, we will define $\alpha_{q+1} = \alpha_q^{\mathfrak{v}_{q+1}}$. Since α_q is obtained from φ through a sequence of renamings, a free occurrence (s,i) of a variable in α_q corresponds to a free occurrence (s,i) of the same variable in φ. Suppose that (s,i) is in the scope of the quantifier occurrence $(Q', i_{q-k}-1)$ in both φ and α_q. Since $\mathfrak{v}$ is a renaming for φ, $y_{q-k} = \mathfrak{v}((x_{q-k}, i_{q-k})) \neq s$, which

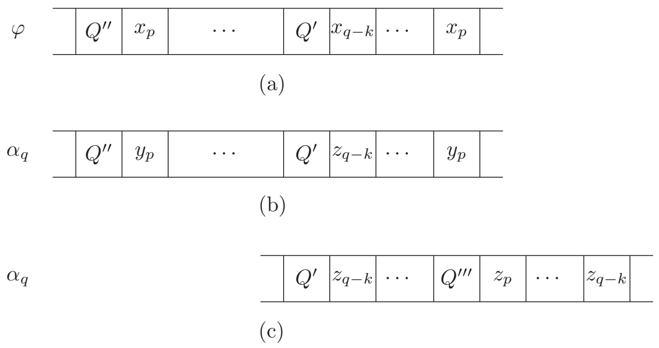


Fig. 4.3. Renamings affecting α_q.

shows that $\mathfrak{v}_{q+1}$ satisfies the first condition of Definition 4.3.33. To prove the satisfaction of the second condition of the definition, we need to consider two cases shown in Figure 4.3(b) and (c) respectively. If α_q contains the occurrences shown in Figure 4.3(b), the original formula φ contains the occurrences shown in Figure 4.3(a). Because $\mathfrak{v}$ is a renaming for φ, we know that $y_{q-k} \neq y_p$. If the situation shown in Figure 4.3(c) occurs, then $y_{q-k} \neq z_p$ because of the definition of the variables $z_0, \ldots, z_{k-1}$. Thus, $\mathfrak{v}_{q+1}$ satisfies the second condition of Definition 4.3.33. This allows us to conclude that $\alpha_{2k} = \varphi^{\mathfrak{u}}$, where $\mathfrak{u} = \mathfrak{v}_{2k} * \cdots * \mathfrak{v}_{k+1} * \mathfrak{w}$. This last composition is $\mathfrak{v}$, by Equality (4.7). □

Theorem 4.6.35. *A formula ψ is variant of a formula φ if and only if there is a renaming function $\mathfrak{v}$ for φ such that $\psi = \varphi^{\mathfrak{v}}$.*

Proof. Suppose that ψ is a variant of φ. Then, there exists a sequence of formulas $\varphi_0, \ldots, \varphi_{n-1}$ such that $\varphi = \varphi_0$, $\varphi_{n-1} = \psi$ and each formula φ_i is an immediate variant of the formula φ_{i-1} for $1 \leq i \leq n-1$. By Lemma 4.6.32, we have renamings $\mathfrak{v}_1, \ldots, \mathfrak{v}_{n-1}$ such that $\varphi_i = \varphi_{i-1}^{\mathfrak{v}_i}$, for $1 \leq i \leq n-1$, which allows us to write $\psi = \varphi^{\mathfrak{v}}$, where $\mathfrak{v} = (\mathfrak{v}_{n-1} * (\cdots * (\mathfrak{v}_2 * \mathfrak{v}_1)))$.

Conversely, suppose that $\psi = \varphi^{\mathfrak{v}}$, where $\mathfrak{v}$ is a renaming for φ. By Theorem 4.6.34, there is a sequence of formulas $\varphi_0, \ldots . \varphi_{n-1}$ and a sequence of unit renamings $\mathfrak{v}_1, \ldots, \mathfrak{v}_{n-1}$ such that $\mathfrak{v}_i$ is a unit renaming for φ_{i-1} and $\varphi_i = \varphi_{i-1}^{\mathfrak{v}_i}$, for $1 \leq i \leq n-1$, $\varphi = \varphi_0$, and $\varphi_{n-1} = \psi$. Applying again Lemma 4.6.32, each formula φ_i is an immediate variant of φ_{i-1}, which allows to conclude that ψ is a variant of φ. □

Theorem 4.6.36. *Given two formulas φ, ψ such that ψ is a variant of φ, we can find effectively a sequence of formulas $(\theta_0, \ldots, \theta_{n-1})$ such that $\varphi = \theta_0$, $\psi = \theta_{n-1}$, and θ_{i+1} is an immediate variant of θ_i, for $0 \leq i \leq n-2$.*

Proof. By Theorem 4.6.35, there is a renaming $\mathfrak{v}$ for φ such that $\psi = \varphi^{\mathfrak{v}}$ and we have observed that this renaming can be effectively determined. By examining the proof of Theorem 4.6.34, we see that we can find effectively a sequence of formulas $(\theta_0, \ldots, \theta_{n-1})$ that satisfies the conditions of the theorem. $\square$

Example 4.6.37. Let

$$\varphi = (\forall x_0)(\forall x_1)R(x_0, x_1) \quad \text{and} \quad \psi = (\forall x_1)(\forall x_0)R(x_1, x_0).$$

Note that $\psi = \varphi^{\mathfrak{v}}$, where $\mathfrak{v} = \mathfrak{V}[\varphi, 2 \to x_1, 6 \to x_0]$ is a renaming for φ. Also observe that $\mathfrak{v}' : \mathtt{ABO}(\varphi) \to \text{VAR}$ given by $\mathfrak{v}'((x_0, 2)) = x_1$ and $\mathfrak{v}'((x_1, 6)) = x_1$ is not a renaming for φ because it violates the second condition of Definition 4.3.33. To decompose $\mathfrak{v}$ into a composition of unit renamings, we could use two new variables z_0, z_1 and consider the following sequence of formulas and renamings:

$$\begin{array}{ll}
\varphi_0 = \varphi & \\
\mathfrak{v}_1 = \mathfrak{V}[\varphi_0, 2 \to z_0] & \varphi_1 = \varphi_0^{\mathfrak{v}_1} = (\forall z_0)(\forall x_1)R(z_0, x_1) \\
\mathfrak{v}_2 = \mathfrak{V}[\varphi_1, 6 \to z_1] & \varphi_2 = \varphi_1^{\mathfrak{v}_2} = (\forall z_0)(\forall z_1)R(z_0, z_1) \\
\mathfrak{v}_3 = \mathfrak{V}[\varphi_2, 2 \to x_1] & \varphi_3 = \varphi_2^{\mathfrak{v}_3} = (\forall x_1)(\forall z_1)R(x_1, z_1) \\
\mathfrak{v}_4 = \mathfrak{V}[\varphi_3, 6 \to x_0] & \varphi_4 = \varphi_3^{\mathfrak{v}_4} = (\forall x_1)(\forall x_0)R(x_1, x_0) = \psi.
\end{array}$$

This illustrates the constructive proof of Theorem 4.6.34. $\square$

Theorem 4.6.38. *Let V be an infinite set of variables and suppose that ψ is a variant of φ such that $\mathtt{BV}(\varphi) \cup \mathtt{BV}(\psi) \subseteq V$. Then, there is a sequence of immediate variants $\varphi_0, \ldots, \varphi_{n-1}$ such that $\varphi = \varphi_0$, $\varphi_{n-1} = \psi$ and $\mathtt{BV}(\varphi_i) \subseteq V$, for $0 \leq i \leq n-1$.*

Proof. This statement can be derived from the proof of Theorem 4.6.34, where we did not restrict the choice of the intermediate variables z_i. Since V is an infinite set, and φ, ψ contain a finite set of variables, it follows we can select the variables z_i from among the variables of the set V. $\square$

Corollary 4.6.39. *Let φ, ψ be two formulas that are variants and let z be a variable that does not occur in either φ or ψ. There is a sequence of immediate variants, $\varphi_0, \ldots, \varphi_{n-1}$ such that $\varphi = \varphi_0$, $\varphi_{n-1} = \psi$ and z does not occur in any of the formulas φ_i for $0 \leq i \leq n-1$.*

Proof. Let $V = \text{VAR} - \{z\}$. It is clear that $\mathtt{BV}(\varphi) \cup \mathtt{BV}(\psi) \subseteq V$ and V is an infinite set, so, by Theorem 4.6.38, there is a sequence of immediate variants $\varphi_0, \ldots, \varphi_{n-1}$ such that $\varphi = \varphi_0$, $\varphi_{n-1} = \psi$ and $\mathtt{BV}(\varphi_i) \subseteq V$, which implies that $z \not\in \mathtt{BV}(\varphi_i)$ for $0 \leq i \leq n-1$. By Theorem 4.6.22, $\mathtt{FV}(\varphi_i) = \mathtt{FV}(\varphi) = \mathtt{FV}(\psi)$, for $0 \leq i \leq n-1$, so $z \not\in \mathtt{FV}(\varphi_i)$, which allows us to conclude that $z \not\in \mathtt{V}(\varphi_i)$, for $0 \leq i \leq n-1$. □

The following technical result is needed in Section 5.2.

Theorem 4.6.40. *Let φ, ψ be two formulas and let t be a term such that $\mathtt{V}(t) \cap (\mathtt{BV}(\varphi) \cup \mathtt{BV}(\psi)) = \emptyset$. If ψ is an immediate variant of φ, then $\mathsf{s}^c_t(\psi)$ is an immediate variant of $\mathsf{s}^c_t(\varphi)$.*

Proof. Suppose that ψ is obtained from φ by replacing an occurrence of a subformula $(Qx)\alpha$ by $(Qy)(\alpha)_{x:=y}$, where y does not occur free in α and y is substitutable for x in α. By obvious properties of textual substitutions, $\mathsf{s}^c_t(\psi)$ is obtained from $\mathsf{s}^c_t(\varphi)$ by replacing an occurrence of $(Qx)\mathsf{s}^c_t(\alpha)$ by $(Qy)\mathsf{s}^c_t((\alpha)_{x:=y})$. Observe that x and y do not occur in t because x occurs bound in φ and y occurs bound in ψ. By Equality (4.2) on page 571, we have $(Qy)\mathsf{s}^c_t((\alpha)_{x:=y}) = (Qy)(\mathsf{s}^c_t(\alpha))_{x:=y}$.

Note that, by Theorem 4.3.69, $y \not\in \mathtt{FV}(\mathsf{s}^c_t(\alpha))$ because y does not occur in t and does not occur free in α. In addition, y is substitutable for x in $\mathsf{s}^c_t(\alpha)$ by Corollary 4.3.79, since t is substitutable for x in α (because no variable of t occurs bound in α) and $x \not\in \mathtt{V}(t)$. Thus, $\mathsf{s}^c_t(\psi)$ is an immediate variant of $\mathsf{s}^c_t(\varphi)$. □

Corollary 4.6.41. *Let φ, ψ be two formulas and let t be a term such that $\mathtt{V}(t) \cap (\mathtt{BV}(\varphi) \cup \mathtt{BV}(\psi)) = \emptyset$. If ψ is a variant of φ, then $\mathsf{s}^c_t(\psi)$ is a variant of $\mathsf{s}^c_t(\varphi)$.*

Proof. By Theorem 4.6.38, there exists a sequence of immediate variants $\varphi_0, \ldots, \varphi_{n-1}$ such that $\varphi = \varphi_0$, $\psi = \varphi_{n-1}$ and no variable of t occurs bound in any of the formulas φ_i. By Theorem 4.6.40, each formula $\mathsf{s}^c_t(\varphi_{i+1})$ is an immediate variant of $\mathsf{s}^c_t(\varphi_i)$, for $0 \leq i \leq n-2$, which gives the desired result. □

If t is a term, which is not substitutable for a variable x in a formula φ, then we can produce a variant of φ in which t is substitutable for x. We will first illustrate this with an example and then prove a general theorem.

Example 4.6.42. Let $\varphi = (\exists y)(\forall z)R(x,y,z)$ and let $t = f(y,z)$, where x, y, z are distinct variables, R is relation symbol and f is a function symbol. It is clear that t is not substitutable for x in φ. However, by applying Theorem 4.6.18, we can rename y, z as the new variables y', z', and thus obtain the variant $\varphi' = (\exists y')(\forall z')R(x,y',z')$ of φ in which t is substitutable for x.

Let $\psi = ((\forall y)P(x,y) \wedge (\forall z)Q(x,z))$, where P and Q are binary relation symbols. Again, it is clear that t is not substitutable for x in ψ. However, t is substitutable for x in the variant $\alpha = ((\forall y')P(x,y') \wedge (\forall z')Q(x,z'))$ of ψ. ☐

Next, we show how to systematically produce a variant of a formula φ such that a term t is substitutable for a variable x in the variant.

Definition 4.6.43. The function $\mathsf{variant} : \text{FORM} \times \text{VAR} \times \text{TERM} \longrightarrow \text{FORM}$ is given by the following recursive definition.

- $\mathsf{variant}(\varphi,x,t)$ is φ if φ is atomic;
- $\mathsf{variant}(\varphi,x,t)$ is $(\neg\mathsf{variant}(\psi,x,t))$ if $\varphi = (\neg\psi)$;
- $\mathsf{variant}(\varphi,x,t)$ is $(\mathsf{variant}(\alpha,x,t)\ C\ \mathsf{variant}(\beta,x,t))$ if $\varphi = (\alpha C \beta)$ for C a binary connective symbol;
- $\mathsf{variant}(\varphi,x,t)$ is φ if $\varphi = (Qy)\psi$ and $x \notin \mathtt{FV}(\varphi)$ for Q a quantifier symbol;
- $\mathsf{variant}(\varphi,x,t)$ is $(Qy)\mathsf{variant}(\psi,x,t)$ if $\varphi = (Qy)\psi$, $x \in \mathtt{FV}(\varphi)$, and y does not occur in t;
- $\mathsf{variant}(\varphi,x,t)$ is $(Qz)(\mathsf{variant}(\psi,x,t))_{y:=z}$ if $\varphi = (Qy)\psi$, $x \in \mathtt{FV}(\varphi)$, y occurs in t and z is the first variable not occurring in $\mathsf{variant}(\psi,x,t)$ or in t.

☐

It is easy to verify by induction on φ that $\mathsf{variant}(\varphi,x,t)$ is indeed a variant of φ and that if t is substitutable for x in φ, then $\mathsf{variant}(\varphi,x,t) = \varphi$.

The following lemma helps in the proof of Theorem 4.6.47.

Example 4.6.44. Let φ, ψ, x, and t be as in Example 4.6.42. Then, the formula $\mathsf{variant}(\varphi, x, t)$ is the formula φ' of that example, where z', y' are the first two variables that are different from x, y, z. However, $\mathsf{variant}(\psi, x, t) = ((\forall y')R(x, y') \wedge (\forall y')Q(x, y'))$, where y' is the first variable different from the variables x, y, z. □

We denote the formula $(\mathsf{variant}(\varphi, x, t))_{x:=t}$ by $\langle\varphi\rangle_{x:=t}$. Observe that if t is substitutable for x in φ, then since $\mathsf{variant}(\varphi, x, t) = \varphi$, $\langle\varphi\rangle_{x:=t} = (\varphi)_{x:=t}$.

Example 4.6.45. Again, if φ, ψ, x, and t are as in Example 4.6.42, we have

$$\langle\varphi\rangle_{x:=t} = (\exists y')(\forall z')R(f(y, z), y', z')$$
$$\langle\psi\rangle_{x:=t} = ((\forall y')P(f(y, z), y') \wedge (\forall y')Q(f(y, z), y')).$$

Contrast these formulas with

$$(\varphi)_{x:=t} = (\exists y)(\forall z)R(f(y, z), y, z)$$
$$(\psi)_{x:=t} = ((\forall y)P(f(y, z), y) \wedge (\forall z)Q(f(y, z), z)).$$

□

Lemma 4.6.46. *Let φ be a formula, t be a term, and let x, y, z be variables such that $x \neq z$. If t is substitutable for x in φ, then t is substitutable for x in $(\varphi)_{y:=z}$.*

Proof. The argument is by induction on φ. We leave to the reader the basis step, when φ is atomic. Also, we leave to the reader the inductive steps that correspond to $\varphi = (\neg\psi)$ and $\varphi = (\alpha C \beta)$. Suppose now that the statement holds for ψ and let $\varphi = (Qw)\psi$, where Q is a quantifier symbol and w is a variable. If $y = w$, then $(\varphi)_{y:=z} = \varphi$, and the result is immediate. If $y \neq w$, then $(\varphi)_{y:=z} = (Qw)(\psi)_{y:=z}$. Since t is substitutable for x in φ, one of the following two cases occurs:

(1) x does not occur free in φ. In this case, if $x = w$, then x does not occur free in $(\varphi)_{y:=z}$, so t is substitutable for x in this formula. If $x \neq w$, then x does not occur free in ψ, so, by Exercise 32 and the fact that $z \neq x$, it follows that x does not occur free in $(\psi)_{y:=z}$.

Therefore, x does not occur free in $(Qw)(\psi)_{y:=z} = (\varphi)_{y:=z}$, which implies the substitutability of t for x in this formula.

(2) t is substitutable for x in ψ and w does not occur in t. Now, by inductive hypothesis, t is substitutable for x in $(\psi)_{y:=z}$ which gives t substitutable for x in $(Qw)(\psi)_{y:=z} = (\varphi)_{y:=z}$.

□

Theorem 4.6.47. *Let φ be a formula, t be a term, and let x be a variable. Then, $\mathsf{variant}(\varphi, x, t) \equiv \varphi$, t is substitutable for x in $\mathsf{variant}(\varphi, x, t)$, and $\mathtt{FV}(\mathsf{variant}(\varphi, x, t)) = \mathtt{FV}(\varphi)$.*

Proof. We have already observed that $\mathsf{variant}(\varphi, x, t)$ is a variant of φ, and hence is logically equivalent to φ and has the same set of free variables by Theorem 4.6.22.

To prove the second part of the theorem, we proceed by induction on φ. We discuss only the inductive step when $\varphi = (Qy)\psi$, $x \in \mathtt{FV}(\varphi)$ and y occurs in t. (The basis step and the remaining inductive steps are straightforward.) We assume, by inductive hypothesis, that t is substitutable for x in $\mathsf{variant}(\psi, x, t)$. Note that $x \in \mathtt{FV}(\varphi)$, which implies $x \in \mathtt{FV}(\psi)$, which in turn implies $x \in \mathtt{FV}(\mathsf{variant}(\psi, x, t))$, by inductive hypothesis. Since $\mathsf{variant}(\varphi, x, t) = (Qz)(\mathsf{variant}(\psi, x, t))_{y:=z}$, where z is the first variable that does not occur in $\mathsf{variant}(\psi, x, t)$ or in t, it follows that $z \neq x$. Thus, by inductive hypothesis and Lemma 4.6.46, it follows that t is substitutable for x in $(\mathsf{variant}(\psi, x, t))_{y:=z}$. Since z does not occur in t, t is substitutable for x in $((Qz)\mathsf{variant}(\psi, x, t))_{y:=z} = \mathsf{variant}(\varphi, x, t)$. □

Corollary 4.6.48. *Let φ be a formula, t be a term, and let x be a variable. Then, $\mathtt{FV}(\langle\varphi\rangle_{x:=t}) \subseteq (\mathtt{FV}(\varphi) - \{x\}) \cup \mathtt{V}(t)$.*

Proof. We have the following:

$$\mathtt{FV}(\langle\varphi\rangle_{x:=t}) = \mathtt{FV}((\mathsf{variant}(\varphi, x, t))_{x:=t})$$
$$\text{(by definition of } \langle\varphi\rangle_{x:=t}\text{)}$$

$$\begin{aligned}
&\subseteq (\mathtt{FV}(\mathsf{variant}(\varphi, x, t)) - \{x\}) \cup \mathtt{V}(t)\\
&\quad \text{(by Corollary 4.3.84)}\\
&= (\mathtt{FV}(\varphi) - \{x\}) \cup \mathtt{V}(t)\\
&\quad \text{(by Theorem 4.6.47)}
\end{aligned}$$

□

Next, we present a version of the Substitution Corollary for $n = 1$, where we drop the requirement of substitutability for the term involved.

Corollary 4.6.49. *Let $\mathcal{L}$ be a first-order language, φ be an $\mathcal{L}$-formula, t be an $\mathcal{L}$-term and x be a variable. If $\mathcal{A}$ is an $\mathcal{L}$-structure and $\sigma \in$ ASSIGN$_{\mathcal{A}}$, then $(\mathcal{A}, \sigma) \models \langle\varphi\rangle_{x:=t}$ if and only if $(\mathcal{A}, [x \to \sigma^{\mathcal{A}}(t)]\sigma) \models \varphi$.*

Proof. We have the following equivalent statements:

$$\begin{aligned}
(\mathcal{A}, \sigma) &\models \langle\varphi\rangle_{x:=t}\\
(\mathcal{A}, \sigma) &\models (\mathsf{variant}(\varphi, x, t))_{x:=t}\\
&\quad \text{(by definition of } \langle\varphi\rangle_{x:=t})\\
(\mathcal{A}, [x \to \sigma^{\mathcal{A}}(t)]\sigma) &\models \mathsf{variant}(\varphi, x, t)\\
&\quad \text{(by the Substitution Corollary and}\\
&\quad \text{Theorem 4.6.47)}\\
(\mathcal{A}, [x \to \sigma^{\mathcal{A}}(t)]\sigma) &\models \varphi\\
&\quad \text{(by Theorem 4.6.47).}
\end{aligned}$$

□

Theorem 4.6.50. *Let φ, ψ be two first-order formulas, x be a variable, and t be a term. Then, we have $\langle(\neg\varphi)\rangle_{x:=t} = (\neg\langle\varphi\rangle_{x:=t})$ and $\langle(\varphi C \psi)\rangle_{x:=t} = (\langle\varphi\rangle_{x:=t} C \langle\psi\rangle_{x:=t})$ for every binary connective symbol C.*

Proof. The first part of the theorem follows from the following equalities:

$$\begin{aligned}
\langle(\neg\varphi)\rangle_{x:=t} &= (\mathsf{variant}((\neg\varphi), x, t))_{x:=t}\\
&\quad \text{(by the definition of } \langle\alpha\rangle_{x:=t})
\end{aligned}$$

$$
\begin{aligned}
&= ((\neg\mathsf{variant}(\varphi, x, t)))_{x:=t} \\
&\quad \text{(by Definition 4.6.43)} \\
&= (\neg(\mathsf{variant}(\varphi, x, t))_{x:=t}) \\
&\quad \text{(by Lemma 4.3.53)} \\
&= (\neg\langle\varphi\rangle_{x:=t}) \\
&\quad \text{(by the definition of } \langle\alpha\rangle_{x:=t}).
\end{aligned}
$$

We leave the second part of the theorem to the reader. □

The following result extends the case $n = 1$ of Theorem 4.6.7 because it drops the requirement of substitutability.

Theorem 4.6.51. *If φ is a formula, x is a variable and t is a term, then $(\forall x)\varphi \models \langle\varphi\rangle_{x:=t}$ and $\langle\varphi\rangle_{x:=t} \models (\exists x)\varphi$.*

Proof. Let $\mathcal{L}$ be a first-order language such that φ is an $\mathcal{L}$-formula and t is an $\mathcal{L}$-term. If $(\mathcal{A}, \sigma) \models (\forall x)\varphi$ for some $\mathcal{L}$-structure $\mathcal{A}$ and $\sigma \in \mathrm{ASSIGN}_{\mathcal{A}}$, then $(\mathcal{A}, [x \to a]\sigma) \models \varphi$ for every $a \in |\mathcal{A}|$. In particular, $(\mathcal{A}, [x \to \sigma^{\mathcal{A}}(t)]\sigma) \models \varphi$, so, by Corollary 4.6.49, $(\mathcal{A}, \sigma) \models \langle\varphi\rangle_{x:=t}$.

For the second part, suppose that $(\mathcal{A}, \sigma) \models \langle\varphi\rangle_{x:=t}$. By Corollary 4.6.49, we have $(\mathcal{A}, [x \to \sigma^{\mathcal{A}}(t)]) \models \varphi$. This, in turn, gives $(\mathcal{A}, \sigma) \models (\exists x)\varphi$, which concludes our argument. □

An immediate consequence of Theorem 4.6.12 is given next.

Theorem 4.6.52. *Let Γ be a set of formulas and let c be a constant symbol that does not occur in any formula of Γ.*

(1) *If $\Gamma \cup \{(\exists x)\varphi\}$ is satisfiable, then $\Gamma \cup \{\langle\varphi\rangle_{x:=c}\}$ is satisfiable.*
(2) *If $\Gamma \cup \{(\neg(\forall x)\varphi)\}$ is satisfiable, then $\Gamma \cup \{\langle(\neg\varphi)\rangle_{x:=c}\}$ is satisfiable.*

Proof. The statements follow immediately from Theorem 4.6.12 and the fact that $\langle\psi\rangle_{x:=c} = (\psi)_{x:=c}$, for every formula ψ. □

Theorem 4.6.53. *Suppose that Δ is a set of signed formulas, $b(Qx)\varphi$ is a $\boldsymbol{\delta}$-formula, and c is a constant symbol that does not occur in any formula of $\Delta \cup \{b(Qx)\varphi\}$. If $\Delta \cup \{b(Qx)\varphi\}$ is satisfiable, then $\Delta \cup \{b\langle\varphi\rangle_{x:=c}\}$ is satisfiable.*

Proof. The statement follows immediately from Theorem 4.6.13 and the fact that $\langle\varphi\rangle_{x:=c} = (\varphi)_{x:=c}$. □

Theorem 4.6.54. *Let φ be a formula, x, y be distinct variables, t be a term and c be a constant symbol. Then, if y does not occur in $\mathit{variant}(\varphi, x, t)$ or in φ, we have*:

$$\mathit{s}_y^c(\mathit{variant}(\varphi, x, t)) = \mathit{variant}(\mathit{s}_y^c(\varphi), x, \mathit{s}_y^c(t)).$$

Proof. We proceed by induction on the formula φ. If φ is atomic, then $\mathsf{s}_y^c(\varphi)$ is also an atomic formula and therefore $\mathsf{s}_y^c(\mathsf{variant}(\varphi, x, t)) = \mathsf{s}_y^c(\varphi) = \mathsf{variant}(\mathsf{s}_y^c(\varphi), x, \mathsf{s}_y^c(t))$.

Suppose the statement holds for the formula ψ and $\varphi = (\neg\psi)$. Then, we can write

$$\begin{aligned}
\mathsf{s}_y^c(\mathsf{variant}(\varphi, x, t)) &= \mathsf{s}_y^c(\mathsf{variant}((\neg\psi), x, t)) \\
&= (\neg\mathsf{s}_y^c(\mathsf{variant}(\psi, x, t))) \\
&\quad \text{(by Definition 4.6.43)} \\
&= (\neg\mathsf{variant}(\mathsf{s}_y^c(\psi), x, \mathsf{s}_y^c(t))) \\
&\quad \text{(by inductive hypothesis, because} \\
&\quad y \text{ does not occur in } \mathsf{variant}(\psi, x, t) \text{ or in } \psi) \\
&= \mathsf{variant}(\mathsf{s}_y^c((\neg\psi)), x, \mathsf{s}_y^c(t)) \\
&\quad \text{(by Definition 4.6.43)} \\
&= \mathsf{variant}(\mathsf{s}_y^c(\varphi), x, \mathsf{s}_y^c(t)).
\end{aligned}$$

The case when $\varphi = (\psi_0 C \psi_1)$, where C is a binary connective symbol, and ψ_0, ψ_1 are two formulas for which the statement holds, is similar to the one above and is omitted.

Suppose now that $\varphi = (Qz)\psi$, where Q is a quantifier symbol and ψ is a formula for which the statement holds. We need to consider three subcases:

(1) $x \notin \mathtt{FV}(\varphi)$;
(2) $x \in \mathtt{FV}(\varphi)$ and z does not occur in t;
(3) $x \in \mathtt{FV}(\varphi)$ and z occurs in t.

In the first case, $x \notin \mathtt{FV}((Qz)\mathsf{s}_y^c(\psi))$. This is because, by Theorem 4.3.69, $\mathtt{FV}((Qz)\mathsf{s}_y^c(\psi)) \subseteq \mathtt{FV}(\varphi) \cup \{y\}$. Then,

$$\begin{aligned}
\mathsf{variant}(\mathsf{s}_y^c(\varphi), x, \mathsf{s}_y^c(t)) &= \mathsf{variant}((Qz)\mathsf{s}_y^c(\psi), x, \mathsf{s}_y^c(t))\\
&= (Qz)\mathsf{s}_y^c(\psi)\\
&= \mathsf{s}_y^c((Qz)\psi)\\
&= \mathsf{s}_y^c(\varphi)\\
&= \mathsf{s}_y^c(\mathsf{variant}(\varphi, x, t)).
\end{aligned}$$

In the second case, we have:

$$\begin{aligned}
\mathsf{s}_y^c(\mathsf{variant}(\varphi, x, t)) &= \mathsf{s}_y^c((Qz)\mathsf{variant}(\psi, x, t))\\
&= (Qz)\mathsf{s}_y^c(\mathsf{variant}(\psi, x, t))\\
&= (Qz)\mathsf{variant}(\mathsf{s}_y^c(\psi), x, \mathsf{s}_y^c(t))\\
&\quad \text{(by inductive hypothesis)}\\
&= \mathsf{variant}((Qz)\mathsf{s}_y^c(\psi), x, \mathsf{s}_y^c(t))\\
&\quad \text{(because } y \neq z\text{)}\\
&= \mathsf{variant}(\mathsf{s}_y^c((Qz)\psi), x, \mathsf{s}_y^c(t))\\
&= \mathsf{variant}(\mathsf{s}_y^c(\varphi), x, \mathsf{s}_y^c(t)).
\end{aligned}$$

In the third case, let w be the first variable that does not occur in $\mathsf{variant}(\psi, x, t)$ or in t, so $\mathsf{variant}(\varphi, x, t) = (Qw)(\mathsf{variant}(\psi, x, t))_{z:=w}$. Since y does not occur in either φ or in $\mathsf{variant}(\varphi, x, t)$, it follows that $y \neq w$, $y \neq z$ and y does not occur in $\mathsf{variant}(\psi, x, t)$ or in ψ. Thus, we can write

$$\begin{aligned}
\mathsf{s}_y^c(\mathsf{variant}(\varphi, x, t)) &= \mathsf{s}_y^c((Qw)(\mathsf{variant}(\psi, x, t))_{z:=w})\\
&= (Qw)\mathsf{s}_y^c((\mathsf{variant}(\psi, x, t))_{z:=w})\\
&= (Qw)(\mathsf{s}_y^c(\mathsf{variant}(\psi, x, t)))_{z:=w}\\
&\quad \text{(by Corollary 4.3.74 because } z \neq y\text{)}\\
&= (Qw)(\mathsf{variant}(\mathsf{s}_y^c(\psi), x, \mathsf{s}_y^c(t)))_{z:=w}\\
&\quad \text{(by inductive hypothesis).}
\end{aligned}$$

On the other hand, we have

$$\begin{aligned}\mathsf{variant}(\mathsf{s}^c_y(\varphi), x, \mathsf{s}^c_y(t)) &= \mathsf{variant}(\mathsf{s}^c_y((Qz)\psi), x, \mathsf{s}^c_y(t)) \\ &= \mathsf{variant}((Qz)\mathsf{s}^c_y(\psi), x, \mathsf{s}^c_y(t)) \\ &= (Qw')(\mathsf{variant}(\mathsf{s}^c_y(\psi), x, \mathsf{s}^c_y(t)))_{z:=w'},\end{aligned}$$

where w' is the first variable in the standard sequence of variables that does not occur in

$$\mathsf{variant}(\mathsf{s}^c_y(\psi), x, \mathsf{s}^c_y(t)) = \mathsf{s}^c_y(\mathsf{variant}(\psi, x, t))$$

or in t. Note that we are again in the third case because x occurs free in $(Qz)\mathsf{s}^c_y(\psi)$ and z occurs in $\mathsf{s}^c_y(t)$.

The argument will be completed if we show that $w' = w$. The variables which precede w in the standard list occur in either $\mathsf{variant}(\psi, x, t)$ (and hence in $\mathsf{s}^c_y(\mathsf{variant}(\psi, x, t))$) or in t. Since $w \neq y$, w does not occur in $\mathsf{s}^c_y(\mathsf{variant}(\psi, x, t))$ or in t. Thus, $w' = w$. □

Corollary 4.6.55. *Let x, y be variables, t be a term, c be a constant symbol, and φ be a formula. If $y \neq x$ and y does not occur in φ or $\mathsf{variant}(\varphi, x, t)$, then*

$$\mathsf{s}^c_y(\langle\varphi\rangle_{x:=t}) = \langle \mathsf{s}^c_y(\varphi)\rangle_{x:=\mathsf{s}^c_y(t)}.$$

Proof. We have

$$\begin{aligned}\mathsf{s}^c_y(\langle\varphi\rangle_{x:=t}) &= \mathsf{s}^c_y((\mathsf{variant}(\varphi, x, t))_{x:=t}) \\ &= (\mathsf{s}^c_y(\mathsf{variant}(\varphi, x, t)))_{x:=\mathsf{s}^c_y(t)} \\ &\quad \text{(by (4.1) since } y \neq x) \\ &= (\mathsf{variant}(\mathsf{s}^c_y(\varphi), x, \mathsf{s}^c_y(t)))_{x:=\mathsf{s}^c_y(t)} \\ &\quad \text{(by Theorem 4.6.54 since } y \neq x \text{ and } y \text{ does} \\ &\quad \text{not occur in } \varphi \text{ or } \mathsf{variant}(\varphi, x, t)) \\ &= \langle \mathsf{s}^c_y(\varphi)\rangle_{x:=\mathsf{s}^c_y(t)}.\end{aligned}$$

□

Our next goal is to provide variants of formulas where bound variables are restricted to a specified set of variables.

Definition 4.6.56. Let θ be a formula and V be an infinite set of variables. The formula $\mathsf{VARIANT}(\theta, V)$ is given by:

- if φ is atomic, then $\mathsf{VARIANT}(\varphi, V) = \varphi$;
- $\mathsf{VARIANT}((\neg\varphi), V) = (\neg\mathsf{VARIANT}(\varphi, V))$;
- for every binary connective symbol C,

$$\mathsf{VARIANT}((\varphi C \psi), V) = (\mathsf{VARIANT}(\varphi, V)\ C\ \mathsf{VARIANT}(\psi, V));$$

- for each quantifier symbol Q and variable x,

$$\begin{aligned}&\mathsf{VARIANT}((Qx)\varphi, V)\\ &= \begin{cases} (Qx)\mathsf{VARIANT}(\varphi, V) & \text{if } x \in V \\ (Qw)(\mathsf{VARIANT}(\varphi, V))_{x:=w} & \text{if } x \notin V \text{ where } w \text{ is the} \\ & \text{first variable in } V \text{ that does} \\ & \text{not occur in } \mathsf{VARIANT}(\varphi, V). \end{cases}\end{aligned}$$

∎

For a signed formula $b\varphi$, we define $\mathsf{VARIANT}(b\varphi, V) = b\mathsf{VARIANT}(\varphi, V)$. If Γ and Δ are sets of unsigned and signed formulas, respectively, the previous notations are further extended by

$$\mathsf{VARIANT}(\Gamma, V) = \bigcup\{\mathsf{VARIANT}(\varphi, V) \mid \varphi \in \Gamma\}$$

$$\mathsf{VARIANT}(\Delta, V) = \bigcup\{\mathsf{VARIANT}(b\varphi, V) \mid b\varphi \in \Delta\}.$$

Theorem 4.6.57. *Let V be an infinite set of variables and let φ be a first-order formula. The formula $\mathsf{VARIANT}(\varphi, V)$ is a variant of φ.*

Proof. The proof is by induction on the formula φ. If φ is atomic, the statement obviously holds. Suppose that $\varphi = (\neg\psi)$ and the statement holds for the formula ψ. Then, by Theorem 4.6.21, the statement holds for the formula φ. The same argument works for the case when $\varphi = (\psi_0 C \psi_1)$.

Let now $\varphi = (Qx)\psi$, where the statement holds for the formula ψ. By Theorem 4.6.21, $(Qx)\mathsf{VARIANT}(\psi, V)$ is a variant of φ. If $x \in V$, then $\mathsf{VARIANT}(\varphi, V) = (Qx)\mathsf{VARIANT}(\psi, V)$, so we are done. If $x \notin V$, then $\mathsf{VARIANT}(\varphi, V) = (Qw)(\mathsf{VARIANT}(\psi, V))_{x:=w}$, which, because w does not occur in $\mathsf{VARIANT}(\psi, V)$, is an immediate variant of $(Qx)\mathsf{VARIANT}(\psi, V)$ and thus a variant of φ. □

Theorem 4.6.58. *Let V be an infinite set of variables and let φ be a first-order formula. If t is a term that is substitutable for a variable z in φ and no variable of t or φ occurs in V, then*

$$\mathit{VARIANT}((\varphi)_{z:=t}, V) = (\mathit{VARIANT}(\varphi, V))_{z:=t}. \tag{4.8}$$

Proof. The argument is by induction on the formula φ. The basis step when φ is atomic is immediate. We discuss here only the inductive step when $\varphi = (Qx)\psi$, where the statement holds for the formula ψ. There are two main cases.

`Case 1:` z does not occur free in φ. Then, z does not occur free in $\mathsf{VARIANT}(\varphi, V)$, because by Theorem 4.6.57, $\mathsf{VARIANT}(\varphi, V)$ is a variant of φ and variants have the same free variables, so the equality follows immediately.

`Case 2:` z occurs free in φ. Then, $z \neq x$, z occurs free in ψ, t is substitutable for z in ψ, and x does not occur in t.

Since x occurs in φ, $x \notin V$ and therefore the left hand side of Equality 4.8, can be written as follows:

$$\begin{aligned}
\mathsf{VARIANT}((\varphi)_{z:=t}, V) &= \mathsf{VARIANT}(((Qx)\psi)_{z:=t}, V)\\
&= \mathsf{VARIANT}((Qx)(\psi)_{z:=t}, V)\\
&\quad \text{(because } z \neq x\text{)}\\
&= (Qw)(\mathsf{VARIANT}((\psi)_{z:=t}, V))_{x:=w},\\
&\quad \text{where } w \text{ is the first variable of } V \text{ that does}\\
&\quad \text{not occur in } \mathsf{VARIANT}((\psi)_{z:=t}, V)\\
&= (Qw)((\mathsf{VARIANT}(\psi, V))_{z:=t})_{x:=w}\\
&\quad \text{(by inductive hypothesis applied to } \psi\text{)}\\
&= (Qw)(\mathsf{VARIANT}(\psi, V))_{z,x:=t,w}.
\end{aligned}$$

Note that the final equality above is justified by Theorem 4.3.86 since x does not occur in t, no variable of V occurs in t and all bound variables of $\mathsf{VARIANT}(\psi, V)$ are in V, so t is substitutable for z in $\mathsf{VARIANT}(\psi, V)$.

The right hand side of the same equality can be transformed as shown next:

$$\begin{aligned}(\mathsf{VARIANT}(\varphi, V))_{z:=t} &= (\mathsf{VARIANT}((Qx)\psi, V))_{z:=t}\\ &= ((Qw')(\mathsf{VARIANT}(\psi, V))_{x:=w'})_{z:=t},\\ &\quad \text{where } w' \text{ is the first variable of } V \text{ that}\\ &\quad \text{does not occur in } \mathsf{VARIANT}(\psi, V);\\ &\quad \text{since } z \text{ occurs in } \varphi, \text{ it follows that } z \notin V;\\ &\quad \text{also, no variable of } t \text{ is in } V, \text{ so } w' = w;\\ &= ((Qw)(\mathsf{VARIANT}(\psi, V))_{x:=w})_{z:=t}\\ &= (Qw)(\mathsf{VARIANT}(\psi, V))_{x,z:=w,t}\\ &\quad \text{by Theorem 4.3.86 since } w \neq z,\end{aligned}$$

which concludes the argument. □

Lemma 4.6.59. *Let φ, φ' be two immediate variants and let V be an infinite set of variables such that no variable of V occurs in either φ or φ'. Then $\mathsf{VARIANT}(\varphi, V) = \mathsf{VARIANT}(\varphi', V)$.*

Proof. The proof is by induction on φ. The basis step, when φ is an atomic formula, is immediate. Among the inductive steps, we discuss only the case when $\varphi = (Qx)\psi$, where Q is a quantifier symbol and the statement holds for ψ.

Since $x \notin V$, we have

$$\mathsf{VARIANT}(\varphi, V) = (Qw)(\mathsf{VARIANT}(\psi, V))_{x:=w},$$

where w is the first variable of V that does not occur in $\mathsf{VARIANT}(\psi, V)$. We distinguish two subcases:

`Case 1:` $\varphi' = (Qx)\psi'$, where ψ' is an immediate variant of ψ. By inductive hypothesis, $\mathsf{VARIANT}(\psi, V) = \mathsf{VARIANT}(\psi', V)$, so

$$\begin{aligned}\mathsf{VARIANT}(\varphi, V) &= (Qw)(\mathsf{VARIANT}(\psi', V))_{x:=w},\\ &\quad \text{where } w \text{ is the first variable of } V \text{ that}\\ &\quad \text{does not occur in } \mathsf{VARIANT}(\psi', V)\end{aligned}$$

$$\begin{aligned}&= \mathsf{VARIANT}((Qx)\psi', V)\\ &= \mathsf{VARIANT}(\varphi', V).\end{aligned}$$

`Case 2:` $\varphi' = (Qz)(\psi)_{x:=z}$, where z is substitutable for x in ψ and z does not occur free in ψ. We may assume that $x \neq z$ since if $x = z$, then $\varphi' = \varphi$ and the result is immediate. Observe that $z \notin V$. We have

$$\mathsf{VARIANT}(\varphi', V) = (Qw')(\mathsf{VARIANT}((\psi)_{x:=z}, V))_{z:=w'},$$

where w' is the first variable of V that does not occur in

$$\mathsf{VARIANT}((\psi)_{x:=z}, V) = (\mathsf{VARIANT}(\psi, V))_{x:=z};$$

the last equality follows from Theorem 4.6.58. Since neither x nor z occurs in V, w' is the first variable in V that does not occur in $\mathsf{VARIANT}(\psi, V)$ and therefore, $w' = w$. Thus,

$$\begin{aligned}\mathsf{VARIANT}(\varphi', V) &= (Qw)((\mathsf{VARIANT}(\psi, V))_{x:=z})_{z:=w}\\ &= (Qw)(\mathsf{VARIANT}(\psi, V))_{x,z:=w,w}\\ &\quad \text{(by Theorem 4.3.86 because } z\\ &\quad \text{is substitutable}\\ &\quad \text{for } x \text{ in } \mathsf{VARIANT}(\psi, V))\\ &= (Qw)(\mathsf{VARIANT}(\psi, V))_{x:=w}\\ &\quad \text{(since } z \text{ is not free in } \psi \text{ and}\\ &\quad \text{therefore in } \mathsf{VARIANT}(\psi, V))\\ &= \mathsf{VARIANT}(\varphi, V).\end{aligned}$$

□

Theorem 4.6.60. *Let φ, φ' be two variants and let V be an infinite set of variables such that $VAR - V$ is also infinite and no variable of V occurs in either φ or φ'. Then $\mathsf{VARIANT}(\varphi, V) = \mathsf{VARIANT}(\varphi', V)$.*

Proof. By Theorem 4.6.38 applied to the set VAR $- V$, there is a sequence of immediate variants $\varphi_0, \ldots, \varphi_{n-1}$ such that $\varphi = \varphi_0$, $\varphi_{n-1} = \varphi'$ and no variable of φ_i occurs in V. The result follows by repeated application of Lemma 4.6.59. □

The following theorem is a technical result that will be useful in Section 5.3.

Theorem 4.6.61. *Let φ, φ' be two formulas such that φ' is a variant of φ and let t be a term. Let V be a set of variables such that both V and its complement are infinite, and no variable in φ, φ' or t occurs in V. Suppose that t is substitutable for x in φ' and that w is the first variable in V that does not occur in $\mathsf{VARIANT}(\varphi, V)$. If $\varphi'' = (\mathsf{VARIANT}(\varphi, V))_{x:=w}$, then t is substitutable for w in φ'' and $(\varphi'')_{w:=t} = (\mathsf{VARIANT}(\varphi', V))_{x:=t}$.*

Proof. Since $\mathtt{BV}(\varphi'') \subseteq V$, it is clear that t is substitutable for w in φ'' and if $x \neq w$, we have:

$$\begin{aligned}
(\varphi'')_{w:=t} &= ((\mathsf{VARIANT}(\varphi, V))_{x:=w})_{w:=t} \\
&= (\mathsf{VARIANT}(\varphi, V))_{x,w:=t,t} \\
&\quad \text{(by Theorem 4.3.86 because } w \text{ is} \\
&\quad \text{ substitutable for } x \text{ in } \mathsf{VARIANT}(\varphi, V)) \\
&= (\mathsf{VARIANT}(\varphi, V))_{x:=t} \\
&\quad \text{(because } w \text{ does not occur in } \mathsf{VARIANT}(\varphi, V)) \\
&= (\mathsf{VARIANT}(\varphi', V))_{x:=t} \\
&\quad \text{(by Theorem 4.6.60).}
\end{aligned}$$

Note that if $x = w$, we can omit from the above chain the second step and get the same result. □

Lemma 4.6.62. *Let Δ_0 be a set of signed formulas and let U be an infinite set of variables such that its complement is also infinite and $\mathtt{V}(\Delta_0) \cap U = \emptyset$. Then, no two distinct formulas in $\mathsf{VARIANT}(\Delta_0, U)$ are variants of each other.*

Proof. Let φ, ψ be two formulas in $\mathsf{VARIANT}(\Delta_0, U)$ such that ψ is a variant of φ. Then, for some formulas $\varphi', \psi' \in \Delta_0$, $\varphi = \mathsf{VARIANT}(\varphi', U)$ and $\psi = \mathsf{VARIANT}(\psi', U)$. Thus, by symmetry and transitivity of variants, ψ' is a variant of φ' and, since $\mathtt{V}(\Delta_0) \cap U = \emptyset$, we have $\varphi = \psi$, by Theorem 4.6.60. □

Lemma 4.6.63. *Let Δ'_0 be a set of signed formulas such that no two distinct formulas in Δ'_0 are variants of each other, Δ_0 be a set of* *signed formulas and let $f : \Delta'_0 \longrightarrow \Delta_0$ be a function such that every formula $b'\varphi' \in \Delta'_0$ is a variant of $f(b'\varphi')$. Then, f is an injection.*

Proof. Suppose that $f(b'_0\varphi'_0) = f(b'_1\varphi'_1)$, where $b'_0\varphi'_0, b'_1\varphi'_1 \in \Delta'_0$. By symmetry and transitivity of variants, $b'_0\varphi'_0$ and $b'_1\varphi'_1$ are variants of each other and therefore must be equal. □

Theorem 4.6.64. *Let Δ'_0 and Δ_0 be sets of signed formulas such that every formula in Δ'_0 is a variant of a formula in Δ_0 and $\mathtt{V}(\Delta'_0) \cap U = \emptyset$, where U is an infinite set of variables such that its complement is also infinite. Then, there is an injection $f : \mathsf{VARIANT}(\Delta'_0, U) \longrightarrow \Delta_0$ such that for all $b''\varphi'' \in \mathsf{VARIANT}(\Delta'_0, U)$, $f(b''\varphi'')$ is a variant of $b''\varphi''$.*

Proof. Let $b''\varphi''$ be an arbitrary signed formula in $\mathsf{VARIANT}(\Delta'_0, U)$, say $b''\varphi'' = \mathsf{VARIANT}(b'\varphi', U)$, where $b'\varphi'$ is the first signed formula in Δ'_0 which satisfies this condition. Next, let $b\varphi$ be the first formula in Δ_0 such that $b'\varphi'$ is a variant of $b\varphi$ and define $f(b''\varphi'') = b\varphi$. By transitivity of variants, $b''\varphi''$ is a variant of $b\varphi$. By Lemma 4.6.62, no two distinct formulas in $\mathsf{VARIANT}(\Delta'_0, U)$ are variants of each other, so by Lemma 4.6.63, f is an injection. □

4.7 Definability in Structures

The purpose of this section is to investigate the possibility of defining relations over domains of structures by using first-order formulas.

Let $\mathcal{A}$ be an $\mathcal{L}$-structure, φ be an $\mathcal{L}$-formula, $y_0, \ldots, y_{n-1}$ be n distinct variables such that $\mathtt{FV}(\varphi) \subseteq \{y_0, \ldots, y_{n-1}\}$, and $a_0, \ldots, a_{n-1}$ be n elements of $|\mathcal{A}|$. We write $(\mathcal{A}, [y_0 \to a_0, \ldots, y_{n-1} \to a_{n-1}]) \models \varphi$ if $(\mathcal{A}, \sigma) \models \varphi$ for some $\sigma \in \text{ASSIGN}_{\mathcal{A}}$ such that $\sigma(y_i) = a_i$ for $0 \le i \le n-1$. By the Agreement Theorem, $(\mathcal{A}, [y_0 \to a_0, \ldots, y_{n-1} \to a_{n-1}]) \models \varphi$ is equivalent to saying that $(\mathcal{A}, \sigma) \models \varphi$ for every $\sigma \in \text{ASSIGN}_{\mathcal{A}}$ such that $\sigma(y_i) = a_i$ for $0 \le i \le n-1$. For the special case $n = 0$, φ is a closed formula and $(\mathcal{A}, [y_0 \to a_0, \ldots, y_{n-1} \to a_{n-1}]) \models \varphi$ if and only if $\mathcal{A} \models \varphi$.

Definition 4.7.1. Let $\mathcal{A}$ be an $\mathcal{L}$-structure, and φ be an $\mathcal{L}$-formula such that $\mathtt{FV}(\varphi) \subseteq \{y_0, \ldots, y_{n-1}\}$, where $y_0, \ldots, y_{n-1}$ are n distinct variables. An n-ary relation $\rho \subseteq |\mathcal{A}|^n$ is *definable* by φ and the sequence $(y_0, \ldots, y_{n-1})$ if

$$\rho = \{(a_0, \ldots, a_{n-1}) \mid (\mathcal{A}, [y_0 \to a_0, \ldots, y_{n-1} \to a_{n-1}]) \models \varphi\}.$$

An n-ary relation on $|\mathcal{A}|$ is *first-order definable in* $\mathcal{A}$ if it is definable by some first-order formula and some sequence of variables of length n.

A function $f : |\mathcal{A}|^n \longrightarrow |\mathcal{A}|$ is *definable by a formula* ψ *and by the sequence* $(y_0, \ldots, y_{n-1}, y_n)$ if the $n+1$-ary relation

$$\texttt{graph}(f) = \{(a_0, \ldots, a_{n-1}, a_n) \in |\mathcal{A}|^{n+1} \mid a_n = f(a_0, \ldots, a_{n-1})\}$$

is definable in $\mathcal{A}$ by ψ and $(y_0, \ldots, y_{n-1}, y_n)$. □

Thus, a 0-ary function $f : |\mathcal{A}|^0 \longrightarrow |\mathcal{A}|$ is definable in $\mathcal{A}$ if and only if there is a formula ψ with one free variable y_0 such that

$$f(\lambda) = \{a_0 \in |\mathcal{A}| \mid (\mathcal{A}, [y_0 \to a_0]) \models \psi\}.$$

Example 4.7.2. Let $\mathcal{L} = \{+, \cdot, =\}$ be a first-order language which is a subset of $\mathcal{L}_{ar}$. Parentheses will be omitted when this can cause no confusion. Consider the $\mathcal{L}$-structure $\mathcal{A}$ which is the reduct of $\mathcal{A}_{ar}$ to $\mathcal{L}$.

The set $\{0\}$ is definable in $\mathcal{A}$ by the formula $x = x + x$ (and the sequence of variables (x)) because 0 is the only natural number n such that $n = n + n$.

Similarly, the set $\{1\}$ is definable in $\mathcal{A}$ by the formula $((x = x \cdot x) \wedge (\neg(x = x + x)))$.

The set E of even natural numbers is definable in $\mathcal{A}$ by the formula $(\exists y)(x = y + y)$.

The relation $\leq$ on $\mathbf{N}$ is definable in $\mathcal{A}$ by the formula $(\exists z)(x + z = y)$ and the sequence (x, y). The same formula with the sequence (y, x) defines the relation $\geq$.

Let $\mathcal{B}$ be the $\mathcal{L}$-structure with $|\mathcal{B}| = \mathbf{Z}$ and the standard interpretations of $+$ and $\cdot$. It is known from number theory that every nonnegative integer can be expressed as the sum of four squares. Therefore, the subset $\mathbf{N}$ of $\mathbf{Z}$ is definable in $\mathcal{B}$ by the formula

$$(\exists y_0)(\exists y_1)(\exists y_2)(\exists y_3)(x = y_0 \cdot y_0 + y_1 \cdot y_1 + y_2 \cdot y_2 + y_3 \cdot y_3).$$

The function $f(n) = 2n$ for $n \in \mathbf{Z}$ is definable in the $\mathcal{L}$-structure $\mathcal{B}$ by the formula $(x + x = y)$ and the sequence (x, y). □

Example 4.7.3. In Example 4.4.3, we introduced the first-order language $\mathcal{L} = \{R, =\}$ in order to associate $\mathcal{L}$-structures with directed graphs. We examine here a few definable sets of vertices in these structures.

Let G be a directed graph. The set of vertices of G of out-degree 1 is definable in the structure $\mathcal{A}_G$ by the formula

$$\varphi_{out} = ((\exists y)R(x,y) \wedge (\forall y)(\forall z)((R(x,y) \wedge R(x,z)) \to y = z)).$$

Note that the formula φ_{out} is the same for every directed graph G.

Similarly, the set of vertices of in-degree 1 of any directed graph G is definable in the corresponding structure $\mathcal{A}_G$ by the formula

$$\varphi_{in} = ((\exists y)R(y,x) \wedge (\forall y)(\forall z)((R(y,x) \wedge R(z,x)) \to y = z)).$$

We refer to a vertex s of a directed graph as a *source* if its in-degree is 0 and for every vertex $v \neq s$ of G, there is an edge (s, v) in the graph. It is clear that a directed graph contains at most one source. The formula

$$((\neg(\exists y)R(y,x)) \wedge (\forall z)((z \neq x) \to R(x,z)))$$

defines the source of a graph G, if one exists, in the structure $\mathcal{A}_G$.

The set of pairs of vertices joined by a path of length 2 in G is definable in $\mathcal{A}_G$ by the formula

$$(\exists z)(R(x,z) \wedge R(z,y))$$

and the sequence (x, y). For each $k \geq 3$, a similar formula and sequence can be used to define the set of pairs of vertices joined by a path of length k. $\square$

Next, we introduce several notations which allow us to show the definability of relations important in number theory.

For $m, n, p \in \mathbf{N}$, we write $m \equiv n \pmod{p}$ if p divides $m - n$. Also, we write $m = n \mod p$ if $m \equiv n \pmod{p}$ and $n < p$, in other words if n is the remainder of the division of m by p. We will also use the equivalence $\equiv_p$ on $\mathbf{N}$ given by

$$\equiv_p = \{(m,n) \in \mathbf{N}^2 \mid m \equiv n \pmod{p}\}$$

for $p \in \mathbf{N}$. Note that $\equiv_0$ is the identity relation on $\mathbf{N}$, $\equiv_1$ is $\mathbf{N}^2$, there are no m, n with $m = n \mod 0$, and $m = n \mod 1$ if and only if $n = 0$.

Example 4.7.4. Let μ be the ternary relation on $\mathbf{N}$ given by

$$\mu = \{(m, n, p) \in \mathbf{N}^3 \mid m \equiv n \pmod{p}\}.$$

The relation μ is definable in $\mathcal{A}_{ar}$ by the formula $\varphi_\mu = (\exists x_0)(((x_0 \cdot x_3 + x_2) = x_1) \vee ((x_0 \cdot x_3 + x_1) = x_2))$ and the sequence (x_1, x_2, x_3).

If δ is the related ternary relation on $\mathbf{N}$ defined by

$$\delta = \{(m, n, p) \in \mathbf{N}^3 \mid m = n \mod p\},$$

then δ is definable by the formula $\varphi_{x,y,z,\delta} = (\exists \dot{x})((x = \dot{x} \cdot z + y) \wedge (y < z))$ and the sequence (x, y, z) in the same structure $\mathcal{A}_{ar}$. $\square$

The next lemma contains a result known as the *Chinese Remainder Theorem*[5].

Lemma 4.7.5 (Chinese Remainder Theorem). *Let* $n_0, \ldots, n_{k-1}$ *be* k *natural numbers that are pairwise relatively prime,* $a_0, \ldots, a_{k-1}$ *be* k *natural numbers such that* $0 \leq a_i < n_i$ *for* $0 \leq i \leq k-1$, *and let* $n = n_0 \cdots n_{k-1}$, *where* $k \geq 1$. *There is a unique* $q \in \mathbf{N}$ *such that* $0 \leq q < n$ *and* $q = a_i \mod n_i$ *for* $0 \leq i \leq k-1$.

Proof. Let $\mathbf{N}_{n_0,\ldots,n_{k-1}}$ be the set

$$\{0, \ldots, n_0 - 1\} \times \cdots \times \{0, \ldots, n_{k-1} - 1\}.$$

Define the function

$$F : \{p \in \mathbf{N} \mid 0 \leq p < n\} \longrightarrow \mathbf{N}_{n_0,\ldots,n_{k-1}},$$

by $F(p) = (p_0, \ldots, p_{k-1})$ if $p_i = p \mod n_i$.

We claim that F is injective, that is, $F(p) = F(p')$ implies $p = p'$. Indeed, note that $F(p) = F(p')$ implies that $p' \equiv p \pmod{n_i}$, so

[5]The earliest version of this result occurs in the Chinese mathematical works *Sun Tzu Suan Ching* written in the late 3rd century by Sun Zi. Little is known about the life of Sun Zi, also known as Master Su; it is believed that he was a Buddhist scholar.

n_i divides $p - p'$ for $0 \leq i \leq k-1$. Since $n_0, \ldots, n_{k-1}$ are pairwise relatively prime it follows that n divides $p - p'$. However, since $|p - p'| < n$, this is possible only if $p = p'$.

Note that $|\mathbf{N}_{n_0,\ldots,n_{k-1}}| = n$. Since the domain of F contains n values and F is injective, if follows that F is a bijection. Thus, for every $(a_0, \ldots, a_{k-1}) \in \mathbf{N}_{n_0,\ldots,n_{k-1}}$, there exists a unique q, $0 \leq q < n$, such that $F(q) = (a_0, \ldots, a_{k-1})$, that is, $q = a_i \mod n_i$ for $0 \leq i \leq k-1$. $\square$

Another useful fact is contained in the next lemma.

Lemma 4.7.6. *Let $n \in \mathbf{N}$ be a natural number. The $n+1$ numbers of the form $n! \cdot (m+1) + 1$, where $0 \leq m \leq n$ are pairwise relatively prime.*

Proof. Let $p = n! \cdot (j+1) + 1$ and $q = n! \cdot (k+1) + 1$ be two distinct numbers, where $0 \leq j, k \leq n$ and let d be a prime common divisor of p and q. It is clear that d cannot divide $n!$, so it cannot be a divisor for any non-zero natural number less than or equal to n. On another hand, d must divide the difference $p - q$, so d must divide $n!(j-k)$. Since d does not divide $n!$, it must divide $|j-k|$ which is a positive number not larger than n. This contradiction shows that p, q are relatively prime. $\square$

Theorem 4.7.7. *Let $B \subseteq \mathbf{N}^4$ be the relation such that $(c, d, i, r) \in B$ if and only if $c = r \mod ((i+1)d + 1)$. Then, for every sequence $(a_0, \ldots, a_{k-1}) \in \mathrm{Seq}(\mathbf{N})$ there are $c, d \in \mathbf{N}$ such that $(c, d, i, a_i) \in B$ for $0 \leq i \leq k-1$.*

Proof. The theorem is trivial if $k = 0$ because any numbers c, d satisfy the conclusion, so assume that $k \geq 1$. Let $\ell = \max\{k, a_0, \ldots, a_{k-1}\}$. Define $d = \ell!$. By Lemma 4.7.6, the numbers $n_m = \ell! \cdot (m+1) + 1 = d(m+1) + 1$ are pairwise relatively prime for $0 \leq m \leq \ell$. Note that $a_m \leq \ell < n_m$ for $0 \leq m \leq k$.

By the Chinese Remainder Theorem (Lemma 4.7.5), there exists a natural number c such that $c = a_i \mod n_i$ for $0 \leq i \leq k-1$, that is, $c = a_i \mod ((i+1)d + 1)$ for $0 \leq i \leq k-1$. Thus, $(c, d, i, a_i) \in B$ for $0 \leq i \leq k-1$. $\square$

The relation B of Theorem 4.7.7 will be referred to as *the Gödel relation.*[6] The purpose of the Gödel relation is to encode sequences of natural number as pairs of natural numbers (c, d). Indeed, the relation B has a functional character in that for every c, d, i, there is a unique r such that $(c, d, i, r) \in B$. We will denote this r by $\mathsf{b}(c, d, i)$. Thus, we obtain the *Gödel function* $\mathsf{b} : \mathbf{N}^3 \longrightarrow \mathbf{N}$.

We can regard a pair (c, d) as encoding the infinite sequence $(a_0, a_1, \ldots)$, where for each i, a_i is the unique number with $(c, d, i, a_i) \in B$. Theorem 4.7.7 asserts that for every finite sequence of natural numbers, there is a pair (c, d) such that the infinite sequence encoded by this pair begins with the finite sequence.

Example 4.7.8. The Gödel relation B is definable in $\mathcal{A}_{ar}$ by the formula

$$\varphi_{x_1,x_2,v,w,B} = (\varphi_{x,y,z,\delta})_{x,y,z:=x_1,w,(v+1)\cdot x_2+1}$$

and the sequence of variables (x_1, x_2, v, w), where $\varphi_{x,y,z,\delta}$ was defined in Example 4.7.4.

This also shows that the Gödel function b is definable in $\mathcal{A}_{ar}$ by the above formula $\varphi_{x_1,x_2,v,w,B}$ and the sequence of variables (x_1, x_2, v, w). □

Example 4.7.9. We prove now that the exponential function $\exp : \mathbf{N}^2 \longrightarrow \mathbf{N}$ is definable in $\mathcal{A}_{ar}$ by showing that its graph $\texttt{graph}(\exp) = \{(m, n, p) \in \mathbf{N}^3 \mid p = m^n\}$ is definable in this structure. The formula that defines $\texttt{graph}(\exp)$ states that there is a sequence of length $n + 1$ of the form $(1, m, m^2, \ldots, m^n)$ such that the last entry of the

[6]Kurt Friedrich Gödel was born on April 28, 1906 in Brno, currently in the Czech Republic. He died on January 14, 1978 in Princeton, New Jersey. Gödel studied at the University of Vienna, receiving his doctorate in 1930 and presented his *Habilitationsschrift* in 1932. He taught at the University of Vienna until 1940 and then, for the rest of his life, at the Institute for Advanced Study in Princeton. Gödel is considered to have been the greatest logician of the twentieth century and made fundamental contributions to the development of logic such as the completeness theorem of first-order logic, the incompleteness theorem, and consistency results in set theory.

sequence equals p. It is not difficult to see that this formula is

$$\begin{aligned}&(\exists y_0)(\exists y_1)((\varphi_{x_1,x_2,v,w,B})_{x_1,x_2,v,w:=y_0,y_1,0,s(0)}\\ &\qquad\wedge(\forall y_2)(\exists y_3)(\exists y_4)((y_2 < y) \to ((\varphi_{x_1,x_2,v,w,B})_{x_1,x_2,v,w:=y_0,y_1,y_2,y_3}\\ &\qquad\wedge(\varphi_{x_1,x_2,v,w,B})_{x_1,x_2,v,w:=y_0,y_1,s(y_2),y_4} \wedge (y_4 = y_3 \cdot x))\\ &\qquad\wedge(\varphi_{x_1,x_2,v,w,B})_{x_1,x_2,v,w:=y_0,y_1,y,z})\end{aligned}$$

where $\varphi_{x_1,x_2,v,w,B}$ is the formula from Example 4.7.8. The set $\texttt{graph}(\exp)$ is defined by the above formula and the sequence (x, y, z). $\Box$

Next, we investigate closure properties of definable relations which are important for describing the retrieval capabilities of relational database systems.

Theorem 4.7.10. *For every $\mathcal{L}$-structure $\mathcal{A}$, the set $|\mathcal{A}|^n$ is definable for every $n \in \mathbf{N}$.*

Proof. Let φ be any logically valid, closed $\mathcal{L}$-formula. Then, $|\mathcal{A}|^n$ is definable by φ and the sequence $(x_0, \dots, x_{n-1})$. $\Box$

Lemma 4.7.11. *Let ρ be an n-ary relation definable in an $\mathcal{L}$-structure $\mathcal{A}$ by the formula φ and the sequence $(y_0, \dots, y_{n-1})$. If $y'_0, \dots, y'_{n-1}$ are n distinct variables that do not occur bound in φ, then ρ is also definable in $\mathcal{A}$ by the formula $\varphi' = (\varphi)_{y_0,\dots,y_{n-1}:=y'_0,\dots,y'_{n-1}}$ and the sequence $(y'_0, \dots, y'_{n-1})$.*

Proof. Note that $y'_0, \dots, y'_{n-1}$ are substitutable for $y_0, \dots, y_{n-1}$, respectively, in φ because $y'_0, \dots, y'_{n-1}$ do not occur bound in φ. Therefore, by Theorem 4.3.83, we have

$$\texttt{FV}(\varphi') \subseteq (\texttt{FV}(\varphi) - \{y_0, \dots, y_{n-1}\}) \cup \{y'_0, \dots, y'_{n-1}\} = \{y'_0, \dots, y'_{n-1}\}.$$

Let $a_0, \dots, a_{n-1}$ be n elements of $|\mathcal{A}|$ and $\sigma' \in \mathrm{ASSIGN}_{\mathcal{A}}$ be such that $\sigma'(y'_i) = a_i$ for $0 \le i \le n-1$. Define $\sigma = [y_0 \to {\sigma'}^{\mathcal{A}}(y'_0)] \cdots [y_{n-1} \to {\sigma'}^{\mathcal{A}}(y'_{n-1})]\sigma'$. Then, $\sigma(y_i) = a_i$ for $0 \le i \le n-1$. The following statements are equivalent.

(1) $(\mathcal{A}, [y'_0 \to a_0, \dots, y'_{n-1} \to a_{n-1}]) \models \varphi'$;
(2) $(\mathcal{A}, \sigma') \models \varphi'$;
(3) $(\mathcal{A}, [y_0 \to {\sigma'}^{\mathcal{A}}(y'_0)] \cdots [y_{n-1} \to {\sigma'}^{\mathcal{A}}(y'_{n-1})]\sigma') \models \varphi$;

(4) $(\mathcal{A}, \sigma) \models \varphi$;
(5) $(\mathcal{A}, [y_0 \to a_0, \ldots, y_{n-1} \to a_{n-1}]) \models \varphi$;
(6) $(a_0, \ldots, a_{n-1}) \in \rho$.

(The equivalence of the second and third statements follows from the Substitution Corollary.) Thus, ρ is definable in $\mathcal{A}$ by φ' and the sequence $(y'_0, \ldots, y'_{n-1})$. $\square$

Theorem 4.7.12. *Let ρ_0, ρ_1 be n-ary relations definable in the $\mathcal{L}$-structure $\mathcal{A}$. The relations $\rho_0 \cup \rho_1, \rho_0 \cap \rho_1$ and $\rho_0 - \rho_1$ are definable in $\mathcal{A}$.*

Proof. Suppose that ρ_0 is definable in $\mathcal{A}$ by φ_0 and $(y_0, \ldots, y_{n-1})$ and ρ_1 is definable in $\mathcal{A}$ by φ_1 and $(z_0, \ldots, z_{n-1})$. Let $y'_0, \ldots, y'_{n-1}$ be n distinct variables that do not occur in either φ_0 or φ_1. Let

$$\varphi'_0 = (\varphi_0)_{y_0,\ldots,y_{n-1}:=y'_0,\ldots,y'_{n-1}}$$
$$\varphi'_1 = (\varphi_1)_{z_0,\ldots,z_{n-1}:=y'_0,\ldots,y'_{n-1}}.$$

By Lemma 4.7.11, ρ_0 is definable in $\mathcal{A}$ by φ'_0 and $(y'_0, \ldots, y'_{n-1})$ and ρ_1 is definable in $\mathcal{A}$ by φ'_1 and the same sequence of variables.

Thus, $\rho_0 \cup \rho_1$ is definable in $\mathcal{A}$ by $(\varphi'_0 \vee \varphi'_1)$, $\rho_0 \cap \rho_1$ is definable by $(\varphi'_0 \wedge \varphi'_1)$, and $\rho_0 - \rho_1$ is definable by $(\varphi'_0 \wedge (\neg\varphi'_1))$, all with the sequence $(y'_0, \ldots, y'_{n-1})$. $\square$

Corollary 4.7.13. *If ρ is an n-ary relation definable in the structure $\mathcal{A}$, then its complement $|\mathcal{A}|^n - \rho$ is also definable in $\mathcal{A}$.*

Proof. This follows immediately from Theorems 4.7.10 and 4.7.12. $\square$

Definition 4.7.14. Let $\rho \subseteq A_0 \times \cdots \times A_{n-1}$ be an n-ary relation and let $(i_0, \ldots, i_{k-1})$ be a sequence of distinct elements of $\{0, \ldots, n-1\}$. Suppose that $\{0, \ldots, n-1\} - \{i_0, \ldots, i_{k-1}\} = \{j_0, \ldots, j_{l-1}\}$, where $j_0 < \cdots < j_{l-1}$. The *projection of ρ on* $(i_0, \ldots, i_{k-1})$ is the relation

$$\rho[i_0, \ldots, i_{k-1}]$$
$$= \{(a_{i_0}, \ldots, a_{i_{k-1}}) \mid \text{ for some } a_{j_0}, \ldots, a_{j_{l-1}}, (a_0, \ldots, a_{n-1}) \in \rho\}.$$

$\square$

Theorem 4.7.15. *If ρ is an n-ary relation definable in the structure $\mathcal{A}$, then any projection $\rho[i_0, \ldots, i_{k-1}]$ of ρ is also definable in $\mathcal{A}$.*

Proof. Suppose that ρ is definable in $\mathcal{A}$ by the formula φ and the sequence $(y_0, \ldots, y_{n-1})$. Let $j_0, \ldots, j_{l-1}$ be as in Definition 4.7.14. Then, $\rho[i_0, \ldots, i_{k-1}]$ is definable in $\mathcal{A}$ by the formula $(\exists y_{j_0}) \cdots (\exists y_{j_{l-1}})\varphi$ and the sequence $(y_{i_0}, \ldots, y_{i_{k-1}})$. □

Definition 4.7.16. Let $\rho \subseteq A_0 \times \cdots \times A_{n-1}$ be an n-ary relation and let $\rho' \subseteq A'_0 \times \cdots \times A'_{m-1}$ be an m-ary relation. The *product of ρ and ρ'* is the relation

$$\rho \times \rho' = \{(a_0, \ldots, a_{n-1}, a'_0, \ldots, a'_{m-1}) \mid (a_0, \ldots, a_{n-1}) \in \rho \text{ and } (a'_0, \ldots, a'_{m-1}) \in \rho'\}.$$

□

Theorem 4.7.17. *If ρ and ρ' are definable relations in $\mathcal{A}$ of arities n and m, respectively, then $\rho \times \rho'$ is definable in $\mathcal{A}$.*

Proof. Let ρ be definable in $\mathcal{A}$ by φ and the sequence $(y_0, \ldots, y_{n-1})$ and ρ' be definable by φ' and the sequence $(y'_0, \ldots, y'_{m-1})$. By Lemma 4.7.11, we may assume without loss of generality that the sets $\{y_0, \ldots, y_{n-1}\}$ and $\{y'_0, \ldots, y'_{m-1}\}$ are disjoint. Then, the formula $(\varphi \wedge \varphi')$ and the sequence $(y_0, \ldots, y_{n-1}, y'_0, \ldots, y'_{m-1})$ define $\rho \times \rho'$. □

Definition 4.7.18. Let $\rho \subseteq A_0 \times \cdots \times A_{n-1}$ be an n-ary relation, $\mathbf{i} = (i_0, \ldots, i_{k-1})$ be a sequence of distinct elements of $\{0, \ldots, n-1\}$, and $\rho' \subseteq A_{i_0} \times \cdots \times A_{i_{k-1}}$. The *$(\rho', \mathbf{i})$-selection of ρ* is the relation

$$\mathrm{sel}_{\rho', \mathbf{i}}(\rho) = \{(a_0, \ldots, a_{n-1}) \in \rho \mid (a_{i_0}, \ldots, a_{i_{k-1}}) \in \rho'\}.$$

□

Theorem 4.7.19. *Suppose that ρ is an n-ary relation definable in a structure $\mathcal{A}$, ρ' is a k-ary relation definable in $\mathcal{A}$ with $k \leq n$, and $\mathbf{i}$ is a sequence $(i_0, \ldots, i_{k-1})$ of k distinct elements of $\{0, \ldots, n-1\}$. Then, the $(\rho', \mathbf{i})$-selection of ρ is definable in $\mathcal{A}$.*

Proof. Let ρ be definable in $\mathcal{A}$ by φ and $(y_0, \ldots, y_{n-1})$ and let ρ' be definable in the same structure by φ' and $(y'_0, \ldots, y'_{k-1})$. By applying Lemma 4.7.11, we may assume that none of $y_0, \ldots, y_{n-1}$ appears bound in φ'. By a second application of the lemma, we may assume that $y'_j = y_{i_j}$ for $0 \leq j \leq k-1$. Then, it is easy to see that $\mathrm{sel}_{\rho',\mathbf{i}}(\rho)$ is definable in $\mathcal{A}$ by the formula $\varphi \wedge \varphi'$ and the sequence $(y_0, \ldots, y_{n-1})$. $\square$

In Section 4.10, we show some limitations on the ability of first-order logic to define relations that are of practical value in Computer Science (see Theorem 4.10.47).

4.8 Propositional Forms and Tautologies

We now discuss a technique for transforming formulas of propositional logic into first-order logic formulas.

Definition 4.8.1. An *inter-substitution* is a substitution that maps the set of statement variables SV into FORM. $\square$

In the usual way, an inter-substitution s can be extended to PLFORM; we will denote this extension by $\overline{s}$ and, as usual for substitutions, we will drop the bar whenever there is no risk of confusion. It is easy to prove by induction on formulas in PLFORM that if the range of an inter-substitution s is included in $\mathrm{FORM}_{\mathcal{L}}$, then $s(\varphi) \in \mathrm{FORM}_{\mathcal{L}}$ for every $\varphi \in$ PLFORM.

For the remainder of this section, we will refer to inter-substitutions simply as substitutions, when there is no risk of confusion. If a substitution has its values in $\mathcal{F}$, where $\mathcal{F}$ is a class of formulas, then we will refer to it as an $\mathcal{F}$-substitution. In particular, if $\mathcal{F}$ is the class of prime formulas (atomic formulas), we will use the terms *prime substitution* (*atomic substitution*). Also, when $\mathcal{F} = \mathrm{FORM}_{\mathcal{L}}$, we will use the term *$\mathcal{L}$-inter-substitution* or just *$\mathcal{L}$-substitution* when the context is unambiguous.

Theorem 4.8.2. *Let z be a propositional substitution and let s be an inter-substitution. Then,*

$$\overline{s}\,\overline{z} = \overline{\overline{s}z}.$$

Proof. We need to verify that

$$\overline{s}\,\overline{z}(\alpha) = \overline{\overline{s}z}(\alpha),$$

for all $\alpha \in$ PLFORM. The argument is by induction on α and is left to the reader. □

Definition 4.8.3. Let Γ be a set of formulas, $\Gamma \subseteq$ FORM. A set of formulas Γ_0 of propositional logic is a *propositional form* for Γ if there is a substitution s such that $s(\Gamma_0) = \Gamma$.

Γ_0 is a *fundamental propositional form* for Γ if $s(\Gamma_0) = \Gamma$ where s is a prime, injective substitution.

A formula $\alpha \in$ PLFORM is a *propositional form* for a formula $\varphi \in$ FORM and φ is a *substitution instance* of α if $\{\alpha\}$ is a propositional form for the set $\{\varphi\}$. If $\{\alpha\}$ is a fundamental propositional form for $\{\varphi\}$, then we refer to α as a *fundamental propositional form* for φ.

A *tautology of first-order logic* is a formula that is a substitution instance of a tautology of propositional logic. □

Theorem 4.8.4. *Let φ be a formula and let R, R' be relation symbols of the same arity such that R' does not occur in φ. Then, φ is a tautology if and only if $\mathsf{s}^{R}_{R'}(\varphi)$ is a tautology.*

Proof. Note that by Theorem 4.3.49, $\mathsf{s}^{R}_{R'}(\varphi)$ is a formula. Suppose that φ is a tautology. Then, there is a tautology of propositional logic α and an inter-substitution s such that $\varphi = s(\alpha)$. Again, by Theorem 4.3.49, $\mathsf{s}^{R}_{R'}(s(p))$ is a formula for every $p \in SV$, so we may define an inter-substitution s' by $s'(p) = \mathsf{s}^{R}_{R'}(s(p))$ for every $p \in SV$. An argument by induction on formulas of propositional logic shows that $s'(\beta) = \mathsf{s}^{R}_{R'}(s(\beta))$ for every $\beta \in$ PLFORM. In particular, $s'(\alpha) = \mathsf{s}^{R}_{R'}(s(\alpha)) = \mathsf{s}^{R}_{R'}(\varphi)$, so $\mathsf{s}^{R}_{R'}(\varphi)$ is a tautology. The inverse implication follows from what we have just shown and the fact that $\mathsf{s}^{R'}_{R}(\mathsf{s}^{R}_{R'}(\varphi)) = \varphi$ as established in Theorem 1.2.16. □

When we say that an inter-substitution s is injective in Definition 4.8.3, we mean injectivity of s as a function on SV. In general, the extension of an injective inter-substitution to PLFORM need not be injective. However, we have the following result.

Theorem 4.8.5. *If s is a prime, injective substitution, then its extension to* PLFORM *is also an injective function.*

Proof. Let s be a prime, injective substitution. We prove by induction on α that $s(\alpha) = s(\beta)$ implies $\alpha = \beta$. For the basis step, suppose that α is a statement variable p. Then, $s(\beta) = s(p)$ is a prime formula and this implies that β is a statement variable q. Thus, $\alpha = \beta$ due to the injectivity of s on SV.

Let now $\alpha = (\alpha_0 C \alpha_1)$, where C is a binary connective symbol. We have $s(\beta) = (s(\alpha_0) C s(\alpha_1))$. Note that β cannot be a statement variable because $s(\beta)$ is not prime. Thus, $\beta = (\beta_0 C \beta_1)$ and $s(\alpha_i) = s(\beta_i)$ for $i = 0, 1$. By inductive hypothesis, $\alpha_0 = \beta_0$ and $\alpha_1 = \beta_1$, so $\alpha = \beta$.

A similar argument works when $\alpha = (\neg \alpha_0)$. □

Example 4.8.6. Let P, R be two unary relation symbols and let a be a constant symbol. It is easy to see that each of the formulas p_0, $(p_0 \to p_1)$, $(p_0 \to (p_1 \to p_2))$ and $(p_0 \to (p_1 \to p_0))$ are propositional forms for the formula $\varphi = (P(a) \to (R(a) \to P(a)))$. Of these, only the last is a fundamental propositional form for φ because we can write $\varphi = s((p_0 \to (p_1 \to p_0)))$, where s, given by $s(p_0) = P(a), s(p_1) = R(a)$, and $s(p_i) = R(x_i)$ for $i \geq 2$, is a prime, injective substitution and no such substitution exists for the other propositional forms. □

Theorem 4.8.7. *Let Γ be a set of formulas. There is a fundamental propositional form Γ_0 for Γ.*

Proof. Let $\psi_0, \psi_1, \ldots$ be an enumeration of all prime formulas of first-order logic. Define the substitution $s(p_i) = \psi_i$, for $i \in \mathbf{N}$. Clearly, s is a prime, injective substitution.

We claim that for every formula φ of first-order logic, there is a formula α of propositional logic such that $s(\alpha) = \varphi$. The argument is by induction on φ and is left to the reader. Thus, for every $\varphi \in \Gamma$, there is a formula $\alpha_\varphi \in$ PLFORM such that $s(\alpha_\varphi) = \varphi$. This allows us to define the set Γ_0 as $\{\alpha_\varphi \mid \varphi \in \Gamma\}$. It is clear that Γ_0 is a fundamental propositional form for Γ. □

Lemma 4.8.8. *If s, s' are two inter-substitutions and α is a formula in* PLFORM *such that $\overline{s}(\alpha) = \overline{s'}(\alpha)$, then $s(p) = s'(p)$ for every $p \in SV(\alpha)$.*

Proof. Suppose that there is some variable $q \in SV(\alpha)$ such that $s(q) \neq s'(q)$ and let (p, i) be the first occurrence of such a variable in α. Then, we can write $\alpha = \epsilon_0 p \epsilon_1$, where $|\epsilon_0| = i$ and $\overline{s}(\epsilon_0) = \overline{s'}(\epsilon_0) = \epsilon$ for some sequence ϵ, because $\overline{s}(a) = \overline{s'}(a)$ when $a \in \{(,), \neg, \vee, \wedge, \rightarrow, \leftrightarrow\}$. Thus, $\overline{s}(\alpha) = \epsilon s(p)\overline{s}(\epsilon_1)$ and $\overline{s'}(\alpha) = \epsilon s'(p)\overline{s'}(\epsilon_1)$. Since $\overline{s}(\alpha) = \overline{s'}(\alpha)$, this means that $s(p)$ is a proper prefix of $s'(p)$ or vice-versa. By Lemma 4.3.18, this is impossible. □

Theorem 4.8.9. *Let φ be a formula. If α is a propositional form for φ and β is a fundamental propositional form for φ, then there is a propositional substitution z such that $\overline{z}(\alpha) = \beta$.*

Proof. The argument is by induction on α. For the basis step, let α be a statement variable p. Then, the conclusion is immediate since any propositional substitution z such that $z(p) = \beta$ will suffice.

Suppose now that the statement holds for α_0 and α_1 and that $\alpha = (\alpha_0 C \alpha_1)$, where C is a binary connective symbol. Assume further that $\varphi = \overline{s}(\alpha) = \overline{s'}(\beta)$, where s' is a prime, injective substitution and s is a substitution. Since $\overline{s'}(\beta) = \overline{s}(\alpha) = (\overline{s}(\alpha_0) C \overline{s}(\alpha_1))$ and s' is a prime substitution, it follows that $\beta = (\beta_0 C \beta_1)$ for some $\beta_0, \beta_1 \in$ PLFORM. Indeed, β cannot be a statement variable because of the primeness of s' and we can exclude the cases $\beta = (\neg\gamma)$ and $\beta = (\gamma_0 C' \gamma_1)$ with $C' \neq C$ because of unique readability. Therefore, $\varphi = \overline{s'}(\beta) = (\overline{s'}(\beta_0) C \overline{s'}(\beta_1)) = (\overline{s}(\alpha_0) C \overline{s}(\alpha_1))$ which implies that $\overline{s}(\alpha_0) = \overline{s'}(\beta_0) = \varphi_0$ and $\overline{s}(\alpha_1) = \overline{s'}(\beta_1) = \varphi_1$ for some formulas φ_0, φ_1, again, by unique readability. This means that β_i is a fundamental propositional form for φ_i, and α_i is a propositional form for φ_i for $i = 0, 1$. By the inductive hypothesis, there are propositional substitutions z_0, z_1 such that $\overline{z_i}(\alpha_i) = \beta_i$ for $i = 0, 1$. It follows from this that

$$\overline{s}(\alpha_i) = \overline{s'}(\beta_i) = \overline{s'}\,\overline{z_i}(\alpha_i) = \overline{\overline{s'} z_i}(\alpha_i)$$

for $i = 0, 1$, by Theorem 4.8.2. By Lemma 4.8.8, we have $\overline{s'} z_i(p) = s(p)$ for all $p \in SV(\alpha_i)$ for $i = 0, 1$, which implies $\overline{s'} z_0(p) = \overline{s'} z_1(p)$ for every $p \in SV(\alpha_0) \cap SV(\alpha_1)$. Theorem 4.8.5 which asserts the injectivity of $\overline{s'}$ implies $z_0(p) = z_1(p)$ for every $p \in SV(\alpha_0) \cap SV(\alpha_1)$. This shows the existence of a propositional substitution z such that $z(p) = z_0(p)$ for every $p \in SV(\alpha_0)$ and $z(p) = z_1(p)$ for every $p \in SV(\alpha_1)$. Thus, $\overline{z}(\alpha_0) = \beta_0$ and $\overline{z}(\alpha_1) = \beta_1$, so $\overline{z}(\alpha) = \beta$, as desired.

We leave to the reader the simpler case when $\alpha = (\neg\gamma)$. □

Corollary 4.8.10. *If β is a fundamental propositional form for the formula φ, then φ is a tautology (of first-order logic) if and only if β is a tautology (of propositional logic).*

Proof. If β is a tautology of propositional logic, it is obvious that φ is a tautology of first-order logic.

Conversely, let φ be a tautology of first-order logic. Then, there is a propositional form α for φ such that α is a tautology. By Theorem 4.8.9, there is a propositional substitution z such that $\beta = \overline{z}(\alpha)$. By Corollary 2.6.8, β is a tautology. $\square$

Corollary 4.8.10 allows us to give an effective procedure for determining whether a given formula φ is a tautology of first-order logic. First, using for example the algorithm implicit in the proof of Theorem 4.8.7, we find a fundamental propositional form α for φ. Then, using one of the methods discussed in Chapter 2, we determine whether α is a tautology of propositional logic.

Lemma 4.8.11. *Let $\mathcal{L}$ be a first-order language, s be an $\mathcal{L}$-substitution, $\mathcal{A}$ be an $\mathcal{L}$-structure and σ be an assignment in* $\mathrm{ASSIGN}_{\mathcal{A}}$. *Define a truth assignment v by:*

$$v(p) = \begin{cases} \mathbf{T} & \text{if } (\mathcal{A}, \sigma) \models s(p) \\ \mathbf{F} & \text{otherwise.} \end{cases}$$

Then, for every formula $\varphi \in \mathrm{PLFORM}$, we have $v(\varphi) = \mathbf{T}$ if and only if $(\mathcal{A}, \sigma) \models s(\varphi)$.

Proof. The argument is by induction on φ and is left to the reader. $\square$

Theorem 4.8.12. *Let Γ be a set of $\mathcal{L}$-formulas, where $\mathcal{L}$ is a first-order language and let Γ_0 be a propositional form for Γ. If Γ is satisfiable, then Γ_0 is satisfiable.*

Proof. Suppose that $\Gamma = s(\Gamma_0)$, where s is an inter-substitution. We may assume that s is an $\mathcal{L}$-substitution by redefining s on the variables that do not occur in the formulas of Γ_0, if necessary. Since Γ is satisfiable, there is an $\mathcal{L}$-structure $\mathcal{A}$ and $\sigma \in \mathrm{ASSIGN}_{\mathcal{A}}$ such that $(\mathcal{A}, \sigma) \models \varphi$ for every $\varphi \in \Gamma$. Define v as in Lemma 4.8.11. Then, by the same Lemma, v satisfies Γ_0. $\square$

The converse of Theorem 4.8.12 is not true in general as we show in the next example. Nevertheless, as we shall see in Section 4.10, it holds for sets of quantifier-free formulas without equality and their fundamental propositional forms.

Example 4.8.13. Let $\Gamma_0 = \{p, (\neg q)\}$, where p, q are distinct statement variables. Then, Γ_0 is a propositional form for both $\Gamma = \{R(x), (\neg R(x))\}$ and $\Gamma' = \{(\forall x)R(x), (\neg R(y))\}$. Note that Γ_0 is satisfiable, but neither Γ nor Γ' is. In the case of Γ, Γ_0 is not a fundamental propositional form, while in the case of Γ', Γ_0 is a fundamental propositional form, but the formulas of Γ' contain quantifiers. ▯

Theorem 4.8.14. *Every tautology of first-order logic is logically valid.*

Proof. Let α be a tautology of propositional logic that is a propositional form for φ. Then, $(\neg\alpha)$ is a propositional form for $(\neg\varphi)$ and, since $(\neg\alpha)$ is unsatisfiable, $(\neg\varphi)$ is unsatisfiable by Theorem 4.8.12. Thus, φ is logically valid. □

The converse of Theorem 4.8.14 is not true in general, as shown below in Example 4.8.15; however, it holds for quantifier-free formulas without equality. This will be shown in Section 4.10.

Example 4.8.15. Let $\psi = ((\forall x)\varphi \to (\exists x)\varphi)$, where φ is an arbitrary formula. We saw in Example 4.5.19 that ψ is logically valid. It is easy to see that ψ has only two types of propositional forms: p and $(p \to q)$, where $p \neq q$ and $p, q \in SV$. Since neither of these forms is a tautology, ψ is not a tautology. ▯

Corollary 4.8.16. *If φ, ψ are two logically equivalent formulas of propositional logic and s is an inter-substitution, then $s(\varphi)$ and $s(\psi)$ are logically equivalent first-order formulas.*

Proof. Since $\varphi \equiv \psi$, the formula $(\varphi \leftrightarrow \psi)$ is a tautology by Theorem 2.3.20, Part (4). Therefore, by Theorem 4.8.14, $s(\varphi \leftrightarrow \psi) = (s(\varphi) \leftrightarrow s(\psi))$ is logically valid. So, by Theorem 4.5.55, Part (4), we have $s(\varphi) \equiv s(\psi)$. □

Theorem 4.8.17. *Let φ, ψ and θ be formulas. Then, we have:*

$$
\begin{array}{rll}
(\varphi \wedge \varphi) \equiv \varphi & & \textit{(idempotency of } \wedge\textit{)} \\
(\varphi \vee \varphi) \equiv \varphi & & \textit{(idempotency of } \vee\textit{)} \\
(\varphi \wedge \psi) \equiv (\psi \wedge \varphi) & & \textit{(commutativity of } \wedge\textit{)} \\
(\varphi \vee \psi) \equiv (\psi \vee \varphi) & & \textit{(commutativity of } \vee\textit{)} \\
(\varphi \wedge (\psi \wedge \theta)) \equiv ((\varphi \wedge \psi) \wedge \theta) & & \textit{(associativity of } \wedge\textit{)} \\
(\varphi \vee (\psi \vee \theta)) \equiv ((\varphi \vee \psi) \vee \theta) & & \textit{(associativity of } \vee\textit{)} \\
(\neg(\neg\varphi)) \equiv \varphi & & \textit{(double negation)} \\
(\varphi \wedge (\varphi \vee \psi)) \equiv \varphi & & \textit{(absorption laws)} \\
(\varphi \vee (\varphi \wedge \psi)) \equiv \varphi & & \\
(\varphi \wedge (\psi \vee \theta)) \equiv ((\varphi \wedge \psi) \vee (\varphi \wedge \theta)) & & \textit{(distributivity laws)} \\
(\varphi \vee (\psi \wedge \theta)) \equiv ((\varphi \vee \psi) \wedge (\varphi \vee \theta)) & & \\
(\neg(\varphi \vee \psi)) \equiv ((\neg\varphi) \wedge (\neg\psi)) & & \textit{(DeMorgan's laws)} \\
(\neg(\varphi \wedge \psi)) \equiv ((\neg\varphi) \vee (\neg\psi)) & & \\
(\varphi \rightarrow \psi) \equiv ((\neg\varphi) \vee \psi) & & \\
(\varphi \leftrightarrow \psi) \equiv ((\varphi \rightarrow \psi) \wedge (\psi \rightarrow \varphi)) & &
\end{array}
$$

Proof. Let φ, ψ, θ be three formulas and let s be the inter-substitution defined by $s(p_0) = \varphi$, $s(p_1) = \psi$, $s(p_2) = \theta$ and $s(p_i) = p_i$ for $i > 2$. Applying this substitution to the pairs of equivalent formulas from Theorem 2.3.30 and taking into account Corollary 4.8.16, we obtain immediately the equivalent formulas mentioned in this theorem. □

Corollary 4.8.18. *Let φ be a formula and x be a variable. Then, we have*:

$$(\neg(\exists x)\varphi) \equiv (\forall x)(\neg\varphi)$$

$$(\neg(\forall x)\varphi) \equiv (\exists x)(\neg\varphi)$$

Proof. Applying Part (2) of Theorem 4.5.57 to the formula $(\neg\varphi)$, we obtain

$$(\forall x)(\neg\varphi) \equiv (\neg(\exists x)(\neg(\neg\varphi))).$$

Taking into account the double negation equivalence of Theorem 4.8.17 and the Replacement Theorem, we obtain the desired result.

The second part of the corollary has a similar proof. □

4.9 Normal Forms for Formulas

As in Section 2.5, we present normal forms for formulas of first-order logic.

We begin with a normal form for quantifier-free formulas.

Definition 4.9.1. A quantifier-free formula φ is in *disjunctive normal form* if it is a disjunction of conjunctions of literals. Each conjunction is called a *disjunct* of the formula.

A quantifier-free formula φ is in *conjunctive normal form* if it is a conjunction of disjunctions of literals. Each disjunction is called a *conjunct* of the formula.

A conjunctive normal form formula φ is a *Horn formula* if each conjunct contains at most one positive literal.

A formula φ is *clausal* if it is a disjunction of literals. ▯

If $\varphi = (\ell_0 \vee \cdots \ell_{n-1})$ is a clausal formula, then φ is in both conjunctive normal form and disjunctive normal form because φ can be regarded as a single conjunct of literals and also as a disjunction of one-literal conjuncts.

Theorem 4.9.2. *Let $\mathcal{L}$ be a first-order language. For every quantifier-free $\mathcal{L}$-formula φ, there is a logically equivalent $\mathcal{L}$-formula in disjunctive normal form (conjunctive normal form) with the same set of variables as φ.*

Proof. Let φ be a quantifier-free $\mathcal{L}$-formula and let α be a fundamental propositional form for φ such that $s(\alpha) = \varphi$ where s is a prime, injective $\mathcal{L}$-substitution. Because φ is quantifier-free, we may assume that s is atomic. Let α' be a disjunctive normal form (conjunctive normal form) for α. By Corollary 2.5.12, we may assume that $SV(\alpha') = SV(\alpha)$. By Corollary 4.8.16, the $\mathcal{L}$-formulas $\varphi = s(\alpha)$ and $s(\alpha')$ are logically equivalent, so $s(\alpha')$ is a disjunctive (conjunctive) normal form for φ. Since α' and α have the same set of statement variables, $s(\alpha')$ has the same set of variables as $s(\alpha) = \varphi$. □

Note that the $\mathcal{L}$-substitution s and the formula α can be effectively found from φ and the normal form α' can be effectively constructed as we observed previously. Therefore, the proof of Theorem 4.9.2

gives an effective procedure for obtaining a disjunctive (conjunctive) normal form of a first-order quantifier-free formula.

Example 4.9.3. Let $\mathcal{L} = \{P, Q, R\}$ be a first-order language, where P, Q are binary relation symbols and R is a ternary relation symbol and let φ be the quantifier-free formula

$$((P(x, y) \to Q(y, z)) \land R(x, y, z)).$$

The formula $\alpha = ((p_4 \to p_0) \land p_6)$ is a fundamental propositional form for φ with an atomic substitution s such that $s(p_0) = Q(y, z)$, $s(p_4) = P(x, y)$, and $s(p_6) = R(x, y, z)$. As noted in Example 2.5.13, the truth-table given in Example 2.5.11 is the truth-table of α, so by Example 2.5.16, the formula $\alpha' = (((\neg p_4) \land p_6) \lor (p_0 \land p_4 \land p_6))$ is a disjunctive normal form for α. Therefore, the first-order formula $s(\alpha')$ given by

$$(((\neg P(x, y)) \land R(x, y, z)) \lor (Q(y, z) \land P(x, y) \land R(x, y, z)))$$

is a disjunctive normal form for φ. □

Definition 4.9.4. A formula φ is in *prenex normal form* if

$$\varphi = (Q_0 y_0) \cdots (Q_{n-1} y_{n-1})\psi,$$

where $n \geq 0$, $Q_0, \ldots, Q_{n-1}$ are quantifier symbols, $y_0, \ldots, y_{n-1}$ are distinct variables, and ψ is a quantifier-free formula.

We will refer to the formula ψ as the *matrix* of φ. □

Observe that if a variable y_i does not occur in the matrix ψ of φ then, if we drop $(Q_i y_i)$, the resulting formula is still in prenex normal form and is logically equivalent to φ because of Corollary 4.5.43 and the Replacement Theorem (Theorem 4.6.16).

Example 4.9.5. The formula $\varphi = (\forall x)(\forall z)(\exists y)(R(x, y) \land R(y, z))$ is in prenex normal form and has $(R(x, y) \land R(y, z))$ as matrix. □

Intuitively, a formula $(\forall y_0)\psi$ is no easier to understand than a formula $(\forall y_0) \cdots (\forall y_{m-1})\psi$. However, it is apparent that the formulas $\alpha = (\forall y_0)(\exists y_1)\psi$ and $\beta = (\exists y_0)(\forall y_1)\psi$ have a higher complexity. This complexity is introduced by the alternation of the quantifiers. Informally, a prenex formula is in Π_n-form if it begins with an occurrence of the universal quantifier symbol and contains $n-1$ alternations of quantifier symbols. For example, α is in Π_2-form. Similarly, a prenex formula is in Σ_n-form if it begins with an occurrence of the existential quantifier symbol and contains $n-1$ alternations of quantifier symbols. The formula β above is in Σ_2-form.

We formalize this intuition by introducing the classes Π_n and Σ_n by a simultaneous inductive definition.

Definition 4.9.6. The classes Π_n and Σ_n of formulas are defined recursively as follows for $n \in \mathbf{N}$:

- The classes Π_0 and Σ_0 both consist of the quantifier-free formulas.
- For $n \geq 0$, Π_{n+1} consists of all formulas of the form

$$(\forall y_0) \cdots (\forall y_{m-1})\varphi$$

  where $\varphi \in \Sigma_n$, $m \geq 1$, and $y_0, \ldots, y_{m-1}$ are distinct variables that do not belong to $\mathtt{BV}(\varphi)$.
- For $n \geq 0$, Σ_{n+1} consists of all formulas of the form

$$(\exists y_0) \cdots (\exists y_{m-1})\varphi$$

  where $\varphi \in \Pi_n$, $m \geq 1$, and $y_0, \ldots, y_{m-1}$ are distinct variables that do not belong to $\mathtt{BV}(\varphi)$.

We say that a formula is in Π_n-form (Σ_n-form) if it belongs to the class Π_n (Σ_n, respectively). $\square$

It is clear that every formula in Π_n or Σ_n is in prenex normal form. Conversely (see Exercise 114), one can prove that every formula in prenex normal form belongs to a class Π_n or Σ_n. It is clear that $\Pi_0 \cup \Pi_1$ is the set of universal formulas; similarly, $\Sigma_0 \cup \Sigma_1$ is the set of existential formulas.

Our goal is to show that for every formula φ, there is a logically equivalent formula in prenex normal form. To this end we need to consider several types of pairs of logically equivalent formulas. Some of the results shown below have been already discussed.

Lemma 4.9.7. *Let φ, ψ be formulas and let x, y be variables. Then, we have the following logical equivalences*:

(1) $(\neg(\forall x)\varphi) \equiv (\exists x)(\neg\varphi)$ *and* $(\neg(\exists x)\varphi) \equiv (\forall x)(\neg\varphi)$;

(2) $((\exists x)\varphi \vee \psi) \equiv (\exists x)(\varphi \vee \psi)$ *and* $((\forall x)\varphi \vee \psi) \equiv (\forall x)(\varphi \vee \psi)$, *where* $x \notin \mathtt{FV}(\psi)$;
(3) $(\varphi \vee (\exists x)\psi) \equiv (\exists x)(\varphi \vee \psi)$ *and* $(\varphi \vee (\forall x)\psi) \equiv (\forall x)(\varphi \vee \psi)$, *where* $x \notin \mathtt{FV}(\varphi)$;
(4) $((\exists x)\varphi \wedge \psi) \equiv (\exists x)(\varphi \wedge \psi)$ *and* $((\forall x)\varphi \wedge \psi) \equiv (\forall x)(\varphi \wedge \psi)$, *where* $x \notin \mathtt{FV}(\psi)$*;*
(5) $(\varphi \wedge (\exists x)\psi) \equiv (\exists x)(\varphi \wedge \psi)$ *and* $(\varphi \wedge (\forall x)\psi) \equiv (\forall x)(\varphi \wedge \psi)$, *where* $x \notin \mathtt{FV}(\varphi)$;
(6) $((\exists x)\varphi \rightarrow \psi) \equiv (\forall x)(\varphi \rightarrow \psi)$ *and* $((\forall x)\varphi \rightarrow \psi) \equiv (\exists x)(\varphi \rightarrow \psi)$, *where* $x \notin \mathtt{FV}(\psi)$*;*
(7) $(\varphi \rightarrow (\exists x)\psi) \equiv (\exists x)(\varphi \rightarrow \psi)$ *and* $(\varphi \rightarrow (\forall x)\psi) \equiv (\forall x)(\varphi \rightarrow \psi)$, *where* $x \notin \mathtt{FV}(\varphi)$*.*

Also, in each of the above logical equivalences, the formulas involved have the same free variables and the same extra-logical symbols.

Proof. Part (1) was shown in Corollary 4.8.18. We proceed now with the argument for Part (2).

Suppose that φ, ψ are $\mathcal{L}$-formulas. If $(\mathcal{A}, \sigma) \models ((\exists x)\varphi \vee \psi)$, where $\mathcal{A}$ is an $\mathcal{L}$-structure and $\sigma \in \text{ASSIGN}_{\mathcal{A}}$, the following two cases may occur:

`Case 1:` $(\mathcal{A}, \sigma) \models (\exists x)\varphi$. Then, there is $a \in |\mathcal{A}|$ such that $(\mathcal{A}, [x \rightarrow a]\sigma) \models \varphi$ which implies $(\mathcal{A}, [x \rightarrow a]\sigma) \models (\varphi \vee \psi)$. Therefore, $(\mathcal{A}, \sigma) \models (\exists x)(\varphi \vee \psi)$.
`Case 2:` $(\mathcal{A}, \sigma) \models \psi$. Then, $(\mathcal{A}, \sigma) \models (\varphi \vee \psi)$. By Theorem 4.5.40, we obtain $(\mathcal{A}, \sigma) \models (\exists x)(\varphi \vee \psi)$.
Conversely, assume that $(\mathcal{A}, \sigma) \models (\exists x)(\varphi \vee \psi)$. Then, there is $a \in |\mathcal{A}|$ such that $(\mathcal{A}, [x \rightarrow a]\sigma) \models (\varphi \vee \psi)$. Again, we distinguish two cases.
`Case 1:` $(\mathcal{A}, [x \rightarrow a]\sigma) \models \varphi$. Then, $(\mathcal{A}, \sigma) \models (\exists x)\varphi$ and this implies $(\mathcal{A}, \sigma) \models ((\exists x)\varphi \vee \psi)$.
`Case 2:` $(\mathcal{A}, [x \rightarrow a]\sigma) \models \psi$. Since $x \notin \mathtt{FV}(\psi)$, σ and $[x \rightarrow a]\sigma$ agree on all free variables of ψ. By the Agreement Theorem, we have $(\mathcal{A}, \sigma) \models \psi$, which gives $(\mathcal{A}, \sigma) \models ((\exists x)\varphi \vee \psi)$.

The proof of the second logical equivalence of Part (2) is similar and is left to the reader.

For the first equivalence of Part (3), observe that

$$\begin{aligned}(\varphi \vee (\exists x)\psi) &\equiv ((\exists x)\psi \vee \varphi) && \text{(commutativity of } \vee\text{)}\\ &\equiv (\exists x)(\psi \vee \varphi) && \text{(Part (2))}\\ &\equiv (\exists x)(\varphi \vee \psi) && \text{(commutativity of } \vee\\ & && \text{and Replacement Theorem)}\end{aligned}$$

The second equivalence of this part is shown similarly.

The first logical equivalence of Part (4) is a consequence of the following logical equivalences.

$$\begin{aligned}((\exists x)\varphi \wedge \psi) &\equiv (\neg((\neg(\exists x)\varphi) \vee (\neg\psi))) && \text{(DeMorgan's Law)}\\ &\equiv (\neg((\forall x)(\neg\varphi) \vee (\neg\psi))) && \text{(Part (1) and}\\ & && \text{Replacement Theorem)}\\ &\equiv (\neg(\forall x)((\neg\varphi) \vee (\neg\psi))) && \text{(Part (2) and}\\ & && \text{Replacement Theorem)}\\ &\equiv (\exists x)(\neg((\neg\varphi) \vee (\neg\psi))) && \text{(Part (1))}\\ &\equiv (\exists x)(\varphi \wedge \psi) && \text{(DeMorgan's Law and}\\ & && \text{Replacement Theorem)}\end{aligned}$$

The remaining logical equivalences of Parts (4) and (5) are left to the reader.

Parts (6) and (7) can be proven in a way similar to the one used for Parts (4) and (5), using the next to last logical equivalence of Theorem 4.8.17 in place of DeMorgan's Laws. (We gave an independent proof of the first logical equivalence of Part (7) in Example 4.5.44.)

We leave to the reader the verification that the formulas involved in each of the previous logical equivalences have the same set of free variables. □

The notation introduced next allows us to write certain parts of the previous lemma in a more concise way. Let Q be a quantifier symbol. Then, $\overline{Q}$ is given by

$$\overline{Q} = \begin{cases} \forall \text{ if } Q = \exists \\ \exists \text{ if } Q = \forall. \end{cases}$$

Now, both equivalences of Part (1) become $(\neg(Qx)\varphi) \equiv (\overline{Q}x)(\neg\varphi)$. Also, Part (6) becomes $((Qx)\varphi \to \psi) \equiv (\overline{Q}x)(\varphi \to \psi)$ if $x \notin \texttt{FV}(\psi)$.

Definition 4.9.8. Let φ be a formula. A *prenex normal form for* φ is a formula ψ in prenex normal form such that $\mathcal{L}_\varphi = \mathcal{L}_\psi$, $\texttt{FV}(\varphi) = \texttt{FV}(\psi)$ and $\varphi \equiv \psi$. □

We now show that every formula has a prenex normal form.

Theorem 4.9.9. *There is a prenex normal form for every formula.*

Proof. The argument is by induction on formulas. The basis is immediate since an atomic formula is already in prenex normal form.

Let $\psi = (Q_0 y_0) \cdots (Q_{n-1} y_{n-1})\theta$, where θ is a quantifier-free formula, be a formula in prenex normal form that is logically equivalent to φ with $\texttt{FV}(\psi) = \texttt{FV}(\varphi)$, and $\mathcal{L}_\psi = \mathcal{L}_\varphi$. Then,

$$(\neg\varphi) \equiv (\neg\psi) \equiv (\overline{Q_0} y_0) \cdots (\overline{Q_{n-1}} y_{n-1})(\neg\theta)$$

by repeated application of the Replacement Theorem and of Part (1) of Lemma 4.9.7. It is clear that the last formula is in prenex normal form, has the same set of free variables and the same set of extralogical symbols as $(\neg\varphi)$.

We consider now the case of the formula $(\varphi_0 C \varphi_1)$ where C is a binary connective symbol. By the inductive hypothesis, we assume the existence of formulas

$$\psi_0 = (Q_{0,0} y_0) \cdots (Q_{0,n-1} y_{n-1})\theta_0$$
$$\psi_1 = (Q_{1,0} z_0) \cdots (Q_{1,m-1} z_{m-1})\theta_1,$$

in prenex normal form such that $\psi_i \equiv \varphi_i$ for $i = 0, 1$, where θ_0, θ_1 are quantifier-free formulas. Also, by the same hypothesis, $\texttt{FV}(\psi_0) = \texttt{FV}(\varphi_0)$, $\texttt{FV}(\psi_1) = \texttt{FV}(\varphi_1)$, $\mathcal{L}_{\psi_0} = \mathcal{L}_{\varphi_0}$, and $\mathcal{L}_{\psi_1} = \mathcal{L}_{\varphi_1}$.

Let $y'_0, \ldots, y'_{n-1}, z'_0, \ldots, z'_{m-1}$ be $n + m$ distinct variables that do not occur in ψ_0 or ψ_1. By Theorem 4.6.18, we have the logical equivalences $\psi_0 \equiv \psi'_0$ and $\psi_1 \equiv \psi'_1$, where

$$\psi'_0 = (Q_{0,0} y'_0) \cdots (Q_{0,n-1} y'_{n-1})\theta'_0$$
$$\psi'_1 = (Q_{1,0} z'_0) \cdots (Q_{1,m-1} z'_{m-1})\theta'_1,$$

and θ_0', θ_1' are the quantifier-free formulas:

$$\theta_0' = (\theta_0)_{y_{n-1}:=y_{n-1}',\dots,y_0:=y_0'}$$
$$\theta_1' = (\theta_1)_{z_{m-1}:=z_{m-1}',\dots,z_0:=z_0'}.$$

Also, by Theorem 4.6.18, we have $\mathtt{FV}(\psi_0') = \mathtt{FV}(\psi_0)$ and $\mathtt{FV}(\psi_1') = \mathtt{FV}(\psi_1)$, so $\mathtt{FV}((\psi_0' C \psi_1')) = \mathtt{FV}((\varphi_0 C \varphi_1))$.

If $C \in \{\vee, \wedge\}$, then, by Parts (2)–(5) of Lemma 4.9.7 we have the following logical equivalences:

$$\begin{aligned}&(\varphi_0 C \varphi_1) \equiv (\psi_0 C \psi_1) \equiv (\psi_0' C \psi_1')\\ &\quad \equiv (Q_{0,0} y_0') \cdots (Q_{0,n-1} y_{n-1}')(Q_{1,0} z_0') \cdots (Q_{1,m-1} z_{m-1}')(\theta_0' C \theta_1').\end{aligned}$$

Clearly, the last formula is in prenex normal form and, by Lemma 4.9.7, it has the same free variables and extra-logical symbols as $(\varphi_0 C \varphi_1)$.

For $C = \rightarrow$, we have:

$$\begin{aligned}&(\varphi_0 \rightarrow \varphi_1) \equiv (\psi_0 \rightarrow \psi_1) \equiv (\psi_0' \rightarrow \psi_1')\\ &\quad \equiv (\overline{Q}_{0,0} y_0') \cdots (\overline{Q}_{0,n-1} y_{n-1}')(Q_{1,0} z_0') \cdots (Q_{1,m-1} z_{m-1}')(\theta_0' \rightarrow \theta_1').\end{aligned}$$

When $C = \leftrightarrow$, we have, by Theorem 4.8.17,

$$\begin{aligned}(\varphi_0 \leftrightarrow \varphi_1) &\equiv ((\varphi_0 \rightarrow \varphi_1) \wedge (\varphi_1 \rightarrow \varphi_0))\\ &\equiv ((\psi_0 \rightarrow \psi_1) \wedge (\psi_1 \rightarrow \psi_0)).\end{aligned}$$

Using the method discussed earlier in the proof, we construct prenex normal form formulas α_0, α_1 for $(\psi_0 \rightarrow \psi_1)$ and $(\psi_1 \rightarrow \psi_0)$, respectively such that $\mathtt{FV}(\alpha_0) = \mathtt{FV}((\psi_0 \rightarrow \psi_1))$ and $\mathtt{FV}(\alpha_1) = \mathtt{FV}((\psi_1 \rightarrow \psi_0))$. Thus, $(\varphi_0 \leftrightarrow \varphi_1) \equiv (\alpha_0 \wedge \alpha_1)$ and $\mathtt{FV}((\varphi_0 \leftrightarrow \varphi_1)) = \mathtt{FV}((\alpha_0 \wedge \alpha_1))$. Finally, again using the method discussed earlier, we obtain a prenex normal form for $(\alpha_0 \wedge \alpha_1)$ which is a prenex normal form for $(\varphi_0 \leftrightarrow \varphi_1)$ and has the same free variables and extra-logical symbols as this formula.

Suppose that ψ is a prenex normal form for φ. If $\alpha = (Qx)\varphi$ where Q is a quantifier symbol and x is a variable, we distinguish two cases. If $x \in \mathtt{FV}(\varphi)$, then $x \in \mathtt{FV}(\psi)$ so the formula $\beta = (Qx)\psi$ is in prenex normal form, $\beta \equiv \alpha$, $\mathtt{FV}(\beta) = \mathtt{FV}(\alpha)$, and $\mathcal{L}_\beta = \mathcal{L}_\alpha$. If $x \notin \mathtt{FV}(\varphi)$, then, by Corollary 4.5.43, $\alpha \equiv \varphi \equiv \psi$, $\mathtt{FV}(\alpha) = \mathtt{FV}(\psi)$ and $\mathcal{L}_\alpha = \mathcal{L}_\psi$, so ψ is the desired prenex normal form formula. $\square$

The argument of Theorem 4.9.9 contains an implicit algorithm for converting a formula into a logically equivalent one in prenex normal form.

For the sake of simplicity of presentation, we did not use the most efficient renaming of bound variables in the proof of Theorem 4.9.9. As the examples below will show, it is possible to obtain a prenex normal form for a formula with less renaming of variables.

Example 4.9.10. Let $\varphi = (((\exists y)R(x,y) \wedge P(y,z)) \to (\exists u)R(x,u))$.

If we apply the algorithm implicitly contained in the proof of Theorem 4.9.9 to $(\exists y)R(x,y)$, $P(y,z)$, and $\psi_1 = (\exists u)R(x,u)$, the same formulas are returned.

Consider now the formula $\varphi_0 = ((\exists y)R(x,y) \wedge P(y,z))$. Following the algorithm, we rename y to a new variable y' to obtain the equivalent formula $((\exists y')R(x,y') \wedge P(y,z))$. Now the quantifier symbol can be moved to the front, thereby yielding the equivalent formula $\psi_0 = (\exists y')(R(x,y') \wedge P(y,z))$ in prenex normal form.

Applying the algorithm to φ requires us to deal with the formulas

$$\psi_0 = (\exists y')(R(x,y') \wedge P(y,z))$$
$$\psi_1 = (\exists u)R(x,u).$$

Let y'', u' be two new variables. The algorithm applied to φ yields the equivalent formula $(\forall y'')(\exists u')((R(x,y'') \wedge P(y,z)) \to R(x,u'))$.

Observe that the renaming of y' and u was not really necessary. A direct application of Lemma 4.9.7 gives another prenex normal form for φ: $(\forall y')(\exists u)((R(x,y') \wedge P(y,z)) \to R(x,u))$.

We could have applied a different sequence of transformations yielding $(\exists u)(\forall y')((R(x,y') \wedge P(y,z)) \to R(x,u))$ as an alternate prenex normal form for φ. $\square$

As demonstrated in the previous example, the algorithm contained in the proof of Theorem 4.9.9 is only one possible way to obtain a prenex normal form for a given formula. In general, the operations of Lemma 4.9.7, together with renaming of bound variables, can be used in many ways to produce different prenex normal form formulas logically equivalent to a given formula.

Example 4.9.11. Let $\varphi = ((\exists x)R(x) \leftrightarrow (\exists x)P(x))$. Since

$$\varphi \equiv (((\exists x)R(x) \to (\exists x)P(x)) \wedge ((\exists x)P(x) \to (\exists x)R(x))),$$

we need to find a prenex normal form for the latter formula. By renaming bound variables and applying the appropriate transformations given in Lemma 4.9.7, we get the following sequence of logically equivalent formulas:

$$\begin{gathered}(((\exists x)R(x) \to (\exists x)P(x)) \wedge ((\exists x)P(x) \to (\exists x)R(x)))\\(((\exists x)R(x) \to (\exists y)P(y)) \wedge ((\exists z)P(z) \to (\exists w)R(w)))\\((\forall x)(\exists y)(R(x) \to P(y)) \wedge (\forall z)(\exists w)(P(z) \to R(w)))\\(\forall x)(\exists y)(\forall z)(\exists w)((R(x) \to P(y)) \wedge (P(z) \to R(w)))\end{gathered}$$

The last formula is in Π_4-form. A different sequence of transformations would produce the following logically equivalent formula in Π_2-form:

$$(\forall x)(\forall z)(\exists y)(\exists w)((R(x) \to P(y)) \wedge (P(z) \to R(w)));$$

it is equally possible to generate the logically equivalent formula

$$(\exists y)(\exists w)(\forall x)(\forall z)((R(x) \to P(y)) \wedge (P(z) \to R(w))),$$

which is in Σ_2-form. $\square$

The following two theorems will be used to obtain another important normal form for first-order formulas called the Skolem[7] normal form. The first theorem is essentially a special case of the second. Nevertheless, we consider it separately, mainly due to a notational issue.

Theorem 4.9.12. *Let $\mathcal{L}$ be a first-order language. Suppose that φ is an $\mathcal{L}$-formula, x is a variable and c is a constant symbol not in $\mathcal{L}$. If $\mathcal{A}$ is an $\mathcal{L}$-structure, $\sigma \in \mathrm{ASSIGN}_{\mathcal{A}}$, and $(\mathcal{A}, \sigma) \models \psi$ where $\psi = (\exists x)\varphi$, then there is an expansion $\mathcal{B}$ of $\mathcal{A}$ to $\mathcal{L}' = \mathcal{L} \cup \{c\}$ such that $(\mathcal{B}, \sigma) \models \theta$, where $\theta = (\varphi)_{x:=c}$ is an $\mathcal{L}'$-formula and $\mathtt{FV}(\theta) = \mathtt{FV}(\psi)$.*

Conversely, $\theta \models \psi$, hence if θ is satisfiable in an $\mathcal{L}'$-structure $\mathcal{B}$, then ψ is also satisfiable in $\mathcal{B}$.

[7] Albert Thoralf Skolem was born in Sandsvaer, Norway on May 23, 1887 and died on March 23, 1963 in Oslo. Skolem studied at the University of Oslo and in Göttingen (between 1915–1916). He received his doctorate in Oslo and taught at the University of Oslo until 1950. His main fields of research were logic and the foundations of mathematics; he also worked on algebra, number theory, set theory, algebraic topology, and Dirichlet series.

Proof. Suppose that $(\mathcal{A}, \sigma) \models (\exists x)\varphi$ for some $\sigma \in \mathrm{ASSIGN}_{\mathcal{A}}$. Thus, there exists $a \in |\mathcal{A}|$ such that $(\mathcal{A}, [x \to a]\sigma) \models \varphi$. Consider the expansion $\mathcal{B}$ of $\mathcal{A}$ to $\mathcal{L}'$ given by $c^{\mathcal{B}} = a$. By Theorem 4.5.24, we have $(\mathcal{B}, [x \to a]\sigma) \models \varphi$, which amounts to $(\mathcal{B}, [x \to \sigma^{\mathcal{B}}(c)]\sigma) \models \varphi$. Since c is substitutable for x in φ, by Corollary 4.6.6, we have $(\mathcal{B}, \sigma) \models (\varphi)_{x:=c}$. By Theorem 4.3.83, $\mathtt{FV}(\theta) = \mathtt{FV}(\varphi) - \{x\} = \mathtt{FV}(\psi)$.

Conversely, $\theta \models \psi$ by Theorem 4.6.7. □

Theorem 4.9.13. *Let $\mathcal{L}$ be a first-order language, $y_0, \ldots, y_{n-1}$ be n distinct variables, $n \geq 1$, and let φ be an $\mathcal{L}$-formula. Suppose that x is a variable and f is an n-ary function symbol not in $\mathcal{L}$ such that $f(y_0, \ldots, y_{n-1})$ is substitutable for x in φ. If $\mathcal{A}$ is an $\mathcal{L}$-structure, $\sigma \in \mathrm{ASSIGN}_{\mathcal{A}}$, and $(\mathcal{A}, \sigma) \models \psi$, where $\psi = (\forall y_0) \cdots (\forall y_{n-1})(\exists x)\varphi$, then there is an expansion $\mathcal{B}$ of $\mathcal{A}$ to $\mathcal{L}' = \mathcal{L} \cup \{f\}$ such that $(\mathcal{B}, \sigma) \models \theta$, where*

$$\theta = (\forall y_0) \cdots (\forall y_{n-1})(\varphi)_{x:=f(y_0,\ldots,y_{n-1})}$$

is an $\mathcal{L}'$-formula with $\mathtt{FV}(\theta) = \mathtt{FV}(\psi)$.

Conversely, $\theta \models \psi$, hence if θ is satisfiable in an $\mathcal{L}'$-structure $\mathcal{B}$, then ψ is also satisfiable in $\mathcal{B}$.

Proof. Suppose that $\sigma \in \mathrm{ASSIGN}_{\mathcal{A}}$ and that $(\mathcal{A}, \sigma) \models \psi$. Then, for all $a_0, \ldots, a_{n-1} \in |\mathcal{A}|$,

$$(\mathcal{A}, [y_{n-1} \to a_{n-1}] \cdots [y_0 \to a_0]\sigma) \models (\exists x)\varphi,$$

that is, for all $a_0, \ldots, a_{n-1} \in |\mathcal{A}|$ there is $b \in |\mathcal{A}|$ such that

$$(\mathcal{A}, [x \to b][y_{n-1} \to a_{n-1}] \cdots [y_0 \to a_0]\sigma) \models \varphi.$$

Therefore, we can define a function $F : |\mathcal{A}|^n \longrightarrow |\mathcal{A}|$ such that for all $a_0, \ldots, a_{n-1} \in |\mathcal{A}|$,

$$(\mathcal{A}, [x \to F(a_0, \ldots, a_{n-1})][y_{n-1} \to a_{n-1}] \cdots [y_0 \to a_0]\sigma) \models \varphi.$$

(We have used the Axiom of Choice when we assert the existence of the function F.)

Define the expansion $\mathcal{B}$ of $\mathcal{A}$ to $\mathcal{L}'$ by $f^{\mathcal{B}} = F$. We claim that $(\mathcal{B}, \sigma) \models \theta$, that is, for all $a_0, \ldots, a_{n-1} \in |\mathcal{A}|$,

$$(\mathcal{B}, \sigma_{a_0,\ldots,a_{n-1}}) \models (\varphi)_{x:=f(y_0,\ldots,y_{n-1})},$$

where $\sigma_{a_0,\ldots,a_{n-1}} = [y_{n-1} \to a_{n-1}] \cdots [y_0 \to a_0]\sigma$. From the Substitution Corollary, we obtain the equivalent statement: for all $a_0, \ldots, a_{n-1} \in |\mathcal{A}|$,

$$(\mathcal{B}, [x \to \sigma^{\mathcal{B}}_{a_0,\ldots,a_{n-1}}(f(y_0, \ldots, y_{n-1}))]\sigma_{a_0,\ldots,a_{n-1}}) \models \varphi.$$

Since the variables $y_0, \ldots, y_{n-1}$ are pairwise distinct, we have

$$\sigma^{\mathcal{B}}_{a_0,\ldots,a_{n-1}}(f(y_0, \ldots, y_{n-1})) = F(a_0, \ldots, a_{n-1})$$

which gives a further equivalent statement

$$(\mathcal{B}, [x \to F(a_0, \ldots, a_{n-1})]\sigma_{a_0,\ldots,a_{n-1}}) \models \varphi,$$

for all $a_0, \ldots, a_{n-1} \in |\mathcal{A}|$. By Theorem 4.5.24, the last statement is equivalent to

$$(\mathcal{A}, [x \to F(a_0, \ldots, a_{n-1})]\sigma_{a_0,\ldots,a_{n-1}}) \models \varphi,$$

for all $a_0, \ldots, a_{n-1} \in |\mathcal{A}|$, which is what we established earlier.

Note that $\mathtt{FV}(\psi) = \mathtt{FV}(\varphi) - \{y_0, \ldots, y_{n-1}, x\}$. By Corollary 4.3.84,

$$\begin{aligned} \mathtt{FV}(\varphi) - \{x\} &\subseteq \mathtt{FV}((\varphi)_{x:=f(y_0,\ldots,y_{n-1})}) \\ &\subseteq (\mathtt{FV}(\varphi) - \{x\}) \cup \{y_0, \ldots, y_{n-1}\} \end{aligned}$$

so $\mathtt{FV}(\theta) = (\mathtt{FV}(\varphi) - \{x\}) - \{y_0, \ldots, y_{n-1}\} = \mathtt{FV}(\psi)$.

To prove the last part of the theorem, note that $(\varphi)_{x:=f(y_0,\ldots,y_{n-1})} \models (\exists x)\varphi$ by Theorem 4.6.7. Therefore, by Theorem 4.5.48,

$$\theta = (\forall y_0) \cdots (\forall y_{n-1})(\varphi)_{x:=f(y_0,\ldots,y_{n-1})} \models (\forall y_0) \cdots (\forall y_{n-1})(\exists x)\varphi = \psi.$$

$\square$

The process described in the next algorithm is called *Skolemization.*

Algorithm 4.9.14 (Skolemization Algorithm).
Input: A first-order language $\mathcal{L}$ and an $\mathcal{L}$-formula φ.
Output: A universal formula ψ such that

- $\mathtt{FV}(\psi) = \mathtt{FV}(\varphi)$,
- $(\mathcal{L}_\psi - \mathcal{L}_\varphi) \cap \mathcal{L} = \emptyset$, and
- φ, ψ are equisatisfiable. In fact:
  - — $\psi \models \varphi$, and
  - — if $\mathcal{A}$ is an $\mathcal{L}$-structure and $(\mathcal{A}, \sigma) \models \varphi$, then $(\mathcal{B}, \sigma) \models \psi$, for some expansion $\mathcal{B}$ of $\mathcal{A}$ to $\mathcal{L} \cup \mathcal{L}_\psi$.

Method:

(A) Let α be an $\mathcal{L}$-formula that is a prenex normal form for φ. (α could be obtained by the algorithm implicit in the proof of Theorem 4.9.9.) We may assume that all quantified variables in α occur in the matrix.

(B) If α is universal, halt and return α. Else, $\alpha = (\forall y_0) \cdots (\forall y_{n-1})(\exists x)\beta$ for some $n \geq 0$. Let f be a n-ary function symbol not in $\mathcal{L}$ (for instance, one could let f be f_i^n where i is minimal such that $f_i^n \notin \mathcal{L}$) and let $t = f$ if $n = 0$ and $t = f(y_0, \ldots, y_{n-1})$ when $n > 0$. Define $\gamma = (\forall y_0) \cdots (\forall y_{n-1})(\beta)_{x:=t}$ and $\widehat{\mathcal{L}} = \mathcal{L} \cup \{f\}$. Repeat Step (B) with α replaced by γ and $\mathcal{L}$ replaced by $\widehat{\mathcal{L}}$.

Proof of Correctness: Note that in each execution of Step (B) except the last, the formula γ yielded by this step is in prenex normal form. Also, in each execution of Step (B) beyond the first, the number of occurrences of existential quantifiers in the formula α is one less than in the previous execution. Thus, the algorithm will halt.

Suppose that the algorithm halts at the mth execution of Step (B). Let α_i be the value of α and let $\mathcal{L}_i$ be the current first-order language at the beginning of the ith execution of Step (B). It is easy to show by induction on i, $1 \leq i \leq m$, using Theorems 4.4.38, 4.9.12 and 4.9.13, that $\mathcal{L}_i = \mathcal{L} \cup \mathcal{L}_{\alpha_i}$, α_i is an $\mathcal{L}_i$-formula with $\mathtt{FV}(\alpha_i) = \mathtt{FV}(\alpha) = \mathtt{FV}(\varphi)$, $\alpha_i \models \varphi$, and, if $\mathcal{A}$ is an $\mathcal{L}$-structure such that $(\mathcal{A}, \sigma) \models \varphi$, then there is an expansion $\mathcal{B}_i$ of $\mathcal{A}$ to $\mathcal{L}_i$ such that $(\mathcal{B}_i, \sigma) \models \alpha_i$. Since α_m is returned by the algorithm, the correctness follows immediately. $\square$

Definition 4.9.15. A *Skolem normal form* of a formula φ is any formula obtained by an application of Algorithm 4.9.14 to a first-order language $\mathcal{L}$ and to φ, where φ is an $\mathcal{L}$-formula. ▯

Example 4.9.16. Let $\varphi = ((\exists x)(\forall y)R(x,y) \wedge (\neg(\forall y)(\exists x)R(x,y)))$ be an $\mathcal{L}$-formula, where $\mathcal{L} = \{R\}$ and $x \neq y$. Using the techniques discussed earlier, we obtain a prenex normal form for φ given by

$$\alpha = (\exists x)(\exists z)(\forall y)(\forall w)(R(x,y) \wedge (\neg R(w,z))).$$

Applying Step (B) twice, we get the following sequence of values for $\mathcal{L}$ and α where c and d are constant symbols.

$\mathcal{L}$	α
$\{R\}$	$(\exists x)(\exists z)(\forall y)(\forall w)(R(x,y) \wedge (\neg R(w,z)))$
$\{R,c\}$	$(\exists z)(\forall y)(\forall w)(R(c,y) \wedge (\neg R(w,z)))$
$\{R,c,d\}$	$(\forall y)(\forall w)(R(c,y) \wedge (\neg R(w,d)))$.

The matrix of the last formula is $(R(c,y) \wedge (\neg R(w,d)))$. ▯

Example 4.9.17. Let $\mathcal{L}$ be the first-order language $\{P,R\}$ and let φ be the $\mathcal{L}$-formula $((\exists x)R(x) \leftrightarrow (\exists x)P(x))$. From Example 4.9.11, we have the prenex normal form $\alpha = (\forall x)(\exists y)(\forall z)(\exists w)((R(x) \to P(y)) \wedge (P(z) \to R(w)))$. Applying Step (B) repeatedly, we get the following sequence of values for $\mathcal{L}$ and α where f is a unary and g is a binary function symbol.

$\mathcal{L}$	α
$\{R,P\}$	$(\forall x)(\exists y)(\forall z)(\exists w)((R(x) \to P(y)) \wedge (P(z) \to R(w)))$
$\{R,P,f\}$	$(\forall x)(\forall z)(\exists w)((R(x) \to P(f(x))) \wedge (P(z) \to R(w)))$
$\{R,P,f,g\}$	$(\forall x)(\forall z)((R(x) \to P(f(x))) \wedge (P(z) \to R(g(x,z))))$

▯

Definition 4.9.18. Let $\mathcal{L}$ be a first-order language, and let Γ be a set of $\mathcal{L}$-formulas. A set of formulas Γ' is called a *Skolemization of* Γ if $\Gamma' = \{\varphi' \mid \varphi \in \Gamma\}$ where each φ' is a Skolem normal form for φ and the following additional condition is satisfied: $(\mathcal{L}_{\varphi'} - \mathcal{L}_{\varphi}) \cap (\mathcal{L}_{\psi'} - \mathcal{L}_{\psi}) = \emptyset$ for all $\varphi, \psi \in \Gamma$ such that $\varphi \neq \psi$. ▯

Theorem 4.9.19. *Let $\mathcal{L}$ be a first-order language and let Γ be a set of $\mathcal{L}$-formulas. Then, there is a first-order language $\mathcal{L}'$ such that $\mathcal{L} \subseteq \mathcal{L}'$, $\mathcal{L}' - \mathcal{L}$ consists solely of function symbols, and a set of $\mathcal{L}'$-formulas Γ' such that Γ' is a Skolemization of Γ.*

Proof. Since $\mathcal{L}$ is a first-order language, for every $n \in \mathbf{N}$, there is an infinite set of n-ary function symbols F_n such that $\mathcal{L} \cup F_n$ is a first-order language and $\mathcal{L} \cap F_n = \emptyset$. In other words, there is an infinite set of n-ary function symbols F_n outside the language $\mathcal{L}$ such that there are infinitely many n-ary function symbols outside $\mathcal{L} \cup F_n$.

Suppose that $\Gamma = \{\varphi_0, \varphi_1, \ldots\}$ (where Γ could be finite or infinite). We define a sequence of first-order languages $\mathcal{L}_0, \mathcal{L}_1, \ldots$ such that $\mathcal{L}_i$ is a finite extension of $\mathcal{L}$ for $i \geq 0$, and a sequence of formulas $\varphi'_0, \varphi'_1, \ldots$ as follows.

Let $\mathcal{L}_0 = \mathcal{L}$. Suppose we have constructed $\mathcal{L}_i$, a finite extension of $\mathcal{L}$ and that there is a φ_i (in other words, Γ contains at least $i+1$ formulas). Since φ_i is an $\mathcal{L}$-formula, it is an $\mathcal{L}_i$-formula. Let φ'_i be the formula obtained through the application of Algorithm 4.9.14 to $\mathcal{L}_i$ and φ_i, where the new function symbols needed for $\mathcal{L}_{\varphi'_i} - \mathcal{L}_i$ are taken from the set $\bigcup\{F_n \mid n \in \mathbf{N}\}$, and let $\mathcal{L}_{i+1} = \mathcal{L} \cup \mathcal{L}_{\varphi'_i}$. Observe that these function symbols always exist because each of the sets F_n is infinite, $\mathcal{L}_i$ is a finite extension of $\mathcal{L}$, and $\mathcal{L}_{i+1} - \mathcal{L}_i$ is finite.

Define $\mathcal{L}' = \mathcal{L}_0 \cup \mathcal{L}_1 \cup \cdots$ and $\Gamma' = \{\varphi'_0, \varphi'_1, \ldots\}$. The set $\mathcal{L}'$ is a language since $\mathcal{L}' \subseteq \mathcal{L} \cup \bigcup\{F_n \mid n \in \mathbf{N}\}$. Suppose that φ_j is defined and $0 \leq i < j$. Since $\mathcal{L}_{\varphi'_i} - \mathcal{L}_{\varphi_i} \subseteq \mathcal{L}_{i+1} \subseteq \mathcal{L}_j$, it follows that $(\mathcal{L}_{\varphi'_i} - \mathcal{L}_{\varphi_i}) \cap (\mathcal{L}_{\varphi'_j} - \mathcal{L}_{\varphi_j})$. Thus, Γ' is a Skolemization of Γ. $\square$

Theorem 4.9.20. *Let $\mathcal{L} \subseteq \mathcal{L}'$ be first-order languages, Γ be a set of $\mathcal{L}$-formulas and $\Gamma' = \{\varphi' \mid \varphi \in \Gamma\}$ be a set of $\mathcal{L}'$-formulas that is a Skolemization of Γ. If $\mathcal{A}$ is an $\mathcal{L}$-structure such that $(\mathcal{A}, \sigma)$ satisfies Γ, then there is an expansion $\mathcal{B}$ of $\mathcal{A}$ to $\mathcal{L}'$ such that $(\mathcal{B}, \sigma)$ satisfies Γ'.*

Conversely, if $\mathcal{B}$ is an $\mathcal{L}'$-structure such that $(\mathcal{B}, \sigma)$ satisfies Γ', then $(\mathcal{B}, \sigma)$ satisfies Γ.

Proof. If $(\mathcal{A}, \sigma)$ satisfies Γ, we have $(\mathcal{A}, \sigma) \models \varphi$ for every $\varphi \in \Gamma$. By the correctness of Algorithm 4.9.14, for each $\varphi \in \Gamma$, there is an expansion $\mathcal{B}_{\varphi'}$ of $\mathcal{A}$ to $\mathcal{L} \cup \mathcal{L}_{\varphi'}$ such that $(\mathcal{B}_{\varphi'}, \sigma) \models \varphi'$. Define the expansion $\mathcal{B}'$ of $\mathcal{A}$ to $\mathcal{L} \cup \bigcup\{\mathcal{L}_{\varphi'} \mid \varphi \in \Gamma\}$ by

$$s^{\mathcal{B}'} = \begin{cases} s^{\mathcal{A}} & \text{if } s \in \mathcal{L} \\ s^{\mathcal{B}_{\varphi'}} & \text{if } s \in \mathcal{L}_{\varphi'} - \mathcal{L}. \end{cases}$$

Since $(\mathcal{L}_{\varphi'} - \mathcal{L}_{\varphi}) \cap (\mathcal{L}_{\psi'} - \mathcal{L}_{\psi}) = \emptyset$ for all $\varphi, \psi \in \Gamma$ with $\varphi \neq \psi$, the structure $\mathcal{B}'$ is well-defined. Observe that $\mathcal{B}'$ is an expansion of every structure $\mathcal{B}_{\varphi'}$ and, therefore, $(\mathcal{B}', \sigma) \models \varphi'$, for $\varphi \in \Gamma$. Since

$\mathcal{L} \cup \bigcup\{\mathcal{L}_{\varphi'} \mid \varphi \in \Gamma\} \subseteq \mathcal{L}'$, we may take $\mathcal{B}$ to be any expansion of $\mathcal{B}'$ to $\mathcal{L}'$.

The converse statement follows from the fact that φ' logically implies φ for each $\varphi \in \Gamma$. □

Corollary 4.9.21. *Let $\mathcal{L}, \mathcal{L}'$ be first-order languages and Γ be a set of $\mathcal{L}$-formulas. If Γ' is a set of $\mathcal{L}'$-formulas that is Skolemization of Γ, then Γ is satisfiable if and only if Γ' is.*

Proof. This follows immediately from Theorem 4.9.20. □

4.10 Reduction of First-Order Logic to Propositional Logic

Definition 4.10.1. Let $\mathcal{L}$ be a first-order language. The *Herbrand*[8] *extension of* $\mathcal{L}$ is the first-order language $\mathcal{H}(\mathcal{L})$ defined by

$$\mathcal{H}(\mathcal{L}) = \begin{cases} \mathcal{L} & \text{if } \mathcal{L} \text{ contains at least one constant symbol} \\ \mathcal{L} \cup \{f_0^0\} & \text{otherwise.} \end{cases}$$

▯

In other words, if $\mathcal{L}$ contains constant symbols, then $\mathcal{H}(\mathcal{L})$ coincides with $\mathcal{L}$; otherwise, we add the constant symbol f_0^0 to $\mathcal{L}$ in order to obtain $\mathcal{H}(\mathcal{L})$. Also, note that $\mathcal{H}(\mathcal{L})$ is a first-order language because, if there are infinitely many constant symbols which are not elements of $\mathcal{L}$, then so is the case for $\mathcal{H}(\mathcal{L})$.

Example 4.10.2. For the first-order language $\mathcal{L}_{ar}$ of Example 4.2.3 we have $\mathcal{H}(\mathcal{L}_{ar}) = \mathcal{L}_{ar}$ because $\mathcal{L}_{ar}$ contains the individual constant symbol 0. ▯

Definition 4.10.3. Let $\mathcal{L}$ be a first-order language and V be an $\mathcal{L}$-suitable set of variables. A *V-Herbrand structure* for $\mathcal{L}$ is an

[8]Jacques Herbrand was born in Paris on February 12, 1908. He was one of the most prominent logicians of the 20th century. Herbrand studied at l'École Normale Supérieure in Paris between 1925 and 1928, where he also completed his dissertation in 1930. His contributions in logic impact heavily the current applications of logic to computer science. Herbrand died in a mountain-climbing accident at La Bérarde (Isère, France) on July 27, 1931.

$\mathcal{L}$-structure $\mathcal{A}$ such that $|\mathcal{A}| = \text{TERM}_{\mathcal{L}}(V)$, $c^{\mathcal{A}} = c$ for all constant symbols $c \in \mathcal{L}$, and $f^{\mathcal{A}}(t_0, \ldots, t_{n-1}) = f(t_0, \ldots, t_{n-1})$ for every n-ary function symbol $f \in \mathcal{L}$ with $n > 0$.

If $V = \emptyset$, we refer to a V-Herbrand structure for $\mathcal{L}$ simply as a *Herbrand structure* for $\mathcal{L}$.

If Γ is a set of $\mathcal{L}$-formulas and $\mathcal{A}$ is V-Herbrand structure that is a model of Γ, then we refer to $\mathcal{A}$ as an $(\mathcal{L}, V)$-*Herbrand model* of Γ. If $\mathcal{A}$ is an $(\mathcal{L}, \emptyset)$-Herbrand model for Γ then we say that $\mathcal{A}$ is an $\mathcal{L}$-*Herbrand model* of Γ. $\Box$

Since the interpretation of the relation symbols of $\mathcal{L}$ is left open in Definition 4.10.3, in general, there is no unique V-Herbrand structure of $\mathcal{L}$. We denote by $\text{HERBRAND}_{\mathcal{L},V}$ the set of all V-Herbrand structures for $\mathcal{L}$. Further, we will write $\text{HERBRAND}_{\mathcal{L}}$ for $\text{HERBRAND}_{\mathcal{L},\emptyset}$.

Example 4.10.4. Let $\mathcal{L}$ be the first-order language $\{0, s, R\}$, where 0 is a constant symbol, s is a unary function symbol and R is a binary relation symbol. The universe of an $\mathcal{L}$-Herbrand structure $\mathcal{A}$ is $|\mathcal{A}| = \{s^n(0) \mid n \geq 0\}$, where $s^0(0) = 0$ and $s^{n+1}(0) = s(s^n(0))$ for $n \geq 0$. The symbols 0 and s are interpreted as $0^{\mathcal{A}} = 0$ and $s^{\mathcal{A}}(s^n(0)) = s^{n+1}(0)$, for $n \in \mathbf{N}$. $R^{\mathcal{A}}$ can be any binary relation on $|\mathcal{A}|$. For instance, we can define

$$R^{\mathcal{A}} = \{(s^n(0), s^m(0)) \mid n < m\},$$

or

$$R^{\mathcal{A}} = \{(s^n(0), s^{np}(0)) \mid n, p \in \mathbf{N}\}.$$

If $\mathcal{L}' = \{s, R\}$, then the set of $\mathcal{L}'$-ground terms is empty. However, if $V = \{x\}$, then the set of $(\mathcal{L}', V)$-terms is $\{s^n(x) \mid n \in \mathbf{N}\}$. In any $(\mathcal{L}', V)$-Herbrand structure $\mathcal{A}'$, we have $s^{\mathcal{A}'}(s^n(x)) = s^{n+1}(x)$ for $n \in \mathbf{N}$. Specific Herbrand structures could be obtained by defining

$$R^{\mathcal{A}'} = \{(s^n(x), s^m(x) \mid n < m\},$$

or

$$R^{\mathcal{A}'} = \{(s^n(x), s^{np}(x)) \mid n, p \in \mathbf{N}\}.$$

$\Box$

It is sometimes convenient to relate V-Herbrand structures for a first-order language $\mathcal{L}$ to sets of non-equality $(\mathcal{L}, V)$-atomic formulas. This is done using the bijections $\mathsf{AF}_{\mathcal{L},V}$ and $\mathsf{STR}_{\mathcal{L},V}$ introduced below.

Definition 4.10.5. Let $\mathcal{L}$ be a first-order language, V be an $\mathcal{L}$-suitable set of variables. For an $(\mathcal{L}, V)$-Herbrand structure $\mathcal{A}$, the *set of non-equality* $(\mathcal{L}, V)$*-atomic formulas defined by* $\mathcal{A}$ is

$$\begin{aligned}\mathsf{AF}_{\mathcal{L},V}(\mathcal{A}) = \{R(t_0, \ldots, t_{n-1}) \mid {} & (t_0, \ldots, t_{n-1}) \in R^{\mathcal{A}} \\ & \text{where } t_0, \ldots, t_{n-1} \in \mathrm{TERM}_{\mathcal{L}}(V) \text{ and} \\ & R \text{ is an } n\text{-ary relation symbol in } \mathcal{L} - \{=\} \text{ and } n > 0\} \\ & \cup\{R \mid R^{\mathcal{A}} = \mathbf{T} \text{ and } R \text{ is a propositional constant of } \mathcal{L}\}.\end{aligned}$$

If S is a set of non-equality $(\mathcal{L}, V)$-atomic formulas, the $(\mathcal{L}, V)$-*Herbrand structure defined by* S is

$$\begin{aligned}\mathsf{STR}_{\mathcal{L},V}(S) = \mathcal{A}, \text{ where } & \mathcal{A} \text{ is the } (\mathcal{L}, V)\text{-Herbrand structure such that} \\ & R^{\mathcal{A}} \text{ is } \{(t_0, \ldots, t_{n-1}) \mid R(t_0, \ldots, t_{n-1}) \in S\} \text{ for} \\ & \text{every } n\text{-ary relation symbol } R \text{ in } \mathcal{L} - \{=\} \text{ with } n > 0 \\ & \text{and } R^{\mathcal{A}} = \mathbf{T} \text{ if and only if } R \in S \\ & \text{for every propositional constant } R \text{ of } \mathcal{L}.\end{aligned}$$

If $\mathcal{L}$ contains a constant symbol and $\mathcal{A}$ is an $\mathcal{L}$-Herbrand structure, we define $\mathsf{GAF}_{\mathcal{L}}(\mathcal{A}) = \mathsf{AF}_{\mathcal{L},\emptyset}(\mathcal{A})$; we refer to $\mathsf{GAF}_{\mathcal{L}}(\mathcal{A})$ as the *set of ground non-equality* $\mathcal{L}$*-atomic formulas determined by* $\mathcal{A}$.

If $S \subseteq \mathrm{GAFORMNE}_{\mathcal{L}}$, the $\mathcal{L}$*-Herbrand structure* $\mathsf{STR}_{\mathcal{L}}(S)$ is defined to be $\mathsf{STR}_{\mathcal{L},\emptyset}(S)$. $\square$

Theorem 4.10.6. *Let* $\mathcal{L}$ *be a first-order language and* V *be an* $\mathcal{L}$*-suitable set of variables. The mappings*

$$\mathsf{AF}_{\mathcal{L},V} : \mathrm{HERBRAND}_{\mathcal{L},V} \longrightarrow \mathcal{P}(\mathrm{AFORMNE}_{\mathcal{L}}(V)),$$
$$\mathsf{STR}_{\mathcal{L},V} : \mathcal{P}(\mathrm{AFORMNE}_{\mathcal{L}}(V)) \longrightarrow \mathrm{HERBRAND}_{\mathcal{L},V}$$

introduced in Definition 4.10.5 *are inverse bijections.*

Proof. The argument (left to the reader) consists of verifying the equalities $\mathsf{STR}_{\mathcal{L},V}(\mathsf{AF}_{\mathcal{L},V}(\mathcal{A})) = \mathcal{A}$ and $\mathsf{AF}_{\mathcal{L},V}(\mathsf{STR}_{\mathcal{L},V}(S)) = S$ for every $(\mathcal{L}, V)$-Herbrand structure $\mathcal{A}$ and every subset S of $\text{AFORMNE}_{\mathcal{L}}(V)$. □

Note that if $\mathcal{L}$ is an algebraic language, then the functions $\mathsf{AF}_{\mathcal{L},V}$ and $\mathsf{STR}_{\mathcal{L},V}$ are bijections between one-element sets.

Corollary 4.10.7. *Let $\mathcal{L}$ be a first-order language that contains at least one constant symbol. Then, the functions*

$$\mathsf{GAF}_{\mathcal{L}} : \text{HERBRAND}_{\mathcal{L}} \longrightarrow \mathcal{P}(\text{GAFORMNE}_{\mathcal{L}}),$$

$$\mathsf{STR}_{\mathcal{L}} : \mathcal{P}(\text{GAFORMNE}_{\mathcal{L}}) \longrightarrow \text{HERBRAND}_{\mathcal{L}}$$

introduced in Definition 4.10.5 *are inverse bijections.*

Proof. The argument consists of taking $V = \emptyset$ in Theorem 4.10.6. □

If $\mathcal{L}$ is an algebraic language, then the functions $\mathsf{GAF}_{\mathcal{L}}$ and $\mathsf{STR}_{\mathcal{L}}$ are bijections between one-element sets.

Theorem 4.10.8. *Let $\mathcal{A}$ be a V-Herbrand structure for a first-order language $\mathcal{L}$ and let $\sigma \in \text{ASSIGN}_{\mathcal{A}}$. If $\sigma(x) = x$ for all $x \in V$, then for all $t \in \text{TERM}_{\mathcal{L}}(V)$, we have $\sigma^{\mathcal{A}}(t) = t$.*

Proof. The argument is a straightforward induction on terms. □

Corollary 4.10.9. *Let $\mathcal{A}$ be a Herbrand structure for a first-order language $\mathcal{L}$. For every ground term t of $\mathcal{L}$, we have $t^{\mathcal{A}} = t$.*

Proof. This statement is a direct consequence of Theorem 4.10.8. □

Corollary 4.10.10. *Let $\mathcal{A}$ be a V-Herbrand structure for a first-order language $\mathcal{L}$, ρ be a congruence of $\mathcal{A}$, and let $\mathcal{B} = \mathcal{A}/\rho$. If $\sigma \in \text{ASSIGN}_{\mathcal{B}}$ is such that $\sigma(x) = [x]_\rho$ for all $x \in V$, then $\sigma^{\mathcal{B}}(t) = [t]_\rho$ for all $t \in \text{TERM}_{\mathcal{L}}(V)$.*

Proof. Define $\tau \in \text{ASSIGN}_{\mathcal{A}}$ by letting $\tau(x)$ be x for $x \in V$, and $\tau(x)$ be the first term in $\sigma(x)$, otherwise. Then, $\sigma = h_\rho \circ \tau$, where h_ρ is the canonical morphism of Definition 4.4.33. By Theorem 4.5.6, $\sigma^{\mathcal{B}}(t) = h_\rho(\tau^{\mathcal{A}}(t)) = [\tau^{\mathcal{A}}(t)]_\rho$ for all $t \in \text{TERM}_{\mathcal{L}}$. For every term $t \in \text{TERM}_{\mathcal{L}}(V)$, we have $\tau^{\mathcal{A}}(t) = t$, by Theorem 4.10.8, since $\tau(x) = x$ for all $x \in V$. Thus, $\sigma^{\mathcal{B}}(t) = [t]_\rho$, for $t \in \text{TERM}_{\mathcal{L}}(V)$. □

Lemma 4.10.11. *Let $S \subseteq \mathrm{AFORM}_{\mathcal{L}-\{=\}}(V)$ and let $\mathcal{A} = \mathsf{STR}_{\mathcal{L},V}(S)$, where $\mathcal{L}$ is a first-order language that is not an algebraic language, and V is an $\mathcal{L}$-suitable set of variables. If $\sigma \in \mathrm{ASSIGN}_{\mathcal{A}}$ is such that $\sigma(x) = x$ for all $x \in V$ and φ is an atomic formula, $\varphi \in \mathrm{AFORM}_{\mathcal{L}-\{=\}}(V)$, then $(\mathcal{A}, \sigma) \models \varphi$ if and only if $\varphi \in S$.*

Proof. Suppose initially that $\varphi = R(t_0, \ldots, t_{n-1})$ where R is an n-ary relation symbol with $n > 0$ different from $=$. We have the following equivalent statements.

$$\begin{aligned}
(\mathcal{A}, \sigma) \models \varphi \text{ if and only if } & (\sigma^{\mathcal{A}}(t_0), \ldots, \sigma^{\mathcal{A}}(t_{n-1})) \in R^{\mathcal{A}} \\
\text{if and only if } & (t_0, \ldots, t_{n-1}) \in R^{\mathcal{A}} \\
& \text{(by Theorem 4.10.8)} \\
\text{if and only if } & R(t_0, \ldots, t_{n-1}) \in S \\
& \text{(by the definition of } \mathsf{STR}) \\
\text{if and only if } & \varphi \in S.
\end{aligned}$$

We leave to the reader the case when R is a propositional constant.

□

The next theorem shows that without quantifier symbols or equality, first-order logic is essentially propositional logic where statement variables are replaced by atomic formulas. We will exploit this connection to transfer results from propositional logic to first-order logic.

Theorem 4.10.12. *Let $\mathcal{L}$ be a first-order language without equality, Γ be a set of quantifier-free $\mathcal{L}$-formulas, and Γ_0 be a fundamental propositional form for Γ. Then, Γ is satisfiable if and only if Γ_0 is satisfiable.*

Proof. One half of the statement was already shown in Theorem 4.8.12. Suppose now that Γ_0 is satisfiable and let v be a truth assignment that satisfies Γ_0. Since Γ_0 is a fundamental propositional form for Γ, there is an injective, prime inter-substitution s such that $\Gamma = s(\Gamma_0)$. Note that if $p \in SV(\Gamma_0)$, then $s(p)$ is an atomic $\mathcal{L}$-formula because $s(p)$ is a prime subformula of a formula in Γ and Γ consists of quantifier-free $\mathcal{L}$-formulas. Consider the set of atomic formulas $S = \{s(p) \mid p \in SV(\Gamma_0) \text{ and } v(p) = \mathbf{T}\}$, the structure

$\mathcal{A} = \mathsf{STR}_{\mathcal{L},\mathrm{VAR}}(S)$, and the assignment σ defined by $\sigma(x) = x$ for every $x \in \mathrm{VAR}$.

We claim that if $\alpha \in \mathrm{PLFORM}$ and $SV(\alpha) \subseteq SV(\Gamma_0)$, then $(\mathcal{A}, \sigma) \models s(\alpha)$ if and only if $v(\alpha) = \mathbf{T}$.

The argument is by induction on α. For the basis step, assume that α is a statement variable p. By Lemma 4.10.11, we have $(\mathcal{A}, \sigma) \models s(p)$ if and only if $s(p) \in S$. In view of the definition of S and the injectivity of s, this is equivalent to $v(p) = \mathbf{T}$.

We discuss one of the inductive steps, namely when $\alpha = (\neg\beta)$, where the statement holds for β, and leave the other steps to the reader. The following statements are easily seen to be equivalent.

$$(\mathcal{A}, \sigma) \models s(\alpha)$$
$$(\mathcal{A}, \sigma) \models (\neg s(\beta))$$
$$(\mathcal{A}, \sigma) \not\models s(\beta)$$
$$v(\beta) = \mathbf{F}$$
$$v(\alpha) = \mathbf{T}.$$

The result now follows from the claim. Indeed, let $\varphi \in \Gamma$. We have $\varphi = s(\alpha)$ where $\alpha \in \Gamma_0$. Since v satisfies Γ_0, $v(\alpha) = \mathbf{T}$, so, by the claim, $(\mathcal{A}, \sigma) \models s(\alpha) = \varphi$. □

Theorem 4.10.13. *A quantifier-free formula that does not contain = is logically valid if and only if it is a tautology.*

Proof. We have already seen in Theorem 4.8.14 that every tautology is logically valid. Conversely, suppose that φ is a quantifier-free logically valid formula and let α be a fundamental propositional form for φ. It is easy to see that $(\neg\alpha)$ is a fundamental propositional form for $(\neg\varphi)$. By the second part of Theorem 4.5.55, $\{(\neg\varphi)\}$ is unsatisfiable. Therefore, by Theorem 4.10.12, $\{(\neg\alpha)\}$ is unsatisfiable, so α is a tautology, which implies that φ is a tautology. □

The next example shows that Theorems 4.10.12 and 4.10.13 do not hold if equality is allowed.

Example 4.10.14. The formula $((x = y) \wedge (\neg(y = x)))$ is not satisfiable even though it has $\varphi = (p \wedge (\neg q))$ as a fundamental propositional form and φ is satisfiable.

Similarly, the formula $((x = y) \to (y = x))$ is logically valid, but is not a tautology since it has $(p \to q)$ as a fundamental propositional form and the latter formula is not a tautology. □

We now use the connection between propositional logic and first-order logic to prove a limited, preliminary version of the Compactness Theorem of first-order logic.

Theorem 4.10.15 (Compactness Theorem for Quantifier-Free Formulas without Equality). *Let $\mathcal{L}$ be a first-order language without equality and let Γ be a set of quantifier-free $\mathcal{L}$-formulas. If Γ is finitely satisfiable, then Γ is satisfiable.*

Proof. Let Γ_0 be a fundamental propositional form for Γ and assume that $\Gamma = s(\Gamma_0)$, where s is an injective, prime substitution. We claim that Γ_0 is finitely satisfiable. Indeed, if Γ_0' is a finite subset of Γ_0, then Γ_0' is a fundamental propositional form for the finite subset $s(\Gamma_0')$ of Γ, which implies that Γ_0' is satisfiable due to the satisfiability of $s(\Gamma_0')$ and to Theorem 4.10.12. By the Compactness Theorem of Propositional Logic, Theorem 2.4.3, Γ_0 is satisfiable, so Γ is satisfiable, again by Theorem 4.10.12. □

Next, we discuss results that help us to reduce the logic of universal first-order formulas to the logic of quantifier-free first-order formulas, which in turn, as we have seen, is reducible to propositional logic, when the formula does not contain equality. The full reduction of first-order logic without equality to propositional logic is completed by the Skolemization process.

Theorem 4.10.16. *Let $\mathcal{L}$ be a first-order language, V be an $\mathcal{L}$-suitable set of variables, $\mathcal{A}$ be an $\mathcal{L}$-structure and $\sigma \in \mathrm{ASSIGN}_{\mathcal{A}}$. Define a V-Herbrand structure $\mathcal{H}_V(\mathcal{A}, \sigma)$ for $\mathcal{L}$ by*

$$R^{\mathcal{H}_V(\mathcal{A},\sigma)} = \{(t_0, \ldots, t_{n-1}) \in \mathrm{TERM}_{\mathcal{L}}(V)^n \mid (\mathcal{A}, \sigma) \models R(t_0, \ldots, t_{n-1})\}$$

for each n-ary relation symbol R of $\mathcal{L} - \{=\}$ with $n > 0$, and $R^{\mathcal{H}_V(\mathcal{A},\sigma)} = R^{\mathcal{A}}$ for every propositional constant R of $\mathcal{L}$. Then, $h : |\mathcal{H}_V(\mathcal{A}, \sigma)| \longrightarrow |\mathcal{A}|$ given by $h(t) = \sigma^{\mathcal{A}}(t)$ is a morphism.

Proof. The first condition in the definition of morphism is shown as follows.

$$\begin{aligned} h(f^{\mathcal{H}_V(\mathcal{A},\sigma)}(t_0,\ldots,t_{n-1})) &= h(f(t_0,\ldots,t_{n-1})) \\ &= \sigma^{\mathcal{A}}(f(t_0,\ldots,t_{n-1})) \\ &= f^{\mathcal{A}}(\sigma^{\mathcal{A}}(t_0),\ldots,\sigma^{\mathcal{A}}(t_{n-1})) \\ &= f^{\mathcal{A}}(h(t_0),\ldots,h(t_{n-1})), \end{aligned}$$

when f is an n-ary function symbol with $n > 0$. We leave to the reader the case when f is a constant symbol.

The second condition is a consequence of the following sequence of equivalent statements.

$$\begin{aligned} &(t_0,\ldots,t_{n-1}) \in R^{\mathcal{H}_V(\mathcal{A},\sigma)} \\ &\quad\text{if and only if } (\mathcal{A},\sigma) \models R(t_0,\ldots,t_{n-1}) \\ &\quad\text{if and only if } (\sigma^{\mathcal{A}}(t_0),\ldots,\sigma^{\mathcal{A}}(t_{n-1})) \in R^{\mathcal{A}} \\ &\quad\text{if and only if } (h(t_0),\ldots,h(t_{n-1})) \in R^{\mathcal{A}}, \end{aligned}$$

when $R \in \mathcal{L} - \{=\}$ is an n-ary relation symbol with $n > 0$. The case of propositional constants is left to the reader. □

Corollary 4.10.17. *Let $\mathcal{L}$ be a first-order language that contains at least one constant symbol and let $\mathcal{A}$ be an $\mathcal{L}$-structure. Define a Herbrand structure $\mathcal{H}(\mathcal{A})$ for $\mathcal{L}$ by*

$$R^{\mathcal{H}(\mathcal{A})} = \{(t_0,\ldots,t_{n-1}) \in \mathrm{GTERM}^n_{\mathcal{L}} \mid \mathcal{A} \models R(t_0,\ldots,t_{n-1})\}$$

for each n-ary relation symbol R of $\mathcal{L} - \{=\}$ with $n > 0$ and $R^{\mathcal{H}(\mathcal{A})} = R^{\mathcal{A}}$ for every propositional constant R of $\mathcal{L}$. Then, the mapping $h : |\mathcal{H}(\mathcal{A})| \longrightarrow |\mathcal{A}|$ given by $h(t) = t^{\mathcal{A}}$ is a morphism.

Proof. This is an immediate consequence of Theorem 4.10.16. □

Example 4.10.18. Let $\mathcal{L}$ be the first-order language from Example 4.10.4 and let $\mathcal{A}$ be the $\mathcal{L}$-structure with $|\mathcal{A}| = \mathbf{N}$, $0^{\mathcal{A}} = 0$, $s^{\mathcal{A}}(n) = n + 1$ and $R^{\mathcal{A}} = \{(n, m) \mid n < m\}$. (Note that in the equality $0^{\mathcal{A}} = 0$, the 0 on the left is a constant symbol whereas the 0 on the right is the number zero.) The Herbrand structure $\mathcal{H}(\mathcal{A})$ is the first Herbrand structure considered in Example 4.10.4 and the morphism h of Corollary 4.10.17 is given by $h(s^n(0)) = n$, for $n \in \mathbf{N}$. It is clear that h is an isomorphism.

Consider now the extension $\mathcal{L}_1$ of $\mathcal{L}$ given by $\mathcal{L}_1 = \{0, s, f, R\}$, where f is a binary function symbol and the $\mathcal{L}_1$-structure $\mathcal{A}_1$ which is the expansion of $\mathcal{A}$ defined by $f^{\mathcal{A}_1}(n, m) = n + m$. In this case, the morphism $h : \mathcal{H}(\mathcal{A}_1) \longrightarrow \mathcal{A}_1$ is an epimorphism, but it is not an isomorphism because, for example, $h(0) = h(f(0, 0))$ and the terms 0 and $f(0, 0)$ are distinct. ∎

Theorem 4.10.19. *Let $\mathcal{L}$ be a first-order language without equality, Γ be a set of universal $\mathcal{L}$-formulas and V be an $\mathcal{L}$-suitable set of variables such that $\mathtt{FV}(\Gamma) \subseteq V$. Then, the following statements are equivalent.*

(1) Γ *is satisfiable.*
(2) Γ *is satisfiable in a V-Herbrand structure for $\mathcal{L}$.*
(3) $\mathrm{INST}_{\mathcal{L},V}(\Gamma)$ *is satisfiable.*
(4) $\mathrm{INST}_{\mathcal{L},V}(\Gamma)$ *is satisfiable in a V-Herbrand structure for $\mathcal{L}$.*

Proof. It is clear that (2) implies (1) and (4) implies (3). From Theorem 4.6.9 we infer that (1) implies (3) and (2) implies (4). Thus, it only remains to show that (3) implies (2). These implications are shown in the following diagram.

$$\begin{tikzcd}
(1) \arrow[r] & (3) \arrow[dl] \\
(2) \arrow[u] \arrow[r] & (4) \arrow[u]
\end{tikzcd}$$

Suppose that $\mathcal{A}$ is an $\mathcal{L}$-structure, $\sigma \in \mathrm{ASSIGN}_{\mathcal{A}}$ and $(\mathcal{A}, \sigma) \models \theta$ for every $\theta \in \mathrm{INST}_{\mathcal{L},V}(\Gamma)$. Let $\mathcal{B}$ be the V-Herbrand structure $\mathcal{H}_V(\mathcal{A}, \sigma)$ and let $h : |\mathcal{B}| \to |\mathcal{A}|$ be the morphism of Theorem 4.10.16. Define $\tau \in \mathrm{ASSIGN}_{\mathcal{B}}$ by $\tau(x) = x$ if $x \in V$ and $\tau(x) = t_0$ for $x \notin V$, where t_0 is an arbitrary term in $\mathrm{TERM}_{\mathcal{L}}(V)$. (Note that t_0 exists because when $V = \emptyset$, $\mathcal{L}$ contains at least one constant symbol.) By Theorem 4.10.8, $\tau^{\mathcal{B}}(t) = t$ for all $t \in \mathrm{TERM}_{\mathcal{L}}(V)$. By the Morphism Theorem, we have $(\mathcal{B}, \tau) \models \psi$ if and only if $(\mathcal{A}, h \circ \tau) \models \psi$, for every quantifier-free formula ψ of $\mathcal{L}$. If, in addition, $\mathtt{FV}(\psi) \subseteq V$, then $h \circ \tau$ agrees with σ on all variables in $\mathtt{FV}(\psi)$ because $h \circ \tau(x) = h(x) = \sigma^{\mathcal{A}}(x) = \sigma(x)$, for $x \in V$. Thus, by the Agreement Theorem,

$(\mathcal{A}, h \circ \tau) \models \psi$ if and only if $(\mathcal{A}, \sigma) \models \psi$, so $(\mathcal{B}, \tau) \models \psi$ if and only if $(\mathcal{A}, \sigma) \models \psi$.

Let $\varphi = (\forall y_0) \cdots (\forall y_{n-1})\psi$, with ψ quantifier-free, be a formula in Γ and let $t_0, \ldots, t_{n-1}$ be n terms in $\mathrm{TERM}_{\mathcal{L}}(V)$. We have the following equivalent statements.

$(\mathcal{B}, [y_{n-1} \to t_{n-1}] \cdots [y_0 \to t_0]\tau) \models \psi$

if and only if $(\mathcal{B}, [y_0 \to t_0] \cdots [y_{n-1} \to t_{n-1}]\tau) \models \psi$
(since the y_j's are distinct variables)

if and only if $(\mathcal{B}, [y_0 \to \tau^{\mathcal{B}}(t_0)] \cdots [y_{n-1} \to \tau^{\mathcal{B}}(t_{n-1})]\tau) \models \psi$
(by previous discussion)

if and only if $(\mathcal{B}, \tau) \models (\psi)_{y_0,\ldots,y_{n-1}:=t_0,\ldots,t_{n-1}}$
(by the Substitution Corollary, as ψ is quantifier-free)

if and only if $(\mathcal{A}, \sigma) \models (\psi)_{y_0,\ldots,y_{n-1}:=t_0,\ldots,t_{n-1}}$
(by previous discussion)

Since $(\psi)_{y_0,\ldots,y_{n-1}:=t_0,\ldots,t_{n-1}} \in \mathrm{INST}_{\mathcal{L},V}(\Gamma)$, the last of the equivalent statements is true, so

$$(\mathcal{B}, [y_{n-1} \to t_{n-1}] \cdots [y_0 \to t_0]\tau) \models \psi$$

for all $t_0, \ldots, t_{n-1} \in \mathrm{TERM}_{\mathcal{L}}(V)$. Thus, $(\mathcal{B}, \tau) \models \varphi$, which shows that Γ is satisfiable on a V-Herbrand structure for $\mathcal{L}$. □

A special case of Theorem 4.10.19 can be obtained by taking $V = \emptyset$.

Corollary 4.10.20. *Let $\mathcal{L}$ be a first-order language without equality that contains at least one constant symbol and Γ be a set of closed universal $\mathcal{L}$-formulas. Then, the following statements are equivalent.*

(1) Γ *has a model.*
(2) Γ *has an $\mathcal{L}$-Herbrand model.*
(3) $\mathrm{GINST}_{\mathcal{L}}(\Gamma)$ *has a model.*
(4) $\mathrm{GINST}_{\mathcal{L}}(\Gamma)$ *has an $\mathcal{L}$-Herbrand model.*

Proof. This follows immediately from Theorem 4.10.19 by taking $V = \emptyset$. □

Theorem 4.10.21 (Compactness Theorem of First-Order Logic without Equality). *Let $\mathcal{L}$ be a first-order language without equality and let Γ be a set of $\mathcal{L}$-formulas. Then, Γ is finitely satisfiable if and only if Γ is satisfiable.*

Proof. It is clear that if Γ is satisfiable, then it is finitely satisfiable.

Conversely, suppose that Γ is finitely satisfiable. By Theorem 4.9.19, there is a first-order language $\mathcal{L}'$ and a set Γ' of $\mathcal{L}'$-formulas such that Γ' is a Skolemization of Γ. If Γ'_0 is a finite subset of Γ', then there is a finite subset Γ_0 of Γ such that Γ'_0 is a Skolemization of Γ_0. Since Γ is finitely satisfiable, Γ_0 is satisfiable, so Γ'_0 is satisfiable by Corollary 4.9.21. Thus, Γ' is finitely-satisfiable.

Next, we claim that $\mathrm{INST}_{\mathcal{L}',\mathrm{VAR}}(\Gamma')$ is finitely satisfiable. Indeed, if Γ''_0 is a finite subset of $\mathrm{INST}_{\mathcal{L}',\mathrm{VAR}}(\Gamma')$, then there is a finite subset Γ'_0 of Γ' such that $\Gamma''_0 \subseteq \mathrm{INST}_{\mathcal{L}',\mathrm{VAR}}(\Gamma'_0)$. By the finite satisfiability of Γ', Γ'_0 is satisfiable, so $\mathrm{INST}_{\mathcal{L}',\mathrm{VAR}}(\Gamma'_0)$ is satisfiable by Theorem 4.10.19. Therefore, Γ''_0 is satisfiable.

By Theorem 4.10.15 (the Compactness Theorem for Quantifier-Free Formulas without Equality), $\mathrm{INST}_{\mathcal{L}',\mathrm{VAR}}(\Gamma')$ is satisfiable. Thus, Γ' is satisfiable by Theorem 4.10.19, Finally, by Corollary 4.9.21, Γ is satisfiable. □

Theorem 4.10.22. *Let $\mathcal{L}$ be a first-order language without equality and Γ be a set of $\mathcal{L}$-formulas. Assume further that $\mathcal{L}'$ is also a first-order language without equality and Γ', a set of $\mathcal{L}'$-formulas, is a Skolemization of Γ, and let $V = \mathtt{FV}(\Gamma) = \mathtt{FV}(\Gamma')$. Finally, assume that V is $\mathcal{L}'$-suitable.*

Then, Γ is unsatisfiable if and only if there is a nonempty, finite subset $\{\theta_0, \ldots, \theta_{n-1}\}$ of $\mathrm{INST}_{\mathcal{L}',V}(\Gamma')$ such that the formula $((\neg\theta_0) \vee \cdots \vee (\neg\theta_{n-1}))$ is a tautology.

Proof. The following statements are equivalent.

(1) Γ is unsatisfiable;
(2) Γ' is unsatisfiable;
(3) $\mathrm{INST}_{\mathcal{L}',V}(\Gamma')$ is unsatisfiable;
(4) there is a nonempty, finite subset $\{\theta_0, \ldots, \theta_{n-1}\}$ of $\mathrm{INST}_{\mathcal{L}',V}(\Gamma')$ that is unsatisfiable;
(5) there is a nonempty, finite subset $\{\theta_0, \ldots, \theta_{n-1}\}$ of $\mathrm{INST}_{\mathcal{L}',V}(\Gamma')$ such that $((\neg\theta_0) \vee \cdots \vee (\neg\theta_{n-1}))$ is logically valid;

(6) there is a nonempty, finite subset $\{\theta_0, \ldots, \theta_{n-1}\}$ of $\mathrm{INST}_{\mathcal{L}',V}(\Gamma')$ such that $((\neg\theta_0) \vee \cdots \vee (\neg\theta_{n-1}))$ is a tautology.

The equivalence of (1) and (2) follows from Corollary 4.9.21. By Theorem 4.10.19, (2) is equivalent to (3) and by Theorem 4.10.15, (3) is equivalent to (4). The equivalence of (4) and (5) follows from Theorem 4.5.54. Finally, since $\mathrm{INST}_{\mathcal{L}',V}(\Gamma')$ consists of quantifier-free formulas without equality, we obtain the equivalence of (5) and (6) by Theorem 4.10.13. □

Corollary 4.10.23. *Let $\mathcal{L}$ be a first-order language without equality and $\Gamma \subseteq \mathrm{SENT}_{\mathcal{L}}$. Assume further that $\mathcal{L}'$ is also a first-order language without equality and $\Gamma' \subseteq \mathrm{SENT}_{\mathcal{L}'}$ is a Skolemization of Γ. Finally, assume that $\mathcal{L}'$ contains at least one constant symbol. Then, Γ is unsatisfiable if and only if there is a nonempty, finite subset $\{\theta_0, \ldots, \theta_{n-1}\}$ of $\mathrm{GINST}_{\mathcal{L}'}(\Gamma')$ such that the formula $((\neg\theta_0) \vee \cdots \vee (\neg\theta_{n-1}))$ is a tautology.*

Proof. This statement follows from Theorem 4.10.22 by taking $V = \emptyset$. □

Corollary 4.10.24. *Let $\mathcal{L}$ be a first-order language without equality, Γ be a set of $\mathcal{L}$-formulas and φ be an $\mathcal{L}$-formula. Assume further that $\mathcal{L}'$ is also a first-order language without equality and Γ', a set of $\mathcal{L}'$-formulas, is a Skolemization of $\Gamma \cup \{(\neg\varphi)\}$, and let $V = \mathtt{FV}(\Gamma \cup \{(\neg\varphi)\}) = \mathtt{FV}(\Gamma')$. Finally, assume that V is $\mathcal{L}'$-suitable. Then, $\Gamma \models \varphi$ if and only if there is a nonempty, finite subset $\{\theta_0, \ldots, \theta_{n-1}\}$ of $\mathrm{INST}_{\mathcal{L}',V}(\Gamma')$ such that $((\neg\theta_0) \vee \cdots \vee (\neg\theta_{n-1}))$ is a tautology.*

Proof. The corollary follows from Theorem 4.10.22 and the fact that $\Gamma \models \varphi$ is equivalent to the unsatisfiability of $\Gamma \cup \{(\neg\varphi)\}$. □

Corollary 4.10.25. *Let $\mathcal{L}$ be a first-order language without equality, $\Gamma \subseteq \mathrm{SENT}_{\mathcal{L}}$ and φ be a closed $\mathcal{L}$-formula. Assume further that $\mathcal{L}'$ is also a first-order language without equality and $\Gamma' \subseteq \mathrm{SENT}_{\mathcal{L}'}$ is a Skolemization of $\Gamma \cup \{(\neg\varphi)\}$. Finally, assume that $\mathcal{L}'$ contains at least one constant symbol. Then, $\Gamma \models \varphi$ if and only if there is a nonempty, finite subset $\{\theta_0, \ldots, \theta_{n-1}\}$ of $\mathrm{GINST}_{\mathcal{L}'}(\Gamma')$ such that $((\neg\theta_0) \vee \cdots \vee (\neg\theta_{n-1}))$ is a tautology.*

Proof. The result follows from Corollary 4.10.24 by taking $V = \emptyset$. □

Corollary 4.10.26. *Let $\mathcal{L}$ be a first-order language without equality and let φ be an $\mathcal{L}$-formula. Assume further that $\mathcal{L}'$ is also a first-order language without equality and φ', an $\mathcal{L}'$-formula, is a Skolemization of $(\neg\varphi)$, and let $V = \mathtt{FV}(\neg\varphi) = \mathtt{FV}(\varphi')$. Finally, assume that V is $\mathcal{L}'$-suitable.*

Then, $\models \varphi$ if and only if there is a nonempty, finite subset $\{\theta_0, \ldots, \theta_{n-1}\}$ of $\mathrm{INST}_{\mathcal{L}',V}(\{\varphi'\})$ such that $((\neg\theta_0) \vee \cdots \vee (\neg\theta_{n-1}))$ is a tautology.

Proof. This statement follows from Corollary 4.10.24 by taking $\Gamma = \emptyset$. $\square$

Corollary 4.10.27 (Herbrand's Theorem). *Let $\mathcal{L}$ be a first-order language without equality and let φ be a closed $\mathcal{L}$-formula. Assume further that $\mathcal{L}'$ is also a first-order language without equality and φ', a closed $\mathcal{L}'$-formula, is a Skolemization of $(\neg\varphi)$. Finally, assume that $\mathcal{L}'$ contains at least one constant symbol.*

Then, $\models \varphi$ if and only if there is a nonempty, finite subset $\{\theta_0, \ldots, \theta_{n-1}\}$ of $\mathrm{GINST}_{\mathcal{L}'}(\{\varphi'\})$ such that $((\neg\theta_0) \vee \cdots \vee (\neg\theta_{n-1}))$ is a tautology.

Proof. This statement follows from Corollary 4.10.26 by taking $V = \emptyset$. $\square$

We use Corollary 4.10.27 to give a construction that allows us to show that a closed formula not containing equality is logically valid.

Procedure 4.10.28.
Input: A closed formula φ not containing equality.
Output: "Yes," if φ is logically valid. (No output is produced if φ is not logically valid.)
Method:

(A) Using Algorithm 4.9.14, find a Skolemization φ' of $(\neg\varphi)$.

(B) Let $\mathcal{L} = \mathcal{H}(\mathcal{L}_{\varphi'})$ be the Herbrand extension of the language $\mathcal{L}_{\varphi'}$ and let $\theta_0, \theta_1, \ldots$ be an effective enumeration without repetitions of $\mathrm{GINST}_{\mathcal{L}}(\{\varphi'\})$. (For instance, we could enumerate the formulas of $\mathrm{GINST}_{\mathcal{L}}(\{\varphi'\})$ in the standard ordering.) Test successively the formulas of the form $((\neg\theta_0) \vee \cdots \vee (\neg\theta_{n-1}))$ for $n = 1, 2, \ldots$ to determine if they are tautologies. Output "Yes" the first time a tautology is encountered.

Proof. We showed on page 675 how we can determine effectively if a formula is a tautology, so the method given above is effective. The correctness of the construction follows from Corollary 4.10.27 and the observation that if a disjunction is a tautology and more disjuncts are added, then the resulting formula is also a tautology. □

If the input φ of the above procedure is not logically valid, the procedure does not return "Yes." This can happen in two ways: if the set $\text{GINST}_{\mathcal{L}}(\{\varphi'\})$ is finite, then the procedure halts after testing all formulas of the form $((\neg\theta_0) \vee \cdots \vee (\neg\theta_{n-1}))$ and we know in a finite amount of time that φ is not logically valid; otherwise, that is, if $\text{GINST}_{\mathcal{L}}(\{\varphi'\})$ is infinite, the procedure never halts. The latter case usually holds because $\text{GINST}_{\mathcal{L}}(\{\varphi'\})$ is infinite whenever $\mathcal{L}$ contains a nonconstant function symbol. We have thus shown that the set of logically valid formulas is semidecidable. The semidecidability of the set of logically valid formulas is the best result we can prove since it can be shown that this set is not decidable; in fact, there are first-order languages $\mathcal{L}$ such that the set of logically valid $\mathcal{L}$-formulas is undecidable. (See, for example, Theorem 4.14.12.) However, there are also some simple languages $\mathcal{L}$ for which the set of logically valid $\mathcal{L}$-formulas is decidable.

Procedures similar to Procedure 4.10.28 can be formulated using Corollary 4.10.26 for formulas that are not necessarily closed, and Corollaries 4.10.24 and 4.10.25 when Γ is a finite set of formulas (closed or arbitrary, respectively).

We will now give some illustrations of applications of these procedures.

Example 4.10.29. Let $\varphi = ((\forall x)R(x) \to (\exists x)R(x))$. To apply Procedure 4.10.28 to φ, we need to put $(\neg\varphi)$ into Skolem normal form which necessitates putting $(\neg\varphi)$ in prenex normal form. This is accomplished by the following sequence of prenex transformations.

$$\begin{gathered}
(\neg((\forall x)R(x) \to (\exists x)R(x)))\\
(\neg((\forall x)R(x) \to (\exists y)R(y)))\\
(\neg(\exists x)(R(x) \to (\exists y)R(y)))\\
(\neg(\exists x)(\exists y)(R(x) \to R(y)))\\
(\forall x)(\forall y)(\neg(R(x) \to R(y)))
\end{gathered}$$

The last formula in this sequence is easily seen to be logically equivalent to $\varphi' = (\forall x)(\forall y)(R(x) \wedge (\neg R(y)))$, which is in Skolem normal form. Since φ' contains no constant symbols, the language $\mathcal{L}$ defined in Step (B) of the procedure is $\{c, R\}$, where c is a constant symbol. The only ground instance of φ' is $\theta_0 = (R(c) \wedge (\neg R(c)))$. Since $(\neg\theta_0)$ is a tautology, the procedure returns "Yes" when applied to φ. ◻

Example 4.10.30. Let $\varphi = (\exists x)(P(x) \to (\forall x)P(x))$. To apply Procedure 4.10.28, we put $(\neg\varphi)$ into prenex normal form by the following sequence of prenex transformations.

$$\begin{gathered}
(\neg(\exists x)(P(x) \to (\forall x)P(x))) \\
(\forall x)(\neg(P(x) \to (\forall x)P(x))) \\
(\forall x)(\neg(P(x) \to (\forall y)P(y))) \\
(\forall x)(\neg(\forall y)(P(x) \to P(y))) \\
(\forall x)(\exists y)(\neg(P(x) \to P(y)))
\end{gathered}$$

The last formula is logically equivalent to $(\forall x)(\exists y)(P(x) \wedge (\neg P(y)))$. Introducing a unary function symbol f, we obtain the Skolem normal form $\varphi' = (\forall x)(P(x) \wedge (\neg P(f(x))))$. Since φ' contains no constant symbols, the language $\mathcal{L}$ defined in Step (B) of the procedure is $\{c, f, P\}$, where c is a constant symbol. Thus, the set $\text{GINST}_{\mathcal{L}}(\{\varphi'\})$ is $\{\theta_i \mid i \geq 0\}$, where $\theta_i = (P(f^i(c)) \wedge (\neg P(f^{i+1}(c))))$. (Here we use the notation $f^i(c)$ defined inductively by $f^0(c) = c$ and $f^{i+1}(c) = f(f^i(c))$.) The formula $(\neg\theta_0)$ is not a tautology; however, $((\neg\theta_0) \vee (\neg\theta_1))$ is a tautology, as the reader can easily verify. ◻

Example 4.10.31. We consider now a case when the set of ground instances of the formula involved is slightly different in that it involves two constant symbols. Let $\varphi = ((\exists x)(\forall y)R(x, y) \to (\forall y)(\exists x)R(x, y))$. A prenex normal form of $(\neg\varphi)$ is obtained through the following steps.

$$\begin{gathered}
(\neg((\exists x)(\forall y)R(x, y) \to (\forall y)(\exists x)R(x, y))) \\
(\neg((\exists x)(\forall y)R(x, y) \to (\forall z)(\exists w)R(w, z))) \\
(\neg(\forall x)(\forall z)((\forall y)R(x, y) \to (\exists w)R(w, z))) \\
(\neg(\forall x)(\forall z)(\exists y)(\exists w)(R(x, y) \to R(w, z))) \\
(\exists x)(\exists z)(\forall y)(\forall w)(R(x, y) \wedge (\neg R(w, z)))
\end{gathered}$$

We introduce constant symbols c, d to produce the following Skolem normal form: $\varphi' = (\forall y)(\forall w)(R(c, y) \wedge (\neg R(w, d)))$. The set of ground instances of $\{\varphi'\}$ consists of the formulas

$$\begin{aligned}\theta_0 &= (R(c,c) \wedge (\neg R(c,d)))\\ \theta_1 &= (R(c,c) \wedge (\neg R(d,d)))\\ \theta_2 &= (R(c,d) \wedge (\neg R(c,d)))\\ \theta_3 &= (R(c,d) \wedge (\neg R(d,d))).\end{aligned}$$

Note that the formula $((\neg\theta_0) \vee (\neg\theta_1) \vee (\neg\theta_2))$ is a tautology because $(\neg\theta_2)$ is a tautology, so the procedure returns "Yes." ∎

Example 4.10.32. In this example, we formalize a proof that shows that every binary relation on a nonempty set that is symmetric, transitive and total is also reflexive. In other words, we show that $\Gamma \models (\forall x)R(x,x)$, where

$$\begin{aligned}\Gamma = \{&(\forall x)(\exists y)R(x,y), (\forall x)(\forall y)(R(x,y) \to R(y,x)),\\ &(\forall x)(\forall y)(\forall z)((R(x,y) \wedge R(y,z)) \to R(x,z))\}.\end{aligned}$$

We will use a procedure similar to Procedure 4.10.28 but based on Corollary 4.10.25.

A Skolemization of the set $\Gamma \cup \{(\neg\varphi)\}$, where $\varphi = (\forall x)R(x,x)$, is

$$\begin{aligned}\Gamma' = \{&(\forall x)R(x,f(x)), (\forall x)(\forall y)(R(x,y) \to R(y,x)),\\ &(\forall x)(\forall y)(\forall z)((R(x,y) \wedge R(y,z)) \to R(x,z)), (\neg R(c,c))\},\end{aligned}$$

where f is a unary function symbol and c is a constant symbol. The language $\mathcal{L}$ is $\{c, f, R\}$. The procedure would list effectively all the ground instances of $\mathrm{GINST}_{\mathcal{L}}(\Gamma')$ as $\{\theta_0, \theta_1, \ldots\}$ in the pursuit of a tautology $((\neg\theta_0) \vee \cdots \vee (\neg\theta_{n-1}))$. Rather than go through this unrewarding exercise, we observe that

$$\begin{aligned}\psi_0 &= R(c, f(c)),\\ \psi_1 &= (R(c,f(c)) \to R(f(c),c))),\\ \psi_2 &= ((R(c,f(c)) \wedge R(f(c),c)) \to R(c,c)),\\ \psi_3 &= (\neg R(c,c))\end{aligned}$$

are ground instances of Γ' such that $((\neg\psi_0) \vee (\neg\psi_1) \vee (\neg\psi_2) \vee (\neg\psi_3))$ is a tautology which means that $((\neg\theta_0) \vee \cdots \vee (\neg\theta_{n-1}))$ is a tautology for sufficiently large n. ∎

Example 4.10.33. Now, we examine the application of Procedure 4.10.28 to a formula that is not logically valid, namely, $\varphi = (\exists x)(\forall y)R(x,y)$. A Skolemization of $(\neg\varphi)$ is $\varphi' = (\forall x)(\neg R(x, f(x)))$, where f is a unary function symbol. We have $\mathcal{L} = \{c, f, R\}$, so the set $\mathrm{GINST}_{\mathcal{L}}(\{\varphi'\})$ consists of the formulas $\theta_i = (\neg R(f^i(c), f^{i+1}(c)))$ for $i \geq 0$, where $f^i(c)$ is the notation introduced in Example 4.10.30. It is easy to see that for $n \geq 1$, a fundamental form for $((\neg\theta_0) \vee \cdots \vee (\neg\theta_{n-1}))$ is $((\neg(\neg p_0)) \vee \cdots \vee (\neg(\neg p_{n-1})))$, which is not a tautology. This implies that $((\neg\theta_0) \vee \cdots \vee (\neg\theta_{n-1}))$ is never a tautology, so φ is not logically valid. The procedure itself will never return an answer because it will continue to search for a tautology. ▯

Theorem 4.10.34 (The Löwenheim[9]-Skolem Theorem for First-Order Logic without Equality). *Let $\mathcal{L}$ be a first-order language without equality and let Γ be a set of $\mathcal{L}$-formulas. Then, Γ is satisfiable if and only if Γ is satisfiable in a countable structure.*[10]

Proof. Suppose that Γ is satisfiable and let Γ' be a set of $\mathcal{L}'$-formulas that is a Skolemization of Γ. By Corollary 4.9.21, Γ' is satisfiable. Since Γ' is a set of universal formulas, by Theorem 4.10.19, Γ' is satisfiable in an $\mathtt{FV}(\Gamma')$-Herbrand structure $\mathcal{A}$; $\mathcal{A}$ has a countable universe. By Theorem 4.9.20, Γ is satisfiable in $\mathcal{A}$.

The reverse implication is immediate. □

Now that we have reduced first-order logic without equality to propositional logic, we can complete the reduction of all of first-order logic by reducing first-order logic with equality to first-order logic without equality. This will enable us to prove results for full first-order logic (with or without equality) that parallel the ones we

[9]Leopold Löwenheim was born on June 26, 1878 in Krefeld, Germany and died on May 5, 1957 in Berlin. Löwenheim studied between 1896 and 1900 at Humboldt-Universität and at the Technische Hochschule in Charlottenburg. His main contributions are in the study of algebraic aspects of logic. Löwenheim was forced to retire from his secondary school teaching position in 1934 but survived the war and resumed his teaching in 1946.

[10]Recall that for us, all first-order languages are countable sets. If this were not the case, this theorem would not be true (see Supplement 126). However, a similar result holds for languages which are not required to be countable.

obtained previously for first-order logic without equality. The following result is the key step in this reduction.

Theorem 4.10.35. *Let $\mathcal{L}$ be a first-order language that includes $=$, R be a binary relation symbol such that $R \notin \mathcal{L}$, $\mathcal{L}'$ be the first-order language $(\mathcal{L} - \{=\}) \cup \{R\}$, and Γ be a set of $\mathcal{L}$-formulas. Then, Γ is satisfiable if and only if the set of $\mathcal{L}'$-formulas $\mathsf{s}_R^{=}(\Gamma) \cup Eq_{R,\mathcal{L}'}$ is satisfiable.*

Proof. Suppose that $\mathcal{A}$ is an $\mathcal{L}$-structure and $\sigma \in \text{ASSIGN}_{\mathcal{A}}$ are such that $(\mathcal{A}, \sigma)$ satisfies Γ, that is, $(\mathcal{A}, \sigma) \models \varphi$ for every $\varphi \in \Gamma$. By Theorem 4.6.1, $(\mathcal{A}_{=\to R}, \sigma) \models \mathsf{s}_R^{=}(\varphi)$ for every $\varphi \in \Gamma$.

Observe that $R^{\mathcal{A}_{=\to R}} ==^{\mathcal{A}}$, so $R^{\mathcal{A}_{=\to R}}$ is a congruence of $\mathcal{A}_{=\to R}$. Therefore, by Theorem 4.5.63, $(\mathcal{A}_{=\to R}, \sigma) \models \text{Eq}_{R,\mathcal{L}'}$. We infer that $(\mathcal{A}_{=\to R}, \sigma)$ satisfies $\mathsf{s}_R^{=}(\Gamma) \cup \text{Eq}_{R,\mathcal{L}'}$.

Conversely, suppose that $\mathcal{A}'$ is an $\mathcal{L}'$-structure and $\sigma \in \text{ASSIGN}_{\mathcal{A}'}$ are such that $(\mathcal{A}', \sigma) \models \mathsf{s}_R^{=}(\Gamma) \cup \text{Eq}_{R,\mathcal{L}'}$. Since $\text{Eq}_{R,\mathcal{L}'}$ is a set of closed formulas, we have $\mathcal{A}' \models \text{Eq}_{R,\mathcal{L}'}$, which, by Theorem 4.5.63, implies that $R^{\mathcal{A}'}$ is a congruence of $\mathcal{A}'$. Thus, we may consider the quotient structure $\mathcal{A}'/R^{\mathcal{A}'}$ and the canonical epimorphism $h_{R^{\mathcal{A}'}}$. By the Morphism Theorem, since none of the formulas in $\mathsf{s}_R^{=}(\Gamma)$ contains $=$, we have $(\mathcal{A}'/R^{\mathcal{A}'}, h_{R^{\mathcal{A}'}} \circ \sigma) \models \mathsf{s}_R^{=}(\Gamma)$. Further, by Theorem 4.4.32, $R^{\mathcal{A}'/R^{\mathcal{A}'}}$ is the equality relation on $|\mathcal{A}'|/R^{\mathcal{A}'}$, so it makes sense to consider the $\mathcal{L}$-structure $(\mathcal{A}'/R^{\mathcal{A}'})_{R\to=}$. By Theorem 4.6.1, we have

$$((\mathcal{A}'/R^{\mathcal{A}'})_{R\to=}, h_{R^{\mathcal{A}'}} \circ \sigma) \models \mathsf{s}_{=}^{R}(\mathsf{s}_R^{=}(\Gamma)).$$

Finally, by Theorem 1.2.16, $\mathsf{s}_{=}^{R}(\mathsf{s}_R^{=}(\Gamma)) = \Gamma$, so Γ is satisfiable. $\square$

The following result records some details of the proof of Theorem 4.10.35.

Theorem 4.10.36. *Let $\mathcal{L}$ be a first-order language that includes $=$, R be a binary relation symbol such that $R \notin \mathcal{L}$, $\mathcal{L}'$ be the first-order language $(\mathcal{L} - \{=\}) \cup \{R\}$, and Γ be a set of $\mathcal{L}$-formulas. If Γ is satisfiable in an $\mathcal{L}$-structure $\mathcal{A}$, then $\mathsf{s}_R^{=}(\Gamma) \cup Eq_{R,\mathcal{L}'}$ is satisfiable in a $\mathcal{L}'$-structure whose universe is $|\mathcal{A}|$. Conversely, if $\mathsf{s}_R^{=}(\Gamma) \cup Eq_{R,\mathcal{L}'}$ is satisfiable in an $\mathcal{L}'$-structure $\mathcal{A}'$, then Γ is satisfiable in a quotient structure of some $\mathcal{L}$-structure $\mathcal{A}$ that has the same universe as $\mathcal{A}'$.*

Proof. The argument is essentially part of the proof of Theorem 4.10.35. More precisely, for the second part of the theorem, suppose that the set $\mathsf{s}^{=}_{R}(\Gamma) \cup \mathrm{Eq}_{R,\mathcal{L}'}$ is satisfiable in an $\mathcal{L}'$-structure $\mathcal{A}'$. Then, by the proof of Theorem 4.10.35 we have that Γ is satisfiable in $(\mathcal{A}'/R^{\mathcal{A}'})_{R\to=}$. Define an $\mathcal{L}$-structure $\mathcal{A}$ by $|\mathcal{A}| = |\mathcal{A}'|$, $f^{\mathcal{A}} = f^{\mathcal{A}'}$ for every function symbol $f \in \mathcal{L}$, $P^{\mathcal{A}} = P^{\mathcal{A}'}$ for every relation symbol $P \in \mathcal{L} - \{=\}$, and $=^{\mathcal{A}}$ is the equality relation on $|\mathcal{A}|$. It is easy to verify that $R^{\mathcal{A}'}$ is a congruence of $\mathcal{A}$ and that $\mathcal{A}/R^{\mathcal{A}'}$ is the same structure as $(\mathcal{A}'/R^{\mathcal{A}'})_{R\to=}$. □

Corollary 4.10.37. *Let $\mathcal{L}$ be a first-order language that includes $=$, R be a binary relation symbol such that $R \notin \mathcal{L}$, $\mathcal{L}'$ be the first-order language $(\mathcal{L} - \{=\}) \cup \{R\}$, and Γ be a set of $\mathcal{L}$-formulas. Then, Γ has a model if and only if the set of $\mathcal{L}'$-formulas $\mathsf{s}^{=}_{R}(\Gamma) \cup Eq_{R,\mathcal{L}'}$ has a model.*

Proof. We prove that the following statements are equivalent:

(1) Γ has a model;
(2) $\Gamma^{\forall}$ has a model;
(3) $\Gamma^{\forall}$ is satisfiable;
(4) $\mathsf{s}^{=}_{R}(\Gamma^{\forall}) \cup \mathrm{Eq}_{R,\mathcal{L}'}$ is satisfiable;
(5) $\mathsf{s}^{=}_{R}(\Gamma)^{\forall} \cup \mathrm{Eq}_{R,\mathcal{L}'}$ is satisfiable;
(6) $\mathsf{s}^{=}_{R}(\Gamma)^{\forall} \cup \mathrm{Eq}_{R,\mathcal{L}'}$ has a model;
(7) $\mathsf{s}^{=}_{R}(\Gamma) \cup \mathrm{Eq}_{R,\mathcal{L}'}$ has a model.

(1) is equivalent to (2) by Corollary 4.5.60. (2) is equivalent to (3) by Corollary 4.5.32. (3) is equivalent to (4) by Theorem 4.10.35. (4) is equivalent to (5) by Supplement 63. Next, (5) is equivalent to (6) by Corollary 4.5.32. Finally, (6) is equivalent to (7) by Corollary 4.5.60. □

The following examples show that parts of Theorem 4.10.19 cease to be valid for logic with equality.

Example 4.10.38. We show that the implication (1) $\longrightarrow$ (2) of Theorem 4.10.19 can fail when the language $\mathcal{L}$ contains the equality symbol, even if the formulas involved are closed. Let $\mathcal{L} = \{a, b, =\}$, where a and b are distinct constant symbols and let $\Gamma = \{a = b\}$. It is clear that Γ has a model; however, Γ has no Herbrand model, for in a Herbrand structure a and b are interpreted as themselves.

Since, in this case, $\text{GINST}_{\mathcal{L}}(\Gamma) = \Gamma$, this example also shows that the implication (3) $\longrightarrow$ (4) can also fail. $\square$

Example 4.10.39. To prove that (4) $\longrightarrow$ (2) can fail when the language $\mathcal{L}$ contains the equality symbol, consider the language $\mathcal{L}= \{a, =\}$, the set of variables $V = \{x\}$ and the set of formulas $\Gamma = \{(\forall x)(x = a), x = x\}$. Observe that $\text{INST}_{\mathcal{L},V}(\Gamma) = \{x = a, a = a, x = x\}$ is satisfiable in any V-Herbrand structure for $\mathcal{L}$ (indeed, in any $\mathcal{L}$-structure). On the other hand, Γ itself cannot be satisfied in any V-Herbrand structure $\mathcal{A}$ for $\mathcal{L}$ since $|\mathcal{A}|$ contains two elements and Γ contains the formula $(\forall x)(x = a)$. $\square$

Our next results show that some parts of Theorem 4.10.19 remain valid for logic with equality.

Theorem 4.10.40. *Let $\mathcal{L}$ be a first-order language with equality, Γ be a set of universal $\mathcal{L}$-formulas and V be an $\mathcal{L}$-suitable set of variables such that $\mathtt{FV}(\Gamma) \subseteq V$. Consider the following statements.*

(1) Γ *is satisfiable.*
(2) Γ *is satisfiable in a V-Herbrand structure for $\mathcal{L}$.*
(3) $\text{INST}_{\mathcal{L},V}(\Gamma)$ *is satisfiable.*
(4) $\text{INST}_{\mathcal{L},V}(\Gamma)$ *is satisfiable in a V-Herbrand structure for $\mathcal{L}$.*

Then, we have the following implications:

$$\begin{tikzcd}
(1) \arrow[r, leftrightarrow] & (3) \\
(2) \arrow[u] \arrow[r] & (4) \arrow[u]
\end{tikzcd}$$

Proof. The implications $(1) \longrightarrow (3), (2) \longrightarrow (1), (2) \longrightarrow (4)$ and $(4) \longrightarrow (3)$ can be shown as in Theorem 4.10.19 since the arguments do not conflict with the existence of equality in $\mathcal{L}$. Therefore, we need to prove only the implication $(3) \rightarrow (1)$.

Suppose that $\text{INST}_{\mathcal{L},V}(\Gamma)$ is satisfiable. Then, by Theorem 4.10.35, the set $\mathsf{s}_R^{=}(\text{INST}_{\mathcal{L},V}(\Gamma)) \cup \text{Eq}_{R,\mathcal{L}'}$ is satisfiable, where R is a binary relation symbol that does not occur in $\mathcal{L}$ and $\mathcal{L}' = (\mathcal{L} - \{=\}) \cup \{R\}$. By Theorem 4.6.9, this implies that the set $\mathsf{s}_R^{=}(\text{INST}_{\mathcal{L},V}(\Gamma)) \cup \text{INST}_{\mathcal{L}',V}(\text{Eq}_{R,\mathcal{L}'})$ is satisfiable. By Corollary 4.3.67, this set of formulas is the same as

$\mathrm{INST}_{\mathcal{L}',V}(\mathsf{s}_R^{=}(\Gamma)) \cup \mathrm{INST}_{\mathcal{L}',V}(\mathrm{Eq}_{R,\mathcal{L}'})$, and this set in turn equals $\mathrm{INST}_{\mathcal{L}',V}(\mathsf{s}_R^{=}(\Gamma) \cup \mathrm{Eq}_{R,\mathcal{L}'})$. By Theorem 4.10.19, we obtain the satisfiability of $\mathsf{s}_R^{=}(\Gamma) \cup \mathrm{Eq}_{R,\mathcal{L}'}$. Using again Theorem 4.10.35, but in the other direction, we obtain the satisfiability of Γ. □

Examples 4.10.38 and 4.10.39 show that no implications can be added to the previous theorem. Note that in Example 4.10.39, $V \neq \emptyset$. If $V = \emptyset$, the following stronger result holds.

Theorem 4.10.41. *Let $\mathcal{L}$ be a first-order language with equality that contains at least one constant symbol and Γ be a set of closed universal $\mathcal{L}$-formulas. Consider the following statements.*

(1) Γ *has a model.*
(2) Γ *has an $\mathcal{L}$-Herbrand model.*
(3) $\mathrm{GINST}_{\mathcal{L}}(\Gamma)$ *has a model.*
(4) $\mathrm{GINST}_{\mathcal{L}}(\Gamma)$ *has an $\mathcal{L}$-Herbrand model.*

Then, we have the following implications:

$$\begin{tikzcd}
(1) \arrow[r, leftrightarrow] & (3) \\
(2) \arrow[u] \arrow[r, leftrightarrow] & (4) \arrow[u]
\end{tikzcd}$$

Proof. All implications except (4) $\longrightarrow$ (2) were shown in Theorem 4.10.40. To prove the remaining implication, suppose that $\mathrm{GINST}_{\mathcal{L}}(\Gamma)$ has an $\mathcal{L}$-Herbrand model $\mathcal{A}$. Let $\varphi = (\forall y_0) \cdots (\forall y_{n-1})\psi$ be a formula in Γ, where ψ is a quantifier-free formula, and let $\sigma \in \mathrm{ASSIGN}_{\mathcal{A}}$. The following statements are equivalent:

(1) $(\mathcal{A}, \sigma) \models \varphi$;
(2) for all ground terms $t_0, \ldots, t_{n-1} \in |\mathcal{A}| = \mathrm{GTERM}_{\mathcal{L}}$, $(\mathcal{A}, [y_{n-1} \to t_{n-1}] \cdots [y_0 \to t_0]\sigma) \models \psi$;
(3) for all ground terms $t_0, \ldots, t_{n-1} \in |\mathcal{A}| = \mathrm{GTERM}_{\mathcal{L}}$, $(\mathcal{A}, [y_0 \to t_0] \cdots [y_{n-1} \to t_{n-1}]\sigma) \models \psi$;
(4) for all ground terms $t_0, \ldots, t_{n-1} \in |\mathcal{A}| = \mathrm{GTERM}_{\mathcal{L}}$, $(\mathcal{A}, [y_0 \to \sigma^{\mathcal{A}}(t_0)] \cdots [y_{n-1} \to \sigma^{\mathcal{A}}(t_{n-1})]\sigma) \models \psi$;
(5) for all ground terms $t_0, \ldots, t_{n-1} \in |\mathcal{A}| = \mathrm{GTERM}_{\mathcal{L}}$,

$$(\mathcal{A}, \sigma) \models (\psi)_{y_0,\ldots,y_{n-1}:=t_0,\ldots,t_{n-1}}.$$

The equivalence of (2) and (3) follows from the fact that the variables $y_0, \ldots, y_{n-1}$ are distinct. The equivalence between (3) and (4) is a consequence of Theorem 4.10.8. Finally, the equivalence of (4) and (5) is an application of Corollary 4.6.6. Since the formula $(\psi)_{y_0,\ldots,y_{n-1}:=t_0,\ldots,t_{n-1}}$ is a ground instance of φ, it follows that the fifth statement holds, hence, so does the first. Thus, $\mathcal{A}$ is a model of Γ. □

Using Theorem 4.10.35, we can extend the Compactness Theorem to all first-order languages (with or without equality).

Theorem 4.10.42 (Compactness Theorem of First-Order Logic). *Let $\mathcal{L}$ be a first-order language and let Γ be a set of $\mathcal{L}$-formulas. Then, Γ is satisfiable if and only if it is finitely satisfiable.*

Proof. If $\mathcal{L}$ does not contain $=$, the statement was already shown in Theorem 4.10.21. Suppose now that $\mathcal{L}$ contains $=$ and that R is a binary relation symbol not in $\mathcal{L}$. Let $\mathcal{L}' = (\mathcal{L} - \{=\}) \cup \{R\}$. Clearly, if Γ is satisfiable, then it is finitely satisfiable. Conversely, suppose that Γ is finitely satisfiable. Then, by Theorem 4.10.35, the set of $\mathcal{L}'$-formulas $\mathsf{s}^{=}_{R}(\Gamma_0) \cup \mathrm{Eq}_{R,\mathcal{L}'}$ is satisfiable for every finite subset Γ_0 of Γ. It follows that every finite subset of $\mathsf{s}^{=}_{R}(\Gamma) \cup \mathrm{Eq}_{R,\mathcal{L}'}$ is satisfiable. By Theorem 4.10.21, $\mathsf{s}^{=}_{R}(\Gamma) \cup \mathrm{Eq}_{R,\mathcal{L}'}$ is satisfiable and this implies that Γ is satisfiable by Theorem 4.10.35. □

We discuss now two other versions of the Compactness Theorem, both of which follow from the one above.

Theorem 4.10.43. *Let $\mathcal{L}$ be a first-order language and let Γ be a set of $\mathcal{L}$-formulas. Then, Γ has a model if and only if every finite subset of Γ has a model.*

Proof. The result is shown by the following sequence of equivalent statements.

(1) Γ has a model;
(2) $\Gamma^{\forall}$ has a model;
(3) $\Gamma^{\forall}$ is satisfiable;
(4) $\Gamma^{\forall}$ is finitely satisfiable;
(5) every finite subset of $\Gamma^{\forall}$ has a model;
(6) every finite subset of Γ has a model.

The equivalences of (1) and (2) and of (5) and (6) follow from Corollary 4.5.60. The equivalences of (2) and (3) and of (4) and (5) are consequences of Corollary 4.5.32. Finally, the equivalence of (3) and (4) follows from the Compactness Theorem 4.10.42. □

The other variant of the Compactness Theorem is given below.

Theorem 4.10.44. *Let $\mathcal{L}$ be a first-order language, Γ be a set of $\mathcal{L}$-formulas and φ be an $\mathcal{L}$-formula. Then, $\Gamma \models \varphi$ if and only if there is a finite subset Γ_0 of Γ such that $\Gamma_0 \models \varphi$.*

Proof. The argument parallels the argument for the first part of Theorem 2.4.1 using the relevant parts of Theorems 4.5.51 and 4.5.52 and the Compactness Theorem of First-Order Logic. □

The next result is a typical application of the Compactness Theorem.

Theorem 4.10.45. *Let Γ be a set of $\mathcal{L}$-formulas, where $\mathcal{L}$ is a first-order language. If Γ is satisfiable in arbitrarily large finite $\mathcal{L}$-structures, then Γ is satisfiable in an infinite $\mathcal{L}$-structure.*

Proof. Let $\mathcal{L}' = \mathcal{L} \cup \{=\}$ and, for $k \geq 1$, let φ_k be a closed $\mathcal{L}'$-formula such that an $\mathcal{L}'$-structure $\mathcal{A}$ is a model of φ_k if and only if $|\mathcal{A}| \geq k$. Let $\Gamma' = \Gamma \cup \{\varphi_k \mid k \geq 1\}$. We claim that Γ' is finitely satisfiable. Let Γ_0 be a finite subset of Γ' and let $k_0 = \max\{k \mid \varphi_k \in \Gamma_0\}$. By hypothesis, there is an $\mathcal{L}$-structure $\mathcal{A}$ with $|\mathcal{A}| \geq k_0$ and $\sigma \in \mathrm{ASSIGN}_{\mathcal{A}}$ such that $(\mathcal{A}, \sigma) \models \Gamma$. If $\mathcal{A}'$ is the unique expansion of $\mathcal{A}$ to $\mathcal{L}'$, by Theorem 4.5.24, $(\mathcal{A}', \sigma) \models \Gamma$. Since $|\mathcal{A}'| \geq k_0$, we also have $(\mathcal{A}', \sigma) \models \varphi_k$ for all k with $1 \leq k \leq k_0$. Thus, $(\mathcal{A}', \sigma) \models \Gamma_0$, which shows that Γ' is finitely satisfiable. By the Compactness Theorem, Γ' is satisfiable in some $\mathcal{L}'$-structure $\mathcal{B}'$. Note that $\mathcal{B}'$ is an infinite structure since $\mathcal{B}'$ is a model of every formula φ_k for $k \geq 1$. By Theorem 4.5.24, Γ is satisfiable in $\mathcal{B}$, the reduct of $\mathcal{B}'$ to $\mathcal{L}$. Since $|\mathcal{B}'| = |\mathcal{B}|$, $\mathcal{B}$ is an infinite structure. □

Corollary 4.10.46. *Let Γ be a set of closed $\mathcal{L}$-formulas, where $\mathcal{L}$ is a first-order language. If Γ has arbitrarily large finite models, then Γ has an infinite model.*

Proof. This follows immediately from Theorem 4.10.45. □

We can now use Corollary 4.10.46 to derive a result that is important in database theory.

A directed graph is said to be *strongly connected* if for every pair of vertices (u, v), there is a path from u to v. As we saw in Example 4.7.3, the language $\mathcal{L} = \{R, =\}$ can be used to express properties of directed graphs. In the next theorem, we will make use of the formulas $\varphi_{out}, \varphi_{in}$ introduced in this example. Also, we will make use of the notation $\mathcal{A}_G$ introduced in Example 4.4.3.

Theorem 4.10.47. *Let $\mathcal{L} = \{R, =\}$. There is no closed $\mathcal{L}$-formula φ such that for all directed graphs G, $\mathcal{A}_G \models \varphi$ if and only if G is strongly connected.*

Proof. Suppose such a formula φ exists. Define $\psi = (\varphi_{out} \wedge \varphi_{in} \wedge \varphi)$. For $n \geq 2$, let G_n be a cycle of length n (see Figure 4.4).

Since every vertex in G_n has both out-degree and in-degree 1 and every vertex is reachable from every other vertex, it follows that $\mathcal{A}_{G_n} \models \psi$. By Corollary 4.10.46, there is an infinite $\mathcal{L}$-structure $\mathcal{A}$ such that $\mathcal{A} \models \psi$. Let $G = (V, E)$ be the infinite graph such that $\mathcal{A}_G = \mathcal{A}$. Let v_0 be an arbitrary but fixed vertex of G. Because every vertex has out-degree 1, for every $n \in \mathbf{N}$, there is a unique path of length n starting at v_0. Let v_n be the vertex at the end of this path. It is clear that for each n, v_{n+1} is the unique vertex in V such that $(v_n, v_{n+1}) \in E$. The strong connectedness of G implies that $V = \{v_n \mid n \in \mathbf{N}\}$. Since every vertex has in-degree 1, there is a vertex v_j such that $(v_j, v_0) \in E$. We claim that $V = \{v_0, \ldots, v_j\}$. To justify this claim, we prove by induction on k that $v_{j+k} \in \{v_0, \ldots, v_j\}$ for all $k \in \mathbf{N}$. The basis step, $k = 0$, is obvious. Suppose $v_{j+k} \in \{v_0, \ldots, v_j\}$, that is $v_{j+k} = v_\ell$ for some ℓ with $0 \leq \ell \leq j$. If $\ell < j$, then we have

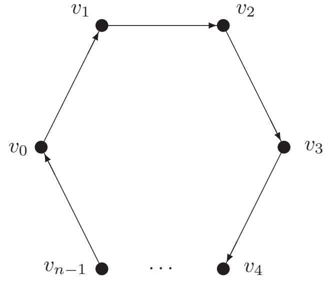


Fig. 4.4. Cycle of length n.

the edges $(v_\ell, v_{\ell+1})$ and (v_ℓ, v_{j+k+1}). Since the out-degree of v_ℓ is 1, we have $v_{j+k+1} = v_{\ell+1} \in \{v_0, \dots, v_j\}$. If $\ell = j$, we have the edges (v_ℓ, v_0) and (v_ℓ, v_{j+k+1}), which implies that $v_{j+k+1} = v_0 \in \{v_0, \dots, v_j\}$. Thus, G is a finite graph, which is a contradiction. $\square$

Corollary 4.10.48. *Let $\mathcal{L} = \{R, =\}$. There is no $\mathcal{L}$-formula θ such that for all directed graphs $G = (V, E)$, the relation that consists of all pairs $(u, v) \in V^2$ such that there is a path of length at least 1 from u to v is definable in $\mathcal{A}_G$ by θ and (x, y).*

Proof. Suppose that such a formula θ exists. Define the formula $\varphi = (\forall x)(\forall y)(x \neq y \to \theta)$. Note that $\mathcal{A}_G \models \varphi$ if and only if G is strongly connected. This is impossible by Theorem 4.10.47. $\square$

Note that if $G = (V, E)$ is a directed graph, then there is a path from u to v of positive length if and only if (u, v) belongs to the transitive closure E^* of the relation E. Since the edge relation can be an arbitrary relation on V, we can paraphrase Corollary 4.10.48 by saying that the transitive closure of a relation is not definable in first-order logic.

We now begin a development that leads to a counterpart of Herbrand's Theorem for first-order logic with equality.

Theorem 4.10.49. *Let $\mathcal{L}$ be a first-order language with equality and Γ be a set of $\mathcal{L}$-formulas. Assume further that $\mathcal{L}'$ is also a first-order language with equality and Γ', a set of $\mathcal{L}'$-formulas, is a Skolemization of Γ, and let $V = \mathtt{FV}(\Gamma) = \mathtt{FV}(\Gamma')$. Finally, assume that V is $\mathcal{L}'$-suitable.*

Then, Γ is unsatisfiable if and only if there is a nonempty, finite subset $\{\theta_0, \dots, \theta_{n-1}\}$ of $\mathrm{INST}_{\mathcal{L}',V}(\Gamma' \cup Eq_{=,\mathcal{L}'})$ such that the formula $((\neg\theta_0) \vee \cdots \vee (\neg\theta_{n-1}))$ is a tautology.

Proof. The following statements are equivalent.

(1) Γ is unsatisfiable;
(2) Γ' is unsatisfiable;
(3) $\mathsf{s}_R^{=}(\Gamma') \cup \mathrm{Eq}_{R,\mathcal{L}''}$ is unsatisfiable, where R is a binary relation symbol not in $\mathcal{L}'$ and $\mathcal{L}'' = (\mathcal{L}' - \{=\}) \cup \{R\}$;
(4) $\mathrm{INST}_{\mathcal{L}'',V}(\mathsf{s}_R^{=}(\Gamma') \cup \mathrm{Eq}_{R,\mathcal{L}''})$ is unsatisfiable;
(5) the set $\mathrm{INST}_{\mathcal{L}'',V}(\mathsf{s}_R^{=}(\Gamma') \cup \mathrm{Eq}_{R,\mathcal{L}''})$ has a nonempty, finite subset $\{\theta''_0, \dots, \theta''_{n-1}\}$ that is unsatisfiable;

(6) the set $\mathrm{INST}_{\mathcal{L}'',V}(\mathsf{s}_R^{=}(\Gamma')\cup\mathrm{Eq}_{R,\mathcal{L}''})$ has a nonempty, finite subset $\{\theta''_0,\dots,\theta''_{n-1}\}$ such that $((\neg\theta''_0)\vee\cdots\vee(\neg\theta''_{n-1}))$ is logically valid;

(7) the set $\mathrm{INST}_{\mathcal{L}'',V}(\mathsf{s}_R^{=}(\Gamma')\cup\mathrm{Eq}_{R,\mathcal{L}''})$ has a nonempty, finite subset $\{\theta''_0,\dots,\theta''_{n-1}\}$ such that $((\neg\theta''_0)\vee\cdots\vee(\neg\theta''_{n-1}))$ is a tautology;

(8) the set $\mathrm{INST}_{\mathcal{L}'',V}(\mathsf{s}_R^{=}(\Gamma')\cup\mathsf{s}_R^{=}(\mathrm{Eq}_{=,\mathcal{L}'}))$ contains a nonempty, finite subset $\{\theta''_0,\dots,\theta''_{n-1}\}$ such that $((\neg\theta''_0)\vee\cdots\vee(\neg\theta''_{n-1}))$ is a tautology;

(9) the set $\mathrm{INST}_{\mathcal{L}'',V}(\mathsf{s}_R^{=}(\Gamma'\cup\mathrm{Eq}_{=,\mathcal{L}'}))$ has a nonempty, finite subset $\{\theta''_0,\dots,\theta''_{n-1}\}$ such that $((\neg\theta''_0)\vee\cdots\vee(\neg\theta''_{n-1}))$ is a tautology;

(10) the set $\mathsf{s}_R^{=}(\mathrm{INST}_{\mathcal{L}',V}(\Gamma'\cup\mathrm{Eq}_{=,\mathcal{L}'}))$ has a nonempty, finite subset

$$\{\theta''_0,\dots,\theta''_{n-1}\}$$

such that $((\neg\theta''_0)\vee\cdots\vee(\neg\theta''_{n-1}))$ is a tautology;

(11) there is a nonempty, finite subset $\{\theta_0,\dots,\theta_{n-1}\}$ of $\mathrm{INST}_{\mathcal{L}',V}(\Gamma'\cup\mathrm{Eq}_{=,\mathcal{L}'})$ such that

$$((\neg\mathsf{s}_R^{=}(\theta_0))\vee\cdots\vee(\neg\mathsf{s}_R^{=}(\theta_{n-1})))$$

is a tautology;

(12) there is a nonempty, finite subset $\{\theta_0,\dots,\theta_{n-1}\}$ of $\mathrm{INST}_{\mathcal{L}',V}(\Gamma'\cup\mathrm{Eq}_{=,\mathcal{L}'})$ such that $\mathsf{s}_R^{=}(((\neg\theta_0)\vee\cdots\vee(\neg\theta_{n-1})))$ is a tautology;

(13) there is a nonempty, finite subset $\{\theta_0,\dots,\theta_{n-1}\}$ of $\mathrm{INST}_{\mathcal{L}',V}(\Gamma'\cup\mathrm{Eq}_{=,\mathcal{L}'})$ such that $((\neg\theta_0)\vee\cdots\vee(\neg\theta_{n-1}))$ is a tautology.

The equivalence of (1) and (2) follows from Corollary 4.9.21. By Theorem 4.10.35, (2) is equivalent to (3). By Theorem 4.10.19, observing that $\mathsf{s}_R^{=}(\Gamma')$ is a set of universal $\mathcal{L}''$-formulas with the same set of free variables as Γ', (3) is equivalent to (4) and by Theorem 4.10.15, (4) is equivalent to (5). The equivalence of (5) and (6) follows from Theorem 4.5.54. Further, since $\mathrm{INST}_{\mathcal{L}'',V}(\mathsf{s}_R^{=}(\Gamma')\cup\mathrm{Eq}_{R,\mathcal{L}''})$ consists of quantifier-free formulas without equality, we obtain the equivalence of (6) and (7) by Theorem 4.10.13. The equivalence of (7) and (8) follows from Theorem 4.5.67, while the equivalence between (8) and (9) is immediate. Corollary 4.3.67 allows us to conclude the equivalence of (9) and (10). The equivalences between (10) and (11), and (11) and (12) are immediate. Finally, the equivalence between (12) and (13) follows from Theorem 4.8.4. □

Corollary 4.10.50. *Let $\mathcal{L}$ be a first-order language with equality and $\Gamma \subseteq \mathrm{SENT}_{\mathcal{L}}$. Assume further that $\Gamma' \subseteq \mathrm{SENT}_{\mathcal{L}'}$ is a Skolemization of Γ. Finally, assume that $\mathcal{L}'$ contains at least one constant symbol. Then, Γ is unsatisfiable if and only if there is a nonempty, finite subset $\{\theta_0, \ldots, \theta_{n-1}\}$ of $\mathrm{GINST}_{\mathcal{L}'}(\Gamma' \cup Eq_{=,\mathcal{L}'})$ such that the formula $((\neg\theta_0) \vee \cdots \vee (\neg\theta_{n-1}))$ is a tautology.*

Proof. This statement follows from Theorem 4.10.49 by taking $V = \emptyset$. □

Corollary 4.10.51. *Let $\mathcal{L}$ be a first-order language with equality, Γ be a set of $\mathcal{L}$-formulas and φ be an $\mathcal{L}$-formula. Assume further that $\mathcal{L}'$ is a first-order language and Γ', a set of $\mathcal{L}'$-formulas, is a Skolemization of $\Gamma \cup \{(\neg\varphi)\}$, and let $V = \mathtt{FV}(\Gamma \cup \{(\neg\varphi)\}) = \mathtt{FV}(\Gamma')$. Finally, assume that V is $\mathcal{L}'$-suitable. We have $\Gamma \models \varphi$ if and only if there is a nonempty, finite subset $\{\theta_0, \ldots, \theta_{n-1}\}$ of $\mathrm{INST}_{\mathcal{L}',V}(\Gamma' \cup Eq_{=,\mathcal{L}'})$ such that $((\neg\theta_0) \vee \cdots \vee (\neg\theta_{n-1}))$ is a tautology.*

Proof. The corollary follows from Theorem 4.10.49 and the fact that $\Gamma \models \varphi$ is equivalent to the unsatisfiability of $\Gamma \cup \{(\neg\varphi)\}$. □

Corollary 4.10.52. *Let $\mathcal{L}$ be a first-order language with equality, $\Gamma \subseteq \mathrm{SENT}_{\mathcal{L}}$ and φ be a closed $\mathcal{L}$-formula. Assume further that $\mathcal{L}'$ is a first-order language and $\Gamma' \subseteq \mathrm{SENT}_{\mathcal{L}'}$ is a Skolemization of $\Gamma \cup \{(\neg\varphi)\}$. Finally, assume that $\mathcal{L}'$ contains at least one constant symbol. We have $\Gamma \models \varphi$ if and only if there is a nonempty, finite subset $\{\theta_0, \ldots, \theta_{n-1}\}$ of $\mathrm{GINST}_{\mathcal{L}'}(\Gamma' \cup Eq_{=,\mathcal{L}'})$ such that $((\neg\theta_0) \vee \cdots \vee (\neg\theta_{n-1}))$ is a tautology.*

Proof. The result follows from Corollary 4.10.51 by taking $V = \emptyset$. □

Corollary 4.10.53. *Let $\mathcal{L}$ be a first-order language with equality and let φ be an $\mathcal{L}$-formula. Assume further that $\mathcal{L}'$ is also a first-order language and φ', an $\mathcal{L}'$-formula, is a Skolemization of $(\neg\varphi)$, and let $V = \mathtt{FV}(\neg\varphi) = \mathtt{FV}(\varphi')$. Finally, assume that V is $\mathcal{L}'$-suitable. We have $\models \varphi$ if and only if there is a nonempty, finite subset $\{\theta_0, \ldots, \theta_{n-1}\}$ of $\mathrm{INST}_{\mathcal{L}',V}(\{\varphi'\} \cup Eq_{=,\mathcal{L}'})$ such that $((\neg\theta_0) \vee \cdots \vee (\neg\theta_{n-1}))$ is a tautology.*

Proof. This statement follows from Corollary 4.10.51 by taking $\Gamma = \emptyset$. □

Corollary 4.10.54 (Herbrand's Theorem for First-Order Logic with Equality). *Let $\mathcal{L}$ be a first-order language with equality and let φ be a closed $\mathcal{L}$-formula. Assume further that $\mathcal{L}'$ is also a first-order language and φ', a closed $\mathcal{L}'$-formula, is a Skolemization of $(\neg\varphi)$. Finally, assume that $\mathcal{L}'$ contains at least one constant symbol. We have $\models \varphi$ if and only if there is a nonempty, finite subset $\{\theta_0, \ldots, \theta_{n-1}\}$ of $\mathrm{GINST}_{\mathcal{L}'}(\{\varphi'\} \cup Eq_{=,\mathcal{L}'})$ such that $((\neg\theta_0) \vee \cdots \vee (\neg\theta_{n-1}))$ is a tautology.*

Proof. This statement follows from Corollary 4.10.53 by taking $V = \emptyset$. □

We use Corollary 4.10.54 to give a procedure that allows us to show that a formula containing equality is logically valid. This procedure is similar to Procedure 4.10.28 used for formulas without equality.

Procedure 4.10.55.
Input: A closed formula φ containing equality.
Output: "Yes," if φ is logically valid. (No output is produced if φ is not logically valid.)
Method:

(A) Using Algorithm 4.9.14, find a Skolemization φ' of $(\neg\varphi)$.
(B) Let $\mathcal{L} = \mathcal{H}(\mathcal{L}_{\varphi'})$ be the Herbrand extension of the language $\mathcal{L}_{\varphi'}$ and let $\theta_0, \theta_1, \ldots$ be an effective enumeration without repetitions of $\mathrm{GINST}_{\mathcal{L}}(\{\varphi'\} \cup \mathrm{Eq}_{=,\mathcal{L}})$. (For instance, we could enumerate the formulas of $\mathrm{GINST}_{\mathcal{L}}(\{\varphi'\} \cup \mathrm{Eq}_{=,\mathcal{L}})$ in the standard ordering.) Test successively the formulas of the form $((\neg\theta_0) \vee \cdots \vee (\neg\theta_{n-1}))$ for $n = 1, 2, \ldots$ to determine if they are tautologies. Output "Yes" the first time a tautology is encountered.

Proof. The argument is entirely similar to the one for Procedure 4.10.28, except that here we make use of Corollary 4.10.54. □

Example 4.10.56. Consider the formula φ given by

$$(\forall x)(\forall y)(((\forall z)(f(x,z) = z) \wedge (\forall w)(f(w,y) = w)) \to (x = y)),$$

where f is a binary function symbol. The formula ψ given by

$$(\exists x)(\exists y)(\forall z)(\forall w)((f(x,z) = z) \wedge (f(w,y) = w) \wedge (x \neq y))$$

is logically equivalent to $(\neg\varphi)$. A Skolemization of $(\neg\varphi)$ is thus the following formula φ':

$$(\forall z)(\forall w)((f(e,z) = z) \land (f(w,e') = w) \land (e \neq e')),$$

where e, e' are two constant symbols. Let θ_0 be the ground instance

$$((f(e,e') = e') \land (f(e,e') = e) \land (e \neq e'))$$

of φ' and let θ_1, θ_2 be the following two ground instances of formulas in $\mathrm{Eq}_{=,\mathcal{L}_{\varphi'}}$:

$$((f(e,e') = e) \to (e = f(e,e')))$$
$$(((e = f(e,e')) \land (f(e,e') = e')) \to (e = e')).$$

We leave to the reader the verification that $((\neg\theta_0) \lor (\neg\theta_1) \lor (\neg\theta_2))$ is a tautology and hence, φ is logically valid. ∎

The Löwenheim-Skolem Theorem can also be extended to arbitrary first-order languages using Theorem 4.10.36.

Theorem 4.10.57 (The Löwenheim-Skolem Theorem). *Let $\mathcal{L}$ be a first-order language and let Γ be a set of $\mathcal{L}$-formulas. Then, Γ is satisfiable if and only if Γ is satisfiable in a countable structure.*[11]

Proof. If $\mathcal{L}$ does not contain $=$, the statement was already shown in Theorem 4.10.34. Suppose now that $\mathcal{L}$ contains $=$ and that R is a binary relation symbol not in $\mathcal{L}$. Let $\mathcal{L}' = (\mathcal{L} - \{=\}) \cup \{R\}$. Suppose that Γ is satisfiable. Then, by Theorem 4.10.35, the set of $\mathcal{L}'$-formulas $\mathsf{s}^{=}_{R}(\Gamma) \cup \mathrm{Eq}_{R,\mathcal{L}'}$ is also satisfiable, so it is satisfiable in an $\mathcal{L}'$-structure $\mathcal{A}'$ that has a countable universe, by Theorem 4.10.34. Therefore, by Theorem 4.10.36, the set Γ is satisfiable in a quotient $\mathcal{C}$ of an $\mathcal{L}$-structure $\mathcal{B}$ that has the same universe as $\mathcal{A}'$. Since $|\mathcal{B}|$ is countable and the canonical morphism from $\mathcal{B}$ to $\mathcal{C}$ is a surjection between $|\mathcal{B}|$ and $|\mathcal{C}|$, it follows (from a result in Section 5.3 of [14]) that $|\mathcal{C}|$ is countable. □

Another variant of the Löwenheim-Skolem Theorem is given next.

Theorem 4.10.58. *Let $\mathcal{L}$ be a first-order language and let Γ be a set of $\mathcal{L}$-formulas. Then, Γ has a model if and only if Γ has a countable model.*

[11] See footnote 10.

Proof. The following statements are equivalent.

(1) Γ has a model;
(2) $\Gamma^{\forall}$ has a model;
(3) $\Gamma^{\forall}$ is satisfiable;
(4) $\Gamma^{\forall}$ is satisfiable in a countable structure;
(5) $\Gamma^{\forall}$ has a countable model;
(6) Γ has a countable model.

The equivalences of (1) and (2) and of (5) and (6) follow from Corollary 4.5.60. The equivalences of (2) and (3) and of (4) and (5) are consequences of Corollary 4.5.32. Finally, the equivalence of (3) and (4) follows from Theorem 4.10.57. $\square$

4.11 Brand's Modification Method

Corollary 4.10.37 provides one way to handle the special status of the equality symbol, namely, treating this symbol as a regular relation symbol and adding the congruence axioms for that symbol. In this section, we discuss another approach, due to Daniel Brand [6], that modifies quantifier-free formulas with equality in such a way that the original formula has a model with equality given special treatment if and only if the modified formula has a model where equality is considered as an arbitrary relation symbol. Thus, no special axioms are needed with the modified formula. In view of the results of the previous section, this modification for quantifier-free formulas suffices for the treatment of the equality symbol.

Definition 4.11.1. Let R be a binary relation symbol. A formula φ is *R-flat* if every atomic subformula of φ has one of the following forms:

(1) $P(v_0, \ldots, v_{n-1})$, where P is an n-ary relation symbol with $n \geq 1$ such that $P \neq R$ and $v_0, \ldots, v_{n-1}$ are variables;
(2) P, where P is a propositional constant;
(3) $R(t_0, t_1)$, where each of t_0, t_1 is a variable, a constant symbol, or a term of the form $f(v_0, \ldots, v_{m-1})$ where $v_0, \ldots, v_{m-1}$ are variables.

A set Γ of formulas is *R-flat* if each member of Γ is an R-flat formula. $\square$

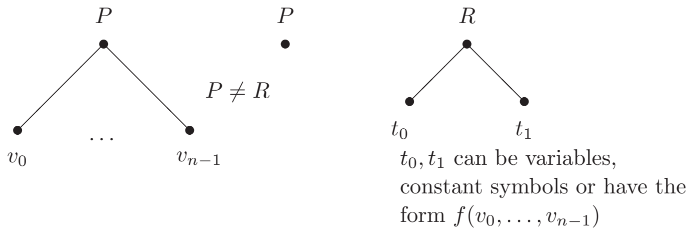

P
v_0
. . .
v_{n-1}
$P \neq R$
P
R
t_0
t_1
t_0, t_1 can be variables, constant symbols or have the form $f(v_0, \ldots, v_{n-1})$

Fig. 4.5. Atomic subformulas of an R-flat formula.

The atomic subformulas of an R-flat formula are shown in Figure 4.5.

Example 4.11.2. Let R be a binary relation symbol and P be a unary relation symbol. The set of formulas

$$\Gamma = \{R(a,b), (P(x) \vee (\neg R(x,a))), ((\neg P(y)) \vee (\neg R(y,b)))\}$$

is an R-flat set of formulas. ▯

Theorem 4.11.3. *Let $\mathcal{L}$ be a first-order language, R be a binary relation symbol in $\mathcal{L}$ and let Γ be a set of R-flat quantifier-free $\mathcal{L}$-formulas. Then, the following three statements are equivalent:*

(1) *Γ has a model $\mathcal{A}$ where $\mathcal{A}$ is an $\mathcal{L}$-structure and $R^{\mathcal{A}}$ is the equality relation on $|\mathcal{A}|$;*
(2) *Γ has a model $\mathcal{B}$ where $\mathcal{B}$ is an $\mathcal{L}$-structure and $R^{\mathcal{B}}$ is a congruence on $|\mathcal{B}|$;*
(3) *Γ has a model $\mathcal{C}$ where $\mathcal{C}$ is an $\mathcal{L}$-structure and $R^{\mathcal{C}}$ is an equivalence relation on $|\mathcal{C}|$.*

Proof. The implications (1) $\longrightarrow$ (2) and (2) $\longrightarrow$ (3) are immediate. Thus, we need to prove only (3) $\longrightarrow$ (1).

Suppose that Γ has a model $\mathcal{C} = (C, \mathcal{J})$ where $R^{\mathcal{C}}$ is an equivalence relation. Define the $\mathcal{L}$-structure $\mathcal{A} = (|C|/R^{\mathcal{C}}, \mathcal{I})$ as follows. By the axiom of choice, there exists a choice function g for the collection of equivalence classes $\{[c]_{R^{\mathcal{C}}} \mid c \in C\}$. In other words, $g(E)$ is a fixed representative of the equivalence class E of $R^{\mathcal{C}}$. The interpretation of the structure $\mathcal{A}$ is given by:

(1) if f is an n-ary function symbol in $\mathcal{L}$, then

$$f^{\mathcal{A}}(E_0,\ldots,E_{n-1}) = [f^{\mathcal{C}}(g(E_0),\ldots,g(E_{n-1}))]_{R^{\mathcal{C}}},$$

for $E_0,\ldots,E_{n-1} \in |\mathcal{A}|$;

(2) if P is an n-ary relation symbol in $\mathcal{L}$, then

$$P^{\mathcal{A}} = \{(E_0,\ldots,E_{n-1}) \mid (g(E_0),\ldots,g(E_{n-1})) \in P^{\mathcal{C}}\}.$$

If $= \in \mathcal{L}$, then for $=^{\mathcal{A}}$ we have:

$$\begin{aligned}
=^{\mathcal{A}} &= \{(E_0,E_1) \mid (g(E_0),g(E_1)) \in =^{\mathcal{C}}\} \\
&= \{(E_0,E_1) \mid g(E_0) = g(E_1)\} \\
&= \{(E_0,E_1) \mid E_0 = E_1\},
\end{aligned}$$

so $=^{\mathcal{A}}$ is indeed the equality relation on $|\mathcal{A}|$.

Note that if c is a constant symbol in $\mathcal{L}$, $c^{\mathcal{A}} = [c^{\mathcal{C}}]_{R^{\mathcal{C}}}$ and if P is a propositional constant in $\mathcal{L}$, $P^{\mathcal{A}} = P^{\mathcal{C}}$.

We show now that $R^{\mathcal{A}}$ is the equality relation on $|\mathcal{A}|$. We have:

$$\begin{aligned}
R^{\mathcal{A}} &= \{(E_0,E_1) \mid (g(E_0),g(E_1)) \in R^{\mathcal{C}}\} \\
&= \{(E_0,E_1) \mid E_0 = E_1\}.
\end{aligned}$$

We will show now that $\mathcal{A}$ is a model of Γ. For $\sigma \in \mathrm{ASSIGN}_{\mathcal{A}}$ we define $\tilde{\sigma} \in \mathrm{ASSIGN}_{\mathcal{C}}$ by $\tilde{\sigma}(x) = g(\sigma(x))$. We shall prove that for every quantifier-free R-flat formula φ and $\sigma \in \mathrm{ASSIGN}_{\mathcal{A}}$, we have $(\mathcal{A},\sigma) \models \varphi$ if and only if $(\mathcal{C},\tilde{\sigma}) \models \varphi$. This would imply the desired conclusion because Γ consists of quantifier-free R-flat formulas and $\mathcal{C}$ is a model of Γ.

We prove the claim by induction on φ.

For the basis step, suppose that φ is an atomic formula. We need to consider two subcases. If φ has the form $\varphi = P(v_0,\ldots,v_{n-1})$ with $n \geq 1$, or $\varphi = P$, where $P \neq R$, the following are equivalent:

(1) $(\mathcal{A},\sigma) \models \varphi$;
(2) $(\sigma^{\mathcal{A}}(v_0),\ldots,\sigma^{\mathcal{A}}(v_{n-1})) \in P^{\mathcal{A}}$;
(3) $(\sigma(v_0),\ldots,\sigma(v_{n-1})) \in P^{\mathcal{A}}$;
(4) $(g(\sigma(v_0)),\ldots,g(\sigma(v_{n-1}))) \in P^{\mathcal{C}}$;

(5) $(\tilde{\sigma}(v_0), \ldots, \tilde{\sigma}(v_{n-1})) \in P^{\mathcal{C}}$;
(6) $(\tilde{\sigma}^{\mathcal{C}}(v_0), \ldots, \tilde{\sigma}^{\mathcal{C}}(v_{n-1})) \in P^{\mathcal{C}}$;
(7) $(\mathcal{C}, \tilde{\sigma}) \models \varphi$.

For the remaining basis step, when $\varphi = R(t_0, t_1)$, we need to prove first that if the term t has one of the forms v, c, or $f(v_0, \ldots, v_{n-1})$, where $v, v_0, \ldots, v_{n-1}$ are variables and c is a constant symbol, then we have $(g(\sigma^{\mathcal{A}}(t)), \tilde{\sigma}^{\mathcal{C}}(t)) \in R^{\mathcal{C}}$. In the case of variables, we have $g(\sigma^{\mathcal{A}}(v)) = g(\sigma(v)) = \tilde{\sigma}(v) = \tilde{\sigma}^{\mathcal{C}}(v)$, so since $R^{\mathcal{C}}$ is reflexive, $(g(\sigma^{\mathcal{A}}(v)), \tilde{\sigma}^{\mathcal{C}}(v)) \in R^{\mathcal{C}}$. If c is a constant symbol of $\mathcal{L}$, then $g(\sigma^{\mathcal{A}}(c)) = g([c^{\mathcal{C}}]_{R^{\mathcal{C}}}) \equiv_{R^{\mathcal{C}}} c^{\mathcal{C}} = \tilde{\sigma}^{\mathcal{C}}(c)$.

For terms $f(v_0, \ldots, v_{n-1})$, we can write

$$\begin{aligned}
g(\sigma^{\mathcal{A}}(f(v_0, \ldots, v_{n-1}))) &= g(f^{\mathcal{A}}(\sigma^{\mathcal{A}}(v_0), \ldots, \sigma^{\mathcal{A}}(v_{n-1}))) \\
&= g(f^{\mathcal{A}}(\sigma(v_0), \ldots, \sigma(v_{n-1}))) \\
&= g([f^{\mathcal{C}}(g(\sigma(v_0)), \ldots, g(\sigma(v_{n-1})))]_{R^{\mathcal{C}}}) \\
&= g([f^{\mathcal{C}}(\tilde{\sigma}(v_0), \ldots, \tilde{\sigma}(v_{n-1}))]_{R^{\mathcal{C}}}) \\
&\equiv_{R^{\mathcal{C}}} f^{\mathcal{C}}(\tilde{\sigma}(v_0), \ldots, \tilde{\sigma}(v_{n-1})) \\
&= \tilde{\sigma}^{\mathcal{C}}(f(v_0, \ldots, v_{n-1})).
\end{aligned}$$

Thus, the following statements involving the formula $\varphi = R(t_0, t_1)$, where t_0, t_1 each have one of the forms v, c, or $f(v_0, \ldots, v_{n-1})$, are equivalent.

(1) $(\mathcal{A}, \sigma) \models \varphi$;
(2) $(\sigma^{\mathcal{A}}(t_0), \sigma^{\mathcal{A}}(t_1)) \in R^{\mathcal{A}}$;
(3) $(g(\sigma^{\mathcal{A}}(t_0)), g(\sigma^{\mathcal{A}}(t_1))) \in R^{\mathcal{C}}$;
(4) $(\tilde{\sigma}^{\mathcal{C}}(t_0), \tilde{\sigma}^{\mathcal{C}}(t_1)) \in R^{\mathcal{C}}$, (because $g(\sigma^{\mathcal{A}}(t_i)) \equiv_{R^{\mathcal{C}}} \tilde{\sigma}^{\mathcal{C}}(t_i))$);
(5) $(\mathcal{C}, \tilde{\sigma}) \models R(t_0, t_1)$;
(6) $(\mathcal{C}, \tilde{\sigma}) \models \varphi$.

The arguments for the inductive steps are immediate. □

Example 4.11.4. The set of R-flat formulas Γ given in Example 4.11.2,

$$\Gamma = \{R(a, b), (P(x) \vee (\neg R(x, a))), ((\neg P(y)) \vee (\neg R(y, b)))\}$$

has no model $\mathcal{A}$ in which $R^{\mathcal{A}}$ is a congruence. Indeed, suppose that $\mathcal{A}$ is a model of Γ where $R^{\mathcal{A}}$ is a congruence. Then, $(a^{\mathcal{A}}, b^{\mathcal{A}}) \in R^{\mathcal{A}}$.

By reflexivity, $(a^{\mathcal{A}}, a^{\mathcal{A}}) \in R^{\mathcal{A}}$. Since $\mathcal{A}$ is a model of $(P(x) \vee (\neg R(x, a)))$, it follows that $a^{\mathcal{A}} \in P^{\mathcal{A}}$. Similarly, we have $b^{\mathcal{A}} \notin P^{\mathcal{A}}$, which contradicts the assumption that $R^{\mathcal{A}}$ is a congruence.

By Theorem 4.11.3, it follows that Γ has no model $\mathcal{A}$ where $R^{\mathcal{A}}$ is an equivalence relation. It is instructive to provide a direct argument for this fact without using the theorem. Indeed, suppose that $\mathcal{A}$ is a model of Γ where $R^{\mathcal{A}}$ is an equivalence relation. By reflexivity, we have $(a^{\mathcal{A}}, a^{\mathcal{A}}) \in R^{\mathcal{A}}$. As above, this implies that $a^{\mathcal{A}} \in P^{\mathcal{A}}$. Since $(a^{\mathcal{A}}, b^{\mathcal{A}}) \in R^{\mathcal{A}}$, it follows that $a^{\mathcal{A}} \notin P^{\mathcal{A}}$, because $\mathcal{A} \models ((\neg P(y)) \vee (\neg R(y, b)))$. This shows that $\mathcal{A}$ cannot exist.

However, it is possible to show that Γ has a model. $\square$

Lemma 4.11.5. *Let $\mathcal{L}$ be a first-order language and let R be a binary relation symbol in $\mathcal{L}$. Define the function $F : \mathrm{TERM}_{\mathcal{L}} \cup \mathrm{FORM}_{\mathcal{L}} \to \mathbf{N}$ by*

(1) $F(x) = 0$ *for every variable x;*
(2) $F(c) = 0$ *for every constant symbol c in $\mathcal{L}$ and $F(P) = 0$ for every propositional constant P in $\mathcal{L}$;*
(3) $F(f(t_0, \ldots, t_{n-1})) = \left(\sum_{i=0}^{n-1} F(t_i)\right) + |\{i \mid t_i \text{ is not a variable}\}|$ *when f is an n-ary function symbol with $n > 0$;*
(4) $F(P(t_0, \ldots, t_{n-1})) = \left(\sum_{i=0}^{n-1} F(t_i)\right) + |\{i \mid t_i \text{ is not a variable}\}|$ *when P is an n-ary relation symbol of $\mathcal{L} - \{R\}$ with $n > 0$;*
(5) $F(R(t_0, t_1)) = F(t_0) + F(t_1)$*;*
(6) $F(\varphi) = \sum\{n_\alpha F(\alpha) \mid \alpha \text{ is an atomic formula that occurs in } \varphi\}$, *for a nonatomic $\mathcal{L}$-formula φ, where n_α is the number of occurrences of α in φ.*

F is extended to sets of $\mathcal{L}$-formulas Γ by $F(\Gamma) = \sum\{F(\varphi) \mid \varphi \in \Gamma\}$. Note that this sum could be infinite if Γ is infinite.

For all sets of $\mathcal{L}$-formulas Γ, we have $F(\Gamma) \geq 0$ and $F(\Gamma) = 0$ if and only if Γ is an R-flat set of formulas.

Proof. It is clear that $F(t) \geq 0$ for every $\mathcal{L}$-term t and $F(\varphi) \geq 0$ for every $\mathcal{L}$-formula φ, as can be easily verified by induction on terms and formulas, respectively. This implies that $F(\Gamma) \geq 0$ for every set of $\mathcal{L}$-formulas Γ.

To prove the second assertion, we note that for all $\mathcal{L}$-terms t, $F(t) = 0$ if and only if t has one of the forms x, where x is a variable, c, where c is a constant symbol, or $f(x_0, \ldots, x_{n-1})$, where f is an n-ary function symbol, $n > 0$, and $x_0, \ldots, x_{n-1}$ are variables.

Suppose that Γ is a set of R-flat $\mathcal{L}$-formulas. To show that $F(\Gamma) = 0$, it suffices to show that $F(\varphi) = 0$ for every R-flat $\mathcal{L}$-formula. By Part 6 of the definition of F, it is enough to prove that $F(\psi) = 0$ for every R-flat atomic formula. The following three cases need to be considered:

(1) if $\psi = P$ where P is a propositional constant, then $F(\psi) = 0$ by Part 2 of the definition of F;
(2) if $\psi = P(x_0, \ldots, x_{n-1})$, where P is an n-ary relation symbol of $\mathcal{L} - \{R\}$, $n > 0$, and $x_0, \ldots, x_{n-1}$ are variables, then $F(\psi) = 0$ by Part 4 of the definition of F;
(3) if $\psi = R(t_0, t_1)$, where t_0, t_1 are variables, constant symbols, or have the form $f(x_0, \ldots, x_{n-1})$, then $F(\psi) = F(t_0) + F(t_1) = 0$, as we established before.

Suppose now that Γ is not an R-flat set of formulas. Then, there is a non-R-flat formula φ in Γ and it suffices to show that $F(\varphi) > 0$. Since φ is not R-flat, there is an atomic subformula α of one of the forms $P(t_0, \ldots, t_{n-1})$, where $P \neq R$ and at least one of the terms t_i is not a variable, or $R(t_0, t_1)$, where at least one of t_0, t_1 has the form $f(u_0, \ldots, u_{p-1})$ such that at least one of the terms u_j is not a variable. If $\alpha = P(t_0, \ldots, t_{n-1})$, then by Part 4 of the definition of F, $F(\alpha) > 0$, which implies $F(\varphi) > 0$. If $\alpha = R(t_0, t_1)$, we have $F(\alpha) = F(t_0) + F(t_1) > 0$, which again implies $F(\varphi) > 0$. □

Next, we will give an algorithm which starts with a finite set of quantifier-free $\mathcal{L}$-formulas Γ and produces a set of R-flat quantifier-free $\mathcal{L}$-formulas Γ' such that Γ has a model $\mathcal{A}$ such that $R^{\mathcal{A}}$ is a congruence if and only if Γ' has a model $\mathcal{A}$ such that $R^{\mathcal{A}}$ is a congruence. In fact, if $\mathcal{A}$ is an $\mathcal{L}$-structure such that $R^{\mathcal{A}}$ is a congruence, then $\mathcal{A} \models \Gamma$ if and only if $\mathcal{A} \models \Gamma'$. To prove the correctness of this algorithm, we need the following preliminary results.

Lemma 4.11.6. *Let $\mathcal{L}$ be a first-order language, φ be a quantifier-free $\mathcal{L}$-formula and let (α, j) be an occurrence of the formula α in φ. Suppose that $\mathcal{A}$ is an $\mathcal{L}$-structure, $\sigma \in \mathrm{ASSIGN}_{\mathcal{A}}$, $\varphi' =$* `replace`$(\varphi, (\alpha, j), \beta)$*, where β is an $\mathcal{L}$-formula and $(\mathcal{A}, \sigma) \models \alpha$ if and only if $(\mathcal{A}, \sigma) \models \beta$. Then, $(\mathcal{A}, \sigma) \models \varphi$ if and only if $(\mathcal{A}, \sigma) \models \varphi'$.*

Proof. We begin by observing that in the special case when $\alpha = \varphi$, we have $\varphi' = \beta$ and the result is immediate.

The argument is by induction on φ. The basis step, when φ is atomic, falls under the previous special case. We consider only one of the inductive steps, namely, when $\varphi = (\neg\psi)$ and the result holds for ψ. If $\alpha = \varphi$, we are in the special case. Otherwise, $(\alpha, j-2)$ is the occurrence in ψ that corresponds to the occurrence (α, j) in φ. The lemma follows from the equivalence of the following statements:

(1) $(\mathcal{A}, \sigma) \models \texttt{replace}\,(\varphi, (\alpha, j), \beta) = (\neg\texttt{replace}\,(\psi, (\alpha, j-2), \beta)))$;
(2) $(\mathcal{A}, \sigma) \not\models \texttt{replace}\,(\psi, (\alpha, j-2), \beta)$;
(3) $(\mathcal{A}, \sigma) \not\models \psi$;
(4) $(\mathcal{A}, \sigma) \models (\neg\psi) = \varphi$.

□

Lemma 4.11.7. *Let $\mathcal{L}$ be a first-order language and let R be a binary relation symbol in $\mathcal{L}$. Suppose (α, j) is an occurrence of an atomic formula $\alpha = P(t_0, \ldots, t_i, \ldots, t_{n-1})$ in a quantifier-free $\mathcal{L}$-formula φ, P is not the equality symbol, w is a variable that does not occur in φ, and*

$$\begin{aligned}\varphi' = (&\texttt{replace}\,(\varphi, (\alpha, j), P(t_0, \ldots, t_{i-1}, w, t_{i+1}, \ldots, t_{n-1}))\\ &\vee(\neg R(w, t_i))).\end{aligned}$$

If $\mathcal{A}$ is an $\mathcal{L}$-structure such that $R^{\mathcal{A}}$ is a congruence, then $\mathcal{A} \models \varphi$ if and only if $\mathcal{A} \models \varphi'$.

Proof. Let $\mathcal{A}$ be an $\mathcal{L}$-structure such that $R^{\mathcal{A}}$ is a congruence. We begin by proving that if $\mathcal{A} \models \varphi$, then $\mathcal{A} \models \varphi'$. Suppose that $\sigma \in \text{ASSIGN}_{\mathcal{A}}$. We have $(\mathcal{A}, \sigma) \models \varphi$. If $(\mathcal{A}, \sigma) \not\models R(w, t_i)$, then it is clear that $(\mathcal{A}, \sigma) \models \varphi'$. Otherwise, $(\mathcal{A}, \sigma) \models R(w, t_i)$, which implies $(\sigma^{\mathcal{A}}(w), \sigma^{\mathcal{A}}(t_i)) \in R^{\mathcal{A}}$. Since $R^{\mathcal{A}}$ is a congruence and P is not the equality symbol, it follows that $(\sigma^{\mathcal{A}}(t_0), \ldots, \sigma^{\mathcal{A}}(t_i), \ldots, \sigma^{\mathcal{A}}(t_{n-1})) \in P^{\mathcal{A}}$ if and only if

$$(\sigma^{\mathcal{A}}(t_0), \ldots, \sigma^{\mathcal{A}}(w), \ldots, \sigma^{\mathcal{A}}(t_{n-1})) \in P^{\mathcal{A}}.$$

Thus, $(\mathcal{A}, \sigma) \models P(t_0, \ldots, t_{n-1})$ if and only if

$$(\mathcal{A}, \sigma) \models P(t_0, \ldots, w, \ldots, t_{n-1}).$$

By Lemma 4.11.6, we have $(\mathcal{A}, \sigma) \models \varphi$ if and only if $(\mathcal{A}, \sigma) \models \varphi'$, which allows us to conclude that $(\mathcal{A}, \sigma) \models \varphi'$. Since for every σ one of the above cases holds, it follow that $\mathcal{A} \models \varphi'$.

Conversely, suppose that $\mathcal{A} \models \varphi'$ and $\sigma \in \mathrm{ASSIGN}_{\mathcal{A}}$. Define $\sigma' = [w \to \sigma^{\mathcal{A}}(t_i)]\sigma$. Since $(\mathcal{A}, \sigma') \models \varphi'$, we have either $(\mathcal{A}, \sigma') \not\models R(w, t_i)$ or $(\mathcal{A}, \sigma') \models \texttt{replace}\,(\varphi, (\alpha, j), P(t_0, \dots, w, \dots, t_{n-1}))$. Note that $(\sigma')^{\mathcal{A}}(t_i) = \sigma^{\mathcal{A}}(t_i)$, because w does not occur in t_i. Furthermore, by the definition of σ', we have $(\sigma')^{\mathcal{A}}(t_i) = \sigma^{\mathcal{A}}(t_i) = \sigma'(w)$. Since $R^{\mathcal{A}}$ is a congruence, it follows that $(\sigma'(w), (\sigma')^{\mathcal{A}}(t_i)) \in R^{\mathcal{A}}$ which means that $(\mathcal{A}, \sigma') \models R(w, t_i)$, so we must have

$$(\mathcal{A}, \sigma') \models \texttt{replace}\,(\varphi, (\alpha, j), P(t_0, \dots, w, \dots, t_{n-1})).$$

Since $(\sigma')^{\mathcal{A}}(t_i) = \sigma'(w)$, we have $(\mathcal{A}, \sigma') \models \alpha$ if and only if $(\mathcal{A}, \sigma') \models P(t_0, \dots, w, \dots, t_{n-1})$. Therefore, by Lemma 4.11.6, we have $(\mathcal{A}, \sigma') \models \varphi$ if and only if

$$(\mathcal{A}, \sigma') \models \texttt{replace}\,(\varphi, (\alpha, j), P(t_0, \dots, w, \dots, t_{n-1})),$$

which allows us to write $(\mathcal{A}, \sigma') \models \varphi$. Since w does not occur in φ, it follows that $(\mathcal{A}, \sigma) \models \varphi$. Since σ was arbitrary, we have shown that $\mathcal{A} \models \varphi$. □

Lemma 4.11.8. *Let $\mathcal{L}$ be a first-order language and R be a binary relation symbol in $\mathcal{L}$.*

Suppose (α, j) is an occurrence of an atomic formula $\alpha = R(t_0, t_1)$ in a quantifier-free $\mathcal{L}$-formula φ and one of the following cases holds:

(1) $t_0 = f(u_0, \dots, u_k, \dots, u_{m-1})$*, u_k is not a variable, w is a variable that does not occur in φ, and*

$$\varphi' = (\texttt{replace}\,(\varphi, (\alpha, j), R(f(u_0, \dots, u_{k-1}, w, u_{k+1}, \dots, u_{m-1}), t_1)) \vee (\neg R(w, u_k))).$$

(2) $t_1 = f(u_0, \dots, u_k, \dots, u_{m-1})$*, u_k is not a variable, w is a variable that does not occur in φ, and*

$$\varphi' = (\texttt{replace}\,(\varphi, (\alpha, j), R)(t_0, f(u_0, \dots, u_{k-1}, w, u_{k+1}, \dots, u_{m-1}))) \vee (\neg R(w, u_k))).$$

If $\mathcal{A}$ is an $\mathcal{L}$-structure such that $R^{\mathcal{A}}$ is a congruence, then $\mathcal{A} \models \varphi$ if and only if $\mathcal{A} \models \varphi'$.

Proof. Without loss of generality, we discuss only the first case because the treatment of the second case is entirely similar.

Let $\mathcal{A}$ be an $\mathcal{L}$-structure such that $R^{\mathcal{A}}$ is a congruence. We begin by proving that if $\mathcal{A} \models \varphi$, then $\mathcal{A} \models \varphi'$. Suppose that $\sigma \in \text{ASSIGN}_{\mathcal{A}}$. We have $(\mathcal{A}, \sigma) \models \varphi$. If $(\mathcal{A}, \sigma) \not\models R(w, u_k)$, then it is clear that $(\mathcal{A}, \sigma) \models \varphi'$. Otherwise, $(\mathcal{A}, \sigma) \models R(w, u_k)$, which implies $(\sigma^{\mathcal{A}}(w), \sigma^{\mathcal{A}}(u_k)) \in R^{\mathcal{A}}$. Since $R^{\mathcal{A}}$ is a congruence, it follows that

$$\begin{aligned}\sigma^{\mathcal{A}}(f(u_0, \ldots, u_{m-1})) &= f^{\mathcal{A}}(\sigma^{\mathcal{A}}(u_0), \ldots, \sigma^{\mathcal{A}}(u_{m-1}))\\ &= f^{\mathcal{A}}(\sigma^{\mathcal{A}}(u_0), \ldots, \sigma^{\mathcal{A}}(w), \ldots, \sigma^{\mathcal{A}}(u_{m-1}))\\ &= \sigma^{\mathcal{A}}(f(u_0, \ldots, w, \ldots, u_{m-1})).\end{aligned}$$

The following statements are readily seen to be equivalent:

(1) $(\mathcal{A}, \sigma) \models \alpha$;
(2) $(\sigma^{\mathcal{A}}(f(u_0, \ldots, u_{m-1})), \sigma^{\mathcal{A}}(t_1)) \in R^{\mathcal{A}}$;
(3) $(\sigma^{\mathcal{A}}(f(u_0, \ldots, w, \ldots, u_{m-1})), \sigma^{\mathcal{A}}(t_1)) \in R^{\mathcal{A}}$;
(4) $(\mathcal{A}, \sigma) \models R(f(u_0, \ldots, w, \ldots, u_{m-1}), t_1)$.

By Lemma 4.11.6, we have $(\mathcal{A}, \sigma) \models \varphi$ if and only if $(\mathcal{A}, \sigma) \models \varphi'$, which allows us to conclude that $(\mathcal{A}, \sigma) \models \varphi'$. Since for every σ one of the above cases holds, it follow that $\mathcal{A} \models \varphi'$.

Conversely, suppose that $\mathcal{A} \models \varphi'$ and $\sigma \in \text{ASSIGN}_{\mathcal{A}}$. Define $\sigma' = [w \to \sigma^{\mathcal{A}}(u_k)]\sigma$. Since $(\mathcal{A}, \sigma') \models \varphi'$, we have either $(\mathcal{A}, \sigma') \not\models R(w, u_k)$ or $(\mathcal{A}, \sigma') \models \texttt{replace}(\varphi, (\alpha, j), R(f(u_0, \ldots, w, \ldots, u_{m-1}), t_1))$. Note that $(\sigma')^{\mathcal{A}}(u_k) = \sigma^{\mathcal{A}}(u_k)$, because w does not occur in u_k. Furthermore, by the definition of σ', we have $(\sigma')^{\mathcal{A}}(u_k) = \sigma^{\mathcal{A}}(u_k) = \sigma'(w)$. Since $R^{\mathcal{A}}$ is a congruence, it follows that $(\sigma'(w), (\sigma')^{\mathcal{A}}(u_k)) \in R^{\mathcal{A}}$ which means that $(\mathcal{A}, \sigma') \models R(w, u_k)$, so we must have

$$(\mathcal{A}, \sigma') \models \texttt{replace}(\varphi, (\alpha, j), R(f(u_0, \ldots, w, \ldots, u_{m-1}), t_1)).$$

Since $(\sigma')^{\mathcal{A}}(u_k) = \sigma'(w)$, we have $(\mathcal{A}, \sigma') \models \alpha$ if and only if $(\mathcal{A}, \sigma') \models R(f(u_0, \ldots, w, \ldots, u_{m-1}), t_1)$. Thus, by Lemma 4.11.6, we have $(\mathcal{A}, \sigma') \models \varphi$ if and only if

$$(\mathcal{A}, \sigma') \models \texttt{replace}(\varphi, (\alpha, j), R(f(u_0, \ldots, w, \ldots, u_{m-1}), t_1)),$$

which allows us to write $(\mathcal{A}, \sigma') \models \varphi$. Since w does not occur in φ, it follows that $(\mathcal{A}, \sigma) \models \varphi$. Since σ was arbitrary, we have shown that $\mathcal{A} \models \varphi$. □

Algorithm 4.11.9.
Input: A binary relation symbol R belonging to a first-order language $\mathcal{L}$ and a finite set of quantifier-free $\mathcal{L}$-formulas Γ.
Output: A finite set of quantifier-free R-flat $\mathcal{L}$-formulas Γ' such that if $=$ is not in $\mathcal{L}$ or R is $=$ then for every $\mathcal{L}$-structure $\mathcal{A}$ with $R^{\mathcal{A}}$ a congruence, we have $\mathcal{A} \models \Gamma$ if and only if $\mathcal{A} \models \Gamma'$.
Method: If Γ is R-flat, then output Γ. If Γ is not R-flat, then there is a formula $\varphi \in \Gamma$ and an occurrence (α, j) of an atomic formula α in φ such that α is not an R-flat formula. Then, two cases may occur.

`Case 1:` $\alpha = P(t_0, \ldots, t_{n-1})$, where $P \neq R$ and there is an i such that the term t_i is not a variable. In this case, let w be a variable that does not occur in Γ and let

$$\varphi' = (\texttt{replace}\,(\varphi, (\alpha, j), P(t_0, \ldots, t_{i-1}, w, t_{i+1}, \ldots, t_{n-1})) \\ \vee (\neg R(w, t_i))).$$

Define $\hat{\Gamma} = (\Gamma - \{\varphi\}) \cup \varphi'$. Apply the algorithm recursively to $\hat{\Gamma}$ to obtain the output Γ'.

`Case 2:` $\alpha = R(t_0, t_1)$ and there are $i \in \{0, 1\}$ and k, $0 \leq k \leq m-1$ such that the term t_i is $f(u_0, \ldots, u_{m-1})$ and u_k not a variable. Without loss of generality, we may assume that $i = 0$. Let w be a variable that does not occur in Γ and let

$$\varphi' = (\texttt{replace}\,(\varphi, (\alpha, j), ,)R(f(u_0, \ldots, u_{k-1}, w, u_{k+1}, \ldots, \\ u_{m-1}), t_1)) \vee (\neg R(w, u_k)))$$

Define $\hat{\Gamma} = (\Gamma - \{\varphi\}) \cup \varphi'$. Apply the algorithm recursively to $\hat{\Gamma}$ to obtain the output Γ'.

Proof of Correctness: It is clear that if the algorithm terminates, it returns an R-flat set of formulas. The remaining argument consists of two parts. In the first part, we prove that the algorithm terminates, while in the second part we will show that if $=$ is not in $\mathcal{L}$ or R is $=$, then for every $\mathcal{L}$-structure $\mathcal{A}$ with $R^{\mathcal{A}}$ a congruence, we have $\mathcal{A} \models \Gamma$ if and only if $\mathcal{A} \models \Gamma'$.

We claim that in both cases of the method, $F(\varphi') < F(\varphi)$. In the first case, we have:

$$\begin{aligned} F(\varphi') &= F(\varphi) - F(P(t_0, \ldots, t_{n-1})) \\ &\qquad + F(P(t_0, \ldots, t_{i-1}, w, t_{i+1}, \ldots, t_{n-1})) + F(R(w, t_i)) \\ &= F(\varphi) - F(t_i) - 1 + F(t_i) < F(\varphi). \end{aligned}$$

In the second case, we have:

$$\begin{aligned} F(\varphi') &= F(\varphi) - F(R(t_0, t_1)) \\ &\qquad + F(R(f(u_0, \ldots, u_{k-1}, w, u_{k+1}, \ldots, u_{m-1}), t_1)) \\ &\qquad + F(R(w, u_k)) = F(\varphi) - F(u_k) - 1 + F(u_k) < F(\varphi), \end{aligned}$$

which shows that $F(\varphi') < F(\varphi)$. This implies immediately that $F(\hat{\Gamma}) < F(\Gamma)$.

It is easy to verify now by induction on $F(\Gamma)$ that the algorithm terminates when run on the finite set of quantifier-free formulas Γ, using Lemma 4.11.5 for the basis step ($F(\Gamma) = 0$).

Now, we show that if $=$ is not in $\mathcal{L}$ or R is $=$, then for every $\mathcal{L}$-structure $\mathcal{A}$ with $R^{\mathcal{A}}$ a congruence, we have $\mathcal{A} \models \Gamma$ if and only if $\mathcal{A} \models \Gamma'$. First we argue that if Γ is not R-flat, then for such an $\mathcal{L}$-structure $\mathcal{A}$, $\mathcal{A} \models \Gamma$ if and only if $\mathcal{A} \models \hat{\Gamma}$. Since $\hat{\Gamma} = (\Gamma - \{\varphi\}) \cup \{\varphi'\}$, this is equivalent to proving that $\mathcal{A} \models \varphi$ if and only if $\mathcal{A} \models \varphi'$. This conclusion follows from Lemmas 4.11.7 and 4.11.8 because in Case 1 of the construction, due to our assumptions, P is not $=$. Now we prove by induction on $F(\Gamma)$ that for any finite set of quantifier-free $\mathcal{L}$-formulas Γ and $\mathcal{L}$-structure $\mathcal{A}$ such that $R^{\mathcal{A}}$ is a congruence, we have the desired equivalence. In the basis step, $F(\Gamma) = 0$, Γ is an R-flat set of formulas, so $\Gamma' = \Gamma$ and the result is immediate. Suppose $F(\Gamma) > 0$ and that the result holds for finite sets of quantifier-free $\mathcal{L}$-formulas Γ_1 with $F(\Gamma_1) < F(\Gamma)$. Then, since $F(\hat{\Gamma}) < F(\Gamma)$, if $\mathcal{A}$ is defined as above, we have $\mathcal{A} \models \hat{\Gamma}$ if and only if $\mathcal{A} \models \Gamma'$. Combined with our previous considerations, we obtain the desired result. □

It is not difficult to see when Algorithm 4.11.9 is applied to a finite set of clausal formulas, the result is also a finite set of clausal formulas. If the original set of formulas Γ does not contain the connective symbol $\leftrightarrow$, then the same is true about the set of formulas produced by the algorithm.

We refer to any set of quantifier-free formulas produced by the algorithm starting from Γ and R as an *R-flattening of* Γ.

Theorem 4.11.10. *Let $\mathcal{L}$ be a first-order language, R be a binary relation symbol in $\mathcal{L}$, Γ be a finite set of quantifier-free $\mathcal{L}$-formulas and let $\Gamma^{\langle R\rangle}$ be an R-flattening of Γ, where either R is $=$, or $=$ does not belong to $\mathcal{L}$. Then, Γ has a model $\mathcal{A}$ where $\mathcal{A}$ is an $\mathcal{L}$-structure and $R^{\mathcal{A}}$ is a congruence if and only if $\Gamma^{\langle R\rangle}$ has a model $\mathcal{B}$ where $\mathcal{B}$ is an $\mathcal{L}$-structure and $R^{\mathcal{B}}$ is an equivalence relation.*

Proof. This statement follows from Theorem 4.11.3 and from the correctness proof of Algorithm 4.11.9. □

Example 4.11.11. Let $\Gamma_0 = \{R(a,b), P(a), (\neg P(b))\}$. Then, the set of formulas Γ introduced in Example 4.11.2 is an R-flattening of Γ_0. We showed in Example 4.11.4 that Γ has no model $\mathcal{A}$ with $R^{\mathcal{A}}$ an equivalence relation, which by Theorem 4.11.10 means that Γ_0 has no model $\mathcal{A}$ with $R^{\mathcal{A}}$ a congruence, a conclusion which follows as well by inspection of this set of formulas. ▯

Theorem 4.11.12. *Let $\mathcal{L}$ be a first-order language, R, R' be binary relation symbols, where $R \in \mathcal{L}$ and $R' \notin \mathcal{L}$, and let Γ be a finite set of quantifier-free $\mathcal{L}$ formulas. For every R-flattening $\Gamma^{\langle R\rangle}$ of Γ, $\mathsf{s}^{R}_{R'}(\Gamma^{\langle R\rangle})$ is an R'-flattening of $\mathsf{s}^{R}_{R'}(\Gamma)$.*

Further, denoting $\mathsf{s}^{R}_{R'}(\Gamma^{\langle R\rangle})$ by $(\mathsf{s}^{R}_{R'}(\Gamma))^{\langle R'\rangle}$, we have

$$\mathsf{s}^{R'}_{R}((\mathsf{s}^{R}_{R'}(\Gamma))^{\langle R'\rangle}) = \Gamma^{\langle R\rangle}.$$

Proof. The set $\Gamma^{\langle R\rangle}$ was obtained by the application of Algorithm 4.11.9 to Γ. Since $R' \notin \mathcal{L}$, in applying the algorithm to $\mathsf{s}^{R}_{R'}(\Gamma)$, we can follow exactly the same steps with R replaced by R', thus yielding $\mathsf{s}^{R}_{R'}(\Gamma^{\langle R\rangle})$ as an R'-flattening of $\mathsf{s}^{R}_{R'}(\Gamma)$. The last part of the theorem follows directly from Theorem 1.2.16. □

We have just shown how to transform sets of quantifier-free formulas in such a way that the original set of formulas has a model in which a binary relation symbol R is interpreted as a congruence if and only if the transformed set of formulas has a corresponding model in which R is interpreted as an equivalence relation. The next step is to transform a set of formulas in such a way that the original set of formulas has a model in which a binary relation symbol R is interpreted as an equivalence relation if and only if the transformed set of formulas has a model.

We are about to introduce notations used in the next two lemmas. Let R be a binary relation symbol in a first-order language $\mathcal{L}$ and let φ be an $\mathcal{L}$-formula. We say that a formula obtained from φ by replacing each occurrence of a subformula of the form $R(t_0, t_1)$ by $(\forall w)(R(t_0, w) \leftrightarrow R(t_1, w))$, where w is a variable that does not occur in t_0, t_1 is an *e-expansion of* φ *by* R. More formally, we have the following definition.

Definition 4.11.13. Let $(R(t_0^0, t_1^0), k_0), \ldots, (R(t_0^{p-1}, t_1^{p-1}), k_{p-1})$ be the occurrences of formulas of the form $R(t_0, t_1)$ in the formula φ, where $k_0 < \cdots < k_{p-1}$ and let $w_0, \ldots, w_{p-1}$ be p distinct variables such that w_i does not occur in φ. Note that these occurrences do not overlap. Define $\alpha_i = (\forall w_i)(R(t_0^i, w_i) \leftrightarrow R(t_1^i, w_i))$ for $0 \leq i \leq p-1$. Then, the formula

$$\texttt{replace}\,(\varphi, ((R(t_0^0, t_1^0), k_0), \ldots, (R(t_0^{p-1}, t_1^{p-1}), k_{p-1})), (\alpha_0, \ldots, \alpha_{p-1}))$$

is said to be obtained by *e-expansion of* φ *by* R. When $w_0, \ldots, w_{p-1}$ are the first p variables in the standard order that do not occur in φ, we denote that e-expansion by R by $\mathsf{EEXP}_R(\varphi)$. □

Because of Exercise 10 of Chapter 1, a formula obtained by e-expansion of φ by R can be obtained from φ by a sequence of replacements of single occurrences of formulas $R(t_0^i, t_1^i)$ by $(\forall w_i)(R(t_0^i, w_i) \leftrightarrow R(t_1^i, w_i))$.

Definition 4.11.14. Let φ be a formula that does not contain $\leftrightarrow$. If we only replace the positive occurrences of subformulas of the form $R(t_0, t_1)$, we say that the resulting formula is a *positive e-expansion by R*.

If we replace only the negative occurrences of subformulas of the form $R(t_0, t_1)$, then we say that the resulting formula is a *negative e-expansion by R*.

When the choice of the variables w_i is made in the same way as in Definition 4.11.13, we denote the positive (negative) e-expansion by $\mathsf{PEEXP}_R(\varphi)$ ($\mathsf{NEEXP}_R(\varphi)$), respectively.

If Γ is a set of $\mathcal{L}$-formulas that do not contain $\leftrightarrow$, then an *e-expansion (positive e-expansion, negative e-expansion)* of Γ by R is a set of the form $\tilde{\Gamma} = \{\tilde{\varphi} \mid \varphi \in \Gamma\}$, where each $\tilde{\varphi}$ is an e-expansion (positive e-expansion, negative e-expansion) of φ by R.

The e-expansion (positive e-expansion, negative e-expansion) of Γ by R using the previously prescribed choice of variables is denoted by $\mathsf{EEXP}_R(\Gamma)$ ($\mathsf{PEEXP}_R(\Gamma)$, $\mathsf{NEEXP}_R(\Gamma)$), respectively. ▯

Lemma 4.11.15. *Let $\mathcal{L}$ be a first-order language, Γ be a set of $\mathcal{L}$-formulas and $\tilde{\Gamma} = \{\tilde{\varphi} \mid \varphi \in \Gamma\}$ be an e-expansion of Γ by R. If $\mathcal{A}$ is a model of Γ where $R^{\mathcal{A}}$ is an equivalence relation, then $\mathcal{A}$ is a model of $\tilde{\Gamma}$.*

If the formulas in Γ do not contain $\leftrightarrow$, $\tilde{\Gamma}$ is a positive (negative) e-expansion of Γ by R, and $\mathcal{A}$ is a model of Γ where $R^{\mathcal{A}}$ is an equivalence relation, then $\mathcal{A}$ is a model of $\tilde{\Gamma}$.

Proof. Let $\varphi \in \Gamma$, so $\mathcal{A} \models \varphi$. As we observed above, there exists a sequence of formulas $\varphi = \varphi_0, \ldots, \varphi_{p-1} = \tilde{\varphi}$ such that each formula φ_{i+1} is obtained from φ_i by replacing a subformula of the form $\alpha_i = R(t_0^i, t_1^i)$ by $\beta_i = (\forall w_i)(R(t_0^i, w_i) \leftrightarrow R(t_1^i, w_i))$, for $0 \leq i \leq p-1$. As shown in Example 4.5.30, since $R^{\mathcal{A}}$ is an equivalence relation, $\alpha_i \equiv_{\mathcal{A}} \beta_i$. By Lemma 4.6.15, we have $\varphi_i \equiv_{\mathcal{A}} \varphi_{i+1}$. Since $\equiv_{\mathcal{A}}$ is an equivalence relation, it follows that $\varphi \equiv_{\mathcal{A}} \tilde{\varphi}$. Since $\mathcal{A} \models \varphi$, we have $\mathcal{A} \models \tilde{\varphi}$, so $\mathcal{A} \models \tilde{\Gamma}$.

The argument for the second part of the lemma follows the pattern established in the first part. □

Lemma 4.11.16. *Let $\mathcal{L}$ be a first-order language, Γ be a set of $\mathcal{L}$-formulas and $\tilde{\Gamma} = \{\tilde{\varphi} \mid \varphi \in \Gamma\}$ be an e-expansion of Γ by R. If $\tilde{\mathcal{B}}$ is a model of $\tilde{\Gamma}$, then the structure $\mathcal{B}$ is a model of Γ, where the interpretation of $\mathcal{B}$ is the same as the interpretation of $\tilde{\mathcal{B}}$ with the exception of $R^{\mathcal{B}}$, which is an equivalence relation defined as*

$$R^{\mathcal{B}} = \{(a_0, a_1) \mid \text{ for all } b \in |\tilde{\mathcal{B}}| \ (a_0, b) \in R^{\tilde{\mathcal{B}}} \text{ if and only if} \\ \times (a_1, b) \in R^{\tilde{\mathcal{B}}}\}.$$

Proof. It is immediate to verify that $R^{\mathcal{B}}$ is an equivalence relation. Note that when R is $=$, the definition of $R^{\mathcal{B}}$ amounts to the equality relation on $|\mathcal{B}|$.

We will show that for every $\mathcal{L}$-formula φ and every e-expansion $\tilde{\varphi}$ of φ by R, we have $(\tilde{\mathcal{B}}, \sigma) \models \tilde{\varphi}$ if and only if $(\mathcal{B}, \sigma) \models \varphi$, for every $\sigma \in \mathrm{ASSIGN}_{\tilde{\mathcal{B}}} = \mathrm{ASSIGN}_{\mathcal{B}}$. The argument is by induction on the formula φ.

If φ is an atomic formula that does not contain R, then $\tilde{\varphi} = \varphi$ and the desired conclusion follows immediately from Corollary 4.5.26. If $\varphi = R(t_0, t_1)$, then $\tilde{\varphi} = (\forall w)(R(t_0, w) \leftrightarrow R(t_1, w))$, where w is a variable that occurs in neither t_0 nor t_1. The following statements are equivalent, for any assignment σ:

(1) $(\mathcal{B}, \sigma) \models \varphi$;
(2) $(\sigma^{\mathcal{B}}(t_0), \sigma^{\mathcal{B}}(t_1)) \in R^{\mathcal{B}}$;
(3) for all $b \in |\mathcal{B}|$, $(\sigma^{\mathcal{B}}(t_0), b) \in R^{\tilde{\mathcal{B}}}$ if and only if $(\sigma^{\mathcal{B}}(t_1), b) \in R^{\tilde{\mathcal{B}}}$;
(4) for all $b \in |\tilde{\mathcal{B}}|$, $(([w \to b]\sigma)^{\tilde{\mathcal{B}}}(t_0), ([w \to b]\sigma)^{\tilde{\mathcal{B}}}(w)) \in R^{\tilde{\mathcal{B}}}$ if and only if $(([w \to b]\sigma)^{\tilde{\mathcal{B}}}(t_1), ([w \to b]\sigma)^{\tilde{\mathcal{B}}}(w)) \in R^{\tilde{\mathcal{B}}}$;
(5) for all $b \in |\tilde{\mathcal{B}}|$, $(\tilde{\mathcal{B}}, [w \to b]\sigma) \models (R(t_0, w) \leftrightarrow R(t_1, w))$;
(6) $(\tilde{\mathcal{B}}, \sigma) \models (\forall w)(R(t_0, w) \leftrightarrow R(t_1, w)) = \tilde{\varphi}$.

We discuss only the inductive step when $\varphi = (\forall x)\psi$, where we assume that the inductive hypothesis holds for ψ. If $\tilde{\varphi}$ is an e-expansion of φ by R, then $\tilde{\varphi} = (\forall x)\tilde{\psi}$, where $\tilde{\psi}$ is an e-expansion of ψ by R. Then, the following statements are equivalent for any assignment σ.

(1) $(\mathcal{B}, \sigma) \models \varphi$;
(2) for all $b \in |\mathcal{B}|$, $(\mathcal{B}, [x \to b]\sigma) \models \psi$;
(3) for all $b \in |\tilde{\mathcal{B}}|$, $(\tilde{\mathcal{B}}, [x \to b]\sigma) \models \tilde{\psi}$;
(4) $(\tilde{\mathcal{B}}, \sigma) \models \tilde{\varphi}$.

Now, since $\tilde{\mathcal{B}} \models \tilde{\Gamma}$, we have $(\tilde{\mathcal{B}}, \sigma) \models \tilde{\varphi}$ for every $\varphi \in \Gamma$. By the previous argument, $(\mathcal{B}, \sigma) \models \varphi$ for every $\varphi \in \Gamma$, so $\mathcal{B} \models \Gamma$. $\square$

Theorem 4.11.17. *Let $\mathcal{L}$ be a first-order language containing a binary relation symbol R, Γ be a set of $\mathcal{L}$-formulas, and let $\tilde{\Gamma}$ be an e-expansion of Γ by R. Then, Γ has a model $\mathcal{A}$ with $R^{\mathcal{A}}$ an equivalence relation if and only if $\tilde{\Gamma}$ has a model.*

Proof. The result follows immediately from Lemmas 4.11.15 and 4.11.16. $\square$

Example 4.11.18. As we saw in Example 4.11.4, the set of formulas Γ given in Example 4.11.2,

$$\Gamma = \{R(a,b), (P(x) \vee (\neg R(x,a))), ((\neg P(y)) \vee (\neg R(y,b)))\}$$

has no model $\mathcal{A}$ in which $R^{\mathcal{A}}$ is an equivalence relation. By Theorem 4.11.17, the e-expansion

$$\begin{aligned}\tilde{\Gamma} = \{&(\forall w_0)(R(a,w_0) \leftrightarrow R(b,w_0)),\\ &(P(x) \vee (\neg(\forall w_0)(R(x,w_0) \leftrightarrow R(a,w_0)))),\\ &((\neg P(y)) \vee (\neg(\forall w_0)(R(y,w_0) \leftrightarrow R(b,w_0))))\}\end{aligned}$$

has no model. We give a direct proof of this fact. Suppose $\mathcal{A} \models \tilde{\Gamma}$. Letting $\sigma \in \text{ASSIGN}_{\mathcal{A}}$ be such that $\sigma(x) = a^{\mathcal{A}}$, the fact that $(\mathcal{A}, \sigma)$ satisfies the second formula in the previous list $\tilde{\Gamma}$ implies that $a^{\mathcal{A}} \in P^{\mathcal{A}}$. If $\sigma' \in \text{ASSIGN}_{\mathcal{A}}$ is such that $\sigma'(y) = a^{\mathcal{A}}$, the fact that $(\mathcal{A}, \sigma')$ satisfies the first formula in the list means that for all $c \in |\mathcal{A}|$, $(a^{\mathcal{A}}, c) \in R^{\mathcal{A}}$ if and only if $(b^{\mathcal{A}}, c) \in R^{\mathcal{A}}$. Since $(\mathcal{A}, \sigma')$ satisfies the third formula, we have $a^{\mathcal{A}} \notin P^{\mathcal{A}}$, which is a contradiction. ▯

Corollary 4.11.19. *Let $\mathcal{L}$ be a first-order language containing a binary relation symbol R such that either $= \notin \mathcal{L}$ or R is $=$, and let Γ be a finite set of quantifier-free $\mathcal{L}$-formulas. Then, Γ has a model $\mathcal{A}$ with $R^{\mathcal{A}}$ a congruence if and only if $\widetilde{\Gamma^{\langle R \rangle}}$ has a model, where $\Gamma^{\langle R \rangle}$ is an R-flattening of Γ and $\widetilde{\Gamma^{\langle R \rangle}}$ is an e-expansion of $\Gamma^{\langle R \rangle}$ by R.*

Proof. This statement is a consequence of Theorem 4.11.10 and Theorem 4.11.17. $\square$

Theorem 4.11.20. *Let $\mathcal{L}$ be a first-order language containing a binary relation symbol R and let R' be a binary relation symbol that is not in $\mathcal{L}$. If Γ is a set of $\mathcal{L}$-formulas and $\tilde{\Gamma}$ is an e-expansion of Γ by R, then $s^R_{R'}(\tilde{\Gamma})$ is an e-expansion of $s^R_{R'}(\Gamma)$ by R'.*

Denoting $s^R_{R'}(\tilde{\Gamma})$ by $\widetilde{s^R_{R'}(\Gamma)}$, we have

$$s^{R'}_R(\widetilde{s^R_{R'}(\Gamma)}) = \tilde{\Gamma}.$$

Proof. This statement follows from the definition of e-expansion. $\square$

The next statement summarizes the modification method.

Theorem 4.11.21. *Let $\mathcal{L}$ be a first-order language with equality and let R be a binary relation symbol, $R \notin \mathcal{L}$. If Γ is a finite set of quantifier-free $\mathcal{L}$- formulas, $\Gamma^{\langle = \rangle}$ is an $=$-flattening of Γ, and $\widetilde{\Gamma^{\langle = \rangle}}$ is an e-expansion of $\Gamma^{\langle = \rangle}$ by $=$, then Γ has a model if and only if $\mathsf{s}^{=}_{R}(\widetilde{\Gamma^{\langle = \rangle}})$ has a model.*

Proof. By Theorem 4.11.20, $\mathsf{s}^{=}_{R}(\widetilde{\Gamma^{\langle = \rangle}})$ is an e-expansion by R of $\mathsf{s}^{=}_{R}(\Gamma^{\langle = \rangle})$, which we denote by $\widetilde{\mathsf{s}^{=}_{R}(\Gamma^{\langle = \rangle})}$, and by Theorem 4.11.12, $\mathsf{s}^{=}_{R}(\Gamma^{\langle = \rangle})$ is an R-flattening of $\mathsf{s}^{=}_{R}(\Gamma)$, which we denote by $(\mathsf{s}^{=}_{R}(\Gamma))^{\langle R \rangle}$. This gives

$$\mathsf{s}^{=}_{R}(\widetilde{\Gamma^{\langle = \rangle}}) = (\widetilde{\mathsf{s}^{=}_{R}(\Gamma))^{\langle R \rangle}}. \tag{4.9}$$

Since Γ is a finite set of quantifier-free $\mathcal{L}$-formulas, $\mathsf{s}^{=}_{R}(\Gamma)$ is a finite set of quantifier-free $((\mathcal{L} - \{=\}) \cup \{R\})$-formulas. We have the following sequence of equivalent statements:

(1) Γ has a model;
(2) $\mathsf{s}^{=}_{R}(\Gamma) \cup \mathrm{Eq}_{R,\mathcal{L}}$ has a model;
(3) $\mathsf{s}^{=}_{R}(\Gamma)$ has a model $\mathcal{A}$ such that $R^{\mathcal{A}}$ a congruence;
(4) $(\widetilde{\mathsf{s}^{=}_{R}(\Gamma))^{\langle R \rangle}}$ has a model;
(5) $\mathsf{s}^{=}_{R}(\widetilde{\Gamma^{\langle = \rangle}})$ has a model.

(1) is equivalent to (2) by Corollary 4.10.37. The equivalence of (2) and (3) follows from Theorem 4.5.63. Next, the equivalence of (3) and (4) is a consequence of Corollary 4.11.19. Finally, the equivalence between (4) and (5) follows from Equality 4.9. □

To paraphrase the previous theorem, the existence of a model for a set of quantifier-free formulas Γ where $=$ is treated as equality is equivalent to the existence of a model for $\widetilde{\Gamma^{\langle = \rangle}}$ in which $=$ is treated as a regular relation symbol.

Lemma 4.11.22. *Let $\mathcal{L}$ be a first-order language, Γ be a set of $\mathcal{L}$-formulas that do not contain $\leftrightarrow$ and $\tilde{\Gamma} = \{\tilde{\varphi} \mid \varphi \in \Gamma\}$ be a positive e-expansion of Γ by R. If $\tilde{\mathcal{B}}$ is a model of $\tilde{\Gamma}$ with $R^{\tilde{\mathcal{B}}}$ a reflexive relation, then the structure $\mathcal{B}$ is a model of Γ, where the interpretation* *of $\mathcal{B}$ is the same as the interpretation of $\tilde{\mathcal{B}}$ with the exception of $R^{\mathcal{B}}$, which is an equivalence relation defined as*

$$R^{\mathcal{B}} = \{(a_0, a_1) \mid \textit{ for all } b \in |\tilde{\mathcal{B}}|, (a_0, b) \in R^{\tilde{\mathcal{B}}} \textit{ if and only if}$$
$$\times\, (a_1, b) \in R^{\tilde{\mathcal{B}}}\}.$$

Proof. It is immediate to verify that $R^{\mathcal{B}}$ is an equivalence relation and if R is $=$, then $R^{\mathcal{B}}$ is the equality relation on $|\mathcal{B}|$.

Next, we show by induction on the formula φ (which does not contain the connective symbol $\leftrightarrow$) a stronger result than the statement of the lemma, namely, that the following statements hold for all $\sigma \in \text{ASSIGN}_{\tilde{\mathcal{B}}} = \text{ASSIGN}_{\mathcal{B}}$:

(1) if $\tilde{\varphi}$ is a positive e-expansion of φ by R, then $(\tilde{\mathcal{B}}, \sigma) \models \tilde{\varphi}$ implies $(\mathcal{B}, \sigma) \models \varphi$;
(2) if $\tilde{\varphi}$ is a negative e-expansion of φ by R, then $(\mathcal{B}, \sigma) \models \varphi$ implies $(\tilde{\mathcal{B}}, \sigma) \models \tilde{\varphi}$.

For the basis step, let φ be an atomic formula. If φ does not contain R, then both a positive e-expansion by R and a negative e-expansion by R of φ coincide with φ. By Corollary 4.5.26, the conclusion follows immediately. Suppose now that $\varphi = R(t_0, t_1)$. A positive e-expansion by R is in this case an e-expansion by R, and the result follows from the argument of Lemma 4.11.16. A negative e-expansion by R of φ coincides again with φ. Observe that $R^{\mathcal{B}} \subseteq R^{\tilde{\mathcal{B}}}$. Indeed, if $(a_0, a_1) \in R^{\mathcal{B}}$, then choosing $b = a_1$ in the definition of $R^{\mathcal{B}}$, we have $(a_0, a_1) \in R^{\tilde{\mathcal{B}}}$ because $(a_1, a_1) \in R^{\tilde{\mathcal{B}}}$ due to the reflexivity of $R^{\tilde{\mathcal{B}}}$. If $(\mathcal{B}, \sigma) \models \varphi$, then $(\sigma^{\mathcal{B}}(t_0), \sigma^{\mathcal{B}}(t_1)) \in R^{\mathcal{B}}$, which implies $(\sigma^{\tilde{\mathcal{B}}}(t_0), \sigma^{\tilde{\mathcal{B}}}(t_1)) \in R^{\tilde{\mathcal{B}}}$ because $\sigma^{\mathcal{B}}(t_i) = \sigma^{\tilde{\mathcal{B}}}(t_i)$, for $i \in \{0, 1\}$, and $R^{\mathcal{B}} \subseteq R^{\tilde{\mathcal{B}}}$. Finally, this implies $(\tilde{\mathcal{B}}, \sigma) \models \varphi = \tilde{\varphi}$.

For the first inductive step, suppose that $\varphi = (\neg\psi)$ and the result holds for ψ. If $\tilde{\varphi}$ is a positive e-expansion of φ by R, then $\tilde{\varphi} = (\neg\tilde{\psi})$, where $\tilde{\psi}$ is a negative e-expansion of ψ by R. Then, in the following list of statements, each statement implies the next:

(1) $(\tilde{\mathcal{B}}, \sigma) \models \tilde{\varphi}$;
(2) $(\tilde{\mathcal{B}}, \sigma) \not\models \tilde{\psi}$;
(3) $(\mathcal{B}, \sigma) \not\models \psi$;
(4) $(\mathcal{B}, \sigma) \models (\neg\psi) = \varphi$.

Note that the implication (2) $\Rightarrow$ (3) follows by the inductive hypothesis.

If $\tilde{\varphi}$ is a negative e-expansion of φ by R, then $\tilde{\varphi} = (\neg\tilde{\psi})$, where $\tilde{\psi}$ is a positive e-expansion of ψ by R. Then, in the following list of statements, each statement implies the next:

(1) $(\mathcal{B}, \sigma) \models \varphi$;
(2) $(\mathcal{B}, \sigma) \not\models \psi$;
(3) $(\tilde{\mathcal{B}}, \sigma) \not\models \tilde{\psi}$;
(4) $(\tilde{\mathcal{B}}, \sigma) \models (\neg\tilde{\psi}) = \tilde{\varphi}$.

Again, the implication (2) $\Rightarrow$ (3) follows by the inductive hypothesis.

For the next inductive step, suppose that $\varphi = (\psi_0 \to \psi_1)$ and the hypothesis holds for ψ_0 and ψ_1. If $\tilde{\varphi}$ is a positive e-expansion of φ by R, then $\tilde{\varphi} = (\tilde{\psi_0} \to \tilde{\psi_1})$, where $\tilde{\varphi_0}$ is a negative e-expansion of ψ_0 by R and $\tilde{\psi_1}$ is a positive e-expansion of ψ_1 by R. Suppose that $(\tilde{\mathcal{B}}, \sigma) \models \tilde{\varphi}$. Two cases may occur.

(1) $(\tilde{\mathcal{B}}, \sigma) \not\models \tilde{\psi_0}$. By inductive hypothesis, $(\mathcal{B}, \sigma) \not\models \psi_0$, which implies $(\mathcal{B}, \sigma) \models \varphi$.
(2) $(\tilde{\mathcal{B}}, \sigma) \models \tilde{\psi_1}$. Again, by inductive hypothesis, $(\mathcal{B}, \sigma) \models \psi_1$, which implies $(\mathcal{B}, \sigma) \models \varphi$.

If $\tilde{\varphi}$ is a negative e-expansion of φ by R, then $\tilde{\varphi} = (\tilde{\psi_0} \to \tilde{\psi_1})$, where $\tilde{\varphi_0}$ is a positive e-expansion of ψ_0 by R and $\tilde{\psi_1}$ is a negative e-expansion of ψ_1 by R and the argument is similar to the one considered above.

We leave to the reader the inductive steps when $\varphi = (\psi_0 \vee \psi_1)$ or $\varphi = (\psi_0 \wedge \psi_1)$.

Assume now that $\varphi = (\exists x)\psi$ and the inductive hypothesis holds for ψ. If $\tilde{\varphi}$ is a positive e-expansion of φ by R, then, $\tilde{\varphi} = (\exists x)\tilde{\psi}$, where $\tilde{\psi}$ is a positive e-expansion of ψ by R. Suppose that $(\tilde{\mathcal{B}}, \sigma) \models \tilde{\varphi}$. Then, for some $b \in |\tilde{\mathcal{B}}| = |\mathcal{B}|$, we have $(\tilde{\mathcal{B}}, [x \to b]\sigma) \models \tilde{\psi}$. By inductive hypothesis, $(\mathcal{B}, [x \to b]\sigma) \models \psi$, so $(\mathcal{B}, \sigma) \models (\exists x)\psi = \varphi$.

If $\tilde{\varphi}$ is a negative e-expansion of φ by R, then, $\tilde{\varphi} = (\exists x)\tilde{\psi}$, where $\tilde{\psi}$ is a negative e-expansion of ψ by R and the argument follows similar lines as above.

We leave to the reader the argument for the inductive step when $\varphi = (\forall x)\psi$.

The statement of the lemma follows immediately since it is the "positive" half of the result shown above. $\square$

Theorem 4.11.23. *Let $\mathcal{L}$ be a first-order language containing a binary relation symbol R, Γ be a set of $\mathcal{L}$-formulas such that no formula in Γ contains $\leftrightarrow$, and let $\tilde{\Gamma}$ be a positive e-expansion of Γ by R. Then, Γ has a model $\mathcal{A}$ with $R^{\mathcal{A}}$ an equivalence relation if and only if $\tilde{\Gamma}$ has model $\mathcal{B}$ with $R^{\mathcal{B}}$ a reflexive relation.*

Proof. This statement follows from Lemmas 4.11.15 and 4.11.22. □

Example 4.11.24. Again, as we saw in Example 4.11.4, the set of formulas Γ given in Example 4.11.2,

$$\Gamma = \{R(a,b), (P(x) \vee (\neg R(x,a))), ((\neg P(y)) \vee (\neg R(y,b)))\}$$

has no model $\mathcal{A}$ in which $R^{\mathcal{A}}$ is an equivalence relation. By Theorem 4.11.23, the positive e-expansion

$$\tilde{\Gamma} = \{(\forall w_0)(R(a,w_0) \leftrightarrow R(b,w_0)), (P(x) \vee (\neg R(x,a))),$$
$$((\neg P(y)) \vee (\neg R(y,b)))\}$$

has no model with $R^{\mathcal{A}}$ a reflexive relation. We give a direct proof of this fact. Suppose $\mathcal{A} \models \tilde{\Gamma}$. Letting $\sigma \in \text{ASSIGN}_{\mathcal{A}}$ be such that $\sigma(x) = a^{\mathcal{A}}$, the fact that $(\mathcal{A},\sigma)$ satisfies the second formula in the previous list $\tilde{\Gamma}$ and the fact that $R^{\mathcal{A}}$ is reflexive implies that $a^{\mathcal{A}} \in P^{\mathcal{A}}$.

If $\sigma' \in \text{ASSIGN}_{\mathcal{A}}$ is such that $\sigma'(y) = a^{\mathcal{A}}$, the fact that $(\mathcal{A},\sigma')$ satisfies the first formula in the list means that for all $c \in |\mathcal{A}|$, $(a^{\mathcal{A}},c) \in R^{\mathcal{A}}$ if and only if $(b^{\mathcal{A}},c) \in R^{\mathcal{A}}$. Taking $c = b^{\mathcal{A}}$ and using the fact that $R^{\mathcal{A}}$ is reflexive, we have $(a^{\mathcal{A}},b^{\mathcal{A}}) \in R^{\mathcal{A}}$. Since $(\mathcal{A},\sigma')$ satisfies the third formula, we have $a^{\mathcal{A}} \notin P^{\mathcal{A}}$, which is a contradiction. ▯

Corollary 4.11.25. *Let $\mathcal{L}$ be a first-order language containing a binary relation symbol R such that either $=\notin \mathcal{L}$ or R is $=$, and let Γ be a finite set of quantifier-free $\mathcal{L}$-formulas such that no formula in Γ contains $\leftrightarrow$. Then, Γ has a model $\mathcal{A}$ with $R^{\mathcal{A}}$ a congruence if and only if $\widetilde{\Gamma^{\langle R\rangle}}$ has a model $\tilde{\mathcal{B}}$ such that $R^{\tilde{\mathcal{B}}}$ is a reflexive relation, where $\Gamma^{\langle R\rangle}$ is an R-flattening of Γ and $\widetilde{\Gamma^{\langle R\rangle}}$ is a positive e-expansion of $\Gamma^{\langle R\rangle}$ by R.*

Proof. This statement is a consequence of Theorem 4.11.10 and Theorem 4.11.23. □

Observe that in Corollary 4.11.25, if Γ is a set of formulas in clausal form, then so is $\Gamma^{\langle R\rangle}$. However, $\widetilde{\Gamma^{\langle R\rangle}}$ is not, in general, in clausal form, and this causes difficulties for proofs that apply only to formulas in clausal form.

Starting from $\mathsf{PEEXP}_R(\varphi)$ where φ is an $\mathcal{L}$-formula in clausal form, we construct a finite set of formulas $\mathsf{CF}_R(\varphi)$ in clausal form such that for any $\mathcal{L}$-structure $\mathcal{A}$, we have $\mathcal{A} \models \mathsf{PEEXP}_R(\varphi)$ if and only if $\mathcal{A} \models \mathsf{CF}_R(\varphi)$.

We refer the reader to Definition 2.5.1 where we introduced the notation φ^b for a formula φ and binary digit b. This notation, which is about to be used in the next algorithm, is extended to formulas of first-order logic.

Algorithm 4.11.26.
Input: A first-order language $\mathcal{L}$, a binary relation symbol $R \in \mathcal{L}$, and a clausal $\mathcal{L}$-formula φ.
Output: A set of clausal $\mathcal{L}$-formulas $\mathsf{CF}_R(\varphi)$ such that $\mathcal{A} \models \mathsf{PEEXP}_R(\varphi)$ if and only if $\mathcal{A} \models \mathsf{CF}_R(\varphi)$, for all $\mathcal{L}$-structures $\mathcal{A}$.
Method: Let $\varphi = (\ell_0 \vee \cdots \vee \ell_{n-1})$.
Partition the set $\{0, \ldots, n-1\}$ into two blocks $I = \{i_0, \ldots, i_{p-1}\}$ and $J = \{j_0, \ldots, j_{q-1}\}$, where $i_0 < \cdots < i_{p-1}$ and $j_0 < \cdots < j_{q-1}$, such that the set $\{\ell_i \mid i \in I\}$ contains all literals of the form $R(t_0, t_1)$ that occur in φ. For $0 \leq r \leq p-1$, let $\ell_{i_r} = R(t_0^r, t_1^r)$ and let $w_0, \ldots, w_{p-1}$ be the first p variables that do not occur in φ.
Using notations introduced prior to Supplement 23 of Chapter 2, let $M = 2^p$, and $(s_0, \ldots, s_{M-1})$ be the sequence of elements of $\{0,1\}^p$, for $p \in \mathbf{N}$.
Return $\mathsf{CF}_R(\varphi)$ given by:

$$\begin{aligned}\mathsf{CF}_R(\varphi) = \{(&R(t_0^0, w_0)^{s_i(0)} \vee R(t_1^0, w_0)^{(1-s_i(0))} \\ &\vee R(t_0^1, w_1)^{s_i(1)} \vee R(t_1^1, w_1)^{(1-s_i(1))} \vee \cdots \\ &\vee R(t_0^{p-1}, w_{p-1})^{s_i(p-1)} \vee R(t_1^{p-1}, w_{p-1})^{(1-s_i(p-1))} \\ &\vee \ell_{j_0} \vee \cdots \vee \ell_{j_{q-1}}) \mid 0 \leq i \leq M-1\}.\end{aligned} \quad (4.10)$$

Proof of Correctness: By Exercise 43, $\mathsf{PEEXP}_R(\varphi)$ is logically equivalent to the formula $\varphi' = (\psi_0 \vee \cdots \vee \psi_{p-1} \vee \ell_{j_0} \vee \cdots \vee \ell_{j_{q-1}})$, where ψ_i is the result of a positive e-expansion by R, $\psi_i = (\forall w_i)(R(t_0^i, w_i) \leftrightarrow R(t_1^i, w_i))$.

By Exercise 118, φ' is logically equivalent to

$$(\forall w_0)\cdots(\forall w_{p-1})(\gamma_0 \vee \cdots \vee \gamma_{p-1} \vee \ell_{j_0} \vee \cdots \vee \ell_{j_{q-1}}),$$

where $\gamma_i = (R(t_0^i, w_i) \leftrightarrow R(t_1^i, w_i))$ for $0 \leq i \leq p-1$. By Theorem 4.5.58, we have $\mathcal{A} \models \mathsf{PEEXP}_R(\varphi)$ if and only if $\mathcal{A} \models \delta = (\gamma_0 \vee \cdots \vee \gamma_{p-1} \vee \ell_{j_0} \vee \cdots \vee \ell_{j_{q-1}})$. Let $\gamma_i^b = (R(t_0^i, w_i)^b \vee R(t_1^i, w_i)^{(1-b)})$ for $0 \leq i \leq p-1$ and $b \in \{0, 1\}$. By Theorems 4.8.17 and 4.6.16, $\delta \equiv ((\gamma_0^0 \wedge \gamma_0^1) \vee (\gamma_1^0 \wedge \gamma_1^1) \vee \cdots \vee (\gamma_{p-1}^0 \wedge \gamma_{p-1}^1) \vee \ell_{j_0} \vee \cdots \vee \ell_{j_{q-1}})$. By Supplement 106, δ is logically equivalent to $\delta' = \bigwedge_{i=0}^{M-1}(\gamma_0^{s_i(0)} \vee \cdots \vee \gamma_{p-1}^{s_i(p-1)} \vee \ell_{j_0} \vee \cdots \vee \ell_{j_{q-1}})$. Thus, $\mathcal{A} \models \mathsf{PEEXP}_R(\varphi)$ is equivalent to $\mathcal{A} \models \delta'$ and this in turn is equivalent to $\mathcal{A} \models \mathsf{CF}_R(\varphi)$, by Supplement 105. □

We refer to the set $\mathsf{CF}_R(\varphi)$ obtained as above as the *R-clausification* of φ. This notation is extended by defining $\mathsf{CF}_R(\Gamma)$ for a set of clausal formulas Γ as $\bigcup\{\mathsf{CF}_R(\varphi) \mid \varphi \in \Gamma\}$.

Example 4.11.27. The set of formulas

$$\Gamma = \{R(a,b), (P(x) \vee (\neg R(x,a))), ((\neg P(y)) \vee (\neg R(y,b)))\}$$

introduced in Example 4.11.2 is in clausal form and the set $\mathsf{CF}_R(\Gamma)$ is given by

$$\Gamma = \{((\neg R(a, w_0)) \vee R(b, w_0)), (R(a, w_0) \vee (\neg R(b, w_0))),$$
$$(P(x) \vee (\neg R(x,a))), ((\neg P(y)) \vee (\neg R(y,b)))\}.$$

▯

Theorem 4.11.28. *Let $\mathcal{L}$ be a first-order language that contains a binary relation R and let Γ be a set of clausal $\mathcal{L}$-formulas. Then, Γ has a model $\mathcal{A}$ with $R^{\mathcal{A}}$ an equivalence relation if and only if $\mathsf{CF}_R(\Gamma)$ has a model $\mathcal{B}$ with $R^{\mathcal{B}}$ a reflexive relation.*

Proof. This follows immediately from Theorem 4.11.23 and the correctness of Algorithm 4.11.26. □

Corollary 4.11.29. *Let $\mathcal{L}$ be a first-order language, R be a binary relation symbol of $\mathcal{L}$ such that $= \notin \mathcal{L}$ or R is $=$ and let Γ be a finite set of clausal $\mathcal{L}$-formulas. If $\Gamma^{\langle R \rangle}$ is an R-flattening of Γ and $\Gamma' = \mathsf{CF}_R(\Gamma^{\langle R \rangle})$, then Γ has a model $\mathcal{A}$ with $R^{\mathcal{A}}$ a congruence if and only if Γ' has a model $\mathcal{B}$ with $R^{\mathcal{B}}$ a reflexive relation.*

Proof. We noted previously that an R-flattening of a set of clausal formulas is a set of clausal formulas, so Γ' exists. The statement follows from Theorems 4.11.10 and 4.11.28. $\Box$

Theorem 4.11.30. *Let $\mathcal{L}$ be a first-order language that contains a binary relation symbol R and let R' be a binary relation symbol not in $\mathcal{L}$. For any clausal $\mathcal{L}$-formula φ we have*

$$\mathsf{s}^R_{R'}(\mathsf{CF}_R(\varphi)) = \mathsf{CF}_{R'}(\mathsf{s}^R_{R'}(\varphi)).$$

Proof. Since R' does not occur in φ, the positive R'-literals in $\mathsf{s}^R_{R'}(\varphi)$ correspond to the positive R-literals in φ. Also, the $\mathsf{s}^R_{R'}$ does not affect the variables that occur in a formula, so the first p variables not occurring in φ are the same as the first p variables not occurring in $\mathsf{s}^R_{R'}(\varphi)$. The result follows from Formula (4.10). $\Box$

4.12 Hintikka Sets and Truth Sets

As we did in the corresponding section of Chapter 2, we introduce the notion of constituent of a first-order formula and use this notion to introduce Hintikka sets and truth sets. Constituents will again be obtained by analyzing formulas, and in some limited sense will represent ways of satisfying formulas. Although we cannot preserve the property that there is one constituent for each distinct way of satisfying a formula, we will see that there is a broad class of structures (including Herbrand structures) such that a formula is satisfied in these structures if and only if one of its constituents is satisfied in the structure.

Hintikka sets in the setting of first-order logic are still sets that are not just satisfiable, but show explicitly how to define a satisfying structure and assignment. We will show that every satisfiable set of $\mathcal{L}$-formulas is contained in a Hintikka set for a suitable extension of the language $\mathcal{L}$.

4.12.1 *Constituents*

Definition 4.12.1. Let $\mathcal{L}$ be a first-order language and V be an $\mathcal{L}$-suitable set of variables. Further, let $t_0, t_1, \ldots$ be the enumeration of

$\mathrm{TERM}_{\mathcal{L}}(V)$ in the standard order. The function

$$\mathtt{d}_{\mathcal{L},V} : \mathrm{FORM}_{\mathcal{L}} - \mathrm{LIT}_{\mathcal{L}} \longrightarrow \mathrm{Seq}(\mathcal{P}(\mathrm{FORM}_{\mathcal{L}})) \cup \mathrm{ISeq}(\mathcal{P}(\mathrm{FORM}_{\mathcal{L}}))$$

is given by the following table:

Formula α	$\mathtt{d}_{\mathcal{L},V}(\alpha)$
$(\neg(\neg\varphi))$	$(\{\varphi\})$
$(\varphi \wedge \psi)$ $(\neg(\varphi \wedge \psi))$	$(\{\varphi, \psi\})$ $(\{(\neg\varphi)\}, \{(\neg\psi)\})$
$(\varphi \vee \psi)$ $(\neg(\varphi \vee \psi))$	$(\{\varphi\}, \{\psi\})$ $(\{(\neg\varphi), (\neg\psi)\})$
$(\varphi \rightarrow \psi)$ $(\neg(\varphi \rightarrow \psi))$	$(\{(\neg\varphi)\}, \{\psi\})$ $(\{\varphi, (\neg\psi)\})$
$(\varphi \leftrightarrow \psi)$ $(\neg(\varphi \leftrightarrow \psi))$	$(\{\varphi, \psi\}, \{(\neg\varphi), (\neg\psi)\})$ $(\{\varphi, (\neg\psi)\}, \{(\neg\varphi), \psi\})$
$(\forall x)\varphi$ $(\neg(\forall x)\varphi)$	$(\{\langle\varphi\rangle_{x:=t} \mid t \in \mathrm{TERM}_{\mathcal{L}}(V)\})$ $(\{\langle(\neg\varphi)\rangle_{x:=t_0}\}, \{\langle(\neg\varphi)\rangle_{x:=t_1}\}, \ldots)$
$(\exists x)\varphi$ $(\neg(\exists x)\varphi)$	$(\{\langle\varphi\rangle_{x:=t_0}\}, \{\langle\varphi\rangle_{x:=t_1}\}, \ldots)$ $(\{\langle(\neg\varphi)\rangle_{x:=t} \mid t \in \mathrm{TERM}_{\mathcal{L}}(V)\})$

The $(\mathcal{L}, V)$-*constituent sequence* of φ is the sequence $\mathtt{d}_{\mathcal{L},V}(\varphi)$. The $(\mathcal{L}, V)$-*constituent set* of φ is the set $\mathtt{D}_{\mathcal{L},V}(\varphi)$ that consists of all sets of formulas that occur in $\mathtt{d}_{\mathcal{L},V}(\varphi)$. Every such set of formulas is called an $(\mathcal{L}, V)$-*constituent* of φ.

If $V = \emptyset$, we denote $\mathtt{d}_{\mathcal{L},V}(\varphi)$ and $\mathtt{D}_{\mathcal{L},V}(\varphi)$ by $\mathtt{d}_{\mathcal{L}}(\varphi)$ and $\mathtt{D}_{\mathcal{L}}(\varphi)$, respectively. Also, we refer to $(\mathcal{L}, V)$-constituents as $\mathcal{L}$-constituents. □

To circumscribe the set of formulas that may appear by repeatedly applying the operation of taking a constituent, we need to introduce several functions.

The function $U : \mathcal{P}(\mathrm{FORM}) \longrightarrow \mathcal{P}(\mathrm{FORM})$, similar to the function U introduced for propositional logic in Theorem 2.2.14, is given by

$$U(\Gamma) = \mathrm{PRSUBF}(\Gamma) \cup \{(\neg\psi) \mid \psi \in \mathrm{PRSUBF}(\Gamma)\},$$

where PRSUBF(Γ) is the set of proper subformulas of the formulas in Γ.

Definition 4.12.2. Let $\mathcal{L}$ be a first-order language and V be an $\mathcal{L}$-suitable set of variables. The function

$$W_{\mathcal{L},V} : \mathcal{P}(\mathrm{FORM}_{\mathcal{L}}(V)) \longrightarrow \mathcal{P}(\mathrm{FORM}_{\mathcal{L}}(V))$$

is given by

$$\begin{aligned} W_{\mathcal{L},V}(\Gamma) = \{(\varphi')_{x:=t} \mid {} & \varphi \in \Gamma \cup U(\Gamma), \\ & \varphi' \text{ is a variant of } \varphi,\ x \in \mathrm{VAR}, t \in \mathrm{TERM}_{\mathcal{L}}(V), \\ & \text{and } t \text{ is substitutable for } x \text{ in } \varphi'\}. \end{aligned}$$

The $(\mathcal{L}, V)$-*analytical universe* is the set $W^*_{\mathcal{L},V}(\Gamma) = \bigcup_{n\in\mathbf{N}} W^n_{\mathcal{L},V}(\Gamma)$, where $W^n_{\mathcal{L},V}$ is the nth iteration:

$$W^n_{\mathcal{L},V}(\Gamma) = \underbrace{W_{\mathcal{L},V}(\cdots W_{\mathcal{L},V}}_{n}(\Gamma)\cdots)$$

for $\Gamma \subseteq \mathrm{FORM}_{\mathcal{L}}(V)$. □

It is easy to see that $W_{\mathcal{L},V}(W^*_{\mathcal{L},V}(\Gamma)) \subseteq W^*_{\mathcal{L},V}(\Gamma)$.

Example 4.12.3. Let $\mathcal{L} = \{c, R, P\}$, where c is a constant symbol, R is a unary relation symbol and P is a binary relation symbol. Consider the formula $\varphi = (\forall x)(R(x) \vee (\forall y)P(x, y))$. Suppose that $V = \emptyset$.

We have:

$$\begin{aligned} U(\{\varphi\}) = \{ & (R(x) \vee (\forall y)P(x,y)), R(x), (\forall y)P(x,y), P(x,y), \\ & (\neg(R(x) \vee (\forall y)P(x,y))), (\neg R(x)), (\neg(\forall y)P(x,y)), (\neg P(x,y))\}. \end{aligned}$$

Clearly, $W^0_{\mathcal{L},V}(\{\varphi\}) = \{\varphi\}$. Further, note that the formulas

$$
\begin{aligned}
&(R(x) \lor (\forall y)P(x,y)), R(x), (\forall y)P(x,y), P(x,y),\\
&\quad (\neg(R(x) \lor (\forall y)P(x,y))), (\neg R(x)), (\neg(\forall y)P(x,y)), (\neg P(x,y))\\
&\quad (R(c) \lor (\forall y)P(c,y)), R(c), (\forall y)P(c,y), P(c,y), P(x,c)\\
&\quad (\neg(R(c) \lor (\forall y)P(c,y))), (\neg R(c)), (\neg(\forall y)P(c,y)),\\
&\qquad (\neg P(c,y)), (\neg P(x,c))\\
&\quad (R(c) \lor (\forall z)P(c,z)), R(c), (\forall z)P(c,z), P(c,y), P(x,c)\\
&\quad (\neg(R(c) \lor (\forall z)P(c,z))), (\neg(\forall z)P(c,z))
\end{aligned}
$$

belong to $W^1_{\mathcal{L},V}(\{\varphi\})$ for any variable z. Formulas such as $P(c,c)$ belong to $W^2_{\mathcal{L},V}(\{\varphi\})$. □

It is easy to verify that if ψ belongs to a constituent of a formula φ, then $\parallel \psi \parallel < \parallel \varphi \parallel$. (We remind the reader that $\parallel \theta \parallel$, the norm of θ, was introduced in Definition 4.3.21.) Further, if K is an $(\mathcal{L},V)$-constituent of φ, then, by Corollary 4.3.84, $K \subseteq W_{\mathcal{L},V}(\{\varphi\})$, which is called the *analyticity* of the constituents of first-order formulas.

Theorem 4.12.4. *Let $\mathcal{L}$ be a first-order language, and let V be an $\mathcal{L}$-suitable set of variables. If $\psi \in K \in \mathtt{D}_{\mathcal{L},V}(\varphi)$, where $\varphi \in \mathrm{FORM}_{\mathcal{L}} - LIT_{\mathcal{L}}$, then $\mathtt{FV}(\psi) \subseteq \mathtt{FV}(\varphi) \cup V$.*

Proof. The proof can be done by inspecting the definition of the constituents of φ and applying Corollary 4.6.48. □

Corollary 4.12.5. *Any member of an $(\mathcal{L},V)$-constituent of a formula in $\mathrm{FORM}_{\mathcal{L}}(V)$ is itself in $\mathrm{FORM}_{\mathcal{L}}(V)$.*

Proof. The statement follows from the previous theorem. □

Corollary 4.12.6. *Let $\mathcal{L}$ be a first-order language that contains a constant symbol. If φ is a closed $\mathcal{L}$-formula that is not a literal, then every formula in an $\mathcal{L}$-constituent of φ is closed.*

Proof. The statement follows immediately from Theorem 4.12.4□

As the next example shows, the analogue of Theorem 2.7.2 does not hold for first-order logic.

Example 4.12.7. Let $\mathcal{L} = \{a, R\}$ be a first-order language, where a is a constant symbol and R is a unary relation symbol. The formula

$\varphi = (\exists x)R(x)$ has a unique $(\mathcal{L}, \emptyset)$-constituent, namely, $K = \{R(a)\}$. Define an $\mathcal{L}$-structure $\mathcal{A}$ by $|\mathcal{A}| = \{0, 1\}$, $R^{\mathcal{A}} = \{0\}$ and $a^{\mathcal{A}} = 1$. Then, for any assignment σ over $\mathcal{A}$, we have $(\mathcal{A}, \sigma) \models \varphi$ but $(\mathcal{A}, \sigma) \not\models K$, so it is possible for a structure and assignment to satisfy a formula without satisfying any of its constituents.

Let now $\mathcal{B}$ be the $\mathcal{L}$-structure identical to $\mathcal{A}$ except that $a^{\mathcal{B}} = 0$ and let $\psi = (\forall x)R(x)$. The $(\mathcal{L}, \emptyset)$-constituent of ψ is $K = \{R(a)\}$. Note that for any assignment σ over $\mathcal{B}$, $(\mathcal{B}, \sigma) \models K$ but $(\mathcal{B}, \sigma) \not\models \varphi$. So, it is possible for a structure and assignment to satisfy a constituent of a formula without satisfying the formula. $\square$

The next two theorems show that Theorem 2.7.2 remains true in a limited sense. Specifically, in Theorem 4.12.11, we restrict to structures whose elements can be "named" by appropriate terms, while in Theorem 4.12.15, we examine what is left of Theorem 2.7.2 for arbitrary structures.

Definition 4.12.8. Let $\mathcal{L}$ be a first-order language and let V be an $\mathcal{L}$-suitable set of variables. A pair $(\mathcal{A}, \sigma)$, where $\mathcal{A}$ is an $\mathcal{L}$-structure and $\sigma \in \mathrm{ASSIGN}_{\mathcal{A}}$, is said to be *$V$-named* if for every $a \in |\mathcal{A}|$, there is a $t \in \mathrm{TERM}_{\mathcal{L}}(V)$ such that $\sigma^{\mathcal{A}}(t) = a$.

If $\mathcal{L}$ contains at least one constant symbol, an $\mathcal{L}$-structure $\mathcal{B}$ is said to be *named* if for every $b \in |\mathcal{B}|$, there is a $t \in \mathrm{GTERM}_{\mathcal{L}}$ such that $t^{\mathcal{B}} = b$. $\square$

Note that the following three statements are equivalent, when $\mathcal{L}$ contains at least one constant symbol:

(1) $\mathcal{A}$ is named;
(2) $(\mathcal{A}, \sigma)$ is $\emptyset$-named for every $\sigma \in \mathrm{ASSIGN}_{\mathcal{A}}$;
(3) $(\mathcal{A}, \sigma)$ is $\emptyset$-named for some $\sigma \in \mathrm{ASSIGN}_{\mathcal{A}}$.

Example 4.12.9. Let $\mathcal{L}$ and $\mathcal{A}$ be as in Example 4.4.4. Then, $\mathcal{A}$ is named. Indeed, we can prove by induction on $n \in \mathbf{N}$ that there is a ground term t with $t^{\mathcal{A}} = n$. We have $0^{\mathcal{A}} = 0$ and, assuming that $t^{\mathcal{A}} = n$, we have $(s(t))^{\mathcal{A}} = s^{\mathcal{A}}(t^{\mathcal{A}}) = s^{\mathcal{A}}(n) = n + 1$. $\square$

Example 4.12.10. Let V be an $\mathcal{L}$-suitable set of variables, where $\mathcal{L}$ is a first-order language. If $\mathcal{A}$ is a V-Herbrand structure for $\mathcal{L}$ and $\sigma \in \mathrm{ASSIGN}_{\mathcal{A}}$ is such that $\sigma(x) = x$ for all $x \in V$, then $(\mathcal{A}, \sigma)$ is a V-named pair, by Theorem 4.10.8.

If ρ is a congruence on $\mathcal{A}$ and $\sigma' \in \mathrm{ASSIGN}_{\mathcal{A}/\rho}$ is such that $\sigma'(x) = [x]_\rho$, for all $x \in V$, then $(\mathcal{A}/\rho, \sigma')$ is V-named by Corollary 4.10.10. ▯

Theorem 4.12.11. *Let $\mathcal{L}$ be a first-order language and V be an $\mathcal{L}$-suitable set of variables. If $\mathcal{A}$ is an $\mathcal{L}$-structure, $\sigma \in \mathrm{ASSIGN}_{\mathcal{A}}$, and $(\mathcal{A}, \sigma)$ is V-named, then for all $\mathcal{L}$-formulas α that are not literals, we have $(\mathcal{A}, \sigma) \models \alpha$ if and only if $(\mathcal{A}, \sigma) \models K$ for some $(\mathcal{L}, V)$-constituent K of α.*

Proof. There are several cases depending on the form of α. We discuss here only some of the cases that are dissimilar to the ones encountered in propositional logic.

Suppose that $\alpha = (\forall x)\varphi$. Then, the following statements are equivalent:

(1) $(\mathcal{A}, \sigma) \models (\forall x)\varphi$;
(2) for every $a \in |\mathcal{A}|$, $(\mathcal{A}, [x \to a]\sigma) \models \varphi$;
(3) for every $t \in \mathrm{TERM}_{\mathcal{L}}(V)$, $(\mathcal{A}, [x \to \sigma^{\mathcal{A}}(t)]\sigma) \models \varphi$;
(4) for every $t \in \mathrm{TERM}_{\mathcal{L}}(V)$, $(\mathcal{A}, [x \to \sigma^{\mathcal{A}}(t)]\sigma) \models \mathsf{variant}(\varphi, x, t)$;
(5) for every $t \in \mathrm{TERM}_{\mathcal{L}}(V)$, $(\mathcal{A}, \sigma) \models (\mathsf{variant}(\varphi, x, t))_{x:=t}$;
(6) for every $t \in \mathrm{TERM}_{\mathcal{L}}(V)$, $(\mathcal{A}, \sigma) \models \langle\varphi\rangle_{x:=t}$.

The equivalences of (1) and (2), and of (5) and (6) are obvious. Statement (2) is equivalent to (3) since $(\mathcal{A}, \sigma)$ is V-named. The equivalence of (3) and (4) follows from Theorem 4.6.47. Finally, (4) is equivalent to (5) by the Substitution Corollary (Corollary 4.6.6). Thus, we may conclude that $(\mathcal{A}, \sigma) \models \alpha$ if and only if $(\mathcal{A}, \sigma) \models K$, where $K = \{\langle\varphi\rangle_{x:=t} \mid t \in \mathrm{TERM}_{\mathcal{L}}(V)\}$ is the $(\mathcal{L}, V)$-constituent of α.

Now suppose that $\alpha = (\neg(\exists x)\varphi)$. Then, $(\mathcal{A}, \sigma) \models \alpha$ if and only if $(\mathcal{A}, \sigma) \models (\forall x)(\neg\varphi)$ (by Lemma 4.9.7), which, by the previous argument, is equivalent to $(\mathcal{A}, \sigma) \models \{\langle(\neg\varphi)\rangle_{x:=t} \mid t \in \mathrm{TERM}_{\mathcal{L}}(V)\}$. Since $\{\langle(\neg\varphi)\rangle_{x:=t} \mid t \in \mathrm{TERM}_{\mathcal{L}}(V)\}$ is the unique constituent of α, the statement holds in this case as well. □

Corollary 4.12.12. *Let $\mathcal{L}$ be a first-order language that contains at least one constant symbol. If $\mathcal{A}$ is a named $\mathcal{L}$-structure, then for all closed $\mathcal{L}$-formulas α that are not literals, $\mathcal{A} \models \alpha$ if and only if $\mathcal{A} \models K$ for some $\mathcal{L}$-constituent K of α.*

Proof. This statement follows from Theorem 4.12.11 by taking $V = \emptyset$ and using the fact that, by Corollary 4.12.6, all formulas in K are closed. $\square$

Corollary 4.12.13. *Let V be an $\mathcal{L}$-suitable set of variables, where $\mathcal{L}$ is a first-order language. Also, let $\mathcal{A}$ be a V-Herbrand structure for $\mathcal{L}$. If $\sigma \in \mathrm{ASSIGN}_{\mathcal{A}}$ is such that $\sigma(x) = x$ for all $x \in V$, then for all $\mathcal{L}$-formulas α that are not literals, $(\mathcal{A}, \sigma) \models \alpha$ if and only if $(\mathcal{A}, \sigma) \models K$ for some $(\mathcal{L}, V)$-constituent K of α.*

If $\mathcal{L}$ contains a constant symbol and $\mathcal{A}_0$ is a Herbrand structure for $\mathcal{L}$, then, for every closed $\mathcal{L}$-formula α which is not a literal, we have $\mathcal{A}_0 \models \alpha$ if and only if $\mathcal{A}_0 \models K$ for some $\mathcal{L}$-constituent K of α.

Proof. Observe that the pair $(\mathcal{A}, \sigma)$ is V-named, by Example 4.12.10. Therefore, the first part of the corollary follows from Theorem 4.12.11. The second part follows from Corollary 4.12.12 and the fact that $\mathcal{A}_0$ is a named structure. $\square$

Corollary 4.12.14. *Let V be an $\mathcal{L}$-suitable set of variables, where $\mathcal{L}$ is a first-order language. Also, let $\mathcal{A}$ be a V-Herbrand structure for $\mathcal{L}$, ρ be a congruence of $\mathcal{A}$ and $\mathcal{B} = \mathcal{A}/\rho$. If $\sigma \in \mathrm{ASSIGN}_{\mathcal{B}}$ is such that $\sigma(x) = [x]$ for all $x \in V$, then for all $\mathcal{L}$-formulas α that are not literals, $(\mathcal{B}, \sigma) \models \alpha$ if and only if $(\mathcal{B}, \sigma) \models K$ for some $(\mathcal{L}, V)$-constituent K of α.*

If $\mathcal{L}$ contains a constant symbol, $\mathcal{A}_0$ is a Herbrand structure for $\mathcal{L}$, ρ is a congruence of $\mathcal{A}_0$, and $\mathcal{B}_0 = \mathcal{A}_0/\rho$, then for all closed formulas α that are not literals, we have $\mathcal{B}_0 \models \alpha$ if and only if $\mathcal{B}_0 \models K$ for some constituent K of α.

Proof. By Example 4.12.10, the pair $(\mathcal{B}, \sigma)$ is V-named. So, the first part of the Corollary follows from Theorem 4.12.11. The second part is a consequence of Corollary 4.12.12 and the fact that $\mathcal{B}_0$ is a named structure. $\square$

Theorem 4.12.15. *Let $\mathcal{L}$ be a first-order language, V be an $\mathcal{L}$-suitable set of variables, and φ be an $\mathcal{L}$-formula. Suppose that $\mathcal{A}$ is an $\mathcal{L}$-structure and $\sigma \in \mathrm{ASSIGN}_{\mathcal{A}}$.*

(1) *If φ is neither a $\boldsymbol{\gamma}$-formula, a $\boldsymbol{\delta}$-formula or a literal, then $(\mathcal{A}, \sigma) \models \varphi$ if and only if $(\mathcal{A}, \sigma) \models K$ for some $(\mathcal{L}, V)$-constituent K of φ.*

(2) *If φ is a $\boldsymbol{\gamma}$-formula, then $(\mathcal{A}, \sigma) \models \varphi$ implies $(\mathcal{A}, \sigma) \models K$ for the $(\mathcal{L}, V)$-constituent K of φ.*
(3) *If φ is a $\boldsymbol{\delta}$-formula, then $(\mathcal{A}, \sigma) \models K$ for some $(\mathcal{L}, V)$-constituent K of φ implies $(\mathcal{A}, \sigma) \models \varphi$.*

Proof. The argument for the first part is immediate.

For the second part, assume first that $\varphi = (\forall x)\psi$. By Theorem 4.6.51, we have $\varphi \models \langle \psi \rangle_{x:=t}$ for every term t, so $(\mathcal{A}, \sigma) \models \varphi$ implies $(\mathcal{A}, \sigma) \models \{\langle \psi \rangle_{x:=t} \mid t \in \mathrm{TERM}_{\mathcal{L}}(V)\}$. Since $\{\langle \psi \rangle_{x:=t} \mid t \in \mathrm{TERM}_{\mathcal{L}}(V)\}$ is the constituent of φ, this is the desired conclusion. If $\varphi = (\neg(\exists x)\psi)$, then, since $\varphi \equiv (\forall x)(\neg\psi)$, by the argument just made, $(\mathcal{A}, \sigma) \models \varphi$ implies $(\mathcal{A}, \sigma) \models \{\langle(\neg\psi)\rangle_{x:=t} \mid t \in \mathrm{TERM}_{\mathcal{L}}(V)\}$. Thus, we reach again the desired conclusion.

For the third part, assume first that $\varphi = (\exists x)\psi$. By Theorem 4.6.51, we have $\langle \psi \rangle_{x:=t} \models \varphi$ for every term t. Since each constituent K of φ has the form $\{\langle \psi \rangle_{x:=t}\}$, $(\mathcal{A}, \sigma) \models K$ implies $(\mathcal{A}, \sigma) \models \varphi$. Finally, suppose that $\varphi = (\neg(\forall x)\psi)$. Since $\varphi \equiv (\exists x)(\neg\psi)$, by applying the previous argument, $(\mathcal{A}, \sigma) \models \{\langle(\neg\psi)\rangle_{x:=t}$ with $t \in \mathrm{TERM}_{\mathcal{L}}(V)$ implies $(\mathcal{A}, \sigma) \models \varphi$. □

4.12.2 *Hintikka Sets of Unsigned Formulas*

Definition 4.12.16. Let $\mathcal{L}$ be a first-order language, V be an $\mathcal{L}$-suitable set of variables, and Γ be a set of $\mathcal{L}$-formulas such that $\mathtt{FV}(\Gamma) \subseteq V$.

If $\mathcal{L}$ does not contain the equality symbol, then Γ is an $(\mathcal{L}, V)$-*Hintikka set* if the following conditions are satisfied:

(1) for every atomic $\mathcal{L}$-formula φ, at most one of the literals $\varphi, (\neg\varphi)$ is in Γ;
(2) if $\varphi \in \Gamma$ and φ is not a literal, then there is an $(\mathcal{L}, V)$-constituent K of φ such that $K \subseteq \Gamma$.

If $\mathcal{L}$ does contain the equality symbol, then Γ is an $(\mathcal{L}, V)$-Hintikka set if Γ satisfies the previous two conditions and $\mathrm{INST}_{\mathcal{L},V}(\mathrm{Eq}_{=,\mathcal{L}}) \subseteq \Gamma$.

We refer to an $(\mathcal{L}, \emptyset)$-Hintikka set as an $\mathcal{L}$-Hintikka set. ▯

Note that an $\mathcal{L}$-Hintikka set consists of closed formulas.

Theorem 4.12.17. *Let $\mathcal{L}$ be a first-order language without equality, V be an $\mathcal{L}$-suitable set of variables, and Γ be an $(\mathcal{L}, V)$-Hintikka set. Then, Γ is satisfiable. In fact, there is a V-Herbrand structure $\mathcal{A}$ for $\mathcal{L}$ and $\sigma \in \mathrm{ASSIGN}_{\mathcal{A}}$ such that $\sigma(x) = x$ for all $x \in V$ and $(\mathcal{A}, \sigma) \models \Gamma$.*

Proof. Let $\mathcal{A} = \mathsf{STR}_{\mathcal{L},V}(S)$ be the V-Herbrand structure for $\mathcal{L}$ determined by $S = \Gamma \cap \mathrm{AFORM}_{\mathcal{L}}$ (as introduced in Definition 4.10.5) and let $\sigma \in \mathrm{ASSIGN}_{\mathcal{A}}$ be an assignment such that $\sigma(x) = x$ for every $x \in V$.

We prove by course-of-values induction on $\| \varphi \|$ that if $\varphi \in \Gamma$, then $(\mathcal{A}, \sigma) \models \varphi$. Suppose that $\varphi \in \Gamma$ and that the result holds for all ψ with $\| \psi \| < \| \varphi \|$. If φ is atomic, then $\varphi \in S$, so, $(\mathcal{A}, \sigma) \models \varphi$ by Lemma 4.10.11. If $\varphi = (\neg \psi)$, where ψ is atomic, then $\psi \notin \Gamma$, by the definition of Hintikka set, so $\psi \notin S$. Consequently, $(\mathcal{A}, \sigma) \not\models \psi$, by the same lemma, which allows us to conclude that $(\varphi, \sigma) \models \varphi$.

Now suppose that φ is not a literal. Then, by the definition of Hintikka set, there is an $(\mathcal{L}, V)$-constituent K of φ such that $K \subseteq \Gamma$. Since the norm of each formula in K is less than the norm of φ, we have $(\mathcal{A}, \sigma) \models K$, by inductive hypothesis. Thus, by Corollary 4.12.13, $(\mathcal{A}, \sigma) \models \varphi$. $\square$

In preparation for extending Theorem 4.12.17 to languages with equality, we need the following technical result.

Lemma 4.12.18. *If Γ is an $(\mathcal{L}, V)$-Hintikka set for some first-order language $\mathcal{L}$ and some set of variables V and Γ contains a formula $(\neg(\varphi_0 \wedge \varphi_1 \wedge \cdots \wedge \varphi_{n-1}))$, then Γ contains at least one formula $(\neg \varphi_i)$ with $0 \leq i \leq n-1$.*

Proof. The argument is by induction on n and is left to the reader. $\square$

Lemma 4.12.19. *Let $\mathcal{L}$ be a first-order language with equality and let V be an $\mathcal{L}$-suitable set of variables.*

Suppose that Γ is an $(\mathcal{L}, V)$-Hintikka set and $S = \Gamma \cap \mathrm{AFORM}_{\mathcal{L}-\{=\}}(V)$. Define the relation ρ on $\mathrm{TERM}_{\mathcal{L}}(V)$ by $t \rho s$ if and only if $t = s \in \Gamma$. Then, ρ is a congruence of the Herbrand structure $\mathcal{A} = \mathsf{STR}_{\mathcal{L},V}(S)$.

Proof. Since Γ is an $(\mathcal{L}, V)$-Hintikka set, Γ contains $\mathrm{INST}_{\mathcal{L},V}(\mathrm{Eq}_{=,\mathcal{L}})$. We prove first that ρ is an equivalence relation on $\mathrm{TERM}_{\mathcal{L}}(V)$. The reflexivity of ρ follows from the fact that $t = t \in \Gamma$ for every $t \in \mathrm{TERM}_{\mathcal{L}}(V)$ because $t = t$ is an instance of the formula $(\forall x_0)(x_0 = x_0)$ from $\mathrm{Eq}_{=,\mathcal{L}}$.

To show the symmetry of ρ, suppose that $t\rho s$, that is, $t = s \in \Gamma$. Since Γ contains the formula $(t = s) \rightarrow (s = t)$ (as an instance of the formula $(\forall x_0)(\forall x_1)((x_0 = x_1) \rightarrow (x_1 = x_0))$), it follows that Γ contains at least one of the formulas $(\neg(t = s))$ and $(s = t)$, because Γ is a $(\mathcal{L}, V)$-Hintikka set. Note that Γ may not contain both $t = s$ and $(\neg(t = s))$, so it contains $s = t$ and hence $s\rho t$.

To prove the transitivity of ρ, suppose that $t\rho s$ and $s\rho u$, that is both $t = s$ and $s = u$ belong to Γ. Note that Γ contains the instance $(((t = s) \wedge (s = u)) \rightarrow (t = u))$ of a formula in $\mathrm{Eq}_{=,\mathcal{L}}$. Since Γ is an $(\mathcal{L}, V)$-Hintikka set, Γ must contain at least one of the formulas $(\neg((t = s) \wedge (s = u)))$ and $t = u$, so it must contain at least one of the formulas $(\neg(t = s)), (\neg(s = u))$, and talk $t = u$. The presence in Γ of the formulas $t = s$ and $s = u$ implies that Γ contains $t = u$, so $t\rho u$.

We have shown that ρ is an equivalence on $\mathrm{TERM}_{\mathcal{L}}(V)$. Let us show now that it is compatible with the operations and relations of $\mathcal{A}$. Suppose that $t_0\rho s_0, \ldots, t_{n-1}\rho s_{n-1}$, where $n > 0$, that is $t_i = s_i \in \Gamma$ for $0 \leq i \leq n-1$. Note that Γ cannot contain $(\neg(t_0 = s_0 \wedge \cdots \wedge t_{n-1} = s_{n-1}))$, since otherwise, by Lemma 4.12.18, it would have to contain at least one of the formulas $(\neg(t_i = s_i))$ for $0 \leq i \leq n-1$.

Let f be an n-ary relation symbol of $\mathcal{L}$. We must show that

$$f^{\mathcal{A}}(t_0, \ldots, t_{n-1})\rho f^{\mathcal{A}}(s_0, \ldots, s_{n-1}),$$

that is $f(t_0, \ldots, t_{n-1})\rho f(s_0, \ldots, s_{n-1})$. By the definition of Hintikka set, Γ contains the instance

$$(((t_0 = s_0) \wedge \cdots \wedge (t_{n-1} = s_{n-1})) \rightarrow f(t_0, \ldots, t_{n-1}) = f(s_0, \ldots, s_{n-1}))$$

of a formula in $\mathrm{Eq}_{=,\mathcal{L}}$. Therefore, Γ contains at least one of $(\neg(t_0 = s_0 \wedge \cdots \wedge t_{n-1} = s_{n-1}))$ and $f(t_0, \ldots, t_{n-1}) = f(s_0, \ldots, s_{n-1})$. Since we have already ruled out the former case, we have $f(t_0, \ldots, t_{n-1})\rho f(s_0, \ldots, s_{n-1})$, as desired.

If R is an n-ary relation symbol of $\mathcal{L} - \{=\}$, we show that $(t_0, \ldots, t_{n-1}) \in R^{\mathcal{A}}$ if and only if $(s_0, \ldots, s_{n-1}) \in R^{\mathcal{A}}$, that is,

$R(t_0, \ldots, t_{n-1}) \in \Gamma$ if and only if $R(s_0, \ldots, s_{n-1}) \in \Gamma$. An argument similar to the one of the previous paragraph shows that we must have in Γ the formula $(R(t_0, \ldots, t_{n-1}) \leftrightarrow R(s_0, \ldots, s_{n-1}))$. Since Γ is an $(\mathcal{L}, V)$-Hintikka set, at least one of the sets

$$\{R(t_0, \ldots, t_{n-1}), R(s_0, \ldots, s_{n-1})\},$$
$$\{(\neg R(t_0, \ldots, t_{n-1})), (\neg R(s_0, \ldots, s_{n-1}))\}$$

is included in Γ. Suppose that $R(t_0, \ldots, t_{n-1}) \in \Gamma$. In this case, Γ cannot contain the second set, so it must contain the first set, which implies that $R(s_0, \ldots, s_{n-1}) \in \Gamma$. The reverse implication is similar. $\square$

The counterpart of Theorem 4.12.17 for languages with equality is given next.

Theorem 4.12.20. *Let $\mathcal{L}$ be a first-order language with equality, V be an $\mathcal{L}$-suitable set of variables, and Γ be an $(\mathcal{L}, V)$-Hintikka set. Then, Γ is satisfiable. In fact, there is a V-Herbrand structure $\mathcal{A}$ for $\mathcal{L}$, a congruence ρ of $\mathcal{A}$ and $\sigma \in \mathrm{ASSIGN}_{\mathcal{A}/\rho}$ such that $\sigma(x) = [x]_\rho$ for all $x \in V$ and $(\mathcal{A}/\rho, \sigma) \models \Gamma$.*

Proof. Define the set S as $\Gamma \cap \mathrm{AFORM}_{\mathcal{L}-\{=\}}$ and let $\mathcal{A}$ be the Herbrand structure $\mathsf{STR}_{\mathcal{L},V}(S)$. Consider the structure $\mathcal{B} = \mathcal{A}/\rho$, where ρ is the congruence of $\mathcal{A}$ defined in Lemma 4.12.19. Let $\sigma \in \mathrm{ASSIGN}_{\mathcal{B}}$ be any assignment such that $\sigma(x) = [x]_\rho$ for every $x \in V$. We claim that $(\mathcal{B}, \sigma) \models \varphi$ for every $\varphi \in \Gamma$.

As in the proof of Theorem 4.12.17, the argument is by course-of-values induction on $\|\varphi\|$. If we show the claim for the literals in Γ, the remaining argument is similar to the one used in Theorem 4.12.17, but uses Theorem 4.12.11 in place of Corollary 4.12.13. There are three cases to consider, depending on whether φ does not contain $=$, is $t = s$, or is $t \neq s$.

(1) Suppose that φ is a literal that does not contain $=$. Define $\sigma' \in \mathrm{ASSIGN}_{\mathcal{A}}$ by $\sigma'(x) = x$ if $x \in V$ and $\sigma'(x) = t$, where t is the first term in $[\sigma(x)]_\rho$, otherwise. Note that $\sigma = h_\rho \circ \sigma'$. We have $(\mathcal{A}, \sigma') \models \varphi$ by the same argument as the one used in Theorem 4.12.17. By the Morphism Theorem, we have, $(\mathcal{A}/\rho, h_\rho \circ \sigma') \models \varphi$, so $(\mathcal{B}, \sigma) \models \varphi$.

(2) Let now φ be $t = s$. Since $t = s \in \Gamma$, we have $t\rho s$, so $[t]_\rho = [s]_\rho$. Thus, by Corollary 4.10.10, $\sigma^{\mathcal{B}}(t) = \sigma^{\mathcal{B}}(s)$, so $(\mathcal{B}, \sigma) \models t = s$.

(3) Finally, suppose that $\varphi = t \neq s$. Since Γ is an $(\mathcal{L}, V)$-Hintikka set, $t = s \notin \Gamma$, so $[t]_\rho \neq [s]_\rho$. By the same argument, this means that $\sigma^{\mathcal{B}}(t) \neq \sigma^{\mathcal{B}}(s)$, so $(\mathcal{B}, \sigma) \models t \neq s$.

□

Corollary 4.12.21. *Let $\mathcal{L}$ be a first-order language and V be an $\mathcal{L}$-suitable set of variables. If Γ is an $(\mathcal{L}, V)$-Hintikka set, then Γ is satisfiable.*

Proof. This statement follows from Theorems 4.12.17 and 4.12.20. □

As in the propositional case, we introduce the idea of a consistency property. The first-order logic counterpart of the propositional notion is not completely parallel to it.

Definition 4.12.22. Let $\mathcal{L}$ be a first-order language and let V be an $\mathcal{L}$-suitable set of variables.

An $(\mathcal{L}, V)$-*consistency property* is a collection $\mathcal{C}$ of sets of $\mathcal{L}$-formulas whose free variables are included in V such that

- No set with property $\mathcal{C}$ contains both an atomic formula and its negation.
- If Γ has property $\mathcal{C}$, $\varphi \in \Gamma - \mathrm{LIT}_{\mathcal{L}}$, and φ is not a $\boldsymbol{\gamma}$-formula, then $\Gamma \cup K$ has property $\mathcal{C}$ for some constituent K of φ.
- If Γ has property $\mathcal{C}$ and $(\forall x)\psi \in \Gamma$, then $\Gamma \cup \{\langle \psi \rangle_{x:=t}\}$ has property $\mathcal{C}$ for every $t \in \mathrm{TERM}_{\mathcal{L}}(V)$.
- If Γ has property $\mathcal{C}$ and $(\neg(\exists x)\psi) \in \Gamma$, then $\Gamma \cup \{\langle (\neg\psi) \rangle_{x:=t}\}$ has property $\mathcal{C}$ for every $t \in \mathrm{TERM}_{\mathcal{L}}(V)$.
- If $\mathcal{L}$ contains $=$ and Γ has property $\mathcal{C}$, then $\Gamma \cup \{\alpha\}$ has property $\mathcal{C}$, for every $\alpha \in \mathrm{INST}_{\mathcal{L},V}(\mathrm{Eq}_{=,\mathcal{L}})$.

An $(\mathcal{L}, V)$-*inconsistency property* is a collection $\mathcal{I}$ of sets of formulas in $\mathrm{FORM}_{\mathcal{L}}(V)$ such that $\mathcal{P}(\mathrm{FORM}_{\mathcal{L}}(V)) - \mathcal{I}$ is an $(\mathcal{L}, V)$-consistency property. ▯

Observe that a collection $\mathcal{I}$ of sets of formulas in $\mathrm{FORM}_{\mathcal{L}}(V)$ is an $(\mathcal{L}, V)$-inconsistency property if the following conditions are satisfied for every $\Gamma \subseteq \mathrm{FORM}_{\mathcal{L}}(V)$:

- If Γ includes $\{\varphi, (\neg\varphi)\}$ for some atomic formula φ, then $\Gamma \in \mathcal{I}$.
- If $\varphi \in \Gamma - \mathrm{LIT}_{\mathcal{L}}$ is such that φ is not a $\boldsymbol{\gamma}$-formula and $\Gamma \cup K \in \mathcal{I}$ for every constituent K of φ, then $\Gamma \in \mathcal{I}$.

- If $(\forall x)\psi \in \Gamma$ and $\Gamma \cup \{\langle\psi\rangle_{x:=t}\} \in \mathcal{I}$ for some $t \in \mathrm{TERM}_{\mathcal{L}}(V)$, then $\Gamma \in \mathcal{I}$.
- If $(\neg(\exists x)\psi) \in \Gamma$ and $\Gamma \cup \{\langle(\neg\psi)\rangle_{x:=t}\} \in \mathcal{I}$ for some $t \in \mathrm{TERM}_{\mathcal{L}}(V)$, then $\Gamma \in \mathcal{I}$.
- If $\mathcal{L}$ contains $=$ and $\Gamma \cup \{\alpha\} \in \mathcal{I}$ for some $\alpha \in \mathrm{INST}_{\mathcal{L},V}(\mathrm{Eq}_{=,\mathcal{L}})$, then $\Gamma \in \mathcal{I}$.

Theorem 4.12.23. *Let $\mathcal{L}$ be a first-order language and let V be an $\mathcal{L}$-suitable set of variables. If $\mathcal{C}$ is an $(\mathcal{L}, V)$-consistency property, then every member of $\mathcal{C}$ is a satisfiable set of formulas. In fact, if Γ is a member of $\mathcal{C}$, there is an $(\mathcal{L}, V)$-Hintikka set Γ' such that $\Gamma \subseteq \Gamma'$. Further, we have*

$$\Gamma' \subseteq \begin{cases} W^*_{\mathcal{L},V}(\Gamma) & \text{if } = \notin \mathcal{L} \\ W^*_{\mathcal{L},V}(\Gamma \cup \mathrm{INST}_{\mathcal{L},V}(Eq_{=,\mathcal{L}})) & \text{if } = \in \mathcal{L} \end{cases}$$

(a property referred to as the analyticity of the construction of Γ').

Proof. Let $\varphi_0, \varphi_1, \ldots$ be the enumeration of $\mathrm{FORM}_{\mathcal{L}}(V)$ in the standard order and let $t_0, t_1, \ldots$ be the enumeration of $\mathrm{TERM}_{\mathcal{L}}(V)$ in the standard order of terms. Fix a pairing function $f : \mathbf{N}^2 \longrightarrow \mathbf{N}$, that is, f is a fixed bijection from $\mathbf{N}^2$ to $\mathbf{N}$. We can list $\mathrm{FORM}_{\mathcal{L}}(V) \times \mathrm{TERM}_{\mathcal{L}}(V)$ by placing the pair (φ_i, t_j) in the $(k+1)$-st place, for $k = f(i, j)$.

Let $\mathcal{C}$ be an $(\mathcal{L}, V)$-consistency property and let Γ have property $\mathcal{C}$.

First, suppose that $\mathcal{L}$ does not contain equality. We shall construct an $(\mathcal{L}, V)$-Hintikka set Γ' such that $\Gamma \subseteq \Gamma' \subseteq W^*_{\mathcal{L},V}(\Gamma)$. By Theorem 4.12.17, we can then conclude that Γ is satisfiable. We build recursively a sequence of sets $\Gamma_0, \Gamma_1, \ldots$, each with property $\mathcal{C}$ and each included in $W^*_{\mathcal{L},V}(\Gamma)$. Then, we set $\Gamma' = \bigcup\{\Gamma_n \mid n \in \mathbf{N}\}$.

We define $\Gamma_0 = \Gamma$. Then, $\Gamma_0 \subseteq W^*_{\mathcal{L},V}(\Gamma)$. Suppose that Γ_n is defined, $\Gamma_n \in \mathcal{C}$, and $\Gamma_n \subseteq W^*_{\mathcal{L},V}(\Gamma)$. Then, there are two cases:

- If Γ_n is an $(\mathcal{L}, V)$-Hintikka set, $\Gamma_{n+1} = \Gamma_n$.
- If Γ_n is not an $(\mathcal{L}, V)$-Hintikka set, then, since no set with property $\mathcal{C}$ may contain both an atomic formula and its negation, there must be a formula $\varphi \in \Gamma_n$ such that none of its constituents is included in Γ_n. Let (φ, t) be the first pair in the enumeration of $\mathrm{FORM}_{\mathcal{L}}(V) \times \mathrm{TERM}_{\mathcal{L}}(V)$ such that one of the following conditions holds:

(a) $\varphi \in \Gamma_n$, φ is not a $\boldsymbol{\gamma}$-formula and none of φ's constituents is contained in Γ_n;
(b) $\varphi \in \Gamma_n$, $\varphi = (\forall x)\psi$ and $\langle\psi\rangle_{x:=t} \notin \Gamma_n$;
(c) $\varphi \in \Gamma_n$, $\varphi = (\neg(\exists x)\psi)$ and $\langle(\neg\psi)\rangle_{x:=t} \notin \Gamma_n$.

In the first case, let K be the first constituent of φ in the sequence $\mathtt{d}_{\mathcal{L},V}(\varphi)$ such that $\Gamma_n \cup K$ is in $\mathcal{C}$. (Such a K exists by the definition of consistency property.) Define $\Gamma_{n+1} = \Gamma_n \cup K$. In the second, let $\Gamma_{n+1} = \Gamma_n \cup \{\langle\psi\rangle_{x:=t}\}$. In the third case, we define $\Gamma_{n+1} = \Gamma_n \cup \{\langle(\neg\psi)\rangle_{x:=t}\}$. In the latter two cases, Γ_{n+1} has property $\mathcal{C}$, again by definition of consistency property. In all cases, $\Gamma_{n+1} \subseteq W^*_{\mathcal{L},V}(\Gamma)$ because of the analyticity of the constituents.

We claim that Γ' is an $(\mathcal{L}, V)$-Hintikka set. Note that Γ' cannot contain both an atomic formula and its negation. Suppose that Γ' is not an $(\mathcal{L}, V)$-Hintikka set. Then, there is a formula $\varphi \in \Gamma'$ which is not a literal such that none of its constituents is contained in Γ'. This means that there is a pair $(\varphi, t) \in \mathrm{FORM}_{\mathcal{L}}(V) \times \mathrm{TERM}_{\mathcal{L}}(V)$ such that $\varphi \in \Gamma'$ and one of the following cases holds:

(1) φ is not a literal, is not a $\boldsymbol{\gamma}$-formula, and none of its constituents is contained in Γ';
(2) $\varphi = (\forall x)\psi$ and $\langle\psi\rangle_{x:=t} \notin \Gamma'$;
(3) $\varphi = (\neg(\exists x)\psi)$ and $\langle(\neg\psi)\rangle_{x:=t} \notin \Gamma'$.

Let (φ, t) be the first such pair in the above listing of $\mathrm{FORM}_{\mathcal{L}}(V) \times \mathrm{TERM}_{\mathcal{L}}(V)$. Take n sufficiently large such that $\varphi \in \Gamma_n$, and for every pair (φ', t') that precedes (φ, t) in the ordering, with $\varphi' \in \Gamma' - \mathrm{LIT}_{\mathcal{L}}$, one of the following holds:

(1) φ' is not a $\boldsymbol{\gamma}$-formula and some constituent of φ' is included in Γ_n;
(2) $\varphi' = (\forall x')\psi'$ and $\langle\psi'\rangle_{x':=t'} \in \Gamma_n$;
(3) $\varphi' = (\neg(\exists x')\psi')$ and $\langle(\neg\psi')\rangle_{x':=t'} \in \Gamma_n$.

Such an n exists because for all predecessors (φ', t') of (φ, t) such that $\varphi' \in \Gamma' - \mathrm{LIT}_{\mathcal{L}}$, if φ is not a $\boldsymbol{\gamma}$-formula, then there is a constituent of φ' included in Γ' and each such constituent is finite. By the construction of the sequence $\Gamma_0, \Gamma_1, \ldots$, the set Γ_{n+1} satisfies one of the following conditions:

(1) if φ is not a $\boldsymbol{\gamma}$-formula, then some constituent of φ is included in Γ_{n+1};
(2) if $\varphi = (\forall x)\psi$, then $\langle\psi\rangle_{x:=t} \in \Gamma_{n+1}$;
(3) if $\varphi = (\neg(\exists x)\psi)$, then $\langle(\neg\psi)\rangle_{x:=t} \in \Gamma_{n+1}$.

This contradicts the choice of (φ, t), since $\Gamma_{n+1} \subseteq \Gamma'$.

Assume now that $\mathcal{L}$ contains $=$. We define $\Gamma_0 = \Gamma$. Suppose that Γ_n is defined and is in $\mathcal{C}$. Then, there are two cases:

- If Γ_n is an $(\mathcal{L}, V)$-Hintikka set, $\Gamma_{n+1} = \Gamma_n$.
- If Γ_n is not an $(\mathcal{L}, V)$-Hintikka set, then, since no set with property $\mathcal{C}$ may contain both an atomic formula and its negation, there must be a formula φ such that either $\varphi \in \Gamma_n$ and none of its constituents is included in Γ_n, or $\varphi \in \mathrm{INST}_{\mathcal{L},V}(\mathrm{Eq}_{=,\mathcal{L}})$ and $\varphi \notin \Gamma_n$. Let (φ, t) be the first pair in the enumeration of $\mathrm{FORM}_{\mathcal{L}}(V) \times \mathrm{TERM}_{\mathcal{L}}(V)$ such that one of the following conditions holds:

  (a) $\varphi \in \Gamma_n$, φ is not a $\boldsymbol{\gamma}$-formula and none of φ's constituents is contained in Γ_n;
  (b) $\varphi \in \Gamma_n$, $\varphi = (\forall x)\psi$ and $\langle\psi\rangle_{x:=t} \notin \Gamma_n$;
  (c) $\varphi \in \Gamma_n$, $\varphi = (\neg(\exists x)\psi)$ and $\langle(\neg\psi)\rangle_{x:=t} \notin \Gamma_n$;
  (d) $\varphi \in \mathrm{INST}_{\mathcal{L},V}(\mathrm{Eq}_{=,\mathcal{L}})$ and $\varphi \notin \Gamma_n$.

  In the first three cases, we proceed as before. In the fourth case, let $\Gamma_{n+1} = \Gamma_n \cup \{\varphi\}$.

We leave it to the reader to prove that Γ' is a $(\mathcal{L}, V)$-Hintikka set and that $\Gamma' \subseteq W^*_{\mathcal{L},V}(\Gamma \cup \mathrm{INST}_{\mathcal{L},V}(\mathrm{Eq}_{=,\mathcal{L}}))$. □

In contrast to Example 2.7.19, satisfiability is not, in general, a first-order consistency property.

Example 4.12.24. Let $\mathcal{L}$ be any first-order language that contains relation symbols that are not propositional constants, V be an $\mathcal{L}$-suitable set of variables and $\mathcal{C}$ be the collection of all satisfiable sets Γ such that $\Gamma \subseteq \mathrm{FORM}_{\mathcal{L}}(V)$. We will show that $\mathcal{C}$ is not an $(\mathcal{L}, V)$-consistency property. Let $R \in \mathcal{L}$ be an n-ary relation symbol with $n > 0$ and let $\Gamma = \{(\exists x)(\neg R(x, t_0, \ldots, t_0)\} \cup \{R(t, t_0, \ldots, t_0) \mid t \in \mathrm{TERM}_{\mathcal{L}}(V)\}$, where t_0 is an arbitrary, fixed term in $\mathrm{TERM}_{\mathcal{L}}(V)$.

We prove first that Γ is satisfiable. To this end, consider the $\mathcal{L}$-structure $\mathcal{A}$ with $|\mathcal{A}| = \{0, 1\}$, $f^{\mathcal{A}}(a_0, \ldots, a_{m-1}) = 0$ for each m-ary function symbol $f \in \mathcal{L}$, $R^{\mathcal{A}} = \{(a_0, \ldots, a_{n-1}) \mid a_0 = \cdots = a_{n-1}\}$ and $P^{\mathcal{A}}$ defined arbitrarily for every relation symbol $P \neq R$ of $\mathcal{L}$.

Note that, by this definition, if R is $=$, then $R^{\mathcal{A}}$ is the equality relation on $|\mathcal{A}|$, as it must be. If $\sigma(y) = 0$ for every variable y, we have $\sigma^{\mathcal{A}}(t) = 0$ for every $\mathcal{L}$-term t, and hence Γ is satisfiable because $(\mathcal{A}, \sigma) \models \Gamma$.

If satisfiability were a consistency property, $\Gamma \cup K$ would be satisfiable, for some $(\mathcal{L}, V)$-constituent K of the formula $\varphi = (\exists x)(\neg R(x, t_0, \ldots, t_0))$. Since each such constituent has the form $\{(\neg R(t, t_0, \ldots, t_0))\}$ for some term $t \in \mathrm{TERM}_{\mathcal{L}}(V)$, each such $\Gamma \cup K$ is unsatisfiable. $\square$

In spite of the negative result of Example 4.12.24, there is still a sense in which satisfiability is a consistency property, as the next theorem shows.

Theorem 4.12.25. *Let $\mathcal{L}$ be a first-order language, V be a set of variables and $\mathcal{C}$ be the collection of all satisfiable subsets Γ of $\mathrm{FORM}_{\mathcal{L}^c}(V)$ with limited constant symbols. Then, $\mathcal{C}$ is an $(\mathcal{L}^c, V)$-consistency property.*

Proof. Suppose $\Gamma \in \mathcal{C}$, $(\mathcal{A}, \sigma) \models \Gamma$ for some $\mathcal{L}^c$-structure $\mathcal{A}$ and $\sigma \in \mathrm{ASSIGN}_{\mathcal{A}}$, and $\varphi \in \Gamma - \mathrm{LIT}_{\mathcal{L}^c}$. If φ is neither a $\boldsymbol{\gamma}$-formula nor a $\boldsymbol{\delta}$-formula, then by Theorem 4.12.15, $(\mathcal{A}, \sigma) \models \Gamma \cup K$ for some $(\mathcal{L}^c, V)$-constituent K of φ. Since K is finite, it is clear that $\Gamma \cup K$ also belongs to $\mathcal{C}$.

If φ is a $\boldsymbol{\gamma}$-formula, then, by the same argument, $\Gamma \cup K$ is satisfiable, where K is the $(\mathcal{L}^c, V)$-constituent of φ. If $\varphi = (\forall x)\psi$, this clearly implies that $\Gamma \cup \{\langle\psi\rangle_{x:=t}\}$ is also satisfiable for every term $t \in \mathrm{TERM}_{\mathcal{L}^c}(V)$. Since $\{\langle\psi\rangle_{x:=t}\}$ contains only finitely many constant symbols, it follows that $\Gamma \cup \{\langle\psi\rangle_{x:=t}\} \in \mathcal{C}$. The argument is similar for $\varphi = (\neg(\exists x)\psi)$.

If φ is the $\boldsymbol{\delta}$-formula $(\exists x)\psi$, then let $c \in \mathcal{L}^c$ be a constant symbol that does not occur in Γ. (Such a c exists because $\Gamma \in \mathcal{C}$.) By Theorem 4.6.52, $\Gamma \cup \{\langle\psi\rangle_{x:=c}\}$ is satisfiable and belongs to $\mathcal{C}$ because $\langle\psi\rangle_{x:=c}$ contains only one constant symbol not in Γ. If $\varphi = (\neg(\forall x)\psi)$, we have $\varphi \equiv \varphi'$, where $\varphi' = (\exists x)(\neg\psi)$. Thus, $\Gamma \cup \{\varphi'\}$ is satisfiable and, by applying again Theorem 4.6.52, we have that $\Gamma \cup \{\varphi', \langle(\neg\psi)\rangle_{x:=c}\}$ is satisfiable, where c is a constant symbol in $\mathcal{L}^c$ that does not occur in Γ. This implies that $\Gamma \cup \{\langle(\neg\psi)\rangle_{x:=c}\}$ is satisfiable and belongs to $\mathcal{C}$.

Finally, suppose that $\mathcal{L}$ contains $=$, $\Gamma \in \mathcal{C}$ and $\alpha \in \mathrm{INST}_{\mathcal{L}^c,V}(\mathrm{Eq}_{=,\mathcal{L}})$. Since α is logically valid and contains only finitely many constant symbols, it is clear that $\Gamma \cup \{\alpha\} \in \mathcal{C}$. $\square$

Corollary 4.12.26. *Let $\mathcal{L}$ be a first-order language and V be a set of variables. A set Γ of $(\mathcal{L}, V)$-formulas is satisfiable if and only if $\Gamma \subseteq \Gamma'$ for some $(\mathcal{L}^c, V)$-Hintikka set Γ'.*

Proof. Suppose that Γ is satisfiable. By Theorem 4.12.25, Γ belongs to an $(\mathcal{L}^c, V)$-consistency property. Therefore, by Theorem 4.12.23, there is an $(\mathcal{L}^c, V)$-Hintikka set Γ' such that $\Gamma \subseteq \Gamma'$.

Conversely, if $\Gamma \subseteq \Gamma'$, where Γ' is an $(\mathcal{L}^c, V)$-Hintikka set, then Γ is satisfiable because Γ' is satisfiable (by Corollary 4.12.21). $\square$

The next Corollary is a sharpening of Supplements 125 and 131 in that the extra symbols contained by the language $\mathcal{L}'$ mentioned in the supplement are restricted to constant symbols.

Corollary 4.12.27. *Let $\mathcal{L}$ be a first-order language, V be a set of variables and Γ be a set of $(\mathcal{L}, V)$-formulas. Then, if $= \notin \mathcal{L}$, Γ is satisfiable if and only if Γ is satisfiable in a V-Herbrand structure for $\mathcal{L}^c$ and, if $= \in \mathcal{L}$, Γ is satisfiable if and only if Γ is satisfiable in a quotient of V-Herbrand structure for $\mathcal{L}^c$.*

Proof. The result follows from Corollary 4.12.26 and Theorems 4.12.17 and 4.12.20. $\square$

Example 4.12.28. As an application of consistency properties, we give another proof of the Compactness Theorem of First-Order Logic. Let $\mathcal{L}$ be a first-order language. The idea of the proof is to show that the collection $\mathcal{C}$ that consists of those sets $\Gamma \subseteq \mathrm{FORM}_{\mathcal{L}^c}$ such that Γ is a finitely satisfiable set with limited constant symbols is an $(\mathcal{L}^c, \mathrm{VAR})$-consistency property. Since every member of a consistency property is satisfiable, and every finitely satisfiable set of $\mathcal{L}$-formulas belongs to $\mathcal{C}$, this would imply the nontrivial part of the Compactness Theorem. It is technically more convenient to show that $\mathcal{I} = \mathcal{P}(\mathrm{FORM}_{\mathcal{L}^c}) - \mathcal{C}$ is an $(\mathcal{L}^c, \mathrm{VAR})$-inconsistency property. Note that if $\Gamma \subseteq \mathrm{FORM}_{\mathcal{L}^c}$, then $\Gamma \in \mathcal{I}$ if and only if Γ is not finitely satisfiable or infinitely many constant symbols from $\mathcal{L}^c - \mathcal{L}$ occur in Γ.

Let $\Gamma \subseteq \mathrm{FORM}_{\mathcal{L}^c}$. If infinitely many constant symbols from $\mathcal{L}^c - \mathcal{L}$ occur in Γ, then $\Gamma \in \mathcal{I}$. Therefore, for the remainder of this example, we assume that only finitely many constant symbols in $\mathcal{L}^c - \mathcal{L}$ occur in Γ, that is, Γ is a set with limited constant symbols.

It is clear that if $\{\varphi, (\neg\varphi)\} \subseteq \Gamma$ for some atomic formula φ, then $\Gamma \in \mathcal{I}$.

Let now $\varphi \in \Gamma - \mathrm{LIT}_{\mathcal{L}^c}$ be neither a $\boldsymbol{\gamma}$-formula nor a $\boldsymbol{\delta}$-formula and be such that $\Gamma \cup K \in \mathcal{I}$ for every constituent K of φ. Note that since each constituent K is finite, only finitely many constant symbols from $\mathcal{L}^c - \mathcal{L}$ occur in $\Gamma \cup K$. Thus, $\Gamma \cup K$ must not be finitely satisfiable. Using an argument virtually identical to the one used in Example 2.7.23, the reader can easily show that Γ is not finitely satisfiable and hence $\Gamma \in \mathcal{I}$. (Note that this argument does not work for $\boldsymbol{\delta}$-formulas since they have infinitely many constituents.)

Assume now that $\varphi \in \Gamma$ is a $\boldsymbol{\delta}$-formula and $\Gamma \cup K \in \mathcal{I}$, that is, $\Gamma \cup K$ is not finitely satisfiable, for each constituent K of φ. Suppose that $\varphi = (\exists x)\psi$. Our assumption means that for every $t \in \mathrm{TERM}_{\mathcal{L}^c}$, $\Gamma \cup \{\langle\psi\rangle_{x:=t}\}$ is not finitely satisfiable. Let c be a constant symbol in $\mathcal{L}^c - \mathcal{L}$ that does not occur in Γ. Since $\Gamma \cup \{\langle\psi\rangle_{x:=c}\}$ is not finitely satisfiable, there is some finite subset Γ_0 of $\Gamma \cup \{\langle\psi\rangle_{x:=c}\}$ that is not satisfiable. Define $\Gamma_0' = \Gamma_0 \cap \Gamma$. The set $\Gamma_0' \cup \{\varphi\}$ is a finite subset of Γ, and we claim that it is unsatisfiable. Suppose this set were satisfiable. Then, since c does not occur in the set, by Theorem 4.6.52, $\Gamma_0' \cup \{\langle\psi\rangle_{x:=c}\}$ is satisfiable, which implies the satisfiability of Γ_0. If $\varphi = (\neg(\forall x)\psi)$, the argument is similar and is left to the reader.

Next we consider the $\boldsymbol{\gamma}$-formula case. Suppose that $\varphi = (\forall x)\psi \in \Gamma$ and that $\Gamma \cup \{\langle\psi\rangle_{x:=t}\} \in \mathcal{I}$, that is this set is not finitely satisfiable for some $t \in \mathrm{TERM}_{\mathcal{L}^c}$. Thus, there is a finite subset Γ_0 of this set that is unsatisfiable. Define $\Gamma_0' = \Gamma_0 \cap \Gamma$. Then, $\Gamma_0' \cup \{\varphi\}$ is a finite subset of Γ and we claim that it is unsatisfiable. Indeed, if $\Gamma_0' \cup \{\varphi\}$ were satisfiable, this would imply that $\Gamma_0' \cup \{\langle\psi\rangle_{x:=t}\}$ is satisfiable, by Theorem 4.6.51, which in turn would imply that Γ_0 is satisfiable. When $\varphi = (\neg(\exists x)\psi)$, the proof is similar, and takes into account that $(\neg(\exists x)\psi) \equiv (\forall x)(\neg\psi)$ which in turn gives $(\neg(\exists x)\psi) \models \langle(\neg\psi)\rangle_{x:=t}$.

Finally, suppose that $\mathcal{L}$ contains $=$ and that $\Gamma \cup \{\alpha\} \in \mathcal{I}$, that is, $\Gamma \cup \{\alpha\}$ is not finitely satisfiable, where $\alpha \in \mathrm{INST}_{\mathcal{L}^c,\mathrm{VAR}}(\mathrm{Eq}_{=,\mathcal{L}^c})$. Let $\Gamma_0 \subseteq \Gamma \cup \{\alpha\}$ be a finite unsatisfiable set and let $\Gamma_0' = \Gamma_0 \cap \Gamma$. Then Γ_0' is a finite subset of Γ and is not satisfiable. Indeed, if Γ_0' were satisfiable, since α is logically valid by Corollary 4.6.10, this would imply that $\Gamma_0' \cup \{\alpha\}$ is satisfiable and this would in turn imply that Γ_0 is satisfiable. $\square$

4.12.3 *Truth Sets*

The concept of truth set in first-order logic lacks some of the satisfying properties which its propositional counterpart enjoys. Nevertheless, it is worth considering it because of its relationship with Hintikka sets.

Theorem 4.12.29. *Let $\mathcal{L}$ be a first-order language and let V be an $\mathcal{L}$-suitable set of variables. Suppose that $\Gamma \subseteq \mathrm{FORM}_{\mathcal{L}}(V)$ is a set of formulas such that for every $\varphi \in \mathrm{FORM}_{\mathcal{L}}(V)$, $(\neg\varphi) \in \Gamma$ if and only if $\varphi \notin \Gamma$.*

For every formula $\varphi = (\alpha C \beta)$ in $\mathrm{FORM}_{\mathcal{L}}(V)$, where C is a binary connective symbol, the following pairs of statements are equivalent:

- *C_{pd}: $\varphi \in \Gamma$ implies $K \subseteq \Gamma$ for some $K \in \mathtt{D}_{\mathcal{L},V}(\varphi)$;*
- *C_{npu}: $H \subseteq \Gamma$ for some $H \in \mathtt{D}_{\mathcal{L},V}((\neg\varphi))$ implies $(\neg\varphi) \in \Gamma$;*

and

- *C_{pu}: $K \subseteq \Gamma$ for some $K \in \mathtt{D}_{\mathcal{L},V}(\varphi)$ implies $\varphi \in \Gamma$;*
- *C_{npd}: $(\neg\varphi) \in \Gamma$ implies $H \subseteq \Gamma$ for some $H \in \mathtt{D}_{\mathcal{L},V}((\neg\varphi))$.*

Further, if $\varphi = (Qx)\psi$, where Q is a quantifier symbol, then the following pairs of statements are equivalent:

- *Q_{pd}: $\varphi \in \Gamma$ implies $K \subseteq \Gamma$ for some $K \in \mathtt{D}_{\mathcal{L},V}(\varphi)$;*
- *Q_{npu}: $H \subseteq \Gamma$ for some $H \in \mathtt{D}_{\mathcal{L},V}((\neg\varphi))$ implies $(\neg\varphi) \in \Gamma$;*

and

- *Q_{pu}: $K \subseteq \Gamma$ for some $K \in \mathtt{D}_{\mathcal{L},V}(\varphi)$ implies $\varphi \in \Gamma$;*
- *Q_{npd}: $(\neg\varphi) \in \Gamma$ implies $H \subseteq \Gamma$ for some $H \in \mathtt{D}_{\mathcal{L},V}((\neg\varphi))$.*

Proof. We discuss only the equivalence of $\forall_{pd}$ and $\forall_{npu}$. Suppose that $\forall_{pd}$ holds for $\varphi = (\forall x)\psi$. Assume that for some $H \in \mathtt{D}_{\mathcal{L},V}((\neg\varphi))$, we have $H \subseteq \Gamma$ but $(\neg\varphi) \notin \Gamma$. Then, we have $\langle(\neg\psi)\rangle_{x:=t_0} = (\neg\langle\psi\rangle_{x:=t_0}) \in \Gamma$, for some $t_0 \in \mathrm{TERM}_{\mathcal{L}}(V)$ and $\varphi \in \Gamma$. Because of $\forall_{pd}$, we have $\{\langle\psi\rangle_{x:=t} \mid t \in \mathrm{TERM}_{\mathcal{L}}V\} \subseteq \Gamma$, so we have both $(\neg\langle\psi\rangle_{x:=t_0})$ and $\langle\psi\rangle_{x:=t_0}$ in Γ, which is a contradiction.

Conversely, suppose that $\forall_{npu}$ holds and that $\varphi \in \Gamma$. If the single $(\mathcal{L}, V)$-constituent K of φ is not included in Γ, then there is $t_0 \in \mathrm{TERM}_{\mathcal{L}}(V)$ such that $\langle\psi\rangle_{x:=t_0} \notin \Gamma$. Since $\langle\psi\rangle_{x:=t_0} \in \mathrm{FORM}_{\mathcal{L}}(V)$, by Theorem 4.12.4, we have $(\neg\langle\psi\rangle_{x:=t_0}) = \langle(\neg\psi)\rangle_{x:=t_0} \in \Gamma$, so, by $\forall_{npu}$, $(\neg\varphi) \in \Gamma$, which is a contradiction. □

Definition 4.12.30. Let $\mathcal{L}$ be a first-order language and let V be an $\mathcal{L}$-suitable set of variables. A set of formulas $\Gamma \subseteq \mathrm{FORM}_{\mathcal{L}}(V)$ is an $(\mathcal{L}, V)$-*truth set* if the following conditions are satisfied:

(1) for every formula $\varphi \in \mathrm{FORM}_{\mathcal{L}}(V)$, $(\neg\varphi) \in \Gamma$ if and only if $\varphi \notin \Gamma$;
(2) for every positive formula $\varphi \in \mathrm{FORM}_{\mathcal{L}}(V)$ that is not atomic, we have $\varphi \in \Gamma$ if and only if at least one of its $(\mathcal{L}, V)$-constituents is included in Γ;
(3) if $\mathcal{L}$ contains the equality symbol, then $\mathrm{INST}_{\mathcal{L},V}(\mathrm{Eq}_{=,\mathcal{L}}) \subseteq \Gamma$.

We will refer to an $(\mathcal{L}, \emptyset)$-truth set as an $\mathcal{L}$-truth set. ▯

Corollary 4.12.31. *Every $(\mathcal{L}, V)$-truth set is an $(\mathcal{L}, V)$-Hintikka set.*

Proof. The argument is similar to that of Theorem 2.7.15, using Theorem 4.12.29. □

Using Theorem 4.12.29, we can give a useful alternative characterization of truth set.

Theorem 4.12.32. *Let $\mathcal{L}$ be a first-order language and let V be an $\mathcal{L}$-suitable set of variables. A set of formulas $\Gamma \subseteq \mathrm{FORM}_{\mathcal{L}}(V)$ is an $(\mathcal{L}, V)$-truth set if and only if the following conditions are satisfied:*

(1) *for every formula $\varphi \in \mathrm{FORM}_{\mathcal{L}}(V)$, $(\neg\varphi) \in \Gamma$ if and only if $\varphi \notin \Gamma$;*
(2) *for every positive or negated positive formula φ that is not a literal, $K \subseteq \Gamma$ for some $K \in \mathtt{D}_{\mathcal{L},V}(\varphi)$ implies $\varphi \in \Gamma$;*
(3) *if $\mathcal{L}$ contains the equality symbol, then $\mathrm{INST}_{\mathcal{L},V}(Eq_{=,\mathcal{L}}) \subseteq \Gamma$.*

Proof. This follows immediately from Theorem 4.12.29. □

Definition 4.12.33. Let $\mathcal{L}$ be a first-order language and let V be a set of variables. A set Γ of formulas is $(\mathcal{L}, V)$-*maximally satisfiable* if $\Gamma \subseteq \mathrm{FORM}_{\mathcal{L}}(V)$, Γ is satisfiable, and there is no satisfiable set Γ' such that $\Gamma \subset \Gamma' \subseteq \mathrm{FORM}_{\mathcal{L}}(V)$. ▯

One can easily show, using the same argument as in Theorem 2.7.12, that a satisfiable subset Γ of $\mathrm{FORM}_{\mathcal{L}}(V)$ is $(\mathcal{L}, V)$-maximally satisfiable if and only if exactly one of the formulas φ and $(\neg\varphi)$ belongs to Γ for every formula $\varphi \in \mathrm{FORM}_{\mathcal{L}}(V)$. This observation is useful in the next theorem.

Theorem 4.12.34. *Let $\mathcal{L}$ be a first-order language and let V be a set of variables. A set Γ is $(\mathcal{L}, V)$-maximally satisfiable if and only if there is an $\mathcal{L}$-structure $\mathcal{A}$ and $\sigma \in \mathrm{ASSIGN}_{\mathcal{A}}$ such that $\Gamma = \{\varphi \in \mathrm{FORM}_{\mathcal{L}}(V) \mid (\mathcal{A}, \sigma) \models \varphi\}$.*

Proof. Suppose that Γ is $(\mathcal{L}, V)$-maximally satisfiable. Since Γ is satisfiable there is an $\mathcal{L}$-structure $\mathcal{A}$ and $\sigma \in \mathrm{ASSIGN}_{\mathcal{A}}$ such that $\Gamma \subseteq \{\varphi \in \mathrm{FORM}_{\mathcal{L}}(V) \mid (\mathcal{A}, \sigma) \models \varphi\}$. If this inclusion were strict, then Γ would not be $(\mathcal{L}, V)$-maximally satisfiable.

Conversely, suppose that $\Gamma = \{\varphi \in \mathrm{FORM}_{\mathcal{L}}(V) \mid (\mathcal{A}, \sigma) \models \varphi\}$ for some $\mathcal{L}$-structure $\mathcal{A}$ and $\sigma \in \mathrm{ASSIGN}_{\mathcal{A}}$. Obviously, Γ is satisfiable and for every $\varphi \in \mathrm{FORM}_{\mathcal{L}}(V)$, exactly one of φ and $(\neg\varphi)$ is in Γ. Thus, Γ is maximally satisfiable. □

Theorem 4.12.35. *Let $\mathcal{L}$ be a first-order language and let V be an $\mathcal{L}$-suitable set of variables. A set $\Gamma \subseteq \mathrm{FORM}_{\mathcal{L}}(V)$ is an $(\mathcal{L}, V)$-truth set if and only if there is an $\mathcal{L}$-structure $\mathcal{A}$ and $\sigma \in \mathrm{ASSIGN}_{\mathcal{A}}$ such that $(\mathcal{A}, \sigma)$ is V-named and*

$$\Gamma = \{\varphi \in \mathrm{FORM}_{\mathcal{L}}(V) \mid (\mathcal{A}, \sigma) \models \varphi\}.$$

Proof. Let Γ be an $(\mathcal{L}, V)$-truth set and, therefore, an $(\mathcal{L}, V)$-Hintikka set by Corollary 4.12.31. Suppose, initially, that $\mathcal{L}$ does not contain $=$. Theorem 4.12.17 shows that there is a V-Herbrand structure $\mathcal{A}$ for $\mathcal{L}$ and $\sigma \in \mathrm{ASSIGN}_{\mathcal{A}}$ such that $\sigma(x) = x$ for all $x \in V$ and $(\mathcal{A}, \sigma) \models \Gamma$. Conversely, if $(\mathcal{A}, \sigma) \models \varphi \in \mathrm{FORM}_{\mathcal{L}}(V)$, then we must have $\varphi \in \Gamma$, since, otherwise, $(\neg\varphi) \in \Gamma$ and then $(\mathcal{A}, \sigma) \models (\neg\varphi)$, and this is a contradiction. By Example 4.12.10, $(\mathcal{A}, \sigma)$ is V-named, and this gives the desired conclusion.

The argument for the case when $= \in \mathcal{L}$ follows along the same line using Theorem 4.12.20 and Example 4.12.10.

Now suppose that $\Gamma = \{\varphi \in \mathrm{FORM}_{\mathcal{L}}(V) \mid (\mathcal{A}, \sigma) \models \varphi\}$, where $(\mathcal{A}, \sigma)$ is V-named. It is clear $(\neg\varphi) \in \Gamma$ if and only if $\varphi \notin \Gamma$, for all $\varphi \in \mathrm{FORM}_{\mathcal{L}}(V)$. The second condition of the definition of truth set follows from Theorem 4.12.11. The third condition is immediate since every formula in $\mathrm{INST}_{\mathcal{L},V}(\mathrm{Eq}_{=,\mathcal{L}})$ is logically valid. □

Corollary 4.12.36. *Every $(\mathcal{L}, V)$-truth set is an $(\mathcal{L}, V)$-maximally satisfiable set.*

Proof. This follows immediately from Theorems 4.12.34 and 4.12.35. □

The next example shows that, contrary to propositional logic, the converse of Corollary 4.12.36 does not hold.

Example 4.12.37. Let $\mathcal{L}$ be any first-order language that contains relation symbols that are not propositional constants and V be an $\mathcal{L}$-suitable set of variables. Let $R \in \mathcal{L}$ be an n-ary relation symbol with $n > 0$ and let $\Gamma = \{(\exists x)(\neg R(x, t_0, \dots, t_0)\} \cup \{R(t, t_0, \dots, t_0) \mid t \in \mathrm{TERM}_{\mathcal{L}}(V)\}$, where t_0 is an arbitrary, fixed term in $\mathrm{TERM}_{\mathcal{L}}(V)$. In Example 4.12.24, we saw that Γ is a satisfiable set. Let $(\mathcal{A}, \sigma) \models \Gamma$. Then Γ is included in the $(\mathcal{L}, V)$-maximally satisfiable set $\Gamma' = \{\varphi \in \mathrm{FORM}_{\mathcal{L}}(V) \mid (\mathcal{A}, \sigma) \models \varphi\}$. If Γ' were an $(\mathcal{L}, V)$-truth set, then it would have to contain $(\neg R(t, t_0, \dots, t_0))$ for some $t \in \mathrm{TERM}_{\mathcal{L}}(V)$ and this would prevent it from being satisfiable. In fact, we have shown the stronger result that Γ' is not even a Hintikka set even though it is maximally satisfiable. ▯

Theorem 4.12.38. *Let $\mathcal{L}$ be a first-order language and V be an $\mathcal{L}$-suitable set of variables. Then, every $(\mathcal{L}, V)$-Hintikka set is contained in an $(\mathcal{L}, V)$-truth set.*

Proof. Let Γ be an $(\mathcal{L}, V)$-Hintikka set. Suppose first that $= \notin \mathcal{L}$. By Theorem 4.12.17, $(\mathcal{A}, \sigma) \models \Gamma$, where $\mathcal{A}$ is a V-Herbrand structure for $\mathcal{L}$ and $\sigma(x) = x$ for all $x \in V$. Since, by Example 4.12.10, $(\mathcal{A}, \sigma)$ is a V-named pair, the set $\{\varphi \in \mathrm{FORM}_{\mathcal{L}}(V) \mid (\mathcal{A}, \sigma) \models \varphi\}$, which contains Γ, is a truth set, by Theorem 4.12.35.

The case when $= \in \mathcal{L}$ is handled similarly, using Theorem 4.12.20. □

In propositional logic, every satisfiable set is contained in a maximally satisfiable set and, therefore, in a truth set. In first-order logic, there are maximally satisfiable sets that are not truth sets, so the situation is more complicated. Nevertheless, we have the following result.

Theorem 4.12.39. *Let $\mathcal{L}$ be a first-order language and V be a set of variables. For every satisfiable set of formulas $\Gamma \subseteq \mathrm{FORM}_{\mathcal{L}}(V)$, there is an $(\mathcal{L}^c, V)$-truth set Γ' with $\Gamma \subseteq \Gamma'$.*

Proof. We saw in Theorem 4.12.25, that every satisfiable set of formulas $\Gamma \subseteq \mathrm{FORM}_{\mathcal{L}}(V)$ is a member of an $(\mathcal{L}^c, V)$-consistency property. Since V is $\mathcal{L}^c$-suitable (even if V is not $\mathcal{L}$-suitable), by Theorem 4.12.23, Γ is included in an $(\mathcal{L}^c, V)$-Hintikka set Γ'. Finally, by Theorem 4.12.38, Γ' is contained in an $(\mathcal{L}^c, V)$-truth set. $\square$

4.12.4 *Hintikka Sets of Signed Formulas*

Definition 4.12.40. Let $\mathcal{L}$ be a first-order language and V be an $\mathcal{L}$-suitable set of variables. Further, let $t_0, t_1, \ldots$ be the enumeration of $\mathrm{TERM}_{\mathcal{L}}(V)$ in the standard order. The mapping

$$\begin{aligned}\mathtt{d}_{\mathcal{L},V} &: (\mathrm{SFORM}_{\mathcal{L}} - (\mathbf{Bool} \times \mathrm{AFORM}_{\mathcal{L}}))\\ &\longrightarrow \mathrm{Seq}(\mathcal{P}(\mathrm{SFORM}_{\mathcal{L}})) \cup \mathrm{ISeq}(\mathcal{P}(\mathrm{SFORM}_{\mathcal{L}}))\end{aligned}$$

is given by the following table:

Signed Formula $b\alpha$	$\mathtt{d}_{\mathcal{L},V}(b\alpha)$
$\mathbf{T}(\neg\varphi)$	$(\{\mathbf{F}\varphi\})$
$\mathbf{F}(\neg\varphi)$	$(\{\mathbf{T}\varphi\})$
$\mathbf{T}(\varphi \wedge \psi)$	$(\{\mathbf{T}\varphi, \mathbf{T}\psi\})$
$\mathbf{F}(\varphi \wedge \psi)$	$(\{\mathbf{F}\varphi\}, \{\mathbf{F}\psi\})$
$\mathbf{T}(\varphi \vee \psi)$	$(\{\mathbf{T}\varphi\}, \{\mathbf{T}\psi\})$
$\mathbf{F}(\varphi \vee \psi)$	$(\{\mathbf{F}\varphi, \mathbf{F}\psi\})$
$\mathbf{T}(\varphi \rightarrow \psi)$	$(\{\mathbf{F}\varphi\}, \{\mathbf{T}\psi\})$
$\mathbf{F}(\varphi \rightarrow \psi)$	$(\{\mathbf{T}\varphi, \mathbf{F}\psi\})$
$\mathbf{T}(\varphi \leftrightarrow \psi)$	$(\{\mathbf{T}\varphi, \mathbf{T}\psi\}, \{\mathbf{F}\varphi, \mathbf{F}\psi\})$
$\mathbf{F}(\varphi \leftrightarrow \psi)$	$(\{\mathbf{T}\varphi, \mathbf{F}\psi\}, \{\mathbf{F}\varphi, \mathbf{T}\psi\})$
$\mathbf{T}(\forall x)\varphi$	$(\{\mathbf{T}\langle\varphi\rangle_{x:=t} \mid t \in \mathrm{TERM}_{\mathcal{L}}(V)\})$
$\mathbf{F}(\forall x)\varphi$	$(\{\mathbf{F}\langle\varphi\rangle_{x:=t_0}\}, \{\mathbf{F}\langle\varphi\rangle_{x:=t_1}\}, \ldots)$
$\mathbf{T}(\exists x)\varphi$	$(\{\mathbf{T}\langle\varphi\rangle_{x:=t_0}\}, \{\mathbf{T}\langle\varphi\rangle_{x:=t_1}\}, \ldots)$
$\mathbf{F}(\exists x)\varphi$	$(\{\mathbf{F}\langle\varphi\rangle_{x:=t} \mid t \in \mathrm{TERM}_{\mathcal{L}}(V)\})$

Let $b\varphi$ be a signed formula such that φ is not a variable. The $(\mathcal{L}, V)$-*constituent sequence* of $b\varphi$ is the sequence $\mathtt{d}_{\mathcal{L},V}(b\varphi)$. The $(\mathcal{L}, V)$-*constituent set* of $b\varphi$ is the set $\mathtt{D}_{\mathcal{L},V}(b\varphi)$ that consists of all sets of formulas that occur in $\mathtt{d}_{\mathcal{L},V}(b\varphi)$. Every such set of formulas is called an $(\mathcal{L}, V)$-*constituent* of $b\varphi$. ▯

Note that if $b'\psi$ belongs to an $(\mathcal{L}, V)$-constituent of a signed formula $b\varphi$, then $\psi = \langle\theta\rangle_{x:=t}$, where θ is an immediate subformula of φ. Therefore, we have $\parallel b'\psi \parallel < \parallel b\varphi \parallel$.

Theorem 4.12.41. *Let $\mathcal{L}$ be a first-order language, and let V be an $\mathcal{L}$-suitable set of variables. If $b'\psi \in K \in \mathtt{D}_{\mathcal{L},V}(b\varphi)$, where $\varphi \in \mathrm{FORM}_{\mathcal{L}} - \mathit{LIT}_{\mathcal{L}}$, then $\mathtt{FV}(b'\psi) \subseteq \mathtt{FV}(b\varphi) \cup V$.*

Proof. The proof can be done by inspecting the definition of the constituents of $b\varphi$ and applying Corollary 4.6.48. □

Corollary 4.12.42. *Any member of an $(\mathcal{L}, V)$-constituent of a signed formula in $\mathrm{SFORM}_{\mathcal{L}}(V)$ is itself in $\mathrm{SFORM}_{\mathcal{L}}(V)$.*

Proof. The statement follows from the previous theorem. □

Corollary 4.12.43. *Let $\mathcal{L}$ be a first-order language that contains a constant symbol. If $b\varphi$ is a closed signed $\mathcal{L}$-formula that is not a literal, then every signed formula in an $\mathcal{L}$-constituent of $b\varphi$ is closed.*

Proof. The statement follows immediately from Corollary 4.12.42 by taking $V = \emptyset$. □

The analogue of Theorem 4.12.11 for signed formulas is given next.

Theorem 4.12.44. *Let $\mathcal{L}$ be a first-order language and V be an $\mathcal{L}$-suitable set of variables. If $\mathcal{A}$ is an $\mathcal{L}$-structure, $\sigma \in \mathrm{ASSIGN}_{\mathcal{A}}$, and $(\mathcal{A}, \sigma)$ is V-named, then for all signed $\mathcal{L}$-formulas $b\alpha$ such that α is not atomic, we have $(\mathcal{A}, \sigma) \models b\alpha$ if and only if $(\mathcal{A}, \sigma) \models K$ for some $(\mathcal{L}, V)$-constituent K of $b\alpha$.*

Proof. The proof is similar to that of Theorem 4.12.11 and is left to the reader. □

Corollary 4.12.45. *Let $\mathcal{L}$ be a first-order language that contains at least one constant symbol. If $\mathcal{A}$ is a named $\mathcal{L}$-structure, then for all* *signed $\mathcal{L}$-formulas $b\alpha$ where α is closed and is not atomic, $\mathcal{A} \models b\alpha$ if and only if $\mathcal{A} \models K$ for some $\mathcal{L}$-constituent K of $b\alpha$.*

Proof. Again, the proof is similar to the proof of Corollary 4.12.12. □

Corollary 4.12.46. *Let V be an $\mathcal{L}$-suitable set of variables, where $\mathcal{L}$ is a first-order language. Also, let $\mathcal{A}$ be a V-Herbrand structure for $\mathcal{L}$. If $\sigma \in \mathrm{ASSIGN}_{\mathcal{A}}$ is such that $\sigma(x) = x$ for all $x \in V$, then for all signed $\mathcal{L}$-formulas $b\alpha$ such that α is not atomic, $(\mathcal{A}, \sigma) \models b\alpha$ if and only if $(\mathcal{A}, \sigma) \models K$ for some $(\mathcal{L}, V)$-constituent K of $b\alpha$.*

If $\mathcal{L}$ contains a constant symbol and $\mathcal{A}_0$ is a Herbrand structure for $\mathcal{L}$, then, for every signed $\mathcal{L}$-formula $b\alpha$ such that α is closed and not atomic, we have $\mathcal{A}_0 \models b\alpha$ if and only if $\mathcal{A}_0 \models K$ for some $\mathcal{L}$-constituent K of $b\alpha$.

Proof. The proof is similar to the proof of Corollary 4.12.13 □

Corollary 4.12.47. *Let V be an $\mathcal{L}$-suitable set of variables, where $\mathcal{L}$ is a first-order language. Also, let $\mathcal{A}$ be a V-Herbrand structure for $\mathcal{L}$, ρ be a congruence of $\mathcal{A}$ and $\mathcal{B} = \mathcal{A}/\rho$. If $\sigma \in \mathrm{ASSIGN}_{\mathcal{B}}$ is such that $\sigma(x) = [x]$ for all $x \in V$, then for all signed $\mathcal{L}$-formulas $b\alpha$ such that α is not atomic, $(\mathcal{B}, \sigma) \models b\alpha$ if and only if $(\mathcal{B}, \sigma) \models K$ for some $(\mathcal{L}, V)$-constituent K of $b\alpha$.*

If $\mathcal{L}$ contains a constant symbol, $\mathcal{A}_0$ is a Herbrand structure for $\mathcal{L}$, ρ is a congruence of $\mathcal{A}_0$, and $\mathcal{B}_0 = \mathcal{A}_0/\rho$, then for all signed formulas $b\alpha$ such that α is closed and not atomic, we have $\mathcal{B}_0 \models b\alpha$ if and only if $\mathcal{B}_0 \models K$ for some constituent K of $b\alpha$.

Proof. The proof is similar to that of Corollary 4.12.14. □

Theorem 4.12.48. *Let $\mathcal{L}$ be a first-order language, V be an $\mathcal{L}$-suitable set of variables, and $b\varphi$ be a signed $\mathcal{L}$-formula. Suppose that $\mathcal{A}$ is an $\mathcal{L}$-structure and $\sigma \in \mathrm{ASSIGN}_{\mathcal{A}}$.*

(1) *If $b\varphi$ is neither a $\boldsymbol{\gamma}$-signed formula nor a $\boldsymbol{\delta}$-signed formula, then $(\mathcal{A}, \sigma) \models b\varphi$ if and only if $(\mathcal{A}, \sigma) \models K$ for some $(\mathcal{L}, V)$-constituent K of $b\varphi$.*
(2) *If φ is a $\boldsymbol{\gamma}$-signed formula, then $(\mathcal{A}, \sigma) \models b\varphi$ implies $(\mathcal{A}, \sigma) \models K$ for the $(\mathcal{L}, V)$-constituent K of $b\varphi$.*
(3) *If $b\varphi$ is a $\boldsymbol{\delta}$-signed formula and K is an $(\mathcal{L}, V)$-constituent of $b\varphi$, then $(\mathcal{A}, \sigma) \models K$ implies $(\mathcal{A}, \sigma) \models b\varphi$.*

Proof. This result can be obtained from Theorem 4.12.15 and Definition 4.5.69. We discuss here only one case. Suppose that $b\varphi$ is the $\boldsymbol{\delta}$-signed formula $\mathbf{F}(\forall x)\psi$ and $(\mathcal{A},\sigma) \models \mathbf{F}\langle\psi\rangle_{x:=t}$ for some $\mathcal{L}$-structure $\mathcal{A}$ and $\sigma \in \text{ASSIGN}_{\mathcal{A}}$. Then, $(\mathcal{A},\sigma) \models (\neg\langle\psi\rangle_{x:=t})$. By Theorem 4.6.50, we have $(\mathcal{A},\sigma) \models \langle(\neg\psi)\rangle_{x:=t}$, which implies $(\mathcal{A},\sigma) \models (\neg(\forall x)\psi)$, by the third part of Theorem 4.12.15. Again, by Definition 4.5.69, we have $(\mathcal{A},\sigma) \models \mathbf{F}(\forall x)\psi$. □

Definition 4.12.49. Let $\mathcal{L}$ be a first-order language, V be an $\mathcal{L}$-suitable set of variables, and Δ be a set of signed $\mathcal{L}$-formulas such that $\mathtt{FV}(\Delta) \subseteq V$.

If $\mathcal{L}$ does not contain the equality symbol, then Δ is an $(\mathcal{L},V)$-*Hintikka set* if $\mathtt{FV}(\Delta) \subseteq V$ and the following conditions are satisfied:

(1) for every atomic $\mathcal{L}$-formula φ, at most one of the signed formulas $\mathbf{T}\varphi, \mathbf{F}\varphi$ is in Δ;
(2) if $b\varphi \in \Delta$ and φ is not atomic, then there is a constituent K of $b\varphi$ such that $K \subseteq \Delta$.

If $\mathcal{L}$ does contain the equality symbol, then Δ is an $(\mathcal{L},V)$-Hintikka set if Δ satisfies the previous two conditions and $\mathbf{T}\varphi \in \Delta$ for every $\varphi \in \text{INST}_{\mathcal{L},V}(\text{Eq}_{=,\mathcal{L}})$.

We refer to an $(\mathcal{L},\emptyset)$-Hintikka set of signed formulas as an $\mathcal{L}$-Hintikka set of signed formulas. ▯

Note that an $\mathcal{L}$-Hintikka set of signed formulas consists of signed closed formulas.

Theorem 4.12.50. *Let $\mathcal{L}$ be a first-order language without equality, V be an $\mathcal{L}$-suitable set of variables, and Δ be an $(\mathcal{L},V)$-Hintikka set. Then, Δ is satisfiable. In fact, there is a V-Herbrand structure $\mathcal{A}$ for $\mathcal{L}$ and $\sigma \in \text{ASSIGN}_{\mathcal{A}}$ such that $\sigma(x) = x$ for all $x \in V$ and $(\mathcal{A},\sigma) \models \Delta$.*

Proof. Let $\mathcal{A} = \mathsf{STR}_{\mathcal{L},V}(S)$ be the V-Herbrand structure for $\mathcal{L}$ determined by $S = \{\varphi \in \text{AFORM}_{\mathcal{L}} \mid \mathbf{T}\varphi \in \Delta\}$ (as introduced in Theorem 4.10.6) and let $\sigma \in \text{ASSIGN}_{\mathcal{A}}$ be an assignment such that $\sigma(x) = x$ for every $x \in V$.

We prove by course-of-values induction on $\| b\varphi \|$ that if $b\varphi \in \Delta$, then $(\mathcal{A},\sigma) \models b\varphi$. Suppose that $b\varphi \in \Delta$ and that the result holds for all $b'\psi$ with $\| b'\psi \| < \| b\varphi \|$. First, suppose that φ is atomic. If $b = \mathbf{T}$, then $\varphi \in S$, so, $(\mathcal{A}, \sigma) \models \varphi$ by Lemma 4.10.11, that is, $(\mathcal{A}, \sigma) \models b\varphi$. If $b = \mathbf{F}$, then $\mathbf{T}\varphi \notin \Delta$, by the definition of Hintikka set, so $\varphi \notin S$. Consequently, $(\mathcal{A}, \sigma) \not\models \varphi$, by the same lemma, which allows us to conclude that $(\varphi, \sigma) \models \mathbf{F}\varphi = b\varphi$.

Now suppose that φ is not atomic. Then, by the definition of Hintikka set, there is a constituent K of $b\varphi$ such that $K \subseteq \Delta$. Since the norm of each signed formula in K is less than the norm of $b\varphi$, we have $(\mathcal{A}, \sigma) \models K$, by inductive hypothesis. Thus, by Corollary 4.12.46, $(\mathcal{A}, \sigma) \models b\varphi$. □

In preparation for extending Theorem 4.12.50 to languages with equality, we need the following technical result.

Lemma 4.12.51. *If Δ is an $(\mathcal{L}, V)$-Hintikka set of signed formulas for some first-order language $\mathcal{L}$ and some $\mathcal{L}$-suitable set of variables V and Δ contains a formula $\mathbf{F}(\varphi_0 \wedge \varphi_1 \wedge \cdots \wedge \varphi_{n-1})$, then Δ contains at least one formula $\mathbf{F}\varphi_i$ with $0 \leq i \leq n-1$.*

Proof. The argument is by induction on n and is left to the reader. □

Lemma 4.12.52. *Let $\mathcal{L}$ be a first-order language with equality and let V be an $\mathcal{L}$-suitable set of variables. Suppose that Δ is an $(\mathcal{L}, V)$-Hintikka set and $S = \{\varphi \in \mathrm{AFORM}_{\mathcal{L}-\{=\}}(V) \mid \mathbf{T}\varphi \in \Delta\}$. Define the relation ρ on $\mathrm{TERM}_{\mathcal{L}}(V)$ by $t\rho s$ if and only if $\mathbf{T}(t = s) \in \Delta$. Then, ρ is a congruence of the Herbrand structure $\mathcal{A} = \mathsf{STR}_{\mathcal{L},V}(S)$.*

Proof. The proof is similar to the one given for Lemma 4.12.19 and it is left to the reader. □

The counterpart of Theorem 4.12.50 for languages with equality is given next.

Theorem 4.12.53. *Let $\mathcal{L}$ be a first-order language with equality, V be an $\mathcal{L}$-suitable set of variables, and Δ be an $(\mathcal{L}, V)$-Hintikka set of signed formulas. Then, Δ is satisfiable. In fact, there is a V-Herbrand structure $\mathcal{A}$ for $\mathcal{L}$, a congruence ρ of $\mathcal{A}$ and $\sigma \in \mathrm{ASSIGN}_{\mathcal{A}/\rho}$ such that $\sigma(x) = [x]_\rho$ for all $x \in V$ and $(\mathcal{A}/\rho, \sigma) \models \Delta$.*

Proof. The proof is along the lines of the proof of Theorem 4.12.20 and is left to the reader. □

4.13 Theories

In this section, we examine the connection between classes of $\mathcal{L}$-structures and classes of $\mathcal{L}$-sentences. We will use Σ with or without subscripts or superscripts to denote sets of sentences.

Definition 4.13.1. Let $\mathcal{L}$ be a first-order language. An *$\mathcal{L}$-theory* is a set of $\mathcal{L}$-sentences Σ such that for every $\mathcal{L}$-sentence φ, $\Sigma \models \varphi$ implies $\varphi \in \Sigma$. ▯

In other words, an $\mathcal{L}$-theory is a set of $\mathcal{L}$-sentences closed under logical implication.

Example 4.13.2. The set $\mathrm{SENT}_{\mathcal{L}}$ of all $\mathcal{L}$-sentences is clearly an $\mathcal{L}$-theory.

$\Sigma_0 = \{\varphi \in \mathrm{SENT}_{\mathcal{L}} \mid \; \models \varphi\}$ is an $\mathcal{L}$-theory that is a subset of any other $\mathcal{L}$-theory. To prove that Σ_0 is a theory, let φ_0 be an $\mathcal{L}$-sentence such that $\Sigma_0 \models \varphi_0$. Since Σ_0 consists of valid $\mathcal{L}$-sentences, we have $\mathcal{A} \models \Sigma_0$ and so $\mathcal{A} \models \varphi_0$ for every $\mathcal{L}$-structure $\mathcal{A}$, hence φ_0 is a valid $\mathcal{L}$-sentence and therefore is in Σ_0.

Let Σ be an arbitrary $\mathcal{L}$-theory. If $\varphi_0 \in \Sigma_0$, then it is a logically valid sentence. Therefore, $\Sigma \models \varphi_0$, which implies $\varphi_0 \in \Sigma$. This allows us to conclude that $\Sigma_0 \subseteq \Sigma$. ▯

In the next theorem, we will use the notion of closure system from Section 1.3.

Theorem 4.13.3. *Let $\mathcal{L}$ be a first-order language. The collection of all $\mathcal{L}$-theories is a closure system on* $\mathrm{SENT}_{\mathcal{L}}$*.*

Proof. Note that the union of all $\mathcal{L}$-theories is the $\mathcal{L}$-theory of all $\mathcal{L}$-sentences discussed in Example 4.13.2. Let $\mathcal{D}$ be a nonempty collection of $\mathcal{L}$-theories and let $\Sigma = \bigcap \mathcal{D}$. If $\Sigma \models \varphi$, then for every $\Sigma' \in \mathcal{D}$, we have $\Sigma' \models \varphi$ by the fifth part of Theorem 4.5.51 which implies $\varphi \in \Sigma'$. Thus, $\varphi \in \bigcap \mathcal{D} = \Sigma$. □

It follows from Theorem 4.13.3 that for every $\Sigma \subseteq \mathrm{SENT}_{\mathcal{L}}$ there is a smallest $\mathcal{L}$-theory that contains Σ which can be obtained as the intersection of all $\mathcal{L}$-theories containing Σ. The next theorem gives an explicit characterization of this smallest theory.

Theorem 4.13.4. *Let Σ be a set of $\mathcal{L}$-sentences, where $\mathcal{L}$ is a first-order language. Then, $\Sigma' = \{\varphi \in \mathrm{SENT}_{\mathcal{L}} \mid \Sigma \models \varphi\}$ is the smallest $\mathcal{L}$-theory that contains Σ.*

Proof. To verify that Σ' is an $\mathcal{L}$-theory, let ψ be an $\mathcal{L}$-sentence such that $\Sigma' \models \psi$. If $\mathcal{A} \models \Sigma$, then, by the definition of Σ', we have $\mathcal{A} \models \Sigma'$ which implies $\mathcal{A} \models \psi$, so $\psi \in \Sigma'$.

Let Σ_1 be an $\mathcal{L}$-theory such that $\Sigma \subseteq \Sigma_1$ and let φ be a sentence in Σ'. Since $\Sigma \subseteq \Sigma_1$ it is clear that $\Sigma_1 \models \varphi$, hence $\varphi \in \Sigma_1$ because Σ_1 is a theory. Thus, $\Sigma' \subseteq \Sigma_1$ for any $\mathcal{L}$-theory Σ_1 that contains Σ. $\square$

We will denote the set Σ' introduced in Theorem 4.13.4 by $\mathrm{Cn}^{\mathcal{L}}(\Sigma)$ and we will refer to this set as the *$\mathcal{L}$-theory generated by* Σ.

Corollary 4.13.5. *Let $\mathcal{L}$ be a first-order language. If Σ, Σ' are sets of $\mathcal{L}$-sentences, we have:*

(1) *if $\Sigma \subseteq \Sigma'$, then $Cn^{\mathcal{L}}(\Sigma) \subseteq Cn^{\mathcal{L}}(\Sigma')$;*
(2) *$Cn^{\mathcal{L}}(Cn^{\mathcal{L}}(\Sigma)) = Cn^{\mathcal{L}}(\Sigma)$.*

Proof. The corollary follows from Theorem 4.13.4 by observing that $\mathrm{Th}^{\mathcal{L}}$ is the closure operator associated with the closure system of $\mathcal{L}$-theories (see Lemma 1.3.10). $\square$

Theorem 4.13.6. *Let $\mathcal{L}$ be a first-order language and let $\mathfrak{A}$ be a collection of $\mathcal{L}$-structures. The set of $\mathcal{L}$-sentences $\Sigma = \{\varphi \in \mathrm{SENT}_{\mathcal{L}} \mid \mathcal{A} \models \varphi$ for all $\mathcal{A} \in \mathfrak{A}\}$ is an $\mathcal{L}$-theory.*

Proof. Let ψ be an $\mathcal{L}$-sentence such that $\Sigma \models \psi$. If $\mathcal{A} \in \mathfrak{A}$, then $\mathcal{A} \models \Sigma$, so $\mathcal{A} \models \psi$ which implies $\psi \in \Sigma$. $\square$

Definition 4.13.7. Let $\mathcal{L}$ be a first-order language and let $\mathfrak{A}$ be a collection of $\mathcal{L}$-structures. The set of $\mathcal{L}$-sentences Σ introduced in Theorem 4.13.6 is called the *theory of* $\mathfrak{A}$ and is denoted $\mathrm{Th}^{\mathcal{L}}(\mathfrak{A})$.

If $\mathfrak{A} = \{\mathcal{A}\}$, we denote $\mathrm{Th}^{\mathcal{L}}(\mathfrak{A})$ by $\mathrm{Th}^{\mathcal{L}}(\mathcal{A})$ and we refer to it as the *theory of* $\mathcal{A}$. ▯

Example 4.13.8. Let $\mathcal{L}_{pra} = \{0, s, +, <, =\}$ be the first-order language introduced in Example 4.2.3. The $\mathcal{L}_{pra}$-structure $\mathcal{A}_{pra}$, was introduced as a reduct of the standard model of arithmetic $\mathcal{A}_{ar}$ and is given by $|\mathcal{A}_{pra}| = \mathbf{N}$, $0^{\mathcal{A}_{pra}} = 0$, $s^{\mathcal{A}_{pra}}$ is the *successor function* on $\mathbf{N}$ given by $s^{\mathcal{A}_{pra}}(n) = n+1$, $+^{\mathcal{A}_{pra}}$ is the addition function on $\mathbf{N}$, and $<^{\mathcal{A}_{pra}} = \{(x,y) \in \mathbf{N}^2 \mid x \text{ is less than } y\}$. The theory $\mathrm{Th}^{\mathcal{L}_{pra}}(\mathcal{A}_{pra})$ is known as *Presburger arithmetic.* □

Example 4.13.9. The theory $\mathrm{Th}^{\mathcal{L}_{ar}}(\mathcal{A}_{ar})$, where $\mathcal{A}_{ar}$ is the structure introduced in Example 4.4.4, is known as the *theory of arithmetic* and we will denote it by Th_{ar}. The term "theory of arithmetic" introduced in this example, justifies the term "standard model of arithmetic" introduced much earlier.

Recall that $s^n(0)$ is the $\mathcal{A}_{ar}$-term $\underbrace{s(s(\cdots s(}_{n}0)\cdots))$ for $n \in \mathbf{N}$.

The following formulas constitute useful examples of members of the theory of arithmetic.

$$(\forall x)(\neg(x < 0)) \tag{4.11}$$

$$(\forall x)(\forall y)(x < y \vee x = y \vee y < x) \tag{4.12}$$

$$\begin{aligned}(\forall x)((x < s^{n+1}(0)) \to (x = 0 \vee x = s(0) \vee \cdots \\ \cdots \vee x = s^n(0))) \text{ for } n \in \mathbf{N}\end{aligned} \tag{4.13}$$

$$s^n(0) \neq s^m(0) \text{ for } n, m \in \mathbf{N}, n \neq m \tag{4.14}$$

$$s^n(0) < s^m(0) \text{ for } n, m \in \mathbf{N}, n < m \tag{4.15}$$

$$s^n(0) + s^m(0) = s^{n+m}(0) \text{ for } n, m \in \mathbf{N} \tag{4.16}$$

$$s^n(0) \cdot s^m(0) = s^{nm}(0) \text{ for } n, m \in \mathbf{N} \tag{4.17}$$

□

Definition 4.13.10. A *nonstandard model of arithmetic* is a model of the theory of arithmetic that is not isomorphic to the standard model $\mathcal{A}_{ar}$. □

We will now show the existence of countable nonstandard models of arithmetic. This is an application of the Compactness Theorem.

Theorem 4.13.11. *There is a countable nonstandard model of arithmetic.*

Proof. Let $\mathcal{L} = \mathcal{L}_{ar} \cup \{c\}$ be the first-order language obtained by adding the constant symbol c to $\mathcal{L}_{ar}$ and let Σ be $\mathrm{Th}_{ar} \cup \{s^n(0) \neq c \mid n \in \mathbf{N}\}$. It is easy to prove by induction on n that $(s^n(0))^{\mathcal{A}_{ar}} = n$ for $n \in \mathbf{N}$.

Let Σ_0 be a finite subset of Σ and let m be the largest n such that $s^n(0) \neq c$ belongs to Σ_0. (If there is no such n, we define m to be -1.) Let $\mathcal{B}$ be the expansion of $\mathcal{A}_{ar}$ to $\mathcal{L}$ obtained by defining $c^{\mathcal{B}} = m+1$. Clearly, $\mathcal{B}$ is a model of Σ_0, so every finite subset of Σ_0 has a model. By the Compactness Theorem (Theorem 4.10.43), there is a model $\mathcal{B}_0$ of Σ. By Theorem 4.10.58, we can assume that $\mathcal{B}_0$ is countable. Let $\mathcal{B}_1$ be the reduct of $\mathcal{B}_0$ to $\mathcal{L}_{ar}$, which is also countable. Then, by Corollary 4.5.25, $\mathcal{B}_1$ is a model of arithmetic. We will show that $\mathcal{B}_1$ is not isomorphic to $\mathcal{A}_{ar}$.

Suppose that $h : \mathbf{N} \longrightarrow |\mathcal{B}_1|$ were an isomorphism. We will prove by induction on n that $h(n) = (s^n(0))^{\mathcal{B}_1}$. For $n = 0$, we have $h(0) = h(0^{\mathcal{A}_{ar}}) = 0^{\mathcal{B}_1}$, by definition of morphism. Suppose that the equality holds for n. Then,

$$\begin{aligned} h(n+1) &= h(s^{\mathcal{A}_{ar}}(n)) \\ &= s^{\mathcal{B}_1}(h(n)) \\ &= s^{\mathcal{B}_1}((s^n(0))^{\mathcal{B}_1}) \\ &\quad \text{(by inductive hypothesis)} \\ &= (s^{n+1}(0))^{\mathcal{B}_1} \\ &\quad \text{(by Theorem 4.5.5).} \end{aligned}$$

Since $c^{\mathcal{B}_0} \neq (s^n(0))^{\mathcal{B}_0} = (s^n(0))^{\mathcal{B}_1} = h(n)$, it follows that h cannot be an onto mapping, so h is not an isomorphism. $\square$

Theorem 4.13.12. *Let $\mathcal{B}$ be a model of arithmetic and let $A = \{(s^n(0))^{\mathcal{B}} \mid n \in \mathbf{N}\}$. Then, A is the domain of a substructure $\mathcal{A}$ of $\mathcal{B}$ that is isomorphic to $\mathcal{A}_{ar}$. Moreover, for every $b \in |\mathcal{B}| - |\mathcal{A}|$, we have $a <^{\mathcal{B}} b$ for every $a \in |\mathcal{A}|$.*

Proof. Define a function $h : \mathbf{N} \longrightarrow |\mathcal{B}|$ by $h(n) = (s^n(0))^{\mathcal{B}}$, for $n \in \mathbf{N}$. We show that h is a monomorphism between $\mathcal{A}_{ar}$ and $\mathcal{B}$.

We have $h(0^{\mathcal{A}_{ar}}) = h(0) = 0^{\mathcal{B}}$ by the definition of h. Note that for $n \in \mathbf{N}$

$$\begin{aligned} h(s^{\mathcal{A}_{ar}}(n)) &= h(n+1) \\ &= (s^{n+1}(0))^{\mathcal{B}} \\ &\quad \text{(by definition of } h) \\ &= s^{\mathcal{B}}((s^n(0))^{\mathcal{B}}) \\ &= s^{\mathcal{B}}(h(n)). \end{aligned}$$

Since $\mathcal{B}$ is a model of the formulas (4.15)–(4.17), it follows that h is a morphism. Injectivity of h follows from the fact that $\mathcal{B}$ is a model of formula (4.14). By Theorem 4.4.26, $A = \text{Ran}(h)$ is the domain of a unique substructure $\mathcal{A}$ of $\mathcal{B}$. Since h is a bijection between $\mathbf{N}$ and A, it follows that $\mathcal{A}$ is isomorphic to $\mathcal{A}_{ar}$.

Let b be in $|\mathcal{B}| - |\mathcal{A}|$ and let $a \in |\mathcal{A}|$. Suppose initially that $a = 0^{\mathcal{B}}$. Since $\mathcal{B}$ is a model of formulas (4.11) and (4.12) and $b \notin |\mathcal{A}|$, it follows that $a <^{\mathcal{B}} b$. If $a \neq 0^{\mathcal{B}}$, then $a = (s^{n+1}(0))^{\mathcal{B}}$ for some $n \in \mathbf{N}$. Note that we can have $a <^{\mathcal{B}} b$, $a = b$, or $b <^{\mathcal{B}} a$, because $\mathcal{B}$ is a model of formula (4.12). If a were equal to b, this would imply $b \in |\mathcal{A}|$, which is not the case. If $b <^{\mathcal{B}} a = (s^{n+1}(0))^{\mathcal{B}}$, the fact that $\mathcal{B}$ is a model of formula (4.13) would imply $b = (s^j(0))^{\mathcal{B}}$ for some $j \leq n$ and this would mean that $b \in |\mathcal{A}|$. Thus, $a <^{\mathcal{B}} b$. $\square$

Definition 4.13.13. Let $\mathcal{B}$ be a model of arithmetic. The *standard part of* $\mathcal{B}$ is the substructure $\mathcal{A}$ introduced in Theorem 4.13.12. Any element of $|\mathcal{A}|$ is called a *standard element*, while every member of $|\mathcal{B}| - |\mathcal{A}|$ will be referred to as a *nonstandard element* of $\mathcal{B}$. ▯

Definition 4.13.14. Let $\mathcal{L}$ be a first-order language and let $\Sigma \subseteq \text{SENT}_{\mathcal{L}}$. The set of $\mathcal{L}$-structures that are models of Σ will be denoted by $\text{Mod}^{\mathcal{L}}(\Sigma)$.

A class of $\mathcal{L}$-structures is called an *$\mathcal{L}$-first-order property* if it equals $\text{Mod}^{\mathcal{L}}(\Sigma)$ for some set of $\mathcal{L}$-sentences Σ. ▯

Theorem 4.13.15. *Let $\mathcal{L}$ be a first-order language, Σ, Σ' be sets of $\mathcal{L}$-sentences and $\mathfrak{A}, \mathfrak{A}'$ be collections of $\mathcal{L}$-structures. Then the following hold:*

(1) *If $\Sigma \subseteq \Sigma'$, then $Mod^{\mathcal{L}}(\Sigma') \subseteq Mod^{\mathcal{L}}(\Sigma)$.*
(2) *If $\mathfrak{A} \subseteq \mathfrak{A}'$, then $Th^{\mathcal{L}}(\mathfrak{A}') \subseteq Th^{\mathcal{L}}(\mathfrak{A})$.*

(3) $\Sigma \subseteq Th^{\mathcal{L}}(Mod^{\mathcal{L}}(\Sigma))$.
(4) $\mathfrak{A} \subseteq Mod^{\mathcal{L}}(Th^{\mathcal{L}}(\mathfrak{A}))$.

Proof. The first two parts of the theorem are immediate consequences of the definitions of $\mathrm{Th}^{\mathcal{L}}$ and $\mathrm{Mod}^{\mathcal{L}}$. To prove the third part, let $\varphi \in \Sigma$. If $\mathcal{A} \models \Sigma$, then clearly $\mathcal{A} \models \varphi$, so $\varphi \in \mathrm{Th}^{\mathcal{L}}(\mathrm{Mod}^{\mathcal{L}}(\Sigma))$. The fourth part is equally simple and is left to the reader. □

Theorem 4.13.16. *Let $\mathcal{L}$ be a first-order language and let Σ be a set of $\mathcal{L}$-sentences. We have $Cn^{\mathcal{L}}(\Sigma) = Th^{\mathcal{L}}(Mod^{\mathcal{L}}(\Sigma))$.*

Proof. Since $\mathrm{Th}^{\mathcal{L}}(\mathrm{Mod}^{\mathcal{L}}(\Sigma))$ is an $\mathcal{L}$-theory which contains Σ, we have $\mathrm{Cn}^{\mathcal{L}}(\Sigma) \subseteq \mathrm{Th}^{\mathcal{L}}(\mathrm{Mod}^{\mathcal{L}}(\Sigma))$ by Theorem 4.13.4. Conversely, let φ be a formula in $\mathrm{Th}^{\mathcal{L}}(\mathrm{Mod}^{\mathcal{L}}(\Sigma))$. This means that for every $\mathcal{L}$-structure $\mathcal{A}$ such that $\mathcal{A} \models \Sigma$ we have $\mathcal{A} \models \varphi$ which implies $\Sigma \models \varphi$. Thus, $\varphi \in \mathrm{Cn}^{\mathcal{L}}(\Sigma)$. □

Corollary 4.13.17. *Let $\mathcal{L}$ be a first-order language, $\mathcal{A}$ be an $\mathcal{L}$-structure and Σ be a set of $\mathcal{L}$-sentences. If $\mathcal{A} \models \Sigma$, then $Cn^{\mathcal{L}}(\Sigma) \subseteq Th^{\mathcal{L}}(\mathcal{A})$.*

Proof. Since $\mathcal{A} \models \Sigma$, we have $\{\mathcal{A}\} \subseteq \mathrm{Mod}^{\mathcal{L}}(\Sigma)$, so $\mathrm{Th}^{\mathcal{L}}(\mathrm{Mod}^{\mathcal{L}}(\Sigma)) \subseteq \mathrm{Th}^{\mathcal{L}}(\mathcal{A})$ by the second part of Theorem 4.13.15. Applying Theorem 4.13.16, we obtain the desired inclusion. □

Definition 4.13.18. Let $\mathcal{L}$ be a first-order language and let Σ be a set of $\mathcal{L}$-sentences. Σ is *$\mathcal{L}$-complete* if for every $\mathcal{L}$-sentence φ we have either $\Sigma \models \varphi$ or $\Sigma \models (\neg\varphi)$.

Σ is *$\mathcal{L}$-semantically consistent* if there is no $\mathcal{L}$-sentence φ such that we have both $\Sigma \models \varphi$ and $\Sigma \models (\neg\varphi)$. ▯

If Σ is an $\mathcal{L}$-theory, we can replace in the above definition $\Sigma \models \varphi$ and $\Sigma \models (\neg\varphi)$ by $\varphi \in \Sigma$ and $(\neg\varphi) \in \Sigma$. Therefore, Σ is an $\mathcal{L}$-complete and $\mathcal{L}$-semantically consistent $\mathcal{L}$-theory if and only if for every $\mathcal{L}$-sentence φ, exactly one of φ and $(\neg\varphi)$ belongs to Σ.

Theorem 4.13.19. *Let Σ be a set of $\mathcal{L}$-sentences. Σ is $\mathcal{L}$-semantically consistent if and only if Σ has a model.*

Proof. Suppose that Σ is $\mathcal{L}$-semantically consistent and let φ be an $\mathcal{L}$-sentence. Then, either $\Sigma \not\models \varphi$ or $\Sigma \not\models (\neg\varphi)$, so there is a formula ψ such that $\Sigma \not\models \psi$. This implies that Σ is satisfiable and therefore Σ has a model since it consists of $\mathcal{L}$-sentences.

Conversely, suppose that Σ has a model $\mathcal{A}$ and is not $\mathcal{L}$-semantically consistent. Then, $\Sigma \models \varphi$ and $\Sigma \models (\neg\varphi)$ for some $\mathcal{L}$-sentence φ. This implies both $\mathcal{A} \models \varphi$ and $\mathcal{A} \models (\neg\varphi)$, which is a contradiction. $\square$

Example 4.13.20. If $\mathcal{A}$ is an $\mathcal{L}$-structure, then $\mathrm{Th}^{\mathcal{L}}(\mathcal{A})$ is an $\mathcal{L}$-complete, $\mathcal{L}$-semantically consistent theory. We will actually prove in Theorem 4.13.27 that only theories of the form $\mathrm{Th}^{\mathcal{L}}(\mathcal{A})$ are complete and semantically consistent. $\square$

Theorem 4.13.21. *Let $\mathcal{L}$ be a decidable first-order language and Σ be a semidecidable, $\mathcal{L}$-complete theory. Then, Σ is decidable.*

Proof. If Σ were not $\mathcal{L}$-semantically consistent, then there would be no model for Σ by Theorem 4.13.19, so $\Sigma = \mathrm{SENT}_{\mathcal{L}}$, which implies the decidability of Σ. Thus, we can assume that Σ is $\mathcal{L}$-semantically consistent. Given an $\mathcal{L}$-sentence φ, we begin by running in parallel the semideciding algorithm for Σ on φ and $(\neg\varphi)$. Exactly one of these computations must stop and output 1, due to the $\mathcal{L}$-completeness and $\mathcal{L}$-semantic consistency of Σ. If the first computation to halt is the one we ran on φ, then $\varphi \in \Sigma$; otherwise, $\varphi \notin \Sigma$. $\square$

Corollary 4.13.22. *If $\mathcal{L}$ is a decidable language and Σ is an $\mathcal{L}$-complete theory that is undecidable, then Σ is not semidecidable.*

Proof. This follows immediately from Theorem 4.13.21. $\square$

Theorem 4.13.23. *Let $\mathcal{L}$ be a first-order language, let Σ be an $\mathcal{L}$-complete theory, and let Σ' be an $\mathcal{L}$-semantically consistent theory. If $\Sigma \subseteq \Sigma'$, then $\Sigma = \Sigma'$.*

Proof. Under the hypotheses of the theorem, suppose that $\varphi \in \Sigma' - \Sigma$. Then, $(\neg\varphi) \in \Sigma$ because Σ is $\mathcal{L}$-complete and thus $(\neg\varphi) \in \Sigma'$, which contradicts the $\mathcal{L}$-semantical consistency of Σ'. $\square$

Theorem 4.13.24. *Let $\mathcal{L}$ be a first-order language and let $\mathfrak{A}$ be a collection of $\mathcal{L}$-structures. The theory $Th^{\mathcal{L}}(\mathfrak{A})$ is $\mathcal{L}$-complete if and only if for every $\mathcal{A}, \mathcal{B} \in \mathfrak{A}$, we have $Th^{\mathcal{L}}(\mathcal{A}) = Th^{\mathcal{L}}(\mathcal{B})$.*

Proof. Suppose that for every $\mathcal{A}, \mathcal{B} \in \mathfrak{A}$, we have $\mathrm{Th}^{\mathcal{L}}(\mathcal{A}) = \mathrm{Th}^{\mathcal{L}}(\mathcal{B})$. If $\mathfrak{A}$ is empty, then $\mathrm{Th}^{\mathcal{L}}(\mathfrak{A}) = \mathrm{SENT}_{\mathcal{L}}$ so it is $\mathcal{L}$-complete. If $\mathfrak{A}$ is not empty, then $\mathrm{Th}^{\mathcal{L}}(\mathfrak{A}) = \bigcap\{\mathrm{Th}^{\mathcal{L}}(\mathcal{A}) \mid \mathcal{A} \in \mathfrak{A}\} = \mathrm{Th}^{\mathcal{L}}(\mathcal{B})$ for every $\mathcal{B} \in \mathfrak{A}$. By Example 4.13.20, $\mathrm{Th}^{\mathcal{L}}(\mathfrak{A})$ is complete.

Conversely, suppose that $\mathrm{Th}^{\mathcal{L}}(\mathfrak{A})$ is complete and $\mathcal{B} \in \mathfrak{A}$. Then $\mathrm{Th}^{\mathcal{L}}(\mathfrak{A}) \subseteq \mathrm{Th}^{\mathcal{L}}(\mathcal{B})$, which, by Theorem 4.13.23, implies $\mathrm{Th}^{\mathcal{L}}(\mathfrak{A}) = \mathrm{Th}^{\mathcal{L}}(\mathcal{B})$. Therefore, for any two $\mathcal{L}$-structures $\mathcal{A}, \mathcal{B} \in \mathfrak{A}$, we have $\mathrm{Th}^{\mathcal{L}}(\mathcal{A}) = \mathrm{Th}^{\mathcal{L}}(\mathcal{B})$. □

The previous theorem suggests the following definition.

Definition 4.13.25. Let $\mathcal{L}$ be a first-order language and let $\mathcal{A}, \mathcal{B}$ be two $\mathcal{L}$-structures. We say that $\mathcal{A}$ is *elementarily equivalent* to $\mathcal{B}$ and we write $\mathcal{A} \equiv \mathcal{B}$ if $\mathrm{Th}^{\mathcal{L}}(\mathcal{A}) = \mathrm{Th}^{\mathcal{L}}(\mathcal{B})$. □

Theorem 4.13.26. *Let $\mathcal{L}$ be a first-order language and let Σ be an $\mathcal{L}$-theory. Σ is $\mathcal{L}$-complete if and only if for all models $\mathcal{A}, \mathcal{B}$ of Σ, $\mathcal{A} \equiv \mathcal{B}$.*

Proof. Since Σ is an $\mathcal{L}$-theory, we have $\Sigma = \mathrm{Cn}^{\mathcal{L}}(\Sigma)$. Theorem 4.13.16 implies $\Sigma = \mathrm{Th}^{\mathcal{L}}(\mathrm{Mod}^{\mathcal{L}}(\Sigma))$. The statement now follows immediately from Theorem 4.13.24. □

Theorem 4.13.27. *Let $\mathcal{L}$ be a first-order language and let Σ be an $\mathcal{L}$-theory. Then, Σ is $\mathcal{L}$-complete and $\mathcal{L}$-semantically consistent if and only if $\Sigma = Th^{\mathcal{L}}(\mathcal{A})$ for some $\mathcal{L}$-structure $\mathcal{A}$.*

Proof. By Example 4.13.20 it is clear that the equality $\Sigma = \mathrm{Th}^{\mathcal{L}}(\mathcal{A})$ is sufficient for completeness and semantical consistency. Conversely, let Σ be $\mathcal{L}$-complete and $\mathcal{L}$-semantically consistent. By Theorem 4.13.19, there is an $\mathcal{L}$-structure $\mathcal{A}$ that is a model of Σ, so $\Sigma \subseteq \mathrm{Th}^{\mathcal{L}}(\mathcal{A})$. Since Σ is $\mathcal{L}$-complete and $\mathrm{Th}^{\mathcal{L}}(\mathcal{A})$ is $\mathcal{L}$-semantically consistent, by Theorem 4.13.23, we have $\Sigma = \mathrm{Th}^{\mathcal{L}}(\mathcal{A})$. □

Theorem 4.13.28. *Let $\mathcal{A}$ be an $\mathcal{L}$-structure and φ be an $\mathcal{L}$-formula. The following statements are equivalent:*

(1) $\mathcal{A} \models \varphi$;
(2) $Th^{\mathcal{L}}(\mathcal{A}) \models \varphi$;
(3) $Th^{\mathcal{L}}(\mathcal{A}) \models\!\!\approx \varphi$.

Proof. The equivalence of the last two statements follows directly from Supplement 60, which leaves us with the proof of the equivalence of the first two statements. Suppose that $\mathcal{A} \models \varphi$. Then, by Theorem 4.5.58, we have $\mathcal{A} \models \varphi^{\forall}$, which implies $\varphi^{\forall} \in \mathrm{Th}^{\mathcal{L}}(\mathcal{A})$, hence $\mathrm{Th}^{\mathcal{L}}(\mathcal{A}) \models \varphi^{\forall}$. By Corollary 4.5.38, we have $\mathrm{Th}^{\mathcal{L}}(\mathcal{A}) \models \varphi$.

Using the equivalence of the last two statements, it suffices to show that the third statement implies the first. If $\mathrm{Th}^{\mathcal{L}}(\mathcal{A}) \mathrel{|\!\!\approx} \varphi$, then every model of $\mathrm{Th}^{\mathcal{L}}(\mathcal{A})$ is a model of φ; in particular, $\mathcal{A} \models \varphi$. □

4.14 Decidability and Undecidability in First-Order Logic

We begin this section by showing the decidability of theories of finite structures over finite languages. We remind the reader that this amounts to showing that for every finite structure $\mathcal{A}$, there is an effective way of deciding whether a sentence φ belongs to the theory of the structure, that is, $\mathcal{A} \models \varphi$.

Before we present this argument, it is convenient to introduce the notion of partial assignment.

Definition 4.14.1. Let $\mathcal{A}$ be an $\mathcal{L}$-structure, where $\mathcal{L}$ is a first-order language. A *partial assignment over* $\mathcal{A}$ is a partial function $\sigma : \mathrm{VAR} \rightsquigarrow |\mathcal{A}|$. ▯

For a partial assignment σ over $\mathcal{A}$, define $\sigma^{\mathcal{A}} : \mathrm{TERM}_{\mathcal{L}}(\mathrm{Dom}(\sigma)) \longrightarrow |\mathcal{A}|$ by $\sigma^{\mathcal{A}}(t) = \tau^{\mathcal{A}}(t)$, where τ is an arbitrary extension of σ to VAR. By Theorem 4.5.3, $\sigma^{\mathcal{A}}$ is well-defined.

Recall that $\mathrm{FORM}_{\mathcal{L}}(V) = \{\varphi \in \mathrm{FORM}_{\mathcal{L}} \mid \mathtt{FV}(\varphi) \subseteq V\}$, where $V \subseteq \mathrm{VAR}$.

It follows from this definition, as the reader can easily verify, that the following equalities hold:

$$\sigma^{\mathcal{A}}(c) = c^{\mathcal{A}} \tag{4.18}$$

$$\sigma^{\mathcal{A}}(x) = \sigma(x) \tag{4.19}$$

$$\sigma^{\mathcal{A}}(f(t_0, \ldots, t_{n-1})) = f^{\mathcal{A}}(\sigma^{\mathcal{A}}(t_0), \ldots, \sigma^{\mathcal{A}}(t_{n-1})), \tag{4.20}$$

for every constant symbol c in $\mathcal{L}$, variable $x \in \mathrm{Dom}(\sigma)$, $t_0, \ldots, t_{n-1} \in \mathrm{TERM}_{\mathcal{L}}(\mathrm{Dom}(\sigma))$ and n-ary function symbol f in $\mathcal{L}$ with $n > 0$.

For a partial assignment σ and $\varphi \in \mathrm{FORM}_{\mathcal{L}}(\mathrm{Dom}(\sigma))$, we write $(\mathcal{A}, \sigma) \models \varphi$ if $(\mathcal{A}, \tau) \models \varphi$, where τ is an arbitrary extension of σ to VAR. By the Agreement Theorem for First-Order Logic, the definition of $(\mathcal{A}, \sigma) \models \varphi$ is sound.

The recursive definition of the meaning of the notation $(\mathcal{A}, \sigma) \models \varphi$ presented on page 598 can be transferred ad literam to partial assignments. In every case, we assume that the set of free variables of the formulas is included in the domains of the appropriate partial assignments. Note that if $(Qx)\varphi \in \mathrm{FORM}_{\mathcal{L}}(\mathrm{Dom}(\sigma))$, then $\varphi \in \mathrm{FORM}_{\mathcal{L}}(\mathrm{Dom}([x \to a]\sigma))$ for every $a \in |\mathcal{A}|$.

Lemma 4.14.2. *Let $\mathcal{A}$ be a finite $\mathcal{L}$-structure, where $\mathcal{L}$ is a finite first-order language. There is an algorithm that, given a partial assignment σ over $\mathcal{A}$ and a term $t \in \mathrm{TERM}_{\mathcal{L}}(\mathrm{Dom}(\sigma))$, computes the element $\sigma^{\mathcal{A}}(t)$ of $|\mathcal{A}|$.*

Proof. Since $\mathcal{A}$ is a finite structure, for any n-ary function symbol $f \in \mathcal{L}$, the function $f^{\mathcal{A}}$ can be specified in tabular form through an array with $n + 1$ columns and m^n rows, where m is the number of elements of $|\mathcal{A}|$. Equalities 4.18–4.20 allow us to compute $\sigma^{\mathcal{A}}(t)$ starting from the values of σ and consulting the necessary tables for the function symbols. $\square$

Theorem 4.14.3. *If $\mathcal{A}$ is a finite $\mathcal{L}$-structure, where $\mathcal{L}$ is a finite first-order language, then $Th^{\mathcal{L}}(\mathcal{A})$ is decidable.*

Proof. We will prove a stronger statement, namely that there is an algorithm that starting from an $\mathcal{L}$-formula φ and a partial assignment σ such that $\mathtt{FV}(\varphi) \subseteq \mathrm{Dom}(\sigma)$ decides whether $(\mathcal{A}, \sigma) \models \varphi$. The algorithm is by recursion on the formula φ.

Suppose that φ is a propositional constant R. In this case the decision is immediate because $(\mathcal{A}, \sigma) \models R$ if and only if $R^{\mathcal{A}} = \mathbf{T}$.

Suppose now that $\varphi = R(t_0, \ldots, t_{n-1})$, where $n \geq 1$. Applying the algorithm outlined in Lemma 4.14.2 allows us to compute the elements $\sigma^{\mathcal{A}}(t_i)$ for $0 \leq i \leq n - 1$. Then, we have $(\mathcal{A}, \sigma) \models R(t_0, \ldots, t_{n-1})$ if and only if the n-tuple $(\sigma^{\mathcal{A}}(t_0), \ldots, \sigma^{\mathcal{A}}(t_{n-1}))$ occurs in the list of n-tuples that specifies the relation $R^{\mathcal{A}}$.

Suppose now that $\varphi = (\forall x)\psi$, where the algorithm is defined for ψ. Recall that $(\mathcal{A}, \sigma) \models \varphi$ if and only if $(\mathcal{A}, [x \to a]\sigma) \models \psi$ for every $a \in |\mathcal{A}|$. Since $|\mathcal{A}|$ is finite, we have reduced the decision on whether $(\mathcal{A}, \sigma) \models \varphi$ to $|\mathcal{A}|$ applications of the decision algorithm for ψ using the assignments $[x \to a]\sigma$ for $a \in |\mathcal{A}|$.

We leave the remaining recursive steps to the reader. $\square$

The next problem is the basis for proving a number of undecidability results in Computer Science.

Definition 4.14.4. An *instance of the Post Correspondence Problem* (abbreviated PCP) is a triple $\Im = (V, (q_0, \ldots, q_{n-1}), (r_0, \ldots, r_{n-1}))$, where V is an alphabet and $(q_0, \ldots, q_{n-1}), (r_0, \ldots, r_{n-1})$ are nonempty sequences of equal length n which consist of nonempty words over V. A *solution* is a nonempty sequence $(i_0, \ldots, i_{k-1})$ such that

(1) $0 \leq i_j \leq n-1$ for $0 \leq j \leq k-1$;
(2) $q_{i_0} \cdots q_{i_{k-1}} = r_{i_0} \cdots r_{i_{k-1}}$.

$\blacksquare$

Another way to describe the existence of a solution of an instance of PCP is through the notion introduced next.

Definition 4.14.5. Let $\Im = (V, (q_0, \ldots, q_{n-1}), (r_0, \ldots, r_{n-1}))$ be an instance of the PCP. A *construction sequence for* $\Im$ is an even-length sequence of words over V, $(s_0, t_0, \ldots, s_k, t_k)$ for $k \geq 0$, such that for every j with $0 \leq j \leq k$ one of the following two cases occurs:

(1) $s_j = t_j = \lambda$;
(2) $j > 0$ and there is $i_{j-1} \in \{0, \ldots, n-1\}$ for which $s_j = s_{j-1}q_{i_{j-1}}$ and $t_j = t_{j-1}r_{i_{j-1}}$.

$\blacksquare$

Theorem 4.14.6. *Let $\Im = (V, (q_0, \ldots, q_{n-1}), (r_0, \ldots, r_{n-1}))$ be an instance of the PCP. There is a solution for $\Im$ if and only if there is a construction sequence $(s_0, t_0, \ldots, s_k, t_k)$ for the instance such that for some i, $0 \leq i \leq k$, $s_i = t_i \neq \lambda$.*

Proof. We leave the argument to the reader. $\square$

Example 4.14.7. Let $V = \{a, b, c\}$ and let

$$\Im = (V, (bba, bc, bba, bba), (bb, abcb, b, ba))$$

be an instance of PCP. The sequence $(0, 1, 2, 1, 3)$ is a solution of $\Im$ because

$$q_0q_1q_2q_1q_3 = bbabcbbabcbba = r_0r_1r_2r_1r_3.$$

$\blacksquare$

Example 4.14.8. Let $V = \{a, b\}$ and let

$$\Im = (V, (abb, bba, bbab), (ba, abb, aab)).$$

Note that if an instance of the PCP has a solution, then there must be a pair of words q_i, r_i that begin with the same symbol. Since this is not the case here, $\Im$ has no solution. $\square$

A fundamental result of Computability Theory states that it is undecidable whether an arbitrary instance of the Post Correspondence Problem has a solution. In this section, we examine consequences of the undecidability of the Post Correspondence Problem for first-order logic.

A stronger form of the undecidability of the PCP is presented in the next definition and theorem.

Definition 4.14.9. Let V be an alphabet. A *V-instance of the PCP* is an instance of PCP whose first component is V. $\square$

Theorem 4.14.10. *Let V be an alphabet with at least two symbols. It is undecidable whether or not a V-instance of the PCP has a solution.*

Proof. Let $\Im = (U, (q_0, \ldots, q_{n-1}), (r_0, \ldots, r_{n-1}))$ be an arbitrary instance of the PCP, where $U = \{a_0, \ldots, a_{m-1}\}$. Let a, b be two fixed symbols of the alphabet V. Choose l to be the least number with $m \leq 2^l$. For $0 \leq i \leq m - 1$, let $\beta_0^i \cdots \beta_{l-1}^i$ be the binary equivalent of i padded with 0s to the left. Define a mapping $f : U \longrightarrow V^*$ by $f(a_i) = c_0 \cdots c_{l-1}$, where

$$c_k = \begin{cases} a & \text{if } \beta_k^i = 0 \\ b & \text{if } \beta_k^i = 1. \end{cases}$$

We extend f to $f : U^* \longrightarrow V^*$ in the obvious way. Starting from $\Im$, and using this extension, we can construct an instance $\Im_V$:

$$(V, (f(q_0), \ldots, f(q_{n-1})), (f(r_0), \ldots, f(r_{n-1}))).$$

Note that the fact that $|V| \geq 2$, allows us to encode all the symbols of the alphabet U as words of equal length over the alphabet V and this makes the mapping $f : U^* \longrightarrow V^*$ an injection. Thus, it is clear that a sequence is a solution for $\Im$ if and only if it is a solution for $\Im_V$. It follows that a decision procedure for instances of the form (V, q, r) would yield a decision procedure for arbitrary instances of the PCP, which we know does not exist. $\square$

Let V be an alphabet. Define the first-order language $\mathcal{L}_V$ as $\mathcal{L}_V = \{R, c\} \cup \{f_a \mid a \in V\}$, where R is a binary relation symbol, c is a constant symbol, and f_a is a unary function symbol for every $a \in V$. If $t \in \mathrm{TERM}_{\mathcal{L}_V}$ and $u = a_{i_0} \cdots a_{i_{k-1}} \in V^*$, then we denote by $f_u(t)$ the $\mathcal{L}_V$-term $f_{a_{i_{k-1}}}(\cdots(f_{a_{i_0}}(t)\cdots)$.

Lemma 4.14.11. *Let $V = \{a_0, \ldots, a_{n-1}\}$ be an alphabet. For every instance $\Im = (V, q, r)$ of the PCP one can effectively construct an $\mathcal{L}_V$-sentence $\varphi_\Im$ such that $\Im$ has a solution if and only if $\models \varphi_\Im$.*

Proof. Let $\Im = (V, (q_0, \ldots, q_{n-1}), (r_0, \ldots, r_{n-1}))$ be an instance of the PCP. Define the $\mathcal{L}_V$-sentence $\varphi_\Im$ as $\varphi_\Im = ((\alpha_\Im \wedge \beta_\Im) \rightarrow \gamma)$, where

$$\alpha_\Im = \bigwedge_{i=0}^{n-1} R(f_{q_i}(c), f_{r_i}(c)),$$

$$\beta_\Im = (\forall x)(\forall y)(R(x, y) \rightarrow \bigwedge_{i=0}^{n-1} R(f_{q_i}(x), f_{r_i}(y))),$$

$$\gamma = (\exists z)R(z, z).$$

It is clear that the construction of $\varphi_\Im$ starting from the PCP instance $\Im$ is effective.

Suppose that $\models \varphi_\Im$. Define the $\mathcal{L}_V$-structure $\mathcal{A}_V = (V^*, \mathcal{I})$, where $c^{\mathcal{A}_V} = \lambda$, $f_a^{\mathcal{A}_V} : V^* \longrightarrow V^*$ is the function given by $f_a^{\mathcal{A}_V}(w) = wa$ for $w \in V^*$ and

$$R^{\mathcal{A}_V} = \{(u, w) \in V^* \times V^* \mid u = q_{i_0} \cdots q_{i_{\ell-1}} \text{ and } w = r_{i_0} \cdots r_{i_{\ell-1}} \text{ for some nonempty sequence } (i_0, \ldots, i_{\ell-1}) \in \mathrm{Seq}(\{0, \ldots, n-1\})\}.$$

Note that $\mathcal{A}_V \models \alpha_\Im$ and $\mathcal{A}_V \models \beta_\Im$. Since $\mathcal{A}_V \models \varphi_\Im$ it follows that $\mathcal{A}_V \models \gamma$. Thus, there exists a word $u \in V^*$ such that

$$u = q_{i_0} \cdots q_{i_{\ell-1}} = r_{i_0} \cdots r_{i_{\ell-1}}$$

which means that $(i_0, \ldots, i_{\ell-1})$ is a solution for $\Im$.

Conversely, suppose that $\Im$ has a solution $(i_0, \ldots, i_{\ell-1})$ and let $\mathcal{A}$ be an $\mathcal{L}_V$-structure such that $\mathcal{A} \models (\alpha_\Im \wedge \beta_\Im)$. Define the mapping $h : V^* \longrightarrow |\mathcal{A}|$ by

$$h(\lambda) = c^{\mathcal{A}}$$
$$h(w) = (f_w(c))^{\mathcal{A}}$$

for $w \in V^*$ and $w \neq \lambda$.

Since $\mathcal{A} \models \alpha_{\Im}$ we have $(h(q_i), h(r_i)) \in R^{\mathcal{A}}$ for $0 \leq i \leq n-1$. Further, since $\mathcal{A} \models \beta_{\Im}$, $(h(u), h(w)) \in R^{\mathcal{A}}$ implies $(h(uq_i), h(wr_i)) \in R^{\mathcal{A}}$. Therefore, it is easy to prove, by induction on ℓ, that $(h(q_{i_0} \cdots q_{i_{\ell-1}}), h(r_{i_0} \cdots r_{i_{\ell-1}})) \in R^{\mathcal{A}}$. Since $(i_0, \ldots, i_{\ell-1})$ is a solution for $\Im$ we have

$$q_{i_0} \cdots q_{i_{\ell-1}} = r_{i_0} \cdots r_{i_{\ell-1}}$$

which implies $\mathcal{A} \models \gamma$. Thus, $\mathcal{A} \models \varphi_{\Im}$, so $\varphi_{\Im}$ is logically valid. □

The next theorem establishes that for certain first-order languages, it is undecidable whether sentences of the language are logically valid.

Theorem 4.14.12. *For every alphabet V that contains at least two symbols, it is undecidable whether an $\mathcal{L}_V$-sentence is logically valid.*

Proof. This statement follows from Theorem 4.14.10 and Lemma 4.14.11. □

Corollary 4.14.13. *If a first-order language $\mathcal{L}$ contains at least one binary relation symbol, at least one constant symbol and at least two unary function symbols, then it is undecidable whether an $\mathcal{L}$-sentence is logically valid.*

Proof. A first-order language $\mathcal{L}$ that satisfies the conditions of the corollary contains a first-order language $\mathcal{L}_V$ for a two element alphabet V. Since validity for $\mathcal{L}_V$-sentences is undecidable, it follows that validity for $\mathcal{L}$-sentences is undecidable. □

Corollary 4.14.14. *It is undecidable whether an arbitrary sentence of first-order logic is logically valid.*

Proof. This follows immediately from Corollary 4.14.13. □

We prove next, using the undecidability of PCP that the theory of arithmetic, i.e., $\mathrm{Th}^{\mathcal{L}_{ar}}(\mathcal{A}_{ar})$ is undecidable. The reader should note the contrast between this result and the decidability of Presburger arithmetic given in Exercise 185.

Theorem 4.14.15. *There is an effective procedure that for every instance $\Im$ of the PCP over the alphabet $V = \{d, e\}$ produces a sentence $\varphi_{\Im,solv}$ of $\mathcal{L}_{ar}$ such that $\Im$ has a solution if and only if $\varphi_{\Im,solv} \in Th^{\mathcal{L}_{ar}}(\mathcal{A}_{ar})$.*

Proof. We define recursively an injective mapping $\phi : V^* \longrightarrow \mathbf{N}$ by

$$\begin{aligned}\phi(\lambda) &= 1\\ \phi(qd) &= 2\phi(q)\\ \phi(qe) &= 2\phi(q) + 1\end{aligned}$$

for every $q \in V^*$.

We begin with several $\mathcal{L}_{ar}$-formulas that play a technical role in our construction.

The formula $\varphi_{z,\text{null}}$ is $z = s(0)$. Note that $\varphi_{z,\text{null}}$ with z defines the set $\{1\} = \{\phi(\lambda)\}$.

To define another type of formula, let $\overline{d}, \overline{e} : \text{TERM}_{\mathcal{L}_{ar}} \longrightarrow \text{TERM}_{\mathcal{L}_{ar}}$ be two functions given by

$$\begin{aligned}\overline{d}(t) &= t + t,\\ \overline{e}(t) &= (t + t) + s(0).\end{aligned}$$

Note that if $t \in \text{TERM}_{\mathcal{L}_{ar}}$, $q \in V^*$ and $\sigma \in \text{ASSIGN}_{\mathcal{A}_{ar}}$ are such that $\sigma^{\mathcal{A}_{ar}}(t) = \phi(q)$, then $\sigma^{\mathcal{A}_{ar}}(\overline{d}(t)) = \phi(qd)$ and $\sigma^{\mathcal{A}_{ar}}(\overline{e}(t)) = \phi(qe)$. It follows from this that if $\sigma^{\mathcal{A}_{ar}}(t) = \phi(r)$, then $\sigma^{\mathcal{A}_{ar}}(\overline{q_{n-1}}(\cdots \overline{q_0}(t) \cdots)) = \phi(rq)$, where $q = q_0 \cdots q_{n-1}$. For $q \in V^*$, the formula $\varphi_{z_0,z_1,q,\text{conc}}$ is given by

$$((z_1 \neq 0) \wedge (z_0 = \overline{q_{n-1}}(\overline{q_{n-2}}(\cdots \overline{q_0}(z_1) \cdots)))).$$

We leave to the reader to verify that the formula $\varphi_{z_0,z_1,q,\text{conc}}$ with the variables z_0 and z_1 defines the set $\{(\phi(r_0), \phi(r_1)) \mid r_0 = r_1 q\}$, that is, that $(\mathcal{A}_{ar}, [z_0 \to n_0][z_1 \to n_1]) \models \varphi_{z_0,z_1,q,\text{conc}}$ if and only if $(n_0, n_1) \in \{(\phi(r_0), \phi(r_1)) \mid r_0 = r_1 q\}$.

We encode finite sequences of natural numbers using triplets of the form (c, d, i), where (c, d, i) encodes the sequence $(a_0, \ldots, a_{i-1})$ such that

$$(c, d, j, a_j) \in B \text{ for } 0 \leq j \leq i - 1.$$

Here, B is the Gödel relation introduced in Theorem 4.7.7 and is defined in $\mathcal{A}_{ar}$ by the formula $\varphi_{x_1,x_2,v,w,B}$ introduced in Example 4.7.8.

The formula $\varphi_{x',y',z',y_0,y_1,x,y,z,\text{pref}}$ is intended to express that the even length sequence of words whose code is (x', y', z') concatenated with the pair of words whose codes are y_0, y_1 is a prefix of the sequence coded by (x, y, z) and it is given by:

$$\Big((\exists w')(z' = w' + w')\wedge \\ \Big((s(z') < z) \wedge (\forall r)(\forall j)\big(((j < z') \wedge (\varphi_{x_1,x_2,v,w,B})_{x_1,x_2,v,w:=x',y',j,r}\big) \to \\ (\varphi_{x_1,x_2,v,w,B})_{x_1,x_2,v,w:=x,y,j,r}\big) \wedge (\varphi_{x_1,x_2,v,w,B})_{x_1,x_2,v,w:=x,y,z',y_0} \\ \wedge(\varphi_{x_1,x_2,v,w,B})_{x_1,x_2,v,w:=x,y,s(z'),y_1}\Big)\Big).$$

To express the fact that the pair (y', y'') represents the codes of words that appear on the last two places of the sequence whose code is $(\hat{x}, \hat{y}, \hat{z})$, we introduce the formula $\varphi_{y',y'',\hat{x},\hat{y},\hat{z},\text{last-entry}}$ given by

$$\Big((\exists \hat{w})(\hat{z} = \hat{w} + \hat{w})\wedge \\ (\forall \dot{z})\Big((s(s(\dot{z})) = \hat{z}) \to \big((\varphi_{x_1,x_2,v,w,B})_{x_1,x_2,v,w:=\hat{x},\hat{y},\dot{z},y'} \\ \wedge(\varphi_{x_1,x_2,v,w,B})_{x_1,x_2,v,w:=\hat{x},\hat{y},s(\dot{z}),y''}\big)\Big)\Big).$$

Let $\Im = (\{d, e\}, q, r)$ be an instance of the PCP, where $q = (q_0, \ldots, q_{n-1})$ and $r = (r_0, \ldots, r_{n-1})$. The next three formulas we are about to introduce depend on the instance $\Im$.

Let $\varphi^{\Im}_{u,v,s,t,\text{one-step}}$ be given by:

$$\bigvee_{0 \le i \le n-1} ((\varphi_{z_0,z_1,q_i,\text{conc}})_{z_0,z_1:=u,s} \wedge (\varphi_{z_0,z_1,r_i,\text{conc}})_{z_0,z_1:=v,t}).$$

This formula, together with the sequence of variables (u, v, s, t) defines the set $\{(\phi(q'), \phi(r'), \phi(q''), \phi(r'')) \in \mathbf{N}^4 \mid q'', r'' \in \{d, e\}^*$ and $q' = q''q_i, r' = r''r_i$ for some $i, 0 \le i \le n-1\}$.

The formula $\varphi^{\Im}_{x,y,z,\text{const}}$ holds when x, y, z encode a construction sequence for $\Im$ and is given by:

$$\begin{aligned}
&\Big((\exists w)(z = w + w)\wedge\\
&(\forall y_0)(\forall y_1)(\forall x')(\forall y')(\forall z')\Big(\varphi_{x',y',z',y_0,y_1,x,y,z,\text{pref}} \rightarrow\\
&\big((\varphi_{z,\text{null}})_{z:=y_0} \wedge (\varphi_{z,\text{null}})_{z:=y_1}\big)\\
&\vee(\exists y_0'')(\exists y_1'')\big((\varphi_{y',y'',\hat{x},\hat{y},\hat{z},\text{last-entry}})_{y',y'',\hat{x},\hat{y},\hat{z}:=y_0'',y_1'',x',y',z'}\\
&\wedge(\varphi^{\Im}_{u,v,s,t,\text{one-step}})_{u,v,s,t:=y_0,y_1,y_0'',y_1''}\big)\Big)\Big).
\end{aligned}$$

Finally, the formula $\varphi^{\Im}_{\text{solv}}$ is given by:

$$\begin{aligned}
&(\exists x)(\exists y)(\exists z)(\exists y_0)(\varphi^{\Im}_{x,y,z,\text{const}}\\
&\wedge(\varphi_{y',y'',\hat{x},\hat{y},\hat{z},\text{last-entry}})_{y',y'',\hat{x},\hat{y},\hat{z}:=y_0,y_0,x,y,z} \wedge (\neg(\varphi_{z,\text{null}})_{z:=y_0}))
\end{aligned}$$

This last formula states that $\Im$ is solvable. Indeed, by Theorem 4.14.6, it follows that $\mathcal{A}_{ar} \models \varphi^{\Im}_{\text{solv}}$ if and only if the instance $\Im$ has a solution. Since the construction of the formula $\varphi^{\Im}_{\text{solv}}$ is effective starting from the instance $\Im$, the statement of the theorem follows. □

Corollary 4.14.16. *The theory of arithmetic is undecidable.*

Proof. This follows immediately from Theorems 4.14.10 and 4.14.15. □

A further strengthening of the previous corollary is given next.

Corollary 4.14.17. *The theory of arithmetic is not semidecidable.*

Proof. The statement follows from Theorem 4.13.21 and Corollary 4.14.16. □

4.15 Exercises and Supplements

First-Order Languages

(1) Prove that there exist two symbols a and b of first-order logic such that $\mathsf{code}(a)$ is a proper prefix of $\mathsf{code}(b)$.

(2) Prove that if n and k are different from 0, we have:

$$\begin{aligned} |\mathsf{code}(x_n)| &= 1 + \lceil \log_2(n+1) \rceil \\ |\mathsf{code}(R^n_k)| &= 2 + \lceil \log_2(n+1) \rceil + \lceil \log_2(k+1) \rceil \\ |\mathsf{code}(f^n_k)| &= 2 + \lceil \log_2(n+1) \rceil + \lceil \log_2(k+1) \rceil. \end{aligned}$$

Terms and Formulas

(3) Let $\mathcal{L}$ be a first-order language and V be a set of variables. Prove that $\mathrm{TERM}_{\mathcal{L}}(V)$ is finite if and only if the following three conditions all hold:

(a) V is finite;
(b) $\mathrm{const}(\mathcal{L})$ is finite;
(c) either $\mathcal{L}$ contains no function symbols that are not constant symbols or $V \cup \mathrm{const}(\mathcal{L}) = \emptyset$,

where $\mathrm{const}(\mathcal{L})$ is the set of constant symbols of $\mathcal{L}$.

(4) Show that:

(a) No term contains a relation symbol.
(b) Every formula contains a relation symbol.

(5) Show that no formula begins with a connective symbol, a quantifier symbol, a right parenthesis or a variable.

(6) Let $\mathcal{L}$ be a finite first-order language. Prove that there is a formula $\varphi \in \mathrm{FORM}_{\mathcal{L}}$ that contains every symbol in $\mathcal{L}$ if and only if one of the following holds:

(a) $\mathcal{L}$ contains no function symbols, or
(b) $\mathcal{L}$ contains a relation symbol of positive arity.

(7) Prove that no proper prefix of a term can be a suffix of a formula.
Solution: Let K be function introduced in Definition 1.5.10 and extended in Lemma 4.3.18. In the argument of the Lemma, we proved that $K(v) > 0$ for every nonnull suffix v of a formula. On the other hand, if z is a proper prefix of a term, we have $K(z) < 1$, as we saw in Lemma 1.5.11. This gives the desired conclusion.

(8) Let t be a term. Then prove the following statements.

(a) If t occurs in an atomic formula $R(t_0, \ldots, t_{n-1})$, then every occurrence of t is a part of a term t_i.

(b) If t occurs in a formula $(\neg\varphi)$, then every occurrence of t is a part of φ.

(c) If t occurs in a formula $(\varphi C \psi)$, where C is a binary connective symbol, then every occurrence of t is a part of φ or a part of ψ.

(d) If t occurs in a formula $(Qx)\varphi$, then every occurrence (t, i) is either $(x, 2)$ or is a part of φ.

Hint. Use Part (a) of Exercise 53 of Chapter 1 for the first case and Supplement 7 for the remaining cases.

Let Ω be a set of signed or unsigned formulas. The $\mathtt{size}(\Omega)$ of Ω is the sum of the sizes of the members of Ω.

(9) Let $\mathcal{L} = \{=, L, c_1, d, s\}$, where L is a unary relation symbol, c_1 is a constant symbol, d is a unary function symbol and s is a binary function symbol and let x, y, z be variables. For $n \in \mathbf{N}$, define Δ_n as the set of signed $\mathcal{L}$-sentences that consists of the following formulas:

$$\begin{aligned}
\mathbf{T}\varphi &= \mathbf{T}(\forall x)(\forall y)(\forall z)(s(x, s(y, z)) = s(s(x, y), z)) \\
\mathbf{T}\psi &= \mathbf{T}(\forall x)(d(x) = s(x, x)) \\
\mathbf{T}\alpha &= \mathbf{T}L(c_1) \\
\mathbf{T}\beta &= \mathbf{T}(\forall x)(L(x) \to L(s(x, c_1))) \\
\mathbf{F}\theta_n &= \mathbf{F}L(\underbrace{d(\cdots d(}_{2^n} c_1) \cdots)).
\end{aligned}$$

Prove that $\mathtt{size}(\Delta_n) = \Theta(2^n)$.

(10) Show that it is possible for a proper suffix of a formula to be a formula.

(11) Let $\mathcal{L}$ be a first-order language and V be a set of variables. Prove that the set $\mathrm{AFORM}_{\mathcal{L}}(V)$ is finite if and only if the following conditions are satisfied:

(a) there are finitely many 0-ary relation symbols in $\mathcal{L}$;

(b) if there is a relation symbol of arity greater than 0 in $\mathcal{L}$, then there are finitely many $(\mathcal{L}, V)$-terms;

(c) if there are infinitely many relation symbols of arity greater than 0 in $\mathcal{L}$, then there are no $(\mathcal{L}, V)$-terms.

(12) Prove the following sharpening of the Occurrence Theorem, for a formula α:

(a) If φ is an atomic formula, then

$$\mathtt{OCC}_\alpha(\varphi) = \begin{cases} \emptyset & \text{if } \alpha \neq \varphi \\ \{(\alpha, 0)\} & \text{otherwise.} \end{cases}$$

(b) If φ is a formula, then

$$\mathtt{OCC}_\alpha((\neg\varphi)) = \begin{cases} T_2(\mathtt{OCC}_\alpha(\varphi)) & \text{if } \alpha \neq (\neg\varphi) \\ \{(\alpha, 0)\} & \text{otherwise.} \end{cases}$$

(c) If φ and ψ are formulas and C is a connective symbol, then

$$\mathtt{OCC}_\alpha((\varphi C \psi)) = \begin{cases} T_1(\mathtt{OCC}_\alpha(\varphi)) \cup T_{|\varphi|+2}(\mathtt{OCC}_\alpha(\psi)) & \text{if } \alpha \neq (\varphi C \psi) \\ \{(\alpha, 0)\} & \text{otherwise.} \end{cases}$$

(d) If φ is a formula, Q is a quantifier and x is a variable, then

$$\mathtt{OCC}_\alpha((Qx)\varphi) = \begin{cases} T_4(\mathtt{OCC}_\alpha(\varphi)) & \text{if } \alpha \neq (Qx)\varphi \\ \{(\alpha, 0)\} & \text{otherwise.} \end{cases}$$

(13) Show that if $\mathcal{L}_1 \subseteq \mathcal{L}_2$, then $\mathrm{FORM}_{\mathcal{L}_1} \subseteq \mathrm{FORM}_{\mathcal{L}_2}$.

(14) Prove that the inductive definition of FORM introduced in Theorem 4.3.15 satisfies the unique readability condition.

(15) Let φ be a formula. Define the mapping $\xi : \mathtt{BO}(\varphi) \longrightarrow \mathtt{OCC}_\exists(\varphi) \cup \mathtt{OCC}_\forall(\varphi)$ by

$$\xi((x,i)) = \begin{cases} (Q,j) & \text{where } \mathsf{binding}_\varphi((x,i)) = j+1 \text{ if} \\ & (x,i) \in \mathtt{PBO}(\varphi) \text{ and } (Q,j) \in \mathtt{OCC}_Q(\varphi), \\ (Q, i-1) & \text{if } (x,i) \in \mathtt{ABO}(\varphi) \text{ and } (Q, i-1) \in \mathtt{OCC}_Q(\varphi). \end{cases}$$

Compute the function ξ for the formulas φ and ψ considered in Examples 4.3.31 and 4.3.32, respectively.

(16) If ξ is the function defined in Exercise 15, prove that:

(a) if $\xi((x,i)) = (Q,j)$, then $j < i$;

(b) if $\xi((x,i)) = (Q,j)$ and (x,ℓ) is an occurrence of x such that $j < \ell < i$, then $(x,\ell) \in \mathtt{BO}(\varphi)$ and if $\xi((x,\ell)) = (Q',k)$, then we have $k \geq j$.

(17) Let φ, ψ be two formulas, C be a binary connective symbol, Q be a quantifier symbol and x be a variable. Prove that

(a) $\| \varphi \| = 0$ if and only if φ is atomic;
(b) $\| (\neg\varphi) \| = \| \varphi \| + 1$;
(c) $\| (\varphi C \psi) \| = \| \varphi \| + \| \psi \| + 2$;
(d) $\| (Qx)\varphi \| = \| \varphi \| + 1$.

(18) (a) Show that $(\varphi)_{x:=x} = \varphi$ for any formula φ and variable x.
(b) Show that if $x \notin \mathtt{FV}(\varphi)$, then $(\varphi)_{x:=t} = \varphi$ for every term t.
(c) Give an example of a formula φ, variables x and y and terms t_0 and t_1 such that

$$((\varphi)_{x:=t_0})_{y:=t_1} \neq ((\varphi)_{y:=t_1})_{x:=t_0}.$$

(19) Let φ be a formula, x and y be variables and t be a term.

(a) Show that if $x \notin \mathtt{FV}(\varphi)$ or t contains no variables, then t is substitutable for x in φ.
(b) Prove that if $y \notin \mathtt{BV}(\varphi)$, then y is substitutable for x in φ.
(c) Prove that x is substitutable for x in φ.

(20) Let φ be a formula, x, y be variables and t, u be terms. Prove that:

(a) If $x \neq y$, then

$$|\mathtt{OCC}_y(u)| \leq |\mathtt{OCC}_y(\mathsf{s}_t^x(u))|$$

and equality holds if and only if either x does not occur in u or y does not occur in t.
Hint: Use induction on u.

(b) If $x \neq y$, then

$$|\mathtt{FO}_y(\varphi)| \leq |\mathtt{FO}_y((\varphi)_{x:=t})|.$$

Hint: Use induction on φ; for the basis, use Part (a).

(c) If $x \neq y$ and t is substitutable for x in φ, then

$$|\mathtt{FO}_y(\varphi)| = |\mathtt{FO}_y((\varphi)_{x:=t})|$$

if and only if x does not occur free in φ or y does not occur in t.
Hint: Use induction on φ, the previous parts of this exercise, and Theorem 4.3.77.

(d) $|\mathtt{BO}(\varphi)| \leq |\mathtt{BO}((\varphi)_{x:=t})|$.
Hint: Use induction on φ and Part (b).

(e) The term t is substitutable for x in φ if and only if

$$|\mathtt{BO}(\varphi)| = |\mathtt{BO}((\varphi)_{x:=t})|.$$

Hint: Use induction on φ, Parts (b), (c), (d) and Theorem 4.3.77.

(21) Let V be a subset of VAR and $s : V \longrightarrow \text{VAR}$ be an injective function. Prove that the extension $\bar{s}$ of s to $\text{LIT}(V)$ is injective.
Hint. Use Supplement 15 of Chapter 1.

(22) Let φ be a formula and let s, s' be substitutions such that $s(x) = s'(x)$ for all $x \in \mathtt{FV}(\varphi)$. Show that s is admissible for φ if and only if s' is admissible for φ.

(23) Let φ be a formula and let s, s' be substitutions such that s is admissible for φ and s' is admissible for $\text{FVSubst}(s, \varphi)$. Prove that $s' * s$ is admissible for φ.
Solution: The argument is by induction on φ. The basis step, when φ is atomic, is immediate. For the inductive step, we discuss only the case when $\varphi = (Qy)\psi$, where we assume that the statement holds for ψ.
Since s is admissible for φ, $[y \to y]s$ is admissible for ψ; also, if $z \in \mathtt{FV}(\varphi)$, then $y \notin \mathtt{V}(s(z))$.
The admissibility of s' for $\text{FVSubst}(s, \varphi)$ means that $[y \to y]s'$ is admissible for $\text{FVSubst}([y \to y]s, \psi)$ and that $w \in \mathtt{FV}(\text{FVSubst}(s, \varphi))$ implies $y \notin \mathtt{V}(s'(w))$.
By applying the inductive hypothesis for ψ to the substitutions $[y \to y]s$ and $[y \to y]s'$, it follows that the substitution $([y \to y]s') * ([y \to y]s)$ is admissible for ψ. By the argument given in the proof of Theorem 4.3.86, the substitutions $([y \to y]s') * ([y \to y]s)$ and $[y \to y](s' * s)$ agree on the free variables of ψ, so by Exercise 22, the substitution $[y \to y](s' * s)$ is admissible for ψ.
It remains to show that if $x \in \mathtt{FV}(\varphi)$, then $y \notin \mathtt{V}(s' * s(x)) = \bigcup_{w \in \mathtt{V}(s(x))} \mathtt{V}(s'(w))$. The last equality follows from Theorem 1.5.23. In other words, we have to show that if $x \in \mathtt{FV}(\varphi)$ and $w \in \mathtt{V}(s(x))$, then $y \notin \mathtt{V}(s'(w))$. Since $w \in V(s(x))$, it follows from Theorem 4.3.82 and the admissibility of s for φ, that $w \in \mathtt{FV}(\text{FVSubst}(s, \varphi))$ and thus $y \notin \mathtt{V}(s'(w))$ by the admissibility of s' for $\text{FVSubst}(s, \varphi)$.

(24) Give an example of a formula φ, a term t, a variable x and a subformula ψ of φ such that t is substitutable for x in φ, but it is not substitutable for x in ψ.

(25) Let a be a constant symbol that does not occur in the term u or the formula φ.

(a) Prove by induction on u that $\mathsf{s}_t^a(\mathsf{s}_a^x(u)) = \mathsf{s}_t^x(u)$.
(b) Prove by induction on φ that $\mathsf{s}_t^a((\varphi)_{x:=a}) = (\varphi)_{x:=t}$.

(26) Let s be a substitution, $s = \mathsf{s}_{t_0 \cdots t_{n-1}}^{y_0 \cdots y_{n-1}}$, where $y_0, \ldots, y_{n-1}$ are distinct variables and $t_0, \ldots, t_{n-1}$ are terms. Prove that if t is a term, then the set $\{u \in \text{TERM} \mid s(u) = t\}$ is finite.
Solution: The proof is by induction on the term t. If $t = x$, where x is a variable, and $s(u) = t$, then $u \in \{x, y_0, \ldots, y_{n-1}\}$. If t is a constant symbol c and $s(u) = t$, then $u = c$. Thus, in either case, the set $\{u \in \text{TERM} \mid s(u) = t\}$ is finite, which concludes the basis steps. If $t = f(v_0, \ldots, v_{m-1})$, where the statement holds for $v_0, \ldots, v_{n-1}$ and $s(u) = t$, then $u = f(w_0, \ldots, w_{m-1})$, where $s(w_i) = v_i$, for $0 \leq i \leq m-1$. By the inductive hypothesis, there are only finitely many terms w_i which are mapped into v_i by s, so there are only finitely many choices for u.

(27) Let s be a substitution, $s = \mathsf{s}_{t_0 \cdots t_{n-1}}^{y_0 \cdots y_{n-1}}$, where $y_0, \ldots, y_{n-1}$ are distinct variables and $t_0, \ldots, t_{n-1}$ are terms. Prove that if φ is a formula, then the set $\{\psi \in \text{FORM} \mid \text{FVSubst}(s, \psi) = \varphi\}$ is finite.
Solution: By induction on φ, we prove that the statement holds for every substitution s that has the prescribed form. Suppose initially that φ is a propositional constant R. Then, if $\text{FVSubst}(s, \psi) = \varphi$, we have $\psi = R$. If $\varphi = R(t_0, \ldots, t_{n-1})$, and $\text{FVSubst}(s, \psi) = \varphi$, then ψ is necessarily of the form $R(w_0, \ldots, w_{n-1})$, where $s(w_i) = t_i$, for $0 \leq i \leq n-1$. By Supplement 26, there are finitely many w_i such that $s(w_i) = t_i$ and therefore there are finitely many atomic formulas ψ such that $\text{FVSubst}(s, \psi) = \varphi$. Thus, we proved the basis steps.
Among the inductive steps, we discuss only the case when $\varphi = (Qx)\varphi_0$. If $\text{FVSubst}(s, \psi) = \varphi$, then ψ is of the form $\psi = (Qx)\psi_0$, where $\text{FVSubst}([x \to x]s, \psi_0) = \varphi_0$. Clearly, $[x \to x]s$ is a substitution of the specified form, so by inductive hypothesis, there are only finitely many formulas ψ_0 such

that $\text{FVSubst}([x \to x]s, \psi_0) = \varphi_0$. Therefore, there are finitely many formulas ψ with $\text{FVSubst}(s, \psi) = \varphi$.

(28) Show that the condition that Γ is quantifier-free is necessary in Theorem 4.3.43.

(29) Let s be a substitution and let φ be a formula. Prove that

$$\texttt{FV}(\text{FVSubst}(s, \varphi)) \subseteq \bigcup\{\texttt{V}(s(y)) \mid y \in \texttt{FV}(\varphi)\}.$$

Solution: This supplement complements Theorem 4.3.82 where instead of the above inclusion, we obtained an equality when s is admissible for φ. We will show by induction on φ that the result holds for every substitution s. The basis step follows from the theorem mentioned above, because every substitution is admissible for an atomic formula.

Among the several inductive steps, we consider only the one when $\varphi = (Qx)\psi$ and the result is assumed for ψ. We have

$$\begin{aligned}
&\texttt{FV}(\text{FVSubst}(s, (Qx)\psi)) \\
&\quad = \texttt{FV}((Qx)\text{FVSubst}([x \to x]s, \psi)) \\
&\quad\quad \text{(by Lemma 4.3.53)} \\
&\quad = \texttt{FV}(\text{FVSubst}([x \to x]s, \psi)) - \{x\} \\
&\quad\quad \text{(by Theorem 4.3.40)} \\
&\quad \subseteq \bigcup\{\texttt{V}([x \to x]s(y) \mid y \in \texttt{FV}(\psi))\} - \{x\} \\
&\quad\quad \text{(by inductive hypothesis)} \\
&\quad \subseteq \bigcup\{\texttt{V}(s(y)) \mid y \in \texttt{FV}(\psi) - \{x\}\} \cup \{x\}) - \{x\} \\
&\quad \subseteq \bigcup\{\texttt{V}(s(y)) \mid y \in \texttt{FV}((Qx)\psi)\} \\
&\quad = \bigcup\{\texttt{V}(s(y)) \mid y \in \texttt{FV}(\varphi)\}.
\end{aligned}$$

(30) Give an example of a formula φ and a substitution s such that

$$\texttt{FV}(\text{FVSubst}(s, \varphi)) \subset \bigcup\{\texttt{V}(s(y)) \mid y \in \texttt{FV}(\varphi)\}.$$

(31) Give an example of a formula φ and a substitution s such that s is not admissible for φ and yet the equality $\texttt{FV}(\text{FVSubst}(s, \varphi)) = \bigcup\{\texttt{V}(s(y)) \mid y \in \texttt{FV}(\varphi)\}$ holds.

(32) Show that the second inclusion of Corollary 4.3.84 holds even if t is not substitutable for x in φ.
Hint. Apply Supplement 29 with $s = \mathsf{s}_t^x$.

(33) Let s be a substitution. Prove that for every term t

$$\{x \in \mathtt{V}(t) \mid x \in \mathtt{V}(s(x))\} \subseteq \mathtt{V}(s(t)).$$

Hint. Use induction on terms.

(34) Prove that for any formula φ and substitution s, we have

$$\{x \in \mathtt{FV}(\varphi) \mid x \in \mathtt{V}(s(x))\} \subseteq \mathtt{FV}(\mathrm{FVSubst}(s, \varphi)).$$

Hint. Use induction on φ and Exercise 33 for the basis.

(35) Let φ be a formula, x be a variable, and t be a term. Prove that

$$\mathtt{FV}(\varphi) - \{x\} \subseteq \mathtt{FV}((\varphi)_{x:=t}).$$

Hint. Apply Exercise 34 with $s = \mathsf{s}_t^x$.

(36) Prove the following generalization of Lemma 4.6.46: let φ be a formula, x, z be variables and let u, t be terms. Prove that if $x \notin \mathtt{V}(u)$ and t is substitutable for x in φ, then t is substitutable for x in $(\varphi)_{z:=u}$.
Solution: The proof is by induction on the formula φ. The basis step, when φ is an atomic formula, is immediate. For the inductive steps, we discuss only the case when $\varphi = (Qw)\psi$.
Recall that t is substitutable for x in $(Qw)\psi$ if and only if either x does not occur free in $(Qw)\psi$ or t is substitutable for x in ψ and w does not occur in t. In the first case, x does not occur free in $((Qw)\psi)_{z:=u}$ by Exercise 32, so t is clearly substitutable for x in this formula. In the second case, by inductive hypothesis, t is substitutable for x in $(\psi)_{z:=u}$. If $z = w$, the conclusion follows immediately because $((Qw)\psi)_{z:=u} = (Qw)\psi$. If $z \neq w$, then $((Qw)\psi)_{z:=u} = (Qw)(\psi)_{z:=u}$ and the conclusion follows because w does not occur in t and t is substitutable for x in $(\psi)_{z:=u}$.

(37) Let α be a formula, x, y, z be variables and let c be a constant symbol. Prove that if y is substitutable for x in α, then y is substitutable for x in $(\alpha)_{z:=c}$.
Solution: The proof follows immediately from Supplement 36.

(38) Let t be a term that is substitutable for a variable x in a formula φ. If x does not occur in the terms $u_0, \ldots, u_{n-1}$ (where $n \geq 0$), prove that t is substitutable for x in $(\varphi)_{y_0:=u_0,\ldots,y_{n-1}:=u_{n-1}}$.
Hint. Use induction on n and Supplement 36.

Structures

(39) Let $\mathcal{L}$ be a first-order language with equality and let h be a morphism between the $\mathcal{L}$-structures $\mathcal{A} = (A, \mathcal{I})$ and $\mathcal{B} = (B, \mathcal{J})$. Prove that if for each n-ary relation symbol $R \in \mathcal{L}$, $a = (a_0, \ldots, a_{n-1}) \in R^{\mathcal{A}}$ if and only if $h \circ a = (h(a_0), \ldots, h(a_{n-1})) \in R^{\mathcal{B}}$, then h is an embedding.
Hint. Apply the above condition to the relation symbol $=$.

(40) Let $\mathcal{A}, \mathcal{B}$ be two $\mathcal{L}$-structures and let h be a morphism from $\mathcal{A}$ to $\mathcal{B}$. Prove that there are two $\mathcal{L}$-structures $\mathcal{C}$ and $\mathcal{D}$, an epimorphism h_{epi} from $\mathcal{A}$ to $\mathcal{C}$, an isomorphism h_{iso} from $\mathcal{C}$ to $\mathcal{D}$, and an embedding h_{emb} from $\mathcal{D}$ to $\mathcal{B}$ such that $h = h_{emb} \circ h_{iso} \circ h_{epi}$.
Hint. Take $\mathcal{C} = \mathcal{A}/\mathbf{ker}(h)$ and $\mathcal{D} = h(\mathcal{A})$.

Semantics of First-Order Logic

(41) Let $\mathcal{A}$ be an $\mathcal{L}$-structure, $\mathcal{B}$ be a substructure of $\mathcal{A}$ and let $\sigma \in \mathrm{ASSIGN}_{\mathcal{B}} \subseteq \mathrm{ASSIGN}_{\mathcal{A}}$.

(a) Show that for every term $t \in \mathrm{TERM}_{\mathcal{L}}$, $\sigma^{\mathcal{A}}(t) = \sigma^{\mathcal{B}}(t)$.
(b) Prove that if ψ is a quantifier-free $\mathcal{L}$-formula, then $(\mathcal{B}, \sigma) \models \psi$ if and only if $(\mathcal{A}, \sigma) \models \psi$.

Hint. The argument for Part (a) is by induction on terms. Part (b) can be shown by induction on quantifier-free formulas and one should use Part (a) for basis step.

(42) Let $\mathcal{L}$ be a first-order language, Γ be a set of $\mathcal{L}$-formulas and $\mathcal{A}$ be an $\mathcal{L}$-structure. Prove that if Γ is satisfiable in $\mathcal{A}$, then $\Gamma^{\exists}$ is satisfiable in $\mathcal{A}$. Give an example of a Γ and $\mathcal{A}$ such that $\Gamma^{\exists}$ is satisfiable in $\mathcal{A}$ but Γ is not.

(43) Let $(\varphi_0, \ldots, \varphi_{n-1})$ and $(\psi_0, \ldots, \psi_{m-1})$ be two nonempty sequences of first-order formulas such that $\{\varphi_0, \ldots, \varphi_{n-1}\} = \{\psi_0, \ldots, \psi_{m-1}\}$. Prove the following logical equivalences.

$$\bigvee_{i=0}^{n-1} \varphi_i \equiv \bigvee_{j=0}^{m-1} \psi_j$$

$$\bigwedge_{i=0}^{n-1} \varphi_i \equiv \bigwedge_{j=0}^{m-1} \psi_j.$$

As a consequence, prove that by permuting the members of a disjunction or a conjunction, we obtain logically equivalent formulas.

(44) Give an example of two logically equivalent formulas φ and ψ that have distinct sets of free variables. In fact, show that for every $\mathcal{L}$-formula φ where $\mathcal{L}$ contains a relation symbol with arity at least 1, there is logically equivalent $\mathcal{L}$-formula with a different set of free variables than φ.

(45) Show that the following formulas are logically valid for all formulas φ, ψ and variables x:

(a) $(((\forall x)\varphi \vee (\forall x)\psi) \to (\forall x)(\varphi \vee \psi))$;
(b) $((\exists x)(\varphi \wedge \psi) \to ((\exists x)\varphi \wedge (\exists x)\psi))$.

(46) Prove that for all formulas $\varphi_0, \ldots, \varphi_{n-1}$ with $n > 0$, and all variables x, we have

$$(\exists x) \bigvee_{i=0}^{n-1} \varphi_i \equiv \bigvee_{i=0}^{n-1} (\exists x)\varphi_i,$$

$$(\forall x) \bigwedge_{i=0}^{n-1} \varphi_i \equiv \bigwedge_{i=0}^{n-1} (\forall x)\varphi_i.$$

(47) Show for each of the following formulas that there is a choice for φ, ψ and x such that the formula is not logically valid.

(a) $(((\forall x)\varphi \vee (\forall x)\psi) \leftrightarrow (\forall x)(\varphi \vee \psi))$;
(b) $((\exists x)(\varphi \wedge \psi) \leftrightarrow ((\exists x)\varphi \wedge (\exists x)\psi))$.

Let $t_0, \ldots, t_{n-1} \in \mathrm{TERM}_{\mathcal{L}_{pra}}$. The term $(\cdots(t_0 + t_1) + \cdots + t_{n-1})$ will be denoted by $t_0 + \cdots + t_{n-1}$. We also introduce the notation mt for $m \in \mathbf{N}$ and $t \in \mathrm{TERM}_{\mathcal{L}_{pra}}$ defined inductively by $0t = 0$ and $(m+1)t = mt + t$.

(48) Prove that for all $\sigma \in \text{ASSIGN}_{\mathcal{A}_{pra}}$ and $t_0, \ldots, t_{n-1} \in \text{TERM}_{\mathcal{L}_{pra}}$, we have

$$\sigma^{\mathcal{A}_{pra}}(t_0 + \cdots + t_{n-1}) = \sigma^{\mathcal{A}_{pra}}(t_0) + \cdots + \sigma^{\mathcal{A}_{pra}}(t_{n-1}).$$

(49) Let t and u be $\mathcal{L}$-terms such that (u, i) is an occurrence in t and let z be a variable that does not occur in t. If $\mathcal{A}$ is an $\mathcal{L}$-structure and $\sigma \in \text{ASSIGN}_{\mathcal{A}}$, then prove that

$$\sigma^{\mathcal{A}}(t) = ([z \to a]\sigma)^{\mathcal{A}}(t'),$$

where $t' = \texttt{replace}\,(t, (u, i), z)$ and $a = \sigma^{\mathcal{A}}(u)$.
Hint. Note that by Theorem 1.5.30, t' is an $\mathcal{L}$-term. Use induction on t.

(50) Let $\mathcal{L}$ be a first-order language, $\mathcal{A}$ be an $\mathcal{L}$-structure, $\sigma \in \text{ASSIGN}_{\mathcal{A}}$ and let u, t_0, t_1 be three $\mathcal{L}$-terms such that $\sigma^{\mathcal{A}}(t_0) = \sigma^{\mathcal{A}}(t_1)$. If ζ is an occurrence of t_0 in u, then, prove that

$$\sigma^{\mathcal{A}}(\texttt{replace}\,(u, \zeta, t_1)) = \sigma^{\mathcal{A}}(u).$$

Solution: First consider the special case when $t_0 = u$. Then,

$$\sigma^{\mathcal{A}}(\texttt{replace}\,(u, \zeta, t_1)) = \sigma^{\mathcal{A}}(t_1) = \sigma^{\mathcal{A}}(t_0) = \sigma^{\mathcal{A}}(u).$$

We now prove the result by induction on u. The basis steps, when u is either a variable or a constant symbol, are covered by the special case. Now suppose that $u = f(u_0, \ldots, u_{n-1})$. If we are not in the special case, by the Occurrence Theorem (Theorem 1.5.27), there is a term u_j and an occurrence $\zeta' = (t_0, k')$ in u_j that corresponds to the occurrence $\zeta = (t_0, k)$ of t_0 in u. We have

$$\begin{aligned}
&\sigma^{\mathcal{A}}(\texttt{replace}\,(u, \zeta, t_1)) \\
&\quad = f^{\mathcal{A}}(\sigma^{\mathcal{A}}(u_0), \ldots, \sigma^{\mathcal{A}}(\texttt{replace}\,(u_j, \zeta', t_1)), \ldots, \sigma^{\mathcal{A}}(u_{n-1})) \\
&\quad = f^{\mathcal{A}}(\sigma^{\mathcal{A}}(u_0), \ldots, \sigma^{\mathcal{A}}(u_j), \ldots, \sigma^{\mathcal{A}}(u_{n-1})) \\
&\qquad \text{(by inductive hypothesis)} \\
&\quad = \sigma^{\mathcal{A}}(f(u_0, \ldots, u_{n-1})) = \sigma^{\mathcal{A}}(u).
\end{aligned}$$

Let $\mathcal{L}$ be a first-order language and let $\mathcal{A}$ be an $\mathcal{L}$-structure.

$\mathcal{L}$-terms t, u are $\mathcal{A}$-*equivalent* (denoted by $t \equiv_{\mathcal{A}} u$) if for every $\sigma \in \text{ASSIGN}_{\mathcal{A}}$, $\sigma^{\mathcal{A}}(t) = \sigma^{\mathcal{A}}(u)$.

(51) Let $\mathcal{L}$ be a first-order language and suppose that $= \in \mathcal{L}$. Prove that if t and u are $\mathcal{L}$-terms, then $t \equiv_{\mathcal{A}} u$ if and only if $\mathcal{A} \models (t = u)$.

(52) Let $\mathcal{L}$ be a first-order language and let $\mathcal{A}$ be an $\mathcal{L}$-structure.

(a) Prove that $\equiv_{\mathcal{A}}$ is an equivalence relation on $\mathrm{FORM}_{\mathcal{L}}$.
(b) Prove that $\varphi \equiv_{\mathcal{A}} \psi$ if and only if $\mathcal{A} \models (\varphi \leftrightarrow \psi)$.

(53) Let $\mathcal{L}$ be a first-order language with equality. Prove that for every $k \geq 1$, there exist three closed $\mathcal{L}$-formulas $\varphi_k, \psi_k, \theta_k$ such that for every $\mathcal{L}$-structure $\mathcal{A}$, $\mathcal{A}$ is a model of these formulas if and only if $|\mathcal{A}| \leq k$, $|\mathcal{A}| \geq k$, and $|\mathcal{A}| = k$, respectively.
Solution: We can take φ_k to be

$$(\exists x_0) \cdots (\exists x_{k-1})(\forall x_k)(x_0 = x_k \vee x_1 = x_k \vee \cdots \vee x_{k-1} = x_k),$$

for $k \geq 1$.
If $k = 1$, we can take $\psi_k = (\exists x_0)(x_0 = x_0)$. If $k > 1$, then we can define ψ_k as

$$(\exists x_0) \cdots (\exists x_{k-1})(x_0 \neq x_1 \wedge \cdots \wedge x_0 \neq x_{k-1} \\ \wedge x_1 \neq x_2 \wedge \cdots \wedge x_{k-2} \neq x_{k-1}).$$

Finally, we can take $\theta_k = (\varphi_k \wedge \psi_k)$.

(54) Let $\mathcal{L}$ be a first-order language, φ be an $\mathcal{L}$-formula and $\mathcal{A}$ be an $\mathcal{L}$-structure with $|\mathcal{A}| = 1$. Show that if φ is satisfiable in $\mathcal{A}$, then φ is valid in $\mathcal{A}$.

(55) Let $\mathcal{L}$ be a first-order language that contains a unary relation symbol R and let φ be the $\mathcal{L}$-formula $(R(x) \wedge (\neg(\forall y)R(y)))$, where x and y are variables. Prove that φ is satisfiable but φ is not valid in any $\mathcal{L}$-structure.

(56) Let $\mathcal{L}$ be a first-order language with equality. Give an example of an $\mathcal{L}$-formula that is satisfiable in every $\mathcal{L}$-structure but is not valid in any $\mathcal{L}$-structure $\mathcal{A}$ such that $|\mathcal{A}| > 1$.

(57) Let $\mathcal{L}$ be a first-order language without equality. If Γ is a satisfiable set of $\mathcal{L}$-formulas, prove that Γ is satisfiable in an infinite structure.

Solution: Suppose that $(\mathcal{A}, \sigma) \models \Gamma$. Define an $\mathcal{L}$-structure $\mathcal{B}$ by $|\mathcal{B}| = |\mathcal{A}| \times \mathbf{N}$ and

$$R^{\mathcal{B}} = \{((a_0, i_0), \ldots, (a_{n-1}, i_{n-1})) \mid (a_0, \ldots, a_{n-1}) \in R^{\mathcal{A}}\}$$

and

$$f^{\mathcal{B}}((a_0, i_0), \ldots, (a_{n-1}, i_{n-1})) = (f^{\mathcal{A}}(a_0, \ldots, a_{n-1}), 0)$$

for every n-ary relation symbol $R \in \mathcal{L}$ and every n-ary function symbol $f \in \mathcal{L}$. Since $|\mathcal{A}| \neq \emptyset$, we have $|\mathcal{B}|$ infinite. Observe that the mapping $h : |\mathcal{B}| \longrightarrow |\mathcal{A}|$ given by $h((a, i)) = a$ for every $(a, i) \in |\mathcal{B}|$ is an epimorphism. Define $\tau \in \mathrm{ASSIGN}_{\mathcal{B}}$ by $\tau(x) = (\sigma(x), 0)$. By the Morphism Theorem, for every $\mathcal{L}$-formula φ, $(\mathcal{B}, \tau) \models \varphi$ if and only if $(\mathcal{A}, h \circ \tau) \models \varphi$. Since $h \circ \tau = \sigma$ and $(\mathcal{A}, \sigma) \models \Gamma$, it follows that $(\mathcal{B}, \tau) \models \Gamma$.

(58) Let $\mathcal{L}$ be a first-order language without equality and Γ be a set of $\mathcal{L}$-formulas. Prove that if Γ has a model, then Γ has an infinite model.

(59) Let $\mathcal{L}$ be a first-order language, Γ be a set of $\mathcal{L}$-formulas, and φ be an $\mathcal{L}$-formula. Show that if $\Gamma \models \varphi$, then $\Gamma \models\!\!\approx \varphi$, where "$\models\!\!\approx$" is the weak logical implication introduced in Definition 4.5.31. Show that if $\mathcal{L}$ is a language with at least one relation symbol with arity at least 1, then there is a set of $\mathcal{L}$-formulas Γ and an $\mathcal{L}$-formula φ such that $\Gamma \models\!\!\approx \varphi$ but $\Gamma \not\models \varphi$.
Solution: We discuss only the counterexample required by the second part of the supplement. Suppose that R is an n-ary relation symbol of $\mathcal{L}$ with $n \geq 1$. Let $\Gamma = \{R(x, \ldots, x)\}$ and $\varphi = R(y, \ldots, y)$, where x and y are distinct variables. Clearly, $\Gamma \models\!\!\approx \varphi$.
On the other hand, let $\mathcal{A}$ be an $\mathcal{L}$-structure with $A = \{a_0, a_1\}$ and $R^{\mathcal{A}} = \{(a_0, \ldots, a_0)\}$. If $\sigma(x) = a_0$ and $\sigma(y) = a_1$, then we have $(\mathcal{A}, \sigma) \models \Gamma$, but $(\mathcal{A}, \sigma) \not\models \varphi$, so $\Gamma \not\models \varphi$.

(60) Let $\mathcal{L}$ be a first-order language, Γ be a set of $\mathcal{L}$-sentences, and φ be an $\mathcal{L}$-formula. Prove that $\Gamma \models\!\!\approx \varphi$ if and only if $\Gamma \models \varphi$.
Solution: In view of Supplement 59, we need to show only that $\Gamma \models\!\!\approx \varphi$ implies $\Gamma \models \varphi$. Suppose that $(\mathcal{A}, \sigma) \models \Gamma$. By Corollary 4.5.32, we have $\mathcal{A} \models \Gamma$, so $\mathcal{A} \models \varphi$, which implies $(\mathcal{A}, \sigma) \models \varphi$.

(61) Let $\mathcal{L}$ be a first-order language, Γ be a set of $\mathcal{L}$-formulas, and φ be an $\mathcal{L}$-formula. Prove that $\Gamma \models\!\!\approx \varphi$ if and only if $\Gamma^{\forall} \models \varphi$.
Solution: Since $\Gamma^{\forall}$ is a set of sentences, we have $\Gamma^{\forall} \models \varphi$ if and only if $\Gamma^{\forall} \models\!\!\approx \varphi$, by Supplement 60. By Corollary 4.5.60, Γ and $\Gamma^{\forall}$ have the same models, which implies that $\Gamma^{\forall} \models\!\!\approx \varphi$ if and only if $\Gamma \models\!\!\approx \varphi$. By combining these two observations, we obtain the desired result.

(62) Prove that the analogue of the third part of Theorem 4.5.52 obtained by replacing $\models$ with $\models\!\!\approx$ is false. In other words, show that the statement "$\Gamma \cup \{\varphi\} \models\!\!\approx \psi$ if and only if $\Gamma \models\!\!\approx (\varphi \to \psi)$" is false. However, show that the implication "if $\Gamma \models\!\!\approx (\varphi \to \psi)$, then $\Gamma \cup \{\varphi\} \models\!\!\approx \psi$" is true.
Solution: Let $\Gamma = \emptyset$, $\varphi = R(x)$ and $\psi = R(y)$, where R is a unary relation symbol and x, y are two distinct variables. It is clear that $R(x) \models\!\!\approx R(y)$ and it is equally clear that $(R(x) \to R(y))$ is not logically valid, which proves the first point of the statement. We leave to the reader the proof of the second part.

(63) Let P, R be two relation symbols of the same arity and let φ be a formula. Prove that:

(a) $\mathtt{FV}(\mathsf{s}_R^P(\varphi)) = \mathtt{FV}(\varphi)$;
(b) $\mathsf{s}_R^P(\varphi^{\forall}) = \mathsf{s}_R^P(\varphi)^{\forall}$.

Further, prove that if Γ is a set of formulas, then $\mathsf{s}_R^P(\Gamma^{\forall}) = \mathsf{s}_R^P(\Gamma)^{\forall}$.
Solution: The first part can be shown by induction on φ and is left to the reader. For the second part, suppose that $\mathtt{FV}(\varphi) = \{y_0, \ldots, y_{n-1}\}$, where the variables are listed in the standard order. We have

$$\begin{aligned}
\mathsf{s}_R^P(\varphi^{\forall}) &= \mathsf{s}_R^P((\forall y_0) \cdots (\forall y_{n-1})\varphi)\\
&= (\forall y_0) \cdots (\forall y_{n-1})\mathsf{s}_R^P(\varphi)\\
&\quad \text{(because } \mathsf{s}_R^P \text{ is a textual substitution not involving}\\
&\quad \text{any of the quantified variables)}\\
&= \mathsf{s}_R^P(\varphi)^{\forall}.
\end{aligned}$$

The last part of the supplement is an immediate consequence of the second part.

(64) Show that the first part of the Morphism Theorem may fail if φ contains $=$ or quantifiers. Similarly, show that the second part may fail if φ contains $=$ and the third part may fail if φ contains quantifiers.

(65) Let $\mathcal{A}, \mathcal{B}$ be two $\mathcal{L}$-structures and let $h : |\mathcal{A}| \longrightarrow |\mathcal{B}|$ be a function such that for all $\sigma \in \text{ASSIGN}_{\mathcal{A}}$ and atomic $\mathcal{L}$-formulas φ not containing $=$, $(\mathcal{A}, \sigma) \models \varphi$ if and only if $(\mathcal{B}, h \circ \sigma) \models \varphi$. Show that h satisfies the second condition of the definition of morphism (Definition 4.4.12). Similarly, prove that if $\mathcal{L}$ contains $=$ and the above condition is satisfied for all atomic formulas, then h is a monomorphism.

(66) Prove that the set of signed formulas Δ_n introduced in Exercise 9 is unsatisfiable.

Solution: Since Δ_n consists of signed sentences, we must show that Δ_n has no model. Suppose that $\mathcal{A}$ were a model for Δ_n. Define inductively the terms t_i for $i \geq 1$ by $t_1 = c_1$ and $t_{i+1} = s(t_i, c_1)$. Since $\mathcal{A} \models \{\mathbf{T}\alpha, \mathbf{T}\beta\}$, it follows immediately that $t_i^{\mathcal{A}} \in L^{\mathcal{A}}$ for all $i \geq 1$.

Next, we show that $s(t_p, t_q)^{\mathcal{A}} = t_{p+q}^{\mathcal{A}}$ for $p, q \geq 1$. The argument is by induction on q. The basis step is immediate from the definition of t_{p+1}. Suppose that $s(t_p, t_q)^{\mathcal{A}} = t_{p+q}^{\mathcal{A}}$. Since $\mathcal{A} \models \mathbf{T}\varphi$, we have the equality $s(u, s(v, w))^{\mathcal{A}} = s(s(u, v), w)^{\mathcal{A}}$ for all ground terms u, v, w. Therefore,

$$\begin{aligned} s(t_p, t_{q+1})^{\mathcal{A}} &= s(t_p, s(t_q, c_1))^{\mathcal{A}} \\ &\quad \text{(by definition of } t_{q+1}) \\ &= s(s(t_p, t_q), c_1)^{\mathcal{A}} \\ &\quad \text{(as shown above)} \\ &= s(t_{p+q}, c_1)^{\mathcal{A}} \\ &\quad \text{(by inductive hypothesis)} \\ &= t_{p+q+1}^{\mathcal{A}}. \end{aligned}$$

We prove now that $\underbrace{d(\cdots d}_{k}(c_1)\cdots)^{\mathcal{A}} = t_{2^k}^{\mathcal{A}}$ by induction on $k \geq 0$. The basis step, $k = 0$, is immediate. For the inductive step, since $\mathcal{A} \models \mathbf{T}\psi$, we have $d(t)^{\mathcal{A}} = s(t, t)^{\mathcal{A}}$, for every ground term t. Suppose that $\underbrace{d(\cdots d}_{k}(c_1)\cdots)^{\mathcal{A}} = t_{2^k}^{\mathcal{A}}$. Then we can write

$$\begin{aligned}\underbrace{d(\cdots d}_{k+1}(c_1)\cdots)^{\mathcal{A}} &= s(\underbrace{d(\cdots d}_{k}(c_1)\cdots), \underbrace{d(\cdots d}_{k}(c_1)\cdots))^{\mathcal{A}} \\ &\quad \text{(since } \mathcal{A} \models \mathbf{T}\psi\text{)} \\ &= s(t_{2^k}, t_{2^k})^{\mathcal{A}} \\ &\quad \text{(by inductive hypothesis)} \\ &= t_{2^{k+1}}^{\mathcal{A}} \\ &\quad \text{(as shown previously).}\end{aligned}$$

Since $t_i^{\mathcal{A}} \in L^{\mathcal{A}}$ for $i \geq 1$, we have $t_{2^{2^n}}^{\mathcal{A}} \in L^{\mathcal{A}}$ for $n \geq 0$. From the equality $\underbrace{d(\cdots d}_{2^n}(c_1)\cdots)^{\mathcal{A}} = t_{2^{2^n}}^{\mathcal{A}}$, it follows that $\underbrace{d(\cdots d}_{2^n}(c_1)\cdots)^{\mathcal{A}} \in L^{\mathcal{A}}$, which contradicts the fact that $\mathcal{A} \models \mathbf{F}\theta_n$. Therefore, Δ_n is unsatisfiable.

(67) Let $\mathcal{L} = \{=, L, c_1, d, s\}$ be the first-order language introduced in Exercise 9 and let γ be the $\mathcal{L}$-formula $(L(y) \wedge (\forall x)(L(x) \rightarrow L(s(x, y))))$. Prove that the set of formulas $\{\mathbf{T}\varphi, \mathbf{T}\psi, \mathbf{T}(\gamma)_{y:=a}, \mathbf{F}(\gamma)_{y:=d(a)}\}$ is unsatisfiable, where $\mathbf{T}\varphi, \mathbf{T}\psi$ are the formulas introduced in Exercise 9.

Solution: Suppose that $\mathcal{A} \models \{\mathbf{T}\varphi, \mathbf{T}\psi, \mathbf{T}(\gamma)_{y:=a}\}$. We will show that $\mathcal{A} \models \mathbf{T}(\gamma)_{y:=d(a)}$, thereby showing that the set of formulas in the statement of the exercise is unsatisfiable. We begin by observing that since $\mathcal{A} \models \{\mathbf{T}\varphi, \mathbf{T}\psi\}$, we have,

$$s^{\mathcal{A}}(a_0, s^{\mathcal{A}}(a_1, a_2)) = s^{\mathcal{A}}(s^{\mathcal{A}}(a_0, a_1), a_2) \tag{4.21}$$

$$d^{\mathcal{A}}(a_0) = s^{\mathcal{A}}(a_0, a_0) \tag{4.22}$$

for all $a_0, a_1, a_2 \in |\mathcal{A}|$. Since $\mathcal{A} \models \mathbf{T}(\gamma)_{y:=a}$, we have

(i) $a^{\mathcal{A}} \in L^{\mathcal{A}}$, and
(ii) for all $b \in L^{\mathcal{A}}$, we have $s^{\mathcal{A}}(b, a^{\mathcal{A}}) \in L^{\mathcal{A}}$.

We must prove that $d^{\mathcal{A}}(a^{\mathcal{A}}) \in L^{\mathcal{A}}$ and that $s^{\mathcal{A}}(c, d^{\mathcal{A}}(a^{\mathcal{A}})) \in L^{\mathcal{A}}$ for all $c \in L^{\mathcal{A}}$. Choosing $b = a^{\mathcal{A}}$ in (ii), we have $s^{\mathcal{A}}(a^{\mathcal{A}}, a^{\mathcal{A}}) \in L^{\mathcal{A}}$, which implies that $d^{\mathcal{A}}(a^{\mathcal{A}}) \in L^{\mathcal{A}}$ by Equation (4.22). If

$c \in L^{\mathcal{A}}$, we have $s^{\mathcal{A}}(c, a^{\mathcal{A}}) \in L^{\mathcal{A}}$, by (ii). A second application of (ii) gives $s^{\mathcal{A}}(s^{\mathcal{A}}(c, a^{\mathcal{A}}), a^{\mathcal{A}}) \in L^{\mathcal{A}}$, which implies $s^{\mathcal{A}}(c, s^{\mathcal{A}}(a^{\mathcal{A}}, a^{\mathcal{A}})) \in L^{\mathcal{A}}$ by Equation (4.21). Finally, by Equation (4.22), $s^{\mathcal{A}}(c, d^{\mathcal{A}}(a^{\mathcal{A}})) \in L^{\mathcal{A}}$.

(68) Let α be an $(\mathcal{L}, V)$-instance of an $\mathcal{L}$-equality axiom φ, a be a constant symbol in $\mathcal{L}$ and t be an $(\mathcal{L}, V)$-term. Prove that $\mathsf{s}_t^a(\alpha)$ is also an $(\mathcal{L}, V)$-instance of φ.
Hint. Use Theorems 4.3.68 and 4.3.69.

Semantics of Substitutions and Replacements

(69) Prove that if φ is an $\mathcal{L}$-formula and R, R' are two relation symbols of the same arity, then $\mathsf{s}_{R'}^{R}(\varphi)$ is an $\mathcal{L}'$-formula, where $\mathcal{L}' = (\mathcal{L} - \{R\}) \cup \{R'\}$.

(70) Let Γ be a set of formulas and let y be a variable that does not occur free in any formula of Γ and is substitutable for x in φ. Prove that:

(a) if $(\exists x)\varphi \in \Gamma$ and Γ is satisfiable, then $\Gamma \cup \{(\varphi)_{x:=y}\}$ is satisfiable;
(b) if $(\neg(\forall x)\varphi) \in \Gamma$ and Γ is satisfiable, then $\Gamma \cup \{((\neg\varphi))_{x:=y}\}$ is satisfiable.

(71) Let Γ be a set of formulas and let y be a variable such that $y \notin \mathtt{FV}(\Gamma)$. Prove that:

(a) If $(\exists x)\varphi \in \Gamma$ and Γ is satisfiable, then $\Gamma \cup \{\langle\varphi\rangle_{x:=y}\}$ is satisfiable.
(b) If $(\neg(\forall x)\varphi) \in \Gamma$ and Γ is satisfiable, then $\Gamma \cup \{\langle(\neg\varphi)\rangle_{x:=y}\}$ is satisfiable.

(72) Using Exercise 38, prove the following result which generalizes both Theorem 4.6.11 and 4.6.18:
Let φ be a formula and let $y_0, \ldots, y_{n-1}, z_0, \ldots, z_{n-1}$ be variables $(n \geq 0)$ such that

(a) $z_0, \ldots, z_{n-1}$ are distinct:
(b) $z_0, \ldots, z_{n-1}$ do not occur free in φ;
(c) for $0 \leq i \leq n-1$, z_i is substitutable for y_i in φ;
(d) for $0 \leq i \leq n-1$, $y_i \notin \{z_{i+1}, \ldots, z_{n-1}\}$.

Then,

$$\begin{aligned}&(Q_0 y_0) \cdots (Q_{n-1} y_{n-1})\varphi \\ &\quad \equiv (Q_0 z_0) \cdots (Q_{n-1} z_{n-1})(\varphi)_{y_{n-1}:=z_{n-1},\ldots,y_0:=z_0}\end{aligned}$$

and these formulas have the same free variables.
Solution: The proof is by induction on n. The basis step ($n = 0$) is immediate. For the inductive step, suppose that the statement holds for $n \geq 0$. Suppose that

(a) $z_0, \ldots, z_n$ are distinct:
(b) $z_0, \ldots, z_n$ do not occur free in φ;
(c) for $0 \leq i \leq n$, z_i is substitutable for y_i in φ;
(d) for $0 \leq i \leq n$, $y_i \notin \{z_{i+1}, \ldots, z_n\}$.

By inductive hypothesis, we have

$$\begin{aligned}&(Q_1 y_1) \cdots (Q_n y_n)\varphi \\ &\quad \equiv (Q_1 z_1) \cdots (Q_n z_n)(\varphi)_{y_n:=z_n,\ldots,y_1:=z_1}\end{aligned}$$

and the two formulas in the previous equivalence have the same set of free variables. Therefore, we have

$$\begin{aligned}&(Q_0 y_0)(Q_1 y_1) \cdots (Q_n y_n)\varphi \\ &\quad \equiv (Q_0 y_0)(Q_1 z_1) \cdots (Q_n z_n)(\varphi)_{y_n:=z_n,\ldots,y_1:=z_1}\end{aligned}$$

by Theorem 4.6.16. Since

$$\begin{aligned}&\mathtt{FV}((Q_1 y_1) \cdots (Q_n y_n)\varphi) \\ &\quad = \mathtt{FV}((Q_1 z_1) \cdots (Q_n z_n)(\varphi)_{y_n:=z_n,\ldots,y_1:=z_1})\end{aligned}$$

and z_0 does not occur free in φ, it follows that z_0 does not occur free in $(Q_1 z_1) \cdots (Q_n z_n)(\varphi)_{y_n:=z_n,\ldots,y_1:=z_1}$. Since z_0 is substitutable for y_0 in φ and $z_0 \notin \{z_1, \ldots, z_n\}$, it follows by Exercise 38 that z_0 is substitutable for y_0 in $(\varphi)_{y_n:=z_n,\ldots,y_1:=z_1}$.

By Theorem 4.6.11, we have

$$\begin{aligned}&(Q_0 y_0)(Q_1 z_1)\cdots(Q_n z_n)(\varphi)_{y_n:=z_n,\ldots,y_1:=z_1}\\ &\quad\equiv (Q_0 z_0)\langle(Q_1 z_1)\cdots(Q_n z_n)(\varphi)_{y_n:=z_n,\ldots,y_1:=z_1}\rangle_{y_0:=z_0}\\ &\quad\equiv (Q_0 z_0)(Q_1 z_1)\cdots(Q_n z_n)(\varphi)_{y_n:=z_n,\ldots,y_0:=z_0}\\ &\qquad(\text{since } y_0\notin\{z_1,\ldots,z_n\}),\end{aligned}$$

so

$$\begin{aligned}&(Q_0 y_0)\cdots(Q_n y_n)\varphi\\ &\quad\equiv (Q_0 z_0)\cdots(Q_n z_n)(\varphi)_{y_n:=z_n,\ldots,y_0:=z_0}.\end{aligned}$$

Also,

$$\begin{aligned}&\texttt{FV}((Q_0 z_0)\cdots(Q_n z_n)(\varphi)_{y_n:=z_n,\ldots,y_0:=z_0})\\ &\quad= \texttt{FV}((Q_0 y_0)(Q_1 z_1)\cdots(Q_n z_n)(\varphi)_{y_n:=z_n,\ldots,y_1:=z_1})\\ &\qquad(\text{by Theorem 4.6.11})\\ &\quad= \texttt{FV}((Q_1 z_1)\cdots(Q_n z_n)(\varphi)_{y_n:=z_n,\ldots,y_1:=z_1}) - \{y_0\}\\ &\quad= \texttt{FV}((Q_1 y_1)\cdots(Q_n y_n)\varphi) - \{y_0\}\\ &\qquad(\text{by inductive hypothesis})\\ &\quad= \texttt{FV}((Q_0 y_0)\cdots(Q_n y_n)\varphi).\end{aligned}$$

(73) Give an example of a formula φ, a variable x and a term t such that $(\varphi)_{x:=t} \not\models (\exists x)\varphi$. (Note that by Theorem 4.6.7, t cannot be substitutable for x in φ.)

(74) Give an example of a set Δ of signed formulas, a $\boldsymbol{\delta}$-formula $b(Qx)\varphi$, and a constant symbol c such that the implication of Theorem 4.6.53 fails. Note that c must appear in $\Delta\cup\{b(Qx)\varphi\}$. **Hint.** Choose $\Delta=\emptyset$ and $b(Qx)\varphi = \mathbf{T}(\exists x)(P(c)\wedge(\neg P(x)))$.

(75) Let $\mathcal{L}$ be a first-order language, $\mathcal{A}$ be an $\mathcal{L}$-structure, and let $u_0, u_1 \in \mathrm{TERM}_{\mathcal{L}}$.

(a) If $t\in\mathrm{TERM}_{\mathcal{L}}$ and $t' = \texttt{replace}\,(t,(u_0,i),u_1)$, where (u_0,i) is an occurrence of u_0 in t, show that t' is a term in $\mathrm{TERM}_{\mathcal{L}}$.

(b) If $u_0 \equiv_{\mathcal{A}} u_1$, then prove that $t\equiv_{\mathcal{A}} t'$.

Hint. The arguments are by induction on the term t and make use of the Occurrence Theorem for Terms (Theorem 1.5.27).

(76) Let $\mathcal{L}$ be a first-order language, $\mathcal{A}$ be an $\mathcal{L}$-structure, and let $u_0, u_1 \in \mathrm{TERM}_{\mathcal{L}}$.

(a) If $\varphi \in \mathrm{FORM}_{\mathcal{L}}$ and $\psi = \texttt{replace}(\varphi, (u_0, i), u_1)$, where (u_0, i) is an occurrence of u_0 in φ that is not an occurrence of a variable immediately following a quantifier symbol, then prove that $\psi \in \mathrm{FORM}_{\mathcal{L}}$.

(b) If $u_0 \equiv_{\mathcal{A}} u_1$, then show that $\varphi \equiv_{\mathcal{A}} \psi$.

(77) Let $t \in \mathrm{TERM}_{\mathcal{L}_{pra}}$ and assume that the variables that occur in t in standard order are $y_0, \ldots, y_{k-1}$. Prove that $t \equiv_{\mathcal{A}_{pra}} s^j(0) + n_0 y_0 + \cdots + n_{k-1} y_{k-1}$ for some $j, n_0, \ldots, n_{k-1} \in \mathbf{N}$ and that there is an effective way to determine the numbers $j, n_0, \ldots, n_{k-1}$ from t.
Hint. Use induction on terms.

(78) Let α be an $\mathcal{L}$-formula and let s, s' be $\mathcal{L}$-substitutions, where $\mathcal{L}$ is a first-order language. Prove that if $s(x) = s'(x)$ for every $x \in \texttt{FV}(\alpha)$, then s is admissible for α if and only if s' is admissible for α.

(79) Let α be an $\mathcal{L}$-formula and let s be an $\mathcal{L}$-substitution, where $\mathcal{L}$ is a first-order language. Prove that if x, y are two variables such that y does not occur in α and $[x \to x]s$ is admissible for α, then $[y \to y]s$ is admissible for $(\alpha)_{x:=y}$.
Solution: The argument is by induction on α. We discuss only the inductive step when $\alpha = (Qz)\beta$. Since $[x \to x]s$ is admissible for α, by the fourth part of Corollary 4.3.81, it follows that $[z \to z][x \to x]s$ is admissible for β and if $w \in \texttt{FV}(\alpha)$ (that is, if $w \in \texttt{FV}(\beta) - \{z\}$), then z does not occur in $([x \to x]s)(w)$. In addition, since y does not occur in α, we have $y \neq z$ and y does not occur in β.
If $z = x$, then $(\alpha)_{x:=y} = \alpha$. Since neither x nor y occurs free in α, the substitutions $[x \to x]s$ and $[y \to y]s$ agree on all free variables of α, which gives the desired conclusion, due to Exercise 78. Therefore, we can assume $z \neq x$, which implies $(\alpha)_{x:=y} = (\forall z)(\beta)_{x:=y}$. To prove the admissibility of $[y \to y]s$ for $(\alpha)_{x:=y}$, we need to show

(A) $[z \to z][y \to y]s$ is admissible for $(\beta)_{x:=y}$, and

(B) if $w' \in \texttt{FV}((\forall z)(\beta)_{x:=y})$, then z does not occur in $([y \to y]s)(w')$.

To show (A), note that $[x \to x][z \to z]s = [z \to z][x \to x]s$ and therefore $[x \to x][z \to z]s$ is admissible for β. By the inductive hypothesis, $[y \to y][z \to z]s$ is admissible for $(\beta)_{x:=y}$ because y does not occur in β. Reversing the fixed variables y and z, we obtain (A).

To show (B), let w' be a free variable of $(\forall z)(\beta)_{x:=y}$. It is clear that $w' \in (\mathtt{FV}(\beta) - \{x, z\}) \cup \{y\}$. If $w' \neq y$, then $([y \to y]s)(w') = s(w') = ([x \to x]s)(w')$ and therefore z does not occur in $([y \to y]s)(w')$ because of the admissibility of $[x \to x]s$ for α. If $w' = y$, then $([y \to y]s)(w') = y$ and the conclusion follows immediately because $z \neq y$.

(80) Let φ and ψ be two formulas which are immediate variants. Prove that if x is a variable and t is a term such that t is substitutable for x in both φ and ψ, then $(\psi)_{x:=t}$ is an immediate variant of $(\varphi)_{x:=t}$.

Solution: The proof is by induction on φ. For the basis step φ is atomic, so $\varphi = \psi$, hence $(\varphi)_{x:=t} = (\psi)_{x:=t}$.

For the inductive step we consider several cases depending on the structure of φ.

Suppose that $\varphi = (\varphi_0 C \varphi_1)$, where C is a binary connective symbol, and the result holds for φ_0, φ_1. Then, either $\psi = (\psi_0 C \varphi_1)$ where ψ_0 is an immediate variant of φ_0, or $\psi = (\varphi_0 C \psi_1)$, where ψ_1 is an immediate variant of φ_1. In the first subcase, since t is substitutable for x in both φ and ψ, t is substitutable for x in φ_0 and ψ_0, so, by the inductive hypothesis, $(\varphi_0)_{x:=t}$ is an immediate variant of $(\psi_0)_{x:=t}$ and therefore $(\varphi)_{x:=t} = ((\varphi_0)_{x:=t} C (\varphi_1)_{x:=t})$ is an immediate variant of $((\psi_0)_{x:=t} C (\varphi_1)_{x:=t}) = (\psi)_{x:=t}$. The other subcase is similar.

The case when $\varphi = (\neg \varphi_0)$ is similar.

Suppose now that $\varphi = (Qy)\varphi_0$ and the result holds for φ_0. If x does not occur free in φ, then it does not occur free in ψ (since variants have the same free variables) and $(\varphi)_{x:=t} = \varphi$ is an immediate variant of $\psi = (\psi)_{x:=t}$, so we can assume that x occurs free in φ and therefore in ψ. Now we need to consider the following two subcases which depend on the place where the renaming was applied.

In the first subcase, $\psi = (Qy)\psi_0$ where ψ_0 is an immediate variant of φ_0. Since x occurs free in φ and ψ and t is substitutable for x in φ, ψ, we have $x \neq y$ and t is substitutable for x in

φ_0, ψ_0. By the inductive hypothesis, $(\varphi_0)_{x:=t}$ is an immediate variant of $(\psi_0)_{x:=t}$. Since $x \neq y$, $(\varphi)_{x:=t} = (Qy)(\varphi_0)_{x:=t}$ is an immediate variant of $(Qy)(\psi_0)_{x:=t} = (\psi)_{x:=t}$.
In the second subcase, $\psi = (Qz)(\varphi_0)_{y:=z}$ where z does not occur free in φ_0 and z is substitutable for y in this formula. Since x occurs free in φ, ψ, x is different from y and z and since t is substitutable for x in φ, ψ, y and z do not occur in t and t is substitutable for x in φ_0 and $(\varphi_0)_{y:=z}$. We have:

$$(\varphi)_{x:=t} = (Qy)(\varphi_0)_{x:=t} \text{ and } (\psi)_{x:=t} = (Qz)((\varphi_0)_{y:=z})_{x:=t}.$$

Further, we can write

$$\begin{aligned}
&((\varphi_0)_{y:=z})_{x:=t} \\
&\quad = \text{FVSubst}(\mathsf{s}^x_t, \text{FVSubst}(\mathsf{s}^y_z, \varphi_0)) \\
&\quad = \text{FVSubst}(\mathsf{s}^x_t * \mathsf{s}^y_z, \varphi_0) \\
&\qquad \text{(by Theorem 4.3.86 since } \mathsf{s}^y_z \text{ is admissible for } \varphi_0) \\
&\quad = \text{FVSubst}(s^{x\,y}_{t\,z}, \varphi_0) \\
&\qquad \text{(by Theorem 1.2.21 since } z \neq x) \\
&\quad = \text{FVSubst}(\mathsf{s}^y_z * \mathsf{s}^x_t, \varphi_0) \\
&\qquad \text{(by Theorem 1.2.21 because } y \text{ does not occur in } t) \\
&\quad = \text{FVSubst}(\mathsf{s}^y_z, \text{FVSubst}(\mathsf{s}^x_t, \varphi_0)) \\
&\qquad \text{(by Theorem 4.3.86 since } \mathsf{s}^x_t \text{ is admissible for } \varphi_0) \\
&\quad = ((\varphi_0)_{x:=t})_{y:=t}.
\end{aligned}$$

Therefore, $(\psi)_{x:=t} = (Qz)((\varphi_0)_{x:=t})_{y:=z}$. Since z does not occur free in φ_0 and does not occur in t, z does not occur free in $(\varphi_0)_{x:=t}$ because $\mathtt{FV}((\varphi_0)_{x:=t}) \subseteq \mathtt{FV}(\varphi_0) \cup \mathtt{V}(t)$ by Corollary 4.3.84. Since z is substitutable for y in φ_0 and t does not contain the variable y, z is substitutable for y in $(\varphi_0)_{x:=t}$ by Supplement 36. Thus $(\psi)_{x:=t} = (Qz)((\varphi_0)_{x:=t})_{y:=z}$ is an immediate variant of $(\varphi)_{x:=t} = (Qy)(\varphi_0)_{x:=t}$.

(81) Let $\varphi = (Qx)\alpha$ and $\psi = (Qy)(\alpha)_{x:=y}$ be two formulas, where y is a variable that does not occur free in α and y is substitutable

for x in α. Prove that if z is a variable and c is a constant symbol, then $(\psi)_{z:=c}$ is an immediate variant of $(\varphi)_{z:=c}$.

Solution: This result follows directly from Supplement 80.

(82) Prove the following generalization of Theorem 4.6.36. Given two formulas φ, ψ such that ψ is a variant of φ, and a variable x and term t such that t is substitutable for x in both φ and ψ, we can find effectively a sequence of formulas $(\theta_0, \ldots, \theta_{n-1})$ such that $\varphi = \theta_0$, $\psi = \theta_{n-1}$, θ_{i+1} is an immediate variant of θ_i, for $0 \leq i \leq n-2$ and t is substitutable for x in each θ_i.

Solution: By Theorem 4.6.35, there is a renaming $\mathfrak{v}$ of φ such that $\psi = \varphi^{\mathfrak{v}}$ and we can determine $\mathfrak{v}$ effectively. Let $\mathfrak{v} = \mathfrak{V}[\varphi, i_0 \rightarrow y_0, \ldots, i_{k-1} \rightarrow y_{k-1}]$, where $\{i_0, \ldots, i_{k-1}\}$ contains all indices of active bound occurrences in φ. We now use the proof of Theorem 4.6.34 to define a sequence of formulas $\alpha_0, \ldots, \alpha_{2k}$ with $\alpha_0 = \varphi$, $\alpha_{2k} = \psi$, and a sequence of mappings $\mathfrak{v}_1, \ldots, \mathfrak{v}_{2k}$ such that $\mathfrak{v}_i$ is a unit renaming for α_{i-1} and $\alpha_i = \alpha_{i-1}^{\mathfrak{v}_i}$, for $1 \leq i \leq 2k$ with the additional requirement that the variables $z_0, \ldots, z_{k-1}$ do not occur in t, where $z_0, \ldots, z_{k-1}$ are the variables introduced in the proof of Theorem 4.6.34. Each α_{i+1} is an immediate variant of α_i and the construction is effective.

We must show that t is substitutable for x in each α_i. Fix ℓ and suppose that (x, j) is a free occurrence of x in α_ℓ which is in the scope of another occurrence (Q, r) in α_ℓ. Then, (x, j) is a free occurrence in both φ and ψ and is in the scope of (Q, r) in both these formulas. Let $(w, r + 1)$ be the active bound occurrence following (Q, r) in α_ℓ. We consider the following cases:

`Case 1:` $r = i_s$ and $\ell \leq s$. Then, $(w, r + 1)$ occurs in φ. Since t is substitutable for x in φ and (x, j) is in the scope of (Q, r) in φ, w does not occur in t.

`Case 2:` $r = i_s$ and $s < \ell < k + s + 1$. Then, $w = z_s$ which does not occur in t.

`Case 3:` $r = i_s$ and $k + s + 1 \leq \ell$. Then, $w = y_s$. Since (x, j) is in the scope of (Q, r) and $(y_s, r + 1)$ occurs in ψ, $w = y_s$ does not occur in t.

Thus, t is substitutable for x in all the formulas α_i.

(83) Let φ and ψ be two formulas which are variants. Prove that if x is a variable and t is a term such that t is substitutable for x in both φ and ψ, then $(\psi)_{x:=t}$ is a variant of $(\varphi)_{x:=t}$.
Solution: This result follows by combining Supplements 82 and 80.

(84) Let φ and ψ be two formulas which are variants. Prove that if z is a variable and c is a constant symbol, then $(\psi)_{z:=c}$ is a variant of $(\varphi)_{z:=c}$.
Solution: Note that this is a special case of Supplement 83.

The following series of supplements extends the function $\mathsf{variant}$ such that the new function would yield variants of formulas for which a given substitution is admissible.

(85) Let S be the set of $(S_{\text{FOL}}, \text{VAR})$-substitutions. Define the function $\mathsf{variant} : \text{FORM} \times S \longrightarrow \text{FORM}$ as

(a) $\mathsf{variant}(\varphi, s)$ is φ if φ is atomic;
(b) $\mathsf{variant}(\varphi, s)$ is $(\neg\mathsf{variant}(\psi, s))$ if $\varphi = (\neg\psi)$;
(c) $\mathsf{variant}(\varphi, s)$ is $(\mathsf{variant}(\alpha, s)\ C\ \mathsf{variant}(\beta, s))$ if $\varphi = (\alpha C \beta)$ for C a binary connective symbol;
(d) $\mathsf{variant}(\varphi, s)$ is $(Qy)\mathsf{variant}(\psi, [y \to y]s)$ if $\varphi = (Qy)\psi$, and there is no $w \in \mathtt{FV}(\varphi)$ such that y occurs in $s(w)$;
(e) $\mathsf{variant}(\varphi, s)$ is $(Qz)(\mathsf{variant}(\psi, [y \to y]s))_{y:=z}$ if $\varphi = (Qy)\psi$, there is a $w \in \mathtt{FV}(\varphi)$ such that y occurs in $s(w)$, and z is the first variable not occurring in $\mathsf{variant}(\psi, [y \to y]s)$ or in a term $s(w)$ with $w \in \mathtt{FV}(\varphi)$.

(a) Prove that $\mathsf{variant}(\varphi, s)$ is a variant of φ in the sense of Definition 4.6.19.
(b) Prove that the substitution s is admissible for $\mathsf{variant}(\varphi, s)$.
(c) Show that if s is admissible for φ, then $\mathsf{variant}(\varphi, s) = \varphi$.

Solution: Part (a) is a straightforward consequence of the definition of variant.
The proof for Part (b) is by induction on the formula φ. We discuss here the more difficult inductive step when $\varphi = (Qy)\psi$ and the result is assumed to hold for ψ.
Suppose initially that there is no $w \in \mathtt{FV}(\varphi)$ such that y occurs in $s(w)$. In this case, $\mathsf{variant}(\varphi, s) = (Qy)\mathsf{variant}(\psi, [y \to y]s)$ and by the inductive hypothesis, $[y \to y]s$ is admissible

for $\mathsf{variant}(\psi, [y \to y]s)$, which implies admissibility of s for $\mathsf{variant}(\varphi, s)$.

Consider now the case when there is $w \in \mathtt{FV}(\varphi)$ such that y occurs in $s(w)$. Now, $\mathsf{variant}(\varphi, s)$ is $(Qz)(\mathsf{variant}(\psi, [y \to y]s))_{y:=z}$ and we have the admissibility of $[y \to y]s$ for $\mathsf{variant}(\psi, [y \to y]s)$ by inductive hypothesis. By Supplement 79, $[z \to z]s$ is admissible for $(\mathsf{variant}(\psi, [y \to y]s))_{y:=z}$. Thus, it remains to show that if $w \in \mathtt{FV}(\mathsf{variant}(\varphi, s))$, then z does not occur in $s(w)$. So, let $w \in \mathtt{FV}(\mathsf{variant}(\varphi, s))$. This implies $w \in \mathtt{FV}((\mathsf{variant}(\psi, [y \to y]s))_{y:=z}) - \{z\}$. By Corollary 4.3.84, $w \in \mathtt{FV}(\mathsf{variant}(\psi, [y \to y]s)) - \{y, z\}$, which implies $w \in \mathtt{FV}(\psi) - \{y, z\}$ by Theorem 4.6.22 and the first part of the current supplement. Thus, $w \in \mathtt{FV}(\varphi)$, so z does not occur in $s(w)$ by choice of z.

The last part can be shown by induction on φ.

(86) Let φ be a formula, x be a variable, and t be a term. Prove that

$$\mathsf{variant}(\varphi, \mathsf{s}_t^x) = \mathsf{variant}(\varphi, x, t),$$

where $\mathsf{variant}(\varphi, x, t)$ is as given in Definition 4.6.43.

Solution: The proof is by induction on φ. We consider the only nontrivial step, namely when $\varphi = (Qy)\psi$ and the statement is true for ψ.

`Case 1:` $y = x$. Then, x does not occur free in φ, so $\mathsf{variant}(\varphi, x, t) = \varphi$. Also, if $y = x$ occurs in $\mathsf{s}_t^x(w)$ for some variable w, then we must have $w = x$, but $x \notin \mathtt{FV}(\varphi)$, so, noting that the identity substitution ι is admissible for any formula, we have

$$\begin{aligned}
\mathsf{variant}(\varphi, \mathsf{s}_t^x) &= \mathsf{variant}((Qy)\psi, \mathsf{s}_t^x) \\
&= (Qy)\mathsf{variant}(\psi, [y \to y]\mathsf{s}_t^x) \\
&= (Qy)\mathsf{variant}(\psi, \iota) \\
&\quad \text{(since } y = x\text{)} \\
&= (Qy)\psi \\
&\quad \text{(by admissibility of } \iota \text{ and Part (c)}
\end{aligned}$$

$$\begin{aligned}
&\quad \text{of Supplement 85)} \\
&= \varphi \\
&= \mathsf{variant}(\varphi, x, t).
\end{aligned}$$

`Case 2:` $y \neq x$ and $x \notin \texttt{FV}(\varphi)$. Then, $\mathsf{variant}(\varphi, x, t) = \varphi$ and $x \notin \texttt{FV}(\psi)$. Also, if y occurs in $\mathsf{s}^x_t(w)$ for some variable w, then we must have $w \in \{x, y\}$, so $w \notin \texttt{FV}(\varphi)$. Thus, we have

$$\begin{aligned}
\mathsf{variant}(\varphi, \mathsf{s}^x_t) &= \mathsf{variant}((Qy)\psi, \mathsf{s}^x_t) \\
&= (Qy)\mathsf{variant}(\psi, [y \to y]\mathsf{s}^x_t) \\
&= (Qy)\mathsf{variant}(\psi, \mathsf{s}^x_t) \\
&\quad \text{(since } y \neq x\text{)} \\
&= (Qy)\psi \\
&= \varphi \\
&= \mathsf{variant}(\varphi, x, t),
\end{aligned}$$

where in the fourth equality we have used Part (c) of Supplement 85 and the fact that s^x_t is admissible for ψ since $x \notin \texttt{FV}(\psi)$.

`Case 3:` $x \in \texttt{FV}(\varphi)$ and y does not occur in t. Then $x \neq y$ and $\mathsf{variant}(\varphi, x, t) = (Qy)\mathsf{variant}(\psi, x, t)$. If y occurs in $\mathsf{s}^x_t(w)$, then $w = y$ (since y does not occur in t) and $y \notin \texttt{FV}(\varphi)$. Thus, we have

$$\begin{aligned}
\mathsf{variant}(\varphi, \mathsf{s}^x_t) &= \mathsf{variant}((Qy)\psi, \mathsf{s}^x_t) \\
&= (Qy)\mathsf{variant}(\psi, [y \to y]\mathsf{s}^x_t) \\
&= (Qy)\mathsf{variant}(\psi, \mathsf{s}^x_t) \\
&\quad \text{(since } y \neq x\text{)} \\
&= (Qy)\mathsf{variant}(\psi, x, t) \\
&\quad \text{(by inductive hypoothesis)} \\
&= \mathsf{variant}(\varphi, x, t).
\end{aligned}$$

`Case 4:` $x \in \texttt{FV}(\varphi)$ and y occurs in t. Then, $x \neq y$ and also

$$\mathsf{variant}(\varphi, x, t) = (Qz)(\mathsf{variant}(\psi, x, t))_{y:=z},$$

where z is the first variable that does not occur in $\mathsf{variant}(\psi, x, t)$ or in t.
On the other hand, since $x \in \mathtt{FV}(\varphi)$ and y occurs in $t = \mathsf{s}_t^x(x)$, we have

$$\begin{aligned}
\mathsf{variant}(\varphi, \mathsf{s}_t^x) &= \mathsf{variant}((Qy)\psi, \mathsf{s}_t^x) \\
&= (Qz')(\mathsf{variant}(\psi, [y \to y]\mathsf{s}_t^x))_{y:=z'} \\
&= (Qz')(\mathsf{variant}(\psi, \mathsf{s}_t^x))_{y:=z'} \\
&\quad \text{(since } y \neq x\text{)} \\
&= (Qz')(\mathsf{variant}(\psi, x, t))_{y:=z'} \\
&\quad \text{(by inductive hypothesis),}
\end{aligned}$$

where z' is the first variable that does not occur in $\mathsf{variant}(\psi, [y \to y]\mathsf{s}_t^x) = \mathsf{variant}(\psi, x, t)$ or in a term $\mathsf{s}_t^x(w)$ with $w \in \mathtt{FV}(\varphi)$.
It suffices to show that $z' = z$. If z'' precedes z in the standard ordering, then by definition of z, either z'' occurs in $\mathsf{variant}(\psi, x, t)$ or z'' occurs in $t = \mathsf{s}_t^x(x)$ with $x \in \mathtt{FV}(\varphi)$, so z'' cannot equal z' and hence z' cannot precede z in the standard ordering. To show that $z' = z$, it thus suffices to show that z satisfies the defining condition of z'. We have that z does not appear in $\mathsf{variant}(\psi, x, t)$ by definition of z. Suppose that $w \in \mathtt{FV}(\varphi)$. If $w = x$, then z does not occur in $\mathsf{s}_t^x(w) = t$ by definition of z. If $w \neq x$, then, since $w \in \mathtt{FV}(\psi) = \mathtt{FV}(\mathsf{variant}(\psi, x, t))$, w occurs in $\mathsf{variant}(\psi, x, t)$, so by definition of z, $z \neq w = \mathsf{s}_t^x(w)$. Thus, $z' = z$, as desired.

(87) Let φ be a formula and let s be a substitution. Define the formula $\langle\varphi\rangle_s$ as $\mathrm{FVSubst}(s, \mathsf{variant}(\varphi, s))$.
Prove that if $\mathcal{A}$ is an $\mathcal{L}$-structure and s is an $\mathcal{L}$-substitution, then $(\mathcal{A}, \sigma) \models \langle\varphi\rangle_s$ if and only if $(\mathcal{A}, \sigma^{\mathcal{A}} \circ s) \models \varphi$.
Solution: Note that the following statements are equivalent:

$$\begin{aligned}
&(\mathcal{A}, \sigma) \models \langle\varphi\rangle_s \\
&(\mathcal{A}, \sigma) \models \mathrm{FVSubst}(s, \mathsf{variant}(\varphi, s)) \\
&\qquad \text{by definition of } \langle\varphi\rangle_s
\end{aligned}$$

$(\mathcal{A}, \sigma^{\mathcal{A}} \circ s) \models \mathsf{variant}(\varphi, s)$

by Corollary 4.6.5 since s is admissible for $\mathsf{variant}(\varphi, s)$

$(\mathcal{A}, \sigma^{\mathcal{A}} \circ s) \models \varphi$

since $\mathsf{variant}(\varphi, s) \equiv \varphi$

(88) Let φ, ψ be two first-order formulas, and let s be a substitution. Prove that $\langle(\neg\varphi)\rangle_s = (\neg\langle\varphi\rangle_s)$ and $\langle(\varphi C \psi)\rangle_s = (\langle\varphi\rangle_s C \langle\psi\rangle_s)$.
Hint. The argument parallels the argument of Theorem 4.6.50.

Define $\langle\varphi\rangle_{y_0,\ldots,y_{n-1}:=t_0,\ldots,t_{n-1}}$ to be $\langle\varphi\rangle_{\mathsf{s}_{t_0\cdots t_{n-1}}^{y_0\cdots y_{n-1}}}$, where $y_0, \ldots, y_{n-1}$ are distinct variables.

(89) If $n = 1$, then the notation just defined becomes $\langle\varphi\rangle_{y_0:=t_0}$, a notation that we have already defined just after Example 4.6.44. Use Supplement 86 to show that these two definitions of the notation agree with each other.

(90) Let $\mathcal{A}$ be an $\mathcal{L}$-structure, $\sigma \in \text{ASSIGN}_{\mathcal{A}}$, φ be an $\mathcal{L}$-formula, and let $y_0, \ldots, y_{n-1}$ be distinct variables. Prove that if $t_0, \ldots, t_{n-1}$ are $\mathcal{L}$-terms, then

$$(\mathcal{A}, \sigma) \models \langle\varphi\rangle_{y_0,\ldots,y_{n-1}:=t_0,\ldots,t_{n-1}}$$

if and only if

$$(\mathcal{A}, [y_0 \to \sigma^{\mathcal{A}}(t_0)] \cdots [y_{n-1} \to \sigma^{\mathcal{A}}(t_{n-1})]\sigma) \models \varphi.$$

Hint. This follows directly from Supplement 87.

(91) Prove the following two logical implications, which generalize Theorem 4.6.51:

$$(\forall y_0) \cdots (\forall y_{n-1})\varphi \models \langle\varphi\rangle_{y_0,\ldots,y_{n-1}:=t_0,\ldots,t_{n-1}}$$

and

$$\langle\varphi\rangle_{y_0,\ldots,y_{n-1}:=t_0,\ldots,t_{n-1}} \models (\exists y_0) \cdots (\exists y_{n-1})\varphi.$$

Solution: Let $\mathcal{L}$ be a first-order language such that φ is an $\mathcal{L}$-formula and $t_0, \ldots, t_{n-1}$ are $\mathcal{L}$-terms. If $(\mathcal{A}, \sigma) \models (\forall y_0) \cdots (\forall y_{n-1})\varphi$ for some $\mathcal{L}$-structure $\mathcal{A}$ and $\sigma \in \text{ASSIGN}_{\mathcal{A}}$, then $(\mathcal{A}, [y_{n-1} \to a_{n-1}] \cdots [y_0 \to a_0]\sigma) \models \varphi$ for every $a_0, \ldots, a_{n-1} \in |\mathcal{A}|$. In particular,

$$(\mathcal{A}, [y_{n-1} \to \sigma^{\mathcal{A}}(t_{n-1})] \cdots [y_0 \to \sigma^{\mathcal{A}}(t_0)]\sigma) \models \varphi,$$

which is equivalent to

$$(\mathcal{A}, \sigma) \models \langle\varphi\rangle_{y_0,\ldots,y_{n-1}:=t_0,\ldots,t_{n-1}},$$

by Exercise 90, since the modifications to the assignment σ can be done in any order due to the fact that the variables $y_0, \ldots, y_{n-1}$ are distinct.
We leave the second part to the reader.

(92) Let $\mathcal{L}$ be a first-order language, φ be an $\mathcal{L}$-formula, t be an $\mathcal{L}$-term, x be a variable and φ' be a variant of φ such that t is substitutable for x in φ'. Prove that if $\mathcal{A}$ is an $\mathcal{L}$-structure and $\sigma \in \mathrm{ASSIGN}_{\mathcal{A}}$, then $(\mathcal{A}, \sigma) \models (\varphi')_{x:=t}$ if and only if $(\mathcal{A}, [x \to \sigma^{\mathcal{A}}(t)]\sigma) \models \varphi$.
Solution: We have the following equivalent statements:

$(\mathcal{A}, \sigma) \models (\varphi')_{x:=t}$
$(\mathcal{A}, [x \to \sigma^{\mathcal{A}}(t)]\sigma) \models \varphi'$
(by the Substitution Corollary)
$(\mathcal{A}, [x \to \sigma^{\mathcal{A}}(t)]\sigma) \models \varphi$
(because two variants are logically equivalent).

(93) Prove that if φ is a formula, x is a variable, t is a term and φ' is a variant of φ such that t is substitutable for x in φ', then $(\forall x)\varphi \models (\varphi')_{x:=t}$ and $(\varphi')_{x:=t} \models (\exists x)\varphi$.
Solution: Let $\mathcal{L}$ be a first-order language such that φ is an $\mathcal{L}$-formula and t is an $\mathcal{L}$-term. If $(\mathcal{A}, \sigma) \models (\forall x)\varphi$ for some $\mathcal{L}$-structure $\mathcal{A}$ and $\sigma \in \mathrm{ASSIGN}_{\mathcal{A}}$, then $(\mathcal{A}, [x \to a]\sigma) \models \varphi$ for every $a \in |\mathcal{A}|$. In particular, $(\mathcal{A}, [x \to \sigma^{\mathcal{A}}(t)]\sigma) \models \varphi$, so, by Supplement 92, $(\mathcal{A}, \sigma) \models (\varphi')_{x:=t}$.
For the second part, suppose that $(\mathcal{A}, \sigma) \models (\varphi')_{x:=t}$. By Supplement 92, we have $(\mathcal{A}, [x \to \sigma^{\mathcal{A}}(t)]) \models \varphi$. This, in turn, gives $(\mathcal{A}, \sigma) \models (\exists x)\varphi$, which concludes our argument.

Definability in Structures

(94) Let $\mathcal{L}$ be a first-order language and $\mathcal{A}$ be an $\mathcal{L}$-structure. Prove that both 0-ary relations on $|\mathcal{A}|$ are definable in $\mathcal{A}$.

(95) Prove that if $\mathcal{A}$ is a structure such that $|\mathcal{A}|$ is uncountable, then there is $a \in |\mathcal{A}|$ such that $\{a\}$ is not definable.

(96) Let $\mathcal{A} = (A, \mathcal{I})$ be an $\mathcal{L}$-structure. Prove that if the n-ary relation $\rho \subseteq |\mathcal{A}|^n$ is definable in $\mathcal{A}$, then for every automorphism h of $\mathcal{A}$ and $(a_0, \ldots, a_{n-1}) \in A^n$, $(a_0, \ldots, a_{n-1}) \in \rho$ if and only if $(h(a_0), \ldots, h(a_{n-1})) \in \rho$.

Solution: Suppose that ρ is definable in $\mathcal{A}$ by the formula φ and the sequence of variables $(y_0, \ldots, y_{n-1})$. The following statements are easily seen to be equivalent for every n-tuple $(a_0, \ldots, a_{n-1}) \in A^n$ and automorphism h:

(i) $(a_0, \ldots, a_{n-1}) \in \rho$;
(ii) $(\mathcal{A}, \sigma) \models \varphi$ for every $\sigma \in \text{ASSIGN}_{\mathcal{A}}$ such that $\sigma(y_i) = a_i$ for $0 \leq i \leq n-1$;
(iii) $(\mathcal{A}, h \circ \sigma) \models \varphi$ for every $\sigma \in \text{ASSIGN}_{\mathcal{A}}$ such that $\sigma(y_i) = a_i$ for $0 \leq i \leq n-1$;
(iv) $(\mathcal{A}, \sigma') \models \varphi$ for every $\sigma' \in \text{ASSIGN}_{\mathcal{A}}$ such that $\sigma'(y_i) = h(a_i)$ for $0 \leq i \leq n-1$;
(v) $(h(a_0), \ldots, h(a_{n-1})) \in \rho$.

The equivalence between (ii) and (iii) follows from the Morphism Theorem; the remaining equivalences follow directly from the relevant definitions.

(97) Let $\mathcal{L} = \{R_1^2\}$ be a first-order language that consists of a binary relation symbol. Define the $\mathcal{L}$-structure $\mathcal{R} = (\mathbf{R}, \mathcal{I})$, where $\mathcal{I}(R_1^2) = \{(x, y) \in \mathbf{R}^2 \mid x < y\}$. Prove that the set $\mathbf{N}$ is not definable in $\mathcal{R}$.

Hint. Consider the automorphism $h : \mathbf{R} \longrightarrow \mathbf{R}$ of $\mathcal{R}$ defined by $h(x) = x^3$ for $x \in \mathbf{R}$ and observe that there exists $x \in \mathbf{R}$ such that $h(x) \in \mathbf{N}$ but $x \notin \mathbf{N}$.

(98) Let $\rho \subseteq A^2$ be a binary relation definable in an $\mathcal{L}$-structure $\mathcal{A}$, where $\mathcal{L}$ is a first-order language. Prove that the sets $\text{Dom}(\rho) = \{a \in A \mid (a, b) \in \rho \text{ for some } b \in A\}$ and $\text{Ran}(\rho) = \{b \in A \mid (a, b) \in \rho \text{ for some } a \in A\}$ are definable in $\mathcal{A}$.

(99) Let $\mathcal{L} = \{f_0^2, =\}$ be a first-order language and let $\mathcal{A} = (\mathbf{N}, \mathcal{I})$, where $\mathcal{I}(f_0^2)(m, n) = mn$ for $m, n \in \mathbf{N}$.

(a) Prove that for $i = 0, 1$, the constant mappings $h_i : \mathbf{N} \longrightarrow \mathbf{N}$ given by $h_i(n) = i$ for $n \in \mathbf{N}$ are endomorphisms of $\mathcal{A}$.

(b) Prove that if h is a endomorphism of $\mathcal{A}$ different from h_1, then $h(0) = 0$; also, if h is different from h_0, then $h(1) = 1$.

(c) Let $\mathbf{PR} = \{p_0, \ldots, p_n, \ldots\}$ be the set of primes, where $p_0 < p_1 < \cdots < p_n < \cdots$ and let $f : \mathbf{PR} \longrightarrow \mathbf{N}$ be a function. Prove that the mapping $h_f : \mathbf{N} \longrightarrow \mathbf{N}$ given by

$$h_f(n) = \begin{cases} 0 & \text{if } n = 0, \\ \prod_{i=0}^{k-1} (f(p_i))^{a_i} & \text{if } n > 0 \text{ and } n = \prod_{i=0}^{k-1} p_i^{a_i} \text{ is} \\ & \text{an expression of } n \text{ as a product} \\ & \text{of primes} \end{cases}$$

is well-defined. Further, show that h_f is an endomorphism of $\mathcal{A}$ and that the set of endomorphisms of $\mathcal{A}$ consists of h_0, h_1 and the functions h_f where $f : \mathbf{PR} \longrightarrow \mathbf{N}$.

(d) Let $h : \mathbf{N} \longrightarrow \mathbf{N}$ be an automorphism of $\mathcal{A}$. Prove that if $m \geq 2$, then $h(m) \geq 2$.

(e) Prove that if $h : \mathbf{N} \longrightarrow \mathbf{N}$ is an automorphism of $\mathcal{A}$, then p is a prime number if and only if $h(p)$ is a prime number.

(f) Prove that the set of automorphisms of $\mathcal{A}$ is the set of mappings of the form h_f, where f is a bijection of the set $\mathbf{PR}$.

(g) Prove that the binary relation $\sigma = \{(m, n) \in \mathbf{N}^2 \mid m < n\}$ is not definable in $\mathcal{A}$.

(h) Prove that the ternary relation

$$\rho = \{(m, n, p) \mid m, n, p \in \mathbf{N} \text{ and } m + n = p\}$$

is not definable in $\mathcal{A}$.

(100) Let $\mathcal{L}$ be a first-order language, $\mathcal{A}$ be an $\mathcal{L}$-structure, and let ρ be a relation definable in $\mathcal{A}$ by the formula φ_ρ and the variables $(y_0, \ldots, y_{n-1})$. Define $\mathcal{L}'$ as the first-order language $\mathcal{L} \cup \{R\}$, where R is a new n-ary relation symbol and let $\mathcal{A}'$ be the $\mathcal{L}'$-structure that is the extension of $\mathcal{A}$ with $R^{\mathcal{A}'} = \rho$. Prove that there is an effectively computable function F : FORM $\longrightarrow$ FORM such that for all $\varphi \in \text{FORM}_{\mathcal{L}'}$ we have $F(\varphi) \in \text{FORM}_{\mathcal{L}}$ and, if $\sigma \in \text{ASSIGN}_{\mathcal{A}}$, then $(\mathcal{A}', \sigma) \models \varphi$ if and only if $(\mathcal{A}, \sigma) \models F(\varphi)$.

Solution: The function F is defined recursively. For atomic formulas, F is defined as follows. If P is an m-ary relation symbol distinct from R, then if $m > 0$,

$$F(P(t_0, \ldots, t_{m-1})) = P(t_0, \ldots, t_{m-1})$$

for all $t_0, \ldots, t_{m-1} \in \text{TERM}$, and if $m = 0$, $F(P) = P$.
If $n > 0$, define $F(R(t_0, \ldots, t_{n-1})) = \langle \varphi_\rho \rangle_{y_0,\ldots,y_{n-1}:=t_0,\ldots,t_{n-1}}$ for all $t_0, \ldots, t_{m-1} \in \text{TERM}$ and if $n = 0$, we define

$$F(R) = \langle \varphi_\rho \rangle_{y_0,\ldots,y_{n-1}:=t_0,\ldots,t_{n-1}}.$$

Note that when $n = 0$, the last expression reduces to φ_ρ.
The recursive step is given by the following formulas:

$$F((\neg\varphi)) = (\neg F(\varphi))$$
$$F((\varphi C \psi)) = (F(\varphi) C F(\psi))$$
$$F((Qx)\varphi) = (Qx)F(\varphi)$$

for all formulas φ, ψ, where C is a binary connective symbol and Q is a quantifier symbol.
We discuss here only the difficult basis substep, when $P = R$. The following statements are equivalent:

$(\mathcal{A}, \sigma) \models \langle \varphi_\rho \rangle_{y_0,\ldots,y_{n-1}:=t_0,\ldots,t_{n-1}}$
$(\mathcal{A}, \sigma^{\mathcal{A}} \circ \mathsf{s}^{y_0 \cdots y_{n-1}}_{t_0 \cdots t_{n-1}}) \models \varphi_\rho$
(by Supplement 87)
$(\mathcal{A}, [y_0 \to \sigma^{\mathcal{A}}(t_0)] \cdots [y_{n-1} \to \sigma^{\mathcal{A}}(t_{n-1})]\sigma) \models \varphi_\rho$
$(\mathcal{A}, [y_0 \to \sigma^{\mathcal{A}}(t_0) \cdots y_{n-1} \to \sigma^{\mathcal{A}}(t_{n-1})]) \models \varphi_\rho$
(because $\texttt{FV}(\varphi_\rho) \subseteq \{y_0, \ldots, y_{n-1}\}$)
$(\sigma^{\mathcal{A}}(t_0), \ldots, \sigma^{\mathcal{A}}(t_{n-1})) \in \rho$
$(\sigma^{\mathcal{A}'}(t_0), \ldots, \sigma^{\mathcal{A}'}(t_{n-1})) \in R^{\mathcal{A}'}$
(because $\sigma^{\mathcal{A}}(t_i) = \sigma^{\mathcal{A}'}(t_i)$)
$(\mathcal{A}', \sigma) \models R(t_0, \ldots, t_{n-1})$ when $n > 0$ and $(\mathcal{A}', \sigma) \models R$ when $n = 0$.

(101) Let $\mathcal{L}$ be a first-order language, $\mathcal{A}$ be an $\mathcal{L}$-structure and let $g : |\mathcal{A}|^n \longrightarrow |\mathcal{A}|$ be a function definable in $\mathcal{A}$ by the formula

φ_g and the sequence of variables $(y_0, \ldots, y_n)$. Define $\mathcal{L}'$ as the first-order language $\mathcal{L} \cup \{f_g\}$, where f_g is a new n-ary function symbol and let $\mathcal{A}'$ be the $\mathcal{L}'$-structure that is the extension of $\mathcal{A}$ with $f_g^{\mathcal{A}'} = g$. Prove that there is an effectively computable function $F : \mathrm{FORM} \longrightarrow \mathrm{FORM}$ such that for all $\varphi \in \mathrm{FORM}_{\mathcal{L}'}$ we have $F(\varphi) \in \mathrm{FORM}_{\mathcal{L}}$ and, if $\sigma \in \mathrm{ASSIGN}_{\mathcal{A}}$, then $(\mathcal{A}', \sigma) \models \varphi$ if and only if $(\mathcal{A}, \sigma) \models F(\varphi)$.

Solution: We begin by defining F for atomic formulas using recursion on the number of occurrences of f_g in the atomic formula φ. If f_g does not occur in φ, then $F(\varphi) = \varphi$.

Suppose now f_g occurs in φ, say $\varphi = R(t_0, \ldots, t_{m-1})$ and that the rightmost occurrence of the function symbol f_g is located in the term t_k, and let $(f_g(u_0, \ldots, u_{n-1}), i)$ be the occurrence of the term beginning with this rightmost occurrence of f_g in the term t_k. Note that f_g does not occur in any of the terms $u_0, \ldots, u_{n-1}$.

Let z be a variable that does not occur in $R(t_0, \ldots, t_{m-1})$. Observe that the atomic formula

$$\theta = R(t_0, \ldots, t_{k-1}, \mathtt{replace}\,(t_k, (f_g(u_0, \ldots, u_{n-1}), i), z), \\ t_{k+1}, \ldots, t_{m-1})$$

contains one fewer occurrence of f_g than $R(t_0, \ldots, t_{m-1})$. Thus, we can complete the recursive definition of F for atomic formulas by:

$$F(R(t_0, \ldots, t_{m-1})) = (\exists z)(\langle \varphi_g \rangle_{y_0, \ldots, y_n := u_0, \ldots, u_{n-1}, z} \wedge F(\theta)).$$

Before defining F for the full set of formulas, we want to show that $F(\varphi)$ is $\mathcal{A}'$-equivalent to φ for any $\mathcal{L}'$-atomic formula φ. For the basis step, when φ contains no occurrences of f_g, this fact is obvious. Suppose that the statement holds for formulas containing fewer than l occurrences of f_g, where $l > 0$, and let $R(t_0, \ldots, t_{m-1})$ be an $\mathcal{L}'$-formula that contains l occurrences of f_g. Let t_k and $u_0, \ldots, u_{n-1}$ be as in the definition of F.

Suppose $(\mathcal{A}', \sigma) \models R(t_0, \ldots, t_{m-1})$, i.e., $(\sigma^{\mathcal{A}'}(t_0), \ldots, \sigma^{\mathcal{A}'}(t_{m-1})) \in R^{\mathcal{A}'}$ and let $a = g(\sigma^{\mathcal{A}'}(u_0), \ldots, \sigma^{\mathcal{A}'}(u_{n-1})) = \sigma^{\mathcal{A}'}(f_g(u_0, \ldots, u_{n-1}))$. Since g is defined by φ_g and $y_0, \ldots, y_n$

in $\mathcal{A}'$, the definition of a implies

$$(\mathcal{A}', [y_0 \to \sigma^{\mathcal{A}'}(u_0)] \cdots [y_{n-1} \to \sigma^{\mathcal{A}'}(u_{n-1})][y_n \to a]) \models \varphi_g$$

which can be written as

$$(\mathcal{A}', [y_0 \to \sigma'^{\mathcal{A}'}(u_0)] \cdots [y_{n-1} \to \sigma'^{\mathcal{A}'}(u_{n-1})]$$
$$[y_n \to \sigma'^{\mathcal{A}'}(z)]\sigma') \models \varphi_g,$$

where $\sigma' = [z \to a]\sigma$, due to the fact that z does not occur in any of the terms $u_0, \ldots, u_{n-1}$. This is equivalent to

$$(\mathcal{A}', \sigma') \models \langle \varphi_g \rangle_{y_0,\ldots,y_n := u_0,\ldots,u_{n-1},z},$$

due to Exercise 90. If $j \neq k$, then $\sigma^{\mathcal{A}'}(t_j) = \sigma'^{\mathcal{A}'}(t_j)$ because z does not occur in t_j. Also, by Exercise 49,

$$\sigma^{\mathcal{A}'}(t_k) = \sigma'^{\mathcal{A}'}(\texttt{replace}\,(t_k, (f_g(u_0, \ldots, u_{n-1}), i), z)).$$

It follows that

$$(\mathcal{A}', \sigma') \models R(t_0, \ldots, t_{k-1}, \texttt{replace}\,(t_k, (f_g(u_0, \ldots, u_{n-1}), i), z),$$
$$t_{k+1}, \ldots, t_{m-1}) = \theta.$$

By the inductive hypothesis, we have $(\mathcal{A}', \sigma') \models F(\theta)$, so $(\mathcal{A}', \sigma') \models (\langle \varphi_g \rangle_{y_0,\ldots,y_n := u_0,\ldots,u_{n-1},z} \wedge F(\theta))$, which implies

$$(\mathcal{A}', \sigma) \models F(R(t_0, \ldots, t_{m-1})).$$

Suppose now that $(\mathcal{A}', \sigma) \models F(R(t_0, \ldots, t_{m-1}))$. By the definition of F, there is an $a \in |\mathcal{A}'|$ such that $(\mathcal{A}', \sigma') \models (\langle \varphi_g \rangle_{y_0,\ldots,y_n := u_0,\ldots,u_{n-1},z} \wedge F(\theta))$, where $\sigma' = [z \to a]\sigma$ and

$$\theta = R(t_0, \ldots, t_{k-1}, \texttt{replace}\,(t_k, (f_g(u_0, \ldots, u_{n-1}), i), z),$$
$$t_{k+1}, \ldots, t_{m-1}).$$

By the inductive hypothesis, $(\mathcal{A}', \sigma') \models F(\theta)$ implies that $(\mathcal{A}', \sigma') \models \theta$, which means that

$$(\sigma'^{\mathcal{A}'}(t_0), \ldots, \sigma'^{\mathcal{A}'}(t_{k-1}), \sigma'^{\mathcal{A}'}(\texttt{replace}\,(t_k, (f_g(u_0, \ldots, u_{n-1}), i), z)),$$
$$\sigma'^{\mathcal{A}'}(t_{k+1}), \ldots, \sigma'^{\mathcal{A}'}(t_{m-1})) \in R^{\mathcal{A}'}.$$

Since z does not occur in any t_i, we obtain

$$(\sigma^{\mathcal{A}'}(t_0), \ldots, \sigma^{\mathcal{A}'}(t_{k-1}), \sigma'^{\mathcal{A}'}(\texttt{replace}\,(t_k, (f_g(u_0, \ldots, u_{n-1}), i), z)),$$
$$\sigma^{\mathcal{A}'}(t_{k+1}), \ldots, \sigma^{\mathcal{A}'}(t_{m-1})) \in R^{\mathcal{A}'}. \tag{4.23}$$

From $(\mathcal{A}', \sigma') \models \langle \varphi_g \rangle_{y_0,\ldots,y_n := u_0,\ldots,u_{n-1},z}$, it follows by Exercise 90 that

$$(\mathcal{A}', [y_0 \to \sigma'^{\mathcal{A}'}(u_0)] \cdots [y_{n-1}$$
$$\to \sigma'^{\mathcal{A}'}(u_{n-1})][y_n \to \sigma'^{\mathcal{A}'}(z)]\sigma') \models \varphi_g.$$

Since z does not occur in any of the terms $u_0, \ldots, u_{n-1}$, we have

$$(\mathcal{A}', [y_0 \to \sigma^{\mathcal{A}'}(u_0)] \cdots [y_{n-1} \to \sigma^{\mathcal{A}'}(u_{n-1})][y_n \to a]) \models \varphi_g,$$

so $a = g(\sigma^{\mathcal{A}'}(u_0), \ldots, \sigma^{\mathcal{A}'}(u_{n-1})) = \sigma^{\mathcal{A}'}(f_g(u_0, \ldots, u_{n-1}))$. Since, by Exercise 49, we have

$$\sigma^{\mathcal{A}'}(t_k) = \sigma'^{\mathcal{A}'}(\texttt{replace}\,(t_k, (f_g(u_0, \ldots, u_{n-1}), i), z)),$$

it follows from (4.23) that $(\sigma^{\mathcal{A}'}(t_0), \ldots, \sigma^{\mathcal{A}'}(t_{m-1})) \in R^{\mathcal{A}'}$ which allows us to conclude that $(\mathcal{A}', \sigma) \models R(t_0, \ldots, t_{m-1})$. For the nonatomic case, we define recursively F by

$$F((\neg\varphi)) = (\neg F(\varphi))$$
$$F((\varphi C \psi)) = (F(\varphi) C F(\psi))$$
$$F((Qx)\varphi) = (Qx)F(\varphi),$$

for any binary connective symbol C and quantifier symbol Q. We leave the argument for these cases to the reader.

Propositional Forms and Tautologies

(102) Give an example of an injective inter-substitution s such that the extension of s to PLFORM is not an injective function.

(103) Prove that if s is an injective prime inter-substitution and $\Gamma \subseteq$ PLFORM, then $s(\Gamma)$ is a closed set of first-order formulas if and only if Γ is closed.

(104) Let s be an inter-substitution and let $\varphi_0, \ldots, \varphi_{n-1}$ be propositional formulas. Prove that

$$s(\varphi_0 \vee \cdots \vee \varphi_{n-1}) = (s(\varphi_0) \vee \cdots \vee s(\varphi_{n-1}))$$
$$s(\varphi_0 \wedge \cdots \wedge \varphi_{n-1}) = (s(\varphi_0) \wedge \cdots \wedge s(\varphi_{n-1}).$$

(105) Let $n, m_0, \ldots, m_{n-1}$ be $n+1$ positive natural numbers and let $\{\varphi_{ij} \mid 0 \leq i \leq n-1, 0 \leq j \leq m_i - 1\}$ be a family of first-order formulas. Prove that

$$\bigvee_{0\leq i\leq n-1} \bigvee_{0\leq j\leq m_i-1} \varphi_{ij}$$
$$\equiv (\varphi_{00} \vee \cdots \vee \varphi_{0\, m_0-1} \vee \cdots \vee \varphi_{n-1\, 0} \vee \cdots \vee \varphi_{n-1\, m_{n-1}-1})$$
$$\bigwedge_{0\leq i\leq n-1} \bigwedge_{0\leq j\leq m_i-1} \varphi_{ij}$$
$$\equiv (\varphi_{00} \wedge \cdots \wedge \varphi_{0\, m_0-1} \wedge \cdots \wedge \varphi_{n-1\, 0} \wedge \cdots \wedge \varphi_{n-1\, m_{n-1}-1}).$$

Hint. Use Exercise 22 of Chapter 2.

(106) Let n be a positive natural number, $m_0, \ldots, m_{n-1}$ be n positive natural numbers, $\mathbf{m} = (m_0, \ldots, m_{n-1})$, and $M = m_0 \cdot \cdots \cdot m_{n-1}$. We use here the notations introduced before Supplement 23 of Chapter 2. For $0 \leq i \leq n-1$, let $\varphi_{i\,0}, \ldots, \varphi_{i\, m_i-1}$ be m_i first-order formulas. Prove that

$$\bigvee_{i=0}^{n-1} \bigwedge_{j=0}^{m_i-1} \varphi_{ij} \equiv \bigwedge_{k=0}^{M-1} \bigvee_{i=0}^{n-1} \varphi_{i\, s_k(i)},$$
$$\bigwedge_{i=0}^{n-1} \bigvee_{j=0}^{m_i-1} \varphi_{ij} \equiv \bigvee_{k=0}^{M-1} \bigwedge_{i=0}^{n-1} \varphi_{i\, s_k(i)}.$$

Hint. Use Supplement 23 of Chapter 2.

Inter-substitutions can be applied to signed formulas in a natural way; namely, if s is an inter-substitution and $b\varphi$ is a signed formula of propositional logic, we define $s(b\varphi)$ as $bs(\varphi)$.

(107) Formulate and prove a statement similar to the one in Exercise 103 for signed formulas.

(108) Give an example of a first-order language $\mathcal{L}$, an injective, atomic, $\mathcal{L}$-substitution and a polynomial $\wp$ such that for every formula $\varphi \in$ PLFORM, $\texttt{size}(s(\varphi)) \leq \wp(\texttt{size}(\varphi))$.
Hint. Choose $\mathcal{L} = \{R_n^0 \mid n \in \mathbf{N}\}$ and $s(p_n) = R_n^0$ for $n \in \mathbf{N}$.

(109) Give examples of logically valid, quantifier-free formulas that are not tautologies.
Hint. Consider, for example, the formula $((x = y) \rightarrow (y = x))$.

(110) Let $\mathcal{L}$ be a first-order language, s be an $\mathcal{L}$-substitution such that $s(p)$ is a sentence for each variable p, and $\mathcal{A}$ be an $\mathcal{L}$-structure. Define a truth assignment v by:

$$v(p) = \begin{cases} \mathbf{T} & \text{if } \mathcal{A} \models s(p) \\ \mathbf{F} & \text{otherwise.} \end{cases}$$

Prove that, for every formula $\varphi \in$ PLFORM, we have $v(\varphi) = \mathbf{T}$ if and only if $\mathcal{A} \models s(\varphi)$.
Hint. Use Lemma 4.8.11.

Normal Forms for Formulas

(111) Construct prenex normal forms for the formulas listed below using the method given in Theorem 4.9.9.

(a) $(((\forall x_1)R_1^1(x_1) \rightarrow (\exists x_2)R_1^2(x_1, x_2)) \rightarrow (\neg(\exists x_3)R_2^1(x_3)))$;
(b) $((\exists x_1)(\forall x_2)R_1^2(x_1, x_2) \rightarrow (\forall x_1)(\exists x_2)R_1^2(x_1, x_2))$.

(112) Construct Skolem normal forms for the formulas given in Exercise 111 using the method given in Algorithm 4.9.14.

(113) Construct a Skolem normal form for the formula

$$(\exists x_0)(\forall x_1)(\forall x_2)(\exists x_3)R(x_0, x_1, x_2, x_3).$$

(114) (a) Prove that every formula in prenex normal form belongs to one of the Π_n or Σ_n classes.
(b) Prove that every formula in prenex normal form that is not quantifier-free belongs to exactly one of the classes Π_n or Σ_n.

Hint. For Part (a), use induction on the number of occurrences of quantifier symbols in the formula in prenex normal form. For Part (b), define precisely the concept of "alternation of quantifier symbol" and show that a formula in Π_n or Σ_n has $n - 1$ alternations of quantifier symbols, for $n \geq 1$.

(115) Let φ be a formula that does not contain the $\leftrightarrow$ connective symbol, and let ψ be a prenex normal form for φ obtained by applying the method of Theorem 4.9.9. Prove that the number of occurrences of each connective symbol is the same in the formulas φ and ψ. Further, prove that the two formulas have the same number of occurrences of quantifier symbols.

(116) Let φ be a formula in Σ_n or Π_n. Prove that for each $k \geq 1$, there is a formula ψ in Σ_{n+k} and a formula θ in Π_{n+k} such that ψ and θ are prenex normal forms φ.
Hint. Use Corollary 4.5.43 and Theorem 4.6.16.

(117) Let φ, ψ be two formulas which both belong to Σ_n or to Π_n for $n \geq 0$. Prove that there are prenex normal forms of the formula $(\varphi \to \psi)$ that belong to Σ_{n+1} and Π_{n+1}, respectively.

(118) Let w be a variable that does not occur free in the formulas

$$\alpha_0, \ldots, \alpha_{i-1}, \alpha_{i+1}, \ldots, \alpha_{n-1},$$

where $0 \leq i \leq n-1$ and $n \geq 1$. Prove that the formulas

$$(\alpha_0 C \cdots C (Qw)\alpha_i C \cdots C \alpha_{n-1})$$

and $(Qw)(\alpha_0 C \cdots C \alpha_{n-1})$ are logically equivalent, where Q is a quantifier symbol and $C \in \{\vee, \wedge\}$.
Hint. The argument is by induction on the number $n - i - 1$ of formulas that follow α_i and uses Lemma 4.9.7.

Reduction of First-Order Logic to Propositional Logic

(119) Let L, L_1, L_2 be first-order languages.

(a) Prove that for every constant symbols a, b we have $\mathcal{H}_a(\mathcal{H}_b(L)) = \mathcal{H}_b(L)$.

(b) If $L_1 \cup L_2$ is first-order language, show that we can have

$$\mathcal{H}_a(L_1 \cup L_2) \neq \mathcal{H}_a(L_1) \cup \mathcal{H}_a(L_2).$$

Under which conditions do we have an equality?

(c) Prove that

$$\mathcal{H}_a(L_1 \cap L_2) = \mathcal{H}_a(L_1) \cap \mathcal{H}_a(L_2).$$

if and only if one of the following cases occur

(i) L_1, L_2 have no constant symbols;

(ii) L_1 and L_2 have some common constant symbols;

(iii) one of L_1, L_2 contains a and the other has no constant symbols.

(120) If S is a subset of $\mathrm{GAFORM}_{\mathcal{L}}$ and R is an n-ary relation symbol of $\mathcal{L}$ with $n > 0$, show that $\mathsf{STR}(S) \models R(t_0, \ldots, t_{n-1})$ if and only if $R(t_0, \ldots, t_{n-1}) \in S$, where $t_0, \ldots, t_{n-1}$ are $\mathcal{L}$-ground terms.

(121) Let $\mathcal{A}$ be a Herbrand structure for a first-order language $\mathcal{L}$ and let t be a term in $\mathrm{TERM}_{\mathcal{L}}$. Prove that for every $\sigma \in \mathrm{ASSIGN}_{\mathcal{A}}$, we have the equality $\sigma^{\mathcal{A}}(t) = \mathsf{s}^{y_0 \cdots y_{n-1}}_{\sigma(y_0) \cdots \sigma(y_{n-1})}(t)$, where $y_0, \ldots, y_{n-1}$ are distinct variables and $\mathtt{V}(t) \subseteq \{y_0, \ldots, y_{n-1}\}$.

(122) Let $\mathcal{L}$ be a first-order language that contains at least one constant symbol and let $\mathcal{A}$ be a Herbrand structure for $\mathcal{L}$. Prove that $\mathcal{A}$ has no substructures different from $\mathcal{A}$ itself.

(123) Let $\mathcal{L}$ be a first-order language, V be a nonempty set of variables, and $\mathcal{A}$ be a V-Herbrand structure for $\mathcal{L}$. Prove that the set of ground terms is the domain of a proper substructure of $\mathcal{A}$.

(124) Give an example of a first-order language $\mathcal{L}$ without equality and a set of closed (but not universal) $\mathcal{L}$-formulas Γ such that Γ has a model, but Γ does not have any $\mathcal{L}$-Herbrand model.

Solution: Let $\mathcal{L} = \{a, R\}$, where a is a constant symbol and R is a unary relation symbol. $\Gamma = \{(R(a) \wedge (\exists x)(\neg R(x)))\}$ has the desired property. Indeed, Γ obviously has a model; however, the domain of any $\mathcal{L}$-Herbrand structure has only one element (namely, a) and no such structure can be a model of Γ.

(125) Let $\mathcal{L}$ be a first-order language without equality, V be a set of variables, and Γ be a satisfiable set of not necessarily quantifier-free $\mathcal{L}$-formulas with $\mathtt{FV}(\Gamma) \subseteq V$. Prove that there is a first-order language $\mathcal{L}'$ such that $\mathcal{L} \subseteq \mathcal{L}'$, $\mathcal{L}' - \mathcal{L}$ contains only function symbols, and Γ is satisfiable in a V-Herbrand structure for $\mathcal{L}'$.

Solution: By Theorem 4.9.19, there is first-order language $\mathcal{L}'$ and a set of $\mathcal{L}'$-formulas Γ' such that $\mathcal{L} \subseteq \mathcal{L}'$ and Γ' is a Skolemization of Γ. Corollary 4.9.21 implies that Γ' is satisfiable. Since $\mathtt{FV}(\Gamma') = \mathtt{FV}(\Gamma) \subseteq V$, by Theorem 4.10.19, Γ' is satisfiable in a V-Herbrand structure $\mathcal{A}$ for $\mathcal{L}'$. Therefore, by Theorem 4.9.20, Γ is satisfiable in $\mathcal{A}$.

(126) Show that Theorem 4.10.34 fails if one allows first-order languages to be uncountable.

Solution: Let $\mathcal{L}$ consist of a binary relation symbol E and uncountably many constant symbols $\{c_r \mid r \in \mathbf{R}\}$ (where $\mathbf{R}$ is the set of real numbers). Consider the set Γ of $\mathcal{L}$-formulas given by

$$\begin{aligned}\Gamma = \{&(\forall x)E(x,x), (\forall x)(\forall y)(E(x,y) \rightarrow E(y,x)),\\ &(\forall x)(\forall y)(\forall z)((E(x,y) \wedge E(y,z)) \rightarrow E(x,z))\}\\ &\cup\{(\neg E(c,d)) \mid c, d \text{ are distinct constant symbols in } \mathcal{L}\}\end{aligned}$$

Consider the $\mathcal{L}$-structure $\mathcal{R}$ such that $|\mathcal{R}| = \mathbf{R}$, $E^{\mathcal{R}} = \{(r,r') \in \mathbf{R}\times\mathbf{R} \mid r = r'\}$ and $c_r^{\mathcal{R}} = r$ for $r \in \mathbf{R}$. Since equality on $\mathbf{R}$ is an equivalence relation, Γ is satisfiable in $\mathcal{R}$. On the other hand, if Γ is satisfiable in an $\mathcal{L}$-structure $\mathcal{A}$, then there is an equivalence relation on $|\mathcal{A}|$ with an uncountable set of equivalence classes, one for each constant symbol c_r. Therefore, if $\mathcal{B}$ is any $\mathcal{L}$-structure with a countable universe, Γ is not satisfiable in $\mathcal{B}$.

(127) Let Γ be a set of $\mathcal{L}$-formulas. Prove that if Γ has arbitrarily large finite models, then Γ has an infinite model.

(128) Let $\mathcal{L}$ be a first-order language and let Γ be a set of universal $\mathcal{L}$-formulas with $\mathtt{FV}(\Gamma) \subseteq V$, where V is an $\mathcal{L}$-suitable set of variables. Show that if $\mathrm{INST}_{\mathcal{L},V}(\Gamma)$ is satisfiable in an $\mathcal{L}$-structure $\mathcal{A}$, then Γ is satisfiable in a substructure $\mathcal{B}$ of $\mathcal{A}$ that is isomorphic to a quotient of a V-Herbrand structure for $\mathcal{L}$.

Solution: Let $\sigma \in \mathrm{ASSIGN}_{\mathcal{A}}$ be such that $(\mathcal{A},\sigma) \models \mathrm{INST}_{\mathcal{L},V}(\Gamma)$, $h : |\mathcal{H}_V(\mathcal{A},\sigma)| \longrightarrow |\mathcal{A}|$ be the morphism of Theorem 4.10.16, and let B be the range of h, that is, $B = \{\sigma^{\mathcal{A}}(t) \mid t \in \mathrm{TERM}_{\mathcal{L}}(V)\}$. Let $\mathcal{B}$ be the substructure of $\mathcal{A}$ with domain B. By Theorem 4.4.35, $\mathcal{B}$ is isomorphic to a quotient of the V-Herbrand structure $\mathcal{H}_V(\mathcal{A},\sigma)$ for $\mathcal{L}$. If Γ is satisfiable in $\mathcal{B}$, then the result follows. Let τ be an assignment over $\mathcal{B}$ such that $\tau(x) = \sigma(x)$ for every $x \in V$.

Let $\varphi = (\forall y_0)\cdots(\forall y_{n-1})\psi$, where ψ is quantifier-free, be a formula in Γ. The following statements are equivalent:

(a) $(\mathcal{B},\tau) \models \varphi$;

(b) for every $t_0, \ldots, t_{n-1} \in \mathrm{TERM}_{\mathcal{L}}(V)$,

$$(\mathcal{B}, [y_0 \to \sigma^{\mathcal{A}}(t_0)] \cdots [y_{n-1} \to \sigma^{\mathcal{A}}(t_{n-1})]\tau) \models \psi;$$

(c) for every $t_0, \ldots, t_{n-1} \in \mathrm{TERM}_{\mathcal{L}}(V)$,

$$(\mathcal{A}, [y_0 \to \sigma^{\mathcal{A}}(t_0)] \cdots [y_{n-1} \to \sigma^{\mathcal{A}}(t_{n-1})]\tau) \models \psi;$$

(d) for every $t_0, \ldots, t_{n-1} \in \mathrm{TERM}_{\mathcal{L}}(V)$,

$$(\mathcal{A}, [y_0 \to \sigma^{\mathcal{A}}(t_0)] \cdots [y_{n-1} \to \sigma^{\mathcal{A}}(t_{n-1})]\sigma) \models \psi;$$

(e) $(\mathcal{A}, \sigma) \models (\psi)_{y_0, \ldots, y_{n-1} := t_0, \ldots, t_{n-1}}$.

The equivalence of (b) and (c) follows from the second part of Exercise 41. The equivalence of (c) and (d) follows from the Agreement Theorem since the assignments involved agree on all variables that occur in ψ. Finally, the equivalence between (d) and (e) follows from Corollary 4.6.6. Since $(\mathcal{A}, \sigma) \models \mathrm{INST}_{\mathcal{L},V}(\Gamma)$, the last statement holds, so the first statement is true.

(129) Show that Supplement 128 can be used to provide alternative arguments for the following implications:

(a) (3) $\longrightarrow$ (1) of Theorem 4.10.40;
(b) (4) $\longrightarrow$ (2) of Theorem 4.10.41.

Hint. Use Exercise 122 for the second part.

(130) Let $\mathcal{L}$ be a first-order language, Γ be a set of universal $\mathcal{L}$-formulas and V be an $\mathcal{L}$-suitable set of variables such that $\mathtt{FV}(\Gamma) \subseteq V$. Prove that the following statements are equivalent.

(a) Γ is satisfiable.
(b) Γ is satisfiable in a quotient of a V-Herbrand structure for $\mathcal{L}$.
(c) $\mathrm{INST}_{\mathcal{L},V}(\Gamma)$ is satisfiable.
(d) $\mathrm{INST}_{\mathcal{L},V}(\Gamma)$ is satisfiable in a quotient of a V-Herbrand structure for $\mathcal{L}$.

Solution: The plan of our argument is shown in the diagram below.

$$\begin{tikzcd}
\text{(a)} \arrow[r, leftrightarrow] & \text{(c)} \arrow[dl] \arrow[d, leftrightarrow] \\
\text{(b)} \arrow[u] & \text{(d)}
\end{tikzcd}$$

Theorems 4.10.19 and 4.10.40 imply the equivalence of (a) and (c). It is clear that (b) implies (a) and (d) implies (c). The fact that (c) implies (b) follows from Supplement 128. Finally, (c) implies (d) follows from the same supplement by replacing Γ with $\mathrm{INST}_{\mathcal{L},V}(\Gamma)$ and observing that $\mathrm{INST}_{\mathcal{L},V}(\mathrm{INST}_{\mathcal{L},V}(\Gamma)) = \mathrm{INST}_{\mathcal{L},V}(\Gamma)$.

(131) Let $\mathcal{L}$ be a first-order language, V be a set of variables, and Γ be a satisfiable set of not necessarily quantifier-free $\mathcal{L}$-formulas with $\mathtt{FV}(\Gamma) \subseteq V$. Prove that there is a first-order language $\mathcal{L}'$ such that $\mathcal{L} \subseteq \mathcal{L}'$, $\mathcal{L}' - \mathcal{L}$ consists of function symbols, and Γ is satisfiable in a quotient of a V-Herbrand structure for $\mathcal{L}'$.

Solution: By Theorem 4.9.19, there is first-order language $\mathcal{L}'$ and a set of $\mathcal{L}'$-formulas Γ' such that $\mathcal{L} \subseteq \mathcal{L}'$ and Γ' is a Skolemization of Γ. Corollary 4.9.21 implies that Γ' is satisfiable. Since $\mathtt{FV}(\Gamma') = \mathtt{FV}(\Gamma) \subseteq V$, by Supplement 130, Γ' is satisfiable in a quotient $\mathcal{B}$ of a V-Herbrand structure $\mathcal{A}$ for $\mathcal{L}'$. Therefore, by Theorem 4.9.20, Γ is satisfiable in $\mathcal{B}$.

Brand's Modification Method

(132) Let R be a binary relation symbol and P be a unary relation symbol and let φ be the R-flat formula $(\exists x)(\exists y)(R(x,y) \wedge P(x) \wedge (\neg P(y)))$. Prove that φ has a model $\mathcal{B}$ with $R^{\mathcal{B}}$ an equivalence, but fails to have a model $\mathcal{A}$ with $R^{\mathcal{A}}$ a congruence. (Note that this exercise shows that the assumption in Theorem 4.11.3 that the set Γ consists of quantifier-free formulas is essential.)

(133) Give a direct argument that the set Γ introduced in Example 4.11.2 has no model $\mathcal{A}$ where $R^{\mathcal{A}}$ is an equivalence relation. Also, prove that Γ has a model.

(134) Give a proof of Theorem 4.11.12 using induction on the number of steps in the application of Algorithm 4.11.9 to Γ.

(135) Let $\Gamma = \{R(a,b), (\neg R(f(a), f(b)))\}$, where R is a binary relation symbol, f is a unary function symbol, and a, b are constant symbols.

(a) Give a direct argument that Γ has no model $\mathcal{A}$ where $R^{\mathcal{A}}$ is a congruence.

(b) Compute an R-flattening $\Gamma^{\langle R\rangle}$ of Γ using Algorithm 4.11.9. By Theorem 4.11.10, $\Gamma^{\langle R\rangle}$ has no model $\mathcal{A}$ with $R^{\mathcal{A}}$ an equivalence relation. Give a direct proof of this fact.

(c) Let $\widetilde{\Gamma^{\langle R\rangle}}$ be an e-expansion of $\Gamma^{\langle R\rangle}$ by R. According to Theorem 4.11.17, $\widetilde{\Gamma^{\langle R\rangle}}$ has no model. Give a direct proof of this fact.

(d) Let Γ' be a positive e-expansion of $\Gamma^{\langle R\rangle}$ by R. According to Theorem 4.11.23, Γ' has no model $\mathcal{A}$ with $R^{\mathcal{A}}$ a reflexive relation. Give a direct proof of this fact.

Hintikka Sets and Truth Sets

(136) Let $\mathcal{L} = \{c, s, f, g, =\}$ be a first-order language, where c is a constant symbol, s is a unary function symbol, and f, g are binary function symbols, and let $V = \{x_0\}$. Consider the $\mathcal{L}$-structure $\mathcal{A}$, with $|\mathcal{A}| = \mathbf{N}[X]$, where $\mathbf{N}[X]$ is the set of all polynomials in X whose coefficients are natural numbers, $c^{\mathcal{A}} = 0$, $s^{\mathcal{A}}(p) = p + 1$, $f^{\mathcal{A}}(p, q) = p + q$, and $g^{\mathcal{A}}(p, q) = qp$. Prove that if $\sigma \in \mathrm{ASSIGN}_{\mathcal{A}}$ is such that $\sigma(x_0) = X$, then the pair $(\mathcal{A}, \sigma)$ is V-named.

(137) Let $\mathcal{L} = \{\neg, \vee, \wedge\}$, where $\neg$ is a unary function symbol and $\vee, \wedge$ are binary function symbols, $\mathcal{A}$ be the $\mathcal{L}$-structure with $|\mathcal{A}| = \mathbf{Bool}^n \longrightarrow \mathbf{Bool}$, for some $n > 0$, where

$$\neg^{\mathcal{A}}(f)(b_0, \ldots, b_{n-1}) = \neg(f(b_0, \ldots, b_{n-1}))$$
$$\vee^{\mathcal{A}}(f, g)(b_0, \ldots, b_{n-1}) = f(b_0, \ldots, b_{n-1}) \vee g(b_0, \ldots, b_{n-1})$$
$$\wedge^{\mathcal{A}}(f, g)(b_0, \ldots, b_{n-1}) = f(b_0, \ldots, b_{n-1}) \wedge g(b_0, \ldots, b_{n-1})$$

for $b_0, \ldots, b_{n-1} \in \mathbf{Bool}$, and let $V = \{x_0, \ldots, x_{n-1}\}$. Prove that $(\mathcal{A}, \sigma)$ is a V-named pair, where $\sigma(x_i) = \pi_i^n$, for $0 \leq i \leq n - 1$.

(138) Let $\mathcal{A}$ be an $\mathcal{L}$-structure such that $|\mathcal{A}|$ is uncountable. Prove that there is no set V of variables and assignment σ such that $(\mathcal{A}, \sigma)$ is V-named.
Hint. Recall that for us first-order languages and sets of variables are countable sets.

(139) (a) Prove that for every formula φ, φ is logically valid if and only if φ belongs to every $({\mathcal{L}_\varphi}^c, \mathtt{FV}(\varphi))$-truth set.

(b) Let $\varphi = ((\exists x)R(x) \to R(a))$. Verify that φ is not logically valid and that φ belongs to every $\mathcal{L}_\varphi$-truth set.

(c) Let $\mathcal{L}$ be a first-order language, $\Gamma' = \Gamma \cup \{\varphi\} \subseteq \text{FORM}_\mathcal{L}$, and $V = \mathtt{FV}(\Gamma')$. Prove that $\Gamma \models \varphi$ if and only if φ belongs to every $(\mathcal{L}^c, V)$-truth set that contains Γ.

(d) Reformulate the notion of logical equivalence using the notion of truth set.

(140) Use Theorems 4.12.35 and 4.12.11 to give a semantic proof of Corollary 4.12.31.

(141) Prove that Exercises 74 and 76 of Chapter 2 remain valid if we replace "Hintikka set" with "$(\mathcal{L}, V)$-Hintikka set," where $\mathcal{L}$ is a first-order language and V is an $\mathcal{L}$-suitable set of variables.

(142) Let $\mathcal{L}$ be a first-order language, V be an $\mathcal{L}$-suitable set of variables, and let φ be a positive formula that is not atomic. Prove that if K is an $(\mathcal{L}, V)$-constituent of φ and H is a $(\mathcal{L}, V)$-constituent of the formula $(\neg\varphi)$, then there is a formula γ such that $\| \gamma \| < \| \varphi \|$ and $K \cup H$ contains both γ and $(\neg\gamma)$.

(143) Show that the second part of Corollary 4.12.13 can fail if α is not a closed formula.

(144) Let $\mathcal{L}$ be a first-order language and V be an $\mathcal{L}$-suitable set of variables. Prove that:

(a) If $\mathcal{A}, \mathcal{B}$ are $\mathcal{L}$-structures, $\sigma \in \text{ASSIGN}_\mathcal{A}$ is such that $(\mathcal{A}, \sigma)$ is V-named, and $k : |\mathcal{A}| \longrightarrow |\mathcal{B}|$ is an epimorphism, then $(\mathcal{B}, \tau)$ is V-named, where $\tau = k \circ \sigma$. Conclude that if k is an isomorphism, then $(\mathcal{A}, \sigma)$ is V-named if and only if $(\mathcal{B}, \tau)$ is V-named.

(b) If $\mathcal{A}, \mathcal{B}$ are $\mathcal{L}$-structures, $k : |\mathcal{A}| \longrightarrow |\mathcal{B}|$ is an epimorphism, and $\mathcal{A}$ is named, then $\mathcal{B}$ is named. Conclude that if k is an isomorphism, then $\mathcal{A}$ is named if and only if $\mathcal{B}$ is named.

The next supplement will show that V-named pairs can be identified up to isomorphism with the particular type of V-named pair identified in Example 4.12.10.

(145) Let $\mathcal{L}$ be a first-order language and V be an $\mathcal{L}$-suitable set of variables.
Show that a pair $(\mathcal{A}, \sigma)$, where $\mathcal{A}$ is an $\mathcal{L}$-structure and σ is an assignment, $\sigma \in \text{ASSIGN}_\mathcal{A}$, is V-named if and only if there is

a V-Herbrand structure $\mathcal{B}$ for $\mathcal{L}$, a congruence ρ on $\mathcal{B}$, and a $\tau \in \mathrm{ASSIGN}_{\mathcal{B}/\rho}$ with $\tau(x) = [x]_\rho$ for all $x \in V$ such that there is an isomorphism $k : |\mathcal{A}| \longrightarrow |\mathcal{B}/\rho|$ with $\tau = k \circ \sigma$.

Solution: Suppose that $(\mathcal{A}, \sigma)$ is a V-named structure. Then, if $\mathcal{B} = \mathcal{H}_V(\mathcal{A}, \sigma)$, as introduced in Theorem 4.10.16, the morphism $h : |\mathcal{B}| \longrightarrow |\mathcal{A}|$, given by $h(t) = \sigma^{\mathcal{A}}(t)$ for $t \in \mathrm{TERM}_{\mathcal{L}}(V)$ is an epimorphism. Consequently, by Theorem 4.4.35, the mapping $\hat{h} : |\mathcal{B}/\rho| \longrightarrow |\mathcal{A}|$, given by $\hat{h}([t]) = \sigma^{\mathcal{A}}(t)$ is an isomorphism, where $\rho = \mathbf{ker}(h)$. Thus, the mapping $k = \hat{h}^{-1}$ (where $k(a) = [t]_\rho$ for $t \in \mathrm{TERM}_{\mathcal{L}}(V)$ such that $\sigma^{\mathcal{A}}(t) = a$) is an isomorphism. Further, by taking $\tau = k \circ \sigma$, we have $\tau(x) = k(\sigma(x)) = k(\sigma^{\mathcal{A}}(x)) = [x]_\rho$ for all $x \in V$.
The converse follows from Exercise 144, Part (a).

(146) Let $\mathcal{L}$ be a first-order language that contains at least one constant symbol. Show that an $\mathcal{L}$-structure $\mathcal{A}$ is named if and only if $\mathcal{A}$ is isomorphic to a quotient of a Herbrand structure for $\mathcal{L}$.

(147) Let $\mathcal{L}$ be a first-order language and V be an $\mathcal{L}$-suitable set of variables. Prove that $\mathtt{D}_{\mathcal{L},V}(\varphi)$ is finite for all $\varphi \in \mathrm{FORM}_{\mathcal{L}}(V)$ if and only if all of the relation symbols of $\mathcal{L}$ are propositional constants or $\mathrm{TERM}_{\mathcal{L}}(V)$ is finite.

(148) Let $\mathcal{L}$ be a first-order language whose relation symbols are all propositional constants. Prove that if V is an $\mathcal{L}$-suitable set of variables, then the collection of all satisfiable subsets of $\mathrm{FORM}_{\mathcal{L}}(V)$ is an $(\mathcal{L}, V)$-consistency property.

(149) Let $\mathcal{L}$ be a first-order language that contains at least one relation symbol R that is not a propositional constant and infinitely many constant symbols; let V be a set of variables. In contrast with Supplement 77 of Chapter 2, give an example of a $(\mathcal{L}, V)$-consistency property $\mathcal{C}$ such that $\mathcal{C}'$, the smallest property of finite character of the subsets of $\mathrm{FORM}_{\mathcal{L}}(V)$ that contains $\mathcal{C}$ is not an $(\mathcal{L}, V)$-consistency property.

Solution: Let $\mathcal{C} = \{\Gamma \subseteq \mathrm{FORM}_{\mathcal{L}}(V) \mid \Gamma$ is satisfiable and Γ contains only finitely many constant symbols $\}$. Following an argument similar to the one in Theorem 4.12.25, one can prove that $\mathcal{C}$ is an $(\mathcal{L}, V)$-consistency property. Let

$$\Gamma = \{(\exists x)(\neg R(x, t_0, \ldots, t_0))\} \cup \{R(t, t_0, \ldots, t_0) \mid t \in \mathrm{TERM}_{\mathcal{L}}(V)\},$$

where t_0 is an arbitrary, fixed term in $\text{TERM}_{\mathcal{L}}(V)$. We proved in Example 4.12.24 that Γ is satisfiable and, therefore, every finite subset of Γ belongs to $\mathcal{C}$, which in turn implies that $\Gamma \in \mathcal{C}'$. If $\mathcal{C}'$ were a consistency property, this would imply that $\Gamma \cup K \in \mathcal{C}'$, where K is a constituent of $(\exists x)(\neg R(x, t_0, \ldots, t_0))$, so $\Gamma \cup K$ is satisfiable. This generates a contradiction, as we saw in Theorem 4.12.25.

(150) Let $\mathcal{C}$ be a $(\mathcal{L}, V)$-consistency property, where $\mathcal{L}$ is a first-order language and V is an $\mathcal{L}$-suitable set of variables. Prove that every maximal element of $\mathcal{C}$ is an $(\mathcal{L}, V)$-Hintikka set.

(151) Formulate and prove an analogue of Exercise 83 of Chapter 2 for first-order logic.

(152) Let $\mathcal{L}$ be a first-order language and let s be an $\mathcal{L}$-substitution. ($\mathcal{L}$-substitutions were defined at the beginning of Section 4.8.) Prove that for every signed propositional formula $b\varphi$ such that φ is not a variable, and set of variables V, we have $\mathtt{d}_{\mathcal{L},V}(s(b\varphi)) = s(\mathtt{d}(b\varphi))$.

(153) Let $\mathcal{L}$ be a first-order language, V be a set of variables and let $\alpha, \beta \in \text{FORM}_{\mathcal{L}}(V)$. For a a constant symbol in $\mathcal{L}$ and $t \in \text{TERM}_{\mathcal{L}}(V)$, prove that if $\varphi = (\alpha C \beta)$ for some binary connective symbol C or $\varphi = (\neg\alpha)$, then $\mathtt{s}_t^a(\mathtt{d}_{\mathcal{L},V}(b\varphi)) = \mathtt{d}_{\mathcal{L},V}(\mathtt{s}_t^a(b\varphi))$ for $b \in \textbf{Bool}$.

(154) Let $\mathcal{L}$ be a first-order language and V be a set of variables. Prove that if Γ is a satisfiable set of $(\mathcal{L}, V)$-formulas, then there is an $(\mathcal{L}^c, V)$-Hintikka set Γ' such that

$$\Gamma \subseteq \Gamma' \subseteq \begin{cases} W^*_{\mathcal{L}^c,V}(\Gamma) & \text{if } = \notin \mathcal{L} \\ W^*_{\mathcal{L}^c,V}(\Gamma \cup \text{INST}_{\mathcal{L}^c,V}(\text{Eq}_{=,\mathcal{L}})) & \text{if } = \in \mathcal{L} \end{cases}$$

Let $\mathcal{L}$ be a first-order language and V be an $\mathcal{L}$-suitable set of variables.

A set of formulas $\Gamma \subseteq \text{FORM}_{\mathcal{L}}(V)$ is *p-downward* $(\mathcal{L}, V)$*-closed* if for every positive formula $\varphi \in \Gamma$ that is not a literal at least one $(\mathcal{L}, V)$-constituent of φ is included in Γ. The notions np-downward closed, nn-downward closed, p-upward closed, np-upward closed, nn-upward closed, downward closed, and upward closed from propositional logic are similarly adopted to first-order logic.

A set of formulas $\Gamma \subseteq \text{FORM}_{\mathcal{L}}(V)$ is $(\mathcal{L}, V)$*-saturated* if the following conditions are satisfied:

- for every $(\mathcal{L}, V)$-formula φ, we have $\varphi \in \Gamma$ if and only if $(\neg\varphi) \notin \Gamma$;
- Γ is both upward and downward closed;
- if $= \in \mathcal{L}$, then $\mathrm{INST}_{\mathcal{L},V}(\mathrm{Eq}_{=,\mathcal{L}}) \subseteq \Gamma$.

(155) Show that Theorems 2.7.5 and 2.7.7, Exercises and Supplements 84 to 86 and Exercises 88 and 90 of Chapter 2 can be suitably modified to hold for first-order logic.
For example, Theorem 2.7.5 can be restated as follows:
Let $\mathcal{L}$ be a first-order language, V be an $\mathcal{L}$-suitable set of variables, and let $\Gamma \subseteq \mathrm{FORM}_{\mathcal{L}}(V)$ be a set of formulas such that for every formula $\varphi \in \mathrm{FORM}_{\mathcal{L}}(V)$, exactly one of φ and $(\neg\varphi)$ belongs to Γ. Prove that

(a) Γ is upward $(\mathcal{L}, V)$-closed if and only if it is p-upward and np-upward $(\mathcal{L}, V)$-closed.
(b) Γ is downward $(\mathcal{L}, V)$-closed if and only if it is p-downward and np-downward $(\mathcal{L}, V)$-closed.

Exercise 86 of Chapter 2 can be restated as follows.
Let $\mathcal{L}$ be a first-order language and V be an $\mathcal{L}$-suitable set of variables. Show, with examples, that a set of $(\mathcal{L}, V)$ formulas Γ such that $\ell \in \Gamma$ if and only if $\overline{\ell} \notin \Gamma$ for all $(\mathcal{L}, V)$-literals ℓ, can satisfy any five of the following six conditions and fail to be a truth set:

(a) Γ is p-upward $(\mathcal{L}, V)$-closed.
(b) Γ is np-upward $(\mathcal{L}, V)$-closed.
(c) Γ is nn-upward $(\mathcal{L}, V)$-closed.
(d) Γ is p-downward $(\mathcal{L}, V)$-closed.
(e) Γ is np-downward $(\mathcal{L}, V)$-closed.
(f) Γ is nn-downward $(\mathcal{L}, V)$-closed.

(156) Let $\mathcal{L}$ be a first-order language without equality, V be an $\mathcal{L}$-suitable set of variables, and $\Gamma \subseteq \mathrm{FORM}_{\mathcal{L}}(V)$ be a set of formulas. Prove that if $\ell \in \Gamma$ if and only if $\overline{\ell} \notin \Gamma$ for every literal ℓ and Γ is upward $(\mathcal{L}, V)$-closed, then Γ contains an $(\mathcal{L}, V)$-truth set.

(157) Let $\mathcal{L}$ be a first-order language and V be an $\mathcal{L}$-suitable set of variables. Prove that Γ is an $(\mathcal{L}, V)$-Hintikka set if and only if Γ is a downward $(\mathcal{L}, V)$-closed subset of an $(\mathcal{L}, V)$-truth set and Γ includes $\mathrm{INST}_{\mathcal{L},V}(\mathrm{Eq}_{=,\mathcal{L}})$, when $= \in \mathcal{L}$.

(158) Show that analogues of Exercise 91 and Supplement 92 of Chapter 2 hold for first-order logic.

(159) Let Γ be a set of $\mathcal{L}$-formulas, $\mathcal{A}$ be an $\mathcal{L}$-structure and σ be an assignment in $\text{ASSIGN}_{\mathcal{A}}$, where $\mathcal{L}$ is a first-order language. Prove that the set of signed formulas $\{\mathcal{S}_{\mathcal{A}}(\varphi)(\sigma)\varphi \mid \varphi \in \Gamma\}$ is satisfiable.

Define $\mathsf{u} : \text{SFORM} \longrightarrow \text{FORM}$ and $\mathsf{s} : \text{FORM} \longrightarrow \text{SFORM}$ by:

$$\mathsf{u}(b\varphi) = \begin{cases} \varphi & \text{if } b = \mathbf{T} \\ (\neg\varphi) & \text{if } b = \mathbf{F} \end{cases} \tag{4.24}$$

for every $b\varphi \in \text{SFORM}$, and

$$\mathsf{s}(\varphi) = \begin{cases} \mathbf{T}\varphi & \text{if } \varphi \text{ is positive} \\ \mathbf{F}\psi & \text{if } \varphi = (\neg\psi). \end{cases} \tag{4.25}$$

for every $\varphi \in \text{FORM}$.

Define a function $\Xi : \mathcal{P}(\text{SFORM}) \longrightarrow \mathcal{P}(\text{FORM})$ by $\Xi(\Delta) = \{\varphi \mid \mathbf{T}\varphi \in \Delta\} \cup \{(\neg\varphi) \mid \mathbf{F}\varphi \in \Delta\}$ and a function $\Upsilon : \mathcal{P}(\text{FORM}) \longrightarrow \mathcal{P}(\text{SFORM})$ by $\Upsilon(\Gamma) = \{\mathbf{T}\varphi \mid \varphi \in \Gamma\} \cup \{\mathbf{F}\varphi \mid (\neg\varphi) \in \Gamma\}$.

Observe that $\Xi(\Delta)$ is the extension of the function u, defined above, to sets of signed formulas.

(160) (a) Prove that Exercises 96, 97, 99(a,b), 100 and Supplement 98(a-f) of Chapter 2 can be suitably reformulated for first-order logic.
(b) Derive Theorem 4.12.50 from Theorem 4.12.17 and Theorem 4.12.53 from Theorem 4.12.20 using the modified version of Supplement 98 of Chapter 2.
(c) Derive Theorem 4.12.17 from Theorem 4.12.50 and Theorem 4.12.20 from Theorem 4.12.53 using the modified version of Exercise 99 of Chapter 2.

(161) Let φ, φ' be two first-order formulas. Prove that φ is a variant of φ' if and only if $\mathsf{s}(\varphi)$ is a variant of $\mathsf{s}(\varphi')$.

(162) Let φ, φ' be two first-order formulas and $b \in \mathbf{Bool}$. Prove that $b\varphi$ is a variant of $b\varphi'$ if and only if $\mathsf{u}(b\varphi)$ is a variant of $\mathsf{u}(b\varphi')$.

Let $\mathcal{L}$ be a first-order language and V be a set of variables.

A set of signed $(\mathcal{L}, V)$-formulas Δ is $(\mathcal{L}, V)$*-maximally satisfiable* if it is satisfiable and there is no satisfiable set $\Delta' \subseteq \text{SFORM}_{\mathcal{L}}(V)$ such that $\Delta \subset \Delta'$.

Assume in addition now that V is $\mathcal{L}$-suitable. A set of signed $(\mathcal{L}, V)$-formulas Δ is **T**-*downward* $(\mathcal{L}, V)$*-closed* if for every signed formula $\mathbf{T}\varphi \in \Delta - (\mathbf{Bool} \times \text{AFORM}_{\mathcal{L}})$, there is an $(\mathcal{L}, V)$-constituent K of $\mathbf{T}\varphi$ such that $K \subseteq \Delta$. The similar notions of **F**-downward $(\mathcal{L}, V)$-closed, downward $(\mathcal{L}, V)$-closed, **T**-upward $(\mathcal{L}, V)$-closed, **F**-upward $(\mathcal{L}, V)$-closed, and upward $(\mathcal{L}, V)$-closed are adapted from their propositional counterparts.

A set of signed $(\mathcal{L}, V)$-formulas Δ is $(\mathcal{L}, V)$*-saturated* if the following conditions are satisfied:

- for every $(\mathcal{L}, V)$-formula φ, we have $\mathbf{T}\varphi \in \Delta$ if and only if $\mathbf{F}\varphi \notin \Delta$;
- Δ is both $(\mathcal{L}, V)$-upward and $(\mathcal{L}, V)$-downward closed;
- if $= \in \mathcal{L}$, then $\mathbf{T}\varphi \in \Delta$ for every $\varphi \in \text{INST}_{\mathcal{L},V}(\text{Eq}_{=,\mathcal{L}})$.

(163) State and prove a characterization of $(\mathcal{L}, V)$-maximally satisfiable sets of signed formulas similar to the one contained in Exercise 101 of Chapter 2.

(164) Let $\mathcal{L}$ be a first-order language and V be a set of variables. Prove that a set of signed formulas Δ is $(\mathcal{L}, V)$-maximally satisfiable if and only if there is an $\mathcal{L}$-structure $\mathcal{A}$ and an assignment $\sigma \in \text{ASSIGN}_{\mathcal{A}}$ such that $\Delta = \{b\varphi \in \text{SFORM}_{\mathcal{L}}(V) \mid (\mathcal{A}, \sigma) \models b\varphi\}$.

(165) Let $\mathcal{L}$ be a first-order language and let V be an $\mathcal{L}$-suitable set of variables. Prove that a set $\Delta \subseteq \text{SFORM}_{\mathcal{L}}(V)$ is an $(\mathcal{L}, V)$-saturated set if and only if there is an $\mathcal{L}$-structure $\mathcal{A}$ and $\sigma \in \text{ASSIGN}_{\mathcal{A}}$ such that $(\mathcal{A}, \sigma)$ is V-named and

$$\Delta = \{b\varphi \in \text{SFORM}_{\mathcal{L}}(V) \mid (\mathcal{A}, \sigma) \models b\varphi\}.$$

Hint. The argument is similar to the one used in Theorem 4.12.35.

(166) Formulate and prove first-order analogues of Exercises and Supplements 104 to 113 of Chapter 2. For example, the first-order analogue of Exercise 104 reads as follows.
Let $\mathcal{L}$ be a first-order language, V be an $\mathcal{L}$-suitable set of variables, and φ be an $(\mathcal{L}, V)$-formula that is not atomic. Suppose that K is an $(\mathcal{L}, V)$-constituent of the signed formula $\mathbf{T}\varphi$ and H is an $(\mathcal{L}, V)$-constituent of the signed formula $\mathbf{F}\varphi$. Prove that $K \cup H$ is closed.

The notion of consistency property can be formulated for signed formulas. Namely, let $\mathcal{L}$ be a first-order language and V be an $\mathcal{L}$-suitable set of variables. An $(\mathcal{L}, V)$-*consistency property* is a collection $\mathcal{C}$ of sets of signed $(\mathcal{L}, V)$-formulas such that

- No set with property $\mathcal{C}$ contains both $\mathbf{T}\varphi$ and $\mathbf{F}\varphi$ for any atomic formula φ.
- If Δ has property $\mathcal{C}$, $b\varphi \in \Delta$ and $b\varphi$ is neither a signed atomic formula nor a $\boldsymbol{\gamma}$-formula, then $\Delta \cup K$ has property $\mathcal{C}$ for some $(\mathcal{L}, V)$-constituent K of $b\varphi$.
- If Δ has property $\mathcal{C}$ and $b\varphi = b(Qx)\psi$ is a $\boldsymbol{\gamma}$-formula, then $\Delta \cup \{b\langle\psi\rangle_{x:=t}\}$ has property $\mathcal{C}$ for every $t \in \mathrm{TERM}_{\mathcal{L}}(V)$.
- If $= \in \mathcal{L}$, Δ has property $\mathcal{C}$, and $\alpha \in \mathrm{INST}_{\mathcal{L},V}(\mathrm{Eq}_{=,\mathcal{L}})$, then $\Delta \cup \{\alpha\}$ has property $\mathcal{C}$.

A collection of sets of signed $(\mathcal{L}, V)$-formulas $\mathcal{I}$ is an $(\mathcal{L}, V)$-*inconsistency property* if $\mathcal{P}(\mathrm{SFORM}_{\mathcal{L}}(V)) - \mathcal{I}$ is an $(\mathcal{L}, V)$-consistency property.

Note that a collection of sets of signed $(\mathcal{L}, V)$-formulas $\mathcal{I}$ is an $(\mathcal{L}, V)$-inconsistency property if and only if the following conditions are satisfied:

- Every set of signed $(\mathcal{L}, V)$-formulas including $\{\mathbf{T}\varphi, \mathbf{F}\varphi\}$ for some atomic formula φ belongs to $\mathcal{I}$.
- If Δ is a set of signed $(\mathcal{L}, V)$-formulas, $b\varphi \in \Delta$ is neither a signed atomic formula nor a $\boldsymbol{\gamma}$-formula, and for every $(\mathcal{L}, V)$-constituent K of φ, $\Delta \cup K \in \mathcal{I}$, then $\Delta \in \mathcal{I}$.
- If Δ is a set of signed $(\mathcal{L}, V)$-formulas, $b\varphi = b(Qx)\psi$ is a $\boldsymbol{\gamma}$-formula in Δ, and $\Delta \cup \{b\langle\psi\rangle_{x:=t}\} \in \mathcal{I}$ for some term $t \in \mathrm{TERM}_{\mathcal{L}}(V)$, then $\Delta \in \mathcal{I}$.
- If $\mathcal{L}$ contains $=$ and $\Delta \cup \{\mathbf{T}\alpha\} \in \mathcal{I}$, for some $\alpha \in \mathrm{INST}_{\mathcal{L},V}(\mathrm{Eq}_{=,\mathcal{L}})$, then $\Delta \in \mathcal{I}$.

(167) Let $\mathcal{L}$ be a first-order language, V be a set of variables and $\mathcal{C}$ be the collection of all satisfiable subsets Δ of $\mathrm{SFORM}_{\mathcal{L}^c}(V)$ such that Δ is a set with limited constant symbols. Prove that $\mathcal{C}$ is a $(\mathcal{L}^c, V)$-consistency property.

(168) Let $\mathcal{L}$ be a first-order language, V be an $\mathcal{L}$-suitable set of variables, $\mathcal{C}$ be an $(\mathcal{L}, V)$-consistency property of signed formulas, Δ be a member of $\mathcal{C}$ and Γ be $\{\varphi \mid b\varphi \in \Delta \text{ for some } b \in \mathbf{Bool}\}$.

Prove that Δ is satisfiable. In fact, show that there is an $(\mathcal{L}, V)$-Hintikka set Δ' such that $\Delta \subseteq \Delta'$ and

$$\Gamma' \subseteq \begin{cases} W^*_{\mathcal{L},V}(\Gamma) & \text{if } = \notin \mathcal{L} \\ W^*_{\mathcal{L},V}(\Gamma \cup \mathrm{INST}_{\mathcal{L},V}(\mathrm{Eq}_{=,\mathcal{L}})) & \text{if } = \in \mathcal{L}, \end{cases}$$

where $\Gamma' = \{\varphi \mid b\varphi \in \Delta' \text{ for some } b \in \mathbf{Bool}\}$ (a property referred to as the analyticity of the construction of Δ').

(169) Let $\mathcal{L}$ be a first-order language and V be a set of variables. Show that every satisfiable set of signed $(\mathcal{L}, V)$-formulas is contained in an $(\mathcal{L}^c, V)$-Hintikka set of signed formulas. Discuss the analyticity of this process.

(170) Formulate and prove the Compactness Theorem for Signed Formulas by translating the corresponding result for unsigned formulas as was required in Exercise 117 of Chapter 2, using the mapping $\mathtt{u}$ introduced before Exercise 160.

(171) Prove the Compactness Theorem for Signed Formulas along the lines of the argument presented in Example 4.12.28, using the notion of consistency property for sets of signed formulas.

Theories

(172) Let $\mathcal{L}$ be a first-order language and let $\mathfrak{A}$ be an $\mathcal{L}$-first-order property. Prove that for any $\mathcal{L}$-structures $\mathcal{A}, \mathcal{B}$, if $\mathcal{A} \in \mathfrak{A}$ and $\mathcal{A} \equiv \mathcal{B}$, then $\mathcal{B} \in \mathfrak{A}$.

(173) Give an example of a first-order language $\mathcal{L}$ and a class of $\mathcal{L}$-structures $\mathfrak{A}$ such that $\mathcal{A} \in \mathfrak{A}$ and $\mathcal{A} \equiv \mathcal{B}$ imply $\mathcal{B} \in \mathfrak{A}$, but $\mathfrak{A}$ is not an $\mathcal{L}$-first-order property.

Solution: Let $\mathcal{L} = \{=\}$ and $\mathfrak{A}$ be the class of all finite $\mathcal{L}$-structures. Suppose that $\mathcal{A} \in \mathfrak{A}$ and let k be the number of elements in the universe of $\mathcal{A}$. If θ_k is the formula introduced in Supplement 53, then $\theta_k \in \mathrm{Th}^{\mathcal{L}}(\mathcal{A})$. If $\mathcal{B} \equiv \mathcal{A}$, then $\theta_k \in \mathrm{Th}^{\mathcal{L}}(\mathcal{B})$, so $\mathcal{B}$ is also a finite $\mathcal{L}$-structure.

Corollary 4.10.46 shows that $\mathfrak{A}$ is not an $\mathcal{L}$-first-order property.

Decidability and Undecidability in First-Order Logic

An $\mathcal{L}$-theory Σ *admits elimination of quantifiers* if for every $\mathcal{L}$-formula φ there is a quantifier-free $\mathcal{L}$-formula ψ such that $\Sigma \models (\varphi \leftrightarrow \psi)$ and $\mathtt{FV}(\psi) \subseteq \mathtt{FV}(\varphi)$. (Note that if φ is an $\mathcal{L}$-sentence, then ψ is an $\mathcal{L}$-sentence.)

We say that Σ *admits an effective elimination of quantifiers* if there is an algorithm that allows us to obtain ψ from φ.

(174) Show that if $\mathcal{L}$-theory admits an effective elimination of quantifiers and there is an algorithm which, given a quantifier-free $\mathcal{L}$-sentence, decides the membership of the sentence in the theory, then the theory is *decidable*, that is, there is an algorithm that starting with an arbitrary $\mathcal{L}$-sentence decides its membership in the theory.

(175) Let $\mathcal{L}$ be a first-order language and Σ be a complete and semantically consistent $\mathcal{L}$-theory. Prove that if there exists an algorithm which, given a ground atomic $\mathcal{L}$-formula θ, determines whether or not $\theta \in \Sigma$, then there is an algorithm that for every quantifier-free $\mathcal{L}$-sentence φ, determines whether $\varphi \in \Sigma$.
Solution: Given φ, a quantifier-free $\mathcal{L}$-sentence, we can determine effectively a fundamental propositional form α for φ and an injective, atomic substitution s with $s(\alpha) = \varphi$, using the argument that is implicit in the proof of Theorem 4.8.7. Let v be the truth assignment given by

$$v(p) = \begin{cases} \mathbf{T} & \text{if } s(p) \in \Sigma \\ \mathbf{F} & \text{otherwise.} \end{cases}$$

By hypothesis, v can be effectively computed because there is an algorithm which allows to determine whether $s(p) \in \Sigma$.
We claim that $\varphi \in \Sigma$ if and only if $v(\alpha) = \mathbf{T}$, which would give the desired algorithm. To justify this claim, note that by Theorem 4.13.27, there is an $\mathcal{L}$-structure $\mathcal{A}$ such that $\Sigma = \mathrm{Th}^{\mathcal{L}}(\mathcal{A})$. Fix $\sigma \in \mathrm{ASSIGN}_{\mathcal{A}}$. We have $(\mathcal{A}, \sigma) \models s(p)$ if and only if $\mathcal{A} \models s(p)$ if and only if $s(p) \in \mathrm{Th}^{\mathcal{L}}(\mathcal{A}) = \Sigma$ for every propositional variable p because $s(p)$ is a ground atomic formula. By Lemma 4.8.11, we have $v(\varphi) = \mathbf{T}$ if and only if $(\mathcal{A}, \sigma) \models \varphi$ which is equivalent to $\mathcal{A} \models \varphi$ which, in turn, is equivalent to $\varphi \in \mathrm{Th}^{\mathcal{L}}(\mathcal{A}) = \Sigma$.

(176) Prove that an $\mathcal{L}$-theory Σ admits elimination of quantifiers if and only if for every $\mathcal{L}$-formula φ of the form $(\exists x)(\alpha_0 \wedge \cdots \wedge \alpha_{n-1})$, where $\alpha_0, \ldots, \alpha_{n-1}$ are literals and $n \geq 1$, there is a quantifier-free $\mathcal{L}$-formula ψ such that $\Sigma \models (\varphi \leftrightarrow \psi)$ and $\mathtt{FV}(\psi) \subseteq \mathtt{FV}(\varphi)$.

Solution: We will discuss only the nontrivial implication contained in the statement.

By the second part of Theorem 4.5.57, any $\mathcal{L}$-formula can be transformed into an equivalent $\mathcal{L}$-formula that does not contain universal quantifier symbols and has the same set of free variables. Thus, without loss of generality, we may assume that our formulas do not contain $\forall$.

We first show that if $\varphi = (\exists x)\theta$ with θ quantifier-free, then there is a quantifier-free $\mathcal{L}$-formula ψ such that $\Sigma \models (\varphi \leftrightarrow \psi)$ and $\mathtt{FV}(\psi) \subseteq \mathtt{FV}(\varphi)$. By Theorem 4.9.2, which asserts the existence of the disjunctive normal form, we can write $\theta \equiv \bigvee_{i=0}^{n-1} \bigwedge_{j=0}^{m_i-1} \beta_{ij}$, where each β_{ij} is a literal and $\mathtt{FV}(\theta) = \mathtt{FV}(\bigvee_{i=0}^{n-1} \bigwedge_{j=0}^{m_i-1} \beta_{ij})$. By Exercise 46, we have $\varphi = (\exists x)\theta \equiv \bigvee_{i=0}^{n-1} (\exists x) \bigwedge_{j=0}^{m_i-1} \beta_{ij}$ and these formulas have the same set of free variables. By the hypothesis of the supplement, for each formula $(\exists x) \bigwedge_{j=0}^{m_i-1} \beta_{ij}$, there is an equivalent quantifier-free formula γ_i such that $\mathtt{FV}(\gamma_i) \subseteq \mathtt{FV}\left((\exists x) \bigwedge_{j=0}^{m_i-1} \beta_{ij}\right)$. This allows to conclude that the quantifier-free formula $\bigvee_{i=0}^{n-1} \gamma_i$ is logically equivalent to φ and $\mathtt{FV}\left(\bigvee_{i=0}^{n-1} \gamma_i\right) \subseteq \mathtt{FV}(\varphi)$.

To prove the implication of the statement, we proceed by induction on φ. The basis step, when φ is atomic, is immediate. The only inductive step of interest is when $\varphi = (\exists x)\psi$ and by inductive hypothesis, ψ is equivalent to a quantifier-free $\mathcal{L}$-formula θ. Then, $\varphi \equiv (\exists x)\theta$ and by the previous argument, φ is equivalent to a quantifier-free $\mathcal{L}$-formula with no more free variables than φ.

(177) By examining the proof of Supplement 176, show that Σ is an $\mathcal{L}$-theory that admits an effective elimination of quantifiers if and only if there is an algorithm that given an $\mathcal{L}$-formula φ of the form $(\exists x)(\alpha_0 \wedge \cdots \wedge \alpha_{n-1})$, where $\alpha_0, \ldots, \alpha_{n-1}$ are literals and $n \geq 1$ produces a quantifier-free $\mathcal{L}$-formula ψ such that $\Sigma \models (\varphi \leftrightarrow \psi)$ and $\mathtt{FV}(\psi) \subseteq \mathtt{FV}(\varphi)$.

(178) Prove that $\mathrm{Th}^{\mathcal{L}_s}(\mathcal{A}_s)$, the theory of the $\mathcal{L}_s$-structure $\mathcal{A}_s$ of Example 4.4.39 admits an effective elimination of quantifiers.

Solution: By Supplement 176, we can restrict our attention to formulas of the form $(\exists x)(\alpha_0 \wedge \cdots \wedge \alpha_{n-1})$, where each α_i

is a literal. Note that the $\mathcal{L}_s$-atomic formulas have the form $s^p(u) = s^q(u')$, where $p, q \in \mathbf{N}$ and $u, u' \in \{0\} \cup \text{VAR}$.
Let $\varphi = (\exists x)(\alpha_0 \wedge \cdots \wedge \alpha_{n-1})$. If α_i has the form $s^p(x) = s^q(x)$, then

$$\alpha_i \equiv_{\mathcal{A}_s} \begin{cases} 0 = 0 & \text{if } p = q \\ 0 \neq 0 & \text{otherwise.} \end{cases}$$

Similarly, if α_i has the form $s^p(x) \neq s^q(x)$, then

$$\alpha_i \equiv_{\mathcal{A}_s} \begin{cases} 0 = 0 & \text{if } p \neq q \\ 0 \neq 0 & \text{otherwise.} \end{cases}$$

By Lemma 4.6.15, we may assume that none of the literals α_i has one of these forms.
If x does not occur in any of the literals $\alpha_0, \ldots, \alpha_{n-1}$, then $\varphi \equiv \psi$, where $\psi = (\alpha_0 \wedge \cdots \wedge \alpha_{n-1})$ is a quantifier-free formula. Therefore, we may assume that x occurs in at least one of the literals α_i and, when it occurs in a literal, it does not appear in both terms of the literal. By Exercise 43, we may assume that x occurs only in the first m literals, where $m \geq 1$. Then,

$$\varphi \equiv_{\mathcal{A}_s} \alpha_m \wedge \cdots \wedge \alpha_{n-1} \wedge (\exists x)(\alpha_0 \wedge \cdots \wedge \alpha_{m-1})$$

by Part 3 of Lemma 4.9.7. Thus, we can limit our discussion to formulas of the form $(\exists x)(\alpha_0 \wedge \cdots \wedge \alpha_{n-1})$, where each of the literals α_i contains x in one term but not both. More specifically, we can assume that our formula φ has the form

$$\begin{aligned} &(\exists x)(s^{p_0}(x) = s^{q_0}(u_0) \wedge \cdots \wedge s^{p_{l-1}}(x) = s^{q_{l-1}}(u_{l-1}) \\ &\wedge s^{p_l}(x) \neq s^{q_l}(u_l) \wedge \cdots \wedge s^{p_{n-1}}(x) \neq s^{q_{n-1}}(u_{n-1})), \end{aligned}$$

where $u_0, \ldots, u_{n-1} \in \{0\} \cup (\text{VAR} - \{x\})$. When $l = 0$, this formula states that for any assignment $\sigma \in \text{ASSIGN}_{\mathcal{A}_s}$, there is a number a with $a + p_i \neq \sigma^{\mathcal{A}_s}(s^{q_i}(u_i))$ for $0 \leq i \leq n-1$. Since this is always true, in this case, $\varphi \equiv_{\mathcal{A}_s} 0 = 0$.
When $l > 0$, we proceed as follows. Suppose that $(\mathcal{A}_s, \sigma) \models \varphi$. Then $q_0 + \sigma^{\mathcal{A}_s}(u_0) = p_0 + \sigma^{\mathcal{A}_s}(x) \geq p_0$. This implies that $q_0 + \sigma^{\mathcal{A}_s}(u_0) \notin \{0, \ldots, p_0 - 1\}$, which, in turn, yields

$$\begin{aligned} (\mathcal{A}_s, \sigma) \models (s^{q_0}(u_0) \neq 0 \wedge s^{q_0}(u_0) \neq s(0) \wedge \cdots \wedge s^{q_0}(u_0) \\ \neq s^{p_0 - 1}(0)). \end{aligned}$$

Further, since $p_r + \sigma^{\mathcal{A}_s}(x) = q_r + \sigma^{\mathcal{A}_s}(u_r)$ for $1 \leq r \leq l-1$, we have $\sigma^{\mathcal{A}_s}(s^{p_r+q_0}(u_0)) = p_r + q_0 + \sigma^{\mathcal{A}_s}(u_0) = p_r + p_0 + \sigma^{\mathcal{A}_s}(x) = p_0 + q_r + \sigma^{\mathcal{A}_s}(u_r) = \sigma^{\mathcal{A}_s}(s^{p_0+q_r}(u_r))$. Similarly, we have $\sigma^{\mathcal{A}_s}(s^{p_r+q_0}(u_0)) \neq \sigma^{\mathcal{A}_s}(s^{p_0+q_r}(u_r))$ for $l \leq r \leq n-1$. It follows that $(\mathcal{A}_s, \sigma) \models \psi$, where ψ is the quantifier-free formula

$$\begin{aligned}&(s^{q_0}(u_0) \neq 0 \wedge s^{q_0}(u_0) \neq s(0) \wedge \cdots \wedge s^{q_0}(u_0) \neq s^{p_0-1}(0)\\ &\wedge s^{p_1+q_0}(u_0) = s^{p_0+q_1}(u_1) \wedge \cdots \wedge s^{p_{l-1}+q_0}(u_0) = s^{p_0+q_{l-1}}(u_{l-1})\\ &\wedge s^{p_l+q_0}(u_0) \neq s^{p_0+q_l}(u_l) \wedge \cdots \wedge s^{p_{n-1}+q_0}(u_0) \neq s^{p_0+q_{n-1}}(u_{n-1})).\end{aligned}$$

Conversely, suppose that $(\mathcal{A}_s, \sigma) \models \psi$. Clearly this implies $(\mathcal{A}_s, \sigma) \models (s^{q_0}(u_0) \neq 0 \wedge s^{q_0}(u_0) \neq s(0) \wedge \cdots \wedge s^{q_0}(u_0) \neq s^{p_0-1}(0))$ which gives $q_0 + \sigma^{\mathcal{A}_s}(u_0) \geq p_0$. Thus, the number $a = q_0 + \sigma^{\mathcal{A}_s}(u_0) - p_0$ is nonnegative. We leave it to the reader to verify that $(\mathcal{A}_s, [x \to a]\sigma) \models (\alpha_0 \wedge \cdots \wedge \alpha_{n-1})$ and hence $(\mathcal{A}_s, \sigma) \models \varphi$. Also left to the reader is the verification that $\mathtt{FV}(\psi) \subseteq \mathtt{FV}(\varphi)$ and that the process of obtaining ψ from φ is effective.

(179) Formulate an algorithm that determines for each ground $\mathcal{L}_s$-atomic formula the membership of the formula in $\mathrm{Th}^{\mathcal{L}_s}(\mathcal{A}_s)$. Conclude that $\mathrm{Th}^{\mathcal{L}_s}(\mathcal{A}_s)$ is a decidable theory.
Hint. For the conclusion of the exercise, use Supplements 178, 175, and 174.

(180) Prove that $\mathrm{Th}^{\mathcal{L}_{s,<}}(\mathcal{A}_{s,<})$, the theory of the $\mathcal{L}_{s,<}$-structure $\mathcal{A}_{s,<}$ of Example 4.4.39 admits an effective elimination of quantifiers.
Solution: As in Supplement 178, we limit the discussion to formulas of the form $(\exists x)(\alpha_0 \wedge \cdots \wedge \alpha_{n-1})$, where each α_i is a literal. In this case however, the $\mathcal{L}_{s,<}$-atomic formulas have the form $s^p(u) = s^q(u')$ or $s^p(u) < s^q(u')$, where $p, q \in \mathbf{N}$ and $u, u' \in \{0\} \cup \mathrm{VAR}$. We may assume without loss of generality that for none of the pairs of terms u, u' do we have $u = u' = x$. Indeed, literals of the form $s^p(x) = s^q(x)$ or $s^p(x) \neq s^q(x)$ can be replaced as in Supplement 178 by ground atomic formulas. Further, literals of the form $s^p(x) < s^q(x)$ can be replaced by $0 = 0$ if $p < q$ and by $0 \neq 0$, otherwise. Literals of the form $(\neg(s^p(x) < s^q(x)))$ can be treated in a similar manner. By an argument similar to the one of Supplement 178, we can assume

that each of the literals α_i contains x in one term but not in both.
A literal of the form $t_0 \neq t_1$ can be replaced by the formula $(t_0 < t_1) \vee (t_1 < t_0)$. After each such replacement, using the distributivity of conjunction over disjunction and first part of Corollary 4.5.56, we can reduce the quantifier elimination for $(\exists x)(\alpha_0 \wedge \cdots \wedge (t_0 \neq t_1) \wedge \cdots \wedge \alpha_{n-1})$ to quantifier elimination for the formulas $(\exists x)(\alpha_0 \wedge \cdots \wedge (t_0 < t_1) \wedge \cdots \wedge \alpha_{n-1})$ and $(\exists x)(\alpha_0 \wedge \cdots \wedge (t_1 < t_0) \wedge \cdots \wedge \alpha_{n-1})$. Similarly, a literal of the form $(\neg(t_0 < t_1))$ can be replaced by $(t_0 = t_1) \vee (t_1 < t_0)$. Thus, we can assume that all literals α_i are atomic formulas that contain x in one term but not the other. Specifically, we can assume that our formula φ has the form $(\exists x)\theta$, where θ is the formula

$$\begin{aligned}
&(s^{p_0}(x) = s^{q_0}(u_0) \wedge \cdots \wedge s^{p_{l-1}}(x) = s^{q_{l-1}}(u_{l-1}) \\
&\wedge s^{p_l}(x) < s^{q_l}(u_l) \wedge \cdots \wedge s^{p_{m-1}}(x) < s^{q_{m-1}}(u_{m-1})) \\
&\wedge s^{p_m}(u_m) < s^{q_m}(x) \wedge \cdots \wedge s^{p_{n-1}}(u_{n-1}) < s^{q_{n-1}}(x)),
\end{aligned}$$

where $u_0, \ldots, u_{n-1} \in \{0\} \cup (\mathrm{VAR} - \{x\})$.
If $l > 0$, we can proceed along similar lines to the argument of Supplement 178. If $l = m = 0$, then $\varphi \equiv_{\mathcal{A}_{s,<}} 0 = 0$ because $(\mathcal{A}_{s,<}, \sigma) \models \varphi$ if and only if there is $a \in \mathbf{N}$ such that $p_r + \sigma^{\mathcal{A}_s}(u_r) < q_r + a$ for all r, $0 \leq r \leq n-1$. If $l = 0$ and $m = n$, then φ can be replaced by $((s^{p_0}(0) < s^{q_0}(u_0)) \wedge \cdots \wedge (s^{p_{n-1}}(0) < s^{q_{n-1}}(u_{n-1})))$. Finally, suppose that $l = 0$ and $0 < m < n$. In this case, $(\mathcal{A}_{s,<}, \sigma) \models \varphi$ if and only if the inequalities

$$\begin{aligned}
p_i + \sigma^{\mathcal{A}_{s,<}}(x) &< q_i + \sigma^{\mathcal{A}_{s,<}}(u_i) \\
p_j + \sigma^{\mathcal{A}_{s,<}}(u_j) &< q_j + \sigma^{\mathcal{A}_{s,<}}(x)
\end{aligned}$$

for $0 \leq i \leq m-1$ and $m \leq j \leq n-1$. These inequalities imply

$$1 + p_i + p_j + \sigma^{\mathcal{A}_{s,<}}(u_j) < q_i + q_j + \sigma^{\mathcal{A}_{s,<}}(u_i) \quad (4.26)$$

$$p_i < q_i + \sigma^{\mathcal{A}_{s,<}}(u_i) \quad (4.27)$$

for $0 \leq i \leq m-1$ and $m \leq j \leq n-1$. This last set of inequalities amounts to $(\mathcal{A}_{s,<}, \sigma) \models \psi$, where ψ is the quantifier-free formula

$$\bigwedge_{i=0}^{m-1}\bigwedge_{j=m}^{n-1}(s^{p_i+p_j+1}(u_j) < s^{q_i+q_j}(u_i)) \wedge \bigwedge_{i=0}^{m-1}(s^{p_i}(0) < s^{q_i}(u_i)).$$

Conversely, suppose that $(\mathcal{A}_{s,<}, \sigma) \models \psi$. The inequalities (4.26) imply

$$1 + \max_j\{\sigma^{\mathcal{A}_{s,<}}(u_j) + p_j - q_j\} < \min_i\{\sigma^{\mathcal{A}_{s,<}}(u_i) + q_i - p_i\}.$$

In addition, the inequalities (4.27) imply that $\min_i\{\sigma^{\mathcal{A}_{s,<}}(u_i) + q_i - p_i\} > 0$. Thus, the number $a = \min_i\{\sigma^{\mathcal{A}_{s,<}}(u_i) + q_i - p_i\} - 1$ is a natural number and we leave it to the reader to verify that $(\mathcal{A}_{s,<}, [x \to a]\sigma) \models \theta$, so $(\mathcal{A}_{s,<}, \sigma) \models \varphi$.

(181) Formulate an algorithm that determines for each ground $\mathcal{L}_{s,<}$-atomic formula the membership of the formula in $\mathrm{Th}^{\mathcal{L}_{s,<}}(\mathcal{A}_{s,<})$. Conclude that $\mathrm{Th}^{\mathcal{L}_{s,<}}(\mathcal{A}_{s,<})$ is a decidable theory.

(182) Let $\mathcal{L} \subseteq \mathcal{L}'$ be two first-order languages and let $\mathcal{A}'$ be an $\mathcal{L}'$-structure. Prove that if $\mathrm{Th}^{\mathcal{L}'}(\mathcal{A}')$ is decidable, then $\mathrm{Th}^{\mathcal{L}}(\mathrm{RED}_{\mathcal{L}}(\mathcal{A}'))$ is also decidable.

Let $\mathcal{L}_{apra}$ be the language $\mathcal{A}_{pra} \cup \{\sim_2, \sim_3, \ldots\}$, where $\sim_i$ is a binary relation symbol. We refer to $\mathcal{L}_{apra}$ as the augmented language of Presburger arithmetic. In the sequel, we will use the $\mathcal{L}_{apra}$-structure $\mathcal{A}_{apra}$ which is the extension of $\mathcal{A}_{pra}$ where $\sim_i^{\mathcal{A}_{apra}}$ is congruence modulo i on $\mathbf{N}$ for $i \geq 2$.

(183) Let $t \in \mathrm{TERM}_{\mathcal{L}_{apra}}$ and assume that the variables that occur in t in standard order are $y_0, \ldots, y_{k-1}$. Prove that $t \equiv_{\mathcal{A}_{apra}} s^j(0) + n_0y_0 + \cdots + n_{k-1}y_{k-1}$ for some $j, n_0, \ldots, n_{k-1} \in \mathbf{N}$ and there is an effective way to determine the numbers $j, n_0, \ldots, n_{k-1}$ starting from t.
Hint. This exercise follows immediately from Exercise 77.

(184) Prove that $\mathrm{Th}^{\mathcal{L}_{apra}}(\mathcal{A}_{apra})$ admits an effective elimination of quantifiers.
Solution: The atomic formulas of $\mathcal{L}_{apra}$ have one of the forms $t = u$, $t < u$, or $t \sim_i u$, for some $i \geq 2$. By Supplement 176, we can limit the discussion to $\mathcal{L}_{apra}$-formulas of the form $\varphi = (\exists x)(\alpha_0 \wedge \cdots \wedge \alpha_{n-1})$, where $\alpha_0, \ldots, \alpha_{n-1}$ are literals and $n \geq 1$.

Our first step will be to eliminate negated literals. For literals of the form $t_0 \neq t_1$ and $(\neg(t_0 < t_1))$, we use the technique shown in the argument of Supplement 180. Negated literals of the form $(\neg(t_0 \sim_i t_1))$ are replaced by

$$t_0 \sim_i (t_1 + s(0)) \vee t_0 \sim_i (t_1 + s^2(0)) \vee \cdots \vee t_0 \sim_i (t_1 + s^{i-1}(0)).$$

Thus, by the above discussion and Exercise 183, we may assume that each α_j is an atomic formula having one of the following forms:

$$\begin{aligned}
&t + qx = u + mx \\
&t + qx \sim_i u + mx \\
&t + mx < u + qx \\
&t + qx < u + mx,
\end{aligned}$$

where $q \geq m$ and t, u are terms that do not contain x. (We may have to add "dummy" terms $0x$ to obtain these standardized forms.) These formulas are $\mathcal{A}_{apra}$-equivalent to:

$$\begin{aligned}
&t + (q - m)x = u \\
&t + (q - m)x \sim_i u \\
&t < u + (q - m)x \\
&t + (q - m)x < u,
\end{aligned}$$

respectively. When $q = m$, the corresponding formulas can be "pulled out", which leaves us with formulas α_j of the forms:

$$\begin{aligned}
&t_j + p_j x = u_j \\
&t_j + p_j x \sim_{i_j} u_j \\
&t_j < u_j + p_j x \\
&t_j + p_j x < u_j,
\end{aligned}$$

where each p_j is positive. Let $P = \operatorname{lcm}\{p_0, \ldots, p_{n-1}\}$ and let $r_j = P/p_j$. Multiplying each formula by r_j, we can assume that the formulas α_j have one of the following forms:

$$\begin{aligned}
&t'_j + Px = u'_j\\
&t'_j + Px \sim_{r_j i_j} u'_j\\
&t'_j < u'_j + Px\\
&t'_j + Px < u'_j.
\end{aligned}$$

Let z be a variable that does not occur in φ. In view of the previous assumption involving the formulas α_j, the formula φ is equivalent to the formula

$$(\exists z)(z \sim_P 0 \wedge \alpha_0 \wedge \cdots \wedge \alpha_{n-1}),$$

where each α_j has one of the following forms:

$$\begin{aligned}
&t'_j + z = u'_j\\
&t'_j + z \sim_{r_j i_j} u'_j\\
&t'_j < u'_j + z\\
&t'_j + z < u'_j,
\end{aligned}$$

where neither t'_j nor u'_j contain z. If one of the formulas α_j is of the first form, we may assume that $j = 0$. In this case, the quantifier-free formula ψ that is $\mathcal{A}_{apra}$-equivalent to φ is $(\alpha'_1 \wedge \cdots \alpha'_{n-1} \wedge ((t'_0 < u'_0) \vee (t'_0 = u'_0)) \wedge (u'_0 \sim_P t'_0))$, where the formulas α'_j are determined according to the following table:

α_j	α'_j
$t'_j + z = u'_j$	$t'_j + u'_0 = u'_j + t'_0$
$t'_j + z \sim_{r_j i_j} u'_j$	$t'_j + u'_0 \sim_{r_j i_j} u'_j + t'_0$
$t'_j < u'_j + z$	$t'_j + t'_0 < u'_j + u'_0$
$t'_j + z < u'_j$	$t'_j + u'_0 < u'_j + t'_0$,

for $1 \leq j \leq n-1$. Thus, we can limit ourselves to formulas that do not contain the equality symbol. In other words, we need to show that the quantifier can be eliminated from a formula

$\varphi = (\exists x)\theta$, where θ is the formula:

$$\begin{aligned}&(t_0 + x \sim_{i_0} u_0 \wedge \cdots \wedge t_{l-1} + x \sim_{i_{l-1}} u_{l-1}\\ &\wedge t_l + x < u_l \wedge \cdots \wedge t_{m-1} + x < u_{m-1}\\ &\wedge t_m < u_m + x \wedge \cdots \wedge t_{n-1} < u_{n-1} + x).\end{aligned}$$

Note that by our construction, we have $l > 0$ and none of the terms t_i, u_i contain x. Since $|\mathcal{A}_{apra}| = \mathbf{N}$, we can add the conjunct $t_n < u_n + x$, where $t_n = 0$ and $u_n = s(0)$, which simply states that x is nonnegative. Let θ' be the formula obtained from θ by this addition. We denote the formula $(\exists x)\theta'$ by φ'. Let ℓ be the least common multiple of the numbers $i_0, \ldots, i_{l-1}$ and let ψ be the formula

$$\begin{aligned}&\bigvee_{m\leq j\leq n} \bigvee_{0<q\leq \ell} \Big(\bigwedge_{m\leq k\leq n} (t_k + u_j < u_k + t_j + s^q(0)) \wedge\\ &\bigwedge_{l\leq k<m} (t_k + t_j + s^q(0) < u_k + u_j) \wedge\\ &\bigwedge_{0\leq k<l} (t_k + t_j + s^q(0) \sim_{i_k} u_j + u_k) \Big).\end{aligned}$$

To prove that $\mathrm{Th}^{\mathcal{L}_{apra}}(\mathcal{A}_{apra})$ admits elimination of quantifiers, it suffices to show that $(\mathcal{A}_{apra}, \sigma) \models \varphi'$ if and only if $(\mathcal{A}_{apra}, \sigma) \models \psi$, for every $\sigma \in \mathrm{ASSIGN}_{\mathcal{A}_{apra}}$.
Suppose that $(\mathcal{A}_{apra}, \sigma) \models \varphi'$. This implies that there is a number $a \in \mathbf{N}$ such that

$$\begin{aligned}&a \equiv_{i_0} \sigma^{\mathcal{A}_{apra}}(u_0) - \sigma^{\mathcal{A}_{apra}}(t_0)\\ &\vdots\\ &a \equiv_{i_{l-1}} \sigma^{\mathcal{A}_{apra}}(u_{l-1}) - \sigma^{\mathcal{A}_{apra}}(t_{l-1})\\ &a < \sigma^{\mathcal{A}_{apra}}(u_l) - \sigma^{\mathcal{A}_{apra}}(t_l)\\ &\vdots\\ &a < \sigma^{\mathcal{A}_{apra}}(u_{m-1}) - \sigma^{\mathcal{A}_{apra}}(t_{m-1})\\ &a > \sigma^{\mathcal{A}_{apra}}(t_m) - \sigma^{\mathcal{A}_{apra}}(u_m)\\ &\vdots\\ &a > \sigma^{\mathcal{A}_{apra}}(t_n) - \sigma^{\mathcal{A}_{apra}}(u_n) = -1\end{aligned}$$

Let a_0 be the least such a. We claim that there is a j such that $m \leq j \leq n$ and $\sigma^{\mathcal{A}_{apra}}(t_j) - \sigma^{\mathcal{A}_{apra}}(u_j) < a_0 \leq$

$\sigma^{\mathcal{A}_{apra}}(t_j) - \sigma^{\mathcal{A}_{apra}}(u_j) + \ell$. Suppose this is not the case. Then, $a > \sigma^{\mathcal{A}_{apra}}(t_j) - \sigma^{\mathcal{A}_{apra}}(u_j) + \ell$ for every j. Let $a' = a_0 - \ell$. Note that we still have $a' \equiv_{i_p} \sigma^{\mathcal{A}_{apra}}(u_p) - \sigma^{\mathcal{A}_{apra}}(t_p)$ for $0 \leq p \leq l-1$ because ℓ is a multiple of i_p and the upper and lower bounds also hold for a'. This contradicts the minimality of a_0, so a_0 belongs to some interval

$$(\sigma^{\mathcal{A}_{apra}}(t_j) - \sigma^{\mathcal{A}_{apra}}(u_j), \sigma^{\mathcal{A}_{apra}}(t_j) - \sigma^{\mathcal{A}_{apra}}(u_j) + \ell].$$

This allows us to write $a_0 = \sigma^{\mathcal{A}_{apra}}(t_j) - \sigma^{\mathcal{A}_{apra}}(u_j) + q$ for some q, where $0 < q \leq \ell$. For this j and q, the inequality $a_0 > \sigma^{\mathcal{A}_{apra}}(t_k) - \sigma^{\mathcal{A}_{apra}}(u_k)$ translates into $\sigma^{\mathcal{A}_{apra}}(t_k) + \sigma^{\mathcal{A}_{apra}}(u_j) < \sigma^{\mathcal{A}_{apra}}(u_k) + \sigma^{\mathcal{A}_{apra}}(t_j) + q$, which shows that $(\mathcal{A}_{apra}, \sigma) \models t_k + u_j < u_k + t_j + s^q(0)$. Similarly, we can prove that $(\mathcal{A}_{apra}, \sigma)$ satisfies the remaining conjuncts of ψ.
Suppose now that $(\mathcal{A}_{apra}, \sigma) \models \psi$. Then, there are j and q with $m \leq j \leq n$ and $0 < q \leq \ell$ such that

$$\begin{aligned}
&\sigma^{\mathcal{A}_{apra}}(t_m) + \sigma^{\mathcal{A}_{apra}}(u_j) < \sigma^{\mathcal{A}_{apra}}(u_m) + \sigma^{\mathcal{A}_{apra}}(t_j) + q \\
&\vdots \\
&\sigma^{\mathcal{A}_{apra}}(t_n) + \sigma^{\mathcal{A}_{apra}}(u_j) < \sigma^{\mathcal{A}_{apra}}(u_n) + \sigma^{\mathcal{A}_{apra}}(t_j) + q \\
&\sigma^{\mathcal{A}_{apra}}(t_l) + \sigma^{\mathcal{A}_{apra}}(t_j) + q < \sigma^{\mathcal{A}_{apra}}(u_l) + \sigma^{\mathcal{A}_{apra}}(u_j) \\
&\vdots \\
&\sigma^{\mathcal{A}_{apra}}(t_{m-1}) + \sigma^{\mathcal{A}_{apra}}(t_j) + q < \sigma^{\mathcal{A}_{apra}}(u_{m-1}) + \sigma^{\mathcal{A}_{apra}}(u_j) \\
&\sigma^{\mathcal{A}_{apra}}(t_0) + \sigma^{\mathcal{A}_{apra}}(t_j) + q \equiv_{i_0} \sigma^{\mathcal{A}_{apra}}(u_j) + \sigma^{\mathcal{A}_{apra}}(u_0) \\
&\vdots \\
&\sigma^{\mathcal{A}_{apra}}(t_{l-1}) + \sigma^{\mathcal{A}_{apra}}(t_j) + q \equiv_{i_{l-1}} \sigma^{\mathcal{A}_{apra}}(u_j) + \sigma^{\mathcal{A}_{apra}}(u_{l-1})
\end{aligned}$$

Let $a = \sigma^{\mathcal{A}_{apra}}(t_j) - \sigma^{\mathcal{A}_{apra}}(u_j) + q$. Since $\sigma^{\mathcal{A}_{apra}}(t_n) + \sigma^{\mathcal{A}_{apra}}(u_j) < \sigma^{\mathcal{A}_{apra}}(u_n) + \sigma^{\mathcal{A}_{apra}}(t_j) + q$ and $t_n = 0$ and $u_n = s(0)$, we have $a \geq 0$. We claim that $(\mathcal{A}_{apra}, [x \to a]\sigma) \models \theta'$ and therefore, $(\mathcal{A}_{apra}, \sigma) \models \varphi'$. We need to prove that $(\mathcal{A}_{apra}, [x \to a]\sigma) \models \beta$ for every conjunct β of θ'.
We argue here the case of the conjuncts of the form $t_k + x \sim_{i_k} u_k$ and leave the remaining cases for the reader. Since $\sigma^{\mathcal{A}_{apra}}(t_k) + \sigma^{\mathcal{A}_{apra}}(t_j) + q \equiv_{i_k} \sigma^{\mathcal{A}_{apra}}(u_j) + \sigma^{\mathcal{A}_{apra}}(u_k)$, we have

$\sigma^{\mathcal{A}_{apra}}(t_k) + a \equiv_{i_k} \sigma^{\mathcal{A}_{apra}}(u_k)$, which amounts to $(\mathcal{A}_{apra}, [x \to a]\sigma) \models t_k + x \sim_{i_k} u_k$.
Also left to the reader is the verification that no new free variables were introduced when the formula ψ was defined and that the process of constructing the formula ψ is effective.

(185) Give an algorithm that determines for each ground $\mathcal{L}_{apra}$-atomic formula the membership of the formula in $\mathrm{Th}^{\mathcal{L}_{apra}}(\mathcal{A}_{apra})$. Conclude that $\mathrm{Th}^{\mathcal{L}_{apra}}(\mathcal{A}_{apra})$ is a decidable theory and thus $\mathrm{Th}^{\mathcal{L}_{pra}}(\mathcal{A}_{pra})$ is a decidable theory.
Hint. Use Exercises 182 and 181.

(186) Let $\mathcal{L}$ be a first-order language that is decidable, in other words, for which there is an algorithm which allows us to decide whether a symbol belongs to the language. Let $\mathcal{A} = (A, \mathcal{I})$ be a finite $\mathcal{L}$-structure such that $\mathcal{I}$ is an effectively computable function. Prove that $\mathrm{Th}^{\mathcal{L}}(\mathcal{A})$ is a decidable theory.
Hint. Adapt the proof of Theorem 4.14.3.

(187) Show that the instances

$$\Im = (\{a, b\}, (aba, bbab), (ab, baa))$$
$$\Im' = (\{a, b\}, (ab, aab), (bb, bab), (a, aa))$$

of the PCP have no solution.

(188) Find a solution of the instance

$$\Im = (\{a, b\}, (a, aaa), (abaaa, ab), (ab, b))$$

of the PCP.

(189) Show that it is undecidable whether an arbitrary formula of first-order logic is satisfiable.

Next, we present an abstract technique for reducing the PCP to the decision problem for the theory of certain structures.

Let $V = \{d, e\}$ be an alphabet, E_V be the set of even length sequences of words over V, and let $\mathcal{A}$ be an $\mathcal{L}$-structure, where $\mathcal{L}$ is a first-order language.

A *PCP-reduction scheme* $\mathcal{S}$ for $\mathcal{A}$ consists of

- an injection $\phi^{\mathcal{S}} : V^* \longrightarrow |\mathcal{A}|$;
- a function $\Phi^{\mathcal{S}} : E_V \longrightarrow |\mathcal{A}|$;
- an $\mathcal{L}$-formula $\varphi^{\mathcal{S}}_{z,\text{null}}$ with $\mathtt{FV}(\varphi^{\mathcal{S}}_{z,\text{null}}) = \{z\}$;

- for each $q \in V^*$, an $\mathcal{L}$-formula, $\varphi^{\mathcal{S}}_{z_0,z_1,q,\text{conc}}$ with $\texttt{FV}(\varphi^{\mathcal{S}}_{z_0,z_1,q,\text{conc}}) = \{z_0, z_1\}$;
- an $\mathcal{L}$-formula $\varphi^{\mathcal{S}}_{x',y_0,y_1,x,\text{pref}}$ with $\texttt{FV}(\varphi^{\mathcal{S}}_{x',y_0,y_1,x,\text{pref}}) = \{x', y_0, y_1, x\}$;
- and an $\mathcal{L}$-formula $\varphi^{\mathcal{S}}_{y',y'',\hat{x},\text{last-entry}}$ with

$$\texttt{FV}(\varphi^{\mathcal{S}}_{y',y'',\hat{x},\text{last-entry}}) = \{y', y'', \hat{x}\}.$$

The formulas $\varphi^{\mathcal{S}}_{z_0,z_1,q,\text{conc}}$ define a binary relation $\sqsubset^{\mathcal{S}}$ on $|\mathcal{A}|$, given by $a \sqsubset^{\mathcal{S}} b$ if there is $q \in V^* - \{\lambda\}$ such that $(\mathcal{A}, [z_0 \to b][z_1 \to a]) \models \varphi^{\mathcal{S}}_{z_0,z_1,q,\text{conc}}$.

For a V-instance of the PCP $\Im = (V, (q_0, \ldots, q_{n-1}), (r_0, \ldots, r_{n-1}))$ we define the formula $\varphi^{\mathcal{S},\Im}_{u,v,s,t,\text{one-step}}$ as

$$\bigvee_{0 \leq i \leq n-1} ((\varphi^{\mathcal{S}}_{z_0,z_1,q_i,\text{conc}})_{z_0,z_1:=u,s} \wedge (\varphi^{\mathcal{S}}_{z_0,z_1,r_i,\text{conc}})_{z_0,z_1:=v,t}).$$

Suppose that u, v, s, t correspond under ϕ to words u', v', s', t' over V, then $\varphi^{\mathcal{S},\Im}_{u,v,s,t,\text{one-step}}$ is meant to express that the pair (u', v') is obtained from (s', t') by adding one of the pairs of the instance as in $u' = s'q_i$ and $v' = t'r_i$.

Intuitively, the formula $\varphi^{\mathcal{S},\Im}_{x,\text{const}}$ holds when x encodes a construction sequence for $\Im$ and is given by:

$$\begin{aligned}&(\forall y_0)(\forall y_1)(\forall x')(\varphi^{\mathcal{S}}_{x',y_0,y_1,x,\text{pref}} \to ((\varphi^{\mathcal{S}}_{z,\text{null}})_{z:=y_0} \wedge (\varphi^{\mathcal{S}}_{z,\text{null}})_{z:=y_1})\\ &\vee(\exists y')(\exists y'')((\varphi^{\mathcal{S}}_{y',y'',\hat{x},\text{last-entry}})_{\hat{x}:=x'} \wedge (\varphi^{\mathcal{S},\Im}_{u,v,s,t,\text{one-step}})_{u,v,s,t:=y_0,y_1,y',y''}))\end{aligned}$$

Finally, the formula $\varphi^{\mathcal{S},\Im}_{\text{solv}}$ is given by:

$$(\exists x)(\exists y)(\varphi^{\mathcal{S},\Im}_{x,\text{const}} \wedge (\varphi^{\mathcal{S}}_{y',y'',\hat{x},\text{last-entry}})_{y',y'',\hat{x}:=y,y,x} \wedge (\neg(\varphi^{\mathcal{S}}_{z,\text{null}})_{z:=y}))$$

This last formula expresses the fact that $\Im$ is solvable.

For a PCP-reduction scheme $\mathcal{S}$ to $\mathcal{A}$, we introduce the axioms listed below which may or may not hold for $\mathcal{S}$:

(NULL) For every $a \in |\mathcal{A}|$, $(\mathcal{A}, [z \to a]) \models \varphi^{\mathcal{S}}_{z,\text{null}}$ if and only if $a = \phi^{\mathcal{S}}(\lambda)$.

(CONC) For every $a \in |\mathcal{A}|$ and for every $q, r \in \{d, e\}^*$, we have $a = \phi^{\mathcal{S}}(rq)$ if and only if

$$(\mathcal{A}, [z_0 \to a][z_1 \to \phi^{\mathcal{S}}(r)]) \models \varphi^{\mathcal{S}}_{z_0,z_1,q,\text{conc}}.$$

(INIT-OF) For every $a, b_0, b_1 \in |\mathcal{A}|$ and $\hat{q} \in E_V$, we have $(\mathcal{A}, [x' \to a][y_0 \to b_0][y_1 \to b_1][x \to \Phi^{\mathcal{S}}(\hat{q})]) \models \varphi^{\mathcal{S}}_{x',y_0,y_1,x,\text{pref}}$ if and only if $a = \Phi^{\mathcal{S}}(\hat{r})$, $b_0 = \phi^{\mathcal{S}}(q_0)$ and $b_1 = \phi^{\mathcal{S}}(q_1)$ for some $\hat{r} \in E_V$, $q_0, q_1 \in V^*$ such that $\hat{r}(q_0, q_1)$ is a prefix of $\hat{q}$.

(LE) If $a = \phi^{\mathcal{S}}(q_0)$, $b = \phi^{\mathcal{S}}(q_1)$, where q_0, q_1 are the last entries of the sequence $\hat{q}$ in E_V, then

$$(\mathcal{A}, [y' \to a][y'' \to b][\hat{x} \to \Phi^{\mathcal{S}}(\hat{q})]) \models \varphi^{\mathcal{S}}_{y',y'',\hat{x},\text{last-entry}}.$$

(WF) There is no infinite sequence of elements $a_0, a_1, a_2, \ldots$ of $|\mathcal{A}|$ such that $\cdots \sqsubset^{\mathcal{S}} a_2 \sqsubset^{\mathcal{S}} a_1 \sqsubset^{\mathcal{S}} a_0$ (Well-foundedness axiom).

(LI) We have:

$$\mathcal{A} \models (\forall y')(\forall y'')(\forall \hat{x})(\varphi^{\mathcal{S}}_{y',y'',\hat{x},\text{last-entry}} \to (\exists x')(\varphi^{\mathcal{S}}_{x',y_0,y_1,x,\text{pref}})_{y_0,y_1,x:=y',y'',\hat{x}}).$$

(ILI) We have:

$$\begin{aligned}\mathcal{A} \models\ & (\forall x')(\forall y_0)(\forall y_1)(\forall x)(\forall y_2)(\forall y_3)\\ & ((\varphi^{\mathcal{S}}_{x',y_0,y_1,x,\text{pref}} \wedge (\varphi^{\mathcal{S}}_{y',y'',\hat{x},\text{last-entry}})_{y',y'',\hat{x}:=y_2,y_3,x'}) \to\\ & (\exists x'')(\varphi^{\mathcal{S}}_{x',y_0,y_1,x,\text{pref}})_{x',y_0,y_1:=x'',y_2,y_3})\end{aligned}$$

(190) Let $\mathcal{L}$ be a first-order language that consists of a constant symbol c, two unary function symbols f_d, f_e, a ternary function symbol g, the equality symbol $=$, and the binary relation symbol $\leq$. Let $\mathcal{A}$ be the $\mathcal{L}$-Herbrand structure, where $\leq^{\mathcal{A}}$ is the subterm relation on $\text{GTERM}_{\mathcal{L}} = |\mathcal{A}|$.
We define a PCP-reduction scheme $\mathcal{S}$ to $\mathcal{A}$. Let $\phi^{\mathcal{S}} : \{d, e\}^* \longrightarrow \text{GTERM}_{\mathcal{L}}$ be given by $\phi^{\mathcal{S}}(s_0 \cdots s_{n-1}) = f_{s_{n-1}}(\cdots (f_{s_0}(c)) \cdots)$ for every $s_0 \cdots s_{n-1} \in \{d, e\}^*$. The function $\Phi^{\mathcal{S}} : E_{\{d,e\}} \longrightarrow \text{GTERM}_{\mathcal{L}}$ is defined recursively by $\Phi^{\mathcal{S}}(\lambda) = c$ and

$$\begin{aligned}&\Phi^{\mathcal{S}}(u_0, v_0, \ldots, u_{i-1}, v_{i-1}, u_i, v_i)\\ &\quad = g(\Phi^{\mathcal{S}}(u_0, v_0, \ldots, u_{i-1}, v_{i-1}), \phi(u_i), \phi(v_i)),\end{aligned}$$

for every $(u_0, v_0, \ldots, u_i, v_i) \in E_{\{d,e\}}$. Consider the definitions of $\mathcal{L}$-formulas shown below:

Formula	Definition
$\varphi^{\mathcal{S}}_{z,\text{null}}$	$z = c$
$\varphi^{\mathcal{S}}_{z_0,z_1,q,\text{conc}}$	$z_0 = f_{q_{n-1}}(\cdots(f_{q_0}(z_1))\cdots)$
$\varphi^{\mathcal{S}}_{x',y_0,y_1,x,\text{pref}}$	$g(x', y_0, y_1) \leq x$
$\varphi^{\mathcal{S}}_{y',y'',\hat{x},\text{last-entry}}$	$(\exists \check{x})(g(\check{x}, y', y'') = \hat{x})$

for $q = q_0 \cdots q_{n-1}$. Prove that:

(a) $\phi^{\mathcal{S}}$ is injective;

(b) the axioms listed above are all satisfied by $\mathcal{S}$.

(191) Let $\mathcal{A}$ be an $\mathcal{L}$-structure and $\mathcal{S}$ be a PCP-reduction scheme for $\mathcal{A}$ and let $\Im = (\{d, e\}, (s_0, \ldots, s_{n-1}), (t_0, \ldots, t_{n-1}))$ be an instance of the PCP. Prove that if $\mathcal{S}$ satisfies the axioms **(NULL), (CONC), (INIT-OF)** and **(LE)**, then, for any construction sequence for $\Im$, $\hat{q}$, $a = \Phi^{\mathcal{S}}(\hat{q})$ implies $(\mathcal{A}, [x \to a]) \models \varphi^{\mathcal{S},\Im}_{x,\text{const}}$.

Solution: Suppose that

$$(\mathcal{A}, [x' \to b][y_0 \to c][y_1 \to c'][x \to a]) \models \varphi^{\mathcal{S}}_{x',y_0,y_1,x,\text{pref}}.$$

By the axiom **(INIT-OF)**, there are $\hat{r} \in E_V$ and $q_0, q_1 \in V^*$ such that $\hat{r}(q_0, q_1)$ is a prefix of $\hat{q}$ and $b = \Phi^{\mathcal{S}}(\hat{r})$, $c = \phi^{\mathcal{S}}(q_0)$ and $c' = \phi^{\mathcal{S}}(q_1)$.Thus, we have either $q_0 = q_1 = \lambda$ or there are $\tilde{r} \in E_V$ and $q_0', q_1' \in V^*$ such that $\hat{r} = \tilde{r}(q_0', q_1')$ and $q_0 = q_0' s_j, q_1 = q_1' t_j$ for some j. The latter case means that (q_0, q_1) is obtained in one step from (q_0', q_1').

In the first case, by the axiom **(NULL)**, we have $(\mathcal{A}, [z \to c]) \models \varphi^{\mathcal{S}}_{z,\text{null}}$ and $(\mathcal{A}, [z \to c']) \models \varphi^{\mathcal{S}}_{z,\text{null}}$. In the second case, by axiom **(LE)**, we have

$$(\mathcal{A}, [y' \to \phi^{\mathcal{S}}(q_0')][y'' \to \phi^{\mathcal{S}}(q_1')][\hat{x} \to b]) \models \varphi^{\mathcal{S}}_{y',y'',\hat{x},\text{last-entry}}$$

and, by axiom **(CONC)**

$$(\mathcal{A}, [z_0 \to \phi^{\mathcal{S}}(q_0)][z_1 \to \phi^{\mathcal{S}}(q_0')]) \models \varphi^{\mathcal{S}}_{z_0,z_1,s_j,\text{conc}}$$

$$(\mathcal{A}, [z_0 \to \phi^{\mathcal{S}}(q_1)][z_1 \to \phi^{\mathcal{S}}(q_1')]) \models \varphi^{\mathcal{S}}_{z_0,z_1,t_j,\text{conc}},$$

which yields

$$(\mathcal{A},[u\to\phi^{\mathcal{S}}(q_0)][v\to\phi^{\mathcal{S}}(q_1)][s\to\phi^{\mathcal{S}}(q_0')][t\to\phi^{\mathcal{S}}(q_1')])\models \varphi^{\mathcal{S},\Im}_{u,v,s,t,\text{one-step}},$$

which proves the desired implication.

(192) Let $\Im=(\{d,e\},(s_0,\dots,s_{n-1}),(t_0,\dots,t_{n-1}))$ be an instance of the PCP. Prove that if $\mathcal{S}$ is a PCP-reduction to $\mathcal{A}$ that satisfies the axioms **(NULL), (CONC), (INIT-OF)** and **(LE)**, then the existence of a solution for $\Im$ implies $\mathcal{A}\models\varphi^{\mathcal{S},\Im}_{\text{solv}}$.
Solution: Suppose that $\hat{q}=(\cdots,q,q)$ is a construction sequence for $\Im$, where $q\neq\lambda$. Let $a=\Phi^{\mathcal{S}}(\hat{q})$ and $b=\phi^{\mathcal{S}}(q)$. Then, by Supplement 191, we have $(\mathcal{A},[x\to a])\models\varphi^{\mathcal{S},\Im}_{x,\text{const}}$. Further, by axiom **(LE)**, we have $(\mathcal{A},[y'\to b][y''\to b][\hat{x}\to a])\models\varphi^{\mathcal{S}}_{y',y'',\hat{x},\text{last-entry}}$. Finally, by axiom **(NULL)**, we have $(\mathcal{A},[z\to b])\models(\neg\varphi^{\mathcal{S}}_{z,\text{null}})$. All these imply $\mathcal{A}\models\varphi^{\mathcal{S},\Im}_{\text{solv}}$.

(193) Let $\mathcal{A}$ be an $\mathcal{L}$-structure, $\mathcal{S}$ be a PCP-reduction scheme to $\mathcal{A}$ and let $\Im=(\{d,e\},(s_0,\dots,s_{n-1}),(t_0,\dots,t_{n-1}))$ be an instance of the PCP. Prove that if $\mathcal{A}$ satisfies the axioms **(WF), (NULL), (ILI)**, and **(CONC)**, then, if $a,b,b_0,b_1\in|\mathcal{A}|$, $(\mathcal{A},[x\to a])\models\varphi^{\mathcal{S},\Im}_{x,\text{const}}$ and $(\mathcal{A},[x'\to b][y_0\to b_0][y_1\to b_1][x\to a])\models\varphi^{\mathcal{S}}_{x',y_0,y_1,x,\text{pref}}$ then there are $s,s'\in V^*$ such that $\phi^{\mathcal{S}}(s)=b_0$ and $\phi^{\mathcal{S}}(s')=b_1$ and there is a construction sequence for $\Im$ whose last members are s and s'.
Solution: Recall that we introduced the relation $\sqsubset^{\mathcal{S}}$ on $|\mathcal{A}|$ with the help of the formulas $\varphi^{\mathcal{S}}_{z_0,z_1,q,\text{conc}}$. This relation is extended to $|\mathcal{A}|^2$ by writing $(c_0,c_1)\sqsubset^{\mathcal{S}}(b_0,b_1)$ if and only if $c_0\sqsubset^{\mathcal{S}}b_0$ and $c_1\sqsubset^{\mathcal{S}}b_1$. Note that $(|\mathcal{A}|^2,\sqsubset^{\mathcal{S}})$ is well-founded by Axiom **(WF)**. We will prove the statement by well-founded induction on (b_0,b_1). Suppose that the statement is true for all pairs (c_0,c_1) such that $(c_0,c_1)\sqsubset^{\mathcal{S}}(b_0,b_1)$ and that $(\mathcal{A},[x\to a])\models\varphi^{\mathcal{S},\Im}_{x,\text{const}}$ and $(\mathcal{A},[x'\to b][y_0\to b_0][y_1\to b_1][x\to a])\models\varphi^{\mathcal{S}}_{x',y_0,y_1,x,\text{pref}}$.
By hypothesis on a,b_0,b_1,b, we have either $(\mathcal{A},[y_0\to b_0])\models\varphi^{\mathcal{S}}_{z,\text{null}}$ and $(\mathcal{A},[y_1\to b_1])\models\varphi^{\mathcal{S}}_{z,\text{null}}$ or there are $c',c''\in|\mathcal{A}|$ such that $(\mathcal{A},[y'\to c'][y''\to c''][x'\to b])\models\varphi^{\mathcal{S}}_{y',y'',\hat{x},\text{last-entry}}$ and $(\mathcal{A},[u\to b_0][v\to b_1][s\to c'][t\to c''])\models\varphi^{\mathcal{S},\Im}_{u,v,s,t,\text{one-step}}$.

In the first case, $b_0 = \phi^{\mathcal{S}}(\lambda)$ and $b_1 = \phi^{\mathcal{S}}(\lambda)$, by the **(NULL)** axiom, which yields the desired conclusion.
In the second case, there is i such that $(\mathcal{A}, [z_0 \to b_0][z_1 \to c']) \models \varphi^{\mathcal{S}}_{z_0,z_1,s_i,\text{conc}}$ and $(\mathcal{A}, [z_0 \to b_1][z_1 \to c'']) \models \varphi^{\mathcal{S}}_{z_0,z_1,t_i,\text{conc}}$. The definition of the relation $\sqsubset^{\mathcal{S}}$ implies that $c' \sqsubset^{\mathcal{S}} b_0$ and $c'' \sqsubset^{\mathcal{S}} b_1$. By the axiom **(ILI)**, there is $\hat{b} \in |\mathcal{A}|$ such that $(\mathcal{A}, [x' \to \hat{b}][y_0 \to c'][y_1 \to c''][x \to a]) \models \varphi^{\mathcal{S}}_{x',y_0,y_1,\text{pref}}$. Thus, by the inductive hypothesis, there are $s', s'' \in V^*$ such that $c' = \phi^{\mathcal{S}}(s')$ and $c'' = \phi^{\mathcal{S}}(s'')$, where there is a construction sequence for $\Im$ whose last members are s', s''. Thus, there is a construction sequence for $\Im$ ending in $s's_i, s''t_i$, and by the **(CONC)** axiom we have $b_0 = \phi^{\mathcal{S}}(s's_i)$ and $b_1 = \phi^{\mathcal{S}}(s''t_i)$, as desired.

(194) Let $\mathcal{A}$ be an $\mathcal{L}$-structure, $\mathcal{S}$ be a PCP-reduction scheme for $\mathcal{A}$, and

$$\Im = (\{d, e\}, (s_0, \ldots, s_{n-1}), (t_0, \ldots, t_{n-1}))$$

be an instance of the PCP. Prove that if $\mathcal{S}$ satisfies the axioms **(WF), (NULL), (ILI), (CONC)**, and **(LI)**, and $\mathcal{A} \models \varphi^{\mathcal{S},\Im}_{\text{solv}}$, then $\Im$ has a solution.

Solution: The hypothesis implies the existence of $a, b \in |\mathcal{A}|$ such that $(\mathcal{A}, [x \to a]) \models \varphi^{\mathcal{S},\Im}_{x,\text{const}}$, $(\mathcal{A}, [y' \to b][y'' \to b][\hat{x} \to a]) \models \varphi^{\mathcal{S}}_{y',y'',\hat{x},\text{last-entry}}$, and $(\mathcal{A}, [z \to b]) \models (\neg\varphi^{\mathcal{S}}_{z,\text{null}})$. By the **(LI)** axiom, there is $\hat{b} \in |\mathcal{A}|$ such that $(\mathcal{A}, [x' \to \hat{b}][y_0 \to b][y_1 \to b][x_0 \to a]) \models \varphi^{\mathcal{S}}_{x',y_0,y_1,x,\text{pref}}$. By Supplement 193, there are $s, s' \in V^*$ such that $b = \phi^{\mathcal{S}}(s) = \phi^{\mathcal{S}}(s')$ and there is a construction sequence for $\Im$ whose last members are s and s'. The injectivity of $\phi^{\mathcal{S}}$ implies $s = s'$. Further, notice that by the **(NULL)** axiom, b cannot be the image of the null word, so $s \neq \lambda$. Thus, $\Im$ has a solution.

(195) Let $\mathcal{A}$ be an $\mathcal{L}$-structure and $\mathcal{S}$ be a PCP-reduction scheme for $\mathcal{A}$ that satisfies the axioms **(NULL),(CONC), (INIT-OF), (LE), (WF), (LI)** and **(ILI)**. Prove that $\mathcal{A}$ has an undecidable theory.

Hint. If the theory of $\mathcal{A}$ were decidable, it would be possible to decide whether an arbitrary instance of the PCP with alphabet $V = \{d, e\}$ has a solution.

(196) Prove that the theory of the structure $\mathcal{A}$ introduced in Exercise 190 is undecidable.

(197) Let $\mathcal{A}$ be an $\mathcal{L}$-structure. Generalize the results that lead to Supplement 195, by considering an alphabet V with $|V| \geq 2$ and encodings of the form $\phi : V^* \longrightarrow |\mathcal{A}|^k$ and $\Phi : E_V \longrightarrow |\mathcal{A}|^l$, where $k, l \in \mathbf{N}$. Reprove the undecidability of arithmetic using this generalized form of Supplement 195.
Hint. For the second part of this exercise, modify the proof of Theorem 4.14.15.

(198) Let V be an alphabet with $|V| = 1$. Prove it is decidable if an arbitrary V-instance of the PCP has a solution.

(199) Let $V = \{1, 2, \ldots, n\}$ be an alphabet, where $n \geq 2$, and let V^* be the set of words over V. The function $l_V : V^* \longrightarrow \mathbf{N}$ defined as $l_V(w) = |V|^{|w|}$, where $|w|$ is the length of the word w, gives the number of words of length $|w|$. Define the function $k_V : V^* \longrightarrow \mathbf{N}$ as $k_V(\lambda) = 0$ and $k_V(ua) = k_V(u)n + a$. Prove that the function k_V is injective and for every $u, v \in V^*$ we have:

$$l_V(uv) = l_V(u) \cdot l_V(v)$$
$$k_V(uv) = k_V(u) \cdot l_V(v) + k_V(v).$$

(200) Let $V = \{1, 2, \ldots, n\}$ be an alphabet, where $n \geq 2$, and let V^* be the set of words of V.

(a) Define the mapping $M_V : V^* \times V^* \longrightarrow \mathbf{R}^{3\times3}$ by

$$M_V(u, v) = \begin{pmatrix} l_V(u) & l_V(v) - l_V(u) & 0 \\ 0 & l_V(v) & 0 \\ k_V(u) & k_V(v) - k_V(u) & 1 \end{pmatrix}$$

for every $u, v \in V^*$. Prove that

$$M_V(u, v) \cdot M_V(u', v') = M_V(uu', vv')$$

for every $u, v, u', v' \in V^*$.

(b) Using the undecidability of the PCP prove that it is undecidable whether for a finite set of 3×3-matrices $\mathcal{M} = \{M_0, \ldots, M_{m-1}\}$ over $\mathbf{R}$ and a pair of numbers $(i, j) \in \{1, 2, 3\}^2$ there is a product $M = M_{i_0} \cdots M_{i_{p-1}}$

of these matrices such that the (i, j)-component of the matrix M equals 0. (For a fixed (i, j), this is called the (i, j)-*mortality problem of* $\mathcal{M}$.)

4.16 Bibliographical Comments

Tarski's "definition of truth" was introduced in [32]. A translation appears in [35].

Skolem's original paper [29] where Skolem normal form was introduced is included in the collection [1] in translation from the original German. In the same paper, Skolem improves the proof of what is known today as the Löwenheim-Skolem Theorem as given in [23].

Herbrand's Theorem appeared in [19].

Procedure 4.10.28 is due to Gilmore [18].

Nonstandard models of arithmetic were introduced in [30].

Corollary 4.14.14 is due to Church (see [7]).

The method of elimination of quantifiers was conceived by Tarski published in [33, 34].

The decidability of Presburger arithmetic was obtained in [25]. See also [13].

The set Δ_n of Supplements 9 and 66 was introduced by Boolos in [5] and will be used in Chapter 5.

Our treatment of the undecidability of the theory of arithmetic follows [36].

Chapter 5

First-Order Logic–Formal Systems

5.1 Introduction

This chapter plays the same role for first-order logic that Chapter 3 played for propositional logic. We present a variety of syntactic methods that generalize corresponding methods of propositional logic which allow us to construct formal proofs to show that $\Gamma \models \varphi$ or, equivalently, that $\Gamma \cup \{(\neg\varphi)\}$ is unsatisfiable, where Γ is a set of first-order formulas and φ is a first-order formula. Each of these methods can be expressed as a formal system which manipulates first-order formulas or sets of such formulas. If every formula that appears in a syntactic proof that $\Gamma \models \varphi$ belongs to the analytic universe of $\Gamma \cup \{\varphi\}$ as defined in Definition 4.12.2, then the formal system is called *analytical*.

As in the case of propositional logic, analytical methods are easier to automate, though they may yield long proofs while nonanalytical methods are closer to ordinary mathematical reasoning.

We discuss both general syntactic methods which can deal with arbitrary formulas as well as a specialized syntactic method that applies to formulas in clausal form. Both nonanalytical methods (Hilbert/Frege systems, tableaux with cut, sequent calculus with cut, and natural deduction) and analytical methods (tableaux without cut, and sequent calculus without cut) are presented.

5.2 A Hilbert/Frege-Style Formal System

The formal system we introduce in this section is an amplification of the Hilbert-Frege system for propositional logic introduced in Section 3.2. The new system has one rule of inference: the familiar modus ponens rule. The collection of axioms is much richer compared to the propositional logic case, in order to deal with the more complex formulas of first-order logic.

Definition 5.2.1. Let $\mathcal{L}$ be a first-order language. The formal system $\mathcal{HF}^{\mathcal{L}}$ consists of:

- $\mathrm{FORM}_{\mathcal{L}}$ as set of objects.
- $\{\,\mathsf{R}^{\mathcal{L}}_{mp}\}$ as set of rules, where $\mathsf{R}^{\mathcal{L}}_{mp}$ is the *modus ponens rule for* $\mathcal{L}$

$$\frac{\varphi, (\varphi \to \psi)}{\psi}$$

  for all formulas $\varphi, \psi \in \mathrm{FORM}_{\mathcal{L}}$.
- A as axiom set, where A consists of all generalizations of the following formulas:

  (1) $(\alpha \to (\beta \to \alpha))$
  (2) $((\alpha \to (\beta \to \gamma)) \to ((\alpha \to \beta) \to (\alpha \to \gamma)))$
  (3) $(\alpha \to \alpha)$
  (4) $(\alpha \to ((\neg\alpha) \to \beta))$
  (5) $(((\neg\alpha) \to \alpha) \to \alpha)$
  (6) $((\alpha \to (\neg\alpha)) \to (\neg\alpha))$
  (7) $(\alpha \to (\alpha \vee \beta))$
  (8) $(\beta \to (\alpha \vee \beta))$
  (9) $(\alpha \to (\beta \to (\alpha \wedge \beta)))$
  (10) $((\neg\alpha) \to (\alpha \to \beta))$
  (11) $(\alpha \to (\beta \to (\alpha \leftrightarrow \beta)))$
  (12) $((\neg\alpha) \to ((\neg\beta) \to (\alpha \leftrightarrow \beta)))$
  (13) $((\neg\alpha) \to ((\neg\beta) \to (\neg(\alpha \vee \beta))))$
  (14) $((\neg\alpha) \to (\neg(\alpha \wedge \beta)))$
  (15) $((\neg\beta) \to (\neg(\alpha \wedge \beta)))$
  (16) $(\alpha \to ((\neg\beta) \to (\neg(\alpha \to \beta))))$
  (17) $(\alpha \to ((\neg\beta) \to (\neg(\alpha \leftrightarrow \beta))))$
  (18) $((\neg\alpha) \to (\beta \to (\neg(\alpha \leftrightarrow \beta))))$
  (19) $((\forall x)\alpha \to \langle\alpha\rangle_{x:=t})$

(20) $(\langle\alpha\rangle_{x:=t} \to (\exists x)\alpha)$
(21) $((\forall x)(\alpha \to \beta) \to ((\forall x)\alpha \to (\forall x)\beta))$
(22) $(\alpha \to (\forall x)\alpha)$, where $x \notin \mathtt{FV}(\alpha)$
(23) $((\forall x)(\neg\alpha) \to (\neg(\exists x)\alpha))$

for all $\mathcal{L}$-formulas α, β, γ, variables x, and $\mathcal{L}$-terms t.

In addition, if $=\in \mathcal{L}$, A also contains all generalizations of the $\mathcal{L}$-instances of the $\mathcal{L}$-equality axioms:

(24) $t = t$
(25) $((t_0 = t_1) \to (t_1 = t_0))$
(26) $(((t_0 = t_1) \wedge (t_1 = t_2)) \to (t_0 = t_2))$
(27) for every n-ary function symbol $f \in \mathcal{L}$ with $n > 0$,

$$(((t_0 = t_n) \wedge \cdots \wedge (t_{n-1} = t_{2n-1})) \\ \to (f(t_0, \ldots, t_{n-1}) = f(t_n, \ldots, t_{2n-1})))$$

(28) for every n-ary relation symbol $R \in \mathcal{L}$ with $R \neq =$ and $n > 0$,

$$(((t_0 = t_n) \wedge \cdots \wedge (t_{n-1} = t_{2n-1})) \\ \to (R(t_0, \ldots, t_{n-1}) \leftrightarrow R(t_n, \ldots, t_{2n-1})))$$

where all t_i mentioned in axioms 24 to 28 are $\mathcal{L}$-terms. □

For $1 \leq i \leq 28$, we refer to the formulas that occur under the rubric i of the previous definition as the *ith formula group of* $\mathcal{HF}^{\mathcal{L}}$. The set of generalizations of these formulas will be referred to as the *ith axiom group of* $\mathcal{HF}^{\mathcal{L}}$.

We begin our discussion of $\mathcal{HF}^{\mathcal{L}}$ by observing that the formulas of the groups 1-18 are all first-order tautologies in the sense of Definition 4.8.3. For instance, any axiom of group 1 is a substitution instance of the propositional tautology $(p_0 \to (p_1 \to p_0))$, which is itself an axiom of the group 1 of the Hilbert/Frege-style system $\mathcal{HF}$ of propositional logic. Therefore, these formulas are logically valid by Theorem 4.8.14.

Theorem 5.2.2 (Soundness of $\mathcal{HF}^{\mathcal{L}}_{\Gamma}$). *Let $\mathcal{L}$ be a first-order language, Γ be a set of $\mathcal{L}$-formulas and let φ be an $\mathcal{L}$-formula. Then, $\Gamma \vdash_{\mathcal{HF}^{\mathcal{L}}} \varphi$ implies $\Gamma \models \varphi$.*

Proof. The argument is by induction on the theorems of $\mathcal{HF}^{\mathcal{L}}_{\Gamma}$. For the basis step, let φ be an axiom of $\mathcal{HF}^{\mathcal{L}}_{\Gamma}$. Then, either φ is an axiom of $\mathcal{HF}^{\mathcal{L}}$ or φ belongs to Γ. In the latter case, we obviously have $\Gamma \models \varphi$. In the former case, φ is a generalization of a member φ' of one of the formula groups. By Corollary 4.5.59, it suffices to show the logical validity of φ'. We consider several subcases. We observed already that all the formulas in the first eighteen formula groups are logically valid. The logical validity of the formulas in formula groups 19 and 20 follows from Theorems 4.6.51 and 4.5.55. The same property for formulas in formula group 21 was established in Example 4.5.23. Next, for formulas in formula group 22, the logical validity follows from Theorems 4.5.37 and 4.5.55. The logical validity of the formulas in formula group 23 is a consequence of Corollary 4.8.18. Finally, for formulas in formula groups 24 to 28, the logical validity is shown by Corollary 4.6.10.

The inductive step for the modus ponens rule is an immediate consequence of the second part of Theorem 4.5.52. □

Our major goal in this section is to prove the completeness of $\mathcal{HF}^{\mathcal{L}}_{\Gamma}$. A preliminary, very limited, version of this result is given in the next theorem.

Theorem 5.2.3. *Let $\mathcal{L}$ be a first-order language. Then, for every $\mathcal{L}$-formula φ that is a tautology of first-order logic, we have $\vdash_{\mathcal{HF}^{\mathcal{L}}} \varphi$.*

Proof. We begin by observing that if s is an $\mathcal{L}$-substitution (that is, an inter-substitution that maps statement variables into $\mathcal{L}$-formulas) and θ is an axiom of $\mathcal{HF}$, then $s(\theta)$ is an axiom of $\mathcal{HF}^{\mathcal{L}}$. Also, if β follows by modus ponens from α and $(\alpha \to \beta)$ in $\mathcal{HF}$, then $s(\beta)$ follows by the same rule from $s(\alpha)$ and $s((\alpha \to \beta)) = (s(\alpha) \to s(\beta))$ in $\mathcal{HF}^{\mathcal{L}}$. Therefore, if $(\theta_0, \ldots, \theta_{n-1})$ is a proof in $\mathcal{HF}$, then $(s(\theta_0), \ldots, s(\theta_{n-1}))$ is a proof in $\mathcal{HF}^{\mathcal{L}}$.

Let ψ be a propositional tautology that is a propositional form for φ and s be an $\mathcal{L}$-substitution such that $s(\psi) = \varphi$. By the completeness of $\mathcal{HF}$, there is a proof $(\theta_0, \ldots, \theta_{n-1})$ of ψ in $\mathcal{HF}$ which transforms into a proof $(s(\theta_0), \ldots, s(\theta_{n-1}))$ in $\mathcal{HF}^{\mathcal{L}}$ of φ. □

Definition 5.2.4. Let $\mathcal{L}$ be a first-order language and let $n \in \mathbf{N}$. The *n-ary tautological consequence rule for $\mathcal{L}$*, $TC^{\mathcal{L}}_n$, is given by

$$\frac{\varphi_0, \cdots, \varphi_{n-1}}{\psi}$$

for all $\mathcal{L}$-formulas $\varphi_0, \dots, \varphi_{n-1}, \psi$ such that

$$(\varphi_0 \to (\varphi_1 \to \cdots (\varphi_{n-2} \to (\varphi_{n-1} \to \psi)) \cdots))$$

is a tautology. ▯

Note that the rule $TC_0^{\mathcal{L}}$ says that any tautology can be deduced without any premises.

Theorem 5.2.5. *For every first-order language $\mathcal{L}$ and $n \in \mathbf{N}$, the n-ary tautological consequence rule for $\mathcal{L}$ is a derived rule of the formal system $\mathcal{HF}^{\mathcal{L}}$.*

Proof. Let $\varphi_0, \dots, \varphi_{n-1}, \psi$ be $\mathcal{L}$-formulas such that

$$(\varphi_0 \to (\varphi_1 \to \cdots (\varphi_{n-2} \to (\varphi_{n-1} \to \psi)) \cdots))$$

is a tautology. By Theorem 5.2.3, this formula is a theorem of $\mathcal{HF}^{\mathcal{L}}$. We need to show that $\{\varphi_0, \dots, \varphi_{n-1}\} \vdash_{\mathcal{HF}^{\mathcal{L}}} \psi$. This follows by n applications of modus ponens. □

If we apply any of the derived rules $TC_n^{\mathcal{L}}$, we will say that we applied *tautological consequence* ($TC^{\mathcal{L}}$).

The next statement is an effectivized form of Theorem 5.2.5.

Corollary 5.2.6. *Let $\mathcal{L}$ be a first-order language and α be a tautology of propositional logic of the form $(\alpha_0 \to (\alpha_1 \to \cdots (\alpha_{n-2} \to (\alpha_{n-1} \to \alpha_n)) \cdots))$. There is an effective construction of an $\mathcal{HF}_\Gamma^{\mathcal{L}}$-proof of $\psi = s(\alpha_n)$ which starts with the restriction of an $\mathcal{L}$-substitution s to the propositional variables that occur in α and with $\mathcal{HF}_\Gamma^{\mathcal{L}}$-proofs of the formulas $\varphi_i = s(\alpha_i)$ for $0 \le i \le n-1$.*

Proof. Let $(\theta_0, \dots, \theta_{p-1})$ be a fixed $\mathcal{HF}$-proof of α. Then,

$$(s(\theta_0), \dots, s(\theta_{p-1}))$$

is an $\mathcal{HF}_\Gamma^{\mathcal{L}}$-proof of $s(\alpha) = \varphi_0 \to (\varphi_1 \to \cdots (\varphi_{n-2} \to (\varphi_{n-1} \to \psi)) \cdots))$. Using the $\mathcal{HF}_\Gamma^{\mathcal{L}}$-proofs of the formulas $\varphi_0, \dots, \varphi_{n-1}$, and applying modus ponens n times, we obtain effectively a proof of ψ. □

Definition 5.2.7. Let $\mathcal{L}$ be a first-order language. Two $\mathcal{L}$-formulas φ, ψ are *provably equivalent in* $\mathcal{HF}^{\mathcal{L}}$ (or *provably equivalent* for short) if $\vdash_{\mathcal{HF}^{\mathcal{L}}} (\varphi \leftrightarrow \psi)$. ▯

Note that to demonstrate that two $\mathcal{L}$-formulas φ, ψ are provably equivalent, it suffices to show that $\vdash_{\mathcal{HF}^{\mathcal{L}}} (\varphi \to \psi)$ and $\vdash_{\mathcal{HF}^{\mathcal{L}}} (\psi \to \varphi)$, by Tautological Consequence.

Example 5.2.8. Using tautological consequence, we can prove that $\vdash_{\mathcal{HF}^{\mathcal{L}}} ((\forall x_0)(\forall x_1)R(x_0, x_1) \to R(f(x_0), g(x_1)))$, where R is a binary relation symbol and f, g are two unary function symbols in the first-order language $\mathcal{L}$.

By Axiom Group 19, we have

$$\vdash_{\mathcal{HF}^{\mathcal{L}}} ((\forall x_0)(\forall x_1)R(x_0, x_1) \to (\forall x_1)R(f(x_0), x_1)).$$

By the same axiom group, we obtain

$$\vdash_{\mathcal{HF}^{\mathcal{L}}} ((\forall x_1)R(f(x_0), x_1) \to R(f(x_0), g(x_1))).$$

Finally, by applying tautological consequence to the previous two formulas, we obtain the desired formula.

We now show that $\vdash_{\mathcal{HF}^{\mathcal{L}}} ((\forall x_0)(\forall x_1)R(x_0, x_1) \to R(f(x_1), g(x_0)))$. Although the formula involved is similar to the previous one, there are certain technical complications. By the same Axiom Group 19, we have

$$\vdash_{\mathcal{HF}^{\mathcal{L}}} ((\forall x_0)(\forall x_1)R(x_0, x_1) \to (\forall x_2)R(f(x_1), x_2)),$$

because x_2 is the first variable not occurring in $R(x_0, x_1)$ or in $f(x_1)$, so $(\forall x_2)R(x_0, x_2) = \mathsf{variant}((\forall x_1)R(x_0, x_1), x_0, f(x_1))$, by the last part of Definition 4.6.43. By the same axiom group, we obtain

$$\vdash_{\mathcal{HF}^{\mathcal{L}}} ((\forall x_2)R(f(x_1), x_2) \to R(f(x_1), g(x_0))).$$

To conclude we apply tautological consequence to the previous two formulas. $\square$

Example 5.2.9. Using a similar approach as in Example 5.2.8, we can show that $\vdash_{\mathcal{HF}^{\mathcal{L}}} ((\forall x)\varphi \to (\exists x)\varphi)$, for every $\mathcal{L}$-formula φ and variable x. Namely, we can write

(1)	$\vdash_{\mathcal{HF}^{\mathcal{L}}} ((\forall x)\varphi \to \varphi)$	Axiom Group 19
(2)	$\vdash_{\mathcal{HF}^{\mathcal{L}}} (\varphi \to (\exists x)\varphi)$	Axiom Group 20
(3)	$\vdash_{\mathcal{HF}^{\mathcal{L}}} ((\forall x)\varphi \to (\exists x)\varphi)$	(1), (2), and tautological consequence

$\square$

Theorem 5.2.10 (Generalization Theorem). *Let $\mathcal{L}$ be a first-order language, Γ be a set of $\mathcal{L}$-formulas, φ be an $\mathcal{L}$-formula, and x be a variable such that $x \notin \mathtt{FV}(\Gamma)$. If $\Gamma \vdash_{\mathcal{HF}^{\mathcal{L}}} \varphi$, then $\Gamma \vdash_{\mathcal{HF}^{\mathcal{L}}} (\forall x)\varphi$.*

Proof. The argument is by induction on the theorems of $\mathcal{HF}^{\mathcal{L}}_{\Gamma}$. There are two basis steps. If φ is an axiom obtained as a generalization of a formula ψ of one of the formula groups of $\mathcal{HF}^{\mathcal{L}}$, then $(\forall x)\varphi$ is also a generalization of ψ and therefore is also an axiom. If $\varphi \in \Gamma$, then $x \notin \mathtt{FV}(\varphi)$, so $\Gamma \vdash_{\mathcal{HF}^{\mathcal{L}}} (\varphi \to (\forall x)\varphi)$ by axiom group 22. Since $\Gamma \vdash_{\mathcal{HF}^{\mathcal{L}}} \varphi$, by modus ponens we obtain $\Gamma \vdash_{\mathcal{HF}^{\mathcal{L}}} (\forall x)\varphi$.

In the inductive step, suppose that φ is obtained by modus ponens from ψ and $(\psi \to \varphi)$ and that $\Gamma \vdash_{\mathcal{HF}^{\mathcal{L}}} (\forall x)\psi$ and $\Gamma \vdash_{\mathcal{HF}^{\mathcal{L}}} (\forall x)(\psi \to \varphi)$. By axiom group 21, we have $\Gamma \vdash_{\mathcal{HF}^{\mathcal{L}}} ((\forall x)(\psi \to \varphi) \to ((\forall x)\psi \to (\forall x)\varphi))$. A double application of modus ponens yields $\Gamma \vdash_{\mathcal{HF}^{\mathcal{L}}} (\forall x)\varphi$. □

Observe that the argument of the Generalization Theorem provides an effective construction of an $\mathcal{HF}^{\mathcal{L}}_{\Gamma}$-proof of $(\forall x)\varphi$ starting from an $\mathcal{HF}^{\mathcal{L}}_{\Gamma}$-proof of φ, whenever $x \notin \mathtt{FV}(\Gamma)$.

Example 5.2.11. Consider the formula

$$\varphi = (\forall x)(\forall y)((((\forall z)(f(x,z) = z) \land (\forall w)(f(w,y) = w)) \to (x = y))$$

discussed in Example 4.5.20.

We show that $\vdash_{\mathcal{HF}^{\mathcal{L}}} \varphi$ by the following sequence of steps:

(1) $((\forall z)(f(x,z) = z) \to (f(x,y) = y))$
(Axiom Group 19)

(2) $((\forall w)(f(w,y) = w) \to (f(x,y) = x))$
(Axiom Group 19)

(3) $((f(x,y) = x) \to (x = f(x,y)))$
(by Axiom Group 25)

(4) $(((x = f(x,y)) \land (f(x,y) = y)) \to (x = y))$
(by Axiom Group 26)

(5) $(((\forall z)(f(x,z) = z) \land (\forall w)(f(w,y) = w)) \to (x = y))$
(by Tautological Consequence and (1)–(4))

(6) $(\forall x)(\forall y)((((\forall z)(f(x,z) = z) \land (\forall w)(f(w,y) = w)) \to (x = y))$
(by a double application of the Generalization Theorem).

□

The analogue of the propositional deduction theorem for $\mathcal{HF}$ is given next.

Theorem 5.2.12 (Deduction Theorem for $\mathcal{HF}^{\mathcal{L}}$). *Let $\mathcal{L}$ be a first-order language, Γ be a set of $\mathcal{L}$-formulas and let φ, ψ be $\mathcal{L}$-formulas. Then, if $\Gamma \cup \{\varphi\} \vdash_{\mathcal{HF}^{\mathcal{L}}} \psi$, we have $\Gamma \vdash_{\mathcal{HF}^{\mathcal{L}}} (\varphi \to \psi)$.*

Proof. The argument follows the lines of the inductive argument of Theorem 3.2.4. $\square$

Corollary 5.2.13. *Let $\mathcal{L}$ be a first-order language, Γ be a set of $\mathcal{L}$-formulas and let φ, ψ be $\mathcal{L}$-formulas. Then, $\Gamma \cup \{\varphi\} \vdash_{\mathcal{HF}^{\mathcal{L}}} \psi$ if and only if $\Gamma \vdash_{\mathcal{HF}^{\mathcal{L}}} (\varphi \to \psi)$.*

Proof. The statement follows from Theorem 5.2.12 exactly the same way as Corollary 3.2.5 was obtained from Theorem 3.2.4. $\square$

As in propositional logic, the construction of the proof of $(\varphi \to \psi)$ in $\mathcal{HF}^{\mathcal{L}}_{\Gamma}$ starting from the proof of ψ in $\mathcal{HF}^{\mathcal{L}}_{\Gamma\cup\{\varphi\}}$ is effective. The converse direction is clearly effective since given a proof $\Gamma \vdash_{\mathcal{HF}^{\mathcal{L}}} (\varphi\to \psi)$, we can obtain effectively a proof $\Gamma \cup \{\varphi\} \vdash_{\mathcal{HF}^{\mathcal{L}}} \psi$ by adding the formulas φ and ψ to end of the proof.

Example 5.2.14. By Example 5.2.8 and the converse of the Deduction Theorem, we have both $\{(\forall x_0)(\forall x_1)R(x_0, x_1)\} \vdash_{\mathcal{HF}^{\mathcal{L}}} R(f(x_0), g(x_1))$ and $\{(\forall x_0)(\forall x_1)R(x_0, x_1)\} \vdash_{\mathcal{HF}^{\mathcal{L}}} R(f(x_1), g(x_0))$, where R is a binary relation symbol and f, g are two unary function symbols in the first-order language $\mathcal{L}$.

Similar observations apply to Examples 5.2.9 and 5.2.11. ▯

Example 5.2.15. Let $\mathcal{L}$ be a first-order language, Γ be a set of $\mathcal{L}$-formulas, and φ, ψ be two $\mathcal{L}$-formulas. The following statements hold:

(1) If $\Gamma \cup \{\varphi\} \vdash_{\mathcal{HF}^{\mathcal{L}}} \psi$, then $\Gamma \cup \{(\neg\psi)\} \vdash_{\mathcal{HF}^{\mathcal{L}}} (\neg\varphi)$.
(2) If $\Gamma \cup \{(\neg\varphi)\} \vdash_{\mathcal{HF}^{\mathcal{L}}} \psi$, then $\Gamma \cup \{(\neg\psi)\} \vdash_{\mathcal{HF}^{\mathcal{L}}} \varphi$.
(3) If $\Gamma \cup \{\varphi\} \vdash_{\mathcal{HF}^{\mathcal{L}}} (\neg\psi)$, then $\Gamma \cup \{\psi\} \vdash_{\mathcal{HF}^{\mathcal{L}}} (\neg\varphi)$.
(4) If $\Gamma \cup \{(\neg\varphi)\} \vdash_{\mathcal{HF}^{\mathcal{L}}} (\neg\psi)$, then $\Gamma \cup \{\psi\} \vdash_{\mathcal{HF}^{\mathcal{L}}} \varphi$.

These statements will be referred to generically as *proofs by contraposition* or simply *contrapositions.*

For the first statement, the Deduction Theorem yields $\Gamma \vdash_{\mathcal{HF}^{\mathcal{L}}} (\varphi \to \psi)$. By Tautological Consequence, $\Gamma \vdash_{\mathcal{HF}^{\mathcal{L}}} ((\neg\psi) \to (\neg\varphi))$. An application of Corollary 5.2.13 gives the desired conclusion.

We leave to the reader the arguments for the remaining statements. $\square$

Observe that the construction of the proofs specified by the contrapositions is effective due to the effectiveness of proofs by Tautological Consequence and of proofs made by Corollary 5.2.13.

Lemma 5.2.16. *Let $\mathcal{L}$ be a first-order language, Γ be a set of $\mathcal{L}$-formulas and let φ, ψ be two $\mathcal{L}$-formulas such that $\Gamma \cup \{\varphi\} \vdash_{\mathcal{HF}^{\mathcal{L}}} \psi$. If x is a variable that does not occur free in $\Gamma \cup \{\psi\}$, then $\Gamma \cup \{(\exists x)\varphi\} \vdash_{\mathcal{HF}^{\mathcal{L}}} \psi$.*

Proof. Since $\Gamma \cup \{\varphi\} \vdash_{\mathcal{HF}^{\mathcal{L}}} \psi$, by contraposition, we have $\Gamma \cup \{(\neg\psi)\} \vdash_{\mathcal{HF}^{\mathcal{L}}} (\neg\varphi)$. Since x does not occur free in $\Gamma \cup \{(\neg\psi)\}$, by the Generalization Theorem, we obtain $\Gamma \cup \{(\neg\psi)\} \vdash_{\mathcal{HF}^{\mathcal{L}}} (\forall x)(\neg\varphi)$. By Axiom Group 23, we have $\Gamma \cup \{(\neg\psi)\} \vdash_{\mathcal{HF}^{\mathcal{L}}} (\neg(\exists x)\varphi)$. Another application of contraposition yields $\Gamma \cup \{(\exists x)\varphi\} \vdash_{\mathcal{HF}^{\mathcal{L}}} \psi$. $\square$

The proof showing that $\Gamma \cup \{(\exists x)\varphi\} \vdash_{\mathcal{HF}^{\mathcal{L}}} \psi$ can be obtained effectively from the proof showing that $\Gamma \cup \{\varphi\} \vdash_{\mathcal{HF}^{\mathcal{L}}} \psi$ because all the necessary steps outlined in the proof of Lemma 5.2.16 have already been shown to be effective.

Theorem 5.2.17. *Let $\mathcal{L}$ be a first-order language, φ be an $\mathcal{L}$-formula and x, y be variables such that y is substitutable for x in φ and $y \notin$ $\mathtt{FV}(\varphi)$. Then, $\{(\forall y)(\varphi)_{x:=y}\} \vdash_{\mathcal{HF}^{\mathcal{L}}} (\forall x)\varphi$ and $\{(\exists y)(\varphi)_{x:=y}\} \vdash_{\mathcal{HF}^{\mathcal{L}}} (\exists x)\varphi$.*

Proof. For the first part of the theorem, we begin by observing that by Corollary 4.3.78, x is substitutable for y in $(\varphi)_{x:=y}$. Therefore, the formula $((\forall y)(\varphi)_{x:=y} \to ((\varphi)_{x:=y})_{y:=x})$ belongs to Axiom Group 19. By Corollary 4.3.87, $((\varphi)_{x:=y})_{y:=x}) = \varphi$, so $\vdash_{\mathcal{HF}^{\mathcal{L}}} ((\forall y)(\varphi)_{x:=y} \to \varphi)$. Consequently, by Corollary 5.2.13, $(\forall y)(\varphi)_{x:=y} \vdash_{\mathcal{HF}^{\mathcal{L}}} \varphi$. Since x does not occur free in $(\forall y)(\varphi)_{x:=y}$, by the Generalization Theorem, we have $(\forall y)(\varphi)_{x:=y} \vdash_{\mathcal{HF}^{\mathcal{L}}} (\forall x)\varphi$.

For the second part of the theorem, since y is substitutable for x in φ, we have by Axiom Group 20, $\vdash_{\mathcal{HF}^{\mathcal{L}}} ((\varphi)_{x:=y} \to (\exists x)\varphi)$, so by Corollary 5.2.13, $(\varphi)_{x:=y} \vdash_{\mathcal{HF}^{\mathcal{L}}} (\exists x)\varphi$. Since y does not occur free in φ, and therefore doesn't occur free in $(\exists x)\varphi$, it follows by Lemma 5.2.16 that $(\exists y)(\varphi)_{x:=y} \vdash_{\mathcal{HF}^{\mathcal{L}}} \rightarrow (\exists x)\varphi$. □

Note that by previous effectiveness results, the proofs showing that

$$\{(\forall y)(\varphi)_{x:=y}\} \vdash_{\mathcal{HF}^{\mathcal{L}}} (\forall x)\varphi \text{ and } \{(\exists y)(\varphi)_{x:=y}\} \vdash_{\mathcal{HF}^{\mathcal{L}}} (\exists x)\varphi$$

can be found effectively given the $\mathcal{L}$-formula φ and the variables x and y satisfying the conditions of Theorem 5.2.17.

Corollary 5.2.18. *Let $\mathcal{L}$ be a first-order language, φ be an $\mathcal{L}$-formula and x, y be variables such that y is substitutable for x in φ and $y \notin \mathtt{FV}(\varphi)$. Then, the formulas $(\forall y)(\varphi)_{x:=y}$ and $(\forall x)\varphi$ are provably equivalent as are the formulas $(\exists y)(\varphi)_{x:=y}$ and $(\exists x)\varphi$.*

Proof. The statements are clearly true if $x = y$. Therefore, suppose that $x \neq y$.

For the first part, note that $\vdash_{\mathcal{HF}^{\mathcal{L}}} ((\forall y)(\varphi)_{x:=y} \rightarrow (\forall x)\varphi)$ by Theorem 5.2.17 and Corollary 5.2.13. By Corollary 4.3.84, $x \notin \mathtt{FV}((\varphi)_{x:=y})$. By Corollary 4.3.78, x is substitutable for y in $(\varphi)_{x:=y}$. Thus, by reversing the roles of x and y and replacing φ with $(\varphi)_{x:=y}$, we obtain by Theorem 5.2.17 and Corollary 5.2.13 that $\vdash_{\mathcal{HF}^{\mathcal{L}}} ((\forall x)((\varphi)_{x:=y})_{y:=x} \rightarrow (\forall y)(\varphi)_{x:=y})$. By Corollary 4.3.87, $((\varphi)_{x:=y})_{y:=x} = \varphi$, which gives the desired result.

For the second part, Theorem 5.2.17 and Corollary 5.2.13 imply that $\vdash_{\mathcal{HF}^{\mathcal{L}}} ((\exists y)(\varphi)_{x:=y} \rightarrow (\exists x)\varphi)$. By exchanging the roles of x and y, we obtain as in the previous case, $\vdash_{\mathcal{HF}^{\mathcal{L}}} ((\exists x)\varphi \rightarrow (\exists y)(\varphi)_{x:=y})$. □

Using the effectiveness of the results used in the corollary, one can find effectively proofs for $((\forall y)(\varphi)_{x:=y} \leftrightarrow (\forall x)\varphi)$ and $((\exists y)(\varphi)_{x:=y} \leftrightarrow (\exists x)\varphi)$ whenever φ, x and y satisfy the conditions of Corollary 5.2.18.

Lemma 5.2.19. *Let θ be an instance of one of the formula groups 19 or 20, c be a constant symbol and let y be a variable that does not occur in θ. Then, $\mathsf{s}_y^c(\theta)$ is an instance of the same formula group.*

Proof. Suppose that θ is the formula

$$\theta = ((\forall x)\varphi \rightarrow \langle\varphi\rangle_{x:=t}) = ((\forall x)\varphi \rightarrow (\mathsf{variant}(\varphi, x, t))_{x:=t}).$$

Since y does not occur in θ, it follows that $y \neq x$ and, further, y does not occur in either φ or $\mathsf{variant}(\varphi, x, t)$. Thus,

$$\begin{aligned}\mathsf{s}_y^c(\theta) &= ((\forall x)\mathsf{s}_y^c(\varphi) \to \mathsf{s}_y^c(\langle\varphi\rangle_{x:=t}))\\ &= ((\forall x)\mathsf{s}_y^c(\varphi) \to \langle\mathsf{s}_y^c(\varphi)\rangle_{x:=\mathsf{s}_y^c(t)})\\ &\quad \text{(by Corollary 4.6.55).}\end{aligned}$$

The last formula is clearly in formula group 19.

The argument for the formula group 20 is similar and is left to the reader. □

Theorem 5.2.20. *Let Γ be a set of $\mathcal{L}$-formulas, $(\theta_0, \ldots, \theta_{n-1})$ be a proof in $\mathcal{HF}_\Gamma^{\mathcal{L}}$, c be a constant symbol of $\mathcal{L}$ that does not occur in Γ, and y be a variable that does not occur in any formula θ_i for $0 \leq i \leq n-1$. Then, the sequence $(\mathsf{s}_y^c(\theta_0), \ldots, \mathsf{s}_y^c(\theta_{n-1}))$ is also a proof in $\mathcal{HF}_\Gamma^{\mathcal{L}}$.*

Proof. Fix i with $0 \leq i \leq n-1$. We need to consider three main cases.

`Case 1:` θ_i is an axiom of $\mathcal{HF}^{\mathcal{L}}$ which results from generalizing a formula θ_i' of formula group j for some j between 1 and 28. Then, $\mathsf{s}_y^c(\theta_i)$ is a generalization of $\mathsf{s}_y^c(\theta_i')$, so it suffices to show that $\mathsf{s}_y^c(\theta_i')$ belongs to the same formula group as θ_i'. This is obvious when j belongs to the set $\{1, \ldots, 18, 21, 23, \ldots, 28\}$.
Suppose $j \in \{19, 20\}$. Then by Lemma 5.2.19, $\mathsf{s}_y^c(\theta_i')$ is in the same formula group as θ_i'.
Now suppose that θ_i' belongs to formula group 22, that is $\theta_i' = (\alpha \to (\forall x)\alpha)$, where $x \notin \mathtt{FV}(\alpha)$. We have $\mathsf{s}_y^c(\theta_i') = (\mathsf{s}_y^c(\alpha) \to (\forall x)\mathsf{s}_y^c(\alpha))$. Since $x \neq y$, by Theorem 4.3.69, $x \notin \mathtt{FV}(\mathsf{s}_y^c(\alpha))$, which shows that $\mathsf{s}_y^c(\theta_i')$ also belongs to formula group 22.

`Case 2:` θ_i is in Γ. Then, since c does not occur in any formula of Γ, $\mathsf{s}_y^c(\theta_i) = \theta_i$, which concludes this case.

`Case 3:` θ_i is obtained from θ_j and θ_k, where $0 \leq j, k < i$ by the modus ponens rule. In this case, $\mathsf{s}_y^c(\theta_i)$ is obtained by the same rule from $\mathsf{s}_y^c(\theta_j)$ and $\mathsf{s}_y^c(\theta_k)$.

□

Theorem 5.2.21. *Let $\mathcal{L}$ be a first-order language, Γ be a set of $\mathcal{L}$-formulas, and φ be an $\mathcal{L}$-formula. If $\Gamma \vdash_{\mathcal{HF}^{\mathcal{L}^c}} \varphi$, then $\Gamma \vdash_{\mathcal{HF}^{\mathcal{L}}} \varphi$.*

Proof. Let $(\theta_0, \ldots, \theta_{n-1})$ be proof in $\mathcal{HF}^{\mathcal{L}^c}$ of φ. If this proof is not in $\mathcal{HF}^{\mathcal{L}}$, let $c_0, \ldots, c_{k-1}$ be the constant symbols in $\mathcal{L}^c - \mathcal{L}$ that appear in the proof. Let y_0 be a variable that does not appear in the proof. By Theorem 5.2.20, the sequence $(\mathsf{s}_{y_0}^{c_0}(\theta_0), \ldots, \mathsf{s}_{y_0}^{c_0}(\theta_{n-1}))$ is also a proof in $\mathcal{HF}_{\Gamma}^{\mathcal{L}^c}$ of φ containing one fewer constant symbol in $\mathcal{L}^c - \mathcal{L}$. Repeating this process, we obtain a proof of φ in $\mathcal{HF}_{\Gamma}^{\mathcal{L}}$. □

The previous theorem provides an effective construction that transforms a proof in $\mathcal{HF}_{\Gamma}^{\mathcal{L}^c}$ into proof in $\mathcal{HF}_{\Gamma}^{\mathcal{L}}$.

Theorem 5.2.22. *Let $\mathcal{L}$ be a first-order language, Γ be a set of $\mathcal{L}$-formulas, φ be an $\mathcal{L}$-formula, and c be a constant symbol in $\mathcal{L}$ that does not occur in $\Gamma \cup \{\varphi\}$. If $\Gamma \vdash_{\mathcal{HF}^{\mathcal{L}}} (\varphi)_{x:=c}$, then $\Gamma \vdash_{\mathcal{HF}^{\mathcal{L}}} (\forall x)\varphi$.*

Proof. Let $(\theta_0, \ldots, \theta_{n-1})$ be a proof of $(\varphi)_{x:=c}$ in $\mathcal{HF}_{\Gamma}^{\mathcal{L}}$ and let Γ_0 be the finite subset of Γ consisting of those formulas of Γ that occur in this proof. Let y be a variable that does not occur in the proof and is different from x. By Theorem 5.2.20, we have $\Gamma_0 \vdash_{\mathcal{HF}^{\mathcal{L}}} \mathsf{s}_y^c((\varphi)_{x:=c})$. By Equality (4.1), $\mathsf{s}_y^c((\varphi)_{x:=c}) = (\mathsf{s}_y^c(\varphi))_{x:=\mathsf{s}_y^c(c)}$. Since c does not occur in φ, we have $\mathsf{s}_y^c(\varphi) = \varphi$; also, it is clear that $\mathsf{s}_y^c(c) = y$ and this yields $\mathsf{s}_y^c((\varphi)_{x:=c}) = (\varphi)_{x:=y}$. Thus, $\Gamma_0 \vdash_{\mathcal{HF}^{\mathcal{L}}} (\varphi)_{x:=y}$. By the Generalization Theorem, we have $\Gamma_0 \vdash_{\mathcal{HF}^{\mathcal{L}}} (\forall y)(\varphi)_{x:=y}$ because y does not occur in Γ_0. Since y does not occur in φ, by Theorem 5.2.17, we have $(\forall y)(\varphi)_{x:=y} \vdash_{\mathcal{HF}^{\mathcal{L}}} (\forall x)\varphi$. Thus, $\Gamma_0 \vdash_{\mathcal{HF}^{\mathcal{L}}} (\forall x)\varphi$, which implies $\Gamma \vdash_{\mathcal{HF}^{\mathcal{L}}} (\forall x)\varphi$. □

In view of the effectiveness of previous results mentioned in the argument of Theorem 5.2.22, this theorem provides an effective construction of an $\mathcal{HF}_{\Gamma}^{\mathcal{L}}$-proof of $(\forall x)\varphi$ given an $\mathcal{HF}_{\Gamma}^{\mathcal{L}}$-proof of $(\varphi)_{x:=c}$ under the syntactic condition mentioned in the statement.

Theorem 5.2.23. *Let $\mathcal{L}$ be a first-order language, Γ be a set of $\mathcal{L}$-formulas, and let φ, θ be two $\mathcal{L}$-formulas. If c is a constant symbol of $\mathcal{L}$ that does not occur in Γ, φ, θ, and $\Gamma \cup \{(\varphi)_{x:=c}\} \vdash_{\mathcal{HF}^{\mathcal{L}}} \theta$, then $\Gamma \cup \{(\exists x)\varphi\} \vdash_{\mathcal{HF}^{\mathcal{L}}} \theta$.*

Proof. Suppose that $\Gamma \cup \{(\varphi)_{x:=c}\} \vdash_{\mathcal{HF}^{\mathcal{L}}} \theta$. By Part (1) of Example 5.2.15, we have $\Gamma \cup \{(\neg\theta)\} \vdash_{\mathcal{HF}^{\mathcal{L}}} (\neg(\varphi)_{x:=c}) = ((\neg\varphi))_{x:=c}$. Since the constant symbol c does not occur in Γ, θ or φ, by Theorem 5.2.22, we have $\Gamma \cup \{(\neg\theta)\} \vdash_{\mathcal{HF}^{\mathcal{L}}} (\forall x)(\neg\varphi)$. Next, by Axiom Group 23, we obtain

$$\Gamma \cup \{(\neg\theta)\} \vdash_{\mathcal{HF}^{\mathcal{L}}} ((\forall x)(\neg\varphi) \rightarrow (\neg(\exists x)\varphi)),$$

which, by modus ponens, yields $\Gamma \cup \{(\neg\theta)\} \vdash_{\mathcal{HF}^{\mathcal{L}}} (\neg(\exists x)\varphi)$. Part (4) of Example 5.2.15 implies $\Gamma \cup \{(\exists x)\varphi\} \vdash_{\mathcal{HF}^{\mathcal{L}}} \theta$. □

The construction of the proof $\Gamma \cup \{(\exists x)\varphi\} \vdash_{\mathcal{HF}^{\mathcal{L}}} \theta$ is effective since it is based on previous effective results.

5.2.1 *Completeness of $\mathcal{HF}^{\mathcal{L}}$*

Definition 5.2.24. Let $\mathcal{L}$ be a first-order language. A set Γ of $\mathcal{L}$-formulas is

- $\mathcal{L}$*-consistent* if there is no $\mathcal{L}$-formula φ such that

$$\Gamma \vdash_{\mathcal{HF}^{\mathcal{L}}} \varphi \text{ and } \Gamma \vdash_{\mathcal{HF}^{\mathcal{L}}} (\neg\varphi);$$

- $\mathcal{L}$*-inconsistent* if it is not $\mathcal{L}$-consistent.

▯

As in propositional logic, every subset of an $\mathcal{L}$-consistent set of formulas is $\mathcal{L}$-consistent; consequently, every superset of an $\mathcal{L}$-inconsistent set of formulas is $\mathcal{L}$-inconsistent.

Also, the fact that consistency in first-order logic is a property of finite character is a consequence of the finiteness of proofs in the formal system.

Theorem 5.2.25. *$\mathcal{L}$-consistency is a property of finite character, for every first-order language $\mathcal{L}$.*

Proof. The argument is similar to the proof of Theorem 3.2.7. □

Theorem 5.2.26. *Let $\mathcal{L}$ be a first-order language. If Γ is a satisfiable set of $\mathcal{L}$-formulas, then Γ is $\mathcal{L}$-consistent.*

Proof. The argument is entirely similar to the argument of Theorem 3.2.8. □

Theorem 5.2.27. *If Γ is an $\mathcal{L}$-consistent set of $\mathcal{L}$-formulas and $\Gamma \vdash_{\mathcal{HF}^{\mathcal{L}}} \varphi$, then $\Gamma \cup \{\varphi\}$ is also $\mathcal{L}$-consistent. Thus, if $\Gamma \cup \{\varphi\}$ is $\mathcal{L}$-inconsistent and $\Gamma \vdash_{\mathcal{HF}^{\mathcal{L}}} \varphi$, then Γ is $\mathcal{L}$-inconsistent.*

Proof. The proof is identical to the proof of Theorem 3.2.9. □

Corollary 5.2.28. *Let Γ be a maximally $\mathcal{L}$-consistent set of formulas. If $\Gamma \vdash_{\mathcal{HF}^{\mathcal{L}}} \varphi$, then $\varphi \in \Gamma$.*

Proof. This follows immediately from Theorem 5.2.27. □

Theorem 5.2.29. *If Γ is an $\mathcal{L}$-inconsistent set of formulas, then for every $\mathcal{L}$-formula ψ, $\Gamma \vdash_{\mathcal{HF}^{\mathcal{L}}} \psi$.*

Proof. The statement can be shown using the argument of Theorem 3.2.11. □

Starting from proofs that show $\Gamma \vdash_{\mathcal{HF}^{\mathcal{L}}} \varphi$ and $\Gamma \vdash_{\mathcal{HF}^{\mathcal{L}}} (\neg\varphi)$, we can construct effectively the proof that shows $\Gamma \vdash_{\mathcal{HF}^{\mathcal{L}}} \psi$, using the steps outlined in the proof of Theorem 3.2.11. (We will revisit this effective construction in Section 5.7.1.)

Theorem 5.2.30. *Let Γ be a set of $\mathcal{L}$-formulas and let φ be an $\mathcal{L}$-formula such that $\Gamma \nvdash_{\mathcal{HF}^{\mathcal{L}}} \varphi$. Then, $\Gamma \cup \{(\neg\varphi)\}$ is $\mathcal{L}$-consistent.*

Proof. Again, the argument is identical to argument of Theorem 3.2.12. □

Corollary 5.2.31. *Let Γ be an $\mathcal{L}$-consistent set of $\mathcal{L}$-formulas. Then, for each $\mathcal{L}$-formula φ, at least one of $\Gamma \cup \{\varphi\}$ and $\Gamma \cup \{(\neg\varphi)\}$ is $\mathcal{L}$-consistent. Therefore, if both $\Gamma \cup \{\varphi\}$ and $\Gamma \cup \{(\neg\varphi)\}$ are $\mathcal{L}$-inconsistent, then so is Γ.*

If Γ is maximally $\mathcal{L}$-consistent and φ is an $\mathcal{L}$-formula, then exactly one of the formulas φ and $(\neg\varphi)$ belongs to Γ.

Proof. See the proof of Corollary 3.2.13. □

The next statement describes a method of derivation in $\mathcal{HF}^{\mathcal{L}}$ known as *reductio ad absurdum*.

Theorem 5.2.32. *Let Γ be a set of $\mathcal{L}$-formulas and let φ be an $\mathcal{L}$-formula such that $\Gamma \cup \{\varphi\}$ is $\mathcal{L}$-inconsistent. Then, $\Gamma \vdash_{\mathcal{HF}^{\mathcal{L}}} (\neg\varphi)$.*

If $\Gamma \cup \{(\neg\varphi)\}$ is $\mathcal{L}$-inconsistent, then $\Gamma \vdash_{\mathcal{HF}^{\mathcal{L}}} \varphi$.

Proof. By Theorem 5.2.29, the inconsistency of $\Gamma \cup \{\varphi\}$ means that we have $\Gamma \cup \{\varphi\} \vdash_{\mathcal{HF}^{\mathcal{L}}} (\neg\varphi)$. By the Deduction Theorem, $\Gamma \vdash_{\mathcal{HF}^{\mathcal{L}}} (\varphi \to (\neg\varphi))$. Observe, that $((\varphi \to (\neg\varphi)) \to (\neg\varphi))$ is tautology of first-order logic by Example 2.6.9, so, by Theorem 5.2.3, $\Gamma \vdash_{\mathcal{HF}^{\mathcal{L}}} ((\varphi \to (\neg\varphi)) \to (\neg\varphi))$. By R_{mp}, we obtain $\Gamma \vdash_{\mathcal{HF}^{\mathcal{L}}} (\neg\varphi)$.

For the second part of the theorem, note that the inconsistency of $\Gamma \cup \{(\neg\varphi)\}$ implies the existence of the proof showing $\Gamma \vdash_{\mathcal{HF}^{\mathcal{L}}} (\neg(\neg\varphi))$ by the first part of the theorem. By Tautological Consequence, we obtain the desired proof for the second part. □

The proof of Theorem 5.2.32 shows that given proofs that show $\Gamma \cup \{\varphi\} \vdash_{\mathcal{HF}^{\mathcal{L}}} \psi$ and $\Gamma \cup \{\varphi\} \vdash_{\mathcal{HF}^{\mathcal{L}}} (\neg\psi)$, we can obtain effectively a proof for $\Gamma \vdash_{\mathcal{HF}^{\mathcal{L}}} (\neg\varphi)$. A similar effectivization holds for the second part of the theorem.

Example 5.2.33. We illustrate the application of several proving techniques in showing that $\vdash_{\mathcal{HF}^{\mathcal{L}}} ((\neg(\forall x)\varphi) \to (\exists x)(\neg\varphi))$, for every $\mathcal{L}$-formula φ and variable x. By Axiom Group 20, we have $\{(\neg(\exists x)(\neg\varphi)), (\neg\varphi)\} \vdash_{\mathcal{HF}^{\mathcal{L}}} (\exists x)(\neg\varphi)$. We also have obviously $\{(\neg(\exists x)(\neg\varphi)), (\neg\varphi)\} \vdash_{\mathcal{HF}^{\mathcal{L}}} (\neg(\exists x)(\neg\varphi))$, which shows that the set $\{(\neg(\exists x)(\neg\varphi)), (\neg\varphi)\}$ is inconsistent. By *reductio ad absurdum* (Theorem 5.2.32), we have $\{(\neg(\exists x)(\neg\varphi))\} \vdash_{\mathcal{HF}^{\mathcal{L}}} \varphi$. Since x is not a free variable in the formula $(\neg(\exists x)(\neg\varphi))$, by the Generalization Theorem, we obtain, $(\neg(\exists x)(\neg\varphi)) \vdash_{\mathcal{HF}^{\mathcal{L}}} (\forall x)\varphi$. An application of the Deduction Theorem yields $\vdash_{\mathcal{HF}^{\mathcal{L}}} ((\neg(\exists x)(\neg\varphi)) \to (\forall x)\varphi)$. By a new application of tautological consequence, we get $\vdash_{\mathcal{HF}^{\mathcal{L}}} ((\neg(\forall x)\varphi) \to (\exists x)(\neg\varphi))$.

By a similar argument, using the inconsistency of the set $\{(\neg(\exists x)\varphi), \varphi\}$, we can show that $\vdash_{\mathcal{HF}^{\mathcal{L}}} ((\neg(\exists x)\varphi) \to (\forall x)(\neg\varphi))$.

Since all steps involved in this example are effective, it follows that we can effectively find proofs that show that $\vdash_{\mathcal{HF}^{\mathcal{L}}} ((\neg(\forall x)\varphi) \to (\exists x)(\neg\varphi))$, and $\vdash_{\mathcal{HF}^{\mathcal{L}}} ((\neg(\exists x)\varphi) \to (\forall x)(\neg\varphi))$, given an $\mathcal{L}$-formula φ and a variable x. ▯

If we were to follow our presentation of completeness of $\mathcal{HF}$ in propositional logic further, the next result would be that every maximally $\mathcal{L}$-consistent set is an $(\mathcal{L}, \mathrm{VAR})$-truth set. However, the proof of the propositional analogue of this statement (Theorem 3.2.18) does not adapt to first-order logic because $\boldsymbol{\gamma}$-formulas have infinite constituents. Therefore, we need to introduce a special property of sets of formulas that together with maximal consistency allows the proof of Theorem 3.2.18 to go through in first-order logic.

Definition 5.2.34. Let $\mathcal{L}$ be a first-order language. A set of $\mathcal{L}$-formulas Γ is an *$\mathcal{L}$-Henkin set*[1] if, for every $\mathcal{L}$-formula φ and variable x, there are constant symbols c, d in $\mathcal{L}$ such that $(\langle\varphi\rangle_{x:=c} \to (\forall x)\varphi)$ and $((\exists x)\varphi \to \langle\varphi\rangle_{x:=d})$ both belong to Γ. $\square$

Theorem 5.2.35. *A maximally $\mathcal{L}$-consistent set of $\mathcal{L}$-formulas Γ that is an $\mathcal{L}$-Henkin set is an $(\mathcal{L}$, VAR$)$-truth set for every first-order language $\mathcal{L}$.*

Proof. The maximal $\mathcal{L}$-consistency of Γ implies that for every $\mathcal{L}$-formula φ, $(\neg\varphi) \in \Gamma$ if and only if $\varphi \notin \Gamma$ by Corollary 5.2.31.

If $\mathcal{L}$ contains the equality symbol and $\alpha \in \mathrm{INST}_{\mathcal{L},\mathrm{VAR}}(\mathrm{Eq}_{=,\mathcal{L}})$, then α is an axiom of $\mathcal{HF}^{\mathcal{L}}$ and therefore $\Gamma \vdash_{\mathcal{HF}^{\mathcal{L}}} \alpha$, which implies α in Γ, by Corollary 5.2.28. Thus, by Theorem 4.12.32, in order to prove that Γ is a truth set, it remains to show that for every positive or negated positive formula φ that is not a literal, if a constituent K of φ is included in Γ, then $\varphi \in \Gamma$.

Let φ be a positive or negated positive formula that is not a literal and let K be an $(\mathcal{L}, \mathrm{VAR})$-constituent of φ such that $K \subseteq \Gamma$. If φ is not a $\boldsymbol{\gamma}$-formula, then $\varphi \in \Gamma$ by an argument similar to the one used in Theorem 3.2.18. (Note that if φ is the $\boldsymbol{\delta}$-formula $(\neg(\forall x)\psi)$, then the formula $((\neg\langle\psi\rangle_{x:=t}) \to (\neg(\forall x)\psi))$ follows by Tautological Consequence from the axiom instance $((\forall x)\psi \to \langle\psi\rangle_{x:=t})$.)

Suppose that φ is the $\boldsymbol{\gamma}$-formula $(\forall x)\psi$ and therefore, $K = \{\langle\psi\rangle_{x:=t} \mid t \in \mathrm{TERM}_{\mathcal{L}}\} \subseteq \Gamma$. Since Γ is an $\mathcal{L}$-Henkin set, there is a constant symbol c such that $(\langle\psi\rangle_{x:=c} \to \varphi) \in \Gamma$. Since $\langle\psi\rangle_{x:=c} \in K \subseteq \Gamma$, we have $\Gamma \vdash_{\mathcal{HF}^{\mathcal{L}}} \langle\psi\rangle_{x:=c}$ and this, by modus ponens, gives $\Gamma \vdash_{\mathcal{HF}^{\mathcal{L}}} \varphi$. Thus, $\varphi \in \Gamma$, by Corollary 5.2.28.

Finally, let φ be the $\boldsymbol{\gamma}$-formula $(\neg(\exists x)\psi)$. We have $K = \{\langle(\neg\psi)\rangle_{x:=t} \mid t \in \mathrm{TERM}_{\mathcal{L}}\} = \{(\neg\langle\psi\rangle_{x:=t}) \mid t \in \mathrm{TERM}_{\mathcal{L}}\}$, because of

[1]Leon Albert Henkin was born in Brooklyn in 1921, got his Ph.D. in mathematics from Princeton University in 1947, and died in 2006. He was a Professor at the University of California-Berkeley. Henkin is known for his contributions in algebraic logic and mathematics education and for what is known today as the standard proof of completeness for Hilbert/Frege systems of first-order logic, which he developed in his doctoral dissertation.

Theorem 4.6.50. Also, we have $K \subseteq \Gamma$. Since Γ is an $\mathcal{L}$-Henkin set, there is a constant symbol $d \in \mathcal{L}$ such that $((\exists x)\psi \rightarrow \langle\psi\rangle_{x:=d}) \in \Gamma$. By tautological consequence, $\Gamma \vdash_{\mathcal{HF}^{\mathcal{L}}} ((\neg\langle\psi\rangle_{x:=d}) \rightarrow (\neg(\exists x)\psi))$. Theorem 4.6.50 implies that $\Gamma \vdash_{\mathcal{HF}^{\mathcal{L}}} (\langle(\neg\psi)\rangle_{x:=d} \rightarrow (\neg(\exists x)\psi))$. Since $\langle(\neg\psi)\rangle_{x:=d} \in K \subseteq \Gamma$, an application of modus ponens yields $\Gamma \vdash_{\mathcal{HF}^{\mathcal{L}}} (\neg(\exists x)\psi) = \varphi$, which implies $\varphi \in \Gamma$ by Corollary 5.2.28. $\square$

To conclude the completeness argument for first-order logic in a manner similar to that of propositional logic, we need to show that every $\mathcal{L}$-consistent set Γ of $\mathcal{L}$-formulas is included in a maximally $\mathcal{L}'$-consistent set of formulas Γ' that is an $\mathcal{L}'$-Henkin set of formulas, where $\mathcal{L}'$ is a suitable extension of $\mathcal{L}$. Since the satisfiability of such a set Γ' is implied by Theorem 5.2.35, this would imply the satisfiability of Γ and therefore the completeness result. The difficulty in proving this result resides in the fact that the Henkin property is not a property of finite character. We must show that we can enlarge Γ to be inclusive enough to become a Henkin set without losing consistency.

The steps needed to establish this result are:

- If Γ is an $\mathcal{L}$-consistent set of $\mathcal{L}$-formulas, then Γ is an $\mathcal{L}^c$-consistent set of $\mathcal{L}^c$-formulas (Theorem 5.2.36).
- If Γ is an $\mathcal{L}$-consistent set of $\mathcal{L}$-formulas and there are infinitely many constant symbols in $\mathcal{L}$ that do not appear in Γ, then there is an $\mathcal{L}$-consistent, $\mathcal{L}$-Henkin set of $\mathcal{L}$-formulas that contains Γ (Theorem 5.2.37).
- Every $\mathcal{L}$-consistent set of $\mathcal{L}$-formulas is contained in a maximally $\mathcal{L}$-consistent set of $\mathcal{L}$-formulas (Theorem 5.2.38).

Once these results are established, we show that every $\mathcal{L}$-consistent set Γ of $\mathcal{L}$-formulas is contained in a maximally $\mathcal{L}^c$-consistent set of $\mathcal{L}^c$-formulas that is an $\mathcal{L}^c$-Henkin set as follows. By Theorem 5.2.36, Γ is an $\mathcal{L}^c$-consistent set of $\mathcal{L}^c$-formulas. Therefore, by Theorem 5.2.37, there is an $\mathcal{L}^c$-consistent, $\mathcal{L}^c$-Henkin set of formulas Γ_1 that includes Γ. In turn, by Theorem 5.2.38, there is a maximally $\mathcal{L}^c$-consistent set of $\mathcal{L}^c$-formulas Γ_2 such that $\Gamma_1 \subseteq \Gamma_2$. Because Γ_2 contains an $\mathcal{L}^c$-Henkin set, it is itself such a set.

Theorem 5.2.36. *If Γ is an $\mathcal{L}$-consistent set of $\mathcal{L}$-formulas, then Γ is an $\mathcal{L}^c$-consistent set of $\mathcal{L}^c$-formulas.*

Proof. We prove the contrapositive of the statement. Suppose that Γ is inconsistent as a set of $\mathcal{L}^c$-formulas. Let φ be an arbitrary $\mathcal{L}$-formula, which is obviously also an $\mathcal{L}^c$-formula. By Theorem 5.2.29, we have both $\Gamma \vdash_{\mathcal{HF}^{\mathcal{L}^c}} \varphi$ and $\Gamma \vdash_{\mathcal{HF}^{\mathcal{L}^c}} (\neg\varphi)$. By Theorem 5.2.21, we have $\Gamma \vdash_{\mathcal{HF}^{\mathcal{L}}} \varphi$ and $\Gamma \vdash_{\mathcal{HF}^{\mathcal{L}}} (\neg\varphi)$, so Γ is inconsistent as a set of $\mathcal{L}$-formulas. □

Theorem 5.2.37. *If Γ is an $\mathcal{L}$-consistent set of $\mathcal{L}$-formulas and there are infinitely many constant symbols in $\mathcal{L}$ that do not appear in Γ, then there is an $\mathcal{L}$-consistent, $\mathcal{L}$-Henkin set of $\mathcal{L}$-formulas Γ' that contains Γ.*

Proof. Suppose that $\{(\varphi_0, y_0, Q_0), (\varphi_1, y_1, Q_1), \ldots\}$ is a list of all triples of the form (φ, y, Q) such that φ is an $\mathcal{L}$-formula, y is a variable and Q is a quantifier symbol.

Define the sequence $\Gamma_0, \Gamma_1, \ldots$ of sets of $\mathcal{L}$-formulas as follows:

$$\Gamma_0 = \Gamma$$

$$\Gamma_{n+1} = \begin{cases} \Gamma_n \cup \{(\langle\varphi_n\rangle_{y_n:=e_n} \to (\forall y_n)\varphi_n)\} & \text{if } Q_n = \forall \\ \Gamma_n \cup \{((\exists y_n)\varphi_n \to \langle\varphi_n\rangle_{y_n:=e_n})\} & \text{if } Q_n = \exists, \end{cases}$$

where e_n is the first constant symbol in the standard order that does not occur in Γ_n or in φ_n. Note that e_n exists because $\Gamma_n - \Gamma$ is finite and by hypothesis there are infinitely many constant symbols of $\mathcal{L}$ that do not occur in Γ.

We prove by induction on n that each set Γ_n is $\mathcal{L}$-consistent. The basis step is immediate by hypothesis. Suppose that Γ_n is $\mathcal{L}$-consistent but Γ_{n+1} is $\mathcal{L}$-inconsistent. We consider the following two cases depending on Q_n.

`Case 1:` $Q_n = \forall$. The inconsistency of Γ_{n+1} implies that $\Gamma_n \vdash_{\mathcal{HF}^{\mathcal{L}}} (\neg(\langle\varphi_n\rangle_{y_n:=e_n} \to (\forall y_n)\varphi_n))$, by Theorem 5.2.32. Since the formulas

$$((\neg(\alpha \to \beta)) \to \alpha) \text{ and } ((\neg(\alpha \to \beta)) \to (\neg\beta))$$

are easily seen to be tautologies, applying twice the rule $TC_1^{\mathcal{L}}$, we have $\Gamma_n \vdash_{\mathcal{HF}^{\mathcal{L}}} \langle\varphi_n\rangle_{y_n:=e_n}$ and $\Gamma_n \vdash_{\mathcal{HF}^{\mathcal{L}}} (\neg(\forall y_n)\varphi_n)$. Since e_n is substitutable for y_n in φ_n, we actually have $\Gamma_n \vdash_{\mathcal{HF}^{\mathcal{L}}} (\varphi_n)_{y_n:=e_n}$. By choice of e_n and Theorem 5.2.22, we have $\Gamma_n \vdash_{\mathcal{HF}^{\mathcal{L}}} (\forall y_n)\varphi_n$, which contradicts the consistency of Γ_n.

`Case 2:` $Q_n = \exists$. Again, the inconsistency of Γ_{n+1} implies that $\Gamma_n \vdash_{\mathcal{HF}^{\mathcal{L}}} (\neg((\exists y_n)\varphi_n \to \langle\varphi_n\rangle_{y_n:=e_n}))$. A double application of $TC_1^{\mathcal{L}}$ gives $\Gamma_n \vdash_{\mathcal{HF}^{\mathcal{L}}} (\exists y_n)\varphi_n$ and $\Gamma_n \vdash_{\mathcal{HF}^{\mathcal{L}}} (\neg(\varphi_n)_{y_n:=e_n})$. Clearly, $(\neg(\varphi_n)_{y_n:=e_n})$ equals the formula $((\neg\varphi_n))_{y_n:=e_n}$, so $\Gamma_n \vdash_{\mathcal{HF}^{\mathcal{L}}} ((\neg\varphi_n))_{y_n:=e_n}$. By choice of e_n and Theorem 5.2.22, we have $\Gamma_n \vdash_{\mathcal{HF}^{\mathcal{L}}} (\forall y_n)(\neg\varphi_n)$. By modus ponens and axiom group 23, we arrive at $\Gamma_n \vdash_{\mathcal{HF}^{\mathcal{L}}} (\neg(\exists y_n)\varphi_n)$, which contradicts the consistency of Γ_n.

Since $\mathcal{L}$-consistency is a property of finite character, it follows from Lemma 1.3.2 that the set $\Gamma' = \bigcup_{n\in\mathbf{N}} \Gamma_n$ is $\mathcal{L}$-consistent. It is clear that $\Gamma \subseteq \Gamma'$ and Γ' is an $\mathcal{L}$-Henkin set by construction. □

The last piece of the puzzle is discussed next.

Theorem 5.2.38. *Every $\mathcal{L}$-consistent set of $\mathcal{L}$-formulas is contained in a maximally $\mathcal{L}$-consistent set of $\mathcal{L}$-formulas.*

Proof. The statement follows from Theorem 1.3.3 since $\mathcal{L}$-consistency is a property of finite character. □

Theorem 5.2.39. *Let $\mathcal{L}$ be a first-order language. If Γ is an $\mathcal{L}$-consistent set of $\mathcal{L}$-formulas, then Γ is satisfiable.*

Proof. By Theorem 5.2.36, Γ is an $\mathcal{L}^c$-consistent set of $\mathcal{L}^c$-formulas. Since Γ is a set of $\mathcal{L}$-formulas, there are infinitely many constant symbols in $\mathcal{L}^c$ that do not appear in Γ. Therefore, by Theorem 5.2.37, there is an $\mathcal{L}^c$-consistent, $\mathcal{L}^c$-Henkin set of formulas Γ_1 that includes Γ. By Theorem 5.2.38, there is a maximally $\mathcal{L}^c$-consistent set of $\mathcal{L}^c$-formulas Γ_2 such that $\Gamma_1 \subseteq \Gamma_2$. Γ_2 is an $\mathcal{L}^c$-Henkin set because it contains an $\mathcal{L}^c$-Henkin set. Consequently, by Theorem 5.2.35, Γ_2 is an $(\mathcal{L}^c, \mathrm{VAR})$-truth set. By Corollary 4.12.36, Γ_2 is satisfiable, which implies that Γ is satisfiable because $\Gamma \subseteq \Gamma_2$□

Corollary 5.2.40. *Let $\mathcal{L}$ be a first-order language. A set of $\mathcal{L}$-formulas Γ is $\mathcal{L}$-consistent if and only if it is satisfiable.*

Proof. This statement follows from Theorems 5.2.26 and 5.2.39. □

Theorem 5.2.41 (Completeness of $\mathcal{HF}_\Gamma^{\mathcal{L}}$). *Let $\mathcal{L}$ be a first-order language, Γ be a set of $\mathcal{L}$-formulas and let φ be an $\mathcal{L}$-formula. If $\Gamma \models \varphi$, then $\Gamma \vdash_{\mathcal{HF}^{\mathcal{L}}} \varphi$.*

Proof. If $\Gamma \not\vdash_{\mathcal{HF}^{\mathcal{L}}} \varphi$, then, by Theorem 5.2.30, $\Gamma \cup \{(\neg\varphi)\}$ is $\mathcal{L}$-consistent, so, by Theorem 5.2.39, $\Gamma \cup \{(\neg\varphi)\}$ is satisfiable. This implies that $\Gamma \not\models \varphi$, by Part 1 of Theorem 4.5.52. □

Corollary 5.2.42. *Let $\mathcal{L}$ be a first-order language, Γ be a set of $\mathcal{L}$-formulas and let φ be an $\mathcal{L}$-formula. Then, $\Gamma \models \varphi$ if and only if $\Gamma \vdash_{\mathcal{HF}^{\mathcal{L}}} \varphi$.*

Proof. This follows immediately from Theorems 5.2.2 and 5.2.41. □

5.2.2 *Building Proofs in $\mathcal{HF}^{\mathcal{L}}$*

Theorem 5.2.43. *Let $\mathcal{L}$ be a first-order language and φ, ψ be $\mathcal{L}$-formulas. We have*

$$(\varphi \leftrightarrow \psi) \vdash_{\mathcal{HF}^{\mathcal{L}}} (\varphi \rightarrow \psi),$$
$$(\varphi \leftrightarrow \psi) \vdash_{\mathcal{HF}^{\mathcal{L}}} (\psi \rightarrow \varphi),$$
$$\{(\varphi \rightarrow \psi), (\psi \rightarrow \varphi)\} \vdash_{\mathcal{HF}} (\varphi \leftrightarrow \psi).$$

Furthermore, these proofs can be found effectively, given φ and ψ.

Proof. The argument is identical to the proofs of the same results in the propositional case given in Lemmas 3.2.25 and 3.2.26. □

Corollary 5.2.44. *Let $\mathcal{L}$ be a first-order language and let $\varphi_0, \ldots, \varphi_{n-1}$ be $\mathcal{L}$-formulas, where $n \geq 2$, and let Γ be a set of $\mathcal{L}$-formulas. Starting from proofs $\Gamma \vdash_{\mathcal{HF}^{\mathcal{L}}} (\varphi_i \leftrightarrow \varphi_{i+1})$ for $0 \leq i \leq n-2$, one can produce effectively a proof for $\Gamma \vdash_{\mathcal{HF}^{\mathcal{L}}} (\varphi_0 \leftrightarrow \varphi_{n-1})$.*

Proof. This statement follows from Theorem 5.2.43 and the first-order analogues of Theorem 3.2.27 and Corollary 3.2.28. □

Theorem 5.2.45. *Let $\mathcal{L}$ be a first-order language, Γ be a set of $\mathcal{L}$-formulas such that the variable x does not occur free in Γ and let φ, ψ be two $\mathcal{L}$-formulas, such that $\Gamma \vdash_{\mathcal{HF}^{\mathcal{L}}} (\varphi \rightarrow \psi)$. Then, we have $\Gamma \vdash_{\mathcal{HF}^{\mathcal{L}}} ((\forall x)\varphi \rightarrow (\forall x)\psi)$ and $\Gamma \vdash_{\mathcal{HF}^{\mathcal{L}}} ((\exists x)\varphi \rightarrow (\exists x)\psi)$.*

Proof. For the first part of the theorem, we start from the fact that $\Gamma \cup \{(\forall x)\varphi\} \vdash_{\mathcal{HF}^{\mathcal{L}}} ((\forall x)\varphi \rightarrow \varphi)$, by Axiom Group 19. Further, since $\Gamma \cup \{(\forall x)\varphi\} \vdash_{\mathcal{HF}^{\mathcal{L}}} (\forall x)\varphi$, by modus ponens, we have $\Gamma \cup \{(\forall x)\varphi\} \vdash_{\mathcal{HF}^{\mathcal{L}}} \varphi$. By hypothesis, $\Gamma \cup \{(\forall x)\varphi\} \vdash_{\mathcal{HF}^{\mathcal{L}}} (\varphi \rightarrow \psi)$, which yields $\Gamma \cup \{(\forall x)\varphi\} \vdash_{\mathcal{HF}^{\mathcal{L}}} \psi$ by modus ponens. Since x does not occur free in $\Gamma \cup \{(\forall x)\varphi\}$, by the Generalization Theorem, we have $\Gamma \cup \{(\forall x)\varphi\} \vdash_{\mathcal{HF}^{\mathcal{L}}} (\forall x)\psi$. A final application of the Deduction Theorem produces $\Gamma \vdash_{\mathcal{HF}^{\mathcal{L}}} ((\forall x)\varphi \to (\forall x)\psi)$.

For the second part, the starting point was obtained in Example 5.2.33: $\Gamma \vdash_{\mathcal{HF}^{\mathcal{L}}} ((\neg(\exists x)\psi) \to (\forall x)(\neg\psi))$. From the hypothesis, by Tautological Consequence, we have $\Gamma \vdash_{\mathcal{HF}^{\mathcal{L}}} ((\neg\psi) \to (\neg\varphi))$. By the first part of the theorem, $\Gamma \vdash_{\mathcal{HF}^{\mathcal{L}}} ((\forall x)(\neg\psi) \to (\forall x)(\neg\varphi))$. By Axiom Group 23, we have $\Gamma \vdash_{\mathcal{HF}^{\mathcal{L}}} ((\forall x)(\neg\varphi) \to (\neg(\exists x)\varphi))$. By Tautological Consequence, we obtain, $\Gamma \vdash_{\mathcal{HF}^{\mathcal{L}}} ((\neg(\exists x)\varphi) \to (\neg(\exists x)\varphi))$. A last application of Tautological Consequence produces $\Gamma \vdash_{\mathcal{HF}^{\mathcal{L}}} ((\exists x)\varphi \to (\exists x)\psi)$. □

All steps involved in the proof of Theorem 5.2.45 being effective, it follows that we can construct effectively proofs showing $\Gamma \vdash_{\mathcal{HF}^{\mathcal{L}}} ((\forall x)\varphi \to (\forall x)\psi)$ and $\Gamma \vdash_{\mathcal{HF}^{\mathcal{L}}} ((\exists x)\varphi \to (\exists x)\psi)$, given a proof showing that $\Gamma \vdash_{\mathcal{HF}^{\mathcal{L}}} (\varphi \to \psi)$.

Example 5.2.46. We show that $\vdash_{\mathcal{HF}^{\mathcal{L}}} (\exists x)(P(x) \to (\forall x)P(x))$. The sequence of steps of the argument is given below.

(1) $((\neg(\forall x)P(x)) \vee (\forall x)P(x))$
(tautology)

(2) $((\forall x)P(x) \to (P(x) \to (\forall x)P(x)))$
(Axiom Group 1)

(3) $((P(x) \to (\forall x)P(x)) \to (\exists x)(P(x) \to (\forall x)P(x)))$
(Axiom Group 20)

(4) $((\forall x)P(x) \to (\exists x)(P(x) \to (\forall x)P(x)))$
(tautological consequence of (2) and (3))

(5) $((\neg(\forall x)P(x)) \to (\exists x)(\neg P(x)))$
(by Example 5.2.33)

(6) $((\neg P(x)) \to (P(x) \to (\forall x)P(x)))$
(Axiom Group 10)

(7) $((\exists x)(\neg P(x)) \to (\exists x)(P(x) \to (\forall x)P(x)))$
(by Theorem 5.2.45 with $\Gamma = \emptyset$ and (6))

(8) $((\neg(\forall x)P(x)) \to (\exists x)(P(x) \to (\forall x)P(x)))$
(tautological consequence of (5) and (7))

(9) $(\exists x)(P(x) \to (\forall x)P(x))$
(tautological consequence of (1), (4) and (8)).

□

Example 5.2.47. Let R be a binary relation symbol in a first-order language $\mathcal{L}$. We prove that $\vdash_{\mathcal{HF}^{\mathcal{L}}} ((\exists x)(\forall y)R(x,y) \to (\forall y)(\exists x)R(x,y))$ using the following sequence:

(1) $\vdash_{\mathcal{HF}^{\mathcal{L}}} ((\forall y)R(x,y) \to R(x,y))$
(Axiom Group 19)

(2) $\vdash_{\mathcal{HF}^{\mathcal{L}}} ((\exists x)(\forall y)R(x,y) \to (\exists x)R(x,y))$
(by Theorem 5.2.45 with $\Gamma = \emptyset$ and (1))

(3) $(\exists x)(\forall y)R(x,y) \vdash_{\mathcal{HF}^{\mathcal{L}}} (\exists x)R(x,y)$
(by Corollary 5.2.13 and (2))

(4) $(\exists x)(\forall y)R(x,y) \vdash_{\mathcal{HF}^{\mathcal{L}}} (\forall y)(\exists x)R(x,y)$
(by the Generalization Theorem, since y does not occur free in $(\exists x)(\forall y)R(x,y)$ and (3))

(5) $\vdash_{\mathcal{HF}^{\mathcal{L}}} ((\exists x)(\forall y)R(x,y) \to (\forall y)(\exists x)R(x,y))$
(by the Deduction Theorem and (4)).

□

The next theorem is the syntactic equivalent of the Replacement Theorem (Theorem 4.6.16).

Lemma 5.2.48. *Let $\mathcal{L}$ be a first-order language. The following tautologies have proofs in the formal system $\mathcal{HF}^{\mathcal{L}}$ that can be found effectively given the $\mathcal{L}$-formulas $\theta_0, \theta_1, \theta_0'$ and the binary connective symbol C:*

$$((\theta_0 \leftrightarrow \theta_0') \to ((\theta_0 C \theta_1) \leftrightarrow (\theta_0' C \theta_1)))$$
$$((\theta_0 \leftrightarrow \theta_0') \to ((\theta_1 C \theta_0) \leftrightarrow (\theta_1 C \theta_0')))$$

In addition, for all θ_0, θ_0', we can find effectively a proof for $((\theta_0 \leftrightarrow \theta_0') \to ((\neg\theta_0) \leftrightarrow (\neg\theta_0')))$.

Proof. The argument parallels the one of Lemma 3.2.31. □

Theorem 5.2.49. *Let $\mathcal{L}$ be a first-order language. If the $\mathcal{L}$-formulas α, β are provably equivalent in $\mathcal{HF}^{\mathcal{L}}$ and ψ is a obtained from the $\mathcal{L}$-formula φ by replacing an occurrence of α by β, then φ and ψ are provably equivalent in $\mathcal{HF}^{\mathcal{L}}$.*

Proof. First note that in the special case when α coincides with φ, we have $\psi = \beta$ and thus φ and ψ are clearly provably equivalent.

The argument is by induction on the formula φ and proceeds along the same lines as the argument of Theorem 3.2.32 except for the case when $\varphi = (Qx)\varphi_0$, where Q is a quantifier symbol. If we are not in the special case, the occurrence of the formula α is located within φ_0. Let φ_0' be the formula obtained from φ_0 by replacing the occurrence of α by β. Then, $\psi = (Qx)\varphi_0'$. By the inductive hypothesis, the formulas φ_0 and φ_0' are provably equivalent and therefore, by Theorem 5.2.43, the formulas $(\varphi_0 \to \varphi_0')$ and $(\varphi_0' \to \varphi_0)$ are theorems of $\mathcal{HF}^{\mathcal{L}}$. Applying Theorem 5.2.45, it follows that both $((Qx)\varphi_0 \to (Qx)\varphi_0')$ and $((Qx)\varphi_0' \to (Qx)\varphi_0)$ are also theorems. Again by Theorem 5.2.43, we obtain the theorem $((Qx)\varphi_0 \leftrightarrow (Qx)\varphi_0')$ and this is $(\varphi \leftrightarrow \psi)$. $\square$

The argument of Theorem 5.2.49 shows that starting from an $\mathcal{L}$-formula φ, two provably equivalent $\mathcal{L}$-formulas α and β and a proof of $(\alpha \leftrightarrow \beta)$, and an occurrence of α in φ, we can effectively find a proof of $(\varphi \leftrightarrow \psi)$, where ψ is the formula obtained from φ by replacing the occurrence of α by β. This holds because we have seen that the proofs provided by the theorems used in the argument can be found effectively.

Lemma 5.2.50. *Let $\mathcal{L}$ be a first-order language and let φ, ψ be two $\mathcal{L}$-formulas such that ψ is an immediate variant of φ. Then, φ and ψ are provably equivalent in $\mathcal{HF}^{\mathcal{L}}$.*

Proof. If $\varphi = \psi$, then by Axiom Group 3 and Theorem 5.2.43, we can prove $(\varphi \leftrightarrow \varphi)$.

Suppose that ψ is obtained from φ by replacing an occurrence of $(Qx)\beta$ by $(Qy)(\beta)_{x:=y}$, where $y \notin \mathtt{FV}(\beta)$ and y is substitutable for x in β. Then, by Corollary 5.2.18, the formulas $(Qx)\beta$ and $(Qy)(\beta)_{x:=y}$ are provably equivalent. Therefore, by Theorem 5.2.49, the formulas φ and ψ are provably equivalent. $\square$

Given two immediate variants φ and ψ, the argument of Lemma 5.2.50 shows that we can effectively find a proof of the formula $(\varphi \leftrightarrow \psi)$.

Theorem 5.2.51. *Let $\mathcal{L}$ be a first-order language. Given two $\mathcal{L}$-formulas φ, ψ such that ψ is a variant of φ, we can effectively find a proof in $\mathcal{HF}^{\mathcal{L}}$ of the formula $(\varphi \leftrightarrow \psi)$.*

Proof. By Theorem 4.6.36, we can effectively find a sequence of formulas $(\theta_0, \ldots, \theta_{n-1})$ such that $\varphi = \theta_0$, $\psi = \theta_{n-1}$, and θ_{i+1} is an immediate variant of θ_i, for $0 \leq i \leq n-2$. By Lemma 5.2.50, we can find effectively proofs for the formulas $(\theta_i \leftrightarrow \theta_{i+1})$, for $0 \leq i \leq n-2$.

By Corollary 5.2.44, starting from these proofs, we can obtain effectively a proof of $(\varphi \leftrightarrow \psi)$. □

Theorem 5.2.52. *Let $\mathcal{L}$ be a first-order language. The rule $R^{\mathcal{L}}_{var}$ given by*

$$\frac{\varphi}{\varphi'},$$

where φ, φ' are $\mathcal{L}$-formulas such that φ' is a variant of φ, is a derived rule of $\mathcal{HF}^{\mathcal{L}}$ called the variant rule for $\mathcal{L}$. *Furthermore, we can effectively eliminate each application of this rule in the augmented formal system obtained by adding this rule.*

Proof. Let φ' be a variant of φ. By Theorem 5.2.51, we can find effectively a proof of $(\varphi \leftrightarrow \varphi')$ and therefore by Theorem 5.2.43 a proof of $\varphi \vdash_{\mathcal{HF}^{\mathcal{L}}} (\varphi \to \varphi')$. By Modus Ponens, we obtain a proof of $\varphi \vdash_{\mathcal{HF}^{\mathcal{L}}} \varphi'$. □

5.3 First-Order Tableaux

In propositional logic, tableaux are used to determine the satisfiability of sets of formulas by attempting to construct Hintikka sets that contain these sets of formulas. The existence of such Hintikka sets for satisfiable sets of formulas is assured by Corollary 2.7.22 (for sets of unsigned formulas) and Exercise 116 of Chapter 2 (for sets of signed formulas). Once a Hintikka set that contains the original set of formulas is found, Theorem 2.7.16 or Theorem 2.7.28 can be used to determine a satisfying truth assignment. If no Hintikka set is found through a systematic search, then we can conclude that the original set of formulas is unsatisfiable.

In first-order logic, a tableau for a set of $(\mathcal{L}, V)$-formulas Ω is essentially a technique to attempt to construct $(\mathcal{L}^c, V)$-Hintikka sets that contain Ω. Here, the existence of such Hintikka sets, if Ω is satisfiable, is guaranteed by Corollary 4.12.26 or Exercise 169 of Chapter 2, depending on whether we deal with unsigned or signed formulas, respectively. If $\mathcal{L}$ does not contain $=$ and such an $(\mathcal{L}^c, V)$-Hintikka set is obtained, then, by Theorem 4.12.17 (for unsigned formulas) or by Theorem 4.12.50 (for signed formulas), one can determine a pair $(\mathcal{A}, \sigma)$, where $\mathcal{A}$ is a V-Herbrand structure for

$\mathcal{L}^c$ and $\sigma \in \text{ASSIGN}_{\mathcal{A}}$, such that $(\mathcal{A}, \sigma)$ satisfies Ω. When $= \in \mathcal{L}$, Theorems 4.12.20 and 4.12.53 show how to obtain a satisfying pair $(\mathcal{B}, \sigma)$, where $\mathcal{B}$ is now a quotient of a Herbrand structure.

This section begins with tableaux for signed first-order formulas. We will use the sequence of constituents $\mathtt{d}_{\mathcal{L},V}(b\varphi)$ of a signed $(\mathcal{L}, V)$-formula $b\varphi$ introduced in Section 4.12.

Definition 5.3.1. Let $\mathcal{L}$ be a first-order language and V be a set of variables. A *signed* $(\mathcal{L}, V)$*-tableau*, or just an $(\mathcal{L}, V)$*-tableau* for brevity, is a lot whose labels are sets of signed $(\mathcal{L}, V)$-formulas.

If $\mathtt{P}$ is a path of an $(\mathcal{L}, V)$-tableau $\mathtt{T}$, then $\mathtt{T}(\mathtt{P})$ is the set of signed formulas that occur in $\mathtt{P}$. $\square$

Definition 5.3.2. Let $\mathcal{L}$ be a first-order language and V be an $\mathcal{L}$-suitable set of variables. A node q of an $(\mathcal{L}, V)$-tableau $\mathtt{T}$ is *closed* if $\mathtt{T}(q)$ is closed, that is, there is a formula φ such that both $\mathbf{T}\varphi$ and $\mathbf{F}\varphi$ belong to $\mathtt{T}(q)$. A branch $\mathtt{B}$ of $\mathtt{T}$ is *closed* if the set of formulas occurring in $\mathtt{B}$ is closed. $\mathtt{B}$ is *strongly closed* if it is finite and q is closed, where q is the endpoint of $\mathtt{B}$.

A branch $\mathtt{B}$ is *complete* if the set of formulas occurring in $\mathtt{B}$ is an $(\mathcal{L}, V)$-Hintikka set. $\square$

An $(\mathcal{L}, V)$-tableau $\mathtt{T}'$ is an extension of an $(\mathcal{L}, V)$-tableau $\mathtt{T}$ if $\text{Dom}(\mathtt{T}) \subseteq \text{Dom}(\mathtt{T}')$ and for every $q \in \text{Dom}(\mathtt{T})$, we have $\mathtt{T}(q) = \mathtt{T}'(q)$.

Let Δ be a set of signed $(\mathcal{L}, V)$-formulas and let $\mathtt{T}$ be an $(\mathcal{L}, V)$-tableau. A notation analogous to the one used in propositional logic defines the $(\mathcal{L}, V)$-tableau $\mathtt{T}'$ as the tableau $(\mathtt{T}; \mathtt{T}(\lambda) \cup \Delta)$, that is, the $(\mathtt{T}(\lambda) \cup \Delta)$-join of $\mathtt{T}$, as introduced in Definition 1.7.17. As before, we will denote $\mathtt{T}'$ by $\mathtt{T} \uplus \Delta$.

Definition 5.3.3. An $(\mathcal{L}, V)$-tableau is *closed* (*strongly closed*) if every branch is closed (strongly closed). An $(\mathcal{L}, V)$-tableau is *completed* (*strongly completed*) if every branch is either closed (strongly closed) or complete. $\square$

Note that if $\mathtt{T}$ is a (strongly) closed $(\mathcal{L}, V)$-tableau and Δ is a set of signed $(\mathcal{L}, V)$-formulas, then $\mathtt{T} \uplus \Delta$ is also (strongly) closed.

Recall that we introduced the notion of finite-to-one function in Definition 1.2.18 on page 8. We will use this concept in the following definition.

Definition 5.3.4. Let $\mathcal{L}$ be a first-order language, V be an $\mathcal{L}$-suitable set of variables and Δ be a set of signed $(\mathcal{L}, V)$-formulas. A $(\Delta, \mathcal{L}, V)$-*tableau* is an $(\mathcal{L}, V)$-tableau $\mathtt{T}$ that satisfies the following conditions:

- The root of $\mathtt{T}$ is labeled by Δ, i.e., $\mathtt{T}(\lambda) = \Delta$.
- If q is an interior node of $\mathtt{T}$, then one of the following cases occurs:
  (1) there is some set of signed formulas Δ' and signed formula $b\varphi$ with φ not atomic and $\mathtt{T}(q) = \Delta' \cup \{b\varphi\}$ such that either
    (a) $b\varphi$ is neither a $\boldsymbol{\gamma}$- nor a $\boldsymbol{\delta}$-formula,
    $$\mathtt{d}_{\mathcal{L},V}(b\varphi) = (K_0, \ldots, K_{n-1}),$$
    q has n immediate descendants, and $\mathtt{T}(qi) = \Delta' \cup K_i$ for $0 \leq i \leq n-1$, or
    (b) $b\varphi$ is a $\boldsymbol{\gamma}$-formula $b(Qx)\psi$, q has one immediate descendant, and $\mathtt{T}(q0) = \Delta' \cup \{b\varphi, b(\psi')_{x:=t}\}$ for some term $t \in \mathrm{TERM}_{\mathcal{L}}(V)$ and variant ψ' of ψ such that t is substitutable for x in ψ', or
    (c) $b\varphi$ is a $\boldsymbol{\delta}$-formula $b(Qx)\psi$, q has one immediate descendant, and $\mathtt{T}(q0) = \Delta' \cup \{b(\psi)_{x:=c}\}$ for some constant symbol $c \in \mathcal{L}$ that does not occur in any signed formula in $\mathtt{T}(q)$, if x occurs free in ψ. Otherwise, that is, if $x \notin \mathtt{FV}(\psi)$, $\mathtt{T}(q0) = \Delta' \cup \{b\psi\}$.
  (2) q has one immediate descendant, $q0$, and there is a function $f : \mathtt{T}(q0) \longrightarrow \mathtt{T}(q)$ which is finite-to-one (that is, for every signed formula $b\psi \in \mathtt{T}(q)$, the set $f^{-1}(b\psi)$ is finite) such that each signed formula $b\varphi \in \mathtt{T}(q0)$ is a variant of the formula $f(b\varphi)$ from $\mathtt{T}(q)$.
  (3) $= \in \mathcal{L}$, q has one immediate descendant, and $\mathtt{T}(q0) = \mathtt{T}(q) \cup \{\mathbf{T}\alpha\}$ for some $\alpha \in \mathrm{INST}_{\mathcal{L},V}(\mathrm{Eq}_{=,\mathcal{L}})$.

If the first case holds at an interior node q, then we say that *regular expansion* was used at q. More particularly, if Case 1(a) holds, we say that *propositional expansion* was used at q; if Cases 1(b) or 1(c) hold, then we have $\boldsymbol{\gamma}$-*expansion* and $\boldsymbol{\delta}$-*expansion*, respectively. In Case 1(c), if x occurs free in the formula ψ, then the constant symbol c is called an *eigenconstant* of $\mathtt{T}$. Note that given a tableau, the eigenconstants used in $\boldsymbol{\delta}$-expansions are uniquely determined. When $x \notin \mathtt{FV}(\psi)$, we say that *degenerate* $\boldsymbol{\delta}$*-expansion* occurred at q.

If the second case holds, we say that *variantizing* was used at q.

Finally, if the third case holds, that is, we add a signed instance of an equality axiom to $\mathtt{T}(q)$ to obtain $\mathtt{T}(q0)$, then we say that *equality expansion* was used at q.

A special case of variantizing occurs when $\mathtt{T}(q0) \subseteq \mathtt{T}(q)$. We will say that $\mathtt{T}(q0)$ was obtained from $\mathtt{T}(q)$ by *thinning*.

We refer to a $(\Delta, \mathcal{L}, \emptyset)$-tableau as a $(\Delta, \mathcal{L})$*-tableau*.

If any of the previous three conditions is met by an interior node q of an arbitrary $(\mathcal{L}, V)$-tableau $\mathtt{T}$, then we say that $\mathtt{T}$ is *locally consistent* at q.

A $(\Delta, \mathcal{L}, V)$-tableau $\mathtt{T}$ is *locally conservative at an interior node* q if either regular expansion or equality expansion is used at q. $\mathtt{T}$ is called a *conservative tableau* if it is locally conservative at each of its interior nodes. ▯

Theorem 5.3.5. *Let $\mathcal{L}$ be a first-order language and let V be an $\mathcal{L}$-suitable set of variables. If Δ is a set of signed $(\mathcal{L}, V)$-formulas, then every formula that occurs in a node of a $(\Delta, \mathcal{L}, V)$-tableau is an $(\mathcal{L}, V)$-formula.*

Proof. This statement follows from Corollary 4.12.42, Theorem 4.6.22 and Corollary 4.3.85. □

The previous theorem shows that every $(\Delta, \mathcal{L}, V)$-tableau is an $(\mathcal{L}, V)$-tableau.

From now on, whenever we consider a $(\Delta, \mathcal{L}, V)$-tableau, we assume that V is $\mathcal{L}$-suitable.

Part 1b of Definition 5.3.4 leaves us the option of choosing ψ' to be $\mathsf{variant}(\psi, x, t)$, which would be sufficient for completeness. However, we use this more flexible rule in order to simplify transformations between several formal systems. Also, variantizing is not necessary for proving completeness. However, it provides technical help in the same circumstances.

In a $(\Delta, \mathcal{L}, V)$-tableau, it is possible that regular expansion, thinning, and equality expansion could be used at the same node. An example would be if $\alpha \in \mathrm{INST}_{\mathcal{L},V}(\mathrm{Eq}_{=,\mathcal{L}})$, $\{\mathbf{F}(\neg\alpha), \mathbf{T}\alpha\} \subseteq \mathtt{T}(q)$, q has one immediate descendant $q0$ in $\mathtt{T}$, and $\mathtt{T}(q0) = \mathtt{T}(q)$.

In the case of regular expansion, as in propositional logic, $b\varphi$ may or may not be a member of Δ' and, if $b\varphi \in \Delta'$, then $b\varphi$ and Δ' may not be uniquely determined. If $b\varphi \notin \Delta'$ and $b\varphi$ is not a $\boldsymbol{\gamma}$-formula, then $b\varphi$ is uniquely determined as the signed formula in $\mathtt{T}(q)$ that does not belong to $\mathtt{T}(qi)$ for any of the immediate descendants qi of q and Δ' is also uniquely determined as $\mathtt{T}(q) - \{b\varphi\}$. When we have regular expansion and either $b\varphi$ is a $\boldsymbol{\gamma}$-formula or $b\varphi \in \Delta'$, we say that the formula expanded at q is *retained*; otherwise, we say that the formula is *removed* at q. The reader will observe that in the case of regular expansion, we cannot have both expansion with retention and expansion with removal at the same node.

Definition 5.3.6. Let $\mathcal{L}$ be a first-order language, V be an $\mathcal{L}$-suitable set of variables, and Δ be a set of signed $(\mathcal{L}, V)$-formulas.

A $(\Delta, \mathcal{L}, V)$-*tableau with retention* is a $(\Delta, \mathcal{L}, V)$-tableau $\mathtt{T}$ such that at every interior node of $\mathtt{T}$ where neither variantizing nor equality expansion was used, the formula expanded is retained.

A $(\Delta, \mathcal{L}, V)$-*tableau with removal* is a $(\Delta, \mathcal{L}, V)$-tableau $\mathtt{T}$ such that at every interior node of $\mathtt{T}$ where none of equality expansion, variantizing, or $\boldsymbol{\gamma}$-expansion were used, the formula expanded is removed. ▯

Note that every $(\Delta, \mathcal{L}, V)$-tableau is finitely branching. Therefore, every strongly closed $(\Delta, \mathcal{L}, V)$-tableau $\mathtt{T}$ is finite by König's Lemma (see Theorem 1.7.7) because every branch of $\mathtt{T}$ is finite.

Example 5.3.7. Let $\mathcal{L}$ be a first-order language, V be an $\mathcal{L}$-suitable set of variables, φ be an $\mathcal{L}$-formula and t be a term in $\mathrm{TERM}_{\mathcal{L}}(V)$. Figure 5.1 contains a strongly closed $(\Delta, \mathcal{L}, V)$-tableau, where $\Delta = \{\mathbf{F}((\forall x)\varphi \to (\exists x)\varphi)\}$. In this figure, t is an arbitrary $(\mathcal{L}, V)$-term. ▯

Example 5.3.8. Figure 5.2 shows a strongly closed $(\Delta, \mathcal{L})$-tableau, where $\mathcal{L} = \{P, c, d\}$, P is a unary relation symbol, c, d are two constant symbols, and $\Delta = \{\mathbf{F}(\exists x)(P(x) \to (\forall x)P(x))\}$. Note that the $\boldsymbol{\gamma}$-formula $\mathbf{F}(\exists x)(P(x) \to (\forall x)P(x))$ is expanded twice, at λ using the term d, and at 000 using the term c. If we were allowed to expand the $\boldsymbol{\delta}$-formula $\mathbf{F}(\forall x)P(x)$ at 00 using d, the tableau would have been

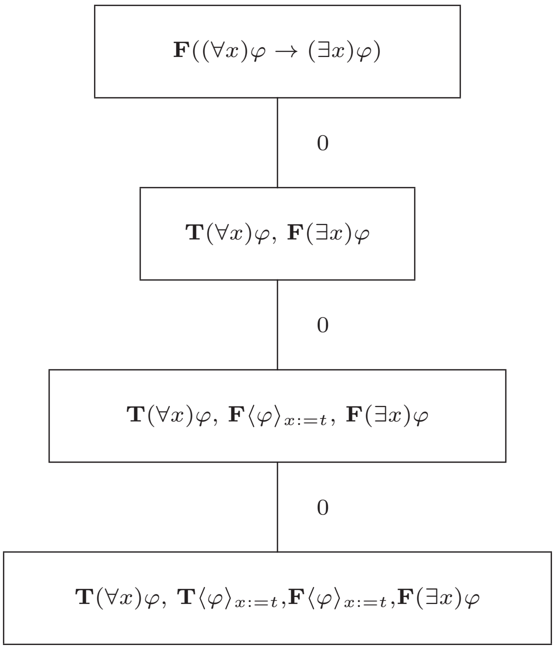

$\mathbf{F}((\forall x)\varphi \to (\exists x)\varphi)$
0
$\mathbf{T}(\forall x)\varphi,\ \mathbf{F}(\exists x)\varphi$
0
$\mathbf{T}(\forall x)\varphi,\ \mathbf{F}\langle\varphi\rangle_{x:=t},\ \mathbf{F}(\exists x)\varphi$
0
$\mathbf{T}(\forall x)\varphi,\ \mathbf{T}\langle\varphi\rangle_{x:=t},\mathbf{F}\langle\varphi\rangle_{x:=t},\mathbf{F}(\exists x)\varphi$

Fig. 5.1. A $(\{\mathbf{F}((\forall x)\varphi \to (\exists x)\varphi)\}, \mathcal{L}, V)$-tableau.

closed at 000. However, we had to use a new constant symbol c, which necessitated the second expansion of the $\boldsymbol{\gamma}$-formula.

□

Example 5.3.9. Let $\mathcal{L} = \{f, g, R\}$, where f, g are unary function symbols and R is a binary relation symbol. In Figure 5.3, we have a strongly closed $(\Delta, \mathcal{L}, \{x_0, x_1\})$-tableau, where

$$\Delta = \{\mathbf{F}((\forall x_0)(\forall x_1)R(x_0, x_1) \to R(f(x_0), g(x_1)))\}.$$

Observe that to get the strongly closed tableau, we need to use non-ground terms in the two $\boldsymbol{\gamma}$-expansions.

Similarly, in Figure 5.4, we construct a strongly closed $(\Delta', \mathcal{L}, \{x_0, x_1\})$-tableau, where

$$\Delta' = \{\mathbf{F}((\forall x_0)(\forall x_1)R(x_0, x_1) \to R(f(x_1), g(x_0)))\}.$$

Observe the use of the variant in expanding the node 00. □

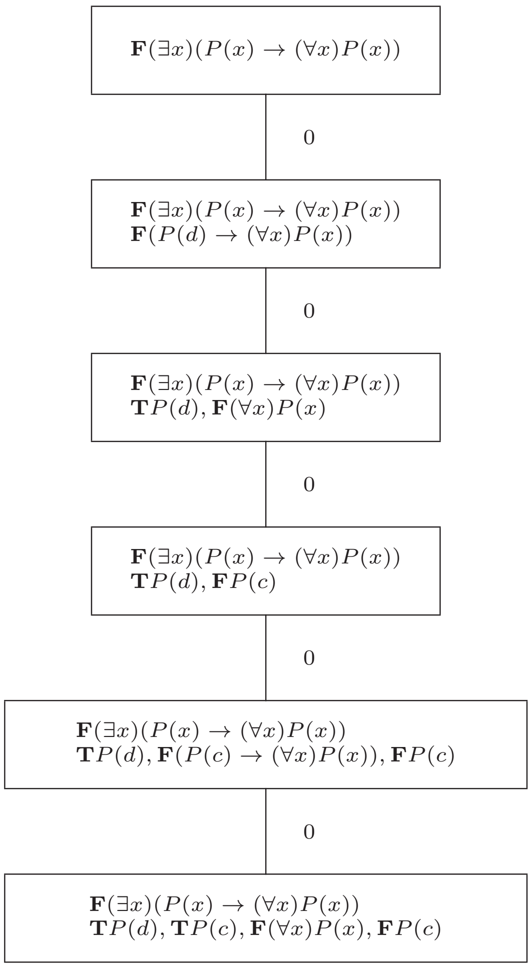

$\mathbf{F}(\exists x)(P(x) \to (\forall x)P(x))$
0
$\mathbf{F}(\exists x)(P(x) \to (\forall x)P(x))$
$\mathbf{F}(P(d) \to (\forall x)P(x))$
0
$\mathbf{F}(\exists x)(P(x) \to (\forall x)P(x))$
$\mathbf{T}P(d), \mathbf{F}(\forall x)P(x)$
0
$\mathbf{F}(\exists x)(P(x) \to (\forall x)P(x))$
$\mathbf{T}P(d), \mathbf{F}P(c)$
0
$\mathbf{F}(\exists x)(P(x) \to (\forall x)P(x))$
$\mathbf{T}P(d), \mathbf{F}(P(c) \to (\forall x)P(x)), \mathbf{F}P(c)$
0
$\mathbf{F}(\exists x)(P(x) \to (\forall x)P(x))$
$\mathbf{T}P(d), \mathbf{T}P(c), \mathbf{F}(\forall x)P(x), \mathbf{F}P(c)$

Fig. 5.2. A $(\{\mathbf{F}(\exists x)(P(x) \to (\forall x)P(x))\}, \mathcal{L})$-tableau.

Example 5.3.10. Let $\mathcal{L} = \{f, =\}$, where f is a binary function symbol and let φ be the formula

$$\varphi = (\forall x)(\forall y)(((\forall z)(f(x,z) = z) \land (\forall w)(f(w,y) = w)) \to (x = y))$$

considered in Example 5.2.11.

Figures 5.5 and 5.6 contain a strongly closed $(\{\mathbf{F}\varphi\}, \mathcal{L} \cup \{c, d\})$-tableau, where c, d are constant symbols. $\square$

$$\begin{tikzcd}
\boxed{\mathbf{F}((\forall x_0)(\forall x_1)R(x_0,x_1) \rightarrow R(f(x_0),g(x_1)))} \arrow[d, dash, "0"] \\
\boxed{\mathbf{T}(\forall x_0)(\forall x_1)R(x_0,x_1), \mathbf{F}R(f(x_0),g(x_1)),} \arrow[d, dash, "0"] \\
\boxed{\begin{array}{l} \mathbf{T}(\forall x_0)(\forall x_1)R(x_0,x_1), \mathbf{T}(\forall x_1)R(f(x_0),x_1), \\ \mathbf{F}R(f(x_0),g(x_1)), \end{array}} \arrow[d, dash, "0"] \\
\boxed{\begin{array}{l} \mathbf{T}(\forall x_0)(\forall x_1)R(x_0,x_1), \mathbf{T}(\forall x_1)R(f(x_0),x_1), \\ \mathbf{T}R(f(x_0),g(x_1)), \mathbf{F}R(f(x_0),g(x_1)) \end{array}}
\end{tikzcd}$$

Fig. 5.3. A strongly closed $(\Delta, \mathcal{L}, \{x_0, x_1\})$-tableau.

$$\begin{tikzcd}
\boxed{\mathbf{F}((\forall x_0)(\forall x_1)R(x_0,x_1) \rightarrow R(f(x_1),g(x_0)))} \arrow[d, dash, "0"] \\
\boxed{\mathbf{T}(\forall x_0)(\forall x_1)R(x_0,x_1), \mathbf{F}R(f(x_1),g(x_0))} \arrow[d, dash, "0"] \\
\boxed{\begin{array}{l} \mathbf{T}(\forall x_0)(\forall x_1)R(x_0,x_1), \mathbf{T}(\forall x_2)R(f(x_1),x_2), \\ \mathbf{F}R(f(x_1),g(x_0)) \end{array}} \arrow[d, dash, "0"] \\
\boxed{\begin{array}{l} \mathbf{T}(\forall x_0)(\forall x_1)R(x_0,x_1), \mathbf{T}(\forall x_2)R(f(x_1),x_2) \\ \mathbf{T}R(f(x_1),g(x_0)), \mathbf{F}R(f(x_1),g(x_0)) \end{array}}
\end{tikzcd}$$

Fig. 5.4. A strongly closed $(\Delta', \mathcal{L}, \{x_0, x_1\})$-tableau.

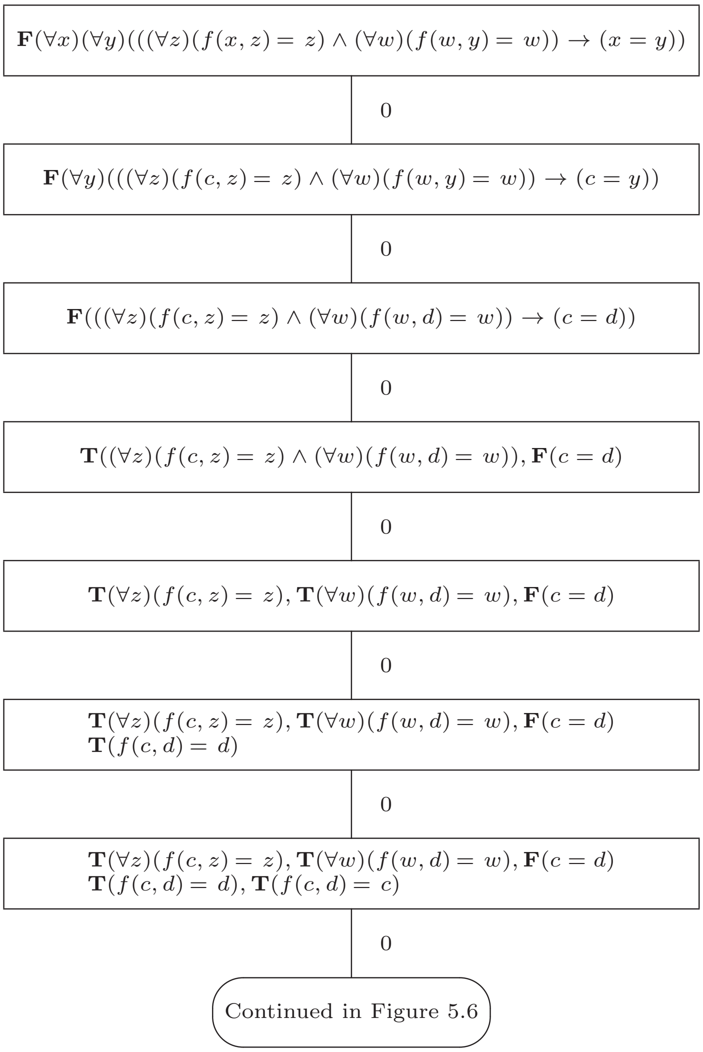

$\mathbf{F}(\forall x)(\forall y)(((\forall z)(f(x,z)= z) \wedge (\forall w)(f(w,y)= w)) \rightarrow (x=y))$
0
$\mathbf{F}(\forall y)(((\forall z)(f(c,z)= z) \wedge (\forall w)(f(w,y)= w)) \rightarrow (c=y))$
0
$\mathbf{F}(((\forall z)(f(c,z)= z) \wedge (\forall w)(f(w,d)= w)) \rightarrow (c=d))$
0
$\mathbf{T}((\forall z)(f(c,z)= z) \wedge (\forall w)(f(w,d)= w)), \mathbf{F}(c=d)$
0
$\mathbf{T}(\forall z)(f(c,z)= z), \mathbf{T}(\forall w)(f(w,d)= w), \mathbf{F}(c=d)$
0
$\mathbf{T}(\forall z)(f(c,z)= z), \mathbf{T}(\forall w)(f(w,d)= w), \mathbf{F}(c=d)$
$\mathbf{T}(f(c,d)= d)$
0
$\mathbf{T}(\forall z)(f(c,z)= z), \mathbf{T}(\forall w)(f(w,d)= w), \mathbf{F}(c=d)$
$\mathbf{T}(f(c,d)= d), \mathbf{T}(f(c,d)= c)$
0
Continued in Figure 5.6

Fig. 5.5. A strongly closed $(\{\mathbf{F}\varphi\}, \mathcal{L} \cup \{c,d\})$-tableau.

Theorem 5.3.11. *Let Δ be a set of signed $\mathcal{L}$-formulas. Suppose that* $\mathtt{T}$ *is a $(\Delta, \mathcal{L}, V)$-tableau whose root has n immediate descendants.*

For every i, $0 \le i \le n-1$, the following hold:

(1) $\mathtt{T}_{[i]}$ *is a $(\mathtt{T}(i), \mathcal{L}, V)$-tableau;*
(2) *if* $\mathtt{T}$ *is strongly closed, then so is* $\mathtt{T}_{[i]}$.

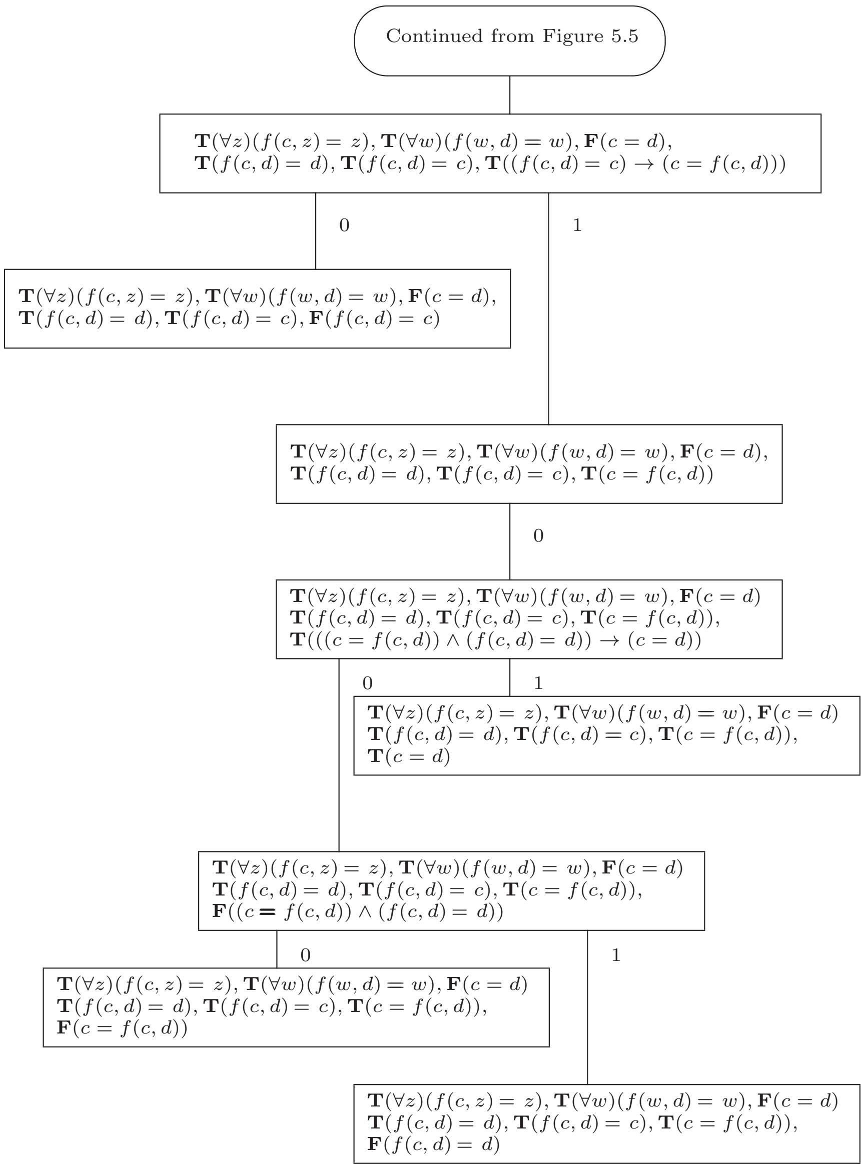

Continued from Figure 5.5
$\mathbf{T}(\forall z)(f(c,z)=z), \mathbf{T}(\forall w)(f(w,d)=w), \mathbf{F}(c=d), \mathbf{T}(f(c,d)=d), \mathbf{T}(f(c,d)=c), \mathbf{T}((f(c,d)=c) \to (c=f(c,d)))$
0
1
$\mathbf{T}(\forall z)(f(c,z)=z), \mathbf{T}(\forall w)(f(w,d)=w), \mathbf{F}(c=d), \mathbf{T}(f(c,d)=d), \mathbf{T}(f(c,d)=c), \mathbf{F}(f(c,d)=c)$
$\mathbf{T}(\forall z)(f(c,z)=z), \mathbf{T}(\forall w)(f(w,d)=w), \mathbf{F}(c=d), \mathbf{T}(f(c,d)=d), \mathbf{T}(f(c,d)=c), \mathbf{T}(c=f(c,d))$
0
$\mathbf{T}(\forall z)(f(c,z)=z), \mathbf{T}(\forall w)(f(w,d)=w), \mathbf{F}(c=d) \mathbf{T}(f(c,d)=d), \mathbf{T}(f(c,d)=c), \mathbf{T}(c=f(c,d)), \mathbf{T}(((c=f(c,d)) \wedge (f(c,d)=d)) \to (c=d))$
0
1
$\mathbf{T}(\forall z)(f(c,z)=z), \mathbf{T}(\forall w)(f(w,d)=w), \mathbf{F}(c=d) \mathbf{T}(f(c,d)=d), \mathbf{T}(f(c,d)=c), \mathbf{T}(c=f(c,d)), \mathbf{T}(c=d)$
$\mathbf{T}(\forall z)(f(c,z)=z), \mathbf{T}(\forall w)(f(w,d)=w), \mathbf{F}(c=d) \mathbf{T}(f(c,d)=d), \mathbf{T}(f(c,d)=c), \mathbf{T}(c=f(c,d)), \mathbf{F}((c=f(c,d)) \wedge (f(c,d)=d))$
0
1
$\mathbf{T}(\forall z)(f(c,z)=z), \mathbf{T}(\forall w)(f(w,d)=w), \mathbf{F}(c=d) \mathbf{T}(f(c,d)=d), \mathbf{T}(f(c,d)=c), \mathbf{T}(c=f(c,d)), \mathbf{F}(c=f(c,d))$
$\mathbf{T}(\forall z)(f(c,z)=z), \mathbf{T}(\forall w)(f(w,d)=w), \mathbf{F}(c=d) \mathbf{T}(f(c,d)=d), \mathbf{T}(f(c,d)=c), \mathbf{T}(c=f(c,d)), \mathbf{F}(f(c,d)=d)$

Fig. 5.6. A strongly closed $(\{\mathbf{F}\varphi\}, \mathcal{L} \cup \{c,d\})$-tableau.

Proof. This straightforward argument is left to the reader. $\square$

Theorem 5.3.12. *If* $\mathtt{T}$ *is a* $(\Delta, \mathcal{L}, V)$*-tableau of depth 1 such that the root has* n *immediate descendants and* $\mathtt{T}_i$ *is a strongly closed* $(\mathtt{T}(i), \mathcal{L}, V)$*-tableau for* $0 \leq i \leq n-1$*, then* $(\mathtt{T}_0, \ldots, \mathtt{T}_{n-1}; \Delta)$ *is a strongly closed* $(\Delta, \mathcal{L}, V)$*-tableau.*

Proof. This statement follows directly from the definition of tableau and it is left to the reader. $\square$

The following theorem gives the analyticity of first-order tableaux.

Theorem 5.3.13. *Let* $\mathcal{L}$ *be a first-order language,* V *be an* $\mathcal{L}$*-suitable set of variables and* Δ *be a set of signed* $(\mathcal{L}, V)$*-formulas. For every node* q *of a* $(\Delta, \mathcal{L}, V)$*-tableau* $\mathtt{T}$*, we have*

$$\mathtt{T}(q) \subseteq \begin{cases} \mathbf{Bool} \times W^*_{\mathcal{L},V}(\Gamma) & \textit{if} = \notin \mathcal{L} \\ \mathbf{Bool} \times W^*_{\mathcal{L},V}(\Gamma \cup \mathrm{INST}_{\mathcal{L},V}(Eq_{=,\mathcal{L}})) & \textit{if} = \in \mathcal{L} \end{cases}$$

where $\Gamma = \{\varphi \mid b\varphi \in \Delta$ *for some* $b \in \mathbf{Bool}\}$.

Proof. We discuss in this proof only the case when $= \notin \mathcal{L}$. The case of languages with equality is similar and is left to the reader.

The argument is by induction on the level of q. If q is the root then it is clear that

$$\mathtt{T}(q) = \Delta \subseteq \mathbf{Bool} \times \Gamma \subseteq \mathbf{Bool} \times W^*_{\mathcal{L},V}(\Gamma).$$

Now suppose that the result holds for all nodes on level i and that r is on level $i+1$. Then, $r = qj$ for some node q on level i and for every member $b_0\alpha$ of $\mathtt{T}(r) - \mathtt{T}(q)$ we have one of the following cases:

`Case 1:` $b_0\alpha$ is an element of an $(\mathcal{L}, V)$-constituent K of a formula $b\theta$ from $\mathtt{T}(q)$. By inductive hypothesis, $b\theta \in \mathbf{Bool} \times W^*_{\mathcal{L},V}(\Gamma)$. The analyticity of constituents and the definition of the analytical universe imply $b_0\alpha \in \mathbf{Bool} \times W_{\mathcal{L},V}(W^*_{\mathcal{L},V}(\Gamma)) \subseteq \mathbf{Bool} \times W^*_{\mathcal{L},V}(\Gamma)$.

`Case 2:` $b_0\alpha$ has the form $b(\varphi')_{x:=t}$ where $b\theta = b(Qx)\varphi$ is a $\boldsymbol{\gamma}$ formula in $\mathtt{T}(q)$, $t \in \mathrm{TERM}_{\mathcal{L}}(V)$ and φ' is a variant of φ such that t is substitutable for x in φ'. By the inductive hypothesis, $b\theta \in \mathbf{Bool} \times W^*_{\mathcal{L},V}(\Gamma)$. By the definition of the analytical universe, $b_0\alpha \in \mathbf{Bool} \times W_{\mathcal{L},V}(W^*_{\mathcal{L},V}(\Gamma)) \subseteq \mathbf{Bool} \times W^*_{\mathcal{L},V}(\Gamma)$.

`Case 3:` $b_0\alpha$ is a variant of a formula $b_0\theta$ in $\mathtt{T}(q)$. Using the inductive hypothesis and the definition of the analytic universe, we obtain $b\theta \in \mathbf{Bool} \times W^*_{\mathcal{L},V}(\Gamma)$. The analyticity of constituents and the definition of the analytical universe imply $b_0\alpha \in \mathbf{Bool} \times W_{\mathcal{L},V}(W^*_{\mathcal{L},V}(\Gamma)) \subseteq \mathbf{Bool} \times W^*_{\mathcal{L},V}(\Gamma)$. □

Theorem 5.3.14. *Let $\mathcal{L}$ be a first-order language, V be an $\mathcal{L}$-suitable set of variables, Δ be a set of signed $(\mathcal{L}, V)$-formulas and $\mathtt{T}$ be a $(\Delta, \mathcal{L}, V)$-tableau. If q is an interior node of $\mathtt{T}$ whose immediate descendants are $q\,0, \ldots q\,n-1$, then:*

(1) *If $\mathtt{T}$ is conservative at q and $(\mathcal{A}, \sigma) \models \mathtt{T}(qi)$, then $(\mathcal{A}, \sigma) \models \mathtt{T}(q)$, for every $\mathcal{L}$-structure $\mathcal{A}$, $\sigma \in \mathrm{ASSIGN}_{\mathcal{A}}$, and $0 \le i \le n-1$.*
(2) *If $\mathtt{T}(q)$ is satisfiable, then $\mathtt{T}(qi)$ is satisfiable for some i with $0 \le i \le n-1$.*

Proof. To prove the first part of the theorem, we have to consider two cases.

- If either $\boldsymbol{\gamma}$-expansion or equality expansion is used at q in $\mathtt{T}$, then we have $\mathtt{T}(q) \subseteq \mathtt{T}(qi)$, so the result is immediate.
- Otherwise, the conclusion follows immediately from the first and third parts of Theorem 4.12.48.

For the second part of the theorem, the argument depends on the nature of the expansion at q.

- If equality expansion was used at q, then $\mathtt{T}(q0)$ differs from $\mathtt{T}(q)$ at most by the addition of a logically valid formula (an instance of an equality axiom), so the satisfiability of $\mathtt{T}(q)$ implies the satisfiability of $\mathtt{T}(q0)$.
- If a $\boldsymbol{\delta}$-formula is expanded at q, then the conclusion follows from Theorem 4.6.53.
- Suppose that a $\boldsymbol{\gamma}$-formula of say the form $\mathbf{T}(\forall x)\varphi$ is expanded at q and the formula $(\varphi')_{x:=t}$ is added, where φ' is a variant of φ and t is substitutable for x in φ'. If $(\mathcal{A}, \sigma)$ satisfy $\mathtt{T}(q)$, we have $(\mathcal{A}, \sigma) \models (\forall x)\varphi$, so $(\mathcal{A}, [x \to \sigma^{\mathcal{A}}(t)]\sigma) \models \varphi$, which is equivalent to $(\mathcal{A}, [x \to \sigma^{\mathcal{A}}(t)]\sigma) \models \varphi'$ because φ' is a variant of φ. Since t is substitutable for x in φ', this is equivalent to $(\mathcal{A}, \sigma) \models (\varphi')_{x:=t}$, so $(\mathcal{A}, \sigma) \models \mathtt{T}(qi)$.

- If variantizing is used at q, then every formula in $\mathtt{T}(qi)$ is a variant of a formula in $\mathtt{T}(q)$ and the statement follows immediately.
- In the remaining propositional case, the statement follows from the first part of Theorem 4.12.48.

$\square$

Corollary 5.3.15. *Let* $\mathtt{T}$ *be a* $(\Delta, \mathcal{L}, V)$*-tableau, where* $\mathcal{L}$ *is a first-order language,* V *is an* $\mathcal{L}$*-suitable set of variables, and* Δ *is a set of signed* $(\mathcal{L}, V)$*-formulas.*

(1) *If* $\mathtt{T}$ *is a conservative tableau and* $\mathtt{T}(q)$ *is satisfiable, then* $\mathtt{T}(\mathtt{P})$ *is satisfiable, where* $\mathtt{P}$ *is the path of* $\mathtt{T}$ *leading to* q*. In fact, if* $(\mathcal{A}, \sigma) \models \mathtt{T}(q)$*, then* $(\mathcal{A}, \sigma) \models \mathtt{T}(\mathtt{P})$*, for every* $\mathcal{L}$*-structure* $\mathcal{A}$ *and* $\sigma \in \mathrm{ASSIGN}_{\mathcal{A}}$*.*

(2) *If* Δ *is satisfiable, then there is a branch* $\mathtt{B}$ *of* $\mathtt{T}$ *such that for every node* q *of* $\mathtt{B}$*,* $\mathtt{T}(q)$ *is satisfiable.*

Proof. The parts of the corollary follow immediately from the corresponding parts of Theorem 5.3.14. $\square$

Corollary 5.3.16. *Let* $\mathtt{T}$ *be a conservative* $(\Delta, \mathcal{L}, V)$*-tableau, where* $\mathcal{L}$ *is a first-order language,* V *is an* $\mathcal{L}$*-suitable set of variables, and* Δ *is a set of signed* $(\mathcal{L}, V)$*-formulas. If* Δ *is satisfiable, then there is a branch* $\mathtt{B}$ *of* $\mathtt{T}$ *such that for every node* q *of* $\mathtt{B}$*,* $\mathtt{T}(\mathtt{P}_q)$ *is satisfiable, where* $\mathtt{P}_q$ *is the path leading to* q*.*

Proof. This statement follows from the two parts of Corollary 5.3.15, combined. $\square$

Example 5.3.17. Let $\mathcal{L} = \{P, Q, c, d\}$ be a first-order language with P, Q unary relation symbols and c, d constant symbols. Consider the set of signed closed formulas $\Delta = \{\mathbf{T}(\forall x)(P(x) \vee Q(d)), \mathbf{T}(\exists x)(\neg P(x))\}$. In Figure 5.7, we show a conservative $(\Delta, \mathcal{L})$-tableau $\mathtt{T}$ that is not completed.

Define the structure $\mathcal{A}$ by $|\mathcal{A}| = \mathbf{N}$, $P^{\mathcal{A}} = \{n \in \mathbf{N} \mid n \text{ is a perfect square}\}$, $Q^{\mathcal{A}} = \{n \in \mathbf{N} \mid n \text{ is even}\}$, $c^{\mathcal{A}} = 5$, and $d^{\mathcal{A}} = 4$. Then, it is easy to verify that $\mathcal{A} \models \mathtt{T}(010)$ and therefore, by Corollary 5.3.15, $\mathcal{A}$ is a model of all the sets $\mathtt{T}(q)$, where q is a node on the path that leads to 010. Note that this is true even though we are "recycling" the constant symbol c; namely, c is used first in the

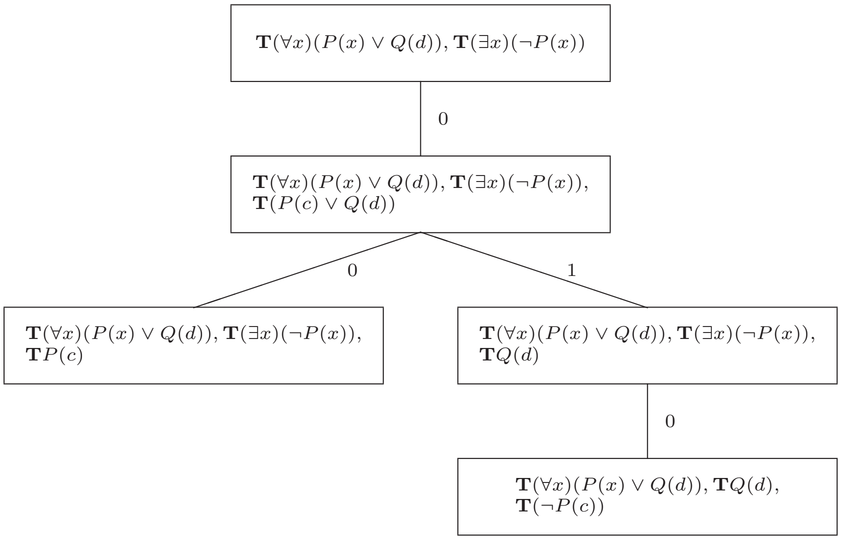

$\mathbf{T}(\forall x)(P(x) \vee Q(d)), \mathbf{T}(\exists x)(\neg P(x))$
0
$\mathbf{T}(\forall x)(P(x) \vee Q(d)), \mathbf{T}(\exists x)(\neg P(x)),$ $\mathbf{T}(P(c) \vee Q(d))$
0
1
$\mathbf{T}(\forall x)(P(x) \vee Q(d)), \mathbf{T}(\exists x)(\neg P(x)),$ $\mathbf{T}P(c)$
$\mathbf{T}(\forall x)(P(x) \vee Q(d)), \mathbf{T}(\exists x)(\neg P(x)),$ $\mathbf{T}Q(d)$
0
$\mathbf{T}(\forall x)(P(x) \vee Q(d)), \mathbf{T}Q(d),$ $\mathbf{T}(\neg P(c))$

Fig. 5.7. A $(\Delta, \mathcal{L})$-tableau that "recycles" a constant symbol.

expansion of $\mathtt{T}(\lambda)$ to produce the formula $\mathbf{T}(P(c) \vee Q(d))$ and then, again, in the expansion of $\mathtt{T}(01)$ to produce the formula $\mathbf{T}(\neg P(c))$. $\square$

Theorem 5.3.18. *Let $\mathcal{L}$ be a first-order language, V be an $\mathcal{L}$-suitable set of variables, Δ be a set of signed $(\mathcal{L}, V)$-formulas and* $\mathtt{T}$ *be a completed $(\Delta, \mathcal{L}, V)$-tableau. The following statements hold:*

(1) *If* $\mathtt{T}$ *is conservative, then Δ is satisfiable if and only if* $\mathtt{T}$ *is not closed.*
(2) *If* $\mathtt{T}$ *is strongly completed, then Δ is satisfiable if and only if* $\mathtt{T}$ *is not closed.*

Proof. Suppose first that $\mathtt{T}$ is conservative. If Δ is satisfiable, then, Corollary 5.3.16 implies that $\mathtt{T}$ has a branch $\mathtt{B}$ such that for every node q of $\mathtt{B}$, $\mathtt{T}(\mathtt{P}_q)$ is satisfiable, where $\mathtt{P}_q$ is the path leading to q. If $\mathtt{T}$ were closed, then $\mathtt{B}$ would be closed, and we could find a node q of $\mathtt{B}$ such that $\mathtt{T}(\mathtt{P}_q)$ is closed by going far enough on the branch. Thus, $\mathtt{T}(\mathtt{P}_q)$ would be unsatisfiable. Therefore, $\mathtt{T}$ is not closed.

Conversely, if $\mathtt{T}$ is not closed, then, since it is completed, there must be a complete branch $\mathtt{B}$ of $\mathtt{T}$. Since $\mathtt{T}(\mathtt{B})$ is an $(\mathcal{L}, V)$-Hintikka set, by Theorems 4.12.50 and 4.12.53, T(B) is satisfiable, so Δ is satisfiable.

Now, suppose that T is strongly completed, that is, each branch is either strongly closed or complete. If Δ is satisfiable, by the second part of Corollary 5.3.15, there is a branch B of T such that for every node q of B, T(q) is satisfiable. It follows that B is not strongly closed and therefore it is complete, which implies that T is not closed.

The converse implication can be justified in a manner similar to the first part of the argument. □

Theorem 5.3.19 (Soundness Theorem for Tableaux of First-Order Logic). *Let $\mathcal{L}$ be a first-order language, V be an $\mathcal{L}$-suitable set of variables, and let Δ be a set of signed $(\mathcal{L}, V)$-formulas. The following statements hold:*

(1) *If there is a conservative, closed $(\Delta, \mathcal{L}, V)$-tableau, then Δ is unsatisfiable.*
(2) *If there is a strongly closed $(\Delta, \mathcal{L}, V)$-tableau, then Δ is unsatisfiable.*

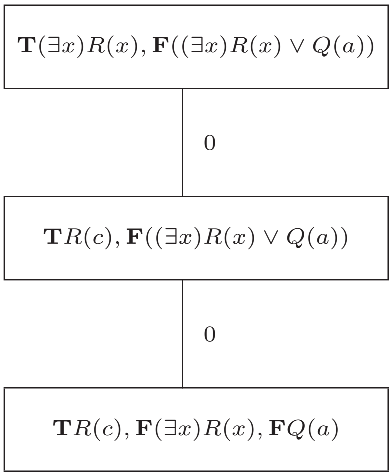
Proof. This is an immediate consequence of Theorem 5.3.18. □

If T is a conservative, completed $(\Delta, \mathcal{L}, V)$-tableau or a strongly completed $(\Delta, \mathcal{L}, V)$-tableau, then either T is closed, which means that Δ is unsatisfiable, or T has a complete branch B which means that T(B) is a Hintikka set. In the latter case, we have an explicit way of constructing a pair $(\mathcal{A}, \sigma)$ that satisfies Δ, as shown in Theorems 4.12.50 and 4.12.53.

Example 5.3.20. The existence of the strongly closed $(\{\mathbf{F}\varphi\}, \mathcal{L} \cup \{c, d\})$-tableau in Example 5.3.10 shows, by the Soundness Theorem, that the formula $\mathbf{F}\varphi$ is not satisfiable and hence that φ is logically valid. From an algebraic point of view, this means that in a groupoid, every left identity is equal to every right identity.

Similarly, the existence of the strongly closed tableaux in Examples 5.3.7 and 5.3.8 shows the logical validity of the formulas $((\forall x)\varphi \rightarrow (\exists x)\varphi)$ and $(\exists x)(P(x) \rightarrow (\forall x)P(x))$, respectively. □

Example 5.3.21. Let $\mathcal{L}$ be a first-order language that contains the constant symbols a and c and the unary relation symbols R and Q. Let Δ be the set of signed $\mathcal{L}$-formulas $\{\mathbf{T}(\exists x)R(x), \mathbf{F}((\exists x)R(x) \vee Q(a))\}$.



$\mathbf{T}(\exists x)R(x), \mathbf{F}((\exists x)R(x) \vee Q(a))$
0
$\mathbf{T}R(c), \mathbf{F}((\exists x)R(x) \vee Q(a))$
0
$\mathbf{T}R(c), \mathbf{F}(\exists x)R(x), \mathbf{F}Q(a)$

Fig. 5.8. Conservative and closed $(\Delta, \mathcal{L})$-tableau.

The tableau $\mathtt{T}$ in Figure 5.8 is a conservative and closed (because $\mathbf{T}(\exists x)R(x) \in \mathtt{T}(\lambda)$ and $\mathbf{F}(\exists x)R(x) \in \mathtt{T}(00)$) but not strongly closed $(\Delta, \mathcal{L})$-tableau. By the Soundness Theorem, Δ is not satisfiable. □

Example 5.3.22. Suppose $\mathcal{L}$ is a first-order language that contains the unary relation symbol R and the constant symbol c. Let Δ be the set of signed $\mathcal{L}$-formulas $\{\mathbf{T}(\exists x)R(x), \mathbf{T}(\exists x)(\neg R(x))\}$. This set is clearly satisfiable.

However, the $(\Delta, \mathcal{L})$-tableau shown in Figure 5.9 is closed. As implied by the Soundness Theorem, this tableau is not strongly closed. □

If we have a set of signed $(\mathcal{L}, V)$-formulas Δ, we may not be able to construct a completed $(\Delta, \mathcal{L}, V)$-tableau because expanding $\boldsymbol{\delta}$-formulas requires the use of "new" constant symbols, which may not exist in the language $\mathcal{L}$. This difficulty may be overcome by constructing a completed $(\Delta, \mathcal{L}^c, V)$-tableau. This construction will help us establish a completeness theorem for first-order tableaux.

A useful technical observation is contained in the following lemma.

Lemma 5.3.23. *Let* $\mathtt{P}$ *be a path of a conservative* $(\Delta, \mathcal{L}, V)$*-tableau ending in the node* q*.*

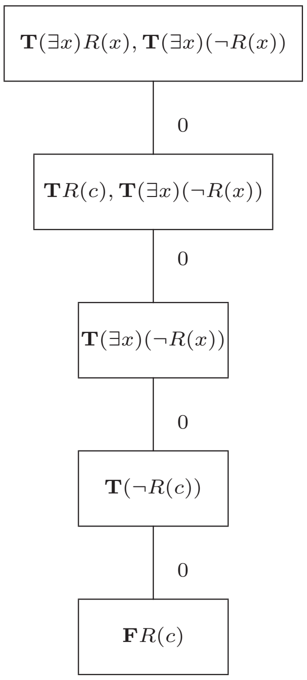

$\mathbf{T}(\exists x)R(x), \mathbf{T}(\exists x)(\neg R(x))$
0
$\mathbf{T}R(c), \mathbf{T}(\exists x)(\neg R(x))$
0
$\mathbf{T}(\exists x)(\neg R(x))$
0
$\mathbf{T}(\neg R(c))$
0
$\mathbf{F}R(c)$

Fig. 5.9. $(\Delta, \mathcal{L})$-tableau that is closed but not strongly closed.

(1) *If $b\varphi$ is a γ-formula such that $b\varphi \in \mathtt{T}(\mathtt{P})$, then $b\varphi \in \mathtt{T}(q)$.*
(2) *If $b\varphi \in \mathtt{T}(\mathtt{P}) - \mathtt{T}(q)$, then there is an $(\mathcal{L}, V)$-constituent K of $b\varphi$ such that $K \subseteq \mathtt{T}(\mathtt{P})$.*

Proof. We leave this argument to the reader. □

Let $b_0\varphi_0, b_1\varphi_1, \ldots$ be the enumeration of $\mathrm{SFORM}_{\mathcal{L}}(V)$ in the standard order and let $t_0, t_1, \ldots$ be the enumeration of $\mathrm{TERM}_{\mathcal{L}}(V)$ in the standard order of terms. Fix a pairing function $f : \mathbf{N}^2 \longrightarrow \mathbf{N}$, that is, f is a fixed bijection from $\mathbf{N}^2$ to $\mathbf{N}$. We can list $\mathrm{SFORM}_{\mathcal{L}}(V) \times \mathrm{TERM}_{\mathcal{L}}(V)$ by placing the pair $(b_i\varphi_i, t_j)$ in the $(k+1)$-st place, for $k = f(i, j)$. We will refer to this enumeration as the *standard ordering* of $\mathrm{SFORM}_{\mathcal{L}}(V) \times \mathrm{TERM}_{\mathcal{L}}(V)$.

The following definition will be useful in Construction 5.3.25.

Definition 5.3.24. Let $\mathtt{T}$ be an $(\Delta, \mathcal{L}, V)$-tableau and let $\mathtt{P}$ be a path of $\mathtt{T}$. The *set of pairs that require attention in* $\mathtt{T}$ *on* $\mathtt{P}$ is the set of all pairs $(b\varphi, t)$ in $\mathrm{SFORM}_{\mathcal{L}}(V) \times \mathrm{TERM}_{\mathcal{L}}(V)$ that satisfy one of the following conditions:

(1) $b\varphi \in \mathtt{T}(\mathtt{P})$, φ is not atomic, $b\varphi$ is not a $\boldsymbol{\gamma}$-formula and none of the constituents of $b\varphi$ is contained in $\mathtt{T}(\mathtt{P})$.
(2) $b\varphi \in \mathtt{T}(\mathtt{P})$, $b\varphi = b(Qx)\psi$ is a $\boldsymbol{\gamma}$-formula and $b\langle\psi\rangle_{x:=t} \notin \mathtt{T}(\mathtt{P})$.
(3) $= \in \mathcal{L}$, $b = \mathbf{T}$, $\varphi \in \mathrm{INST}_{\mathcal{L},V}(\mathrm{Eq}_{=,\mathcal{L}})$ and $b\varphi \notin \mathtt{T}(\mathtt{P})$.

If $\mathtt{P}$ is the path leading to the node q, we will refer to the set of pairs that require attention in $\mathtt{T}$ on $\mathtt{P}$ as the *set of pairs that require attention in* $\mathtt{T}$ *at* q. □

Construction 5.3.25.
Input: A first-order language $\mathcal{L}$, a set of variables V and a set Δ of signed $(\mathcal{L}, V)$-formulas.
Output: A sequence $\mathtt{T}_0, \mathtt{T}_1, \ldots$ of finite conservative $(\Delta, \mathcal{L}^c, V)$-tableaux such that each $\mathtt{T}_{i+1}$ is a leaf extension of $\mathtt{T}_i$ and $\mathtt{T} = \bigcup\{\mathtt{T}_i \mid i \geq 0\}$ is a conservative, completed $(\Delta, \mathcal{L}^c, V)$-tableau.
Method:

(A) Let $\mathtt{T}_0$ be the one-node $(\Delta, \mathcal{L}^c, V)$-tableau with root labeled by Δ.

(B) Suppose that $\mathtt{T}_i$ has been defined. Then, if $\mathtt{T}_i$ is completed, the construction stops with $\mathtt{T}_i$. Otherwise, $\mathtt{T}_i$ has branches that are neither closed nor complete. Select nondeterministically among the shortest such branches a branch $\mathtt{B}$ ending in the leaf q.
Choose $(b_q\varphi_q, t_q) \in \mathtt{T}_i(\mathtt{B}) \times \mathrm{TERM}_{\mathcal{L}^c}(V)$ to be the first pair $(b\varphi, t)$ in the standard order of $\mathrm{SFORM}_{\mathcal{L}^c}(V) \times \mathrm{TERM}_{\mathcal{L}^c}(V)$ that requires attention in $\mathtt{T}_i$ at q. By Lemma 5.3.23, if Case 1 or 2 of Definition 5.3.24 applies, then $b_q\varphi_q \in \mathtt{T}_i(q)$.
The conservative tableau $\mathtt{T}_{i+1}$ is defined as follows:

- If Case 1 of Definition 5.3.24 holds, $b_q\varphi_q$ is not a $\boldsymbol{\delta}$-formula, and $(K_0, \ldots, K_{n-1})$ is the $(\mathcal{L}^c, V)$-constituent sequence of $b_q\varphi_q$, then define $\mathtt{T}_{i+1}$ by adding to $\mathrm{Dom}(\mathtt{T}_i)$ the nodes $q0, \ldots, q_{n-1}$ and letting $\mathtt{T}_{i+1}(qj) = \Delta_q \cup K_j$, where Δ_q is chosen such that $\mathtt{T}_i(q) = \Delta_q \cup \{b_q\varphi_q\}$.
- If Case 1 of Definition 5.3.24 holds and $b_q\varphi_q = b_q(Qx)\psi_q$ is a $\boldsymbol{\delta}$-formula, then there is a constant symbol $c \in \mathcal{L}^c$ that does not occur in $\mathtt{T}_i(q)$. This is the case because there are infinitely many constant symbols in $\mathcal{L}^c$ that do not belong to $\mathtt{T}_i(\lambda) = \Delta$ and at each expansion of a node of $\mathtt{T}_i$ we add a finite set of signed formulas and, therefore, a finite set of

new constant symbols. Define $\mathtt{T}_{i+1}$ by adding $q0$ to $\mathrm{Dom}(\mathtt{T}_i)$ and letting $\mathtt{T}_{i+1}(q0) = \Delta_q \cup \{b_q(\psi_q)_{x:=c}\}$, where Δ_q is chosen such that $\mathtt{T}_i(q) = \Delta_q \cup \{b_q\varphi_q\}$ and c is the first constant symbol in $\mathcal{L}^c$ that does not occur in $\mathtt{T}_i(q)$.

- If Case 2 of Definition 5.3.24 occurs, define $\mathtt{T}_{i+1}$ by adding $q0$ to $\mathrm{Dom}(\mathtt{T}_i)$ and letting $\mathtt{T}_{i+1}(q0) = \mathtt{T}_i(q) \cup \{b_q\langle\psi_q\rangle_{x:=t_q}\}$.
- If Case 3 of Definition 5.3.24 occurs, define $\mathtt{T}_{i+1}$ be adding $q0$ to $\mathrm{Dom}(\mathtt{T}_i)$ and letting $\mathtt{T}_{i+1}(q0) = \mathtt{T}_i(q) \cup \{b_q\varphi_q\}$.

Proof of Correctness: We begin by observing that for each i such that $\mathtt{T}_{i+1}$ is defined, there is a unique node q such that q is a leaf of $\mathtt{T}_i$ but not of $\mathtt{T}_{i+1}$. We will refer to q as the *node expanded at stage* $i+1$ of the construction. Further, if $i < j$ and $\mathtt{T}_{j+1}$ is defined, then the node expanded at stage $j+1$ is different from the node expanded at stage $i+1$. It follows that for each n, there are only finitely many i such that the node expanded at stage i has length n.

We also observe that if $q \in \mathrm{Dom}(\mathtt{T}_i)$, q is not a leaf of $\mathtt{T}_i$, $\mathtt{P}$ is the path leading to q and $(b_q\varphi_q, t_q)$ is the first pair in $\mathrm{SFORM}_{\mathcal{L}^c}(V) \times \mathrm{TERM}_{\mathcal{L}^c}(V)$ that requires attention in $\mathtt{T}_i$ at q, then for every immediate descendant qj of q, the pair $(b_q\varphi_q, t_q)$ does not require attention in $\mathtt{T}_i$ at qj.

Suppose that $\mathtt{T}$ is not completed. Then, there is a branch $\mathtt{B}$ that is neither closed not complete. Let $(b\varphi, t)$ be the first pair in $\mathrm{SFORM}_{\mathcal{L}^c}(V) \times \mathrm{TERM}_{\mathcal{L}^c}(V)$ that requires attention in $\mathtt{T}$ on $\mathtt{B}$. Let $q \in \mathtt{B}$ be such that (i) $b\varphi \in \mathtt{T}(\mathtt{P})$, if $b\varphi \in \mathtt{T}(\mathtt{B})$, and (ii) none of the pairs $(b'\varphi', t')$ that precede $(b\varphi, t)$ require attention in $\mathtt{T}$ at q, where $\mathtt{P}$ is the path leading to q.

Since $\mathtt{T}$ is the union of the sequence of conservative tableaux $\mathtt{T}_0, \mathtt{T}_1, \ldots$, it is clear that $\mathtt{T}$ itself is a conservative tableau.

Suppose initially that $\mathtt{B}$ contains some immediate descendant qj of q and let k be such that $qj \in \mathrm{Dom}(\mathtt{T}_k)$. Then, $\mathtt{T}_k(\mathtt{P}) = \mathtt{T}(\mathtt{P})$, so $(b\varphi, t)$ is the first pair that requires attention in $\mathtt{T}_k$ at q. Thus, $(b\varphi, t)$ does not require attention in $\mathtt{T}_k$ at qj, which means that $(b\varphi, t)$ does not require attention in $\mathtt{T}$ on $\mathtt{B}$. We thus obtain a contradiction.

If there is no such immediate descendant, $\mathtt{B}$ is the path leading to q. Choose k such that $q \in \mathrm{Dom}(\mathtt{T}_k)$ and, at stage $k+1$, no node r such that $|r| \leq |q|$ is expanded. Then, q is a leaf of $\mathtt{T}_k$, $\mathtt{T}_k(\mathtt{P})$ is not closed, $b\varphi \in \mathtt{T}_k(\mathtt{P})$ if $b\varphi \in \mathtt{T}(\mathtt{B})$, and the pair $(b\varphi, t)$ requires attention in $\mathtt{T}_k$ at q. This makes q a node eligible to be expanded at stage $k+1$, so the node expanded at stage $k+1$ must have length no greater than $|q|$, thus contradicting our choice of k. □

Theorem 5.3.26 (Completeness Theorem for Tableaux of First-Order Logic). *Let $\mathcal{L}$ be a first-order language, V be an $\mathcal{L}$-suitable set of variables and Δ be a set of signed $(\mathcal{L}, V)$-formulas. If Δ is unsatisfiable, then there exists a finite, conservative, and closed $(\Delta, \mathcal{L}^c, V)$-tableau.*

Proof. The proof is identical to that of Theorem 3.3.34. □

Corollary 5.3.27. *Let Δ be a set of signed $(\mathcal{L}, V)$-formulas. Then, the following conditions are equivalent.*

(1) *Δ is unsatisfiable.*
(2) *There exists a finite, conservative, and closed $(\Delta, \mathcal{L}^c, V)$-tableau.*
(3) *There exists a conservative, closed $(\Delta, \mathcal{L}^c, V)$-tableau.*

Proof. This follows immediately from Theorems 5.3.19 and 5.3.26. □

The next construction yields a more stringent form of completed tableau that will be useful in studying other formal systems.

Construction 5.3.28.
Input: A first-order language $\mathcal{L}$, a set of variables V and a set Δ of signed $(\mathcal{L}, V)$-formulas.

Output: A sequence $\mathtt{T}_0, \mathtt{T}_1, \ldots$ of finite $(\Delta, \mathcal{L}^c, V)$-tableaux such that each $\mathtt{T}_{i+1}$ is a leaf extension of $\mathtt{T}_i$ and $\mathtt{T} = \bigcup\{\mathtt{T}_i \mid i \geq 0\}$ is a conservative, strongly completed $(\Delta, \mathcal{L}^c, V)$-tableau.

Method:

(A) Let $\mathtt{T}_0$ be the one-node $(\Delta, \mathcal{L}^c, V)$-tableau with root labeled by Δ.

(B) Suppose that $\mathtt{T}_i$ has been defined. Then, if $\mathtt{T}_i$ is strongly completed, the construction stops with $\mathtt{T}_i$. Otherwise, $\mathtt{T}_i$ has branches that are neither strongly closed nor complete. Select nondeterministically among the shortest such branches a branch $\mathtt{B}$ ending in the leaf q.
Choose $(b_q\varphi_q, t_q) \in \mathtt{T}_i(\mathtt{B}) \times \mathrm{TERM}_{\mathcal{L}^c}(V)$ to be the first pair $(b\varphi, t)$ in the standard order of $\mathrm{SFORM}_{\mathcal{L}^c}(V) \times \mathrm{TERM}_{\mathcal{L}^c}(V)$ that requires attention in $\mathtt{T}_i$ at q. (We can show the existence of such a pair as follows. Since $\mathtt{T}_i(\mathtt{B})$ is not a Hintikka set,

either $\{\mathbf{T}\psi, \mathbf{F}\psi\} \subseteq \mathtt{T}_i(\mathtt{B})$ for some atomic formula ψ or such a pair exists. By Lemma 5.3.23, in the first case, we would have $\{\mathbf{T}\psi, \mathbf{F}\psi\} \subseteq \mathtt{T}_i(q)$, which would contradict the fact that $\mathtt{B}$ is not strongly closed.) By Lemma 5.3.23, if Case 1 or 2 of Definition 5.3.24 applies, $b_q\varphi_q \in \mathtt{T}_i(q)$.
The conservative tableau $\mathtt{T}_{i+1}$ is defined in exactly the same way as in Construction 5.3.25.

Proof of Correctness: Note that the statements contained in the first two paragraphs of the proof of correctness of Construction 5.3.25 remain valid for the current construction. Also, one can show by induction on i that if $\mathtt{T}_i(q)$ is defined, then, for some $j \leq i$, q is a leaf of $\mathtt{T}_j$. In addition, if $\mathtt{T}_i(q)$ is defined, q is a leaf of $\mathtt{T}_i$, and $\mathtt{T}_i(q)$ contains both $\mathbf{T}\varphi$ and $\mathbf{F}\varphi$ for some formula φ, then for every $k \geq i$, q is a leaf of $\mathtt{T}_k$ and, therefore, q is a leaf of $\mathtt{T}$. This can be verified by induction on k.

The tableau $\mathtt{T}$ is conservative as a union of a sequence of conservative tableaux. Suppose that $\mathtt{T}$ is not strongly completed. Then, there is a branch $\mathtt{B}$ that is neither strongly closed not complete. Observe that $\mathtt{T}(\mathtt{B})$ cannot contain both $\mathbf{T}\psi$ and $\mathbf{F}\psi$ for any atomic formula ψ. Indeed, if r, r' are two nodes of $\mathtt{B}$ such that $\mathbf{T}\psi \in \mathtt{T}(r)$ and $\mathbf{F}\psi \in \mathtt{T}(r')$, then, by Lemma 5.3.23, q, the longer of r and r', is such that $\{\mathbf{T}\psi, \mathbf{F}\psi\} \subseteq \mathtt{T}(q)$. By the previous remark, there is an h such that q is a leaf of $\mathtt{T}_h$ and consequently, q is a leaf of $\mathtt{T}$. This implies that $\mathtt{B}$ is strongly closed, which contradicts our assumption. Thus, $\mathtt{T}(\mathtt{B})$ satisfies the first condition of the definition of a Hintikka set. The noncompleteness of $\mathtt{B}$ implies the existence of a pair in $\mathrm{SFORM}_{\mathcal{L}^c}(V) \times \mathrm{TERM}_{\mathcal{L}^c}(V)$ that requires attention in $\mathtt{T}$ on $\mathtt{B}$. Let $(b\varphi, t)$ be the first such pair and let $q \in \mathtt{B}$ be such that (i) $b\varphi \in \mathtt{T}(\mathtt{P})$, if $b\varphi \in \mathtt{T}(\mathtt{B})$, and (ii) none of the pairs $(b'\varphi', t')$ that precede $(b\varphi, t)$ require attention in $\mathtt{T}$ at q, where $\mathtt{P}$ is the path leading to q.

Suppose initially that $\mathtt{B}$ contains some immediate descendant qj of q and let k be such that $qj \in \mathrm{Dom}(\mathtt{T}_k)$. Then, $\mathtt{T}_k(\mathtt{P}) = \mathtt{T}(\mathtt{P})$, so $(b\varphi, t)$ is the first pair that requires attention in $\mathtt{T}_k$ at q. Thus, $(b\varphi, t)$ does not require attention in $\mathtt{T}_k$ at qj, which means that $(b\varphi, t)$ does not require attention in $\mathtt{T}$ on $\mathtt{B}$. We thus obtain a contradiction.

If there is no such immediate descendant, $\mathtt{B}$ is the path leading to q. Choose k such that $q \in \mathrm{Dom}(\mathtt{T}_k)$ and, at stage $k + 1$, no node r such that $|r| \leq |q|$ is expanded. Then, q is a leaf of $\mathtt{T}_k$, $\mathtt{P}$ is not strongly closed, $b\varphi \in \mathtt{T}_k(\mathtt{P})$ if $b\varphi \in \mathtt{T}(\mathtt{B})$, and the pair $(b\varphi, t)$ requires attention in $\mathtt{T}_k$ at q. This makes q a node eligible to be expanded at stage $k+1$, so the node expanded at stage $k+1$ must have length no greater than $|q|$, thus contradicting our choice of k. □

We saw earlier that a strongly closed $(\Delta, \mathcal{L}, V)$-tableau is finite.

Theorem 5.3.29 (Strong Completeness Theorem for Tableaux of First-Order Logic). *Let Δ be a set of signed $(\mathcal{L}, V)$-formulas. If Δ is unsatisfiable, then there exists a conservative, strongly closed (hence, finite) $(\Delta, \mathcal{L}^c, V)$-tableau.*

Proof. If Δ is unsatisfiable, then Construction 5.3.28 yields a conservative strongly completed $(\Delta, \mathcal{L}^c, V)$-tableau $\mathtt{T}$ which, by Theorem 5.3.18, is closed. Since a strongly completed closed $(\Delta, \mathcal{L}^c, V)$-tableau is strongly closed, we are done. □

Generalizing a remark we made earlier, we note that Theorem 5.3.29 can be strengthened by asserting the existence of strongly closed tableaux with retention or with removal.

Corollary 5.3.30. *Let Δ be a set of signed $(\mathcal{L}, V)$-formulas. Then, the following conditions are equivalent.*

(1) *Δ is unsatisfiable.*
(2) *There exists a finite, conservative, and closed $(\Delta, \mathcal{L}^c, V)$-tableau.*
(3) *There exists a conservative, closed $(\Delta, \mathcal{L}^c, V)$-tableau.*
(4) *There exists a conservative, strongly closed $(\Delta, \mathcal{L}^c, V)$-tableau.*
(5) *There exists a strongly closed $(\Delta, \mathcal{L}^c, V)$-tableau.*

Proof. This follows from Corollary 5.3.27, Theorem 5.3.29, and Theorem 5.3.18. □

Example 5.3.31. Let $\mathcal{L} = \{R, c\}$ be a first-order language, where R is a unary relation symbol and c is a constant symbol and let

$$\Delta = \{\mathbf{F}((\exists x)(\forall y)R(x,y) \to (\forall y)(\exists x)R(x,y))\}.$$

Figure 5.10 contains a conservative, strongly closed $(\Delta, \mathcal{L}^c)$-tableau, where d is a constant symbol in $\mathcal{L}^c - \mathcal{L}$.

▯

$$\begin{tikzcd}
\boxed{\mathbf{F}((\exists x)(\forall y)R(x,y) \to (\forall y)(\exists x)R(x,y))} \arrow[d, dash, "0"] \\
\boxed{\mathbf{T}(\exists x)(\forall y)R(x,y), \mathbf{F}(\forall y)(\exists x)R(x,y)} \arrow[d, dash, "0"] \\
\boxed{\mathbf{T}(\forall y)R(c,y), \mathbf{F}(\forall y)(\exists x)R(x,y)} \arrow[d, dash, "0"] \\
\boxed{\mathbf{T}(\forall y)R(c,y), \mathbf{F}(\exists x)R(x,d)} \arrow[d, dash, "0"] \\
\boxed{\begin{array}{l}\mathbf{T}(\forall y)R(c,y), \mathbf{F}(\exists x)R(x,d),\\ \mathbf{T}R(c,d)\end{array}} \arrow[d, dash, "0"] \\
\boxed{\begin{array}{l}\mathbf{T}(\forall y)R(c,y), \mathbf{F}(\exists x)R(x,d),\\ \mathbf{T}R(c,d), \mathbf{F}R(c,d)\end{array}}
\end{tikzcd}$$

Fig. 5.10. A strongly closed $(\Delta, \mathcal{L}^c)$-tableau.

As in the propositional case, we can use tableaux to give an alternative proof of the Compactness Theorem (in a formulation using signed formulas).

Theorem 5.3.32. *There is an effective, syntactic construction that begins with a strongly closed $(\Delta, \mathcal{L}, V)$-tableau $\mathtt{T}$, where $\mathcal{L}$ is a first-order language and V is an $\mathcal{L}$-suitable set of variables and produces a strongly closed $(\Delta_0, \mathcal{L}, V)$-tableau $\mathtt{T}_0$, where Δ_0 is a finite subset of Δ.*

Proof. We proceed recursively on the definition of $\mathtt{T}$. If $\mathtt{T}$ is a one node tree, then Δ contains $\mathbf{T}\varphi, \mathbf{F}\varphi$ for some formula φ, so $\mathtt{T}_0$ is the one node tableau with $\mathtt{T}_0(\lambda) = \{\mathbf{T}\varphi, \mathbf{F}\varphi\}$ and $\Delta_0 = \{\mathbf{T}\varphi, \mathbf{F}\varphi\}$.

When $\mathtt{T}$ has more than one node, several cases occur depending on the type of expansion used at the root of $\mathtt{T}$. If propositional expansion is used, then the argument is virtually identical to the one used in Theorem 3.3.40.

We consider now the remaining cases.

(1) If $\boldsymbol{\gamma}$-expansion is used at the root of $\mathtt{T}$, let the formula expanded at the root be $b(Qx)\psi$. The root has one immediate descendant, and $\mathtt{T}(0) = \Delta \cup \{b(\psi')_{x:=t}\}$ for some term $t \in \mathrm{TERM}_{\mathcal{L}}(V)$ and variant ψ' of ψ such that t is substitutable for x in ψ. By the inductive hypothesis, we construct a finite subset Δ'_0 of $\mathtt{T}(0)$ and a strongly closed $(\Delta'_0, \mathcal{L}, V)$-tableau $\mathtt{T}'$. If $\Delta'_0 \subseteq \Delta$, the construction returns $\mathtt{T}'$; otherwise, let Δ_0 be the finite subset $(\Delta'_0 \cap \Delta) \cup \{b(Qx)\psi\}$ of Δ. By applying thinning, we obtain the tableau $\mathtt{T}'' = (\mathtt{T}'; \Delta_0 \cup \{b(\psi')_{x:=t}\})$. Finally, let $\mathtt{T}_0 = (\mathtt{T}''; \Delta_0)$. Note that this a strongly closed Δ_0-tableau in which we have applied γ-expansion at the root.

(2) If nondegenerate $\boldsymbol{\delta}$-expansion is used at the root, let $b(Qx)\psi$ be the formula expanded at the root. Then, for some set of signed formulas Δ', we have $\Delta = \Delta' \cup \{b(Qx)\psi\}$, the root has one immediate descendant, and $\mathtt{T}(0) = \Delta' \cup \{b(\psi)_{x:=c}\}$ for some constant symbol c that does not occur in any signed formula in Δ. By inductive hypothesis, there is a finite subset Δ'_0 of $\Delta' \cup \{b(\psi)_{x:=c}\}$ and a strongly closed $(\Delta'_0, \mathcal{L}, V)$-tableau $\mathtt{T}'$. If $\Delta'_0 \subseteq \Delta'$, we return the tableau $\mathtt{T}'$. Otherwise, consider the finite subset $\Delta_0 = (\Delta'_0 \cap \Delta') \cup \{b(Qx)\psi\}$ of Δ. Now return $\mathtt{T}_0 = (\mathtt{T}'; \Delta_0)$. Since $\Delta'_0 = (\Delta'_0 \cap \Delta') \cup \{b(\psi)_{x:=c}\}$, $\mathtt{T}_0$ is a strongly closed $(\Delta_0, \mathcal{L}, V)$-tableau that uses $\boldsymbol{\delta}$-expansion at the root.
If degenerate $\boldsymbol{\delta}$-expansion was used at the root, the argument is similar, except that ψ is used in place of $(\psi)_{x:=c}$.

(3) If variantization was used at the root of $\mathtt{T}$, then we have $\mathtt{T}(0) = \Delta'$ and there is a finite-to-one function $f : \Delta' \longrightarrow \Delta$ such that for all $b\alpha \in \Delta'$, $f(b\alpha)$ is a variant of $b\alpha$. By inductive hypothesis, we obtain a finite subset Δ'_0 of Δ' and a strongly closed $(\Delta'_0, \mathcal{L}, V)$-tableau $\mathtt{T}'$. The construction returns $\mathtt{T}_0 = (\mathtt{T}'; f(\Delta'_0))$, which is a strongly closed tableau obtained by applying variantization at the root.

(4) If $= \in \mathcal{L}$ and equality expansion is used at the root of $\mathtt{T}$, then the root has one immediate descendant, and $\mathtt{T}(0) = \Delta \cup \{\mathbf{T}\alpha\}$ for

some $\alpha \in \mathrm{INST}_{\mathcal{L},V}(\mathrm{Eq}_{=,\mathcal{L}})$. By the inductive hypothesis, there is a finite subset Δ_0' of $\Delta \cup \{\mathbf{T}\alpha\}$ and a strongly closed $(\Delta_0', \mathcal{L}, V)$-tableau $\mathtt{T}'$. If $\Delta_0' \subseteq \Delta$, we return $\mathtt{T}'$. Otherwise, let Δ_0 be the finite subset $\Delta_0' - \{\mathbf{T}\alpha\}$ of Δ. The construction returns $\mathtt{T}_0 = (\mathtt{T}'; \Delta_0)$, which uses equality expansion at the root.

□

Theorem 5.3.33 (Compactness Theorem for Signed Formulas of First-Order Logic). *Let Δ be a set of signed $\mathcal{L}$-formulas. Then Δ is satisfiable if and only if every finite subset of Δ is satisfiable.*

Proof. If Δ is satisfiable, then clearly every finite subset of Δ is satisfiable.

Conversely, let Δ be unsatisfiable and let $V = \mathtt{FV}(\Delta)$. There is a conservative, (finite) strongly closed $(\Delta, \mathcal{L}^c, V)$-tableau $\mathtt{T}$ by the Strong Completeness Theorem for Tableaux. By Theorem 5.3.32, there is a strongly closed $(\Delta_0, \mathcal{L}^c, V)$-tableau, where Δ_0 is a finite subset of Δ. By the Soundness Theorem, Δ_0 is unsatisfiable. □

Corollary 5.3.34. *Let $\mathtt{T}$ be a conservative $(\Delta, \mathcal{L}, V)$-tableau, where $\mathcal{L}$ is a first-order language, V is an $\mathcal{L}$-suitable set of variables, and Δ is a set of signed $(\mathcal{L}, V)$-formulas. If Δ is satisfiable, then there is a branch $\mathtt{B}$ of $\mathtt{T}$ such that $\mathtt{T}(\mathtt{B})$ is satisfiable.*

Proof. This follows from Corollary 5.3.16 and the Compactness Theorem. □

As we did in propositional logic, we will restate some of the results obtained in this section in terms of formal systems.

Definition 5.3.35. Let $\mathcal{L}$ be a first-order language and V be an $\mathcal{L}$-suitable set of variables. $\mathcal{F}_{\mathcal{L},V}^{\text{tabl,cons}}$ is the formal system whose set of objects is the collection of all sets of signed $(\mathcal{L}, V)$-formulas, set of axioms is the collection of all closed sets of such formulas, and set of rules of inference consists of:

- *The Propositional Rule*:

$$\frac{\Delta \cup K_0, \ldots, \Delta \cup K_{n-1}}{\Delta \cup \{b\varphi\}}\ \mathsf{R}_{prop}$$

where $b\varphi$ is neither a $\boldsymbol{\gamma}$- nor a $\boldsymbol{\delta}$-formula, φ is not atomic and the sequence of $(\mathcal{L}, V)$-constituents of $b\varphi$ is $\mathtt{d}_{\mathcal{L},V}(b\varphi) = (K_0, \ldots, K_{n-1})$.

- *The $\boldsymbol{\gamma}$-Rule*:

$$\frac{\Delta \cup \{b(Qx)\psi\} \cup \{b(\psi')_{x:=t}\}}{\Delta \cup \{b(Qx)\psi\}}\ \mathsf{R}_{\gamma}$$

where $b(Qx)\psi$ is a $\boldsymbol{\gamma}$-formula,$t \in \mathrm{TERM}_{\mathcal{L}}(V)$ and ψ' is a variant of ψ such that t is substitutable for x in ψ'.

- *The $\boldsymbol{\delta}$-Rule*:

$$\frac{\Delta \cup \{b(\psi)_{x:=c}\}}{\Delta \cup \{b(Qx)\psi\}}\ \mathsf{R}_{\delta}$$

where $b(Qx)\psi$ is a $\boldsymbol{\delta}$-formula and c is a constant symbol of $\mathcal{L}$ that does not occur in $\Delta \cup \{b(Qx)\psi\}$ and $x \in \mathtt{FV}(\psi)$.

- *The Degenerate $\boldsymbol{\delta}$-Rule*:

$$\frac{\Delta \cup \{b\psi\}}{\Delta \cup \{b(Qx)\psi\}}\ \mathsf{R}_{d\delta}$$

where $b(Qx)\psi$ is a $\boldsymbol{\delta}$-formula and $x \notin \mathtt{FV}(\psi)$.

- If $= \in \mathcal{L}$, we include the *Equality Rule*:

$$\frac{\Delta \cup \{\mathbf{T}\alpha\}}{\Delta}\ \mathsf{R}_{=}$$

where $\alpha \in \mathrm{INST}_{\mathcal{L},V}(\mathrm{Eq}_{=,\mathcal{L}})$.

If we add the *variant rule*

$$\frac{\Delta'}{\Delta}\ \mathsf{R}_{vrt}$$

where Δ, Δ' are sets of signed $(\mathcal{L}, V)$-formulas for which there is a finite-to-one function $f : \Delta' \to \Delta$ such that each formula $b\varphi$ of Δ' is a variant of the formula $f(b\varphi)$ of Δ, we denote the expanded formal system by $\mathcal{F}^{\text{tabl}}_{\mathcal{L},V}$.

We will denote the formal systems $\mathcal{F}^{\text{tabl,cons}}_{\mathcal{L},\emptyset}$ and $\mathcal{F}^{\text{tabl}}_{\mathcal{L},\emptyset}$ by $\mathcal{F}^{\text{tabl,cons}}_{\mathcal{L}}$ and $\mathcal{F}^{\text{tabl}}_{\mathcal{L}}$, respectively. □

Denote by $\mathsf{SCTCONS}(\mathcal{L}, V)$ the set of all $(\mathcal{L}, V)$-tableaux $\mathtt{T}$ such that $\mathtt{T}$ is a conservative, strongly closed $(\mathtt{T}(\lambda), \mathcal{L}, V)$-tableau. General $\mathcal{F}^{\text{tabl,cons}}_{\mathcal{L},V}$-deduction trees for Δ and conservative $(\Delta, \mathcal{L}, V)$-tableaux are the same, as are $\mathcal{F}^{\text{tabl,cons}}_{\mathcal{L},V}$-proof trees for Δ and conservative, strongly closed $(\Delta, \mathcal{L}, V)$-tableaux. Thus, the set $\mathcal{PT}_{\mathcal{F}^{\text{tabl,cons}}_{\mathcal{L},V}}$ equals the set $\mathsf{SCTCONS}(\mathcal{L}, V)$.

If $\mathsf{SCT}(\mathcal{L}, V)$ is the set of all strongly closed $(\mathcal{L}, V)$-tableaux $\mathtt{T}$ such that $\mathtt{T}$ is a strongly closed $(\mathtt{T}(\lambda), \mathcal{L}, V)$-tableau, similar observations about the identity between proof trees and deduction trees in the formal system $\mathcal{F}^{\text{tabl}}_{\mathcal{L},V}$ apply.

Theorem 5.3.36 (Soundness of $\mathcal{F}^{\textbf{tabl}}_{\mathcal{L},V}$). *Let $\mathcal{L}$ be a first-order language and V be an $\mathcal{L}$-suitable set of variables. For every theorem Δ of $\mathcal{F}^{tabl}_{\mathcal{L},V}$, Δ is an unsatisfiable set of $(\mathcal{L}, V)$-formulas.*

Proof. If Δ is a theorem of $\mathcal{F}^{\text{tabl}}_{\mathcal{L},V}$, there is an $\mathcal{F}^{\text{tabl}}_{\mathcal{L},V}$-proof tree $\mathtt{T}$ for Δ which is a strongly closed $(\Delta, \mathcal{L}, V)$-tableau. By Theorem 5.3.19, Δ is unsatisfiable. □

Corollary 5.3.37. *Let $\mathcal{L}$ be a first-order language and V be an $\mathcal{L}$-suitable set of variables. For every theorem Δ of $\mathcal{F}^{tabl,cons}_{\mathcal{L},V}$, Δ is an unsatisfiable set of $(\mathcal{L}, V)$-formulas.*

Proof. The soundness of $\mathcal{F}^{\text{tabl}}_{\mathcal{L},V}$ clearly implies the soundness of $\mathcal{F}^{\text{tabl,cons}}_{\mathcal{L},V}$. □

Theorem 5.3.38 (Partial Completeness of $\mathcal{F}^{\textbf{tabl,cons}}_{\mathcal{L}^c,V}$). *Let $\mathcal{L}$ be a first-order language and V be a set of variables. If Δ is an unsatisfiable set of signed $(\mathcal{L}, V)$-formulas, then Δ is a theorem of $\mathcal{F}^{tabl,cons}_{\mathcal{L}^c,V}$.*

Proof. Since Δ is unsatisfiable, by Theorem 5.3.29 there is a conservative, strongly closed $(\Delta, \mathcal{L}^c, V)$-tableau which is an $\mathcal{F}^{\text{tabl,cons}}_{\mathcal{L}^c,V}$-proof tree for Δ. □

Corollary 5.3.39. *Let $\mathcal{L}$ be a first-order language and V be a set of variables. If Δ is an unsatisfiable set of signed $(\mathcal{L}, V)$-formulas, then Δ is a theorem of $\mathcal{F}^{tabl}_{\mathcal{L}^c,V}$.*

Proof. The partial completeness of $\mathcal{F}^{\text{tabl,cons}}_{\mathcal{L},V}$ implies immediately the partial completeness of $\mathcal{F}^{\text{tabl}}_{\mathcal{L},V}$. □

Corollary 5.3.40. *Let $\mathcal{L}$ be a first-order language, V be a set of variables and Δ be a set of signed $(\mathcal{L}, V)$-formulas. Then, the following three statements are equivalent.*

(1) Δ *is unsatisfiable*;
(2) Δ *is a theorem of* $\mathcal{F}^{tabl,cons}_{\mathcal{L}^c,V}$;
(3) Δ *is a theorem of* $\mathcal{F}^{tabl}_{\mathcal{L}^c,V}$.

Proof. This follows immediately from Theorems 5.3.36 (applied to $\mathcal{L}^c$) and 5.3.38 and the fact that $\mathcal{F}^{\text{tabl}}_{\mathcal{L}^c,V}$ is an extension of $\mathcal{F}^{\text{tabl,cons}}_{\mathcal{L}^c,V}$. □

The term "partial completeness" is used in Theorem 5.3.38 rather than "completeness" because we do not show that the collection of unsatisfiable sets of signed $(\mathcal{L}, V)$-formulas is the same as the collection of theorems of $\mathcal{F}^{\text{tabl,cons}}_{\mathcal{L},V}$. The next example shows that it is possible to have an unsatisfiable set of signed $(\mathcal{L}, V)$-formulas that is not a theorem of $\mathcal{F}^{\text{tabl,cons}}_{\mathcal{L},V}$. However, as Exercise 12 shows, it is possible to obtain a completeness result if the language $\mathcal{L}$ contains an infinite set of constant symbols and we limit attention to finite sets of signed $\mathcal{L}$-formulas.

Example 5.3.41. Let $\mathcal{L} = \{R, c\}$, where R is unary relation symbol and c is a constant symbol. The set of signed $\mathcal{L}$-formulas

$$\Delta = \{\mathbf{T}(\forall y)R(c, y), \mathbf{F}(\forall y)(\exists x)R(x, y)\}$$

is unsatisfiable because `T` is a (conservative) strongly closed $(\Delta, \mathcal{L}^c)$-tableau, where `T` is the tableau shown in Figure 5.11.

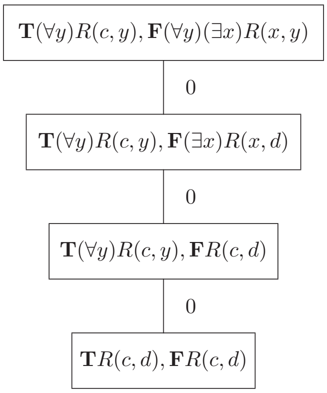

$\mathbf{T}(\forall y)R(c,y), \mathbf{F}(\forall y)(\exists x)R(x,y)$
0
$\mathbf{T}(\forall y)R(c,y), \mathbf{F}(\exists x)R(x,d)$
0
$\mathbf{T}(\forall y)R(c,y), \mathbf{F}R(c,d)$
0
$\mathbf{T}R(c,d), \mathbf{F}R(c,d)$

Fig. 5.11. Conservative strongly closed tableau.

We claim that there is no strongly closed $(\Delta, \mathcal{L})$-tableau. Let $\mathtt{T}'$ be a $(\Delta, \mathcal{L})$-tableau and let $q \in \mathrm{Dom}(\mathtt{T}')$. Define the sets of signed formulas

$$\Delta_0 = \{\mathbf{T}\varphi_0 \mid \varphi_0 \text{ is a variant of } (\forall y)R(c, y)\},$$
$$\Delta_1 = \{\mathbf{F}\varphi_1 \mid \varphi_1 \text{ is a variant of } (\forall y)(\exists x)R(x, y)\},$$
$$\Delta_2 = \{\mathbf{T}\varphi_2 \mid \varphi_2 \text{ is a variant of } (\exists x)R(x, c)\}.$$

We will prove that one of the following two alternatives holds:

$$\mathtt{T}(q) \subseteq \Delta_0 \cup \Delta_1 \cup \{\mathbf{T}R(c, c)\}, \tag{5.1}$$
$$\mathtt{T}(q) \subseteq \Delta_1 \cup \Delta_2 \cup \{\mathbf{F}R(c, c)\}. \tag{5.2}$$

The proof is by induction on $|q|$. The statement holds in the basis when $q = \lambda$. Suppose that $|q| > 0$ and the statement holds for its predecessor q'.

We first consider the case where q' satisfies the inclusion (5.1). If a formula $\mathbf{T}(\forall y')R(c, y')$ is expanded in q', then $\mathtt{T}(q)$ satisfies the same inclusion, because c is the unique ground term of $\mathcal{L}$. If a formula $\mathbf{F}(\forall y')(\exists x')R(x', y')$ is expanded (which is possible only if $\mathtt{T}(q') \subseteq \Delta_1$), then $\mathtt{T}(q) \subseteq \Delta_1 \cup \Delta_2$, which means that $\mathtt{T}(q)$ satisfies inclusion (5.2). If variantizing takes place at q', then the same inclusion holds for q.

Three subcases may occur when q' satisfies the inclusion (5.2). If variantizing was used to produce $\mathtt{T}(q)$, then the same inclusion holds for q. If a formula $\mathbf{F}(\forall y')(\exists x')R(x', y')$ was expanded, the expansion may produce only $\mathbf{F}(\exists x')R(x', c)$, which implies that $\mathtt{T}(q)$ satisfies the same inclusion. The last case, when a formula $\mathbf{F}(\exists x')R(x', c)$ is expanded yields $\mathbf{F}R(c, c)$ and the same conclusion follows.

Thus, $\mathbf{T}R(c, c), \mathbf{F}R(c, c)$ cannot coexist in any node of $\mathtt{T}$, which prevents $\mathtt{T}$ from being strongly closed. $\square$

It is clear that the formal systems $\mathcal{F}^{\text{tabl,cons}}_{\mathcal{L},V}$ and $\mathcal{F}^{\text{tabl}}_{\mathcal{L},V}$ remain sound if in the definition of any of the first three rules, we add one of the additional restrictions $b\varphi \in \Delta$ or $b\varphi \notin \Delta$, corresponding to tableaux with retention and tableaux with removal, respectively. By the remark preceding Corollary 5.3.30, the partial completeness of $\mathcal{F}^{\text{tabl,cons}}_{\mathcal{L}^c,V}$ and $\mathcal{F}^{\text{tabl}}_{\mathcal{L}^c,V}$ is preserved if we make the same modifications to the rules.

Tableaux for sets of unsigned formulas can be developed in a manner parallel to the one used for tableaux for sets of signed formulas.

Definition 5.3.42. Let $\mathcal{L}$ be a first-order language and V be a set of variables. An *unsigned* $(\mathcal{L}, V)$*-tableau*, is a lot whose labels are sets of unsigned $(\mathcal{L}, V)$-formulas. □

The concepts introduced in Definitions 5.3.1 through 5.3.3 can be readily transferred to unsigned $(\mathcal{L}, V)$-tableaux. The definition of unsigned $(\Gamma, \mathcal{L}, V)$-tableau is slightly more complicated and we give it here.

Definition 5.3.43. Let $\mathcal{L}$ be a first-order language, V be an $\mathcal{L}$-suitable set of variables and Γ be a set of unsigned $(\mathcal{L}, V)$-formulas. A $(\Gamma, \mathcal{L}, V)$*-tableau* is an unsigned $(\mathcal{L}, V)$-tableau $\mathtt{T}$ that satisfies the following conditions:

- The root of $\mathtt{T}$ is labeled by Γ, i.e., $\mathtt{T}(\lambda) = \Gamma$.
- If q is an interior node of $\mathtt{T}$, then one of the following cases occurs:
  (1) there is some set of unsigned formulas Γ' and formula φ with φ not a literal and $\mathtt{T}(q) = \Gamma' \cup \{\varphi\}$ such that either
    (a) φ is neither a $\boldsymbol{\gamma}$- nor a $\boldsymbol{\delta}$-formula, the sequence of $(\mathcal{L}, V)$-constituents of φ is $\mathtt{d}_{\mathcal{L},V}(\varphi) = (K_0, \ldots, K_{n-1})$, q has n immediate descendants, and $\mathtt{T}(qi) = \Gamma' \cup K_i$ for $0 \leq i \leq n-1$, or
    (b) φ is a $\boldsymbol{\gamma}$-formula $(\forall x)\psi$, $\varphi \in \Gamma'$, q has one immediate descendant, and $\mathtt{T}(q0) = \Gamma' \cup \{(\psi')_{x:=t}\}$ for some term $t \in \mathrm{TERM}_{\mathcal{L}}(V)$ and variant ψ' of ψ such that t is substitutable for x in ψ', or
    (c) φ is a $\boldsymbol{\gamma}$-formula $(\neg(\exists x)\psi)$, $\varphi \in \Gamma'$, q has one immediate descendant, and $\mathtt{T}(q0) = \Gamma' \cup \{((\neg\psi'))_{x:=t}\}$ for some term $t \in \mathrm{TERM}_{\mathcal{L}}(V)$ and variant ψ' of ψ such that t is substitutable for x in ψ', or
    (d) φ is a $\boldsymbol{\delta}$-formula $(\exists x)\psi$, q has one immediate descendant, and $\mathtt{T}(q0) = \Gamma' \cup \{(\psi)_{x:=c}\}$ for some constant symbol $c \in \mathcal{L}$ that does not occur in any formula in $\mathtt{T}(q)$, if $x \in \mathtt{FV}(\psi)$. Otherwise, that is, if $x \notin \mathtt{FV}(\psi)$, $\mathtt{T}(q0) = \Gamma' \cup \{\psi\}$.
    (e) φ is a $\boldsymbol{\delta}$-formula $(\neg(\forall x)\psi)$, q has one immediate descendant, and $\mathtt{T}(q0) = \Gamma' \cup \{((\neg\psi))_{x:=c}\}$ for some constant

symbol $c \in \mathcal{L}$ that does not occur in any formula in $\mathtt{T}(q)$, if $x \in \mathtt{FV}(\psi)$. Otherwise, $\mathtt{T}(q0) = \Gamma' \cup \{(\neg\psi)\}$.

(2) q has one immediate descendant, $q0$, and there is a finite-to-one function $f : \mathtt{T}(q0) \longrightarrow \mathtt{T}(q)$ such that every formula φ in $\mathtt{T}(q0)$ is a variant of the formula $f(\varphi)$ in $\mathtt{T}(q)$.

(3) $= \in \mathcal{L}$, q has one immediate descendant, and $\mathtt{T}(q0) = \mathtt{T}(q) \cup \{\alpha\}$ for some $\alpha \in \mathrm{INST}_{\mathcal{L},V}(\mathrm{Eq}_{=,\mathcal{L}})$.

If the first case holds at an interior node q, then we say that *regular expansion* was used at q. If the second case holds, then we say that *variantizing* was used at q. When in the second case $\mathtt{T}(q0) \subseteq \mathtt{T}(q)$, then we say that *thinning* was used. Finally, if the third case holds, then we say that *equality expansion* was used at q. ▯

The notions of $(\Gamma, \mathcal{L}, V)$*-tableau with retention* and with *removal* parallel the corresponding notions for $(\Delta, \mathcal{L}, V)$-tableau. Similarly, the notion of *(locally) conservative unsigned tableau* can be introduced here by analogy with the corresponding notions that apply to signed tableaux. Further, in Exercises 18 to 25, we present results for unsigned tableaux similar to the ones discussed in this section for signed tableaux. Some of these results can be obtained alternatively by applying the translation given below in Algorithm 5.3.46. This translation is obtained using the function $\mathtt{u}$ defined after Exercise 159 in Section 4.15; namely, $\mathtt{u}(\mathbf{T}\varphi) = \varphi$ and $\mathtt{u}(\mathbf{F}\varphi) = (\neg\varphi)$ for every $\varphi \in \mathrm{FORM}$.

The following result helps to prove the correctness of the translation algorithm.

Theorem 5.3.44. *Let $\mathcal{L}$ be a first-order language, Γ be a set of $\mathcal{L}$-formulas, and V be an $\mathcal{L}$-suitable set of variables. If $\mathtt{T}$ is an unsigned $(\Gamma, \mathcal{L}, V)$-tableau of depth 1 such that the root has n immediate descendants and $\mathtt{T}_i$ is a strongly closed $(\mathtt{T}(i), \mathcal{L}, V)$-tableau for $0 \leq i \leq n-1$, then $(\mathtt{T}_0, \ldots, \mathtt{T}_{n-1}; \Gamma)$ is a strongly closed unsigned $(\Gamma, \mathcal{L}, V)$-tableau.*

Proof. This statement follows directly from Definition 5.3.43 and it is left to the reader. □

The following lemma explains how the translation of signed tableaux into unsigned tableaux takes place locally.

Lemma 5.3.45. *Let* $\mathtt{T}$ *be a signed* $(\Delta, \mathcal{L}, V)$*-tableau of depth 1 such that the root has* n *immediate descendants. Suppose that the expansion at the root of* $\mathtt{T}$ *does not involve regular expansion of a formula* $\mathbf{T}(\neg\psi)$*. If* $\mathtt{T}'$ *is the lot defined by* $\mathrm{Dom}(\mathtt{T}') = \mathrm{Dom}(\mathtt{T})$ *and* $\mathtt{T}'(q) = u(\mathtt{T}(q))$*, for* $q \in \mathrm{Dom}(\mathtt{T}')$*, then* $\mathtt{T}'$ *is an unsigned* $(u(\Delta), \mathcal{L}, V)$*-tableau. Moreover, the same type of expansion is used at the root of* $\mathtt{T}'$ *as at the root of* $\mathtt{T}$*.*

Proof. We leave this verification to the reader. □

Algorithm 5.3.46.
Input: A signed tableau $\mathtt{T}$ that is a strongly closed $(\Delta, \mathcal{L}, V)$-tableau for some set Δ of signed formulas.
Output: A strongly closed unsigned $(\mathtt{u}(\Delta), \mathcal{L}, V)$-tableau.
Method: If $\mathtt{T}$ is a one-node tree, then output the one-node tree $\mathtt{T}'$ with $\mathtt{T}'(\lambda) = \mathtt{u}(\mathtt{T}(\lambda))$.
If $\mathtt{T}$ has more than one node, we consider two cases.
If there is a signed formula $\mathbf{T}(\neg\varphi)$ and a set of signed formulas Δ' such that $\mathtt{T}(\lambda) = \Delta' \cup \{\mathbf{T}(\neg\varphi)\}$, λ has 1 immediate descendent in $\mathtt{T}$, and $\mathtt{T}(0) = \Delta' \cup \{\mathbf{F}\varphi\}$, then apply the algorithm recursively to the subtree $\mathtt{T}_{[0]}$ and output the unsigned tableau resulting from this application.
Otherwise, suppose that the root has n immediate descendants. Apply the algorithm recursively to each of the subtrees $\mathtt{T}_{[i]}$, $0 \leq i \leq n-1$ to obtain the n unsigned tableaux $\mathtt{T}'_0, \ldots, \mathtt{T}'_{n-1}$ and output the unsigned tableau $\mathtt{T}' = (\mathtt{T}'_0, \ldots, \mathtt{T}'_{n-1}; \mathtt{u}(\Delta))$.

Proof of Correctness: By Theorem 5.3.11, recursive calls of the algorithm are applied to the right kind of tableaux. Also, by induction on the number of nodes of the input tableau, one can easily verify that the algorithm always terminates.

We prove now by induction on the number of nodes of the input tableau $\mathtt{T}$ that if $\mathtt{T}$ is a strongly closed $(\Delta, \mathcal{L}, V)$-tableau, then the output is a strongly closed $(\mathtt{u}(\Delta), \mathcal{L}, V)$-tableau. The basis step is easy and is left to the reader. For the inductive step, we distinguish two cases as in the method.

Suppose initially that the first case occurs. Let $\mathbf{T}(\neg\varphi), \Delta'$ be as in the algorithm. Then, $\mathtt{T}_{[0]}$ is a $(\Delta' \cup \{\mathbf{F}\varphi\}, \mathcal{L}, V)$-tableau, so, by inductive hypothesis, the output of the algorithm applied to $\mathtt{T}_{[0]}$ is a strongly closed $(\mathtt{u}(\Delta' \cup \{\mathbf{F}\varphi\}), \mathcal{L}, V)$-tableau. Since $\mathtt{u}(\Delta' \cup \{\mathbf{F}\varphi\}) = \mathtt{u}(\Delta') \cup \{(\neg\varphi)\} = \mathtt{u}(\Delta' \cup \{\mathbf{T}(\neg\varphi)\}) = \mathtt{u}(\Delta)$, this output is actually a strongly closed $(\mathtt{u}(\Delta), \mathcal{L}, V)$-tableau, as desired.

In the second case, suppose that the root of $\mathtt{T}$ has n immediate descendants. By Lemma 5.3.45, the lot $\mathtt{S}$ with $\mathrm{Dom}(\mathtt{S}) = \{\lambda, 0, \ldots, n-1\}$ given by $\mathtt{S}(q) = \mathtt{u}(\mathtt{T}(q))$ for $q \in \mathrm{Dom}(\mathtt{S})$ is a depth one unsigned $(\mathtt{u}(\Delta), \mathcal{L}, V)$-tableau. By inductive hypothesis, $\mathtt{T}'_i$ is a strongly closed unsigned $(\mathtt{u}(\mathtt{T}(i)), \mathcal{L}, V)$-tableau, so, by Theorem 5.3.44, $\mathtt{T}'$ is a strongly closed, unsigned $(\mathtt{u}(\Delta), \mathcal{L}, V)$-tableau. $\square$

Note that if $\mathtt{T}$ is a conservative signed tableau, then Algorithm 5.3.46 produces a conservative unsigned tableau.

The following technical result is useful for transformations between formal systems and uses the notation $\mathsf{VARIANT}(\Delta, U)$ introduced in Definition 4.6.56.

Theorem 5.3.47. *Let* $\mathtt{T}$ *be a* $(\Delta, \mathcal{L}, V)$*-tableau and let* U *be a set of variables such that both* U *and its complement are infinite and for all* $q \in \mathrm{Dom}(\mathtt{T})$, $U \cap \mathtt{V}(\mathtt{T}(q)) = \emptyset$. *Define* $\mathtt{T}'$ *such that* $\mathrm{Dom}(\mathtt{T}') = \mathrm{Dom}(\mathtt{T})$ *and* $\mathtt{T}'(q) = \mathsf{VARIANT}(\mathtt{T}(q), U)$ *for every* $q \in \mathrm{Dom}(\mathtt{T}')$. *Then, the tableau* $\mathtt{T}'$ *is a* $(\mathsf{VARIANT}(\Delta, U), \mathcal{L}, V)$*-tableau.*

Proof. Since the set of free variables of $\mathsf{VARIANT}(\Delta', U)$ is the same as the set of free variables of Δ', for any set of signed formulas Δ', $\mathtt{FV}(\mathtt{T}(q)) \subseteq V$, for every node q of $\mathtt{T}'$.

Let q be an interior node of $\mathtt{T}$. The argument for local consistency for $\mathtt{T}'$ at q depends on the part of Definition 5.3.4 that was applied at q in $\mathtt{T}$. We omit the cases when propositional expansion and equality expansion were used at q. Suppose that $\mathtt{T}(q0)$ was obtained by variantization from $\mathtt{T}(q)$. Observe that in this case, $\mathsf{VARIANT}(\mathtt{T}(q0), U) \subseteq \mathsf{VARIANT}(\mathtt{T}(q), U)$ because each formula $b\psi$ in $\mathtt{T}(q0)$ is variant of a formula $b\varphi$ from $\mathtt{T}(q)$ and, by Theorem 4.6.60 $\mathsf{VARIANT}(b\psi, U) = \mathsf{VARIANT}(b\varphi, U)$. Thus, $\mathtt{T}'(q0)$ was obtained from $\mathtt{T}'(q)$ by thinning.

Suppose now that $\boldsymbol{\delta}$-expansion was applied at q, so $\mathtt{T}(q) = \Delta_0 \cup \{b(Qx)\psi\}$ and $\mathtt{T}(q0) = \Delta_0 \cup \{b(\psi)_{x:=c}\}$, where c is a constant symbol that does not occur in $\mathtt{T}(q)$. (If $x \in \mathtt{FV}(\psi)$, then the constant symbol c is in $\mathcal{L}$. Otherwise, c can be any constant symbol that does not occur in $\mathtt{T}(q)$.) Since two variants have the same constant symbols,

c does not occur in $\mathtt{T}'(q)$ either. Then, because $x \notin U$,

$$\mathtt{T}'(q) = \mathsf{VARIANT}(\Delta_0, U) \cup \{b(Qw)(\mathsf{VARIANT}(\psi, U))_{x:=w}\},$$

where w is the first variable in U that does not occur in $\mathsf{VARIANT}(\psi, U)$. One the other hand,

$$\begin{aligned}\mathtt{T}'(q0) &= \mathsf{VARIANT}(\Delta_0, U) \cup \{b\mathsf{VARIANT}((\psi)_{x:=c}, U)\}\\ &= \mathsf{VARIANT}(\Delta_0, U) \cup \{b(\mathsf{VARIANT}(\psi, U))_{x:=c}\}\\ &\quad \text{(by Theorem 4.6.58)}\\ &= \mathsf{VARIANT}(\Delta_0, U) \cup \{b((\mathsf{VARIANT}(\psi, U))_{x:=w})_{w:=c}\}.\end{aligned}$$

The final step in the previous sequence of equalities uses the fact that

$$\begin{aligned}&b((\mathsf{VARIANT}(\psi, U))_{x:=w})_{w:=c}\\ &\quad = b(\mathsf{VARIANT}(\psi, U))_{x,w:=c,c}\\ &\qquad \text{(by Theorem 4.3.86 because } w\\ &\qquad \text{ is substitutable for } x \text{ in } \mathsf{VARIANT}(\psi, U))\\ &\quad = b(\mathsf{VARIANT}(\psi, U))_{x:=c}\\ &\qquad \text{(because } w \text{ does not occur free in } \mathsf{VARIANT}(\psi, U))\end{aligned}$$

If γ expansion is used at q, then

$$\begin{aligned}\mathtt{T}(q) &= \Delta_0 \cup \{b(Qx)\psi\}\\ \mathtt{T}(q0) &= \Delta_0 \cup \{b(Qx)\psi, b(\psi')_{x:=t}\},\end{aligned}$$

where ψ' is a variant of ψ such that t is substitutable for x in ψ'. In the tableau $\mathtt{T}'$, we have

$$\begin{aligned}\mathtt{T}'(q) &= \mathsf{VARIANT}(\Delta_0, U) \cup \{b(Qw)(\mathsf{VARIANT}(\psi, U))_{x:=w}\}\\ \mathtt{T}'(q0) &= \mathsf{VARIANT}(\Delta_0, U)\\ &\quad \cup \{b(Qw)(\mathsf{VARIANT}(\psi, U))_{x:=w}, b\mathsf{VARIANT}((\psi')_{x:=t}, U)\},\end{aligned}$$

where w is the first variable in U that does not occur in $\mathsf{VARIANT}(\psi, U)$.

If x occurs free in ψ', then, by Theorem 4.6.58, we have

$$b\mathsf{VARIANT}((\psi')_{x:=t}, U) = b(\mathsf{VARIANT}(\psi', U))_{x:=t}.$$

The result then follows from Theorem 4.6.61.

If x does not occur free in ψ', then

$$\begin{aligned} \mathtt{T}(q0) &= \Delta_0 \cup \{b(Qx)\psi, b\psi'\} \\ \mathtt{T}'(q) &= \mathsf{VARIANT}(\Delta_0, U) \cup \{b(Qw)\mathsf{VARIANT}(\psi, U)\} \\ \mathtt{T}'(q0) &= \mathsf{VARIANT}(\Delta_0, U) \\ &\qquad \cup \{b(Qw)\mathsf{VARIANT}(\psi, U), b\mathsf{VARIANT}(\psi', U)\}, \end{aligned}$$

which shows that $\boldsymbol{\gamma}$-expansion is applied in $\mathtt{T}'$ at q. □

We use $\mathsf{VARIANT}(\mathtt{T}, U)$ to denote the tableau $\mathtt{T}'$ constructed in Theorem 5.3.47.

5.4 Cut Rule for First-Order Tableaux

Definition 5.4.1. Let $\mathcal{L}$ be a first-order language, V be an $\mathcal{L}$-suitable set of variables and Δ be a set of signed $(\mathcal{L}, V)$-formulas. A $(\Delta, \mathcal{L}, V)$-*tableau with cut* is an $(\mathcal{L}, V)$-tableau $\mathtt{T}$ that satisfies the following conditions:

- The root of $\mathtt{T}$ is labeled by Δ, i.e., $\mathtt{T}(\lambda) = \Delta$.
- If q is an interior node of $\mathtt{T}$, then one of the following cases occurs:
  (1) regular expansion is used at q, or
  (2) variantizing is used at q, or
  (3) $= \in \mathcal{L}$ and equality expansion is used at q, or
  (4) q has two immediate descendants and there is an $(\mathcal{L}, V)$-formula φ such that $\mathtt{T}(q0) = \mathtt{T}(q) \cup \{\mathbf{T}\varphi\}$ and $\mathtt{T}(q1) = \mathtt{T}(q) \cup \{\mathbf{F}\varphi\}$.

If the last case holds at an interior node q, then we say that *the cut rule* was used at q. □

Theorem 5.4.2. *Let $\mathcal{L}$ be a first-order language, V be an $\mathcal{L}$-suitable set of variables, Δ be a set of signed $(\mathcal{L}, V)$-formulas and $\mathtt{T}$ be a completed $(\Delta, \mathcal{L}, V)$-tableau with cut. The following statements hold:*

(1) *If $\mathtt{T}$ is conservative, then Δ is satisfiable if and only if $\mathtt{T}$ is not closed.*

(2) *If* T *is strongly completed, then* Δ *is satisfiable if and only if* T *is not closed.*

Proof. The argument is similar to that of Theorem 5.3.18 using analogues of Corollaries 5.3.15 and 5.3.16 for signed tableaux with cut. We leave the proof of these analogues for the reader. □

Corollary 5.4.3 (Soundness and Partial Strong Completeness for First-Order Tableaux with Cut). *A set of signed* $(\mathcal{L}, V)$*-formulas* Δ *is unsatisfiable if and only if there exists a strongly closed* $(\Delta, \mathcal{L}^c, V)$*-tableau with cut.*

Proof. If Δ is unsatisfiable, then, by Theorem 5.3.29, there is a strongly closed $(\Delta, \mathcal{L}^c, V)$-tableau, which is also a strongly closed $(\Delta, \mathcal{L}^c, V)$-tableau with cut.

Conversely, if there is a strongly closed $(\Delta, \mathcal{L}^c, V)$-tableau with cut, then, by Theorem 5.4.2, Δ is unsatisfiable. □

Theorem 5.4.4. *There is an effective, syntactic construction that begins with a a strongly closed* $(\Delta, \mathcal{L}, V)$*-tableau* T *with cut, where* $\mathcal{L}$ *is a first-order language and* V *is an* $\mathcal{L}$*-suitable set of variables and produces a strongly closed* $(\Delta_0, \mathcal{L}, V)$*-tableau with cut* $\mathtt{T}_0$*, where* Δ_0 *is a finite subset of* Δ*.*

Proof. The proof follows the same lines as the proof of Theorem 5.3.32, except for the case when cut is used at the root, which is handled as in Theorem 3.4.8. □

Following the pattern for propositional logic tableaux, we introduce a formal system whose proof trees are first-order tableaux with cut.

Definition 5.4.5. The formal system $\mathcal{F}^{\text{tabl,cut}}_{\mathcal{L},V}$ is the formal system obtained from $\mathcal{F}^{\text{tabl}}_{\mathcal{L},V}$ by adding the following "cut" rule:

$$\frac{\Delta \cup \{\mathbf{T}\varphi\}, \Delta \cup \{\mathbf{F}\varphi\}}{\Delta}$$

for every set of signed $(\mathcal{L}, V)$-formulas Δ and formula φ.

If we add the above cut rule to the formal system $\mathcal{F}^{\text{tabl,cons}}_{\mathcal{L},V}$, we obtain the formal system $\mathcal{F}^{\text{tabl,cons,cut}}_{\mathcal{L},V}$. ▯

General $\mathcal{F}_{\mathcal{L},V}^{\text{tabl,cut}}$-deduction trees ($\mathcal{F}_{\mathcal{L},V}^{\text{tabl,cons,cut}}$-deduction trees) for Δ and (conservative) $(\Delta, \mathcal{L}, V)$-tableaux with cut are the same. Similarly, $\mathcal{F}_{\mathcal{L},V}^{\text{tabl,cut}}$-proof trees ($\mathcal{F}_{\mathcal{L},V}^{\text{tabl,cons,cut}}$-proof trees) for Δ coincide with strongly closed (conservative) $(\Delta, \mathcal{L}, V)$-tableaux with cut. Thus, the set $\mathcal{PT}_{\mathcal{F}_{\mathcal{L},V}^{\text{tabl,cut}}}$ ($\mathcal{PT}_{\mathcal{F}_{\mathcal{L},V}^{\text{tabl,cons,cut}}}$) equals the set $\mathsf{SCTCUT}(\mathcal{L}, V)$ ($\mathsf{SCTCONSCUT}(\mathcal{L}, V)$) of all $(\mathcal{L}, V)$-tableaux $\mathtt{T}$ that are strongly closed (conservative) $(\mathtt{T}(\lambda), \mathcal{L}, V)$-tableaux with cut.

Theorem 5.4.6 (Soundness of $\mathcal{F}_{\mathcal{L},V}^{\text{tabl,cut}}$). *Let $\mathcal{L}$ be a first-order language and V be an $\mathcal{L}$-suitable set of variables. If Δ is a theorem of the formal system $\mathcal{F}_{\mathcal{L},V}^{\text{tabl,cut}}$, then Δ is an unsatisfiable set of $(\mathcal{L}, V)$-formulas.*

Proof. If Δ is a theorem of $\mathcal{F}_{\mathcal{L},V}^{\text{tabl,cut}}$, there is an $\mathcal{F}_{\mathcal{L},V}^{\text{tabl,cut}}$-proof tree $\mathtt{T}$ for Δ which is a strongly closed $(\Delta, \mathcal{L}, V)$-tableau with cut. By Theorem 5.4.2, Δ is unsatisfiable. $\square$

Theorem 5.4.7 (Partial Completeness of $\mathcal{F}_{\mathcal{L}^c,V}^{\textbf{tabl,cons,cut}}$). *Let $\mathcal{L}$ be a first-order language and V be a set of variables. If Δ is an unsatisfiable set of signed $(\mathcal{L}, V)$-formulas, then Δ is a theorem of $\mathcal{F}_{\mathcal{L}^c,V}^{\textit{tabl,cons,cut}}$.*

Proof. Since $\mathcal{F}_{\mathcal{L}^c,V}^{\text{tabl,cons,cut}}$ is obviously an extension of $\mathcal{F}_{\mathcal{L}^c,V}^{\text{tabl,cons}}$, the partial completeness of the latter (Theorem 5.3.38) implies the partial completeness of the former. $\square$

As in propositional logic, the cut rule is superfluous since whatever set of signed formulas can be shown unsatisfiable using a closed tableau with cut can also be shown to be unsatisfiable by a closed tableau without cut. Nevertheless, as we show in Supplements 28 and 38, proofs that use the cut rule can be much shorter than proofs without the cut rule. The same point is made for propositional logic by Supplements 40 and 47 of Chapter 3.

We intend to present a cut-elimination construction similar to the corresponding propositional result. We now prove some preliminary technical results needed for the cut-elimination construction.

Theorem 5.4.8. *Let $\mathtt{T}$ be a $(\Delta, \mathcal{L}, V)$-tableau, a be a constant symbol of $\mathcal{L}$ and t be an $(\mathcal{L}, V)$-term. Suppose that the constant symbol a is* *not an eigenconstant of* $\mathtt{T}$*, the term* t *contains no eigenconstant of* $\mathtt{T}$ *and no variable that occurs in* t *occurs bound anywhere in* $\mathtt{T}$*.*

Define the lot $\mathtt{T}'$ *by* $\mathrm{Dom}(\mathtt{T}') = \mathrm{Dom}(\mathtt{T})$ *and* $\mathtt{T}'(q) = \mathsf{s}_t^a(\mathtt{T}(q))$ *for all* $q \in \mathrm{Dom}(\mathtt{T})$*. Then,* $\mathtt{T}'$ *is an* $(\mathsf{s}_t^a(\Delta), \mathcal{L}, V)$*-tableau that uses at each node the same type of expansion as* $\mathtt{T}$ *does. Further, if* $\mathtt{T}$ *is (strongly) closed, then so is* $\mathtt{T}'$*. (We will denote* $\mathtt{T}'$ *by* $\mathsf{s}_t^a(\mathtt{T})$*.)*

Proof. Clearly, $\mathtt{T}'(\lambda) = \mathsf{s}_t^a(\Delta)$. We need to prove that the tableau $\mathtt{T}'$ is locally consistent at every interior node q. Five cases must be considered depending on the type of expansion used at q in $\mathtt{T}$.

Case 1: If propositional expansion is used on a formula $b\varphi$ at q in $\mathtt{T}$, then, by Exercise 153 of Chapter 4, propositional expansion is used on the formula $\mathsf{s}_t^a(b\varphi)$ at q in $\mathtt{T}'$.

Case 2: If regular expansion is used on a $\boldsymbol{\gamma}$-formula $b\varphi = b(Qx)\psi$ at q in $\mathtt{T}$, say

$$\begin{aligned} \mathtt{T}(q) &= \Delta \cup \{b(Qx)\psi\} \\ \mathtt{T}(q0) &= \Delta \cup \{b(Qx)\psi, b(\psi')_{x:=u}\}, \end{aligned}$$

for an $(\mathcal{L}, V)$-term u and a variant ψ' of ψ such that u is substitutable for x in ψ', then

$$\begin{aligned} \mathtt{T}'(q) &= \mathsf{s}_t^a(\Delta) \cup \{\mathsf{s}_t^a(b(Qx)\psi)\} \\ &= \mathsf{s}_t^a(\Delta) \cup \{b(Qx)\mathsf{s}_t^a(\psi)\} \\ \mathtt{T}'(q0) &= \mathsf{s}_t^a(\Delta) \cup \{\mathsf{s}_t^a(b(Qx)\psi), \mathsf{s}_t^a(b(\psi')_{x:=u})\} \\ &= \mathsf{s}_t^a(\Delta) \cup \{b(Qx)\mathsf{s}_t^a(\psi), b\mathsf{s}_t^a((\psi')_{x:=u})\} \\ &= \mathsf{s}_t^a(\Delta) \cup \{b(Qx)\mathsf{s}_t^a(\psi), b(\mathsf{s}_t^a(\psi'))_{x:=u'}\}, \end{aligned}$$

where $u' = \mathsf{s}_t^a(u)$. The last equality follows from Equality (4.1) on page 496 since x occurs bound in $\mathtt{T}$ and therefore does not occur in t. By Corollary 4.6.41, the formula $\mathsf{s}_t^a(\psi')$ is a variant of $\mathsf{s}_t^a(\psi)$ because no variable of t occurs bound in either ψ or ψ'. By Corollary 4.3.79, the term u' is substitutable for x in $\mathsf{s}_t^a(\psi')$ Thus, since u' is an $(\mathcal{L}, V)$-term, we have $\boldsymbol{\gamma}$-expansion of the formula $b(Qx)\mathsf{s}_t^a(\psi)$ at the node q of $\mathtt{T}'$.

Case 3: If nondegenerate regular expansion is used on a $\boldsymbol{\delta}$-formula $b\varphi = b(Qx)\psi$ at q in $\mathtt{T}$, say

$$\begin{aligned}\mathtt{T}(q) &= \Delta \cup \{b(Qx)\psi\}\\ \mathtt{T}(q0) &= \Delta \cup \{b(\psi)_{x:=c}\},\end{aligned}$$

where c does not occur in $\mathtt{T}(q)$, then

$$\begin{aligned}\mathtt{T}'(q) &= \mathsf{s}_t^a(\Delta) \cup \{\mathsf{s}_t^a(b(Qx)\psi)\},\\ &= \mathsf{s}_t^a(\Delta) \cup \{b(Qx)\mathsf{s}_t^a(\psi)\},\\ \mathtt{T}'(q0) &= \mathsf{s}_t^a(\Delta) \cup \{\mathsf{s}_t^a(b(\psi)_{x:=c})\},\\ &= \mathsf{s}_t^a(\Delta) \cup \{b\mathsf{s}_t^a((\psi)_{x:=c})\},\\ &= \mathsf{s}_t^a(\Delta) \cup \{b(\mathsf{s}_t^a(\psi))_{x:=c}\},\end{aligned}$$

by Equality (4.1) because $c = \mathsf{s}_t^a(c)$, due to the fact that a is different from any constant symbol used in a nondegenerate $\boldsymbol{\delta}$-expansion in $\mathtt{T}$. Note that the constant symbol c occurs neither in $\mathtt{T}(q)$ (because we have a $\boldsymbol{\delta}$-expansion at q in $\mathtt{T}$) nor in t (since t contains no constant symbol used in a nondegenerate $\boldsymbol{\delta}$-expansion in $\mathtt{T}$). Thus, we have a $\boldsymbol{\delta}$-expansion at q in $\mathtt{T}'$.

If degenerate regular expansion is used on a $\boldsymbol{\delta}$-formula $b\varphi = b(Qx)\psi$ at q in $\mathtt{T}$, say

$$\begin{aligned}\mathtt{T}(q) &= \Delta \cup \{b(Qx)\psi\}\\ \mathtt{T}(q0) &= \Delta \cup \{b\psi\},\end{aligned}$$

where $x \notin \mathtt{FV}(\psi)$, then

$$\begin{aligned}\mathtt{T}'(q) &= \mathsf{s}_t^a(\Delta) \cup \{b(Qx)\mathsf{s}_t^a(\psi),\\ \mathtt{T}'(q0) &= \mathsf{s}_t^a(\Delta) \cup \{b\mathsf{s}_t^a(\psi)\}.\end{aligned}$$

Since x occurs bound in $\mathtt{T}$, by hypothesis, $x \notin \mathtt{V}(t)$, so by Theorem 4.3.69, $x \notin \mathtt{FV}(\mathsf{s}_t^a(\psi))$ and therefore degenerate $\boldsymbol{\delta}$-expansion is used at q in $\mathtt{T}'$.

Case 4: Suppose that variantization is applied at $\mathtt{T}(q)$, that is, there is a finite-to-one function $f : \mathtt{T}(q0) \longrightarrow \mathtt{T}(q)$ such that each formula $f(b\varphi)$ is a variant of the formula $b\varphi$. Let $b\psi$ be a formula in $\mathtt{T}'(q0)$.

Choose the formula $b\varphi \in \mathtt{T}(q0)$ as the first formula in the standard order of signed formulas such that $\mathsf{s}_t^a(b\varphi) = b\psi$ and define $g(b\psi) = \mathsf{s}_t^a(f(b\varphi))$. Observe that no variable of t occurs bound in either $b\varphi$ or in $f(b\varphi)$ because we assume that no variable of t occurs bound in $\mathtt{T}$. Therefore, by Corollary 4.6.41, the formula $\mathsf{s}_t^a(f(b\varphi))$ is a variant of $\mathsf{s}_t^a(b\varphi)$, which means that $g(b\psi)$ is a variant of $b\psi$. By Theorem 1.2.17, s_t^a is a finite-to-one function and so is f, which implies that g is a finite-to-one function. Thus, variantization is applied at q in $\mathtt{T}'$.

Case 5: If there is equality expansion at q in $\mathtt{T}$, then, by Exercise 68, we have the same type of expansion at q in $\mathtt{T}'$.

It is clear that if $\mathtt{T}$ is (strongly) closed, then so is $\mathtt{T}'$. □

Theorem 5.4.9. *Let* $\mathtt{T}$ *be a* $(\Delta, \mathcal{L}, V)$*-tableau. Suppose that* q *is a node of* $\mathtt{T}$ *where we have a nondegenerate* $\boldsymbol{\delta}$*-expansion using a constant symbol* a *and this constant symbol is not used for nondegenerate* $\boldsymbol{\delta}$*-expansions in any other node of* $\mathtt{T}_{[q]}$*. Further, assume that* a' *is a constant symbol that does not occur in* $\mathtt{T}_{[q]}$*. Let* $\mathtt{T}^q_{a,a'}$ *be the tableau given by* $\mathrm{Dom}(\mathtt{T}^q_{a,a'}) = \mathrm{Dom}(\mathtt{T})$ *and*

$$\mathtt{T}^q_{a,a'}(r) = \begin{cases} \mathtt{T}(r) & \text{if } q \text{ is not a proper prefix of } r \\ \mathsf{s}^a_{a'}(\mathtt{T}(r)) & \text{otherwise.} \end{cases}$$

Then, $\mathtt{T}^q_{a,a'}$ *is a* $(\Delta, \mathcal{L}, V)$*-tableau that uses the same type of expansion at every node as* $\mathtt{T}$ *and the same formula is expanded at* q *in both tableaux. Further, if* $\mathtt{T}$ *is strongly closed, then so is* $\mathtt{T}^q_{a,a'}$*.*

Proof. We need to show the local consistency of $\mathtt{T}^q_{a,a'}$ at every interior node r. If q is not a prefix of r, the local consistency of $\mathtt{T}^q_{a,a'}$ at r follows immediately from the local consistency of $\mathtt{T}$ at the same node. We need to consider two more cases.

If $r = q$, then $\mathtt{T}(q) = \Delta \cup \{b(Qx)\varphi\}$, q has only one immediate descendent $q0$, and $\mathtt{T}(q0) = \Delta \cup \{b(\varphi)_{x:=a}\}$, since we have $\boldsymbol{\delta}$-expansion at q. The definition of $\boldsymbol{\delta}$-expansion implies that a occurs neither in Δ nor in φ. Then, $\mathtt{T}^q_{a,a'}(q) = \mathtt{T}(q)$ and

$$\begin{aligned} \mathtt{T}^q_{a,a'}(q0) &= \mathsf{s}^a_{a'}(\mathtt{T}(q0)) \\ &= \mathsf{s}^a_{a'}(\Delta \cup \{b(\varphi)_{x:=a})\} \\ &= \Delta \cup \mathsf{s}^a_{a'}(b(\varphi)_{x:=a}) \end{aligned}$$

$$
\begin{aligned}
&\quad \text{(because } a \text{ does not occur in } \Delta\text{)}\\
&= \Delta \cup b(\mathsf{s}^a_{a'}(\varphi))_{x:=\mathsf{s}^a_{a'}(a)}\\
&\quad \text{(by Equality (4.1) of Chapter 4)}\\
&= \Delta \cup b(\varphi)_{x:=a'}\\
&\quad \text{(because } a \text{ does not occur in } \varphi\text{).}
\end{aligned}
$$

Thus, we have $\boldsymbol{\delta}$-expansion at q in $\mathtt{T}^q_{a,a'}$ because a' does not occur in $\mathtt{T}(q) = \mathtt{T}^q_{a,a'}(q)$.

Finally, suppose that q is a proper prefix of r. Then, since neither a nor a' is used in a nondegenerate $\boldsymbol{\delta}$-expansion in $\mathtt{T}_{[q0]}$, by Theorem 5.4.8, $\mathsf{s}^a_{a'}(\mathtt{T}_{[q0]})$ is an $(\mathsf{s}^a_{a'}(\mathtt{T}(q0)), \mathcal{L}, V)$-tableau. This implies local consistency of $\mathtt{T}^q_{a,a'}$ at r, so $\mathtt{T}^q_{a,a'}$ is a $(\Delta, \mathcal{L})$-tableau.

The fact that the strong closure of $\mathtt{T}$ is preserved in $\mathtt{T}^q_{a,a'}$ is immediate. □

The previous theorem is used in the construction of a strongly closed $(\Delta, \mathcal{L}^c, V)$-tableau that avoids the members of a finite set of constant symbols in its $\boldsymbol{\delta}$-expansions.

Construction 5.4.10.
Input: A first-order language $\mathcal{L}$ that contains infinitely many constant symbols, a set of variables V, a finite set Δ of signed $(\mathcal{L}, V)$-formulas, a strongly closed $(\Delta, \mathcal{L}, V)$-tableau $\mathtt{T}$ and a finite set S of $\mathcal{L}$-constant symbols.
Output: A strongly closed $(\Delta, \mathcal{L}, V)$-tableau denoted by $\mathsf{AVOID}_{\mathcal{L},S}(\mathtt{T})$ such that:

- no constant symbol in S is used as an eigenconstant in $\mathsf{AVOID}_{\mathcal{L},S}(\mathtt{T})$;
- no constant symbol is used as an eigenconstant at two distinct nodes of $\mathsf{AVOID}_{\mathcal{L},S}(\mathtt{T})$;
- $\mathrm{Dom}(\mathsf{AVOID}_{\mathcal{L},S}(\mathtt{T})) = \mathrm{Dom}(\mathtt{T})$;
- the new tableau uses at each node the same type of expansion as $\mathtt{T}$ does;
- if $\mathtt{T}$ uses regular expansion with removal or retention at the root, then the same formula is expanded in the same manner at root of $\mathsf{AVOID}_{\mathcal{L},S}(\mathtt{T})$.

Method: If no eigenconstant of $\mathtt{T}$ is in S and no eigenconstant of $\mathtt{T}$ is used twice, then we define $\mathsf{AVOID}_{\mathcal{L},S}(\mathtt{T}) = \mathtt{T}$. Otherwise, let q be a node of minimal depth in $\mathtt{T}$ where a constant symbol a is used as an eigenconstant in a $\boldsymbol{\delta}$-expansion and either $a \in S$ or a is used elsewhere as an eigenconstant. Let a' be the first constant symbol in $\mathcal{L}$ that does not occur in either S or $\mathtt{T}$. Replace $\mathtt{T}$ by $\mathtt{T}^q_{a,a'}$, where $\mathtt{T}^q_{a,a'}$ was introduced in the statement of Theorem 5.4.9, and repeat the process.

Proof of Correctness: First note that the constant symbol a' exists since S is finite and only finitely many constant symbols occur in $\mathtt{T}$ because a strongly closed tableau is finite and every set of the form $\mathtt{T}(q)$ is finite.

The proof of correctness is by induction on n, the number of nodes where a constant symbol of S is used as an eigenconstant in a $\boldsymbol{\delta}$-expansion in $\mathtt{T}$ or the eigenconstant used at the node is used as an eigenconstant elsewhere. For $n = 0$, the correctness is obvious. Suppose the construction is correct for tableaux that contain n or fewer undesirable nodes and let $\mathtt{T}$ be a tableau with $n+1$ such nodes. By Theorem 5.4.9, the tableau $\mathtt{T}^q_{a,a'}$, obtained as above, is a strongly closed $(\Delta, \mathcal{L}, V)$-tableau which contains fewer than $n+1$ "bad" nodes. By the inductive hypothesis applied to this tableau, the construction is correct for $\mathtt{T}$. □

Construction 5.4.10 becomes effective, that is becomes an algorithm, if the language $\mathcal{L}$ is decidable.

As we did in the propositional case, we construct a function $\mathsf{cet}_{\mathcal{L},V}$ that eliminates a single use of the cut rule. Recall that when we use the term "tableau", we mean tableau without the cut rule.

Construction 5.4.11.
Input: A first-order language $\mathcal{L}$ that contains infinitely many constant symbols, two finite sets Δ_0, Δ_1 of signed $(\mathcal{L}, V)$-formulas, two strongly closed $(\mathcal{L}, V)$-tableaux $\mathtt{T}_0, \mathtt{T}_1$, and an $(\mathcal{L}, V)$-formula φ such that $\mathtt{T}_0$ is a $(\Delta_0 \cup \{\mathbf{T}\varphi\}, \mathcal{L}, V)$-tableau and $\mathtt{T}_1$ is a $(\Delta_1 \cup \{\mathbf{F}\varphi\}, \mathcal{L}, V)$-tableau.

Output: A strongly closed $(\Delta_0 \cup \Delta_1, \mathcal{L}, V)$-tableau denoted by

$$\mathsf{cet}_{\mathcal{L},V}(\mathtt{T}_0, \mathtt{T}_1, \Delta_0, \Delta_1, \varphi).$$

Method: Proceed according to which of the following cases holds:

`Case 1:` Either $\mathbf{T}\varphi \in \Delta_0$ or $\mathbf{F}\varphi \in \Delta_1$.
The subcases of this case are identical to the subcases of Case 1 of Construction 3.4.11.

`Case 2:` Either $\Delta_0 \cup \{\mathbf{T}\varphi\}$ or $\Delta_1 \cup \{\mathbf{F}\varphi\}$ is closed and neither $\mathbf{T}\varphi \in \Delta_0$ nor $\mathbf{F}\varphi \in \Delta_1$.
The treatment of the subcases of this case is identical to the one of Case 2 of Construction 3.4.11.

`Case 3:` Neither $\Delta_0 \cup \{\mathbf{T}\varphi\}$ nor $\Delta_1 \cup \{\mathbf{F}\varphi\}$ is closed and neither $\mathbf{T}\varphi \in \Delta_0$ nor $\mathbf{F}\varphi \in \Delta_1$. Then, both $\mathtt{T}_0$ and $\mathtt{T}_1$ have more than one node.

`Case 3.1:` Equality expansion by $\mathbf{T}\alpha$ is used at the root of at least one of $\mathtt{T}_0$ or $\mathtt{T}_1$. We have either $\mathtt{T}_0(0) = \Delta_0 \cup \{\mathbf{T}\varphi, \mathbf{T}\alpha\}$ or $\mathtt{T}_1(0) = \Delta_1 \cup \{\mathbf{F}\varphi, \mathbf{T}\alpha\}$. In the first subcase, define $\mathtt{V} = \mathsf{cet}_{\mathcal{L},V}((\mathtt{T}_0)_{[0]}, \mathtt{T}_1, \Delta_0 \cup \{\mathbf{T}\alpha\}, \Delta_1, \varphi)$ and return $(\mathtt{V}; \Delta_0 \cup \Delta_1)$. The other case is treated similarly.

`Case 3.2:` Variantization is used at the root of at least one of $\mathtt{T}_0$ or $\mathtt{T}_1$. If variantization is used at the root of $\mathtt{T}_0$, the root of this tree has one child and there is a finite-to-one function $f : \mathtt{T}_0(0) \longrightarrow \mathtt{T}_0(\lambda)$ such that every formula in $f^{-1}(b\psi)$ is a variant of $b\psi$, for every $b\psi \in \mathtt{T}(\lambda)$. Now we distinguish two subcases depending on whether or not $f^{-1}(\mathbf{T}\varphi)$ is empty. (If variantization occurs at the root of $\mathtt{T}_1$, the corresponding cases are similar to the cases outlined below.)

`Case 3.2.1:` $f^{-1}(\mathbf{T}\varphi) = \emptyset$. Then return $((\mathtt{T}_0)_{[0]}; \Delta_0 \cup \Delta_1)$.

`Case 3.2.2:` $f^{-1}(\mathbf{T}\varphi) \neq \emptyset$. The finiteness of Δ_0 and Δ_1 means that the set of variables that occur in either $\mathtt{T}_0$ or $\mathtt{T}_1$ is finite. Therefore the complement of this set of variables is infinite. Let U be the odd numbered variables of this complement. It is clear that both U and $\mathrm{VAR} - U$ are infinite sets. Let $\mathtt{T}'_0 = \mathsf{VARIANT}((\mathtt{T}_0)_{[0]}, U)$ and $\mathtt{T}'_1 = \mathsf{VARIANT}(\mathtt{T}_1, U)$. Compute

$$\mathtt{T}' = \mathsf{cet}_{\mathcal{L},V}(\mathtt{T}'_0, \mathtt{T}'_1, \mathsf{VARIANT}(f^{-1}(\Delta_0), U), \mathsf{VARIANT}(\Delta_1, U), \mathsf{VARIANT}(\varphi, U))$$

and return $(\mathtt{T}'; \Delta_0 \cup \Delta_1)$.

`Case 3.3:` Neither equality expansion nor variantization is used at the roots of $\mathtt{T}_0$ and $\mathtt{T}_1$. Then, there are Δ'_0, Δ'_1, $b_0\psi_0 \in \Delta_0 \cup \{\mathbf{T}\varphi\}$, $b_1\psi_1 \in \Delta_1 \cup \{\mathbf{F}\varphi\}$, and sequences of sets of $(\mathcal{L}, V)$-formulas $(K_0, \ldots, K_{n-1})$ and $(H_0, \ldots, H_{m-1})$ such that $\Delta_0 \cup \{\mathbf{T}\varphi\} = \Delta'_0 \cup \{b_0\psi_0\}$, $\Delta_1 \cup \{\mathbf{F}\varphi\} = \Delta'_1 \cup \{b_1\psi_1\}$, $\mathtt{T}_0(i) = \Delta'_0 \cup K_i$ for $0 \leq i \leq n-1$, and $\mathtt{T}_1(j) = \Delta'_1 \cup H_j$ for $0 \leq j \leq m-1$, where the roots of $\mathtt{T}_0, \mathtt{T}_1$ have n and m immediate descendants. Note that if we have propositional expansion at the root of $\mathtt{T}_0$, then $(K_0, \ldots, K_{n-1})$ is $\mathtt{d}_{\mathcal{L}}(b_0\psi_0)$. If $b_0\psi_0$ is a $\boldsymbol{\gamma}$- or $\boldsymbol{\delta}$-formula, then $n = 1$ and $K_0 = \{b_0(\psi'_0)_{x:=t}\}$ or $K_0 = \{b_0(\psi_0)_{x:=c}\}$, respectively, where t is an $(\mathcal{L}, V)$-term, ψ'_0 is variant of ψ_0 such that t is substitutable for x in ψ'_0, and c is a constant symbol which does not occur in $\mathtt{T}_0(\lambda)$. Similar comments apply to $\mathtt{T}_1$. (There could be several such choices of $b_0\psi_0, b_1\psi_1$, any of which would work. To be definite, we choose the first ones in the standard ordering of the signed formulas.)

`Case 3.3.1:` $b_0\psi_0 \in \Delta_0$ (so $b_0\psi_0 \neq \mathbf{T}\varphi$ because $\mathbf{T}\varphi \notin \Delta_0$) and $\boldsymbol{\delta}$-expansion is not used at the root of $\mathtt{T}_0$. Then, let $\mathtt{V}_i = \mathsf{cet}_{\mathcal{L},V}((\mathtt{T}_0)_{[i]}, \mathtt{T}_1, (\Delta_0' - \{\mathbf{T}\varphi\}) \cup K_i, \Delta_1, \varphi)$ for $0 \leq i \leq n-1$ and return $(\mathtt{V}_0, \ldots, \mathtt{V}_{n-1}; \Delta_0 \cup \Delta_1)$.

`Case 3.3.2:` $b_0\psi_0 \in \Delta_0$ (so $b_0\psi_0 \neq \mathbf{T}\varphi$ because $\mathbf{T}\varphi \notin \Delta_0$) and $\boldsymbol{\delta}$-expansion is used at the root of $\mathtt{T}_0$. Then, let S be the finite set of $\mathcal{L}$-constant symbols that occur in $\Delta_0 \cup \Delta_1$. Define $\mathtt{V} = \mathsf{AVOID}_{\mathcal{L},S}(\mathtt{T}_0)$. Since the same formula is expanded at the root of $\mathtt{V}$ as at the root of $\mathtt{T}_0$ and in the same manner, the root of $\mathtt{V}$ has one immediate descendant, $\mathtt{V}(0) = \Delta_0' \cup b_0(\theta_0)_{x:=c}$, where $b_0\psi_0$ is the $\boldsymbol{\delta}$-formula $b_0(Qx)\theta_0$ and $c \notin S$. (Note that in the degenerate case, c exists because $\mathcal{L}$ contains infinitely many constant symbols.) Let $\mathtt{V}' = \mathsf{cet}_{\mathcal{L},V}((\mathtt{V})_{[0]}, \mathtt{T}_1, (\Delta_0' - \{\mathbf{T}\varphi\}) \cup \{b_0(\theta_0)_{x:=c}\}, \Delta_1, \varphi)$. Finally, return $(\mathtt{V}'; \Delta_0 \cup \Delta_1)$.

`Case 3.3.3:` Neither Case 3.3.1 nor Case 3.3.2 holds, $b_1\psi_1 \in \Delta_1$ and $\boldsymbol{\delta}$-expansion is not used at the root of $\mathtt{T}_1$. The construction proceeds as in Case 3.3.1.

`Case 3.3.4:` Neither Case 3.3.1 nor Case 3.3.2 holds, $b_1\psi_1 \in \Delta_1$ and $\boldsymbol{\delta}$-expansion is used at the root of $\mathtt{T}_1$. The construction proceeds as in Case 3.3.2.

`Case 3.3.5:` Neither $b_0\psi_0 \in \Delta_0$ nor $b_1\psi_1 \in \Delta_1$, so $b_0\psi_0 = \mathbf{T}\varphi$ and $b_1\psi_1 = \mathbf{F}\varphi$.

`Case 3.3.5.1:` φ is either $(\neg\alpha)$ or $(\alpha C \beta)$, where C is a binary connective symbol. Then, proceed as in Case 3.2.3 of Construction 3.4.11.

`Case 3.3.5.2:` $\varphi = (\forall x)\alpha$. We have $\Delta_0' - \{\mathbf{T}(\forall x)\alpha\} = \Delta_0$. The roots of both $\mathtt{T}_0$ and $\mathtt{T}_1$ have one descendant each and we have $(\mathtt{T}_0)(0) = \Delta_0' \cup \{\mathbf{T}(\forall x)\alpha, \mathbf{T}(\alpha')_{x:=t}\} = \Delta_0 \cup \{\mathbf{T}(\forall x)\alpha, \mathbf{T}(\alpha')_{x:=t}\}$, where t is an $(\mathcal{L}, V)$-term and α' is a variant of α such that t is substitutable for x in α'. Let S_0 be the set of constant symbols that occur in $\mathtt{T}_0$ or in t. Define $\mathtt{W} = \mathsf{AVOID}_{\mathcal{L},S_0}(\mathtt{T}_1)$, which is a strongly closed $(\Delta_1 \cup \{\mathbf{F}(\forall x)\alpha\}, \mathcal{L}, V)$-tableau such that the formula $\mathbf{F}(\forall x)\alpha$ is expanded at the root. Thus, there is a set of formulas Δ_1'' such that $\mathtt{W}(\lambda) = \Delta_1'' \cup \{\mathbf{F}(\forall x)\alpha\}$ and $\mathtt{W}(0) = \Delta_1'' \cup \{\mathbf{F}(\alpha)_{x:=c}\}$, where c is an $\mathcal{L}$-constant symbol that does not appear in $\mathtt{W}(\lambda)$, $\mathtt{T}_0$ or t and is not used as an eigenconstant in $\mathtt{W}_{[0]}$. (Note that in the degenerate case, c exists because $\mathcal{L}$ contains infinitely many constant symbols.)

`Case 3.3.5.2.1:` $\mathbf{F}\forall x)\alpha \in \Delta_1''$. In this case, $\Delta_1'' = \Delta_1 \cup \{\mathbf{F}(\forall x)\alpha\}$. Then, define $\mathtt{U} = \mathsf{cet}_{\mathcal{L},V}((\mathtt{T}_0)_{[0]}, \mathtt{W}, \Delta_0 \cup \{\mathbf{T}(\alpha')_{x:=t}\}, \Delta_1, (\forall x)\alpha)$; also define $\mathtt{Y} = \mathsf{cet}_{\mathcal{L},V}(\mathtt{T}_0, (\mathtt{W})_{[0]}, \Delta_0, \Delta_1 \cup \{\mathbf{F}(\alpha)_{x:=c}\}, (\forall x)\alpha)$. Let V' be the set of odd numbered variables that do not occur in $\mathtt{U}, \mathtt{Y}$ or t and are different from x. Define the set of constant symbols S' that consists of c and the constant symbols that occur in t, and define the tableaux

$$\mathtt{U}' = \mathsf{VARIANT}(\mathtt{U}, V')$$
$$\mathtt{Y}' = \mathsf{AVOID}_{\mathcal{L},S'}(\mathsf{VARIANT}(\mathtt{Y}, V'))$$
$$\mathtt{Y}'' = \mathsf{s}^c_t(\mathtt{Y}').$$

Compute

$$\mathtt{Z} = \mathsf{cet}_{\mathcal{L},V}(\mathtt{U}', \mathtt{Y}'', \mathsf{VARIANT}(\Delta_0 \cup \Delta_1, V'), \mathsf{VARIANT}(\Delta_0 \cup \Delta_1, V'), (\mathsf{VARIANT}(\alpha, V'))_{x:=t}).$$

Return $(\mathtt{Z}; \Delta_0 \cup \Delta_1)$.

`Case 3.3.5.2.2:` $\mathbf{F}(\forall x)\alpha \notin \Delta_1''$. In this case, $\Delta_1'' = \Delta_1$ and so $\mathtt{W}(0) = \Delta_1 \cup \{\mathbf{F}(\alpha)_{x:=c}\}$. Then, define $\mathtt{U} = \mathsf{cet}_{\mathcal{L},V}((\mathtt{T}_0)_{[0]}, \mathtt{W}, \Delta_0 \cup \{\mathbf{T}(\alpha')_{x:=t}\}, \Delta_1, (\forall x)\alpha)$; also define $\mathtt{Y} = \mathtt{W}_{[0]}$. Let V' be the set of odd numbered variables that do not occur in $\mathtt{U}, \mathtt{Y}$ or t. Define now the tableaux

$$\mathtt{U}' = \mathsf{VARIANT}(\mathtt{U}, V')$$
$$\mathtt{Y}' = \mathsf{VARIANT}(\mathtt{Y}, V')$$
$$\mathtt{Y}'' = \mathsf{s}^c_t(\mathtt{Y}').$$

Compute

$$\mathtt{Z} = \mathsf{cet}_{\mathcal{L},V}(\mathtt{U}', \mathtt{Y}'', \mathsf{VARIANT}(\Delta_0 \cup \Delta_1, V'), \mathsf{VARIANT}(\Delta_1, V'), (\mathsf{VARIANT}(\alpha, V'))_{x:=t}).$$

Return $(\mathtt{Z}; \Delta_0 \cup \Delta_1)$.

`Case 3.3.5.3:` $\varphi = (\exists x)\alpha$. We have $\Delta_1' - \{\mathbf{F}(\exists x)\alpha\} = \Delta_1$. The roots of both $\mathtt{T}_0$ and $\mathtt{T}_1$ have one descendant each and we have $(\mathtt{T}_1)(0) = \Delta_1' \cup \{\mathbf{F}(\exists x)\alpha, \mathbf{F}(\alpha')_{x:=t}\} = \Delta_1 \cup \{\mathbf{F}(\exists x)\alpha, \mathbf{F}(\alpha')_{x:=t}\}$, where t is an $(\mathcal{L}, V)$-term and α' is a variant of α such that t is substitutable for x in α'. Let S_1 be the set of constant symbols that occur in $\mathtt{T}_1$ or in t. Define $\mathtt{W} = \mathsf{AVOID}_{\mathcal{L},S_1}(\mathtt{T}_0)$, which is a strongly closed $(\Delta_0 \cup \{\mathbf{T}(\exists x)\alpha\}, \mathcal{L}, V)$-tableau such that the formula $\mathbf{T}(\exists x)\alpha$ is expanded at the root. Thus, there is a set of formulas Δ_0'' such that $\mathtt{W}(\lambda) = \Delta_0'' \cup \{\mathbf{T}(\exists x)\alpha\}$ and $\mathtt{W}(0) = \Delta_0'' \cup \{\mathbf{T}(\alpha)_{x:=c}\}$, where c is an $\mathcal{L}$-constant symbol that does not appear in $\mathtt{W}(\lambda)$, $\mathtt{T}_1$ or t and is not used as an eigenconstant in $\mathtt{W}_{[0]}$. (Note that in the degenerate case, c exists because $\mathcal{L}$ contains infinitely many constant symbols.)

`Case 3.3.5.3.1:` $\mathbf{T}(\exists x)\alpha \in \Delta_0''$. In this case, $\Delta_0'' = \Delta_0 \cup \{\mathbf{T}(\exists x)\alpha\}$. Then, define $\mathtt{U} = \mathsf{cet}_{\mathcal{L},V}(\mathtt{W}, (\mathtt{T}_1)_{[0]}, \Delta_0, \Delta_1 \cup \{\mathbf{F}(\alpha')_{x:=t}\}, (\exists x)\alpha)$; also define $\mathtt{Y} = \mathsf{cet}_{\mathcal{L},V}((\mathtt{W})_{[0]}, \mathtt{T}_1, \Delta_0 \cup \{\mathbf{T}(\alpha)_{x:=c}\}, \Delta_1, (\exists x)\alpha)$. Let V' be the set of odd numbered variables that do not occur in $\mathtt{Y}, \mathtt{U}$ or t and are different from x. Define the set of constant symbols S' that consists of c and the constant symbols that occur in t, and define the tableaux

$$\mathtt{U}' = \mathsf{VARIANT}(\mathtt{U}, V')$$

$$\mathtt{Y}' = \mathsf{AVOID}_{\mathcal{L},S'}(\mathsf{VARIANT}(\mathtt{Y}, V'))$$

$$\mathtt{Y}'' = \mathsf{s}_t^c(\mathtt{Y}').$$

```
                          Compute

                          $\mathtt{Z} = \mathsf{cet}_{\mathcal{L},V}(\mathtt{Y}'', \mathtt{U}',$
                              $\mathsf{VARIANT}(\Delta_0 \cup \Delta_1, V'),$
                              $\mathsf{VARIANT}(\Delta_0 \cup \Delta_1, V'),$
                              $(\mathsf{VARIANT}(\alpha, V'))_{x:=t}).$

                          Return $(\mathtt{Z}; \Delta_0 \cup \Delta_1)$.
              Case 3.3.5.3.2: $\mathbf{T}(\exists x)\alpha \notin \Delta_0''$. In this case, $\Delta_0'' = \Delta_0$
                          and so $\mathtt{W}(0) = \Delta_0 \cup \{\mathbf{T}(\alpha)_{x:=c}\}$. Then,
                          define $\mathtt{U} = \mathsf{cet}_{\mathcal{L},V}(\mathtt{W}, (\mathtt{T}_1)_{[0]}, \Delta_0, \Delta_1 \cup$
                          $\{\mathbf{F}(\alpha')_{x:=t}\}, (\exists x)\alpha)$; also define $\mathtt{Y} =$
                          $\mathtt{W}_{[0]}$ Let $V'$ be the set of odd numbered
                          variables that do not occur in $\mathtt{Y}, \mathtt{U}$ or
                          $t$ and are distinct from $x$. Define now
                          the tableaux

                              $\mathtt{U}' = \mathsf{VARIANT}(\mathtt{U}, V')$
                              $\mathtt{Y}' = \mathsf{VARIANT}(\mathtt{Y}, V')$
                              $\mathtt{Y}'' = \mathsf{s}_t^c(\mathtt{Y}').$

                          Compute

                          $\mathtt{Z} = \mathsf{cet}_{\mathcal{L},V}(\mathtt{Y}'', \mathtt{U}',$
                              $\mathsf{VARIANT}(\Delta_0, V'),$
                              $\mathsf{VARIANT}(\Delta_0 \cup \Delta_1, V'),$
                              $(\mathsf{VARIANT}(\alpha, V'))_{x:=t}).$

                          Return $(\mathtt{Z}; \Delta_0 \cup \Delta_1)$.
```

Proof of Correctness: Fix the first-order language $\mathcal{L}$ assumed to contain an infinite number of constant symbols and an $\mathcal{L}$-suitable set of variables V. We show by course-of-values induction on the norm of φ that if the algorithm is applied to a finite strongly closed $(\Delta_0 \cup \{\mathbf{T}\varphi\}, \mathcal{L}, V)$-tableau $\mathtt{T}_0$ and a finite strongly closed $(\Delta_1 \cup \{\mathbf{F}\varphi\}, \mathcal{L}, V)$-tableau $\mathtt{T}_1$, then it halts and produces the desired finite strongly closed $(\Delta_0 \cup \Delta_1, \mathcal{L}, V)$-tableau.

Suppose that the result is true for formulas with smaller norm than φ. We now show the result for φ by course-of-values induction on $n = |\mathtt{T}_0| + |\mathtt{T}_1|$. The "inner" inductive hypothesis means that for all $\Delta'_0, \Delta'_1, \mathtt{T}'_0, \mathtt{T}'_1$, if $\mathtt{T}'_0$ is a finite, strongly closed $(\Delta'_0 \cup \{\mathbf{T}\varphi\}, \mathcal{L}, V)$-tableau, $\mathtt{T}'_1$ is a finite, strongly closed $(\Delta'_1 \cup \{\mathbf{F}\varphi\}, \mathcal{L}, V)$-tableau, and $|\mathtt{T}'_0| + |\mathtt{T}'_1| < n$, then the algorithm halts and produces a finite, strongly closed $(\Delta'_0 \cup \Delta'_1, \mathcal{L}, V)$-tableau.

If Case 1 or 2 occurs, the correctness of the construction is shown exactly as in the corresponding case for Construction 3.4.11. For the first subcase of Case 3.1, $|(\mathtt{T}_0)_{[0]}| + |\mathtt{T}_1| < |\mathtt{T}_0| + |\mathtt{T}_1|$. By inductive hypothesis, since $(\mathtt{T}_0)_{[0]}$ is a strongly closed $(\Delta_0 \cup \{\mathbf{T}\varphi, \mathbf{T}\alpha\}, \mathcal{L}, V)$-tableau and $\mathtt{T}_1$ is strongly closed $(\Delta_1 \cup \{\mathbf{F}\varphi\}, \mathcal{L}, V)$-tableau, $\mathtt{V}$ is a strongly closed $(\Delta_0 \cup \Delta_1 \cup \{\mathbf{T}\alpha\}, \mathcal{L}, V)$-tableau, so the tableau returned is a strongly closed $(\Delta_0 \cup \Delta_1, \mathcal{L}, V)$ tableau with equality expansion used at the root. The second subcase of Case 3.1 can be treated similarly.

The correctness of the construction in Case 3.2.1 is obvious. In Case 3.2.2, by Theorem 5.3.47, $\mathtt{T}'_0$ is a $(\mathsf{VARIANT}(\mathtt{T}_0(0), U), \mathcal{L}, V)$-tableau and $\mathtt{T}'_1$ is a $(\mathsf{VARIANT}(\Delta_1, U) \cup \{\mathbf{F}\mathsf{VARIANT}(\varphi, U)\}, \mathcal{L}, V)$-tableau. Observe that by the definition of variantization, all formulas of $f^{-1}(\mathbf{T}\varphi)$ are variants of $\mathbf{T}\varphi$ and, therefore, by Theorem 4.6.60,

$$\mathsf{VARIANT}(f^{-1}(\mathbf{T}\varphi), U) = \{\mathbf{T}\mathsf{VARIANT}(\varphi, U)\}.$$

Thus, $\mathtt{T}'_0$ is a $(\mathsf{VARIANT}(f^{-1}(\Delta_0), U) \cup \{\mathbf{T}\mathsf{VARIANT}(\varphi, U)\}, \mathcal{L}, V)$-tableau. $\mathtt{T}'$ is a $(\mathsf{VARIANT}(f^{-1}(\Delta_0), U) \cup \mathsf{VARIANT}(\Delta_1, U), \mathcal{L}, V)$-tableau by the inductive hypothesis, because $|(\mathtt{T}_0)_{[0]}| + |\mathtt{T}_1| < |\mathtt{T}_0| + |\mathtt{T}_1|$. By Theorem 4.6.64, the tableau returned is a $(\Delta_0 \cup \Delta_1, \mathcal{L}, V)$-tableau where variantization was applied at the root.

The argument for Case 3.3.1 is identical to the one used for Case 3.2.1 for Construction 3.4.11 from propositional logic. The same is true for Case 3.3.3 relative to Case 3.2.2 of the same construction.

Observe that in Case 3.3.2, we have the equality $\Delta_0 = (\Delta'_0 - \{\mathbf{T}\varphi\}) \cup \{b_0\psi_0\}$. Since $|(\mathtt{V})_{[0]}| + |\mathtt{T}_1| < |\mathtt{T}_0| + |\mathtt{T}_1|$, by inductive hypothesis, $\mathtt{V}'$ is a $((\Delta'_0 - \{\mathbf{T}\varphi\}) \cup \{b_0(\theta_0)_{x:=c}\} \cup \Delta_1, \mathcal{L}, V)$-tableau. By the equality mentioned above, since c does not occur in $\Delta_0 \cup \Delta_1$, it follows that the tableau returned by this case, $(\mathtt{V}'; \Delta_0 \cup \Delta_1)$, is constructed by applying $\boldsymbol{\delta}$-expansion at the root.

Case 3.3.4 is similar to Case 3.3.2.

Now we give the proof of correctness for the subcases of Case 3.3.5. In Case 3.3.5.1, the argument is identical to the one used in Case 3.2.3 of Construction 3.4.11.

In Case 3.3.5.2, recall that $\mathtt{W}$ is a strongly closed $(\Delta_1 \cup \{\mathbf{F}(\forall x)\alpha\}, \mathcal{L}, V)$-tableau such that the formula $\mathbf{F}(\forall x)\alpha$ is expanded at the root and we can write $\mathtt{W}(\lambda) = \Delta_1'' \cup \{\mathbf{F}(\forall x)\alpha\}$ and $\mathtt{W}(0) = \Delta_1'' \cup \{\mathbf{F}(\alpha)_{x:=c}\}$, where c is an $\mathcal{L}$-constant symbol that does not appear in $\mathtt{W}(\lambda)$, $\mathtt{T}_0$ or t.

In Case 3.3.5.2.1, we have $|(\mathtt{T}_0)_{[0]}| + |\mathtt{W}| < |\mathtt{T}_0| + |\mathtt{T}_1|$, so, by inductive hypothesis, $\mathtt{U}$ is a strongly closed $(\Delta_0 \cup \{\mathbf{T}(\alpha')_{x:=t}\} \cup \Delta_1, \mathcal{L}, V)$-tableau. Similarly, $\mathtt{Y}$ is a strongly closed $(\Delta_0 \cup \Delta_1 \cup \{\mathbf{F}(\alpha)_{x:=c}\}, \mathcal{L}, V)$-tableau.

The newly generated tableau $\mathtt{U}'$ is a strongly closed $(\mathsf{VARIANT}(\Delta_0 \cup \Delta_1, V') \cup \{\mathsf{VARIANT}(\mathbf{T}(\alpha')_{x:=t}, V')\}, \mathcal{L}, V)$-tableau by Theorem 5.3.47, and therefore, a $(\mathsf{VARIANT}(, \Delta_0 \cup \Delta_1)V' \cup \{\mathbf{T}(\mathsf{VARIANT}(\alpha', V'))_{x:=t}\}, \mathcal{L}, V)$-tableau, by Theorem 4.6.58. Finally, applying Theorem 4.6.60, $\mathtt{U}'$ is a strongly closed $(\mathsf{VARIANT}(\Delta_0 \cup \Delta_1, V') \cup \{\mathbf{T}(\mathsf{VARIANT}(\alpha, V'))_{x:=t}\}, \mathcal{L}, V)$-tableau.

$\mathtt{Y}'$ is a $(\mathsf{VARIANT}(\Delta_0 \cup \Delta_1, V') \cup \{\mathsf{VARIANT}(\mathbf{F}(\alpha)_{x:=c}, V')\}, \mathcal{L}, V)$-tableau which is strongly closed and does not use either c or any constant symbol of t as an eigenconstant, by Theorem 5.3.47 and the correctness of Construction 5.4.10. Further, $\mathtt{Y}'$ is a

$$(\mathsf{VARIANT}(\Delta_0 \cup \Delta_1, V') \cup \{\mathbf{F}(\mathsf{VARIANT}(\alpha, V'))_{x:=c}\}, \mathcal{L}, V)\text{-tableau}$$

by Theorem 4.6.58. $\mathtt{Y}''$ is an

$$(\mathsf{s}_t^c(\mathsf{VARIANT}(\Delta_0 \cup \Delta_1, V')) \cup \mathsf{s}_t^c(\{\mathbf{F}(\mathsf{VARIANT}(\alpha, V'))_{x:=c}\}), \mathcal{L}, V)$$
$$\text{-tableau}$$

by Theorem 5.4.8 and is strongly closed. Since the constant symbol c does not occur in $\mathsf{VARIANT}(\alpha, V')$, we have

$$\mathsf{s}_t^c(\{\mathbf{F}(\mathsf{VARIANT}(\alpha, V'))_{x:=c}\}) = \mathbf{F}(\mathsf{VARIANT}(\alpha, V'))_{x:=t},$$

so $\mathtt{Y}''$ is an $(\mathsf{s}_t^c(\mathsf{VARIANT}(\Delta_0 \cup \Delta_1, V')) \cup \{\mathbf{F}(\mathsf{VARIANT}(\alpha, V'))_{x:=t}\}, \mathcal{L}, V)$-tableau by Exercise 25 of Chapter 4. Observe that the constant symbol c does not occur in Δ_0 because of the definition of $\mathtt{W}$ and it does not in Δ_1 because it is used as an eigenconstant at the root of $\mathtt{W}$. Therefore, $\mathtt{Y}''$ is actually a $(\mathsf{VARIANT}(\Delta_0 \cup \Delta_1, V') \cup \{\mathbf{F}(\mathsf{VARIANT}(\alpha, V'))_{x:=t}\}, \mathcal{L}, V)$-tableau. Since the norm of a formula is invariant under taking variants and applying substitutions, and $\| \alpha \| < \| \varphi \|$, by the inductive hypothesis, $\mathtt{Z}$ is a strongly closed $(\mathsf{VARIANT}(\Delta_0 \cup \Delta_1, V'), \mathcal{L}, V)$-tableau. By Theorem 4.6.64, the tableau returned is a $(\Delta_0 \cup \Delta_1, \mathcal{L}, V)$-tableau where variantization was applied at the root.

To prove the correctness in Case 3.3.5.2.2, we start by observing that $\mathtt{U}$ is a strongly closed $(\Delta_0 \cup \{\mathbf{T}(\alpha')_{x:=t}\} \cup \Delta_1, \mathcal{L}, V)$-tableau, so $\mathtt{U}'$ is a strongly closed $(\mathsf{VARIANT}(\Delta_0 \cup \Delta_1, V') \cup \{\mathsf{VARIANT}(\mathbf{T}(\alpha')_{x:=t}, V')\}, \mathcal{L}, V)$-tableau. So, as in the previous case, $\mathtt{U}'$ is a $(\mathsf{VARIANT}(\Delta_0 \cup \Delta_1, V') \cup \{\mathbf{T}(\mathsf{VARIANT}(\alpha, V'))_{x:=t}\}, \mathcal{L}, V)$-tableau.

The definition of $\mathtt{Y}$ implies that it is a $(\Delta_1 \cup \{\mathbf{F}(\alpha)_{x:=c}\}, \mathcal{L}, V)$-tableau which is strongly closed. It is easy to see that $\mathtt{Y}'$ is a strongly closed $(\mathsf{VARIANT}(\Delta_1, V') \cup \{\mathsf{VARIANT}(\mathbf{F}(\alpha)_{x:=c}, V')\}, \mathcal{L}, V)$-tableau and does not use either c or any constant symbol of t as an eigenconstant, by Theorem 5.3.47 and the correctness of Construction 5.4.10. Further, $\mathtt{Y}'$ is a $(\mathsf{VARIANT}(\Delta_1, V') \cup \{\mathbf{F}(\mathsf{VARIANT}(\alpha, V'))_{x:=c}\}, \mathcal{L}, V)$-tableau by Theorem 4.6.58.

By an argument similar to the one used in the previous case, $\mathtt{Y}''$ is a $(\mathsf{VARIANT}(\Delta_1, V') \cup \{\mathbf{F}(\mathsf{VARIANT}(\alpha, V'))_{x:=t}\}, \mathcal{L}, V)$-tableau.

By the inductive hypothesis, since $\| \alpha \| < \| \varphi \|$, $\mathtt{Z}$ is a strongly closed $(\mathsf{VARIANT}(\Delta_0 \cup \Delta_1, V'), \mathcal{L}, V)$-tableau. By Theorem 4.6.64, the tableau returned is a $(\Delta_0 \cup \Delta_1, \mathcal{L}, V)$-tableau where variantization was applied at the root.

We omit the proof of correctness for Cases 3.3.5.3.1 and 3.3.5.3.2, which are similar to the Cases 3.3.5.2.1 and 3.3.5.2.2, where the roles of the left and right subtableaux are reversed. $\square$

Construction 5.4.11 becomes effective, that is becomes an algorithm, if the language $\mathcal{L}$ is decidable.

This construction eliminates a single application of the cut rule "at the root" and serves as the recursive core of the next construction which allows us to build syntactically a cut-free strongly closed tableau starting from a strongly closed $(\Delta, \mathcal{L}, V)$-tableau with cut, where $\mathcal{L}$ is a first-order language that contains infinitely many constant symbols.

Construction 5.4.12.
Input: A first-order language $\mathcal{L}$ with infinitely many constant symbols, a finite set Δ of signed $(\mathcal{L}, V)$-formulas, and a strongly closed $(\Delta, \mathcal{L}, V)$-tableau with cut $\mathtt{T}$.
Output: A strongly closed $(\Delta, \mathcal{L}, V)$-tableau (without cut) $\mathsf{CET}_{\mathcal{L},V}(\mathtt{T})$.
Method: If $\mathtt{T}$ is a one-node strongly closed $(\Delta, \mathcal{L}, V)$-tableau with cut, then $\mathsf{CET}_{\mathcal{L},V}(\mathtt{T}) = \mathtt{T}$.
It $\mathtt{T}$ has more than one node, the root of $\mathtt{T}$ has n immediate descendants and the cut rule was not used at the root of $\mathtt{T}$, let

$$\mathsf{CET}_{\mathcal{L},V}(\mathtt{T}) = (\mathsf{CET}_{\mathcal{L},V}(\mathtt{T}_{[0]}), \ldots, \mathsf{CET}_{\mathcal{L},V}(\mathtt{T}_{[n-1]}); \Delta).$$

If neither of the two previous cases hold, then the cut rule was applied at the root of $\mathtt{T}$ and there is a formula φ such that $\mathtt{T}(0) = \Delta \cup \{\mathbf{T}\varphi\}$, and $\mathtt{T}(1) = \Delta \cup \{\mathbf{F}\varphi\}$. There could be several possible choices for φ. We select the first formula φ in the standard order for which the decomposition can be made. Now, we can define

$$\mathsf{CET}_{\mathcal{L},V}(\mathtt{T}) = \mathsf{cet}_{\mathcal{L},V}(\mathsf{CET}_{\mathcal{L},V}(\mathtt{T}_{[0]}), \mathsf{CET}_{\mathcal{L},V}(\mathtt{T}_{[1]}), \Delta, \Delta, \varphi).$$

Proof of Correctness: As in the propositional case, one can show by induction on $|\mathtt{T}|$ that the algorithm halts with proper output on $\mathtt{T}$. We leave this straightforward argument to the reader. □

We extend Construction 5.4.12 to $(\Delta, \mathcal{L}, V)$-tableaux with cut, where Δ is an arbitrary set of signed formulas.

Construction 5.4.13.
Input: A first-order language $\mathcal{L}$ with infinitely many constant symbols, a set Δ of signed $(\mathcal{L}, V)$-formulas, and a strongly closed $(\Delta, \mathcal{L}, V)$-tableau with cut $\mathtt{T}$.
Output: A strongly closed $(\Delta, \mathcal{L}, V)$-tableau (without cut) $\mathsf{CET}_{\mathcal{L},V}(\mathtt{T})$.
Method: Apply the method contained in the proof of Theorem 5.4.4 to obtain effectively a strongly closed $(\Delta_0, \mathcal{L}, V)$-tableau with cut $\mathtt{T}_0$, where Δ_0 is a finite subset of Δ. Then, apply Construction 5.4.12 to $\mathtt{T}_0$ to obtain a strongly closed cut-free $(\Delta_0, \mathcal{L}, V)$-tableau $\mathtt{T}'_0$. Finally, $\mathsf{CET}_{\mathcal{L},V}(\mathtt{T}) = (\mathtt{T}'_0; \Delta)$.

Proof of Correctness: The correctness follows immediately by observing that $(\mathtt{T}_0'; \Delta)$ is a strongly closed $(\Delta, \mathcal{L}, V)$-tableau obtained by applying thinning at the root. □

Example 5.4.14. Let Δ be a set of signed $(\mathcal{L}, V)$-formulas, where $\mathcal{L}$ is a first-order language with infinitely many constant symbols. Suppose that $\mathtt{U}$ is a strongly closed $(\Delta \cup \{\mathbf{F}(\forall x)\varphi\}, \mathcal{L}, V)$-tableau and $\mathtt{V}$ is a strongly closed $(\Delta \cup \{\mathbf{T}\langle\varphi\rangle_{x:=t}\}, \mathcal{L}, V)$-tableau, where t is a $(\mathcal{L}, V)$-term. The unsatisfiability of $\Delta \cup \{\mathbf{F}(\forall x)\varphi\}$ and $\Delta \cup \{\mathbf{T}\langle\varphi\rangle_{x:=t}\}$ means that $\Delta \models \mathbf{T}(\forall x)\varphi$ and $\Delta \models \mathbf{F}\langle\varphi\rangle_{x:=t}$, which in turn implies that Δ is unsatisfiable, which insures the existence of a strongly closed $(\Delta, \mathcal{L}, V)$-tableau. We will show below that cut elimination provides us with a syntactic way of constructing such a tableau.

Define the strongly closed $(\Delta \cup \{\mathbf{T}(\forall x)\varphi, \mathbf{T}\langle\varphi\rangle_{x:=t}\}, \mathcal{L}, V)$-tableau $\mathtt{W} = \mathtt{V} \uplus \{\mathbf{T}(\forall x)\varphi\}$, which uses thinning at the root. Next, let $\mathtt{W}' = (\mathtt{W}; \Delta \cup \{\mathbf{T}(\forall x)\varphi\})$, which is a strongly closed $(\Delta \cup \{\mathbf{T}(\forall x)\varphi\}, \mathcal{L}, V)$-tableau using $\boldsymbol{\gamma}$-expansion at the root. We obtain the desired tableau as $\mathsf{CET}_{\mathcal{L},V}(\mathtt{W}'')$, where $\mathtt{W}'' = (\mathtt{W}', \mathtt{U}; \Delta)$ is obtained by applying the cut rule at the root. ▯

Recall that $\mathrm{FVSubst}(\mathsf{s}_t^x, \varphi)$ is another notation for $(\varphi)_{x:=t}$ when φ is a formula, x is a variable and t is term. If Δ is a set of signed formulas, denote by $(\Delta)_{x:=t}$ the set of signed formulas $\{b(\theta)_{x:=t} \mid b\theta \in \Delta\}$. Similarly, if Γ is a set of formulas, we define $(\Gamma)_{x:=t}$ as the set $\{(\theta)_{x:=t} \mid \theta \in \Gamma\}$.

Theorem 5.4.15. *Let* $\mathtt{T}$ *be a* $(\Delta, \mathcal{L}, V)$*-tableau with cut, where* $\mathcal{L}$ *is a first-order language,* V *be an* $\mathcal{L}$*-suitable set of variables,* x *be a variable and* c *be a constant symbol of* $\mathcal{L}$ *that is not an eigenconstant of* $\mathtt{T}$*. Define a tableau* $\mathtt{T}'$ *by* $\mathrm{Dom}(\mathtt{T}') = \mathrm{Dom}(\mathtt{T})$ *and* $\mathtt{T}'(q) = (\mathtt{T}(q))_{x:=c}$*, for* $q \in \mathrm{Dom}(\mathtt{T})$*. Then,* $\mathtt{T}'$ *is a* $(\Delta', \mathcal{L}, V)$*-tableau with cut, where* $\Delta' = (\Delta)_{x:=c}$*.*

Proof. The reader can easily verify that $\mathtt{T}'(q)$ is a set of signed $(\mathcal{L}, V)$-formulas and that $\mathtt{T}'(\lambda) = \Delta'$. It remains to show that for each interior node $q \in \mathrm{Dom}(\mathtt{T}) = \mathrm{Dom}(\mathtt{T}')$, the same type of expansion is used in both tableaux. If propositional expansion or the cut rule is used at q in $\mathtt{T}$, then it is clear that the same device is used at q in $\mathtt{T}'$.

Suppose $\mathtt{T}(q0) = \mathtt{T}(q) \cup \{\mathbf{T}\alpha\}$, for some formula $\alpha \in \mathrm{INST}_{\mathcal{L},V}(\mathrm{Eq}_{=,\mathcal{L}})$. This means that there is an $\mathcal{L}$-equality axiom $(\forall y_0) \cdots (\forall y_{n-1})\beta$, where β is a quantifier-free formula, and there are $(\mathcal{L}, V)$-terms $t_0, \ldots, t_{n-1}$ such that $\alpha = (\beta)_{y_0,\ldots,y_{n-1}:=t_0,\ldots,t_{n-1}}$. It follows that

$$\mathtt{T}'(q0) = \mathtt{T}'(q) \cup \{\mathbf{T}(\alpha)_{x:=c}\} = \mathtt{T}'(q) \cup \{\mathbf{T}((\beta)_{y_0,\ldots,y_{n-1}:=t_0,\ldots,t_{n-1}})_{x:=c}\}.$$

Since α was obtained by applying the admissible substitution $\mathsf{s}^{y_0\cdots y_{n-1}}_{t_0\cdots t_{n-1}}$ to a quantifier-free formula, by Theorem 4.3.86, $(\alpha)_{x:=c} = (\mathsf{s}^x_c * \mathsf{s}^{y_0\cdots y_{n-1}}_{t_0\cdots t_{n-1}})(\alpha)$. We need now to consider two cases depending on whether $x \in \{y_0, \ldots, y_{n-1}\}$. Suppose x belongs to this set. Then, by Theorem 1.2.21, $\mathsf{s}^x_c * \mathsf{s}^{y_0\cdots y_{n-1}}_{t_0\cdots t_{n-1}} = \mathsf{s}^{y_0\cdots y_{n-1}}_{t'_0\cdots t'_{n-1}}$, where $t'_i = \mathsf{s}^x_c(t_i)$, for $0 \leq i \leq n-1$. Since the terms t'_i are $(\mathcal{L}, V)$-terms, equality expansion was also used at q in $\mathtt{T}'$. If $x \notin \{y_0, \ldots, y_{n-1}\}$, then $\mathsf{s}^x_c * \mathsf{s}^{y_0\cdots y_{n-1}}_{t_0\cdots t_{n-1}} = \mathsf{s}^{x\, y_0\cdots y_{n-1}}_{c\, t'_0\cdots t'_{n-1}}$. Since x does not occur in β, we have $\mathsf{s}^{x\, y_0\cdots y_{n-1}}_{c\, t'_0\cdots t'_{n-1}}(\beta) = \mathsf{s}^{y_0\cdots y_{n-1}}_{t'_0\cdots t'_{n-1}}(\beta)$, which shows that in this case, we also have equality expansion at q in $\mathtt{T}'$.

Suppose that we use variantization at q in $\mathtt{T}$. This means that there is a finite-to-one function $f : \mathtt{T}(q0) \longrightarrow \mathtt{T}(q)$ such that for every signed formula $b\varphi \in \mathtt{T}(q0)$, $f(b\varphi)$ is a variant of $b\varphi$. The function $f' : \mathtt{T}'(q0) \longrightarrow \mathtt{T}'(q)$ is defined by $f'(b'\psi') = (f(b'\varphi))_{x:=c}$, where φ is the first formula in the standard order such that $\psi' = (\varphi)_{x:=c}$. Taking into account the observation which precedes the theorem, we have

$$(f')^{-1}(b'\theta') \subseteq \mathsf{s}^x_c(f^{-1}((\mathsf{s}^x_c)^{-1}(b'\theta')))$$

for each $b'\theta' \in \mathtt{T}'(q)$. By Supplement 27 of Chapter 4, the set $(\mathsf{s}^x_c)^{-1}(b'\theta')$ is finite. Since f is finite-to-one, the above inclusion means that f' is finite-to-one. By Supplement 84 of Chapter 4, every formula $b'\psi'$ in $\mathtt{T}'(q0)$ is a variant of $f'(b'\psi')$, so variantization was used at q in $\mathtt{T}'$.

Assume now that nondegenerate $\boldsymbol{\delta}$-expansion is used at q in $\mathtt{T}$, that is, $\mathtt{T}(q) = \Delta_0 \cup \{b(Qy)\psi\}$ and $\mathtt{T}(q0) = \Delta_0 \cup \{b(\psi)_{y:=d}\}$. We can write $\mathtt{T}'(q) = (\Delta_0)_{x:=c} \cup \{b((Qy)\psi)_{x:=c}\}$ and $\mathtt{T}'(q0) = (\Delta_0)_{x:=c} \cup \{b((\psi)_{y:=d})_{x:=c}\}$. Observe that when $x \neq y$, by a double application of Theorem 4.3.86, we have $\mathtt{T}'(q0) = (\Delta_0)_{x:=c} \cup \{b((\psi)_{x:=c})_{y:=d}\}$.

We need to distinguish two cases depending on whether $x = y$. In the first case, let $x \neq y$. It follows that we have $\boldsymbol{\delta}$-expansion at q in $\mathtt{T}'$ because $d \neq c$ and d does not occur in $\mathtt{T}(q)$ implies that d does not occur in $\mathtt{T}'(q)$. In the second case, $x = y$, we have

$\mathtt{T}'(q) = (\Delta_0)_{x:=c} \cup \{b(Qy)\psi\}$ and $\mathtt{T}'(q0) = (\Delta_0)_{x:=c} \cup \{b(\psi)_{y:=d}\}$ which shows that $\boldsymbol{\delta}$-expansion is used at q in $\mathtt{T}'$.

Consider now the case when degenerate $\boldsymbol{\delta}$-expansion is used at q in $\mathtt{T}$, that is $\mathtt{T}(q) = \Delta_0 \cup \{b(Qy)\psi\}$ and $\mathtt{T}(q0) = \Delta_0 \cup \{b\psi\}$ and y does not occur free in ψ. Now we have, $\mathtt{T}'(q) = (\Delta_0)_{x:=c} \cup \{b((Qy)\psi)_{x:=c}\}$ and $\mathtt{T}'(q0) = (\Delta_0)_{x:=c} \cup \{b(\psi)_{x:=c}\}$. Again, we need to consider two cases depending on whether $x = y$. Suppose that $x \neq y$. This implies that $\mathtt{T}'(q) = (\Delta_0)_{x:=c} \cup \{(Qy)(\psi)_{x:=c}\}$. Since y does not occur free in $(\psi)_{x:=c}$, degenerate $\boldsymbol{\delta}$-expansion is used at q in $\mathtt{T}'$. If $x = y$, then x does not occur free in ψ and we have

$$\mathtt{T}'(q) = (\Delta_0)_{x:=c} \cup \{b(Qy)\psi\},$$
$$\mathtt{T}'(q0) = (\Delta_0)_{x:=c} \cup \{b\psi\},$$

which shows that we have degenerate $\boldsymbol{\delta}$-expansion at q in $\mathtt{T}'$.

Suppose now that $\boldsymbol{\gamma}$-expansion is used at q in $\mathtt{T}$, that is, $\mathtt{T}(q) = \Delta_0 \cup \{b(Qy)\psi\}$ and $\mathtt{T}(q0) = \Delta_0 \cup \{b(Qy)\psi, b(\psi')_{y:=t}\}$, where ψ' is a variant of ψ such that the $(\mathcal{L}, V)$-term t is substitutable for y in ψ'. Then, $\mathtt{T}'(q) = (\Delta_0)_{x:=c} \cup \{b((Qy)\psi)_{x:=c}\}$ and

$$\mathtt{T}'(q0) = (\Delta_0)_{x:=c} \cup \{b((Qy)\psi)_{x:=c}, b((\psi')_{y:=t})_{x:=c}\}.$$

Assume that $x = y$. Then,

$$\mathtt{T}'(q) = (\Delta_0)_{x:=c} \cup \{b(Qy)\psi\}$$
$$\mathtt{T}'(q0) = (\Delta_0)_{x:=c} \cup \{b(Qy)\psi, b(\psi')_{y:=\mathsf{s}_c^y(t)}\},$$

by Theorem 1.2.21 and Theorem 4.3.86. Thus, we have $\boldsymbol{\gamma}$-expansion at q in $\mathtt{T}'$ because substitutability of the term t for y in ψ' implies the substitutability of $\mathsf{s}_c^y(t)$ for y in ψ'.

Consider now the case when $x \neq y$. Now, we have:

$$\begin{aligned}\mathtt{T}'(q) &= (\Delta_0)_{x:=c} \cup \{b(Qy)(\psi)_{x:=c}\}\\ \mathtt{T}'(q0) &= (\Delta_0)_{x:=c} \cup \{b(Qy)(\psi)_{x:=c}, b(\psi')_{y,x:=\mathsf{s}_c^x(t),c}\\ &= (\Delta_0)_{x:=c} \cup \{b(Qy)(\psi)_{x:=c}, b((\psi')_{x:=c})_{y:=\mathsf{s}_c^x(t)},\end{aligned}$$

by a double application of Theorem 1.2.21 and Theorem 4.3.86, taking into account the substitutability of t for y and of c for x in ψ'. By Supplement 84 of Chapter 4, the formula $(\psi')_{x:=c}$ is a variant of $(\psi)_{x:=c}$. Further the term $(t)_{x:=c}$ is substitutable for y in the formula $(\psi')_{x:=c}$ by Supplement 36 of Chapter 4. This implies that $\boldsymbol{\gamma}$-expansion is used at q in $\mathtt{T}'$, which concludes the argument. $\square$

5.5 First-Order Sequents

First-order sequents are linked to tableaux of first-order logic in a manner similar to the linkage between propositional sequents and propositional tableaux. Therefore, this section parallels the corresponding section of Chapter 3 closely.

Definition 5.5.1. Let $\mathcal{L}$ be a first-order language and let V be a set of variables. An $(\mathcal{L}, V)$*-sequent* is a pair $\kappa = (\Gamma, \Gamma')$ of sets of $(\mathcal{L}, V)$-formulas. We will denote the $(\mathcal{L}, V)$-sequent (Γ, Γ') by $\Gamma \Rightarrow \Gamma'$. If $\mathcal{L}$ and V are clear from the context, we will use the term "sequent" in place of "$(\mathcal{L}, V)$"-sequent.

Γ is the *antecedent* and Γ' is the *succedent* of the sequent $\Gamma \Rightarrow \Gamma'$.

The sequent $\Gamma \Rightarrow \Gamma'$ is *finite* if both Γ and Γ' are finite.

We will denote the set of all $(\mathcal{L}, V)$-sequents by $\mathsf{SQT}_{\mathcal{L},V}$ and the set of all finite $(\mathcal{L}, V)$-sequents by $\mathsf{SQT}^{fin}{}_{\mathcal{L},V}$.

An $(\mathcal{L}, \emptyset)$-sequent will be referred to as a *sentential $\mathcal{L}$-sequent.*

$\Gamma \Rightarrow \Gamma'$ is an *$\mathcal{L}$-sequent* if it is an $(\mathcal{L}, V)$-sequent for some set of variables V. The set of all $\mathcal{L}$-sequents will be denoted by $\mathsf{SQT}_{\mathcal{L}}$. ▯

The notational conventions given after Definition 3.5.1 will also be used in the current section.

Definition 5.5.2. Let $\mathcal{A}$ be an $\mathcal{L}$-structure and let $\sigma \in \mathrm{ASSIGN}_{\mathcal{A}}$. The pair, $(\mathcal{A}, \sigma)$ *satisfies* an $\mathcal{L}$-sequent $\Gamma \Rightarrow \Gamma'$ if there exists a formula $\varphi \in \Gamma$ such that $(\mathcal{A}, \sigma) \not\models \varphi$ or there exists a formula $\psi \in \Gamma'$ such that $(\mathcal{A}, \sigma) \models \psi$. If $(\mathcal{A}, \sigma)$ does not satisfy the sequent, then it *falsifies* the sequent.

An $\mathcal{L}$-sequent is *valid* if it is satisfied by every pair $(\mathcal{A}, \sigma)$. ▯

Example 5.5.3. Any sequent $\Gamma \Rightarrow \Gamma'$ such that $\Gamma \cap \Gamma' \neq \emptyset$ is valid. ▯

Theorem 5.5.4. *Let $\mathcal{L}$ be a first-order language, Γ be a set of $\mathcal{L}$-formulas and let φ be an $\mathcal{L}$-formula. Then,*

(1) $\Gamma \models \varphi$ *if and only if the sequent* $\Gamma \Rightarrow \varphi$ *is valid. (In particular, φ is logically valid if and only if* $\Rightarrow \varphi$ *is valid.)*
(2) Γ *is satisfiable if and only if* $\Gamma \Rightarrow$ *is not valid.*

Proof. The argument for this simple theorem is left for the reader. □

The transformation between sequents and tableaux requires that we introduce the mutually inverse bijections

$$\begin{aligned}&\mathtt{sf}_{\mathcal{L},V} : \mathsf{SQT}_{\mathcal{L},V} \longrightarrow \mathcal{P}(\mathrm{SFORM}_{\mathcal{L}}(V)) \text{ and}\\ &\mathtt{sqt}_{\mathcal{L},V} : \mathcal{P}(\mathrm{SFORM}_{\mathcal{L}}(V)) \longrightarrow \mathsf{SQT}_{\mathcal{L},V},\end{aligned}$$

where $\mathcal{L}$ is a first-order language and V is a set of variables.

Definition 5.5.5. Let $\kappa = \Gamma \Rightarrow \Gamma'$ be an $(\mathcal{L}, V)$-sequent. The set of signed $(\mathcal{L}, V)$-formulas $\mathtt{sf}_{\mathcal{L},V}(\kappa)$ is defined to be $\{\mathbf{T}\varphi \mid \varphi \in \Gamma\} \cup \{\mathbf{F}\varphi \mid \varphi \in \Gamma'\}$.

Let Δ be a set of signed $(\mathcal{L}, V)$-formulas. The $(\mathcal{L}, V)$-sequent $\mathtt{sqt}_{\mathcal{L},V}(\Delta)$ is given by $\{\varphi \mid \mathbf{T}\varphi \in \Delta\} \Rightarrow \{\varphi \mid \mathbf{F}\varphi \in \Delta\}$. □

Let $\kappa_0 = \Gamma_0 \Rightarrow \Gamma'_0, \kappa_1 = \Gamma_1 \Rightarrow \Gamma'_1$ be two $(\mathcal{L}, V)$-sequents. We denote by $\kappa_0 \cup \kappa_1$ the $(\mathcal{L}, V)$-sequent $\Gamma_0 \cup \Gamma_1 \Rightarrow \Gamma'_0 \cup \Gamma'_1$. The reader can verify that

$$\begin{aligned}\mathtt{sqt}_{\mathcal{L},V}(\Delta_0 \cup \Delta_1) &= \mathtt{sqt}_{\mathcal{L},V}(\Delta_0) \cup \mathtt{sqt}_{\mathcal{L},V}(\Delta_1)\\ \mathtt{sf}_{\mathcal{L},V}(\kappa_0 \cup \kappa_1) &= \mathtt{sf}_{\mathcal{L},V}(\kappa_0) \cup \mathtt{sf}_{\mathcal{L},V}(\kappa_1)\end{aligned}$$

for $\Delta_0, \Delta_1 \subseteq \mathrm{SFORM}_{\mathcal{L}}(V)$ and $\kappa_0, \kappa_1 \in \mathsf{SQT}_{\mathcal{L},V}$.

Let $\Gamma \Rightarrow \Gamma'$ be a sequent, x be a variable and t be a term. The sequent $(\Gamma)_{x:=t} \Rightarrow (\Gamma')_{x:=t}$ will be denoted by $(\Gamma \Rightarrow \Gamma')_{x:=t}$. The reader can easily verify that

$$(\mathtt{sqt}_{\mathcal{L},V}(\Delta))_{x:=t} = \mathtt{sqt}_{\mathcal{L},V}((\Delta)_{x:=t}) \tag{5.3}$$

for any $(\mathcal{L}, V)$-term t, variable x and set of signed $(\mathcal{L}, V)$-formulas Δ.

Theorem 5.5.6. *Let $\mathcal{L}$ be a first-order language, V be a set of variables, κ be an $(\mathcal{L}, V)$-sequent, $\mathcal{A}$ be an $\mathcal{L}$-structure, and let $\sigma \in \mathrm{ASSIGN}_{\mathcal{A}}$. Then, the following two conditions are equivalent:*

(1) *$(\mathcal{A}, \sigma)$ satisfies κ.*
(2) *$(\mathcal{A}, \sigma)$ does not satisfy the set of signed formulas $\mathtt{sf}_{\mathcal{L},V}(\kappa)$.*

In addition, if $\kappa = \{\varphi_0, \ldots, \varphi_{n-1}\} \Rightarrow \{\psi_0, \ldots, \psi_{m-1}\}$, and $n, m \geq 1$, then the above conditions are equivalent to:

(3) *$(\mathcal{A}, \sigma) \models ((\varphi_0 \wedge \cdots \wedge \varphi_{n-1}) \to (\psi_0 \vee \cdots \vee \psi_{m-1}))$.*

Proof. The argument is straightforward and it is left to the reader. □

Corollary 5.5.7. *Let $\mathcal{L}$ be a first-order language, V be a set of variables, κ be an $(\mathcal{L}, V)$-sequent. Then the following two conditions are equivalent:*

(1) *The sequent κ is valid.*
(2) *The set of signed formulas $\mathtt{sf}_{\mathcal{L},V}(\kappa)$ is unsatisfiable.*

In addition, if $\kappa = \{\varphi_0, \dots, \varphi_{n-1}\} \Rightarrow \{\psi_0, \dots, \psi_{m-1}\}$, and $n, m \geq 1$, then the above conditions are equivalent to:

(3) *The formula $((\varphi_0 \wedge \cdots \wedge \varphi_{n-1}) \to (\psi_0 \vee \cdots \vee \psi_{m-1}))$ is logically valid.*

Proof. This follows immediately from Theorem 5.5.6. □

Corollary 5.5.7 shows that it is possible to determine the validity of sequents using tableaux. Actually, in this section we will introduce a formal system $\mathcal{F}^{\text{seq}}_{\mathcal{L},V}$ whose theorems are the valid $(\mathcal{L}, V)$-sequents. This formal system is the counterpart of the formal system $\mathcal{F}^{\text{tabl}}_{\mathcal{L},V}$ which we introduced previously for tableaux. A proof tree in $\mathcal{F}^{\text{seq}}_{\mathcal{L},V}$ for an $(\mathcal{L}, V)$-sequent κ will be another way of representing the construction of a strongly closed tableau that shows that the set $\mathtt{sf}_{\mathcal{L},V}(\kappa)$ is unsatisfiable. In fact, proof trees in $\mathcal{F}^{\text{seq}}_{\mathcal{L},V}$ will be translations under $\mathtt{sqt}_{\mathcal{L},V}$ of proof trees in $\mathcal{F}^{\text{tabl}}_{\mathcal{L},V}$.

Definition 5.5.8. Let $\mathcal{L}$ be a first-order language and V be an $\mathcal{L}$-suitable set of variables. The formal system $\mathcal{F}^{\text{seq,cons}}_{\mathcal{L},V} = (\mathsf{SQT}_{\mathcal{L},V}, A, I)$ is given by:

- The set of axioms A consists of all $(\mathcal{L}, V)$-sequents $\Gamma \Rightarrow \Gamma'$ such that $\Gamma \cap \Gamma' \neq \emptyset$.
- The set of rules of inference I consists of the rules $\mathsf{R}_{s,l}$ and $\mathsf{R}_{s,r}$, where $s \in \{\neg, \vee, \wedge, \to, \leftrightarrow, \forall, \exists\}$, given in Figure 5.12.

If $= \in \mathcal{L}$, we add the rule:

$$\frac{\Gamma, \alpha \Rightarrow \Gamma'}{\Gamma \Rightarrow \Gamma'}\,\mathsf{R}_{=,l}\,,$$

$$\frac{\Gamma \Rightarrow \Gamma', \varphi}{\Gamma, (\neg\varphi) \Rightarrow \Gamma'}\,\mathsf{R}_{\neg,l} \qquad \frac{\Gamma, \varphi \Rightarrow \Gamma'}{\Gamma \Rightarrow \Gamma', (\neg\varphi)}\,\mathsf{R}_{\neg,r}$$

$$\frac{\Gamma, \varphi \Rightarrow \Gamma' \quad \Gamma, \psi \Rightarrow \Gamma'}{\Gamma, (\varphi \vee \psi) \Rightarrow \Gamma'}\,\mathsf{R}_{\vee,l} \qquad \frac{\Gamma \Rightarrow \Gamma', \varphi, \psi}{\Gamma \Rightarrow \Gamma', (\varphi \vee \psi)}\,\mathsf{R}_{\vee,r}$$

$$\frac{\Gamma, \varphi, \psi \Rightarrow \Gamma'}{\Gamma, (\varphi \wedge \psi) \Rightarrow \Gamma'}\,\mathsf{R}_{\wedge,l} \qquad \frac{\Gamma \Rightarrow \Gamma', \varphi \quad \Gamma \Rightarrow \Gamma', \psi}{\Gamma \Rightarrow \Gamma', (\varphi \wedge \psi)}\,\mathsf{R}_{\wedge,r}$$

$$\frac{\Gamma \Rightarrow \Gamma', \varphi \quad \Gamma, \psi \Rightarrow \Gamma'}{\Gamma, (\varphi \rightarrow \psi) \Rightarrow \Gamma'}\,\mathsf{R}_{\rightarrow,l} \qquad \frac{\Gamma, \varphi \Rightarrow \Gamma', \psi}{\Gamma \Rightarrow \Gamma', (\varphi \rightarrow \psi)}\,\mathsf{R}_{\rightarrow,r}$$

$$\frac{\Gamma, \varphi, \psi \Rightarrow \Gamma' \quad \Gamma \Rightarrow \Gamma', \varphi, \psi}{\Gamma, (\varphi \leftrightarrow \psi) \Rightarrow \Gamma'}\,\mathsf{R}_{\leftrightarrow,l} \qquad \frac{\Gamma, \varphi \Rightarrow \Gamma', \psi \quad \Gamma, \psi \Rightarrow \Gamma', \varphi}{\Gamma \Rightarrow \Gamma', (\varphi \leftrightarrow \psi)}\,\mathsf{R}_{\leftrightarrow,r}$$

$$\frac{\Gamma, (\forall x)\varphi, (\varphi')_{x:=t} \Rightarrow \Gamma'}{\Gamma, (\forall x)\varphi \Rightarrow \Gamma'}\,\mathsf{R}_{\forall,l} \qquad \frac{\Gamma \Rightarrow \Gamma', (\varphi)_{x:=c}}{\Gamma \Rightarrow \Gamma', (\forall x)\varphi}\,\mathsf{R}_{\forall,r}(\textit{nondegenerate})$$

$$\frac{\Gamma \Rightarrow \Gamma', \varphi}{\Gamma \Rightarrow \Gamma', (\forall x)\varphi}\,\mathsf{R}_{\forall,r}(\textit{degenerate})$$

$$\frac{\Gamma, (\varphi)_{x:=c} \Rightarrow \Gamma'}{\Gamma, (\exists x)\varphi \Rightarrow \Gamma'}\,\mathsf{R}_{\exists,l}(\textit{nondegenerate}) \qquad \frac{\Gamma \Rightarrow \Gamma', (\exists x)\varphi, (\varphi')_{x:=t}}{\Gamma \Rightarrow \Gamma', (\exists x)\varphi}\,\mathsf{R}_{\exists,r}$$

$$\frac{\Gamma, \varphi \Rightarrow \Gamma'}{\Gamma, (\exists x)\varphi \Rightarrow \Gamma'}\,\mathsf{R}_{\exists,l}(\textit{degenerate})$$

for all sets of $(\mathcal{L}, V)$-formulas Γ, Γ', $(\mathcal{L}, V)$-formulas φ, ψ, $(\mathcal{L}, V)$-terms t and constant symbols c of $\mathcal{L}$ such that c does not occur in $\Gamma \cup \Gamma' \cup \{\varphi\}$. In rules $R_{\forall,i}$ and $R_{\exists,r}$ φ' is any variant of φ such that t is substitutable for x in φ'. In the nondegenerate cases of rules $\mathsf{R}_{\exists,l}$ and $\mathsf{R}_{\forall,r}$, we assume $x \in \mathtt{FV}(\varphi)$. Note that in both these cases, the constant symbol c that replaces the variable x is uniquely determined and it is referred to as an *eigenconstant*. In the degenerate cases of these rules, we require $x \notin \mathtt{FV}(\varphi)$.

Fig. 5.12. Rules of inference for sequents.

for all sets of $(\mathcal{L}, V)$-formulas Γ, Γ' and for every $\alpha \in \mathrm{INST}_{\mathcal{L},V}$ $(\mathrm{Eq}_{=,\mathcal{L}})$.

As in propositional logic, the rules $\mathsf{R}_{s,l}$ and $\mathsf{R}_{s,r}$ are often called the *s-left* and the *s-right* rule, respectively, for each symbol s. The rule $R_{=,l}$ is the *equality rule*; all other rules are referred to as *non-equality rules.*

The *variant rule* is

$$\frac{\Gamma_0 \Rightarrow \Gamma'_0}{\Gamma_1 \Rightarrow \Gamma'_1}\ \mathsf{R}_{vrs}\ ,$$

where $\Gamma_0, \Gamma'_0, \Gamma_1, \Gamma'_1$ are sets of $(\mathcal{L}, V)$-formulas, for which there are two finite-to-one functions $f : \Gamma_0 \longrightarrow \Gamma_1$ and $f' : \Gamma'_0 \longrightarrow \Gamma'_1$ such that each formula $\psi_0 \in \Gamma_0$ is a variant of $f(\psi_0)$ and each formula $\psi'_0 \in \Gamma'_0$ is a variant of $f'(\psi'_0)$. If we add this rule, we obtain the formal system $\mathcal{F}^{\mathrm{seq}}_{\mathcal{L},V}$.

In the special case when both f and f' are identity functions, we refer to the instance of the variant rule as an instance of *thinning.* Clearly, in this case, we have $\Gamma_0 \subseteq \Gamma_1$ and $\Gamma'_0 \subseteq \Gamma'_1$. □

Observe that, for every instance of a rule of $\mathcal{F}^{\mathrm{seq,cons}}_{\mathcal{L},V}$, the conclusion is a finite sequent if and only if all of the hypotheses are finite sequents. For the formal system $\mathcal{F}^{\mathrm{seq}}_{\mathcal{L},V}$, the finiteness of the conclusion of a rule entails the finiteness of the premises.

Example 5.5.9. Let $\mathtt{T}_0, \ldots, \mathtt{T}_{n-1}$ be $\mathcal{F}^{\mathrm{seq}}_{\mathcal{L},V}$-proof trees for the sequents $\Gamma \Rightarrow \varphi_0, \ldots, \Gamma \Rightarrow \varphi_{n-1}$, respectively, where $n \geq 1$. Starting from these proof trees, we can construct effectively, a proof tree for the sequent $\Gamma \Rightarrow (\varphi_0 \wedge \cdots \wedge \varphi_{n-1})$, using the following recursive process. We are given a proof tree for $\Gamma \Rightarrow \varphi_0$. Assume that $0 \leq i \leq n-2$ and we have a proof tree $\mathtt{T}'_i$ for $\Gamma \Rightarrow (\varphi_0 \wedge \cdots \wedge \varphi_i)$. By applying the $\mathsf{R}_{\wedge,r}$ rule we obtain the proof tree $\mathtt{T}'_{i+1} = (\mathtt{T}'_i, \mathtt{T}_{i+1}; \Gamma \Rightarrow (\varphi_0 \wedge \cdots \wedge \varphi_{i+1}))$. The desired proof tree is $\mathtt{T}'_{n-1}$. □

Example 5.5.10. Let φ, ψ be two $(\mathcal{L}, V)$-formulas. Since $\varphi \Rightarrow \psi, \varphi$ and $\varphi, \psi \Rightarrow \psi$ are axioms and

$$\frac{\varphi \Rightarrow \psi, \varphi \quad \varphi, \psi \Rightarrow \psi}{\varphi, (\varphi \to \psi) \Rightarrow \psi}$$

is an instance of the rule $\mathsf{R}_{\to,l}$ of the formal system $\mathcal{F}^{\mathrm{seq,cons}}_{\mathcal{L},V}$, it follows that $\varphi, (\varphi \to \psi) \Rightarrow \psi$ is a theorem of $\mathcal{F}^{\mathrm{seq,cons}}_{\mathcal{L},V}$ and a proof tree can be found effectively given φ and ψ. This is clearly the counterpart of modus ponens in the realm of sequents. ▯

Example 5.5.11. Let φ, ψ be two $(\mathcal{L}, V)$-formulas. Starting from the axioms $\varphi, \psi \Rightarrow \psi$ and $\varphi \Rightarrow \varphi, \psi$, by using the rule $\mathsf{R}_{\leftrightarrow,l}$, we obtain the sequent $\varphi, (\varphi \leftrightarrow \psi) \Rightarrow \psi$. An application of the rule $\mathsf{R}_{\wedge,l}$ yields the sequent $(\varphi \wedge (\varphi \leftrightarrow \psi)) \Rightarrow \psi$. Thus, $(\varphi \wedge (\varphi \leftrightarrow \psi)) \Rightarrow \psi$ is a theorem of $\mathcal{F}^{\text{seq,cons}}_{\mathcal{L},V}$ and a proof tree can be constructed effectively starting from φ and ψ. ▯

Example 5.5.12. Let x, y be two variables, $\mathcal{L}$ be a first-order language and let α be an $\mathcal{L}$-formula such that y is substitutable for x in α and y does not occur free in α.

To produce a proof of the sequent $(\forall x)\alpha \Rightarrow (\forall y)(\alpha)_{x:=y}$ in $\mathcal{F}^{\text{seq,cons}}_{\mathcal{L}^c,\text{VAR}}$, we need to show first that if c is a constant symbol of $\mathcal{L}^c - \mathcal{L}$, $(\alpha)_{x:=c} = ((\alpha)_{x:=y})_{y:=c}$. We may assume that x is different from y since otherwise the proof is immediate. The equality we need follows from Theorem 4.3.86, Theorem 1.2.21, and from the facts that y is substitutable for x in α and y does not occur free in α. Furthermore, the previous equality can now be written as $\langle\alpha\rangle_{x:=c} = \langle(\alpha)_{x:=y}\rangle_{y:=c}$, which yields the following proof in $\mathcal{F}^{\text{seq,cons}}_{\mathcal{L}^c,\text{VAR}}$:

(1) $(\forall x)\alpha, \langle\alpha\rangle_{x:=c} \Rightarrow \langle\alpha\rangle_{x:=c}$
(Axiom of $\mathcal{F}^{\text{seq,cons}}_{\mathcal{L}^c,\text{VAR}}$)
(2) $(\forall x)\alpha \Rightarrow \langle(\alpha)_{x:=y}\rangle_{y:=c}$
(By $R_{\forall,l}$ and the equality proved above)
(3) $(\forall x)\alpha \Rightarrow (\forall y)(\alpha)_{x:=y}$
(By $R_{\forall,r}$).

Note that the passage from Step (2) to Step (3) is valid regardless of whether we have the nondegenerate or degenerate application of the rule $\mathsf{R}_{\forall,r}$.

Similarly, a proof in $\mathcal{F}^{\text{seq,cons}}_{\mathcal{L}^c,\text{VAR}}$ of the sequent $(\exists y)(\alpha)_{x:=y} \Rightarrow (\exists x)\alpha$ can be produced as:

(1) $\langle\alpha\rangle_{x:=c} \Rightarrow (\exists x)\alpha, \langle\alpha\rangle_{x:=c}$
(Axiom of $\mathcal{F}^{\text{seq,cons}}_{\mathcal{L}^c,\text{VAR}}$)
(2) $\langle(\alpha)_{x:=y}\rangle_{y:=c} \Rightarrow (\exists x)\alpha$
(By $R_{\exists,r}$ and the equality proved above)
(3) $(\exists y)(\alpha)_{x:=y} \Rightarrow (\exists x)\alpha$
(By $R_{\exists,l}$).

▯

The soundness and completeness of the formal systems $\mathcal{F}^{\text{seq,cons}}_{\mathcal{L},V}$ and $\mathcal{F}^{\text{seq}}_{\mathcal{L},V}$ are shown using $(\mathcal{L}, V)$-tableaux. To this end, we need the next theorem, which explains the form of the rules of the sequent formal systems.

Theorem 5.5.13. *Let $\mathcal{L}$ be a first-order language, V be an $\mathcal{L}$-suitable set of variables and Δ be a set of signed $\mathcal{L}$-formulas.*

(1) *If $b\varphi$ is a signed $\mathcal{L}$-formula that is neither a $\boldsymbol{\gamma}$- nor a $\boldsymbol{\delta}$-formula, and the sequence of $(\mathcal{L}, V)$-constituents $\mathtt{d}_{\mathcal{L},V}(b\varphi)$ is $(K_0, \ldots, K_{n-1})$, then*

$$\frac{\mathtt{sqt}_{\mathcal{L},V}(\Delta \cup K_0), \ldots, \mathtt{sqt}_{\mathcal{L},V}(\Delta \cup K_{n-1})}{\mathtt{sqt}_{\mathcal{L},V}(\Delta \cup \{b\varphi\})}$$

is an instance of a rule of $\mathcal{F}^{\text{seq,cons}}_{\mathcal{L},V}$.

(2) *If $b\varphi = b(Qx)\psi$ is a $\boldsymbol{\gamma}$-formula, $t \in \mathrm{TERM}_{\mathcal{L}}(V)$, and ψ' is a variant of ψ such that t is substitutable for x in ψ', then*

$$\frac{\mathtt{sqt}_{\mathcal{L},V}(\Delta \cup \{b\varphi\} \cup \{b(\psi')_{x:=t}\})}{\mathtt{sqt}_{\mathcal{L},V}(\Delta \cup \{b\varphi\})}$$

is an instance of a rule of $\mathcal{F}^{\text{seq,cons}}_{\mathcal{L},V}$.

(3) *If $b\varphi = b(Qx)\psi$ is a $\boldsymbol{\delta}$-formula, $x \in \mathtt{FV}(\psi)$, and c is a constant symbol of $\mathcal{L}$ that does not occur in $\Delta \cup \{b\varphi\}$, then*

$$\frac{\mathtt{sqt}_{\mathcal{L},V}(\Delta \cup \{b(\psi)_{x:=c}\})}{\mathtt{sqt}_{\mathcal{L},V}(\Delta \cup \{b\varphi\})}$$

is an instance of a rule of $\mathcal{F}^{\text{seq,cons}}_{\mathcal{L},V}$. If $x \notin \mathtt{FV}(\psi)$, then

$$\frac{\mathtt{sqt}_{\mathcal{L},V}(\Delta \cup \{b\psi\})}{\mathtt{sqt}_{\mathcal{L},V}(\Delta \cup \{b\varphi\})}$$

is an instance of a rule of $\mathcal{F}^{\text{seq,cons}}_{\mathcal{L},V}$.

(4) *If $\alpha \in \mathrm{INST}_{\mathcal{L},V}(Eq_{=,\mathcal{L}})$, then*

$$\frac{\mathtt{sqt}_{\mathcal{L},V}(\Delta \cup \{\mathbf{T}\alpha\})}{\mathtt{sqt}_{\mathcal{L},V}(\Delta)}$$

is an instance of a rule of $\mathcal{F}^{\text{seq,cons}}_{\mathcal{L},V}$.

(5) *If* Δ_0, Δ_1 *are two sets of signed* $(\mathcal{L}, V)$*-formulas and* $f : \Delta_0 \longrightarrow \Delta_1$ *is a finite-to-one function such that for all* $b\varphi \in \Delta_0$*,* $f(b\varphi)$ *is a variant of* $b\varphi$*, then*

$$\frac{\mathtt{sqt}_{\mathcal{L},V}(\Delta_0)}{\mathtt{sqt}_{\mathcal{L},V}(\Delta_1)}$$

is an instance of a rule of $\mathcal{F}^{\mathrm{seq}}_{\mathcal{L},V}$*.*

Furthermore, every instance of every rule of $\mathcal{F}^{\mathrm{seq,cons}}_{\mathcal{L},V}$ *can be obtained in one of the first four ways and every instance of every rule of* $\mathcal{F}^{\mathrm{seq}}_{\mathcal{L},V}$ *can be obtained in one of the five ways given above.*

Proof. The theorem follows by inspecting the definition of $\mathtt{d}_{\mathcal{L},V}(b\varphi)$ and the form of the rules of $\mathcal{F}^{\mathrm{seq}}_{\mathcal{L},V}$. □

Rules of the first form are called *propositional rules* due to their similarity to the rules for sequents introduced in propositional logic. Rules of the second and third forms are called $\boldsymbol{\gamma}$*-rules* and $\boldsymbol{\delta}$*-rules*, respectively.

Corollary 5.5.14. *For each instance*

$$\frac{\kappa_0, \ldots, \kappa_{n-1}}{\kappa}$$

of a rule R *of* $\mathcal{F}^{\mathrm{seq}}_{\mathcal{L},V}$ *one of the following three cases holds:*

(1) *R is neither the equality rule nor the variant rule, there is set of signed $(\mathcal{L}, V)$-formulas Δ and a signed $(\mathcal{L}, V)$-formula $b\varphi$ (where φ is not an atomic formula) such that* $\mathtt{sf}_{\mathcal{L},V}(\kappa) = \Delta \cup \{b\varphi\}$ *and one of the following situations occurs:*

  (a) $b\varphi$ *is neither a* $\boldsymbol{\gamma}$*- nor a* $\boldsymbol{\delta}$*-formula, the sequence of* $(\mathcal{L}, V)$*-constituents* $\mathtt{d}_{\mathcal{L},V}(b\varphi)$ *is* $(K_0, \ldots, K_{n-1})$*, and* $\mathtt{sf}_{\mathcal{L},V}(\kappa_i) = \Delta \cup K_i$ *for* $0 \leq i \leq n-1$*.*

  (b) $n = 1$*,* $b\varphi = b(Qx)\psi$ *is a* $\boldsymbol{\gamma}$*-formula, and there are a* $t \in \mathrm{TERM}_{\mathcal{L}}(V)$ *and a variant* ψ' *of* ψ *such that* t *is substitutable for* x *in* ψ' *and* $\mathtt{sf}_{\mathcal{L},V}(\kappa_0) = \Delta \cup \{b\varphi, b\langle\psi\rangle_{x:=t}\}$*.*

  (c) $n = 1$*,* $b\varphi = b(Qx)\psi$ *is a* $\boldsymbol{\delta}$*-formula, and either* $x \in \mathtt{FV}(\psi)$ *and there is a constant symbol* c *of* $\mathcal{L}$ *that does not occur in* $\Delta \cup \{b\varphi\}$ *such that* $\mathtt{sf}_{\mathcal{L},V}(\kappa_0) = \Delta \cup \{b(\psi)_{x:=c}\}$*, or* $x \notin \mathtt{FV}(\psi)$ *and* $\mathtt{sf}_{\mathcal{L},V}(\kappa_0) = \Delta \cup \{b\psi\}$*.*

(2) *R is the equality rule, $n = 1$, there is $\varphi \in \mathrm{INST}_{\mathcal{L},V}(Eq_{=,\mathcal{L}})$ such that $\mathtt{sf}_{\mathcal{L},V}(\kappa_0) = \Delta \cup \{\mathbf{T}\varphi\}$ and $\mathtt{sf}_{\mathcal{L},V}(\kappa) = \Delta$.*

(3) *R is the variant rule, $n = 1$, and there is a finite-to-one function $f : \mathtt{sf}_{\mathcal{L},V}(\kappa_0) \longrightarrow \mathtt{sf}_{\mathcal{L},V}(\kappa)$ such that if $f(b_0\varphi_0) = b\varphi$, then $b\varphi$ is a variant of $b_0\varphi_0$.*

Proof. The statement follows immediately from Theorem 5.5.13, taking into account the fact that $\mathtt{sqt}_{\mathcal{L},V}$ and $\mathtt{sf}_{\mathcal{L},V}$ are inverse bijections. □

If

$$\frac{\Gamma_0 \Rightarrow \Gamma'_0, \ldots, \Gamma_{n-1} \Rightarrow \Gamma'_{n-1}}{\Gamma \Rightarrow \Gamma'}$$

is an instance of a rule of $\mathcal{F}^{\mathrm{seq}}_{\mathcal{L},V}$, then every member of Γ_i, Γ'_i belongs to the analytical universe $W^*_{\mathcal{L},V}(\Gamma \cup \Gamma')$ or is an instance of an equality axiom, for $0 \leq i \leq n-1$. This shows the analyticity of the formal system $\mathcal{F}^{\mathrm{seq}}_{\mathcal{L},V}$.

Definition 5.5.15. Let

$$\frac{\kappa_0, \ldots, \kappa_{n-1}}{\kappa}$$

be an instance of a rule of $\mathcal{F}^{\mathrm{seq}}_{\mathcal{L},V}$ that is neither the equality rule nor the variant rule and let Δ and $b\varphi$ be as in the first case of Corollary 5.5.14. Then we call φ a *principal formula* of the instance. If $b\varphi$ is a $\boldsymbol{\gamma}$-formula or $b\varphi \in \Delta$, we call the instance an *instance with retention*; otherwise, we refer to this instance as an *instance with removal.* ▯

As in the case of propositional logic, it is not difficult to see that an instance of a rule of $\mathcal{F}^{\mathrm{seq}}_{\mathcal{L},V}$ cannot be both an instance with retention and an instance with removal. Further, in an instance with removal, there is only one principal formula.

The connection between deduction trees of $\mathcal{F}^{\mathrm{seq}}_{\mathcal{L},V}$ and $\mathcal{F}^{\mathrm{tabl}}_{\mathcal{L},V}$ is presented in the next theorem.

Theorem 5.5.16. *Let $\mathcal{L}$ be a first-order language and V be an $\mathcal{L}$-suitable set of variables. For $\mathtt{T} \in \mathcal{GDT}_{\mathcal{F}^{\mathrm{seq}}_{\mathcal{L},V}}$, let $\Phi_{\mathcal{L},V}(\mathtt{T}) = \mathtt{sf}_{\mathcal{L},V} \circ \mathtt{T}$ and for $\mathtt{T} \in \mathcal{GDT}_{\mathcal{F}^{\mathrm{tabl}}_{\mathcal{L},V}}$, let $\Psi_{\mathcal{L},V}(\mathtt{T}) = \mathtt{sqt}_{\mathcal{L},V} \circ \mathtt{T}$. Then, $\Phi_{\mathcal{L},V}$:*

$\mathcal{GDT}_{\mathcal{F}^{\mathrm{seq}}_{\mathcal{L},V}} \to \mathcal{GDT}_{\mathcal{F}^{\mathrm{tabl}}_{\mathcal{L},V}}$ and $\Psi_{\mathcal{L},V} : \mathcal{GDT}_{\mathcal{F}^{\mathrm{tabl}}_{\mathcal{L},V}} \longrightarrow \mathcal{GDT}_{\mathcal{F}^{\mathrm{seq}}_{\mathcal{L},V}}$ are inverse bijections. Further, we have $\Phi_{\mathcal{L},V}(\mathcal{PT}_{\mathcal{F}^{\mathrm{seq}}_{\mathcal{L},V}}) = \mathcal{PT}_{\mathcal{F}^{\mathrm{tabl}}_{\mathcal{L},V}}$ and $\Psi_{\mathcal{L},V}(\mathcal{PT}_{\mathcal{F}^{\mathrm{tabl}}_{\mathcal{L},V}}) = \mathcal{PT}_{\mathcal{F}^{\mathrm{seq}}_{\mathcal{L},V}}$.

Proof. Corollary 5.5.14 implies that $\Phi_{\mathcal{L},V}$ maps $\mathcal{GDT}_{\mathcal{F}^{\mathrm{seq}}_{\mathcal{L},V}}$ into $\mathcal{GDT}_{\mathcal{F}^{\mathrm{tabl}}_{\mathcal{L},V}}$ and Theorem 5.5.13 implies that $\Psi_{\mathcal{L},V}$ maps $\mathcal{GDT}_{\mathcal{F}^{\mathrm{tabl}}_{\mathcal{L},V}}$ into $\mathcal{GDT}_{\mathcal{F}^{\mathrm{seq}}_{\mathcal{L},V}}$. Since $\mathtt{sf}_{\mathcal{L},V}$ and $\mathtt{sqt}_{\mathcal{L},V}$ are inverse mappings, it follows immediately that $\Phi_{\mathcal{L},V}$ and $\Psi_{\mathcal{L},V}$ are inverse mappings. The second part of the theorem follows from the fact that the mappings $\mathtt{sf}_{\mathcal{L},V}$ and $\mathtt{sqt}_{\mathcal{L},V}$ preserve the axioms of the formal systems $\mathcal{F}^{\mathrm{seq}}_{\mathcal{L},V}$ and $\mathcal{F}^{\mathrm{tabl}}_{\mathcal{L},V}$; in other words, if κ is an axiom of $\mathcal{F}^{\mathrm{seq}}_{\mathcal{L},V}$, then $\mathtt{sf}_{\mathcal{L},V}(\kappa)$ is an axiom of $\mathcal{F}^{\mathrm{tabl}}_{\mathcal{L},V}$, and if Δ is an axiom of $\mathcal{F}^{\mathrm{tabl}}_{\mathcal{L},V}$, then $\mathtt{sqt}_{\mathcal{L},V}(\Delta)$ is an axiom of $\mathcal{F}^{\mathrm{seq}}_{\mathcal{L},V}$. □

Theorem 5.5.17. *Let $\mathcal{L}$ be a first-order language and V be an $\mathcal{L}$-suitable set of variables. For $\mathtt{T} \in \mathcal{GDT}_{\mathcal{F}^{\mathrm{seq,cons}}_{\mathcal{L},V}}$, let $\Phi_{\mathcal{L},V}(\mathtt{T}) = \mathtt{sf}_{\mathcal{L},V} \circ \mathtt{T}$ and for $\mathtt{T} \in \mathcal{GDT}_{\mathcal{F}^{\mathrm{tabl,cons}}_{\mathcal{L},V}}$, let $\Psi_{\mathcal{L},V}(\mathtt{T}) = \mathtt{sqt}_{\mathcal{L},V} \circ \mathtt{T}$. Then, $\Phi_{\mathcal{L},V} : \mathcal{GDT}_{\mathcal{F}^{\mathrm{seq,cons}}_{\mathcal{L},V}} \to \mathcal{GDT}_{\mathcal{F}^{\mathrm{tabl,cons}}_{\mathcal{L},V}}$ and $\Psi_{\mathcal{L},V} : \mathcal{GDT}_{\mathcal{F}^{\mathrm{tabl,cons}}_{\mathcal{L},V}} \longrightarrow \mathcal{GDT}_{\mathcal{F}^{\mathrm{seq,cons}}_{\mathcal{L},V}}$ are inverse bijections. We also have*

$$\Phi_{\mathcal{L},V}(\mathcal{PT}_{\mathcal{F}^{\mathrm{seq,cons}}_{\mathcal{L},V}}) = \mathcal{PT}_{\mathcal{F}^{\mathrm{tabl,cons}}_{\mathcal{L},V}} \text{ and } \Psi_{\mathcal{L},V}(\mathcal{PT}_{\mathcal{F}^{\mathrm{tabl,cons}}_{\mathcal{L},V}}) = \mathcal{PT}_{\mathcal{F}^{\mathrm{seq,cons}}_{\mathcal{L},V}}.$$

Proof. The argument is virtually identical to that of Theorem 5.5.16. □

Theorem 5.5.18 (Soundness of $\mathcal{F}^{\mathrm{seq}}_{\mathcal{L},V}$). *Let $\mathcal{L}$ be a first-order language and V be a set of variables. Every theorem of the formal system $\mathcal{F}^{\mathrm{seq}}_{\mathcal{L},V}$ is a valid $(\mathcal{L}, V)$-sequent.*

Proof. Let κ be a theorem of $\mathcal{F}^{\mathrm{seq}}_{\mathcal{L},V}$ and let $\mathtt{T}$ be an $\mathcal{F}^{\mathrm{seq}}_{\mathcal{L},V}$-proof tree for κ. Then, $\Phi_{\mathcal{L},V}(\mathtt{T})$ is an $\mathcal{F}^{\mathrm{tabl}}_{\mathcal{L},V}$-proof tree for $\mathtt{sf}_{\mathcal{L},V}(\kappa)$, so $\mathtt{sf}_{\mathcal{L},V}(\kappa)$ is an unsatisfiable set of signed formulas by Theorem 5.3.36 which, by Corollary 5.5.7, implies the validity of κ. □

Corollary 5.5.19. *Let $\mathcal{L}$ be a first-order language and let V be an $\mathcal{L}$-suitable set of variables. Every theorem of the formal system $\mathcal{F}^{\mathrm{seq,cons}}_{\mathcal{L},V}$ is a valid $(\mathcal{L}, V)$-sequent.*

Proof. The statement follows immediately from Theorem 5.5.18. □

Theorem 5.5.20 (Partial Completeness of $\mathcal{F}^{\text{seq,cons}}_{\mathcal{L}^c,V}$). *Let $\mathcal{L}$ be a first-order language and let V be an $\mathcal{L}$-suitable set of variables. Every valid $(\mathcal{L}, V)$-sequent is a theorem of $\mathcal{F}^{\text{seq,cons}}_{\mathcal{L}^c,V}$.*

Proof. Let κ be a valid $(\mathcal{L}, V)$-sequent. Then, by Corollary 5.5.7 $\mathtt{sf}_{\mathcal{L},V}(\kappa)$ is unsatisfiable and, by Corollary 5.3.40, there is an $\mathcal{F}^{\text{tabl,cons}}_{\mathcal{L}^c,V}$-proof tree $\mathtt{T}$ for $\mathtt{sf}_{\mathcal{L},V}(\kappa)$. By Theorem 5.5.17, $\Psi_{\mathcal{L}^c,V}(\mathtt{T})$ is a $\mathcal{F}^{\text{seq,cons}}_{\mathcal{L}^c,V}$-proof tree for $\mathtt{sqt}_{\mathcal{L}^c,V}(\mathtt{sf}_{\mathcal{L},V}(\kappa)) = \mathtt{sqt}_{\mathcal{L}^c,V}(\mathtt{sf}_{\mathcal{L}^c,V}(\kappa)) = \kappa$. □

Corollary 5.5.21. *Let $\mathcal{L}$ be a first-order language and let V be a set of variables. Every valid $(\mathcal{L}, V)$-sequent is a theorem of $\mathcal{F}^{\text{seq}}_{\mathcal{L}^c,V}$.*

Proof. This statement follows immediately from Theorem 5.5.20. □

Corollary 5.5.22. *Let $\mathcal{L}$ be a first-order language, V be a set of variables and κ be an $(\mathcal{L}, V)$-sequent. Then, the following three statements are equivalent.*

(1) *κ is valid;*
(2) *κ is a theorem of $\mathcal{F}^{\text{seq,cons}}_{\mathcal{L}^c,V}$;*
(3) *κ is a theorem of $\mathcal{F}^{\text{seq}}_{\mathcal{L}^c,V}$.*

Proof. This follows immediately from Theorems 5.5.20 (applied to $\mathcal{L}^c$) and 5.5.18 and the fact that $\mathcal{F}^{\text{seq}}_{\mathcal{L}^c,V}$ is an extension of $\mathcal{F}^{\text{seq,cons}}_{\mathcal{L}^c,V}$. □

Definition 5.5.23. Let $\mathcal{L}$ be a first-order language and V be an $\mathcal{L}$-suitable set of variables. An $(\mathcal{L}, V)$-sequent κ is *dominated by* a set of $(\mathcal{L}, V)$-sequents S if $\mathtt{sf}_{\mathcal{L},V}(\kappa) \subseteq \mathtt{sf}_{\mathcal{L},V}(S)$. If $S = \{\kappa'\}$, then we say that κ is dominated by κ' when κ is dominated by S. In other words, $\Gamma_0 \Rightarrow \Gamma_1$ is dominated by $\Gamma'_0 \Rightarrow \Gamma'_1$ if $\Gamma_0 \subseteq \Gamma'_0$ and $\Gamma_1 \subseteq \Gamma'_1$. □

Theorem 5.5.24. *Let $\mathcal{L}$ be a first-order language and let V be an $\mathcal{L}$-suitable set of variables. There is an effective, syntactic construction that starts with an $\mathcal{F}^{\text{seq}}_{\mathcal{L},V}$-proof tree $\mathtt{T}$ for a sequent κ and produces an $\mathcal{F}^{\text{seq}}_{\mathcal{L},V}$-proof tree $\mathtt{T}'$ for a finite sequent κ' that is dominated by κ.*

Proof. The argument involves the translation mappings

$$\Phi_{\mathcal{L},V} : \mathcal{GDT}_{\mathcal{F}^{\rm seq}_{\mathcal{L},V}} \to \mathcal{GDT}_{\mathcal{F}^{\rm tabl}_{\mathcal{L},V}} \text{ and } \Psi_{\mathcal{L},V} : \mathcal{GDT}_{\mathcal{F}^{\rm tabl}_{\mathcal{L},V}} \longrightarrow \mathcal{GDT}_{\mathcal{F}^{\rm seq}_{,\mathcal{V}}}.$$

We saw in Theorem 5.5.16 that these functions map proof trees to proof trees. Let $\mathtt{T}$ be an $\mathcal{F}^{\rm seq}_{\mathcal{L},V}$-proof tree for a sequent κ. Then, $\mathtt{T}_0 = \Phi_{\mathcal{L},V}(\mathtt{T})$ is an $\mathcal{F}^{\rm tabl}_{\mathcal{L},V}$-proof tree for the set $\Delta = \mathtt{sf}_{\mathcal{L},V}(\kappa)$, that is a strongly closed $(\Delta, \mathcal{L}, V)$-tableau. By Theorem 5.3.32, we effectively obtain a strongly closed $(\Delta', \mathcal{L}, V)$-tableau $\mathtt{T}'_0$, where Δ' is a finite subset of Δ. Thus, there is a finite sequent $\kappa' = \mathtt{sqt}_{\mathcal{L},V}(\Delta')$ such that $\mathtt{T}' = \Psi_{\mathcal{L},V}(\mathtt{T}'_0)$ is an $\mathcal{F}^{\rm seq}_{\mathcal{L},V}$-proof tree for κ'. Since $\mathtt{sf}_{\mathcal{L},V}(\kappa) = \Delta$ and $\mathtt{sf}_{\mathcal{L},V}(\kappa') = \Delta'$, κ dominates κ'. □

Theorem 5.5.25 (Compactness Theorem for Sequents). *Let $\mathcal{L}$ be a first-order language and V be an $\mathcal{L}$-suitable set of variables. For every $(\mathcal{L}, V)$-sequent κ, κ is valid if and only if it dominates a valid finite $(\mathcal{L}, V)$-sequent κ'.*

Proof. It is clear that if κ dominates a valid (finite) sequent κ', then κ is valid.

Suppose that κ is a valid sequent. By the Completeness Theorem, there is an $\mathcal{F}^{\rm seq}_{\mathcal{L},V}$-proof tree for κ. By Theorem 5.5.24, there is an $\mathcal{F}^{\rm seq}_{\mathcal{L},V}$-proof tree for a finite sequent κ' dominated by κ. By the Soundness Theorem, κ' is valid. □

Definition 5.5.26. A general $\mathcal{F}^{\rm seq}_{\mathcal{L},V}$-deduction tree $\mathtt{T}$ is *finished* if $\Phi_{\mathcal{L},V}(\mathtt{T})$ is a strongly completed tableau. ▯

Definition 5.5.27. An $(\mathcal{L}, V)$-sequent κ' is an $(\mathcal{L}, V)$*-constituent* of the $(\mathcal{L}, V)$-sequent $\varphi \Rightarrow$ if $\kappa' = \mathtt{sqt}_{\mathcal{L},V}(K)$ where K is a constituent of $\mathbf{T}\varphi$; κ' is a constituent of the sequent $\Rightarrow \varphi$ if $\kappa' = \mathtt{sqt}_{\mathcal{L},V}(K)$ where K is a constituent of $\mathbf{F}\varphi$.

A branch $\mathtt{B}$ of a general $\mathcal{F}^{\rm seq}_{\mathcal{L},V}$-deduction tree $\mathtt{T}$ is called *finished* if

(1) there is no atomic formula φ such that $\mathtt{T}(\mathtt{B})$ dominates both $\varphi \Rightarrow$ and $\Rightarrow \varphi$;
(2) for every sequent κ of the form $\varphi \Rightarrow$ or $\Rightarrow \varphi$, with φ not atomic, if κ is dominated by $\mathtt{T}(\mathtt{B})$, then there is an $(\mathcal{L}, V)$-constituent κ' of κ that is dominated by $\mathtt{T}(\mathtt{B})$;

(3) if $=$ is in $\mathcal{L}$ and $\varphi \in \mathrm{INST}_{\mathcal{L},V}(\mathrm{Eq}_{=,\mathcal{L}})$, then $\mathtt{T(B)}$ dominates $\varphi \Rightarrow$.

▯

Theorem 5.5.28. *Let $\mathcal{L}$ be a first-order language and let V be an $\mathcal{L}$-suitable set of variables. A general $\mathcal{F}^{\mathrm{seq}}_{\mathcal{L},V}$-deduction tree* T *is finished if and only if every branch of* T *is either finished or ends with a node labeled by an axiom of $\mathcal{F}^{\mathrm{seq}}_{\mathcal{L},V}$.*

Proof. The theorem follows immediately from the observation that a branch B of a general $\mathcal{F}^{\mathrm{seq}}_{\mathcal{L},V}$-deduction tree T is finished (in the sense of Definition 5.5.27) if and only if $\mathtt{sf}_{\mathcal{L},V}(\mathtt{T(B)})$ is a Hintikka set. □

Theorem 5.5.29. *Let $\mathcal{L}$ be a first-order language and let V be a set of variables. If κ is an $(\mathcal{L},V)$-sequent, there is a finished general $\mathcal{F}^{\mathrm{seq,cons}}_{\mathcal{L}^c,V}$-deduction tree for κ.*

Proof. Using Construction 5.3.25, it is possible to construct a conservative, strongly completed $(\mathtt{sf}_{\mathcal{L},V}(\kappa), \mathcal{L}^c, V)$-tableau T which is general deduction tree of $\mathcal{F}^{\mathrm{tabl,cons}}_{\mathcal{L}^c,V}$. Then, $\mathtt{sqt}_{\mathcal{L}^c,V} \circ \mathtt{T}$ is a finished general $\mathcal{F}^{\mathrm{seq,cons}}_{\mathcal{L}^c,V}$-deduction tree for κ. □

The proof of Theorem 5.5.29 in fact gives an indirect construction of a finished general $\mathcal{F}^{\mathrm{seq,cons}}_{\mathcal{L}^c,V}$-deduction tree for a given sequent κ. We can formulate this construction in terms of sequents.

Definition 5.5.30. Let $\mathcal{L}$ be a first-order language and let V be an $\mathcal{L}$-suitable set of variables. Let T be a general $\mathcal{F}^{\mathrm{seq}}_{\mathcal{L},V}$-deduction tree and let P be a path of T. The *set of pairs that require attention in* T *on* P is the set of all pairs $(b\varphi, t)$ in $\mathrm{SFORM}_{\mathcal{L}}(V) \times \mathrm{TERM}_{\mathcal{L}}(V)$ that satisfy one of the following conditions for $\kappa = \mathtt{sqt}_{\mathcal{L},V}(\{b\varphi\})$:

(1) κ is dominated by $\mathtt{T(P)}$, φ is not atomic, $b\varphi$ is not a $\boldsymbol{\gamma}$-formula and none of the constituents of κ is dominated by $\mathtt{T(P)}$.
(2) κ is dominated by $\mathtt{T(P)}$, $b\varphi = b(Qx)\psi$ is a $\boldsymbol{\gamma}$-formula and the sequent $\mathtt{sqt}_{\mathcal{L},V}(\{b\langle\psi\rangle_{x:=t}\})$ is not dominated by $\mathtt{T(P)}$.
(3) $=\in \mathcal{L}$, $b = \mathbf{T}$, $\varphi \in \mathrm{INST}_{\mathcal{L},V}(\mathrm{Eq}_{=,\mathcal{L}})$ and κ is not dominated by $\mathtt{T(P)}$.

If P is the path leading to the node q, we will refer to the set of pairs that require attention in T on P as the *set of pairs that require attention in* T *at q.* ▯

Construction 5.5.31.
Input: A first-order language $\mathcal{L}$, a set of variables V and an $(\mathcal{L}, V)$-sequent κ.
Output: A sequence $\mathtt{T}_0, \mathtt{T}_1, \ldots$ of $\mathcal{F}^{\text{seq,cons}}_{\mathcal{L}^c,V}$-deduction trees for κ such that each $\mathtt{T}_{i+1}$ is a leaf extension of $\mathtt{T}_i$ and $\mathtt{T} = \bigcup\{\mathtt{T}_i \mid i \geq 0\}$ is a finished general $\mathcal{F}^{\text{seq,cons}}_{\mathcal{L}^c,V}$-deduction tree for κ.
Method:

(A) Let $\mathtt{T}_0$ be the one-node tree with root labeled by κ.

(B) Suppose that $\mathtt{T}_i$ has been defined. Then, if $\mathtt{T}_i$ is finished, the construction stops with $\mathtt{T}_i$. Otherwise, $\mathtt{T}_i$ has branches that neither are finished nor end with an axiom. Select nondeterministically among the shortest such branches a branch $\mathtt{B}$ ending in the leaf q.
Choose $(b_q\varphi_q, t_q) \in \mathtt{T}_i(\mathtt{B}) \times \mathrm{TERM}_{\mathcal{L}^c}(V)$ to be the first pair $(b\varphi, t)$ in the standard order of $\mathrm{SFORM}_{\mathcal{L}^c}(V) \times \mathrm{TERM}_{\mathcal{L}^c}(V)$ that requires attention in $\mathtt{T}_i$ at q. (By an analog of Lemma 5.3.23, if Case 1 or 2 of Definition 5.5.30 applies, then $b_q\varphi_q \in \mathtt{T}_i(q)$.) The tableau $\mathtt{T}_{i+1}$ is defined as follows:

- If Case 1 of Definition 5.5.30 holds, $b_q\varphi_q$ is not a $\boldsymbol{\delta}$-formula, and $(K_0, \ldots, K_{n-1})$ is the $(\mathcal{L}^c, V)$-constituent sequence of $b_q\varphi_q$, then define $\mathtt{T}_{i+1}$ by adding to $\mathrm{Dom}(\mathtt{T}_i)$ the nodes $q0, \ldots, q_{n-1}$ and letting $\mathtt{T}_{i+1}(qj) = \mathtt{sqt}_{\mathcal{L}^c,V}(\Delta_q \cup K_j)$, where Δ_q is chosen such that $\mathtt{T}_i(q) = \mathtt{sqt}_{\mathcal{L}^c,V}(\Delta_q \cup \{b_q\varphi_q\})$.
- If Case 1 of Definition 5.5.30 holds and $b_q\varphi_q = b_q(Qx)\psi_q$ is a $\boldsymbol{\delta}$-formula, then there is a constant symbol $c \in \mathcal{L}^c$ that does not occur in $\mathtt{T}_i(q)$. This is the case because there are infinitely many constant symbols in $\mathcal{L}^c$ that do not occur in $\mathtt{T}_i(\lambda) = \kappa$ and at each expansion of a node of $\mathtt{T}_i$ we expand the corresponding sequent by adding a finite set of formulas to each side and, therefore, a finite set of new constant symbols. Define $\mathtt{T}_{i+1}$ by adding $q0$ to $\mathrm{Dom}(\mathtt{T}_i)$ and letting $\mathtt{T}_{i+1}(q0) = \mathtt{sqt}_{\mathcal{L}^c,V}(\Delta_q \cup \{\langle\psi_q\rangle_{x:=c}\})$, where Δ_q is chosen such that $\mathtt{sf}_{\mathcal{L}^c,V}(\mathtt{T}_i(q)) = \Delta_q \cup \{b_q\varphi_q\}$ and c is the first constant symbol in $\mathcal{L}^c$ that does not occur in $\mathtt{T}_i(q)$. (Note that if $x \notin \mathtt{FV}(\psi_q)$, then $\mathtt{T}_{i+1}(q0) = \mathtt{sqt}_{\mathcal{L}^c,V}(\Delta_q \cup \{b_q\psi_q\})$.)

- If Case 2 of Definition 5.5.30 occurs, define $\mathtt{T}_{i+1}$ by adding $q0$ to $\mathrm{Dom}(\mathtt{T}_i)$ and letting $\mathtt{T}_{i+1}(q0) = \mathtt{T}_i(q) \cup \mathtt{sqt}_{\mathcal{L}^c,V}(\{b_q\langle\psi_q\rangle_{x:=t_q}\})$.
- If Case 3 of Definition 5.5.30 occurs, define $\mathtt{T}_{i+1}$ be adding $q0$ to $\mathrm{Dom}(\mathtt{T}_i)$ and letting $\mathtt{T}_{i+1}(q0) = \mathtt{T}_i(q) \cup \mathtt{sqt}_{\mathcal{L}^c,V}(\{b_q\varphi_q\})$.

Proof of Correctness: It is clear that each $\mathtt{T}_i$, if defined, is a $\mathcal{F}^{\mathrm{seq,cons}}_{\mathcal{L}^c,V}$-deduction tree for κ and that $\mathtt{T}_{i+1}$, if defined, is a leaf extension of $\mathtt{T}_i$.

The sequence of trees $\mathtt{sf}_{\mathcal{L}^c,V} \circ \mathtt{T}_0, \ldots$ consists of conservative, $\mathtt{sf}_{\mathcal{L}^c,V}(\kappa)$-tableaux that could have been obtained by applying Construction 5.3.28 to $\mathtt{sf}_{\mathcal{L}^c,V}(\kappa)$. By the correctness of that construction, $\mathtt{T}' = \bigcup\{\mathtt{sf}_{\mathcal{L}^c,V} \circ \mathtt{T}_i \mid i \geq 0\}$ is a conservative, strongly completed $\mathtt{sf}_{\mathcal{L}^c,V}(\kappa)$-tableau. Therefore, since $\mathtt{sf}_{\mathcal{L}^c,v} \circ \mathtt{T} = \mathtt{T}'$, $\mathtt{T}$ is a finished general $\mathcal{F}^{\mathrm{seq,cons}}_{\mathcal{L}^c,V}$-deduction tree for κ. □

By analogy with Definition 5.3.6, we introduce $\mathcal{F}^{\mathrm{seq}}_{\mathcal{L},V}$-deduction trees with retention and removal. Namely, such a deduction tree is said to be with retention if at every interior node where neither the equality rule nor variantization is used, the principal formula is retained; a deduction tree is said to be with removal if at every interior node where neither the equality rule nor the γ-rule nor variantization was used, the principal formula is removed.

As usual, we observe that the algorithm can be used to produce a general $\mathcal{F}^{\mathrm{seq,cons}}_{\mathcal{L}^c,V}$-deduction tree with retention or with removal.

As just shown, given an $(\mathcal{L}, V)$-sequent κ, we can construct a finished general $\mathcal{F}^{\mathrm{seq,cons}}_{\mathcal{L}^c,V}$-deduction tree $\mathtt{T}$ for κ. If $\mathtt{T}$ is a proof tree, then κ is valid. Otherwise, $\mathtt{T}$ contains a finished branch $\mathtt{B}$, that is, a branch for which $\mathtt{sf}_{\mathcal{L}^c,V}(\mathtt{T}(\mathtt{B}))$ is an $(\mathcal{L}^c, V)$-Hintikka set. Using the proof Theorem 4.12.17 or 4.12.20, we can determine a pair $(\mathcal{A}, \sigma)$ that satisfies $\mathtt{sf}_{\mathcal{L}^c,V}(\mathtt{T}(\mathtt{B}))$ and therefore falsifies κ.

The cut rule for $(\mathcal{L}, V)$-sequents that we are about to introduce corresponds to the cut rule for $(\mathcal{L}, V)$-tableaux.

Definition 5.5.32. Let $\mathcal{L}$ be a first-order language and let V be an $\mathcal{L}$-suitable set of variables. $\mathcal{F}^{\mathrm{seq,cut}}_{\mathcal{L},V}$ is the formal system obtained from $\mathcal{F}^{\text{seq}}_{\mathcal{L},V}$ by adding the following "cut" rule:

$$\frac{\Gamma, \varphi \Rightarrow \Gamma' \quad \Gamma \Rightarrow \Gamma', \varphi}{\Gamma \Rightarrow \Gamma'} \mathsf{R}_{cut}$$

for all sets of formulas Γ, Γ' and $(\mathcal{L}, V)$-formulas φ. ▯

As with all formal systems with cut, the formal system $\mathcal{F}^{\text{seq,cut}}_{\mathcal{L},V}$ is not analytical. Cut elimination for sequents, which converts a non-analytical proof into an analytical one, is obtained by translating cut elimination for tableaux.

Theorem 5.5.33. *Let $\mathcal{L}$ be a first-order language and let V be an $\mathcal{L}$-suitable set of variables. For every $\mathtt{T} \in \mathcal{GDT}_{\mathcal{F}^{\text{seq,cut}}_{\mathcal{L},V}}$, let $\Phi^{\text{cut}}_{\mathcal{L},V}(\mathtt{T}) = \mathtt{sf}_{\mathcal{L},V} \circ \mathtt{T}$ and for $\mathtt{T} \in \mathcal{GDT}_{\mathcal{F}^{\text{tabl,cut}}_{\mathcal{L},V}}$, let $\Psi^{\text{cut}}_{\mathcal{L},V}(\mathtt{T}) = \mathtt{sqt}_{\mathcal{L},V} \circ \mathtt{T}$.*

The mappings

$$\Phi^{\text{cut}}_{\mathcal{L},V} : \mathcal{GDT}_{\mathcal{F}^{\text{seq,cut}}_{\mathcal{L},V}} \to \mathcal{GDT}_{\mathcal{F}^{\text{tabl,cut}}_{\mathcal{L},V}} \quad \textit{and}$$

$$\Psi^{\text{cut}}_{\mathcal{L},V} : \mathcal{GDT}_{\mathcal{F}^{\text{tabl,cut}}_{\mathcal{L},V}} \longrightarrow \mathcal{GDT}_{\mathcal{F}^{\text{seq,cut}}_{\mathcal{L},V}}$$

are inverse bijections. Further, we have $\Phi^{\text{cut}}_{\mathcal{L},V}(\mathcal{PT}_{\mathcal{F}^{\text{seq,cut}}_{\mathcal{L},V}}) = \mathcal{PT}_{\mathcal{F}^{\text{tabl,cut}}_{\mathcal{L},V}}$ and $\Psi^{\text{cut}}_{\mathcal{L},V}(\mathcal{PT}_{\mathcal{F}^{\text{tabl,cut}}_{\mathcal{L},V}}) = \mathcal{PT}_{\mathcal{F}^{\text{seq,cut}}_{\mathcal{L},V}}$.

Proof. The argument is essentially the same as the argument for Theorem 3.5.32. □

Example 5.5.34. As in Example 5.5.9, if $\mathtt{T}_0, \ldots, \mathtt{T}_{n-1}$ are $\mathcal{F}^{\text{seq,cut}}_{\mathcal{L},V}$-proof trees for the sequents $\Gamma \Rightarrow \varphi_0, \ldots, \Gamma \Rightarrow \varphi_{n-1}$, respectively, where $n \geq 1$, it is possible to construct effectively an $\mathcal{F}^{\text{seq,cut}}_{\mathcal{L},V}$-proof tree for the sequent $\Gamma \Rightarrow (\varphi_0 \wedge \cdots \wedge \varphi_{n-1})$. ▯

Theorem 5.5.35 (Soundness of $\mathcal{F}^{\text{seq,cut}}_{\mathcal{L},V}$). *Let $\mathcal{L}$ be a first-order language and let V be an $\mathcal{L}$-suitable set of variables. Every theorem of $\mathcal{F}^{\text{seq,cut}}_{\mathcal{L},V}$ is a valid sequent.*

Proof. The argument is similar to the argument of Theorem 5.5.18, using $\Phi^{cut}_{\mathcal{L},V}$ in place of $\Phi_{\mathcal{L},V}$ and Theorem 5.4.6 in place of Theorem 5.3.36. □

Theorem 5.5.36 (Partial Completeness of $\mathcal{F}^{\text{seq,cut}}_{\mathcal{L},V}$). *Let $\mathcal{L}$ be a first-order language and let V be an $\mathcal{L}$-suitable set of variables. Every valid $(\mathcal{L}, V)$-sequent is a theorem of $\mathcal{F}^{\text{seq,cut}}_{\mathcal{L}^c,V}$.*

Proof. The argument follows immediately from the partial completeness of $\mathcal{F}^{\text{seq}}_{\mathcal{L}^c,V}$. □

Theorem 5.5.37. *Let $\mathcal{L}$ be a first-order language and let V be an $\mathcal{L}$-suitable set of variables. There is an effective, syntactic construction that starts with a $\mathcal{F}^{\text{seq,cut}}_{\mathcal{L},V}$-proof tree $\mathtt{T}$ for a sequent κ and produces a $\mathcal{F}^{\text{seq,cut}}_{\mathcal{L},V}$-proof tree $\mathtt{T}'$ for a finite sequent κ' that is dominated by κ.*

Proof. The argument for this theorem is similar to that for Theorem 5.5.24, using translations between tableaux with cut and proof trees with cut for sequents and Theorem 5.4.4. □

Definition 5.5.38. Let $\mathcal{L}$ be a first-order language and let V be an $\mathcal{L}$-suitable set of variables. A general $\mathcal{F}^{\text{seq,cut}}_{\mathcal{L},V}$-deduction tree $\mathtt{T}$ is *finished* if $\Phi^{cut}_{\mathcal{L},V}(\mathtt{T})$ is a strongly completed tableau with cut. □

The notion of finished branch of a general $\mathcal{F}^{\text{seq,cut}}_{\mathcal{L},V}$-deduction tree is defined in exactly the same was as the corresponding notion for $\mathcal{F}^{\text{seq}}_{\mathcal{L},V}$-deduction trees (see Definition 5.5.27).

Theorem 5.5.39. *Let $\mathcal{L}$ be a first-order language and let V be an $\mathcal{L}$-suitable set of variables. A general $\mathcal{F}^{\text{seq,cut}}_{\mathcal{L},V}$-deduction tree $\mathtt{T}$ is finished if and only if every branch of $\mathtt{T}$ is either finished or ends with a node labeled by an axiom of $\mathcal{F}^{\text{seq,cut}}_{\mathcal{L},V}$.*

Proof. The argument is similar to that of Theorem 5.5.28. □

The existence of a finished general $\mathcal{F}^{\text{seq,cut}}_{\mathcal{L},V}$-deduction tree for every sequent κ follows from Theorem 5.5.29.

We are now ready to discuss cut elimination for sequents.

Theorem 5.5.40 (Cut Elimination for First-OrderSequents). *Let $\mathcal{L}$ be a first-order language with infinitely many constant symbols and V be an $\mathcal{L}$-suitable set of variables. There is a syntactic transformation $\mathsf{CETS}_{\mathcal{L},V}$ whose domain is $\mathcal{PT}_{\mathcal{F}^{\text{seq,cut}}_{\mathcal{L},V}}$ and whose range is $\mathcal{PT}_{\mathcal{F}^{\text{seq}}_{\mathcal{L},V}}$ such that the sequents $\mathsf{CETS}_{\mathcal{L},V}(\mathtt{T})(\lambda)$ and $\mathtt{T}(\lambda)$ are the same.*

Proof. Define $\mathsf{CETS}_{\mathcal{L},V} = \Psi_{\mathcal{L},V} \circ \mathsf{CET}_{\mathcal{L},V} \circ \Phi^{\mathrm{cut}}_{\mathcal{L},V}$. Since all three mappings involved in this definition are syntactic transformations, then so is $\mathsf{CETS}_{\mathcal{L},V}$. Let $\mathtt{T} \in \mathcal{PT}_{\mathcal{F}^{\mathrm{seq,cut}}_{\mathcal{L},V}}$. Define $\mathtt{T}' = \Phi^{\mathrm{cut}}_{\mathcal{L},V}(\mathtt{T})$, $\mathtt{T}'' = \mathsf{CET}_{\mathcal{L},V}(\mathtt{T}')$ and $\mathtt{T}_1 = \Psi_{\mathcal{L},V}(\mathtt{T}'')$. We have $\mathtt{T}' \in \mathcal{PT}_{\mathcal{F}^{\mathrm{tabl,cut}}_{\mathcal{L},V}}$ and $\mathtt{T}'(\lambda) = \mathtt{sf}_{\mathcal{L},V}(\mathtt{T}(\lambda))$. Next, $\mathtt{T}'' \in \mathcal{PT}_{\mathcal{F}^{\mathrm{tabl}}_{\mathcal{L},V}}$ and $\mathtt{T}''(\lambda) = \mathtt{T}'(\lambda)$. (See Construction 5.4.13.) Finally, $\mathtt{T}_1 \in \mathcal{PT}_{\mathcal{F}^{\mathrm{seq}}_{\mathcal{L},V}}$ and $\mathtt{T}_1(\lambda) = \mathtt{sqt}_{\mathcal{L},V}(\mathtt{T}''(\lambda)) = \mathtt{sqt}_{\mathcal{L},V}(\mathtt{sf}_{\mathcal{L},V}(\mathtt{T}(\lambda))) = \mathtt{T}(\lambda)$. □

Theorem 5.5.41. *Let $\mathcal{L}$ be a first-order language and let V be an $\mathcal{L}$-suitable set of variables. There is an effective, syntactic construction that, given $\mathcal{F}^{\mathrm{seq,cut}}_{\mathcal{L},V}$-proofs for $\Gamma \Rightarrow \varphi$ and $\varphi \Rightarrow \Gamma'$, where Γ, Γ' are sets of $(\mathcal{L}, V)$-formulas and φ is an $(\mathcal{L}, V)$-formula, produces a proof of the sequent $\Gamma \Rightarrow \Gamma'$ in $\mathcal{F}^{\mathrm{seq,cut}}_{\mathcal{L},V}$.*

Proof. Let r_0, r_0' be proofs of $\Gamma \Rightarrow \varphi$ and $\varphi \Rightarrow \Gamma'$, respectively. Define the proofs $r_1 = r_0(\Gamma \Rightarrow \Gamma', \varphi)$ and $r_1' = r_0'(\Gamma, \varphi \Rightarrow \Gamma')$ obtained from r_0 and r_0', respectively, by applying thinning. Then, the sequence $r_1 r_1'(\Gamma \Rightarrow \Gamma')$ is an $\mathcal{F}^{\mathrm{seq,cut}}_{\mathcal{L},V}$-proof where we apply the cut rule at the last step. □

Theorem 5.5.42. *Let $\mathcal{L}$ be a first-order language, V be an $\mathcal{L}$-suitable set of variables, $\mathtt{T}$ be an $\mathcal{F}^{\mathrm{seq,cut}}_{\mathcal{L},V}$-deduction tree, x be a variable and c be a constant symbol of $\mathcal{L}$ that is not an eigenconstant of $\mathtt{T}$. Define a tree $\mathtt{T}'$ by $\mathrm{Dom}(\mathtt{T}') = \mathrm{Dom}(\mathtt{T})$ and $\mathtt{T}'(q) = (\mathtt{T}(q))_{x:=c}$, for $q \in \mathrm{Dom}(\mathtt{T})$. Then, $\mathtt{T}'$ is an $\mathcal{F}^{\mathrm{seq,cut}}_{\mathcal{L},V}$-deduction tree.*

Proof. This result follows immediately from Theorems 5.5.33 and 5.4.15 and Equation (5.3) on page 937. □

Corollary 5.5.43. *Let $\mathcal{L}$ be a first-order language, V be an $\mathcal{L}$-suitable set of variables, $\mathtt{T}$ be an $\mathcal{F}^{\mathrm{seq,cut}}_{\mathcal{L},V}$-proof tree, x be a variable and c be a constant symbol of $\mathcal{L}$ that is not an eigenconstant of $\mathtt{T}$. Define a tree $\mathtt{T}'$ by $\mathrm{Dom}(\mathtt{T}') = \mathrm{Dom}(\mathtt{T})$ and $\mathtt{T}'(q) = (\mathtt{T}(q))_{x:=c}$, for $q \in \mathrm{Dom}(\mathtt{T})$. Then, $\mathtt{T}'$ is an $\mathcal{F}^{\mathrm{seq,cut}}_{\mathcal{L},V}$-proof tree.*

Proof. This follows immediately from Theorem 5.5.42. □

5.6 First-Order Natural Deduction

We will extend the natural deduction introduced in propositional logic to include quantified first-order formulas.

The structure of the system $\mathcal{F}^{\mathcal{L}}_{\text{fond}}$ is informed by the construction of the formal system $\mathcal{F}_{\text{nd}}$ introduced in propositional logic. The objects of the new system are $\mathcal{L}$-formulas. If $\mathcal{L}$ does not contain equality, then there are no axioms in $\mathcal{F}^{\mathcal{L}}_{\text{fond}}$. Otherwise, $\mathcal{F}^{\mathcal{L}}_{\text{fond}}$ has the axioms $t = t$ for all $\mathcal{L}$-terms t.

Definition 5.6.1. Let $\mathcal{L}$ be a first-order language. $\mathcal{F}^{\mathcal{L}}_{\text{fond}}$ is the formal system $(\text{FORM}_{\mathcal{L}}, A, I)$, where

$$A = \begin{cases} \emptyset & \text{if } = \notin \mathcal{L} \\ \{t = t \mid t \in \text{TERM}_{\mathcal{L}}\} & \text{otherwise.} \end{cases}$$

The set of rules I when $= \notin \mathcal{L}$ is given in Figure 5.13, for all $\mathcal{L}$-formulas φ, ψ, θ, $\mathcal{L}$-constant symbols c, and $\mathcal{L}$-terms t.

If $= \in \mathcal{L}$, we supplement I with the rules contained in Figure 5.14. In this figure, the t_is are $\mathcal{L}$-terms, f is a function symbol of $\mathcal{L}$ of positive arity and P is a relation symbol of positive arity in the same language.

If $\mathtt{T}$ is an $\mathcal{F}^{\mathcal{L}}_{\text{fond}}$-deduction tree for φ, we will refer to the labels of the leaves of $\mathtt{T}$ as the *hypotheses* of $\mathtt{T}$ and to φ as the *conclusion* of $\mathtt{T}$. □

As in propositional logic, rules indexed by I and E are referred to as *introduction rules* and *elimination rules*, respectively.

This system is not be sound because, unlike the propositional case, there is a rule $(\mathsf{R}_{\forall,I})$ such that the premise of the rule does not necessarily logically imply the conclusion of the rule. For example, for the instance

$$\frac{R(x)}{(\forall x)R(x)}$$

of this rule, the premise clearly does not logically imply the conclusion. However, by imposing appropriate cancellation rules, we will be able to overcome this difficulty.

Definition 5.6.2. A *marked* $\mathcal{F}^{\mathcal{L}}_{\text{fond}}$*-deduction tree* is a pair $(\mathtt{T}, M)$ where $\mathtt{T}$ is an $\mathcal{F}^{\mathcal{L}}_{\text{fond}}$-deduction tree and $M \subseteq \text{LEAVES}(\mathtt{T})$. The leaves that belong to the set M are said to be *cancelled*.

$\wedge$-rules:

$$\frac{\varphi, \psi}{(\varphi \wedge \psi)}\mathsf{R}_{\wedge I} \qquad \frac{(\varphi \wedge \psi)}{\varphi}\mathsf{R}_{\wedge El} \qquad \frac{(\varphi \wedge \psi)}{\psi}\mathsf{R}_{\wedge Er}$$

$\vee$-rules:

$$\frac{\varphi}{(\varphi \vee \psi)}\mathsf{R}_{\vee Il} \qquad \frac{\psi}{(\varphi \vee \psi)}\mathsf{R}_{\vee Ir} \qquad \frac{(\varphi \vee \psi), \theta, \theta}{\theta}\mathsf{R}_{\vee E}$$

$\rightarrow$-rules:

$$\frac{\psi}{(\varphi \rightarrow \psi)}\mathsf{R}_{\rightarrow I} \qquad \frac{\varphi, (\varphi \rightarrow \psi)}{\psi}\mathsf{R}_{\rightarrow E}$$

$\leftrightarrow$-rules:

$$\frac{\psi, \varphi}{(\varphi \leftrightarrow \psi)}\mathsf{R}_{\leftrightarrow I} \qquad \frac{\varphi, (\varphi \leftrightarrow \psi)}{\psi}\mathsf{R}_{\leftrightarrow El} \qquad \frac{\psi, (\varphi \leftrightarrow \psi)}{\varphi}\mathsf{R}_{\leftrightarrow Er}$$

$\neg$-rules:

$$\frac{\psi, (\neg\psi)}{(\neg\varphi)}\mathsf{R}_{\neg I} \qquad \frac{\psi, (\neg\psi)}{\varphi}\mathsf{R}_{\neg E}$$

$\forall$-rules:

$$\frac{\varphi}{(\forall x)\varphi}\mathsf{R}_{\forall I} \qquad \frac{(\forall x)\varphi}{\langle\varphi\rangle_{x:=t}}\mathsf{R}_{\forall E}$$

$\exists$-rules:

$$\frac{\langle\varphi\rangle_{x:=t}}{(\exists x)\varphi}\mathsf{R}_{\exists I} \qquad \frac{(\exists x)\varphi, \psi}{\psi}\mathsf{R}_{\exists E}$$

Fig. 5.13. The set of rules of $\mathcal{F}^{\mathcal{L}}_{\text{fond}}$.

The set of *uncancelled hypotheses* of a marked $\mathcal{F}^{\mathcal{L}}_{\text{fond}}$-deduction tree $(\mathtt{T}, M)$ is the set

$$\mathcal{UNC}(\mathtt{T}, M) = \{\mathtt{T}(q) \mid q \in \text{LEAVES}(\mathtt{T}) - M\}.$$

$\square$

In the next definition, we denote the one-node lot whose root is labelled o as $\mathtt{T}_o$ and the one-node marked lot $(\mathtt{T}_o, \emptyset)$ by $\mathcal{T}_o$. As in propositional logic, we denote by $L_\varphi(\mathcal{T})$ the marked $\mathcal{F}^{\mathcal{L}}_{\text{fond}}$-deduction

Symmetry rule:

$$\frac{t_0 = t_1}{t_1 = t_0}\,\mathsf{R}_{sym}$$

Transitivity rule:

$$\frac{t_0 = t_1, t_1 = t_2}{t_0 = t_2}\,\mathsf{R}_{tran}$$

f-rule:

$$\frac{t_0 = t_n, \dots, t_{n-1} = t_{2n-1}}{f(t_0, \dots, t_{n-1}) = f(t_n, \dots, t_{2n-1})}\,\mathsf{R}_f$$

P-rule:

$$\frac{t_0 = t_n, \dots, t_{n-1} = t_{2n-1}, P(t_0, \dots, t_{n-1})}{P(t_n, \dots, t_{2n-1})}\,\mathsf{R}_P$$

Fig. 5.14. The additional equality rules of $\mathcal{F}^{\mathcal{L}}_{\text{fond}}$.

tree obtained from $\mathtt{T}$ by cancelling the uncancelled leaves labelled by φ.

Definition 5.6.3. The set FONDT$^{\mathcal{L}}$ of *natural deduction trees of the first-order language* $\mathcal{L}$ is the set of marked $\mathcal{F}^{\mathcal{L}}_{\text{fond}}$-deduction trees given inductively by the following:

(1) If $\mathtt{T}$ is a one-node $\mathcal{F}^{\mathcal{L}}_{\text{fond}}$-deduction tree, then $(\mathtt{T}, \emptyset) \in \text{FONDT}^{\mathcal{L}}$.

(2) ($\to$-introduction) If $\mathcal{T} = (\mathtt{T}, M) \in \text{FONDT}^{\mathcal{L}}$, $\mathtt{T}(\lambda) = \psi$, and φ is an $\mathcal{L}$-formula, then

$$(L_\varphi(\mathcal{T}); (\varphi \to \psi)) \in \text{FONDT}^{\mathcal{L}}.$$

(3) ($\vee$-elimination) If $\mathcal{T}_i = (\mathtt{T}_i, M_i) \in \text{FONDT}^{\mathcal{L}}$ for $0 \leq i \leq 2$ and $\mathtt{T}_0(\lambda) = (\varphi \vee \psi)$, $\mathtt{T}_1(\lambda) = \mathtt{T}_2(\lambda) = \theta$, then

$$(\mathcal{T}_0, L_\varphi(\mathcal{T}_1), L_\psi(\mathcal{T}_2); \theta)$$

belongs to FONDT$^{\mathcal{L}}$.

(4) ($\leftrightarrow$-introduction) If $\mathcal{T}_0 = (\mathtt{T}_0, M_0)$ and $\mathcal{T}_1 = (\mathtt{T}_1, M_1)$ belong to FONDT$^{\mathcal{L}}$, $\mathtt{T}_0(\lambda) = \psi$, and $\mathtt{T}_1(\lambda) = \varphi$, then

$$(L_\varphi(\mathcal{T}_0), L_\psi(\mathcal{T}_1); (\varphi \leftrightarrow \psi))$$

belongs to FONDT$^{\mathcal{L}}$.

(5) ($\neg$-introduction) If $\mathcal{T}_0, \mathcal{T}_1 \in \text{FONDT}^{\mathcal{L}}$, where $\mathcal{T}_0 = (\mathtt{T}_0, M_0)$, $\mathcal{T}_1 = (\mathtt{T}_1, M_1)$, $\mathtt{T}_0(\lambda) = \psi$, $\mathtt{T}_1(\lambda) = (\neg\psi)$ and φ is an $\mathcal{L}$-formula, then

$$(L_\varphi(\mathcal{T}_0), L_\varphi(\mathcal{T}_1); (\neg\varphi)) \in \text{FONDT}^{\mathcal{L}}.$$

(6) ($\neg$-elimination) If $\mathcal{T}_0 = (\mathtt{T}_0, M_0)$ and $\mathcal{T}_1 = (\mathtt{T}_1, M_1)$ belong $\mathrm{FONDT}^{\mathcal{L}}$, $\mathtt{T}_0(\lambda) = \psi$, $\mathtt{T}_1(\lambda) = (\neg\psi)$ and φ is an $\mathcal{L}$-formula, then

$$(L_{(\neg\varphi)}(\mathcal{T}_0), L_{(\neg\varphi)}(\mathcal{T}_1); \varphi) \in \mathrm{FONDT}^{\mathcal{L}}.$$

(7) ($\forall$-introduction) If $\mathcal{T} = (\mathtt{T}, M) \in \mathrm{FONDT}^{\mathcal{L}}$, $\mathtt{T}(\lambda) = \varphi$, where x does not occur free in $\mathcal{UNC}(\mathcal{T})$, then $(\mathcal{T}; (\forall x)\varphi) \in \mathrm{FONDT}^{\mathcal{L}}$.
(8) ($\exists$-elimination) If $\mathcal{T}_0 = (\mathtt{T}_0, M_0), \mathcal{T}_1 = (\mathtt{T}_1, M_1) \in \mathrm{FONDT}^{\mathcal{L}}$, $\mathtt{T}_0(\lambda) = (\exists x)\varphi$, $\mathtt{T}_1(\lambda) = \psi$ and $x \notin \mathtt{FV}(\psi) \cup \mathtt{FV}(\mathcal{UNC}(\mathcal{T}_1) - \{\varphi\})$, then we have $(\mathcal{T}_0, L_{\varphi}(\mathcal{T}_1); \psi) \in \mathrm{FONDT}^{\mathcal{L}}$.
(9) If $\frac{\alpha}{\beta}$ is an instance of one of the remaining rules that have one hypothesis (except rules involving the equality symbol) and $\mathcal{T} = (\mathtt{T}, M) \in \mathrm{FONDT}^{\mathcal{L}}$ with $\mathtt{T}(\lambda) = \alpha$, then $(\mathcal{T}; \beta) \in \mathrm{FONDT}^{\mathcal{L}}$.
(10) If $\dfrac{\alpha_0, \alpha_1}{\beta}$ is an instance of one of the remaining rules that have two hypotheses (except rules involving the equality symbol), and $\mathcal{T}_0 = (\mathtt{T}_0, M_0)$ and $\mathcal{T}_1 = (\mathtt{T}_1, M_1)$ belong to $\mathrm{FONDT}^{\mathcal{L}}$ with $\mathtt{T}_0(\lambda) = \alpha_0$, and $\mathtt{T}_1(\lambda) = \alpha_1$ then $(\mathcal{T}_0, \mathcal{T}_1; \beta) \in \mathrm{FONDT}^{\mathcal{L}}$.

In addition, if $\mathcal{L}$ contains the equality symbol, we extend the definition of natural deduction trees by adding the following parts:

(11) For every $\mathcal{L}$-term t, the marked lot $(\mathtt{T}_{t=t}, \{\lambda\}) \in \mathrm{FONDT}^{\mathcal{L}}$.
(12) (=-symmetry) For all $\mathcal{L}$-terms t_0, t_1, if $\mathcal{T} = (\mathtt{T}, M) \in \mathrm{FONDT}^{\mathcal{L}}$ with $\mathtt{T}(\lambda) = t_0 = t_1$, then $(\mathcal{T}; t_1 = t_0) \in \mathrm{FONDT}^{\mathcal{L}}$.
(13) (=-transitivity) For all $\mathcal{L}$-terms t_0, t_1, t_2 and natural deduction trees $\mathcal{T}_0 = (\mathtt{T}_0, M_0), \mathcal{T}_1 = (\mathtt{T}_1, M_1)$, with $\mathtt{T}_0(\lambda) = t_0 = t_1$ and $\mathtt{T}_1(\lambda) = t_1 = t_2$, we have $(\mathcal{T}_0, \mathcal{T}_1; t_0 = t_2) \in \mathrm{FONDT}^{\mathcal{L}}$.
(14) (f-congruency) If $t_0, \ldots, t_{2n-1}$ are $\mathcal{L}$-terms, f is an n-ary function symbol of $\mathcal{L}$ with $n > 0$, and $\mathcal{T}_i = (\mathtt{T}_i, M_i)$ in $\mathrm{FONDT}^{\mathcal{L}}$ is such that $\mathtt{T}_i(\lambda) = t_i = t_{i+n}$ for $0 \leq i \leq n-1$, then

$$(\mathcal{T}_0, \ldots, \mathcal{T}_{n-1}; f(t_0, \ldots, t_{n-1}) = f(t_n, \ldots, t_{2n-1})) \in \mathrm{FONDT}^{\mathcal{L}}.$$

(15) (R-congruency) If $t_0, \ldots, t_{2n-1}$ are $\mathcal{L}$-terms, R is an n-ary relation symbol of $\mathcal{L}$ with $n > 0$, $\mathcal{T}_i = (\mathtt{T}_i, M_i)$ in $\mathrm{FONDT}^{\mathcal{L}}$ is such that $\mathtt{T}_i(\lambda) = t_i = t_{i+n}$ for $0 \leq i \leq n-1$, and

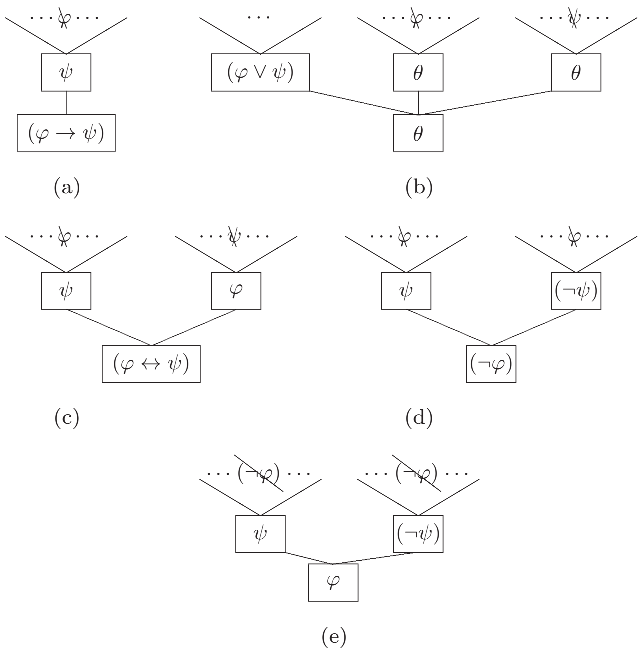


Fig. 5.15. Application of Connective Rules in Natural Deduction

for $\mathcal{T}_n = (\mathtt{T}_n, M_n)$, we have $\mathtt{T}_n(\lambda) = R(t_0, \ldots, t_{n-1})$, then $(\mathcal{T}_0, \ldots, \mathcal{T}_n; R(t_n, \ldots, t_{2n-1}))$ belongs to $\mathrm{FONDT}^{\mathcal{L}}$.

$\square$

Figure 5.15 illustrates the inductive parts of Definition 5.6.3 that involve $\rightarrow$-introduction, $\vee$-elimination, $\leftrightarrow$-introduction, $\neg$-introduction, and $\neg$-elimination rules. In Figure 5.16(a)-(d) we show the constructions involving quantifiers, while in Figure 5.16(e)-(f) we show the application of the remaining inductive parts that have one or two hypotheses, respectively. Finally, Figure 5.17 illustrates the parts of Definition 5.6.3 that involve the equality symbol.

The $\forall$-introduction part of Definition 5.6.3 is justified by Theorem 4.5.61. The $\exists$-elimination part is explained by the following theorem.

Theorem 5.6.4. *Let $\mathcal{L}$ be a first-order language, Γ_0, Γ_1 be two sets of $\mathcal{L}$-formulas and φ, ψ be formulas such that $x \notin \mathtt{FV}(\psi) \cup \mathtt{FV}(\Gamma_1 - \{\varphi\})$ for a variable x. If $\Gamma_0 \models (\exists x)\varphi$ and $\Gamma_1 \models \psi$, then $\Gamma_0 \cup (\Gamma_1 - \{\varphi\}) \models \psi$.*

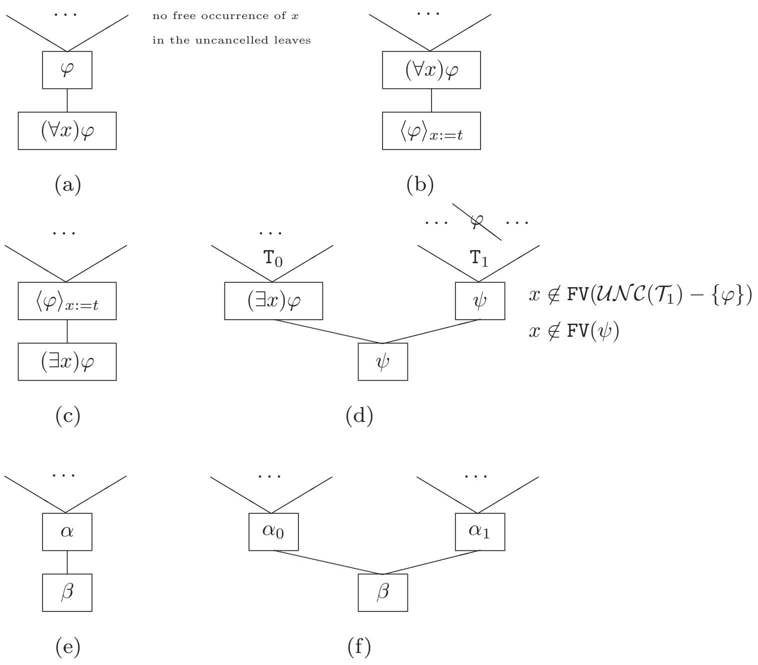


Fig. 5.16. Application of Quantifier Rules in Natural Deduction (a)–(d) and of the Remaining Inductive Parts That Have One or Two Hypotheses (e)–(f)

Proof. Suppose that $(\mathcal{A}, \sigma) \models \Gamma_0 \cup (\Gamma_1 - \{\varphi\})$. Then, $(\mathcal{A}, \sigma) \models (\exists x)\varphi$, so, for some $a \in |\mathcal{A}|$, $(\mathcal{A}, [x \to a]\sigma) \models \varphi$. Since x does not occur free in $\Gamma_1 - \{\varphi\}$, we have $(\mathcal{A}, [x \to a]\sigma) \models \Gamma_1 - \{\varphi\}$, so $(\mathcal{A}, [x \to a]\sigma) \models \Gamma_1$. Therefore, $(\mathcal{A}, [x \to a]\sigma) \models \psi$. Since x does not occur free in ψ, we have $(\mathcal{A}, \sigma) \models \psi$. □

Definition 5.6.5. Let $\mathcal{L}$ be a first-order language, Γ be a set of $\mathcal{L}$-formulas and let φ be an $\mathcal{L}$-formula. We write $\Gamma \overset{\bullet}{\vdash}_{\mathrm{fond}^{\mathcal{L}}} \varphi$ if there is a first-order natural deduction tree $\mathcal{T} = (\mathtt{T}, M)$ of $\mathcal{L}$ such that $\mathtt{T}(\lambda) = \varphi$ and $\mathcal{UNC}(\mathtt{T}, M) \subseteq \Gamma$.

If $\mathcal{UNC}(\mathtt{T}, M) = \emptyset$ we refer to $\mathcal{T}$ as a *natural deduction tree for* φ. ▯

As was the case with propositional logic, first-order natural deduction is not an analytical formalism. Drawings of first-order natural deduction trees make use of the same graphical conventions as the corresponding drawings of propositional logic.

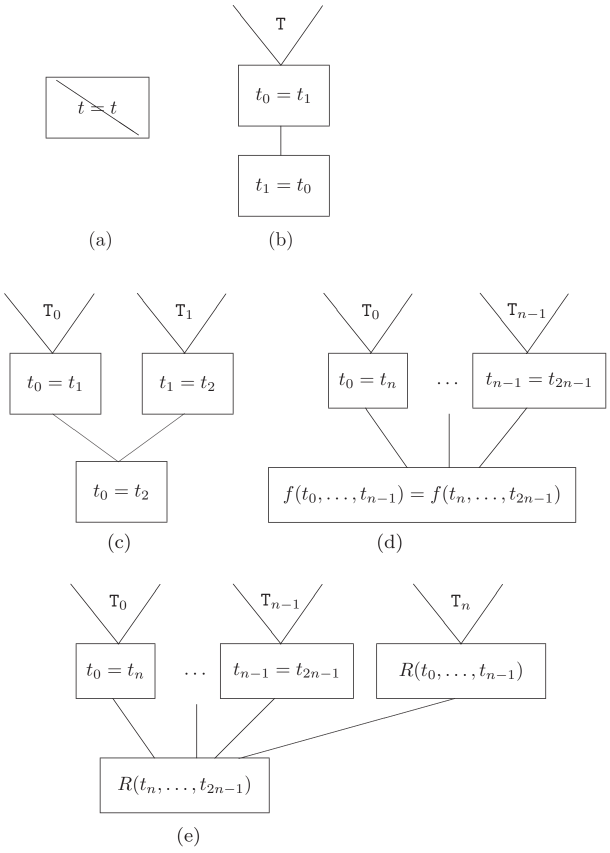


Fig. 5.17. Natural Deduction Trees with Equality

Example 5.6.6. The trees shown in Figures 3.17 and 3.18 remain first-order natural deduction trees for the formulas $(\varphi \vee (\neg\varphi))$ and $((\varphi \vee (\alpha \wedge \beta)) \to ((\varphi \vee \alpha) \wedge (\varphi \vee \beta)))$, respectively, where φ, α, β are $\mathcal{L}$-formulas. Thus, we have $\emptyset \overset{\bullet}{\vdash}_{\text{fond}^{\mathcal{L}}} (\varphi \vee (\neg\varphi))$ and $\emptyset \overset{\bullet}{\vdash}_{\text{fond}^{\mathcal{L}}} ((\varphi \vee (\alpha \wedge \beta)) \to ((\varphi \vee \alpha) \wedge (\varphi \vee \beta)))$. ▯

Example 5.6.7. Let $\mathcal{L}$ be a first-order language and let φ be an $\mathcal{L}$-formula. A natural deduction tree for the $\mathcal{L}$-formula $((\exists x)(\forall y)\varphi \to (\forall y)(\exists x)\varphi)$ is given in Figure 5.18.

Since all the leaves of this tree are cancelled, we have

$$\emptyset \overset{\bullet}{\vdash}_{\text{fond}^{\mathcal{L}}} ((\exists x)(\forall y)\varphi \to (\forall y)(\exists x)\varphi).$$

▯

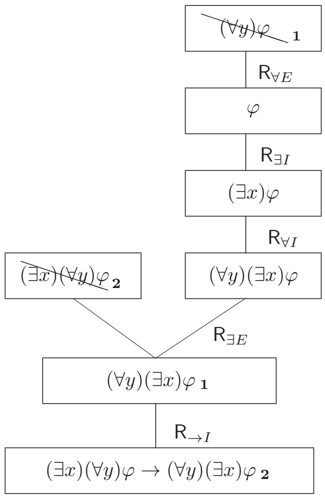

$\cancel{(\forall y)\varphi}$ 1
$R_{\forall E}$
φ
$R_{\exists I}$
$(\exists x)\varphi$
$R_{\forall I}$
$\cancel{(\exists x)(\forall y)\varphi}$ 2
$(\forall y)(\exists x)\varphi$
$R_{\exists E}$
$(\forall y)(\exists x)\varphi$ 1
$R_{\rightarrow I}$
$(\exists x)(\forall y)\varphi \rightarrow (\forall y)(\exists x)\varphi$ 2

Fig. 5.18. Natural Deduction Tree for $((\exists x)(\forall y)\varphi \rightarrow (\forall y)(\exists x)\varphi)$

Example 5.6.8. Let $\mathcal{L}$ be a first-order language and let R be a unary relation symbol in $\mathcal{L}$. Figure 5.19 contains a natural deduction tree for the $\mathcal{L}$-formula $(\exists x)(R(x) \rightarrow (\forall x)R(x))$. In this figure, $\mathtt{T}_0$ is the natural deduction tree of Figure 3.17 with $\varphi = (\exists x)(\neg R(x))$. Since all the leaves of the full tree are cancelled, we have $\emptyset \overset{\bullet}{\vdash}_{\mathrm{fond}^{\mathcal{L}}} (\exists x)(R(x) \rightarrow (\forall x)R(x))$. □

Example 5.6.9. The existence of the natural deduction tree shown in Figure 5.20 proves that $\emptyset \overset{\bullet}{\vdash}_{\mathrm{fond}^{\mathcal{L}}} ((\forall x)\varphi \rightarrow (\exists x)\varphi))$, for every $\mathcal{L}$-formula φ and variable x.

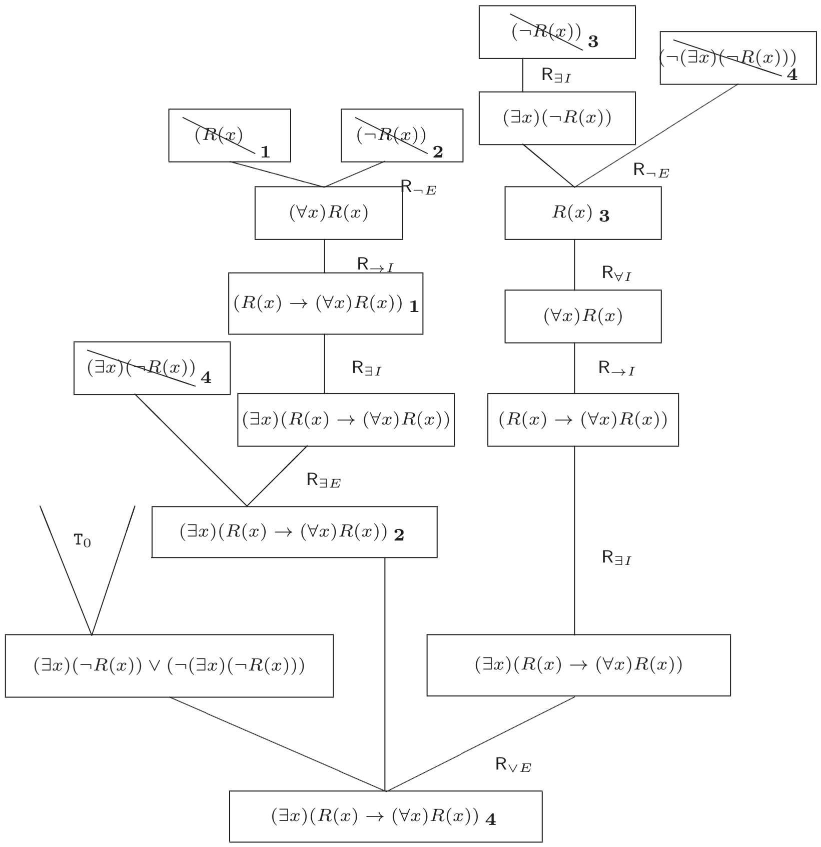
□

Example 5.6.10. Let $\mathcal{L}$ be a first-order language that includes a binary relation symbol R, and two unary function symbols f and g, and let θ_0 and θ_1 be the formulas defined as

$$\theta_0 = ((\forall x_0)(\forall x_1)R(x_0, x_1) \rightarrow R(f(x_0), g(x_1))) \text{ and}$$
$$\theta_1 = ((\forall x_0)(\forall x_1)R(x_0, x_1) \rightarrow R(f(x_1), g(x_0))).$$

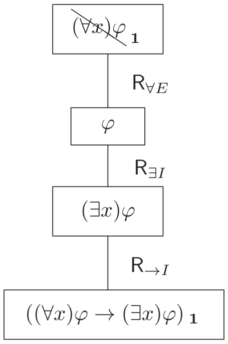
The natural deduction tree in Figure 5.21(a) proves that $\emptyset \overset{\bullet}{\vdash}_{\mathrm{fond}^{\mathcal{L}}} \theta_0$. Figure 5.21(b) shows the related natural deduction tree which proves that $\emptyset \overset{\bullet}{\vdash}_{\mathrm{fond}^{\mathcal{L}}} \theta_1$.

□



$\cancel{(\neg R(x))}$ 3
$\cancel{(\neg(\exists x)(\neg R(x)))}$ 4
$\mathsf{R}_{\exists I}$
$(\exists x)(\neg R(x))$
$\cancel{(R(x)}$ 1
$\cancel{(\neg R(x))}$ 2
$\mathsf{R}_{\neg E}$
$\mathsf{R}_{\neg E}$
$(\forall x)R(x)$
$R(x)$ 3
$\mathsf{R}_{\rightarrow I}$
$\mathsf{R}_{\forall I}$
$(R(x) \rightarrow (\forall x)R(x))$ 1
$(\forall x)R(x)$
$\cancel{(\exists x)(\neg R(x))}$ 4
$\mathsf{R}_{\exists I}$
$\mathsf{R}_{\rightarrow I}$
$(\exists x)(R(x) \rightarrow (\forall x)R(x))$
$(R(x) \rightarrow (\forall x)R(x))$
$\mathsf{R}_{\exists E}$
$\mathtt{T}_0$
$(\exists x)(R(x) \rightarrow (\forall x)R(x))$ 2
$\mathsf{R}_{\exists I}$
$(\exists x)(\neg R(x)) \vee (\neg(\exists x)(\neg R(x)))$
$(\exists x)(R(x) \rightarrow (\forall x)R(x))$
$\mathsf{R}_{\vee E}$
$(\exists x)(R(x) \rightarrow (\forall x)R(x))$ 4

Fig. 5.19. A Natural Deduction Tree for $(\exists x)(R(x) \rightarrow (\forall x)R(x))$



$\cancel{(\forall x)\varphi}$ 1
$\mathsf{R}_{\forall E}$
φ
$\mathsf{R}_{\exists I}$
$(\exists x)\varphi$
$\mathsf{R}_{\rightarrow I}$
$((\forall x)\varphi \rightarrow (\exists x)\varphi)$ 1

Fig. 5.20. Natural Deduction Tree for $((\forall x)\varphi \rightarrow (\exists x)\varphi)$

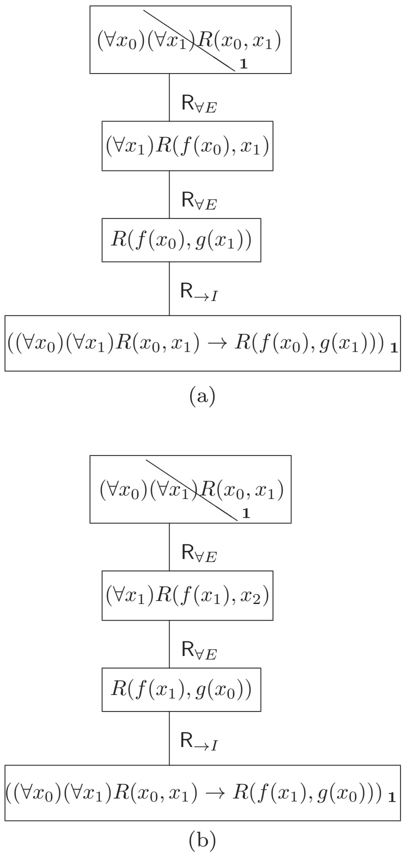

$\cancel{(\forall x_0)(\forall x_1)R(x_0, x_1)}$ 1
$\mathsf{R}_{\forall E}$
$(\forall x_1)R(f(x_0), x_1)$
$\mathsf{R}_{\forall E}$
$R(f(x_0), g(x_1))$
$\mathsf{R}_{\to I}$
$((\forall x_0)(\forall x_1)R(x_0, x_1) \to R(f(x_0), g(x_1)))$ 1
(a)
$\cancel{(\forall x_0)(\forall x_1)R(x_0, x_1)}$ 1
$\mathsf{R}_{\forall E}$
$(\forall x_1)R(f(x_1), x_2)$
$\mathsf{R}_{\forall E}$
$R(f(x_1), g(x_0))$
$\mathsf{R}_{\to I}$
$((\forall x_0)(\forall x_1)R(x_0, x_1) \to R(f(x_1), g(x_0)))$ 1
(b)

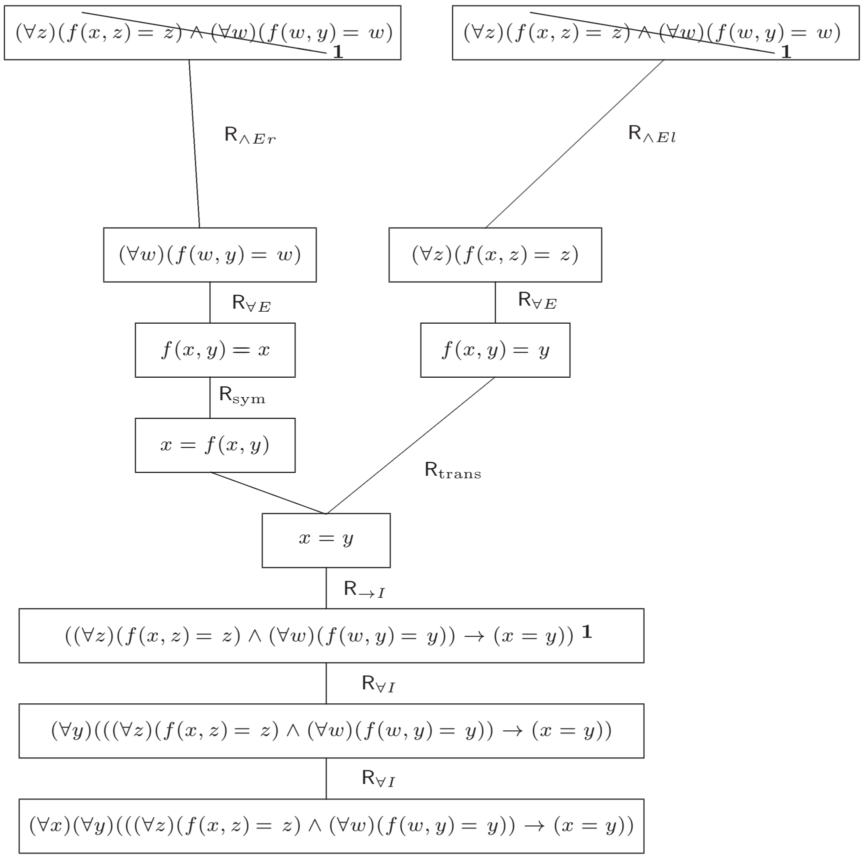
Fig. 5.21. Natural Deduction Trees for Formulas θ_0 and θ_1

Example 5.6.11. Let f be a binary function symbol of a first-order language $\mathcal{L}$. Figure 5.22 contains a natural deduction tree for the $\mathcal{L}$-formula

$$\theta = (\forall x)(\forall y)((((\forall z)(f(x, z) = z) \wedge (\forall w)(f(w, y) = w)) \to (x = y)).$$

Since all the leaves of the full tree are cancelled, we have $\emptyset \overset{\bullet}{\vdash}_{\mathrm{fond}^{\mathcal{L}}} \theta$. $\square$

Theorem 5.6.12 (Soundness of First-Order Natural Deduction). *Let $\mathcal{L}$ be a first-order language, Γ be a set of $\mathcal{L}$-formulas and let θ be an $\mathcal{L}$-formula. If $\Gamma \overset{\bullet}{\vdash}_{\mathrm{fond}^{\mathcal{L}}} \theta$, then $\Gamma \models \theta$.*



$(\forall z)(f(x,z)=z)\wedge(\forall w)(f(w,y)=w)$ 1
$(\forall z)(f(x,z)=z)\wedge(\forall w)(f(w,y)=w)$ 1
$\mathsf{R}_{\wedge Er}$
$\mathsf{R}_{\wedge El}$
$(\forall w)(f(w,y)=w)$
$(\forall z)(f(x,z)=z)$
$\mathsf{R}_{\forall E}$
$\mathsf{R}_{\forall E}$
$f(x,y)=x$
$f(x,y)=y$
R_{sym}
$x=f(x,y)$
$\mathsf{R}_{\text{trans}}$
$x=y$
$\mathsf{R}_{\to I}$
$((\forall z)(f(x,z)=z)\wedge(\forall w)(f(w,y)=y))\to(x=y))$ 1
$\mathsf{R}_{\forall I}$
$(\forall y)(((\forall z)(f(x,z)=z)\wedge(\forall w)(f(w,y)=y))\to(x=y))$
$\mathsf{R}_{\forall I}$
$(\forall x)(\forall y)(((\forall z)(f(x,z)=z)\wedge(\forall w)(f(w,y)=y))\to(x=y))$

Fig. 5.22. A Natural Deduction Tree for Formula θ of Example 5.6.11

Proof. We must show that if $(\mathtt{T}, M)$ is a natural deduction tree of $\mathcal{L}$ such that $\mathcal{UNC}(\mathtt{T}, M) \subseteq \Gamma$ and $\mathtt{T}(\lambda) = \theta$, then $\Gamma \models \theta$. We proceed by induction on the definition of natural deduction trees (Definition 5.6.3).

For the first basis step, $\mathtt{T}$ is a one-node $\mathcal{F}^{\mathcal{L}}_{\text{fond}}$-deduction tree and $M = \emptyset$. Since $\mathcal{UNC}(\mathtt{T}, \emptyset) = \{\mathtt{T}(\lambda)\}$ and the result is immediate. If $= \in \mathcal{L}$, the remaining basis step is immediate by Corollary 4.6.10.

Most of the inductive steps are formally identical to the inductive steps used in the proof of Theorem 3.6.11. We discuss here only the inductive steps related to quantifiers and equality.

Suppose that $(\mathtt{T}, M)$ is obtained from $(\mathtt{T}_0, M_0)$ using the $\forall$-introduction rule (Case 7 of Definition 5.6.3). Then, there is an

$\mathcal{L}$-formula φ and a variable x such that $\theta = \mathtt{T}(\lambda) = (\forall x)\varphi$, $\mathtt{T}_0(\lambda) = \varphi$ and x does not occur free in $\mathcal{UNC}(\mathtt{T}_0, M_0)$. We have $\mathcal{UNC}(\mathtt{T}_0, M_0) \subseteq \Gamma$ because $\mathcal{UNC}(\mathtt{T}_0, M_0) = \mathcal{UNC}(\mathtt{T}, M) \subseteq \Gamma$. By the inductive hypothesis, we have $\mathcal{UNC}(\mathtt{T}, M) = \mathcal{UNC}(\mathtt{T}_0, M_0) \models \varphi$, so, $\mathcal{UNC}(\mathtt{T}, M) \models (\forall x)\varphi$ by Theorem 4.5.61. Since, $\mathcal{UNC}(\mathtt{T}, M) \subseteq \Gamma$, we have $\Gamma \models (\forall x)\varphi$.

Next, suppose that $(\mathtt{T}, M)$ is obtained from $(\mathtt{T}_0, M_0), (\mathtt{T}_1, M_1)$ using the $\exists$-elimination rule (Case 8 of Definition 5.6.3). Then, there are $\mathcal{L}$-formulas φ, ψ and a variable x such that $\mathtt{T}_0(\lambda) = (\exists x)\varphi$, $\theta = \mathtt{T}(\lambda) = \mathtt{T}_1(\lambda) = \psi$, and x does not occur free in $\mathcal{UNC}(\mathtt{T}_1, M_1) - \{\varphi\}$ or in ψ. We have $\mathcal{UNC}(\mathtt{T}, M) = \mathcal{UNC}(\mathtt{T}_0, M_0) \cup (\mathcal{UNC}(\mathtt{T}_1, M_1) - \{\varphi\}) \subseteq \Gamma$. By the inductive hypothesis, we have $\mathcal{UNC}(\mathtt{T}_0, M_0) \models (\exists x)\varphi$ and $\mathcal{UNC}(\mathtt{T}_1, M_1) \models \psi$ so, $\mathcal{UNC}(\mathtt{T}, M) = \mathcal{UNC}(\mathtt{T}_0, M_0) \cup (\mathcal{UNC}(\mathtt{T}_1, M_1) - \{\varphi\}) \models \psi$ by Theorem 5.6.4, which implies $\Gamma \models \psi$.

The remaining two cases involving quantifiers are immediate by observing that, by Theorem 4.6.51, $(\forall x)\varphi \models \langle\varphi\rangle_{x:=t}$ and $\langle\varphi\rangle_{x:=t} \models (\exists x)\varphi$.

If $= \in \mathcal{L}$, we have four types of additional inductive steps corresponding to the trees of Figure 5.17 (b)–(e). We discuss here only the case of the tree in Part (d). Suppose that f is an n-ary function symbol of $\mathcal{L}$, where $n > 0$, and $(\mathtt{T}, M)$ is obtained from $(\mathtt{T}_0, M_0), (\mathtt{T}_1, M_1), \ldots, (\mathtt{T}_{n-1}, M_{n-1})$ by an application of the rule R_f. Then, there are $\mathcal{L}$-terms $t_0, \ldots, t_{2n-1}$ such that $\mathtt{T}_i(\lambda) = t_i = t_{n+i}$ for $0 \leq i \leq n-1$ and

$$\mathtt{T} = (\mathtt{T}_0, \ldots, \mathtt{T}_{n-1}; f(t_0, \ldots, t_{n-1}) = f(t_n, \ldots, t_{2n-1})).$$

We have $\mathcal{UNC}(\mathtt{T}, M) = \bigcup_{i=0}^{n-1} \mathcal{UNC}(\mathtt{T}_i, M_i)$. Thus,

$$\mathcal{UNC}(\mathtt{T}_i, M_i) \subseteq \mathcal{UNC}(\mathtt{T}, M) \subseteq \Gamma,$$

which, by inductive hypotheses, implies that $\Gamma \models t_i = t_{n+i}$ for $0 \leq i \leq n-1$. It follows from Corollary 4.6.10 that $\Gamma \models f(t_0, \ldots, t_{n-1}) = f(t_n, \ldots, t_{2n-1})$. □

Lemma 5.6.13. *Let $\mathcal{L}$ be a first-order language, Γ be a set of $\mathcal{L}$-formulas and φ be an $\mathcal{L}$-formula.*

If $\Gamma \overset{\bullet}{\vdash}_{\mathrm{fond}^{\mathcal{L}}} \varphi$ and $x \notin \mathtt{FV}(\Gamma)$, then $\Gamma \overset{\bullet}{\vdash}_{\mathrm{fond}^{\mathcal{L}}} (\forall x)\varphi$.

Proof. The conclusion follows from an application of Part 7 of Definition 5.6.3. □

Theorem 5.6.14 (Completeness for First-Order Natural Deduction). *Let $\mathcal{L}$ be a first-order language, Γ be a set of $\mathcal{L}$-formulas and let θ be an $\mathcal{L}$-formula. If $\Gamma \models \theta$, then $\Gamma \overset{\bullet}{\vdash}_{\mathrm{fond}^{\mathcal{L}}} \theta$.*

Proof. If $\Gamma \models \theta$, then, by the completeness of $\mathcal{HF}^{\mathcal{L}}_{\Gamma}$ (Theorem 5.2.41), we have $\Gamma \vdash_{\mathcal{HF}^{\mathcal{L}}} \theta$. Therefore, it suffices to prove by induction on the theorems of $\mathcal{HF}^{\mathcal{L}}_{\Gamma}$ that if $\Gamma \vdash_{\mathcal{HF}^{\mathcal{L}}} \theta$, then $\Gamma \overset{\bullet}{\vdash}_{\mathrm{fond}^{\mathcal{L}}} \theta$. Figures 3.19–3.25 still show that that for every formula γ of formula groups 1 to 18 of $\mathcal{HF}^{\mathcal{L}}$, $\emptyset \overset{\bullet}{\vdash}_{\mathrm{fond}^{\mathcal{L}}} \gamma$. Figures 5.23 to 5.27 show that the same holds for formulas in groups 19 to 23.

If $= \in \mathcal{L}$, then Figures 5.28 to 5.30 show that if γ is in formula groups 24 to 26, then $\emptyset \overset{\bullet}{\vdash}_{\mathrm{fond}^{\mathcal{L}}} \gamma$.

To deal with formula group 27, let f be an n-ary function symbol of $\mathcal{L}$ with $n > 0$ and let $t_0, \ldots, t_{2n-1} \in \mathrm{TERM}_{\mathcal{L}}$. Define the formulas $\theta_{k,n} = (t_0 = t_n \wedge \cdots \wedge t_k = t_{n+k})$, for $0 \leq k \leq n-1$. The natural deduction tree in Figure 5.31 shows that

$$\begin{aligned}&\emptyset \overset{\bullet}{\vdash}_{\mathrm{fond}^{\mathcal{L}}} ((t_0 = t_n \wedge \cdots \wedge t_{n-1} = t_{2n-1}) \rightarrow f(t_0, \ldots, t_{n-1}) \\ &\quad = f(t_n, \ldots, t_{2n-1})).\end{aligned}$$

For formula group 28, consider an n-ary relation symbol R of $\mathcal{L}$. Figure 5.34 contains a natural deduction tree which proves that

$$\begin{aligned}&\emptyset \overset{\bullet}{\vdash}_{\mathrm{fond}^{\mathcal{L}}} ((t_0 = t_n \wedge \cdots \wedge t_{n-1} = t_{2n-1}) \\ &\quad \rightarrow (R(t_0, \ldots, t_{n-1}) \leftrightarrow R(t_n, \ldots, t_{2n-1})))\end{aligned}$$

In building this tree, we use the natural deduction trees $\mathtt{T}'$ and $\mathtt{T}''$ shown in Figures 5.32 and 5.33, where $\mathtt{T}'(\lambda) = R(t_n, \ldots, t_{2n-1})$ and $\mathtt{T}''(\lambda) = R(t_0, \ldots, t_{n-1})$. When we insert $\mathtt{T}', \mathtt{T}''$ into the natural deduction tree of Figure 5.34, the leaves of these trees are all cancelled.

At this point, we have shown that for every formula γ in the formula groups 1 to 28, we have $\emptyset \overset{\bullet}{\vdash}_{\mathrm{fond}^{\mathcal{L}}} \gamma$. It follows from Lemma 5.6.13 that for every formula θ in the corresponding axiom groups (obtained by generalizing the previous formulas), we have $\emptyset \overset{\bullet}{\vdash}_{\mathrm{fond}^{\mathcal{L}}} \theta$. Thus $\Gamma \overset{\bullet}{\vdash}_{\mathrm{fond}^{\mathcal{L}}} \theta$ for all axioms θ of $\mathcal{HF}^{\mathcal{L}}$.

If $\theta \in \Gamma$, the existence of the one-node natural deduction tree $(\mathtt{T}, \emptyset)$ such that $\mathtt{T}(\lambda) = \theta$ shows that $\Gamma \overset{\bullet}{\vdash}_{\mathrm{nd}} \theta$.

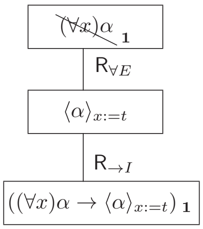

$\cancel{(\forall x)\alpha}$ 1
$\mathsf{R}_{\forall E}$
$\langle\alpha\rangle_{x:=t}$
$\mathsf{R}_{\to I}$
$((\forall x)\alpha \to \langle\alpha\rangle_{x:=t})$ 1

Fig. 5.23. Natural Deduction Tree for $((\forall x)\alpha \to \langle\alpha\rangle_{x:=t})$

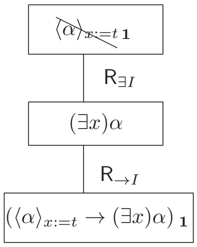

$\cancel{\langle\alpha\rangle_{x:=t}}$ 1
$\mathsf{R}_{\exists I}$
$(\exists x)\alpha$
$\mathsf{R}_{\to I}$
$(\langle\alpha\rangle_{x:=t} \to (\exists x)\alpha)$ 1

Fig. 5.24. Natural Deduction Tree for $(\langle\alpha\rangle_{x:=t} \to (\exists x)\alpha)$

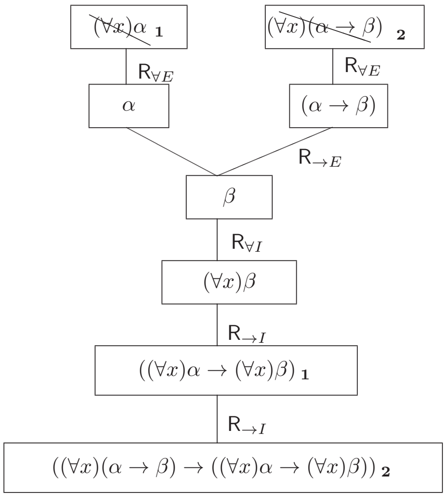

$\cancel{(\forall x)\alpha}$ 1
$\cancel{(\forall x)(\alpha \to \beta)}$ 2
$\mathsf{R}_{\forall E}$
$\mathsf{R}_{\forall E}$
α
$(\alpha \to \beta)$
$\mathsf{R}_{\to E}$
β
$\mathsf{R}_{\forall I}$
$(\forall x)\beta$
$\mathsf{R}_{\to I}$
$((\forall x)\alpha \to (\forall x)\beta)$ 1
$\mathsf{R}_{\to I}$
$((\forall x)(\alpha \to \beta) \to ((\forall x)\alpha \to (\forall x)\beta))$ 2

Fig. 5.25. Natural Deduction Tree for $((\forall x)(\alpha \to \beta) \to ((\forall x)\alpha \to (\forall x)\beta))$

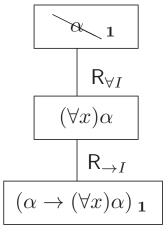


Fig. 5.26. Natural Deduction Tree for $(\alpha \to (\forall x)\alpha)$ where $x \notin \mathtt{FV}(\alpha)$

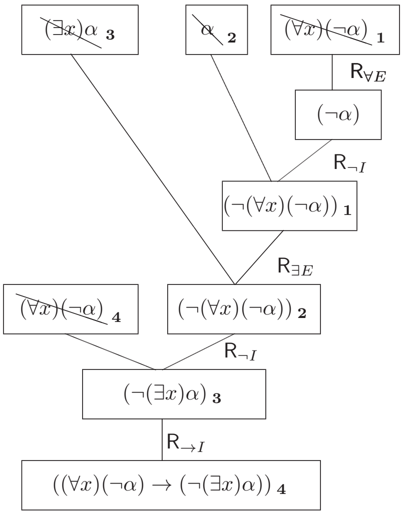

$\cancel{(\exists x)\alpha}$ **3**
$\cancel{\alpha}$ **2**
$\cancel{(\forall x)(\neg\alpha)}$ **1**
$R_{\forall E}$
$(\neg\alpha)$
$R_{\neg I}$
$(\neg(\forall x)(\neg\alpha))$ **1**
$R_{\exists E}$
$\cancel{(\forall x)(\neg\alpha)}$ **4**
$(\neg(\forall x)(\neg\alpha))$ **2**
$R_{\neg I}$
$(\neg(\exists x)\alpha)$ **3**
$R_{\to I}$
$((\forall x)(\neg\alpha) \to (\neg(\exists x)\alpha))$ **4**

Fig. 5.27. Natural Deduction Tree for $((\forall x)(\neg\alpha) \to (\neg(\exists x)\alpha))$

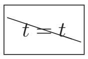


Fig. 5.28. Natural Deduction Tree for $t = t$

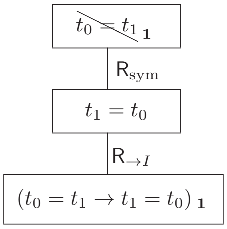


Fig. 5.29. Natural Deduction Tree for $(t_0 = t_1 \to t_1 = t_0)$

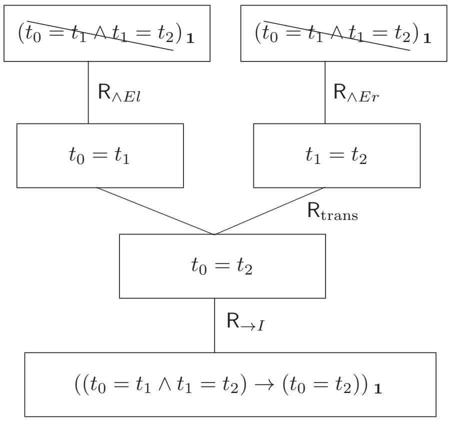

$\cancel{(t_0 = t_1 \wedge t_1 = t_2)}$ 1
$\cancel{(t_0 = t_1 \wedge t_1 = t_2)}$ 1
$R_{\wedge El}$
$R_{\wedge Er}$
$t_0 = t_1$
$t_1 = t_2$
R_{trans}
$t_0 = t_2$
$R_{\to I}$
$((t_0 = t_1 \wedge t_1 = t_2) \to (t_0 = t_2))$ 1

Fig. 5.30. Natural Deduction Tree for $((t_0 = t_1 \wedge t_1 = t_2) \to (t_0 = t_2))$

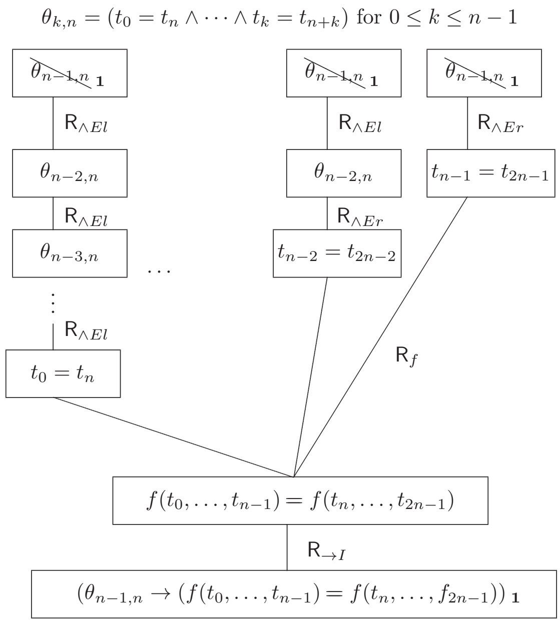

$\theta_{k,n} = (t_0 = t_n \wedge \cdots \wedge t_k = t_{n+k})$ for $0 \le k \le n-1$
$\cancel{\theta_{n-1,n}}$ 1
$\cancel{\theta_{n-1,n}}$ 1
$\cancel{\theta_{n-1,n}}$ 1
$R_{\wedge El}$
$R_{\wedge El}$
$R_{\wedge Er}$
$\theta_{n-2,n}$
$\theta_{n-2,n}$
$t_{n-1} = t_{2n-1}$
$R_{\wedge El}$
$R_{\wedge Er}$
$\theta_{n-3,n}$
$\ldots$
$t_{n-2} = t_{2n-2}$
$R_{\wedge El}$
$t_0 = t_n$
R_f
$f(t_0, \ldots, t_{n-1}) = f(t_n, \ldots, t_{2n-1})$
$R_{\to I}$
$(\theta_{n-1,n} \to (f(t_0, \ldots, t_{n-1}) = f(t_n, \ldots, f_{2n-1}))$ 1

Fig. 5.31. Natural Deduction Tree for Formula Group 27

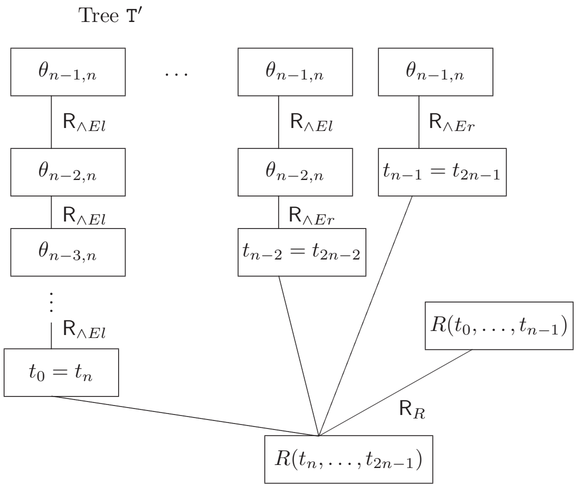


Fig. 5.32. Natural Deduction Tree $\mathtt{T}'$ Used in Proof of Theorem 5.6.14

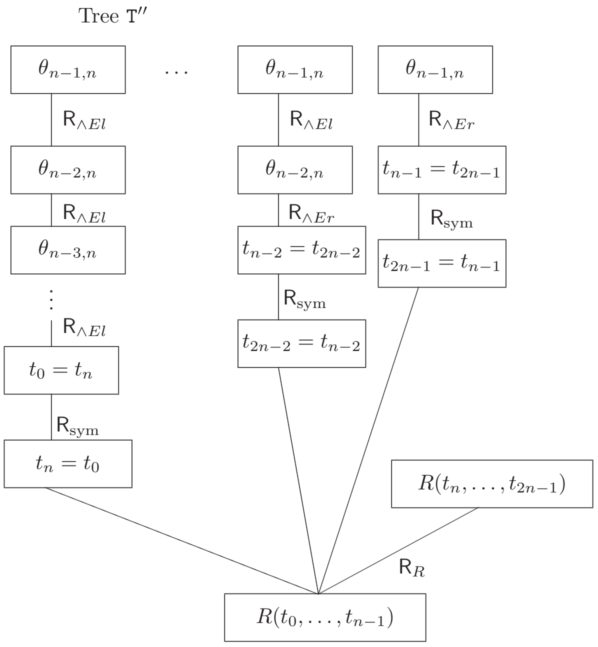
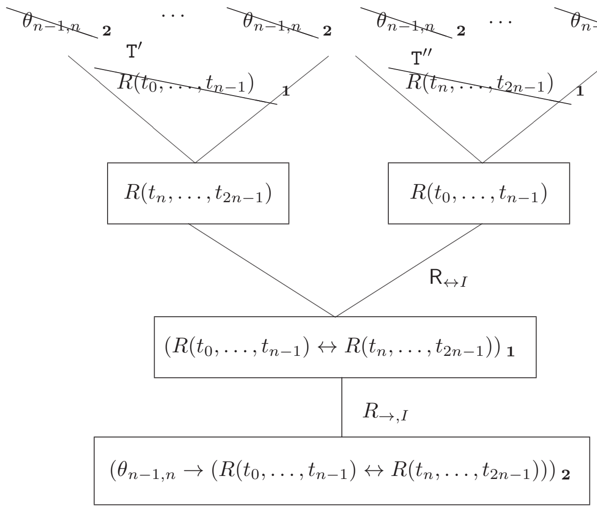


Fig. 5.33. Natural Deduction Tree $\mathtt{T}''$ Used in Proof of Theorem 5.6.14



$\theta_{n-1,n}$ 2
$\theta_{n-1,n}$ 2
$\theta_{n-1,n}$ 2
$\theta_{n-1,n}$ 2
T′
T″
$R(t_0,\ldots,t_{n-1})$ 1
$R(t_n,\ldots,t_{2n-1})$ 1
$R(t_n,\ldots,t_{2n-1})$
$R(t_0,\ldots,t_{n-1})$
$R_{\leftrightarrow I}$
$(R(t_0,\ldots,t_{n-1}) \leftrightarrow R(t_n,\ldots,t_{2n-1}))$ 1
$R_{\to,I}$
$(\theta_{n-1,n} \to (R(t_0,\ldots,t_{n-1}) \leftrightarrow R(t_n,\ldots,t_{2n-1})))$ 2

Fig. 5.34. Natural Deduction Tree for Formulas in Group 27

Suppose now that $(\mathtt{T}_0, M_0)$ and $(\mathtt{T}_1, M_1)$ are natural deduction trees for the $\mathcal{L}$-formulas φ and $(\varphi \to \psi)$, respectively, with

$$\mathcal{UNC}(\mathtt{T}_0, M_0), \mathcal{UNC}(\mathtt{T}_1, M_1) \subseteq \Gamma.$$

Then, by Rule 10 of Definition 5.6.3, $(\mathtt{T}, M) = ((\mathtt{T}_0, M_0), (\mathtt{T}_1, M_1); \psi)$ is a natural deduction tree for ψ such that $\mathcal{UNC}(\mathtt{T}, M) \subseteq \Gamma$. Therefore, $\Gamma \overset{\bullet}{\vdash}_{\mathrm{fond}^{\mathcal{L}}} \psi$. □

Theorem 5.6.15. *There is an effective, syntactic algorithm that, starting from a proof in* $\mathcal{HF}^{\mathcal{L}}_{\Gamma}$ *of an* $\mathcal{L}$*-formula* φ*, yields a natural deduction tree* $(\mathtt{T}, M)$ *such that* $\mathtt{T}(\lambda) = \varphi$ *and* $\mathcal{UNC}(\mathtt{T}, M) \subseteq \Gamma$.

Proof. This was shown in the proof of the Completeness Theorem for First-Order Natural Deduction. □

In presenting first-order natural deduction, we followed a path parallel to the path followed in presenting propositional natural deduction, that is, we combined a formal system with an annotation of the proof trees of the system. This parallel is continued by presenting first-order natural deduction purely as a formal system (called $\mathcal{FOND}^{\mathcal{L}}$).

Definition 5.6.16. The set of objects of the formal system $\mathcal{FOND}^{\mathcal{L}}$ is $\mathcal{P}(\mathrm{FORM}_{\mathcal{L}}) \times \mathrm{FORM}_{\mathcal{L}}$; the set of axioms is

$$A = \begin{cases} \{(\Gamma, \varphi) \mid \varphi \in \Gamma\} & \text{if } = \notin \mathcal{L} \\ \{(\Gamma, \varphi) \mid \varphi \in \Gamma\} \cup \{(\Gamma, t = t) \mid t \in \mathrm{TERM}_{\mathcal{L}}\} & \text{otherwise.} \end{cases}$$

The *rules for introducing connective symbols* remain the same as the rules for propositional logic natural deduction and they are:

$\dfrac{(\Gamma_0, \varphi), (\Gamma_1, \psi)}{(\Gamma_0 \cup \Gamma_1, (\varphi \wedge \psi))}$	$\wedge$- introduction
$\dfrac{(\Gamma, \varphi)}{(\Gamma, (\varphi \vee \psi))} \qquad \dfrac{(\Gamma, \psi)}{(\Gamma, (\varphi \vee \psi))}$	$\vee$- introduction
$\dfrac{(\Gamma \cup \{\varphi\}, \psi)}{(\Gamma, (\varphi \to \psi))}$	$\to$ - introduction
$\dfrac{(\Gamma_0 \cup \{\varphi\}, \psi), (\Gamma_1 \cup \{\psi\}, \varphi)}{(\Gamma_0 \cup \Gamma_1, (\varphi \leftrightarrow \psi))}$	$\leftrightarrow$ - introduction
$\dfrac{(\Gamma_0 \cup \{\varphi\}, \psi), (\Gamma_1 \cup \{\varphi\}, (\neg\psi))}{(\Gamma_0 \cup \Gamma_1, (\neg\varphi))}$	$\neg$- introduction

The *rules for eliminating connective symbols in* $\mathcal{FOND}^{\mathcal{L}}$ are:

$\dfrac{(\Gamma, (\varphi \wedge \psi))}{(\Gamma, \varphi)} \qquad \dfrac{(\Gamma, (\varphi \wedge \psi))}{(\Gamma, \psi)}$	$\wedge$-elimination
$\dfrac{(\Gamma_0, (\varphi \vee \psi)), (\Gamma_1 \cup \{\varphi\}, \alpha), (\Gamma_2 \cup \{\psi\}, \alpha)}{(\Gamma_0 \cup \Gamma_1 \cup \Gamma_2, \alpha)}$	$\vee$-elimination
$\dfrac{(\Gamma_0, \varphi), (\Gamma_1, (\varphi \to \psi))}{(\Gamma_0 \cup \Gamma_1, \psi)}$	$\to$-elimination
$\dfrac{(\Gamma_0, \varphi), (\Gamma_1, (\varphi \leftrightarrow \psi))}{(\Gamma_0 \cup \Gamma_1, \psi)} \qquad \dfrac{(\Gamma_0, \psi), (\Gamma_1, (\varphi \leftrightarrow \psi))}{(\Gamma_0 \cup \Gamma_1, \varphi)}$	$\leftrightarrow$-elimination
$\dfrac{(\Gamma_0 \cup \{(\neg\varphi)\}, \psi), (\Gamma_1 \cup \{(\neg\varphi))\}, (\neg\psi))}{(\Gamma_0 \cup \Gamma_1, \varphi)}$	$\neg$-elimination

The *rules for introducing quantifier symbols in* $\mathcal{FOND}^{\mathcal{L}}$ are:

$\dfrac{(\Gamma, \varphi)}{(\Gamma, (\forall x)\varphi)} \qquad x \notin \mathtt{FV}(\Gamma)$	$\forall$-introduction
$\dfrac{(\Gamma, \langle\varphi\rangle_{x:=t})}{(\Gamma, (\exists x)\varphi)}$	$\exists$-introduction

The *rules for eliminating quantifier symbols in $\mathcal{FOND}^{\mathcal{L}}$* are:

$\dfrac{(\Gamma, (\forall x)\varphi)}{(\Gamma, \langle\varphi\rangle_{x:=t})}$		$\forall$-elimination
$\dfrac{(\Gamma_0, (\exists x)\varphi), (\Gamma_1 \cup \{\varphi\}, \psi)}{(\Gamma_0 \cup \Gamma_1, \psi)}$	$x \notin \mathtt{FV}(\Gamma_1 \cup \{\psi\})$	$\exists$-elimination

The *expansion rule* is:

$$\frac{(\Gamma, \varphi)}{(\Gamma', \varphi)}$$

for all sets of $\mathcal{L}$-formulas Γ, Γ' such that $\Gamma \subseteq \Gamma'$ and $\mathcal{L}$-formulas φ. (Note that there is no counterpart of the expansion rule in the propositional formal system $\mathcal{ND}$ because this is a derived rule in that system, as stated in Exercise 73 of Chapter 3.)

In addition, if $= \in \mathcal{L}$, then we include in $\mathcal{FOND}^{\mathcal{L}}$ the following *equality rules*:

$\dfrac{(\Gamma, t_0 = t_1)}{(\Gamma, t_1 = t_0)}$	Symmetry of equality
$\dfrac{(\Gamma_0, t_0 = t_1), (\Gamma_1, t_1 = t_2)}{(\Gamma_0 \cup \Gamma_1, t_0 = t_2)}$	Transitivity of equality
$\dfrac{(\Gamma_0, t_0 = t_n), \ldots, (\Gamma_{n-1}, t_{n-1} = t_{2n-1})}{(\bigcup_{i=0}^{n-1} \Gamma_i, f(t_0, \ldots, t_{n-1}) = f(t_n, \ldots, t_{2n-1}))}$	f-congruence rule
$\dfrac{(\Gamma_0, t_0 = t_n), \ldots, (\Gamma_{n-1}, t_{n-1} = t_{2n-1}), (\Gamma_n, R(t_0, \ldots, t_{n-1}))}{(\bigcup_{i=0}^{n} \Gamma_i, R(t_n, \ldots, t_{2n-1})}$	R-congruence rule

for every function symbol f of positive arity and every relation symbol R of positive arity. ▯

Example 5.6.17. We prove that $\vdash_{\mathcal{FOND}^{\mathcal{L}}} (\emptyset, ((\exists x)(\forall y)\varphi \rightarrow (\forall y)(\exists x)\varphi))$ for all $\mathcal{L}$-formulas φ, where $\mathcal{L}$ is a first-order language.

Consider the following proof:

(1) $(\{(\forall y)\varphi\}, (\forall y)\varphi)$	axiom
(2) $(\{(\forall y)\varphi\}, \varphi)$	(1) and $\forall$-elimination
(3) $(\{(\forall y)\varphi\}, (\exists x)\varphi)$	(2) and $\exists$-introduction
(4) $(\{(\forall y)\varphi\}, (\forall y)(\exists x)\varphi)$	(3) and $\forall$-introduction
(5) $(\{(\exists x)(\forall y)\varphi\}, (\exists x)(\forall y)\varphi)$	axiom
(6) $(\{(\exists x)(\forall y)\varphi\}, (\forall y)(\exists x)\varphi)$	(5), (4), and $\exists$-elimination
(7) $(\emptyset, ((\exists x)(\forall y)\varphi \rightarrow (\forall y)(\exists x)\varphi))$	(6) and $\rightarrow$-introduction

▯

Theorem 5.6.18. *Let $\mathcal{L}$ be a first-order language, Γ be a set of $\mathcal{L}$-formulas and let φ be an $\mathcal{L}$-formula. Then, $\Gamma \overset{\bullet}{\vdash}_{\mathrm{fond}^{\mathcal{L}}} \varphi$ if and only if $\vdash_{\mathcal{FOND}^{\mathcal{L}}} (\Gamma, \varphi)$.*

Proof. The argument expands the proof of the corresponding propositional result (Theorem 3.6.16). We show first that $\Gamma \overset{\bullet}{\vdash}_{\mathrm{fond}^{\mathcal{L}}} \varphi$ implies $\vdash_{\mathcal{FOND}^{\mathcal{L}}} (\Gamma, \varphi)$. To this end, we use induction on natural deduction trees of $\mathcal{L}$, to prove that if $\mathcal{T} = (\mathtt{T}, M)$ is a natural deduction tree of $\mathcal{L}$ such that $\mathcal{UNC}(\mathtt{T}, M) \subseteq \Gamma$ and $\mathtt{T}(\lambda) = \varphi$, then $\vdash_{\mathcal{FOND}^{\mathcal{L}}} (\Gamma, \varphi)$. If $\mathcal{T}$ is a one-node natural deduction tree, we have either $\varphi \in \Gamma$ or φ is $t = t$, for some $\mathcal{L}$-term t. In either case, (Γ, φ) is an axiom of $\mathcal{FOND}^{\mathcal{L}}$. This establishes the basis step.

According to the definition of natural deduction trees, we need to consider several inductive steps. The arguments for the rules of the form C-elimination or C-introduction, where C is a connective symbol remain the same. We discuss here the inductive steps for the $\forall$-introduction and $\exists$-elimination rules.

Suppose that $\mathcal{T}_0 = (\mathtt{T}_0, M_0)$ is a natural deduction tree such that $\mathtt{T}_0(\lambda) = \psi$, $x \not\in \mathtt{FV}(\mathcal{UNC}(\mathcal{T}_0))$ and that $\mathcal{T} = (\mathtt{T}, M)$ is $(\mathcal{T}_0; (\forall x)\psi)$, with $\mathcal{UNC}(\mathcal{T}) \subseteq \Gamma$.

By inductive hypothesis, $(\mathcal{UNC}(\mathcal{T}_0), \psi)$ is a theorem of $\mathcal{FOND}^{\mathcal{L}}$. We obtain $\vdash_{\mathcal{FOND}^{\mathcal{L}}} (\mathcal{UNC}(\mathcal{T}_0), (\forall x)\psi)$, by applying the $\forall$-introduction rule of $\mathcal{FOND}^{\mathcal{L}}$, because $x \not\in \mathtt{FV}(\mathcal{T}_0)$. Since $\mathcal{UNC}(\mathcal{T}_0) = \mathcal{UNC}(\mathcal{T})$, an application of the expansion rule gives $\vdash_{\mathcal{FOND}^{\mathcal{L}}} (\Gamma, (\forall x)\psi)$.

Now let $\mathcal{T}_0 = (\mathtt{T}_0, M_0), \mathcal{T}_1 = (\mathtt{T}_1, M_1)$ be natural deduction trees such that $\mathtt{T}_0(\lambda) = (\exists x)\theta$, $\mathtt{T}_1(\lambda) = \psi$ and $x \not\in \mathtt{FV}(\{\psi\} \cup (\mathcal{UNC}(\mathcal{T}_1) - \{\theta\}))$. These trees yield the natural deduction tree $\mathcal{T} = (\mathcal{T}_0, L_\theta(\mathcal{T}_1); \psi)$. Suppose that $\mathcal{UNC}(\mathcal{T}) = \mathcal{UNC}(\mathcal{T}_0) \cup (\mathcal{UNC}(\mathcal{T}_1) - \{\theta\}) \subseteq \Gamma$. By the inductive hypothesis (and possibly the expansion rule), we have $\vdash_{\mathcal{FOND}^{\mathcal{L}}} (\mathcal{UNC}(\mathcal{T}_0), (\exists x)\theta)$ and $\vdash_{\mathcal{FOND}^{\mathcal{L}}} ((\mathcal{UNC}(\mathcal{T}_1) - \{\theta\}) \cup \{\theta\}, \psi)$. Applying the $\exists$-elimination rule, we obtain $\vdash_{\mathcal{FOND}^{\mathcal{L}}} (\mathcal{UNC}(\mathcal{T}_0) \cup (\mathcal{UNC}(\mathcal{T}_1) - \{\theta\}), \psi) = (\mathcal{UNC}(\mathcal{T}), \psi)$, because $x \not\in \mathtt{FV}(\{\psi\} \cup (\mathcal{UNC}(\mathcal{T}_1) - \{\theta\}))$. An application of the expansion rule yields $\vdash_{\mathcal{FOND}^{\mathcal{L}}} (\Gamma, \psi)$.

If $= \in \mathcal{L}$, we need to deal with additional inductive cases. We discuss here only the case that involves n-ary function symbols. Suppose that $t_0, \ldots, t_{2n-1}$ are $\mathcal{L}$-terms, f is an n-ary function symbol of $\mathcal{L}$ with $n > 0$, and $\mathcal{T}_i = (\mathtt{T}_i, M_i)$

is such that $\mathtt{T}_i(\lambda) = t_i = t_{i+n}$ for $0 \leq i \leq n-1$. Let $\mathcal{T} = (\mathcal{T}_0, \ldots, \mathcal{T}_{n-1}; f(t_0, \ldots, t_{n-1}) = f(t_n, \ldots, t_{2n-1}))$ and suppose that $\mathcal{UNC}(\mathcal{T}) = \bigcup_{i=0}^{n-1} \mathcal{UNC}(\mathcal{T}_i) \subseteq \Gamma$. By inductive hypothesis, we have $\vdash_{\mathcal{FOND}^{\mathcal{L}}} (\Gamma, t_i = t_{i+n})$, for $0 \leq i \leq n-1$, which implies $\vdash_{\mathcal{FOND}^{\mathcal{L}}} (\Gamma, f(t_0, \ldots, t_{n-1}) = f(t_n, \ldots, t_{2n-1}))$ by the f-congruence rule.

Conversely, we show by induction on the theorems of $\mathcal{FOND}^{\mathcal{L}}$ that $\vdash_{\mathcal{FOND}^{\mathcal{L}}} (\Gamma, \varphi)$ implies $\Gamma \overset{\bullet}{\vdash}_{\mathrm{fond}^{\mathcal{L}}} \varphi$. For the basis step, let (Γ, φ) be an axiom of $\mathcal{FOND}^{\mathcal{L}}$. If $\varphi \in \Gamma$, the existence of the one node natural deduction tree $(\mathtt{T}, \emptyset)$ with $\mathtt{T}(\lambda) = \varphi$ implies $\Gamma \overset{\bullet}{\vdash}_{\mathrm{fond}^{\mathcal{L}}} \varphi$. If φ is $t = t$, then the existence of the one node natural deduction tree $(\mathtt{T}, \{\lambda\})$ with $\mathtt{T}(\lambda) = \varphi$ implies $\Gamma \overset{\bullet}{\vdash}_{\mathrm{fond}^{\mathcal{L}}} \varphi$.

There is one inductive step for each of the rules of $\mathcal{FOND}^{\mathcal{L}}$. The steps for C-introduction and C-elimination where C is a connective symbol are the same as the corresponding propositional steps in the proof of Theorem 3.6.16.

We discuss here only $\forall$-introduction, $\exists$-elimination, and expansion.

Suppose that $(\Gamma, (\forall x)\varphi)$ is obtained from (Γ, φ) using the $\forall$-introduction rule. This means that $x \notin \mathtt{FV}(\Gamma)$. By the inductive hypothesis, $\Gamma \overset{\bullet}{\vdash}_{\mathrm{fond}^{\mathcal{L}}} \varphi$. Consequently, there is a first-order natural deduction tree $\mathcal{T}_0 = (\mathtt{T}_0, M_0)$ such that $\mathtt{T}_0(\lambda) = \varphi$ and $\mathcal{UNC}(\mathcal{T}_0) \subseteq \Gamma$. Thus, $x \notin \mathtt{FV}(\mathcal{UNC}(\mathcal{T}_0))$, which allows us to obtain the first-order natural deduction tree $\mathcal{T} = (\mathcal{T}_0; (\forall x)\varphi)$ with $\mathcal{UNC}(\mathcal{T}) = \mathcal{UNC}(\mathcal{T}_0) \subseteq \Gamma$. Thus, $\Gamma \overset{\bullet}{\vdash}_{\mathrm{fond}^{\mathcal{L}}} (\forall x)\varphi$.

For the $\exists$-elimination case, suppose that $(\Gamma_0 \cup \Gamma_1, \psi)$ is obtained from $(\Gamma_0, (\exists x)\varphi)$ and $(\Gamma_1 \cup \{\varphi\}, \psi)$ by the application of the $\exists$-elimination rule, which means that $x \notin \mathtt{FV}(\Gamma_1 \cup \{\psi\})$. The inductive hypothesis implies that $\Gamma_0 \overset{\bullet}{\vdash}_{\mathrm{fond}^{\mathcal{L}}} (\exists x)\varphi$ and $\Gamma_1 \overset{\bullet}{\vdash}_{\mathrm{fond}^{\mathcal{L}}} \psi$. Thus, there are natural deduction trees $\mathcal{T}_0 = (\mathtt{T}_0, M_0)$ and $\mathcal{T}_1 = (\mathtt{T}_1, M_1)$ with $\mathtt{T}_0(\lambda) = (\exists x)\varphi$ and $\mathtt{T}_1(\lambda) = \psi$ such that $\mathcal{UNC}(\mathcal{T}_0) \subseteq \Gamma_0$ and $\mathcal{UNC}(\mathcal{T}_1) \subseteq \Gamma_1 \cup \{\varphi\}$. The previous condition imposed on x means that $x \notin \mathtt{FV}(\psi) \cup \mathtt{FV}(\mathcal{UNC}(\mathcal{T}_1) - \{\varphi\})$, which allows us to form the first-order natural deduction tree $\mathcal{T} = (\mathcal{T}_0, L_\varphi(\mathcal{T}_1); \psi)$. For $\mathcal{T}$, we have $\mathcal{UNC}(\mathcal{T}) = \mathcal{UNC}(\mathcal{T}_0) \cup (\mathcal{UNC}(\mathcal{T}_1) - \{\varphi\}) \subseteq \Gamma_0 \cup \Gamma_1$, so $\Gamma_0 \cup \Gamma_1 \overset{\bullet}{\vdash}_{\mathrm{fond}^{\mathcal{L}}} \psi$.

For the expansion rule, suppose that (Γ', φ) is obtained from (Γ, φ) by the application of the expansion rule, which means that $\Gamma \subseteq \Gamma'$. By inductive hypothesis, $\Gamma \overset{\bullet}{\vdash}_{\mathrm{fond}^{\mathcal{L}}} \varphi$, so there is a natural deduction tree $\mathcal{T} = (\mathtt{T}, M)$ such that $\mathcal{UNC}(\mathcal{T}) \subseteq \Gamma$ and $\mathtt{T}(\lambda) = \varphi$. Since $\Gamma \subseteq \Gamma'$, the same tree shows that $\Gamma' \overset{\bullet}{\vdash}_{\mathrm{fond}^{\mathcal{L}}} \varphi$. $\square$

Corollary 5.6.19. *Let $\mathcal{L}$ be a first-order language, Γ be a set of $\mathcal{L}$-formulas and φ be an $\mathcal{L}$-formula. The following three statements are equivalent:*

(1) $\Gamma \overset{\bullet}{\vdash}_{\mathrm{fond}^{\mathcal{L}}} \varphi$;
(2) $\vdash_{\mathcal{FOND}^{\mathcal{L}}} (\Gamma, \varphi)$;
(3) $\Gamma \models \varphi$.

Proof. The statement follows from Theorems 5.6.12, 5.6.14 and 5.6.18. $\square$

5.7 Transformations Between Formal Systems

In this section, we do for first-order logic what Section 3.7 did for propositional logic, that is, discuss syntactic transformations between proofs in different formalisms of the fact that $\Gamma \models \varphi$. The formalisms we have considered so far are:

- a Hilbert-Frege system;
- tableaux with and without cut;
- sequent systems with and without cut;
- natural deduction.

Some of these transformations have already been presented. Others will be introduced here.

5.7.1 *From Unsigned Tableaux to Hilbert-Frege Proofs*

Definition 5.7.1. Let $\mathcal{L}$ be a first-order language, $\mathcal{H}$ be a formal system whose objects are the $\mathcal{L}$-formulas and Γ be a set of $\mathcal{L}$-formulas. An $(\mathcal{H}, \Gamma)$-*certificate of inconsistency* is a pair (q, q') of proofs in $\mathcal{H}_\Gamma$ such that for some $\mathcal{L}$-formula α, q is a proof of α and q' is a proof of $(\neg\alpha)$. ▯

Effective versions of Theorems 5.2.29 and 5.2.30, and Corollary 5.2.31 are given in the following theorem.

Theorem 5.7.2.

(1) *There is an effective, syntactic construction that starts with an $\mathcal{L}$-formula φ and an $(\mathcal{HF}^{\mathcal{L}}, \Gamma)$-certificate of inconsistency (q, q') and produces a proof in $\mathcal{HF}^{\mathcal{L}}_{\Gamma}$ for φ.*
(2) *There is an effective, syntactic construction that starts with (q, q'), an $(\mathcal{HF}^{\mathcal{L}}, \Gamma \cup \{(\neg\varphi)\})$-certificate of inconsistency, and yields a proof of φ in $\mathcal{HF}^{\mathcal{L}}_{\Gamma}$.*
(3) *There is an effective, syntactic construction that starts with (q, q'), an $(\mathcal{HF}^{\mathcal{L}}, \Gamma \cup \{\varphi\})$-certificate of inconsistency and (r, r'), an $(\mathcal{HF}^{\mathcal{L}}, \Gamma \cup \{(\neg\varphi)\})$-certificate of inconsistency and yields (s, s') an $(\mathcal{HF}^{\mathcal{L}}, \Gamma)$-certificate of inconsistency.*

Proof. The argument follows the same lines as the proof of Theorem 3.2.15. □

Lemma 5.7.3. *Let $\mathcal{L}$ be a first-order language, Γ be a set of $\mathcal{L}$-formulas and let φ be an $\mathcal{L}$-formula that is either $(\alpha C \beta)$ or $(\neg(\alpha C \beta))$. Then, there is an effective, syntactic construction that, starting from $(\mathcal{HF}^{\mathcal{L}}, \Gamma \cup K)$-certificates of inconsistency for all constituents K of φ, produces $(\mathcal{HF}^{\mathcal{L}}, \Gamma')$-certificates of inconsistency when Γ' is $\Gamma \cup \{\varphi, \alpha, \beta\}$, $\Gamma \cup \{\varphi, (\neg\alpha), \beta\}$, $\Gamma \cup \{\varphi, \alpha, (\neg\beta)\}$, $\Gamma \cup \{\varphi, (\neg\alpha), (\neg\beta)\}$.*

Proof. The proof is similar to the argument of Lemma 3.2.16. □

Theorem 5.7.4. *Let $\mathcal{L}$ be a first-order language. There is an effective, syntactic construction that starts with a set Γ of $\mathcal{L}$-formulas, an $\mathcal{L}$-formula φ of one of the forms $(\alpha C \beta)$, $(\neg(\alpha C \beta))$, or $(\neg(\neg\alpha))$, and $(\mathcal{HF}^{\mathcal{L}}, \Gamma \cup K)$-certificates of inconsistency for all constituents K of φ, and produces an $(\mathcal{HF}, \Gamma \cup \{\varphi\})$-certificate of inconsistency.*

Proof. If φ is a positive formula or a negated positive formula, apply Lemma 5.7.3 and Theorem 5.7.2, Part (3).

If $\varphi = (\neg(\neg\alpha))$, then we have an $(\mathcal{HF}, \Gamma \cup \{\alpha\})$-certificate of inconsistency, which is also an $(\mathcal{HF}, \Gamma \cup \{\varphi, \alpha\})$-certificate of inconsistency. Also, note that $((\neg\alpha), (\neg(\neg\alpha)))$ is an $(\mathcal{HF}, \Gamma \cup \{\varphi, (\neg\alpha)\})$-certificate of inconsistency. Using Theorem 5.7.2, Part (3), we obtain an $(\mathcal{HF}, \Gamma \cup \{\varphi\})$-certificate of inconsistency. □

Theorem 5.7.5. *Let $\mathcal{L}$ be a first-order language. There is an effective, syntactic construction that starts with a set Γ of $\mathcal{L}$-formulas,* *a γ-formula $\varphi = (\forall x)\psi$ ($\varphi = (\neg(\exists x)\psi)$) in* $\mathrm{FORM}_{\mathcal{L}}$ *and an $(\mathcal{HF}^{\mathcal{L}}, \Gamma \cup \{\varphi, \beta\})$-certificate of inconsistency, where $\beta = (\psi')_{x:=t}$ ($\beta = ((\neg\psi'))_{x:=t}$) with ψ' a variant of ψ such that t is substitutable for x in ψ' and produces an $(\mathcal{HF}^{\mathcal{L}}, \Gamma \cup \{\varphi\})$-certificate of inconsistency.*

Proof. Let (q_0, q_1) be an $(\mathcal{HF}^{\mathcal{L}}, \Gamma \cup \{\varphi, \beta\})$-certificate of inconsistency. The formulas $\mathsf{variant}(\psi, x, t)$ and ψ' are both variants of ψ, so they are variants of each other. Since t is substitutable for x in both of these formulas, by Supplement 83 of Chapter 4, $\langle\psi\rangle_{x:=t} = (\mathsf{variant}(\psi, x, t))_{x:=t}$ and $(\psi')_{x:=t}$ are variants. By Theorem 5.2.51, we can effectively find an $\mathcal{HF}^{\mathcal{L}}$ proof $(\langle\psi\rangle_{x:=t} \leftrightarrow (\psi')_{x:=t})$. Further, by Theorem 5.2.43, we can effectively find $\mathcal{HF}^{\mathcal{L}}$ proofs r_0 of $(\langle\psi\rangle_{x:=t} \to (\psi')_{x:=t})$ and r_1 of $((\psi')_{x:=t} \to \langle\psi\rangle_{x:=t})$.

First suppose that $\varphi = (\forall x)\psi$ and $\beta = (\psi')_{x:=t}$. Then, (q_0', q_1') is an $(\mathcal{HF}^{\mathcal{L}}, \Gamma \cup \{\varphi\})$-certificate of inconsistency, where

$$q_i' = (((\forall x)\psi \to \langle\psi\rangle_{x:=t}), (\forall x)\psi, \langle\psi\rangle_{x:=t})r_0 q_i.$$

Now suppose that $\varphi = (\neg(\exists x)\psi)$ and $\beta = ((\neg\psi'))_{x:=t} = (\neg(\psi')_{x:=t})$. Let r be a propositional proof in $\mathcal{HF}$ of the formula $((p \to q) \to ((\neg q) \to (\neg p)))$. We remind the reader that one such proof was provided in Part (d) of Supplement 3 of Chapter 3. Let r' be the proof in $\mathcal{HF}^{\mathcal{L}}$ obtained by replacing p with $\langle\psi'\rangle_{x:=t}$ and q with $(\exists x)\psi$ and let r'' be the $\mathcal{HF}^{\mathcal{L}}$ proof obtained by replacing p with $(\psi')_{x:=t}$ and q with $\langle\psi\rangle_{x:=t}$. Then the sequence r''' given by

$$\begin{aligned} r'((\langle\psi\rangle_{x:=t} &\to (\exists x)\psi), \\ ((\neg(\exists x)\psi) &\to (\neg\langle\psi\rangle_{x:=t})))r_1 \cdot \\ r''(((\neg\langle\psi\rangle_{x:=t}) &\to (\neg(\psi')_{x:=t}))) \end{aligned}$$

is an $\mathcal{HF}^{\mathcal{L}}$-proof. This allows us to obtain an $(\mathcal{HF}^{\mathcal{L}}, \Gamma \cup \{\varphi\})$-certificate of inconsistency as (q_0', q_1'), where $q_i' = r'''((\neg(\exists x)\psi), (\neg\langle\psi\rangle_{x:=t}))q_i$, for $i \in \{0, 1\}$. □

Theorem 5.7.6. *Let $\mathcal{L}$ be a first-order language. There is an effective, syntactic construction that starts with a set Γ of $\mathcal{L}$-formulas,* *a* $\boldsymbol{\delta}$*-formula* φ *in* $\mathrm{FORM}_{\mathcal{L}}$ *and an* $(\mathcal{HF}^{\mathcal{L}}, \Gamma \cup \{\beta\})$*-certificate of inconsistency, where*

$$\beta = \begin{cases} \langle\psi\rangle_{x:=c} & \textit{if } \varphi = (\exists x)\psi \\ \langle(\neg\psi)\rangle_{x:=c} & \textit{if } \varphi = (\neg(\forall x)\psi), \end{cases}$$

and $c \in \mathcal{L}$ *is a constant symbol that does not occur in either* Γ *or* φ*, and produces an* $(\mathcal{HF}, \Gamma \cup \{\varphi\})$*-certificate of inconsistency.*

Proof. Starting from an $(\mathcal{HF}^{\mathcal{L}}, \Gamma \cup \{\beta\})$-certificate of inconsistency (q_0, q_1), by the first part of Theorem 5.7.2, we can construct effectively an $(\mathcal{HF}^{\mathcal{L}}, \Gamma \cup \{\beta\})$-certificate of inconsistency (r_0, r_1) such that there is a formula α that does not contain c, r_0 is a proof of α, and r_1 is a proof of $(\neg\alpha)$.

In the first case, we have $\varphi = (\exists x)\psi$ and $\beta = \langle\psi\rangle_{x:=c}$. Applying Theorem 5.2.23 with Γ replaced by $\Gamma \cup \{(\exists x)\psi\}$ to the certificate of inconsistency (r_0, r_1), we obtain effectively the $(\mathcal{HF}^{\mathcal{L}}, \Gamma \cup \{\varphi\})$-certificate of inconsistency (r_0', r_1'), where the formula proved by r_i' is the same as the formula proved by r_i for $i \in \{0, 1\}$.

In the second case, we have $\varphi = (\neg(\forall x)\psi)$ and $\beta = \langle(\neg\psi)\rangle_{x:=c}$. Again, by Theorem 5.2.23, we obtain effectively the $(\mathcal{HF}^{\mathcal{L}}, \Gamma \cup \{(\neg(\forall x)\psi), (\exists x)(\neg\psi)\})$-certificate of inconsistency (r_0', r_1'). In Example 5.2.33, we saw that one can find effectively a proof r for $((\neg(\forall x)\psi) \to (\exists x)(\neg\psi))$. Let r' be the sequence $r((\neg(\forall x)\psi))$. Then, the pair $(r'r_0', r'r_1')$ is the desired $(\mathcal{HF}^{\mathcal{L}}, \Gamma \cup \{\varphi\})$-certificate of inconsistency. $\square$

Theorem 5.7.7. *Let* $\mathcal{L}$ *be a first-order language that contains the equality symbol. There is an effective, syntactic construction that starts with an* $(\mathcal{HF}^{\mathcal{L}}, \Gamma \cup \{\alpha\})$*-certificate of inconsistency, where* Γ *is a set of* $\mathcal{L}$*-formulas and* $\alpha \in \mathrm{INST}_{\mathcal{L}, VAR}(Eq_{=,\mathcal{L}})$ *and produces an* $(\mathcal{HF}^{\mathcal{L}}, \Gamma)$*-certificate of inconsistency.*

Proof. Since α is an axiom of $\mathcal{HF}^{\mathcal{L}}$, when $\mathcal{L}$ contains the equality symbol, the statement follows immediately because the same pair of proofs that serves as an $(\mathcal{HF}^{\mathcal{L}}, \Gamma \cup \{\alpha\})$-certificate of inconsistency will serve as an $(\mathcal{HF}^{\mathcal{L}}, \Gamma)$-certificate of inconsistency. $\square$

Lemma 5.7.8. *Let* $\mathcal{L}$ *be a first-order language,* Γ, Γ' *be two sets of* $\mathcal{L}$*-formulas and* f *be a function* $f : \Gamma \longrightarrow \Gamma'$ *such that for all* $\varphi \in \Gamma$*,*

$f(\varphi)$ is a variant of φ. There is an effective, syntactic construction that starts with an $\mathcal{HF}^{\mathcal{L}}_{\Gamma}$ proof of ψ and produces an $\mathcal{HF}^{\mathcal{L}}_{\Gamma'}$ proof of the same formula.

Proof. We first argue that for every formula $\varphi \in \Gamma$, we can construct effectively a proof for $\Gamma' \vdash_{\mathcal{HF}^{\mathcal{L}}} \varphi$. Indeed, since $f(\varphi)$ is a variant of φ, starting from $f(\varphi)$ and φ, we can construct effectively a proof for $(f(\varphi) \leftrightarrow \varphi)$ by Theorem 5.2.51, and hence, by Theorem 5.2.43, a proof of $(f(\varphi) \rightarrow \varphi)$. Since $f(\varphi) \in \Gamma'$, by Modus Ponens, we obtain effectively a proof of $\Gamma' \vdash_{\mathcal{HF}^{\mathcal{L}}} \varphi$.

Now, suppose that the sequence $(\psi_0, \ldots, \psi_i, \ldots, \psi_{n-1})$ is a proof in $\mathcal{HF}^{\mathcal{L}}_{\Gamma}$ of $\psi = \psi_{n-1}$. For each formula ψ_i that is a member of Γ, we replace ψ_i by its proof in $\mathcal{HF}^{\mathcal{L}}_{\Gamma'}$ as given by the first part of the argument. This results in a proof of ψ in $\mathcal{HF}^{\mathcal{L}}_{\Gamma'}$. □

Theorem 5.7.9. *Let $\mathcal{L}$ be a first-order language. There is an effective, syntactic construction that starts with two sets Γ, Γ' of $\mathcal{L}$-formulas, a function $f : \Gamma \longrightarrow \Gamma'$ such that for all $\varphi \in \Gamma$, $f(\varphi)$ is a variant of φ and an $(\mathcal{HF}^{\mathcal{L}}, \Gamma)$-certificate of inconsistency, and produces an $(\mathcal{HF}, \Gamma')$-certificate of inconsistency.*

Proof. This statement follows immediately from Lemma 5.7.8. □

Using the previous results, we can show that the collection of sets of $\mathcal{L}^c$-formulas $\mathcal{C}_{\mathcal{L}}$ that consists of those sets of formulas that are $\mathcal{L}^c$-consistent and contain a finite number of constant symbols in $\mathcal{L}^c - \mathcal{L}$ is an $(\mathcal{L}^c, \mathrm{VAR})$-consistency property. Recall that in Theorem 4.12.23, we have shown that for every first-order language $\mathcal{L}$ and $\mathcal{L}$-suitable set of variables V, every member of an $(\mathcal{L}, V)$-consistency property is a satisfiable set of formulas. Thus, we will be able to obtain an alternative proof of the completeness of $\mathcal{HF}^{\mathcal{L}}$.

Theorem 5.7.10. *Let $\mathcal{L}$ be a first-order language. The collection $\mathcal{C}_{\mathcal{L}}$ defined above is an $(\mathcal{L}^c$, VAR$)$-consistency property.*

Proof. We claim that $\mathcal{C}_{\mathcal{L}}$ satisfies the conditions of Definition 4.12.22. We deal here only with the second condition. The remaining conditions have relatively simpler arguments and are left to the reader.

Let $\Gamma \in \mathcal{C}_{\mathcal{L}}$ and let $\varphi \in \Gamma - \mathrm{LIT}_{\mathcal{L}^c}$, where φ is not a $\boldsymbol{\gamma}$-formula. The case when φ is a propositional formula follows from Theorem 5.7.4

plus the fact that if Γ contains finitely many constant symbols in $\mathcal{L}^c - \mathcal{L}$, the same applies to $\Gamma \cup K$ for any constituent K of φ. Suppose therefore that φ is a $\boldsymbol{\delta}$-formula. There are two subcases to consider depending on whether φ has the form $(\exists x)\psi$ or $(\neg(\forall x)\psi)$.

In the first subcase, let c be an $\mathcal{L}^c$-constant symbol that does not occur in Γ. The existence of c is assured by the fact that Γ contains finitely many constant symbols in $\mathcal{L}^c - \mathcal{L}$. By Theorem 5.7.6, $\Gamma \cup \{\langle \psi \rangle_{x:=c}\}$ is $\mathcal{L}^c$-consistent and clearly contains only finitely many constant symbols in $\mathcal{L}^c - \mathcal{L}$. The second subcase is similar and can be shown using the same theorem. $\square$

The previous developments provide an alternative argument for Theorem 5.2.39, which is a key ingredient in proving the completeness of $\mathcal{HF}^{\mathcal{L}}$. Let Γ be an $\mathcal{L}$-consistent set. Then, by Theorem 5.2.36, Γ is $\mathcal{L}^c$-consistent set and thus it belongs to $\mathcal{C}_{\mathcal{L}}$. Since $\mathcal{C}_{\mathcal{L}}$ is a consistency property and every member of such a collection is satisfiable by Theorem 4.12.23, it follows that Γ is satisfiable.

The main result of the subsection is given next.

Theorem 5.7.11. *For a first-order language $\mathcal{L}$, there is an effective, syntactic construction that starts with a strongly closed* $(\Gamma \cup \{(\neg\varphi)\}, \mathcal{L}, \mathit{VAR})$*-tableau* $\mathtt{T}$ *and produces a proof in $\mathcal{HF}^{\mathcal{L}}_{\Gamma}$ of φ, where Γ is a set of $\mathcal{L}$-formulas and φ is an $\mathcal{L}$-formula.*

Proof. Since $\mathtt{T}$ is strongly closed, we have $(\mathcal{HF}^{\mathcal{L}}, \mathtt{T}(q))$-certificates of inconsistency for each of its leaves q. Using Theorems 5.7.4 to 5.7.9 repeatedly, we can construct an $(\mathcal{HF}^{\mathcal{L}}, \mathtt{T}(\lambda))$-certificate of inconsistency, where $\mathtt{T}(\lambda) = \Gamma \cup \{(\neg\varphi)\}$. By applying Part (2) of Theorem 5.7.2, we can construct effectively a proof of φ in $\mathcal{HF}^{\mathcal{L}}$. $\square$

5.7.2 *From Natural Deduction Trees to Sequent Proofs*

Theorem 5.7.12. *Let $\mathcal{L}$ be a first-order language with infinitely many constant symbols. There is a syntactic algorithm that takes as input a first-order natural deduction tree $\mathcal{T} = (\mathtt{T}, M)$ in* $\mathit{FONDT}^{\mathcal{L}}$ *and produces as output an $\mathcal{F}^{\text{seq,cut}}_{\mathcal{L},\mathit{VAR}}$-proof tree of the sequent $\mathcal{UNC}(\mathcal{T}) \Rightarrow \mathtt{T}(\lambda)$.*

Proof. We give a recursive algorithm, based on the inductive definition of natural deduction tree (cf. Definition 5.6.3). We proceed according to the first case of this definition that is applicable to $\mathcal{T}$.

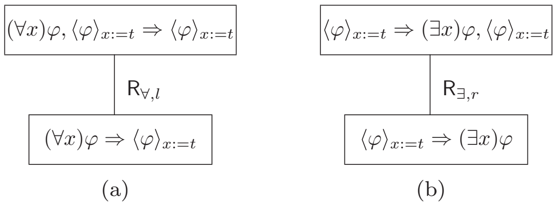
For the basis step, when $\mathcal{T}$ consists of one node, we have either $\mathcal{T} = (\mathtt{T}, \emptyset)$ or $\mathcal{T} = (\mathtt{T}_{t=t}, \{\lambda\})$. The first case can be dealt with exactly in the same manner as in the proof of Theorem 3.7.3. For the second case, observe that $t = t \Rightarrow t = t$ is an axiom of $\mathcal{F}^{\text{seq,cut}}_{\mathcal{L},\text{VAR}}$, so we can derive $\emptyset \Rightarrow t = t$ by rule $R_{=,l}$.

If $\mathcal{T}$ is obtained by an application at the root of a propositional introduction or elimination rule, the argument proceeds along the same lines as the corresponding propositional argument.

If $\mathcal{T} = (\mathtt{T}, M)$ is obtained by an application of the $\forall$-introduction rule at the root, then we have $\mathcal{T} = (\mathcal{T}_0; (\forall x)\varphi)$, where $\mathcal{T}_0$ is a first-order natural deduction tree in $\text{FONDT}^{\mathcal{L}}$, whose root is labelled with φ and x does not occur free in $\mathcal{UNC}(\mathcal{T}_0) = \mathcal{UNC}(\mathcal{T})$. By inductive hypothesis, we have an $\mathcal{F}^{\text{seq,cut}}_{\mathcal{L},\text{VAR}}$-proof tree $\mathtt{T}'$ of the sequent $\mathcal{UNC}(\mathcal{T}_0) \Rightarrow \varphi$. Define the tree $\mathtt{T}''$ by $\text{Dom}(\mathtt{T}'') = \text{Dom}(\mathtt{T}')$ and $\mathtt{T}''(q) = (\mathtt{T}'(q))_{x:=c}$, where c is a constant symbol of $\mathcal{L}$ that is not an eigenconstant of $\mathtt{T}'$ and does not occur in $\mathcal{UNC}(\mathcal{T}_0)$ or in φ. The existence of c follows from the fact that $\mathcal{L}$ contains an infinite supply of constant symbols and from the finiteness of $\mathtt{T}'$. By Corollary 5.5.43, $\mathtt{T}''$ is a $\mathcal{F}^{\text{seq,cut}}_{\mathcal{L},\text{VAR}}$-proof tree of the sequent $(\mathcal{UNC}(\mathcal{T}_0))_{x:=c} \Rightarrow (\varphi)_{x:=c}$, which is actually $\mathcal{UNC}(\mathcal{T}_0) \Rightarrow (\varphi)_{x:=c}$ because x does not occur free in $\mathcal{UNC}(\mathcal{T}_0) = \mathcal{UNC}(\mathcal{T})$. Thus, applying the rule $\mathsf{R}_{\forall,r}$ at the root of $\mathtt{T}''$, we obtain the desired $\mathcal{F}^{\text{seq,cut}}_{\mathcal{L},\text{VAR}}$-proof of the sequent $\mathcal{UNC}(\mathcal{T}) \Rightarrow (\forall x)\varphi$.

Suppose now that $\mathcal{T} = (\mathtt{T}, M)$ is obtained by an application of the $\forall$-elimination rule at the root. This means that $\mathcal{T} = (\mathcal{T}_0; \langle\varphi\rangle_{x:=t})$, for some $\mathcal{L}$-term t, where $\mathcal{T}_0$ is a first-order natural deduction tree in $\text{FONDT}^{\mathcal{L}}$, whose root is labelled with $(\forall x)\varphi$. By inductive hypothesis, we have an $\mathcal{F}^{\text{seq,cut}}_{\mathcal{L},\text{VAR}}$-proof tree $\mathtt{T}'$ of the sequent $\mathcal{UNC}(\mathcal{T}_0) \Rightarrow (\forall x)\varphi$. Consider now the $\mathcal{F}^{\text{seq,cut}}_{\mathcal{L},\text{VAR}}$-proof tree $\mathtt{T}_1$ shown in Figure 5.35(a), obtained by applying the rule $\mathsf{R}_{\forall,l}$ to the axiom $(\forall x)\varphi, \langle\varphi\rangle_{x:=t} \Rightarrow \langle\varphi\rangle_{x:=t}$. By Theorem 5.5.41, we obtain from $\mathtt{T}'$ and $\mathtt{T}_1$ an $\mathcal{F}^{\text{seq,cut}}_{\mathcal{L},\text{VAR}}$-proof tree for the sequent $\mathcal{UNC}(\mathcal{T}_0) \Rightarrow \langle\varphi\rangle_{x:=t}$ which equals $\mathcal{UNC}(\mathcal{T}) \Rightarrow \langle\varphi\rangle_{x:=t}$.

Suppose now that $\mathcal{T} = (\mathtt{T}, M)$ is obtained by an application of the $\exists$-introduction rule at the root. This means that $\mathcal{T} = (\mathcal{T}_0; (\exists x)\varphi)$,



$(\forall x)\varphi, \langle\varphi\rangle_{x:=t} \Rightarrow \langle\varphi\rangle_{x:=t}$
$\mathsf{R}_{\forall,l}$
$(\forall x)\varphi \Rightarrow \langle\varphi\rangle_{x:=t}$
(a)
$\langle\varphi\rangle_{x:=t} \Rightarrow (\exists x)\varphi, \langle\varphi\rangle_{x:=t}$
$\mathsf{R}_{\exists,r}$
$\langle\varphi\rangle_{x:=t} \Rightarrow (\exists x)\varphi$
(b)

Fig. 5.35. $\mathcal{F}^{\text{seq,cut}}_{\mathcal{L},\text{VAR}}$-proof Trees

where $\mathcal{T}_0$ is a first-order natural deduction tree in FONDT$^{\mathcal{L}}$, whose root is labelled with $\langle\varphi\rangle_{x:=t}$, for some $\mathcal{L}$-term t. By inductive hypothesis, we have an $\mathcal{F}^{\text{seq,cut}}_{\mathcal{L},\text{VAR}}$-proof tree $\mathtt{T}'$ of the sequent $\mathcal{UNC}(\mathcal{T}_0) \Rightarrow \langle\varphi\rangle_{x:=t}$. Consider now the $\mathcal{F}^{\text{seq,cut}}_{\mathcal{L},\text{VAR}}$-proof tree $\mathtt{T}_1$ shown in Figure 5.35(b), obtained by applying the rule $\mathsf{R}_{\exists,r}$ to the axiom $\langle\varphi\rangle_{x:=t} \Rightarrow (\exists x)\varphi, \langle\varphi\rangle_{x:=t}$. By Theorem 5.5.41, we obtain from $\mathtt{T}'$ and $\mathtt{T}_1$ an $\mathcal{F}^{\text{seq,cut}}_{\mathcal{L},\text{VAR}}$-proof tree for the sequent $\mathcal{UNC}(\mathcal{T}_0) \Rightarrow (\exists x)\varphi$ which equals $\mathcal{UNC}(\mathcal{T}) \Rightarrow (\exists x)\varphi$.

Now, assume that $\mathcal{T} = (\mathtt{T}, M)$ results from an application of the $\exists$-elimination rule at the root. We can write $\mathcal{T} = (\mathcal{T}_0, L_{\varphi}(\mathcal{T}_1); \psi)$, where $\mathcal{T}_0$ is a first-order natural deduction tree in FONDT$^{\mathcal{L}}$, whose root is labelled with $(\exists x)\varphi$, $\mathcal{T}_1$ is the same kind of tree whose root is labelled ψ and $x \notin \mathtt{FV}(\{\psi\} \cup (\mathcal{UNC}(\mathcal{T}_1) - \{\varphi\}))$. By the inductive hypothesis, we have $\mathcal{F}^{\text{seq,cut}}_{\mathcal{L},\text{VAR}}$-proof trees $\mathtt{T}'$ of the sequent $\mathcal{UNC}(\mathcal{T}_0) \Rightarrow (\exists x)\varphi$ and $\mathtt{T}''$ of the sequent $\mathcal{UNC}(\mathcal{T}_1) \Rightarrow \psi$. By thinning, we obtain the $\mathcal{F}^{\text{seq,cut}}_{\mathcal{L},\text{VAR}}$-proof tree $\mathtt{T}''_1 = (\mathtt{T}''; \mathcal{UNC}(\mathcal{T}_1), \varphi \Rightarrow \psi)$. Note that the root of this tree is labelled by $(\mathcal{UNC}(\mathcal{T}_1) - \{\varphi\}), \varphi \Rightarrow \psi$. Suppose that $x \in \mathtt{FV}(\varphi)$. Then, by applying Corollary 5.5.43, we can construct the $\mathcal{F}^{\text{seq,cut}}_{\mathcal{L},\text{VAR}}$-proof tree $\mathtt{T}''_2$ with the same domain as $\mathtt{T}''_1$ and $\mathtt{T}''_2(q) = (\mathtt{T}''_1(q))_{x:=c}$, where c is a constant symbol of $\mathcal{L}$ that is not an eigenconstant in $\mathtt{T}''_1$. Since $x \notin \mathtt{FV}(\{\psi\} \cup (\mathcal{UNC}(\mathcal{T}_1) - \{\varphi\}))$, it follows that the root of $\mathtt{T}''_2$ is labelled by $(\mathcal{UNC}(\mathcal{T}_1) - \{\varphi\}), (\varphi)_{x:=c} \Rightarrow \psi$. By a nondegenerate application of the rule $\mathsf{R}_{\exists,l}$, we obtain the $\mathcal{F}^{\text{seq,cut}}_{\mathcal{L},\text{VAR}}$-proof tree $\mathtt{T}''_3 = (\mathtt{T}''_2; (\mathcal{UNC}(\mathcal{T}_1) - \{\varphi\}), (\exists x)\varphi \Rightarrow \psi)$. If x does not occur free in φ, we define $\mathtt{T}''_3 = (\mathtt{T}''_1; (\mathcal{UNC}(\mathcal{T}_1) - \{\varphi\}), (\exists x)\varphi \Rightarrow \psi)$, using a degenerate application of $\mathsf{R}_{\exists,l}$. An application of the thinning rule at the root of $\mathtt{T}''_3$ yields $\mathtt{T}''_4 = (\mathtt{T}''_3; \mathcal{UNC}(\mathcal{T}_0) \cup (\mathcal{UNC}(\mathcal{T}_1) - \{\varphi\}), (\exists x)\varphi \Rightarrow \psi)$. By applying thinning to $\mathtt{T}'$, we obtain $\mathtt{T}'_1 = (\mathtt{T}'; \mathcal{UNC}(\mathcal{T}_0) \cup (\mathcal{UNC}(\mathcal{T}_1) - \{\varphi\}) \Rightarrow \psi, (\exists x)\varphi)$. Finally, an application of the cut rule produces the $\mathcal{F}^{\text{seq,cut}}_{\mathcal{L},\text{VAR}}$-proof tree $(\mathtt{T}''_4, \mathtt{T}'_1; \mathcal{UNC}(\mathcal{T}_0) \cup (\mathcal{UNC}(\mathcal{T}_1) - \{\varphi\}) \Rightarrow \psi)$, which is the desired proof tree.

Now, we treat the cases that arise when $=$ is in $\mathcal{L}$. Let $\mathcal{T} = (\mathtt{T}, M)$ result from an application of $=$-symmetry at the root. We have $\mathcal{T} = (\mathcal{T}_0; t_1 = t_0)$, where $\mathcal{T}_0$ is a first-order natural deduction tree whose root is labelled with the formula $t_0 = t_1$. By inductive hypothesis, we have a proof tree $\mathtt{T}'$ for the sequent $\mathcal{UNC}(\mathcal{T}_0) \Rightarrow t_0 = t_1$. By Example 5.5.10, we can find a proof tree for the sequent $t_0 = t_1, (t_0 = t_1 \rightarrow t_1 = t_0) \Rightarrow t_1 = t_0$. By the rule $\mathsf{R}_{=,l}$ we obtain a proof tree for the sequent $t_0 = t_1 \Rightarrow t_1 = t_0$. Finally, by Theorem 5.5.41, we obtain a proof tree for $\mathcal{UNC}(\mathcal{T}_0) \Rightarrow t_1 = t_0$, which is $\mathcal{UNC}(\mathcal{T}) \Rightarrow t_1 = t_0$.

If $\mathcal{T}$ involves an application of $=$-transitivity at the root, we can write $\mathcal{T} = (\mathcal{T}_0, \mathcal{T}_1; t_0 = t_2)$, where $\mathcal{T}_0, \mathcal{T}_1$ are first-order natural deduction trees whose roots are labelled by $t_0 = t_1$ and $t_1 = t_2$, respectively. By the inductive hypothesis, we have the proof trees $\mathtt{T}'_0, \mathtt{T}'_1$ whose roots are labelled by the sequents $\mathcal{UNC}(\mathcal{T}_0) \Rightarrow t_0 = t_1$ and $\mathcal{UNC}(\mathcal{T}_1) \Rightarrow t_1 = t_2$, respectively. By applying thinning, the proof trees $\mathtt{T}'_0, \mathtt{T}'_1$ yield the proof trees $\mathtt{T}''_0, \mathtt{T}''_1$ for the sequents $\mathcal{UNC}(\mathcal{T}_0), \mathcal{UNC}(\mathcal{T}_1) \Rightarrow t_0 = t_1$ and $\mathcal{UNC}(\mathcal{T}_0), \mathcal{UNC}(\mathcal{T}_1) \Rightarrow t_1 = t_2$, respectively. By an application of the rule $\mathsf{R}_{\wedge,r}$, we get the proof tree $\mathtt{T}'''$ for the sequent $\mathcal{UNC}(\mathcal{T}_0), \mathcal{UNC}(\mathcal{T}_1) \Rightarrow (t_0 = t_1 \wedge t_1 = t_2)$. Example 5.5.10 produces a proof tree $\mathtt{T}^{(iv)}$ for the sequent

$$(t_0 = t_1 \wedge t_1 = t_2), ((t_0 = t_1 \wedge t_1 = t_2) \rightarrow t_0 = t_2) \Rightarrow t_0 = t_2.$$

An application of $\mathsf{R}_{=,l}$ generates the proof tree $\mathtt{T}^{(v)}$ for the sequent $(t_0 = t_1 \wedge t_1 = t_2) \Rightarrow t_0 = t_2$. By combining the proof trees $\mathtt{T}'''$ with $\mathtt{T}^{(v)}$ using Theorem 5.5.41, we obtain a proof tree for $\mathcal{UNC}(\mathcal{T}_0), \mathcal{UNC}(\mathcal{T}_1) \Rightarrow t_0 = t_2$.

Suppose now that we apply f-congruency at the root of $\mathcal{T}$, where f is an n-ary function symbol with $n > 0$. Then,

$$\mathcal{T} = (\mathcal{T}_0, \ldots, \mathcal{T}_{n-1}; f(t_0, \ldots, t_{n-1}) = f(t_n, \ldots, t_{2n-1})),$$

where the root of $\mathcal{T}_i$ is labelled with $t_i = t_{i+n}$ for $0 \leq i \leq n-1$. By the inductive hypothesis, we can construct $\mathcal{F}^{\text{seq,cut}}_{\mathcal{L},V}$-proof trees $\mathtt{T}'_i$ for the sequents $\mathcal{UNC}(\mathcal{T}_i) \Rightarrow t_i = t_{i+n}$, where $0 \leq i \leq n-1$. By applying thinning, we obtain the proof trees $\mathtt{T}''_i$ for $\mathcal{UNC}(\mathcal{T}_0), \ldots, \mathcal{UNC}(\mathcal{T}_{n-1}) \Rightarrow t_i = t_{i+n}$, for $0 \leq i \leq n-1$. By the argument presented in Example 5.5.34, we obtain a $\mathcal{F}^{\text{seq,cut}}_{\mathcal{L},V}$-proof tree $\mathtt{T}'''$ for $\mathcal{UNC}(\mathcal{T}_0), \ldots, \mathcal{UNC}(\mathcal{T}_{n-1}) \Rightarrow (t_0 = t_n \wedge \cdots \wedge t_{n-1} = t_{2n-1})$. By applying "modus ponens" for sequents (discussed in Example 5.5.10), we have a proof tree $\mathtt{T}^{(iv)}$ for the sequent

$$\begin{aligned}&(t_0 = t_n \wedge \cdots \wedge t_{n-1} = t_{2n-1}),\\ &((t_0 = t_n \wedge \cdots \wedge t_{n-1} = t_{2n-1}) \to f(t_0, \ldots, t_{n-1}) = f(t_n, \ldots, t_{2n-1}))\\ &\Rightarrow f(t_0, \ldots, t_{n-1}) = f(t_n, \ldots, t_{2n-1}).\end{aligned}$$

By the rule $\mathsf{R}_{=,l}$, we can derive from $\mathtt{T}^{(iv)}$ the proof tree $\mathtt{T}^{(v)}$ for

$$(t_0 = t_n \wedge \cdots \wedge t_{n-1} = t_{2n-1}) \Rightarrow f(t_0, \ldots, t_{n-1}) = f(t_n, \ldots, t_{2n-1}).$$

Finally, by Theorem 5.5.41, we obtain a proof tree for the sequent

$$\mathcal{UNC}(\mathcal{T}_0), \ldots, \mathcal{UNC}(\mathcal{T}_{n-1}) \Rightarrow f(t_0, \ldots, t_{n-1}) = f(t_n, \ldots, t_{2n-1}).$$

Let $\mathcal{T}$ be obtained by applying P-congruency at the root, where P is an n-ary relation symbol with $n > 0$. We have $\mathcal{T} = (\mathcal{T}_0, \ldots, \mathcal{T}_n; P(t_n, \ldots, t_{2n-1}))$, where the root of $\mathcal{T}_i$ is labelled by $t_i = t_{n+i}$, for $0 \le i \le n-1$ and the root of $\mathcal{T}_n$ is labelled by $P(t_0, \ldots, t_{n-1})$. By the inductive hypothesis, we can construct $\mathcal{F}^{\text{seq,cut}}_{\mathcal{L},V}$-proof trees $\mathtt{T}_0, \ldots, \mathtt{T}_n$ for the sequents $\mathcal{UNC}(\mathcal{T}_0) \Rightarrow t_0 = t_n, \ldots, \mathcal{UNC}(\mathcal{T}_{n-1}) \Rightarrow t_{n-1} = t_{2n-1}$ and $\mathcal{UNC}(\mathcal{T}_n) \Rightarrow P(t_0, \ldots, t_{n-1})$, respectively.

By applying thinning, we obtain proof trees $\mathtt{T}'_0, \ldots, \mathtt{T}'_n$ for the sequents

$$\begin{gathered}\mathcal{UNC}(\mathcal{T}_0), \ldots, \mathcal{UNC}(\mathcal{T}_n) \Rightarrow t_0 = t_n\\ \vdots\\ \mathcal{UNC}(\mathcal{T}_0), \ldots, \mathcal{UNC}(\mathcal{T}_n) \Rightarrow t_{n-1} = t_{2n-1}\\ \mathcal{UNC}(\mathcal{T}_0), \ldots, \mathcal{UNC}(\mathcal{T}_n) \Rightarrow P(t_0, \ldots, t_{n-1}),\end{gathered}$$

respectively. By applying the argument from Example 5.5.34, we can construct a proof tree $\mathtt{T}''$ starting from $\mathtt{T}'_0, \ldots, \mathtt{T}'_{n-1}$ for the sequent

$$\mathcal{UNC}(\mathcal{T}_0), \ldots, \mathcal{UNC}(\mathcal{T}_n) \Rightarrow (t_0 = t_n \wedge \cdots \wedge t_{n-1} = t_{2n-1}).$$

By using "modus ponens" for sequents, we construct a proof tree $\mathtt{T}'''$ for the sequent

$$\begin{aligned}&(t_0 = t_n \wedge \cdots \wedge t_{n-1} = t_{2n-1}),\\ &((t_0 = t_n \wedge \cdots \wedge t_{n-1} = t_{2n-1}) \to (P(t_0, \ldots, t_{n-1})\\ &\quad \leftrightarrow P(t_n, \ldots, t_{2n-1})))\\ &\Rightarrow (P(t_0, \ldots, t_{n-1}) \leftrightarrow P(t_n, \ldots, t_{2n-1})).\end{aligned}$$

Starting from $\mathtt{T}'''$ and applying $\mathsf{R}_{=,l}$, we obtain the proof tree $\mathtt{T}^{(iv)}$ for the sequent

$$(t_0 = t_n \wedge \cdots \wedge t_{n-1} = t_{2n-1}) \Rightarrow (P(t_0, \ldots, t_{n-1}) \leftrightarrow P(t_n, \ldots, t_{2n-1})).$$

By applying Theorem 5.5.41 to $\mathtt{T}''$ and $\mathtt{T}^{(iv)}$, we obtain a proof tree $\mathtt{T}^{(v)}$ for the sequent

$$\mathcal{UNC}(\mathcal{T}_0), \ldots, \mathcal{UNC}(\mathcal{T}_n) \Rightarrow (P(t_0, \ldots, t_{n-1}) \leftrightarrow P(t_n, \ldots, t_{2n-1})).$$

By applying the argument from Example 5.5.34 to $\mathtt{T}'_n$ and $\mathtt{T}^{(v)}$, we obtain a proof tree $\mathtt{T}^{(vi)}$ for the sequent

$$\begin{aligned}&\mathcal{UNC}(\mathcal{T}_0), \ldots, \mathcal{UNC}(\mathcal{T}_n)\\ &\quad \Rightarrow (P(t_0, \ldots, t_{n-1}) \wedge (P(t_0, \ldots, t_{n-1}) \leftrightarrow P(t_n, \ldots, t_{2n-1}))).\end{aligned}$$

A proof tree $\mathtt{T}^{(vii)}$ for the sequent

$$\begin{aligned}&(P(t_0, \ldots, t_{n-1}) \wedge (P(t_0, \ldots, t_{n-1})\\ &\quad \leftrightarrow P(t_n, \ldots, t_{2n-1})) \Rightarrow P(t_n, \ldots, t_{2n-1})\end{aligned}$$

is obtained using Example 5.5.11. Finally, we apply Theorem 5.5.41 to the proof trees $\mathtt{T}^{(vi)}$ and $\mathtt{T}^{(vii)}$ to obtain a proof tree for the sequent $\mathcal{UNC}(\mathcal{T}_0), \ldots, \mathcal{UNC}(\mathcal{T}_n) \Rightarrow (P(t_n, \ldots, t_{2n-1})$. □

Corollary 5.7.13. *Let $\mathcal{L}$ be a first-order language with infinitely many constant symbols and let Γ be a set of $\mathcal{L}$-formulas. There is a syntactic algorithm that takes as input a first-order natural deduction tree $\mathcal{T} = (\mathtt{T}, M)$ in $FONDT^{\mathcal{L}}$ with $\mathcal{UNC}(\mathcal{T}) \subseteq \Gamma$ and produces as output an $\mathcal{F}^{\text{seq,cut}}_{\mathcal{L},VAR}$-proof tree of the sequent $\Gamma \Rightarrow \mathtt{T}(\lambda)$.*

Proof. Applying Theorem 5.7.12, we can transform effectively the natural deduction tree $\mathcal{T} = (\mathtt{T}, M)$ into an $\mathcal{F}^{\text{seq,cut}}_{\mathcal{L},V}$-proof tree for the

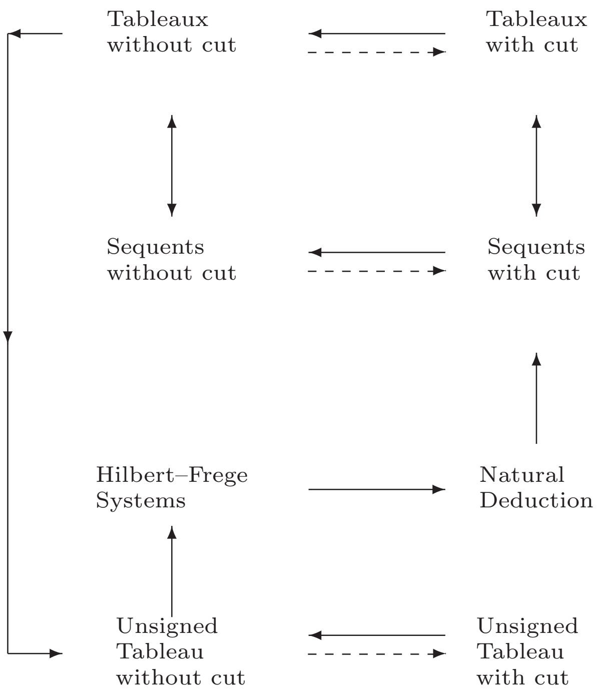


Fig. 5.36. General Layout of Transformations

sequent $\mathcal{UNC}(\mathcal{T}) \Rightarrow \mathtt{T}(\lambda)$. An application of thinning yields a proof tree for $\Gamma \Rightarrow \mathtt{T}(\lambda)$. $\square$

5.7.3 *Closing the Circle*

We now close the circle; that is, using results we have proven so far, we show how to transform a formal proof that $\Gamma \models \varphi$ in one system into a formal in another one. The general layout of these transformations is similar to the one used in propositional logic and is shown in Figure 5.36. However, in first-order logic, we need to consider the language and the set of free variables used in the formal proofs. If Γ and φ are respectively a set of $\mathcal{L}$-formulas and an $\mathcal{L}$-formula, the language which allows us to perform these transformations is $\mathcal{L}^c$. Whenever the set of free variables is relevant, we use the set of all variables VAR.

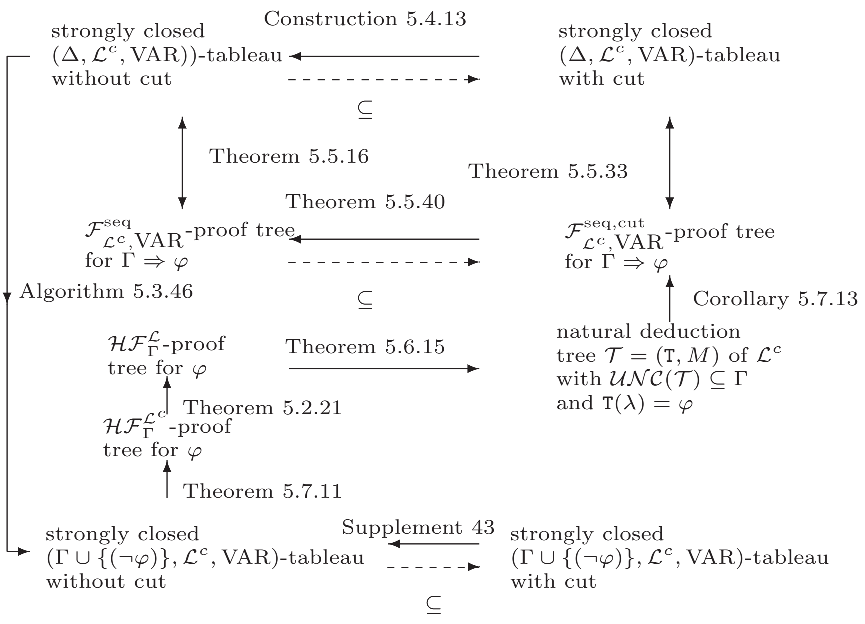

strongly closed $(\Delta, \mathcal{L}^c, \mathrm{VAR}))$-tableau without cut
Construction 5.4.13
strongly closed $(\Delta, \mathcal{L}^c, \mathrm{VAR})$-tableau with cut
$\subseteq$
Theorem 5.5.16
Theorem 5.5.33
Theorem 5.5.40
$\mathcal{F}^{\mathrm{seq}}_{\mathcal{L}^c,\mathrm{VAR}}$-proof tree for $\Gamma \Rightarrow \varphi$
$\mathcal{F}^{\mathrm{seq,cut}}_{\mathcal{L}^c,\mathrm{VAR}}$-proof tree for $\Gamma \Rightarrow \varphi$
Algorithm 5.3.46
$\subseteq$
Corollary 5.7.13
$\mathcal{HF}^{\mathcal{L}}_{\Gamma}$-proof tree for φ
Theorem 5.6.15
natural deduction tree $\mathcal{T} = (\mathtt{T}, M)$ of $\mathcal{L}^c$ with $\mathcal{UNC}(\mathcal{T}) \subseteq \Gamma$ and $\mathtt{T}(\lambda) = \varphi$
Theorem 5.2.21
$\mathcal{HF}^{\mathcal{L}^c}_{\Gamma}$-proof tree for φ
Theorem 5.7.11
strongly closed $(\Gamma \cup \{(\neg\varphi)\}, \mathcal{L}^c, \mathrm{VAR})$-tableau without cut
Supplement 43
strongly closed $(\Gamma \cup \{(\neg\varphi)\}, \mathcal{L}^c, \mathrm{VAR})$-tableau with cut
$\subseteq$

Fig. 5.37. Objects Being Transformed

The specific objects being transformed as well as the results involved are shown in Figure 5.37, where Δ is the set of signed formulas $\Delta = \{\mathbf{T}\psi \mid \psi \in \Gamma\} \cup \{\mathbf{F}\varphi\}$.

5.8 First-Order Resolution

We introduce now the first-order analog of the notion of clause from propositional logic.

Definition 5.8.1. Let $\mathcal{L}$ be a first-order language. An $\mathcal{L}$-*clause* is a finite set of literals of $\mathcal{L}$. If C is an $\mathcal{L}$-clause for some first-order language $\mathcal{L}$, then C is a *clause*.

We denote by $\mathsf{CLAUSES}_{\mathcal{L}}$ the set of all $\mathcal{L}$-clauses.

A set of clauses $\mathcal{C}$ is *admissible* if there is a first-order language $\mathcal{L}$ such that $\mathcal{C} \subseteq \mathsf{CLAUSES}_{\mathcal{L}}$.

If a clause C consists of ground literals, then we say that C is a *ground clause*.

As in propositional logic, the empty clause will be called "box" and denoted by $\square$. ∎

We use the letters C, D, E, with or without subscripts, to denote clauses and $\mathcal{C}, \mathcal{D}, \mathcal{E}$ to denote sets of clauses.

If C is a clause, let $\mathtt{V}(C)$ be the set of all variables that occur in the formulas of C. For a set of clauses $\mathcal{C}$, we define the sets $\mathtt{V}(\mathcal{C}) = \bigcup_{C \in \mathcal{C}} \mathtt{V}(C)$ and $\mathrm{LIT}(\mathcal{C}) = \bigcup \mathcal{C}$. In other words, $\mathrm{LIT}(\mathcal{C})$ is the set of literals which appear in some clause of $\mathcal{C}$.

For first-order logic, we maintain the same notational differentiation between the empty clause denoted by $\square$ and the empty set of clauses denoted by $\emptyset$.

When we write clauses, we will systematically omit outer parentheses for a negative literal.

The next definition introduces several kinds of clauses we will be using.

Definition 5.8.2. A clause is a

- *tautologous clause* if it contains both a literal and its complement;
- *Horn clause* if it contains at most one positive literal;
- *positive clause* if it contains only positive literals;
- *negative clause* if it contains only negative literals;
- *unit clause* if it consists of a single literal.

∎

Definition 5.8.3. Let $\mathcal{C}$ be a set of clauses of first-order logic. A set of clauses $\mathcal{C}_0$ of propositional logic is a *propositional form* for $\mathcal{C}$ if there is an inter-substitution s such that $s(\mathcal{C}_0) = \mathcal{C}$. ∎

Using the usual extension of functions to sets and collections of sets, $s(\mathcal{C}_0)$ is the set of clauses $\{s(C) \mid C \in \mathcal{C}_0\}$. Note that $s(C)$ is already defined because C is a set of propositional formulas.

Example 5.8.4. Let $\mathcal{C} = \{\{R(x), R(y)\}, \{(\neg R(x)), R(z)\}\}$. Observe that all of the following sets of propositional clauses are propositional forms for $\mathcal{C}$:

$$\mathcal{C}_0 = \{\{p_0, p_1\}, \{(\neg p_0), p_2\}\}$$

$$\mathcal{C}_1 = \{\{p_0, p_1, p_2\}, \{p_3, p_4\}\}$$

$$\mathcal{C}_2 = \{\{p_0, p_1\}, \{(\neg p_3), p_4\}\}$$

For example, for $\mathcal{C}_1$, an adequate inter-substitution is any inter-substitution s such that $s(p_0) = s(p_1) = R(x)$, $s(p_2) = R(y)$, $s(p_3) = (\neg R(x))$ and $s(p_4) = R(z)$. Clearly, this is not an injective substitution. Observe that $\mathcal{C}_0$ is, in a certain sense, the "best" propositional form for $\mathcal{C}$ because it uses the smallest number of propositional variables each of which is mapped to a positive literal. ▯

Definition 5.8.5. Let $\mathcal{C}$ be a set of clauses of first-order logic. If s a prime injective inter-substitution and $\mathcal{C}_0$ is a set of propositional clauses such that $s(\mathcal{C}_0) = \mathcal{C}$, then we say that $\mathcal{C}_0$ is a *fundamental propositional form for* $\mathcal{C}$. ▯

Recall that a prime inter-substitution maps propositional variables into atomic or quantified formulas. In the case of clauses, this amounts to mapping propositional variables just to atomic formulas.

Example 5.8.6. For the set of clauses $\mathcal{C}$ considered in Example 5.8.4, the inter-substitution s defined by $s(p_0) = R(x)$, $s(p_1) = R(y)$, $s(p_2) = R(z)$ and $s(p_k) = R(x_{k_0+k})$, for $k \geq 3$, where $\{x, y, z\} \subseteq \{x_0, \ldots, x_{k_0-1}\}$ is an injective, prime inter-substitution with $s(\mathcal{C}_0) = \mathcal{C}$. Thus, $\mathcal{C}_0$ is a fundamental propositional form for $\mathcal{C}$. ▯

Theorem 5.8.7. *Let $\mathcal{C}$ be a set of clauses of first-order logic. There is a fundamental propositional form $\mathcal{C}_0$ for $\mathcal{C}$.*

Proof. The argument is similar to the one of Theorem 4.8.7 and it is left to the reader. □

Definition 5.8.8. Let C be an $\mathcal{L}$-clause, $\mathcal{A}$ be an $\mathcal{L}$-structure and σ be an assignment over $\mathcal{A}$; $(\mathcal{A}, \sigma)$ *satisfies* C if $(\mathcal{A}, \sigma) \models \ell$ for some literal $\ell \in C$. This will be denoted by $(\mathcal{A}, \sigma) \models C$.

The pair $(\mathcal{A}, \sigma)$ satisfies a set of $\mathcal{L}$-clauses $\mathcal{C}$ if $(\mathcal{A}, \sigma)$ satisfies every clause in $\mathcal{C}$. This will be denoted by $(\mathcal{A}, \sigma) \models \mathcal{C}$.

An $\mathcal{L}$-clause is *satisfiable* if there is an $\mathcal{L}$-structure $\mathcal{A}$ and an assignment σ over $\mathcal{A}$ such that $(\mathcal{A}, \sigma) \models C$; likewise, a set of $\mathcal{L}$-clauses is *satisfiable* if there is such a pair $(\mathcal{A}, \sigma)$ that satisfies the set of clauses.

An $\mathcal{L}$-structure $\mathcal{A}$ is a *model for an $\mathcal{L}$-clause* C if $(\mathcal{A}, \sigma) \models C$ for all $\sigma \in \mathrm{ASSIGN}_{\mathcal{A}}$. This is denoted by $\mathcal{A} \models C$.

An $\mathcal{L}$-structure $\mathcal{A}$ is a *model for a set of $\mathcal{L}$-clauses* $\mathcal{C}$ if $\mathcal{A} \models C$ for all $C \in \mathcal{C}$. This is denoted by $\mathcal{A} \models \mathcal{C}$. ▯

As in propositional logic, note that the notion of satisfaction of a set of formulas introduced in Definition 4.5.9 is quite different from the notion of satisfying a clause given above. We rely on the context to differentiate between these concepts.

Note that every nonempty clause is satisfiable.

We shall prove later that satisfiability and having a model are independent of the language used.

The following theorem gives the basic properties of clause satisfaction.

Theorem 5.8.9. *Let $\mathcal{L}$ be a first-order language, $\mathcal{C}$ and $\mathcal{D}$ be sets of $\mathcal{L}$-clauses, and let $\mathcal{A}$ be an $\mathcal{L}$-structure and $\sigma \in \mathrm{ASSIGN}_{\mathcal{A}}$.*

(1) *If $\mathcal{C} \subseteq \mathcal{D}$, then $(\mathcal{A}, \sigma)$ satisfies $\mathcal{D}$ implies that $(\mathcal{A}, \sigma)$ satisfies $\mathcal{C}$.*
(2) *If $\mathcal{C} \subseteq \mathcal{D}$, then if $\mathcal{D}$ is satisfiable, $\mathcal{C}$ is satisfiable and if $\mathcal{C}$ is unsatisfiable, $\mathcal{D}$ is unsatisfiable.*
(3) *$\square$ is an unsatisfiable clause and $\{\square\}$ is an unsatisfiable set of clauses.*
(4) *$\emptyset$ is a satisfiable set of clauses.*
(5) *If $\mathcal{C}$ contains $\square$, then $\mathcal{C}$ is unsatisfiable.*
(6) *If C is a tautologous clause, then $(\mathcal{A}, \sigma)$ satisfies C.*

Proof. The argument is similar to that of Theorem 3.8.6. $\square$

The next result deals with the existence of models for sets of clauses.

Theorem 5.8.10. *Let $\mathcal{L}$ be a first-order language and $\mathcal{C}$ and $\mathcal{D}$ be sets of $\mathcal{L}$-clauses.*

(1) *If $\mathcal{C} \subseteq \mathcal{D}$ and $\mathcal{D}$ has a model, then $\mathcal{C}$ has the same model.*
(2) *$\square$ is a clause that has no model and $\{\square\}$ is a set of clauses that has no model.*
(3) *$\emptyset$ is a set of clauses that has a model.*
(4) *If $\mathcal{C}$ contains $\square$, then $\mathcal{C}$ has no model.*

Proof. The proof is straightforward and is left to the reader. $\square$

Definition 5.8.11. Let $C = \{\ell_0, \ldots, \ell_{n-1}\}$ be a nonempty clause, where the literals are listed in the standard order of formulas.

The *disjunctive normal form formula* φ_C *that represents* C is

$$\varphi_C = (\ell_0 \vee \cdots \vee \ell_{n-1}).$$

If $\mathcal{C}$ is a set of clauses such that $\square \notin \mathcal{C}$, then the set of formulas $\Gamma_{\mathcal{C}}$ is the set $\{\varphi_C \mid C \in \mathcal{C}\}$.

Let

$$\varphi = \bigwedge_{i=0}^{n-1} (\ell_{i0} \vee \cdots \vee \ell_{im_i-1})$$

where each ℓ_{ij} is a literal, be a formula in conjunctive normal form. The *clause set associated with* φ is

$$\mathcal{C}_\varphi = \{\{\ell_{i0}, \ldots, \ell_{im_i-1}\} \mid 0 \leq i \leq n-1\}.$$

If Γ is a set of $\mathcal{L}$-formulas in conjunctive normal form, then the set of clauses $\mathcal{C}_\Gamma$ is $\bigcup\{\mathcal{C}_\varphi \mid \varphi \in \Gamma\}$. $\square$

Note that if $\mathcal{C}$ is a set of $\mathcal{L}$-clauses such that $\square \notin \mathcal{C}$, then $\Gamma_{\mathcal{C}}$ is a set of $\mathcal{L}$-formulas and if φ is an $\mathcal{L}$-formula in conjunctive normal form, then $\mathcal{C}_\varphi$ is a set of $\mathcal{L}$-clauses.

We leave to the reader to verify that

$$\mathcal{C}_{\Gamma_{\mathcal{C}}} = \mathcal{C} \tag{5.4}$$

for any set of clauses $\mathcal{C}$ such that $\square \notin \mathcal{C}$.

Note also that if φ is in conjunctive normal form, then $\mathcal{C}_\varphi$ is a set of Horn clauses if and only if φ is a Horn formula.

The following result allows us to translate satisfiability of formulas and the existence of models for formulas in conjunctive normal form into the corresponding properties of sets of clauses.

Theorem 5.8.12. *Let* $\mathcal{L}$ *be a first-order language,* $\mathcal{A}$ *be an* $\mathcal{L}$*-structure and* $\sigma \in \mathrm{ASSIGN}_{\mathcal{A}}$.

If φ *is an* $\mathcal{L}$*-formula in conjunctive normal form, then* $(\mathcal{A}, \sigma) \models \varphi$ *if and only if* $(\mathcal{A}, \sigma) \models \mathcal{C}_\varphi$*. Further,* $\mathcal{A} \models \varphi$ *if and only if* $\mathcal{A} \models \mathcal{C}_\varphi$.

If $\mathcal{C}$ *is a set of* $\mathcal{L}$*-clauses such that* $\square \notin \mathcal{C}$ *, then* $(\mathcal{A}, \sigma) \models \mathcal{C}$ *if and only if* $(\mathcal{A}, \sigma) \models \Gamma_{\mathcal{C}}$*. Also,* $\mathcal{A} \models \mathcal{C}$ *if and only if* $\mathcal{A} \models \Gamma_{\mathcal{C}}$.

Proof. The argument is straightforward. $\square$

Corollary 5.8.13. *If Γ is a set of $\mathcal{L}$-formulas in conjunctive normal form, then Γ is satisfiable (has a model) if and only if $\mathcal{C}_\Gamma$ is satisfiable (has a model) as a set of $\mathcal{L}$-clauses.*

If $\mathcal{C}$ is a set of $\mathcal{L}$-clauses that does not contain $\square$, then $\mathcal{C}$ is satisfiable (has a model) as a set of $\mathcal{L}$-clauses if and only if $\Gamma_\mathcal{C}$ is satisfiable (has a model).

Proof. The proof is immediate and is left to the reader. $\square$

Corollary 5.8.14. *If $\mathcal{C}$ is a set of ground $\mathcal{L}$-clauses, then $\mathcal{C}$ has a model if and only if $\mathcal{C}$ is satisfiable.*

Proof. If $\square \notin \mathcal{C}$, this follows from Corollary 5.8.13 and the fact that a set of closed formulas has a model if and only if it is satisfiable. If $\square \in \mathcal{C}$, then $\mathcal{C}$ is not satisfiable and has no model. $\square$

The next corollary establishes the independence of satisfiability (existence of a model) for sets of clauses relative to the first-order language in which satisfiability (having a model) is considered.

Corollary 5.8.15. *Let $\mathcal{L}$ and $\mathcal{L}'$ be two first-order languages such that $\mathcal{C}$ is both a set of $\mathcal{L}$-clauses and a set of $\mathcal{L}'$-clauses. Then, $\mathcal{C}$ is satisfiable (has a model) as a set of $\mathcal{L}$-clauses if and only if $\mathcal{C}$ is satisfiable (has a model) as a set of $\mathcal{L}'$-clauses.*

Proof. If $\square \notin \mathcal{C}$, the statement follows from Corollary 5.8.13. If $\square \in \mathcal{C}$, then $\mathcal{C}$ is not satisfiable and has no model both as a set of $\mathcal{L}$-clauses and as a set of $\mathcal{L}'$-clauses. $\square$

In first-order logic, we have the same central issue as in propositional logic, namely, determining whether a set of $\mathcal{L}$-formulas Γ logically implies an $\mathcal{L}$-formula φ. We will show that this problem is equivalent to the nonexistence of a model for a set of clauses.

We discuss first the case when Γ is a set of sentences and φ is a sentence. As we saw, by the first part of Theorem 4.5.52, we have $\Gamma \models \varphi$ if and only if the set $\Gamma \cup \{(\neg\varphi)\}$ is unsatisfiable. Let Γ' be a Skolemization of the set $\Gamma \cup \{(\neg\varphi)\}$, that is, a set of universal $\mathcal{L}'$-formulas, where $\mathcal{L}'$ is the extension of $\mathcal{L}$ produced by the Skolemization process. Then, by Corollary 4.9.21, $\Gamma \cup \{(\neg\varphi)\}$ is unsatisfiable if and only if Γ' is unsatisfiable. As we observed in the description of the Skolemization algorithm, the set of free variables of a Skolemized formula is the same as the set of free variables of the initial formula.

Thus, the formulas in Γ' are $\mathcal{L}'$-sentences and, therefore, by Corollary 4.5.32, the set Γ' is unsatisfiable if and only if it has no model. Let Γ'' be the set of matrices of formulas in Γ'. The set Γ'' consists of quantifier-free formulas. By repeated application of the first part of Theorem 4.5.58, the set Γ' has no model if and only if Γ'' has no model. Let Γ''' be obtained from Γ'' by replacing each formula in Γ'' by an equivalent formula in conjunctive normal form, using the method of Theorem 4.9.2. Again, Γ'' has no model if and only if Γ''' has no model. Finally, as follows from Corollary 5.8.13, Γ''' has no model if and only if the collection of clauses $\mathcal{C}_{\Gamma'''}$ has no model. Thus, we have shown that $\Gamma \models \varphi$ is equivalent to the nonexistence of a model for the set of clauses $\mathcal{C}_{\Gamma'''}$.

Let now Γ and φ be a set of $\mathcal{L}$-formulas and an $\mathcal{L}$-formula which are not necessarily sentences. By Theorem 4.6.14, there is a substitution s that replaces all variables by constant symbols such that $\Gamma \models \varphi$ if and only if $\mathrm{FVSubst}(s, \Gamma) \models \mathrm{FVSubst}(s, \varphi)$. Since $\mathrm{FVSubst}(s, \Gamma)$ is a set of $\mathcal{L}^c$-sentences and $\mathrm{FVSubst}(s, \varphi)$ is an $\mathcal{L}^c$-sentence, we have reduced this case to the previous one.

The next examples illustrate the transformation of the logical implication problem into the nonexistence of a model for a set of clauses, which we outlined above.

Example 5.8.16. Let $\varphi = ((\forall x)P(x) \to (\exists x)P(x))$. To prove the logical validity of φ, observe that this amounts to $\emptyset \models \varphi$, which is equivalent to $\{(\neg\varphi)\}$ being unsatisfiable. The formula $(\neg\varphi)$ is logically equivalent to the formula $(\neg(\exists x)(\exists y)(P(x) \to P(y)))$, which is logically equivalent to the formula $(\forall x)(\forall y)(\neg(P(x) \to P(y)))$, which is in prenex normal form and also in Skolem normal form. Thus, φ is logically valid if and only if the matrix $\psi = (\neg(P(x) \to P(y)))$ has no model. A conjunctive normal form of ψ is $(P(x) \wedge (\neg P(y)))$, which produces the set of clauses $\{\{P(x)\}, \{(\neg P(y))\}\}$, so φ is logically valid if and only if this set of clauses has no model. $\square$

Example 5.8.17. Let

$$\varphi = (\forall x_0)(\forall x_1)(((\forall x_2)(f(x_0, x_2) = x_2) \\ \wedge(\forall x_3)(f(x_3, x_1) = x_3)) \to (x_0 = x_1)),$$

This formula asserts that if x_0 is a left unit of a binary operation on a set and x_1 is a right unit of the same operation, then x_0 and x_1

are the same. The formula $(\neg\varphi)$ is logically equivalent to the prenex normal form

$$(\exists x_0)(\exists x_1)(\forall x_2)(\forall x_3)(\neg(((f(x_0,x_2)=x_2)$$
$$\wedge(f(x_3,x_1)=x_3))\to(x_0=x_1))),$$

as the reader can easily verify. The Skolemization of this formula requires the introduction of two constant symbols c_0, c_1 and produces the formula

$$(\forall x_2)(\forall x_3)(\neg(((f(c_0,x_2)=x_2)\wedge(f(x_3,c_1)=x_3))\to(c_0=c_1))),$$

whose matrix $(\neg(((f(c_0,x_2)=x_2)\wedge(f(x_3,c_1)=x_3))\to(c_0=c_1)))$ is equivalent to the conjunctive normal form $((f(c_0,x_2)=x_2)\wedge(f(x_3,c_1)=x_3)\wedge(\neg(c_0=c_1)))$. Thus, φ is logically valid if and only if the set of clauses

$$\mathcal{C}=\{\{(f(c_0,x_2)=x_2)\},\{(f(x_3,c_1)=x_3)\},\{(\neg(c_0=c_1))\}\}$$

has no model. □

Example 5.8.18. The logical validity of the formula $\varphi = (\exists x)(P(x) \to (\forall x)P(x))$ is equivalent to the unsatisfiability of $(\neg\varphi)$. A prenex normal form of $(\neg\varphi)$ is $(\forall x)(\exists y)(\neg(P(x)\to P(y)))$. A Skolemization of this formula is $(\forall x)(\neg(P(x)\to P(f(x))))$ whose matrix has the conjunctive normal form $(P(x)\wedge(\neg P(f(x))))$. Note that this formula was obtained by introducing a unary function symbol f. The logical validity of φ is equivalent to the nonexistence of a model for the set of clauses $\{\{P(x)\},\{(\neg P(f(x)))\}\}$. □

Example 5.8.19. To construct a set of clauses corresponding to the formula $\varphi = ((\exists x)(\forall y)R(x,y) \to (\forall y)(\exists x)R(x,y))$, we begin by constructing a prenex normal form of the formula $(\neg\varphi)$. After straightforward transformations, this prenex normal form is $(\exists x)(\exists y)(\forall z)(\forall w)(\neg(R(x,z)\to R(w,y)))$. A Skolem normal form of this formula is $(\forall z)(\forall w)(\neg(R(c_0,z)\to R(w,c_1)))$, where c_0, c_1 are two constant symbols. A conjunctive normal form of the matrix of this formula is $(R(c_0,z)\wedge(\neg R(w,c_1)))$, which yields the set of clauses $\{\{R(c_0,z)\},\{(\neg R(w,c_1))\}\}$. □

Example 5.8.20. In this example, we consider two formulas

$$\varphi = ((\forall x_0)(\forall x_1)R(x_0, x_1) \to R(f(x_0), g(x_1)))$$
$$\psi = ((\forall x_0)(\forall x_1)R(x_0, x_1) \to R(f(x_1), g(x_0))),$$

where R is a binary relation symbol and f, g are unary function symbols. By previous discussion, φ is logically valid if and only if the formula $((\forall x_0)(\forall x_1)R(x_0, x_1) \to R(f(d_0), g(d_1)))$ is logically valid, where we substituted d_0 and d_1 for the free occurrences of x_0 and x_1, respectively. Thus, φ is logically valid if and only if the sentence $(\neg((\forall x_0)(\forall x_1)R(x_0, x_1) \to R(f(d_0), g(d_1))))$ is unsatisfiable, which is equivalent to the unsatisfiability of the formula

$$\theta = (\forall x_0)(\forall x_1)(\neg(R(x_0, x_1) \to R(f(d_0), g(d_1)))),$$

which is in Skolem normal form. Since θ is a sentence, the logical validity of φ is equivalent to the nonexistence of a model for θ. A conjunctive normal form of the matrix of θ is $((\neg R(x_0, x_1)) \wedge R(f(d_0), g(d_1)))$. The corresponding set of clauses is $\{\{(\neg R(x_0, x_1))\}, \{R(f(d_0), g(d_1))\}\}$.

The set of clauses for ψ is identical if we substitute d_0 for x_1 and d_1 for x_0. □

Definition 5.8.21. Let $\mathcal{C}$ be a set of $\mathcal{L}$-clauses, where $\mathcal{L}$ is a first-order language. A *renaming for* $\mathcal{C}$ is an injective substitution $s : \mathtt{V}(\mathcal{C}) \to \mathrm{VAR}$.

If $\mathcal{C} = \{C\}$, then we say that s is a *renaming for* C. □

In Definition 5.8.21, we identified variables with sequences of variables of length one. The renaming is extended as usual to sets of clauses and we denote by $\overline{s}(\mathcal{C})$ the collection $\{\overline{s}(C) \mid C \in \mathcal{C}\}$. The set $\overline{s}(\mathcal{C})$ will be referred to as a *renaming* of $\mathcal{C}$.

Theorem 5.8.22. *Let $\mathcal{C}, \mathcal{C}', \mathcal{C}''$ be sets of $\mathcal{L}$-clauses, where $\mathcal{L}$ is a first-order language. If $\mathcal{C}'$ is a renaming of $\mathcal{C}$, then $\mathcal{C}$ is a renaming of $\mathcal{C}'$. Further, if $\mathcal{C}''$ is a renaming of $\mathcal{C}'$ and $\mathcal{C}'$ is a renaming of $\mathcal{C}$, then $\mathcal{C}''$ is a renaming of $\mathcal{C}$.*

Proof. Let $s : \mathtt{V}(\mathcal{C}) \longrightarrow \mathrm{VAR}$ be a renaming such that $\mathcal{C}' = \overline{s}(\mathcal{C})$. Observe that the range of s is $\mathtt{V}(\mathcal{C}')$. The injectivity of s allows us to define the renaming s' :

$\mathtt{V}(\mathcal{C}') \longrightarrow \mathrm{VAR}$ as $s'(z) = x$ whenever $s(x) = z$. If ℓ is a literal in a clause C of $\mathcal{C}$, we have $\overline{s'}(\overline{s}(\ell)) = \overline{s' * s}(\ell) = \ell$, which shows that $\overline{s'}(\mathcal{C}') = \mathcal{C}$.

Suppose $s_{01} : \mathtt{V}(\mathcal{C}) \longrightarrow \mathrm{VAR}$ is a renaming such that $\mathcal{C}' = \overline{s_{01}}(\mathcal{C})$ and $s_{12} : \mathtt{V}(\mathcal{C}') \longrightarrow \mathrm{VAR}$ is a renaming such that $\mathcal{C}'' = \overline{s_{12}}(\mathcal{C}')$. Since $\mathrm{Ran}(s_{01}) = \mathtt{V}(\mathcal{C}')$, we can define $s_{02} : \mathtt{V}(\mathcal{C}) \to \mathrm{VAR}$ by $s_{02} = s_{12} \circ s_{01}$. Observe that for all $x \in \mathtt{V}(\mathcal{C})$, $s_{02}(x) = s_{12} * s_{01}(x)$. By Theorem 1.2.19, we have $\overline{s_{02}}(\mathcal{C}) = \overline{s_{12} * s_{01}}(\mathcal{C}) = \overline{s_{12}}(\overline{s_{01}}(\mathcal{C})) = \overline{s_{12}}(\mathcal{C}') = \mathcal{C}''$. $\square$

Theorem 5.8.23. *Let $\mathcal{L}$ be a first-order language, s be an $\mathcal{L}$-substitution, $\mathcal{C}$ be a set of $\mathcal{L}$-clauses and $\mathcal{A}$ be an $\mathcal{L}$-structure. If $\mathcal{A} \models \mathcal{C}$, then $\mathcal{A} \models \bar{s}(\mathcal{C})$.*

Proof. Suppose that $\mathcal{A}$ is a model of $\mathcal{C}$ and $\sigma \in \mathrm{ASSIGN}_{\mathcal{A}}$. Under this assumption, we have $(\mathcal{A}, \sigma^{\mathcal{A}} \circ s) \models \mathcal{C}$. Therefore, for every clause $C \in \mathcal{C}$, there is $\ell \in C$ such that $(\mathcal{A}, \sigma^{\mathcal{A}} \circ s) \models \ell$. By Corollary 4.6.5, we have $(\mathcal{A}, \sigma) \models \mathrm{FVSubst}(s, \ell)$. Since ℓ contains no bound variables, we have $\mathrm{FVSubst}(s, \ell) = \bar{s}(\ell)$. Consequently, $(\mathcal{A}, \sigma) \models \bar{s}(C)$ for every $C \in \mathcal{C}$, so $(\mathcal{A}, \sigma) \models \bar{s}(\mathcal{C})$. $\square$

Corollary 5.8.24. *Let $\mathcal{C}, \mathcal{C}'$ be sets of $\mathcal{L}$-clauses, where $\mathcal{L}$ is a first-order language, such that $\mathcal{C}'$ is a renaming of $\mathcal{C}$. Then, an $\mathcal{L}$-structure $\mathcal{A}$ is a model of $\mathcal{C}$ if and only if $\mathcal{A}$ is a model of $\mathcal{C}'$.*

Proof. Let s be a renaming of $\mathcal{C}$ such that $\mathcal{C}' = \bar{s}(\mathcal{C})$ and let s_0 be an extension of s to VAR. Suppose that $\mathcal{A}$ is a model of $\mathcal{C}$. Then, by Theorem 5.8.23, $\mathcal{A} \models \bar{s}_0(\mathcal{C}) = \mathcal{C}'$.

The converse follows from the first part of this argument and from Theorem 5.8.22. $\square$

Definition 5.8.25. Let Γ be a finite set of atomic formulas. A *unifier of* Γ is an $(S_{\mathrm{FOL}}, \mathrm{VAR})$-substitution s such that $\bar{s}(\varphi) = \bar{s}(\psi)$ for all φ, $\psi \in \Gamma$.

A *most general unifier* (mgu) for a finite set of atomic formulas Γ is a unifier s' for Γ such that any unifier s'_0 for Γ can be written as $s'_0 = s_1 * s'$ for some $(S_{\mathrm{FOL}}, \mathrm{VAR})$-substitution s_1. $\square$

Example 5.8.26. Let

$$\Gamma = \{R(x_1, x_2), R(f(x_3, x_4), g(x_4, x_5)), R(f(a, x_4), x_6)\},$$

where R is a binary relation symbol, f, g are binary function symbols and a is a constant symbol of $\mathcal{L}$. Then, any substitution s with

$$\begin{array}{ll} s(x_1) = f(a, x_7) & s(x_2) = g(x_7, x_8) \\ s(x_3) = a & s(x_4) = x_7 \\ s(x_5) = x_8 & s(x_6) = g(x_7, x_8) \end{array}$$

unifies Γ to the formula $R(f(a, x_7), g(x_7, x_8))$. ▯

We are going to introduce an extension of the signature of first-order logic because in this section we need to treat atomic formulas as terms belonging to this signature.

Definition 5.8.27. The *extended signature of first-order logic* is the extension $S^{ext}_{\text{FOL}} = (F \cup R, \nu^{ext})$ of the signature $S_{\text{FOL}} = (F, \nu)$, where $F = \{f^n_k \mid n, k \in \mathbf{N}\}$, $R = \{R^n_k \mid n, k \in \mathbf{N}\}$, $\nu^{ext}(f) = \nu(f)$ for every f in F and $\nu^{ext}(R^n_k) = n$ for every $n, k \in \mathbf{N}$.

Let $\mathcal{L}$ be a first-order language. The *extended signature of* $\mathcal{L}$ is the reduct $S^{ext}_{\mathcal{L}}$ of S^{ext}_{FOL} to $\mathcal{L}$. ▯

It follows that every atomic formula is an $(S^{ext}_{\text{FOL}}, \text{VAR})$-term and every $\mathcal{L}$-atomic formula is an $(S^{ext}_{\mathcal{L}}, \text{VAR})$-term.

Theorem 5.8.28. *Let* Γ *be a finite set of atomic formulas. A substitution* $s : VAR \longrightarrow \text{TERM}$ *is a unifier of* Γ *in the sense of Definition* 5.8.25 *if and only if it is an* $(S^{ext}_{FOL}, S_{FOL}, VAR)$*-unifier in the sense of Definition* 1.6.1.

Proof. The statement follows immediately from the definitions mentioned above. □

Corollary 5.8.29. *Let* Γ *be a finite set of atomic formulas. A substitution* $s : VAR \longrightarrow \text{TERM}$ *is an mgu of* Γ *in the sense of Definition* 5.8.25 *if and only if it is an* $(S^{ext}_{FOL}, S_{FOL}, VAR)$*-mgu in the sense of Definition* 1.6.5.

Proof. This follows directly from Theorem 5.8.28. □

Definition 5.8.30. Let $\mathcal{L}$ be a first-order language and Γ be a finite set of atomic $\mathcal{L}$-formulas. An $(S_{\mathcal{L}}, \text{VAR})$-substitution s such that $\bar{s}(\varphi) = \bar{s}(\psi)$ for all $\varphi, \psi \in \Gamma$, is called an $\mathcal{L}$*-unifier of* Γ.

An $\mathcal{L}$*-most general unifier* ($\mathcal{L}$-mgu) is an $\mathcal{L}$-unifier s' such that for any $\mathcal{L}$-unifier s'_0 can be factored as $s'_0 = s_1 * s'$ for some $(S_{\mathcal{L}}, \text{VAR})$-substitution s_1. ▯

Theorem 5.8.31. *Let $\mathcal{L}$ be a first-order language and let Γ be a finite set of atomic $\mathcal{L}$-formulas. A substitution $s : VAR \longrightarrow \mathrm{TERM}_{\mathcal{L}}$ is an $\mathcal{L}$-unifier of Γ in the sense of Definition* 5.8.30 *if and only if it is an $(S^{ext}_{\mathcal{L}}, S_{\mathcal{L}}, VAR)$-unifier in the sense of Definition* 1.6.1.

Proof. The statement is immediate. □

Corollary 5.8.32. *Let $\mathcal{L}$ be a first-order language and let Γ be a finite set of atomic $\mathcal{L}$-formulas. A substitution $s : VAR \longrightarrow \mathrm{TERM}_{\mathcal{L}}$ is an $\mathcal{L}$-mgu of Γ in the sense of Definition* 5.8.30 *if and only if it is an $(S^{ext}_{\mathcal{L}}, S_{\mathcal{L}}, VAR)$-mgu in the sense of Definition* 1.6.5.

Proof. This follows directly from Theorem 5.8.31. □

By Corollary 5.8.32, if Γ is a finite set of $\mathcal{L}$-atomic formulas, the application of Algorithm 1.6.18 with $S = S^{ext}_{\mathcal{L}}$, $V = \mathrm{VAR}$, $S' = S_{\mathcal{L}}$ and $T = \Gamma$, will determine if Γ is $\mathcal{L}$-unifiable and will produce an $\mathcal{L}$-mgu if Γ is $\mathcal{L}$-unifiable.

Theorem 5.8.33. *Let Γ be a finite set of atomic formulas. The following three statements are equivalent*:

(1) Γ *is $\mathcal{L}$-unifiable for all languages $\mathcal{L}$ such that* $\Gamma \subseteq \mathrm{AFORM}_{\mathcal{L}}$;
(2) Γ *is $\mathcal{L}$-unifiable for some language $\mathcal{L}$ such that* $\Gamma \subseteq \mathrm{AFORM}_{\mathcal{L}}$;
(3) Γ *is unifiable.*

Proof. The implication (1) $\Rightarrow$ (2) is immediate because the set Γ is finite and hence there is a first-order language $\mathcal{L}$ such that $\Gamma \subseteq \mathrm{AFORM}_{\mathcal{L}}$. Also, (2) $\Rightarrow$ (3) is immediate because an $\mathcal{L}$-substitution is a substitution.

To prove the implication (3) $\Rightarrow$ (1), let $\mathcal{L}$ be a first-order language such that $\Gamma \subseteq \mathrm{AFORM}_{\mathcal{L}}$. We observed that unifiability of a set Γ implies $(S^{ext}_{\mathrm{FOL}}, S_{\mathrm{FOL}}, \mathrm{VAR})$-unifiability, which is equivalent to $(\mathtt{r}_{S^{ext}_{\mathrm{FOL}},\Gamma,min}, S_{\mathrm{FOL}} \sqcap \mathtt{r}_{S^{ext}_{\mathrm{FOL}},\Gamma,min}, \mathtt{V}(\Gamma))$-unifiabilty, by Lemma 1.6.23. In turn, this unifiability is equivalent to $(\mathtt{r}_{S^{ext}_{\mathcal{L}},\Gamma,min}, S_{\mathcal{L}} \sqcap \mathtt{r}_{S^{ext}_{\mathcal{L}},\Gamma,min}, \mathtt{V}(\Gamma))$-unifiability, since Γ is a set of $\mathcal{L}$-formulas. By Lemma 1.6.23, this is equivalent to $(S^{ext}_{\mathcal{L}}, S_{\mathcal{L}}, \mathrm{VAR})$-unifiability, i.e., $\mathcal{L}$-unifiability. □

Theorem 5.8.34. *Let $\mathcal{L}$ be a first-order language, Γ be a finite set of $\mathcal{L}$-atomic formulas and s be an (S_{FOL}, VAR)-substitution. Then, s is a most general unifier of Γ if and only if s is a most general $\mathcal{L}$-unifier of Γ.*

Proof. This result follows from Supplement 79 of Chapter 1 by taking $S = S^{ext}_{\text{FOL}}$, $S' = S_{\text{FOL}}$, $S_0 = S^{ext}_{\mathcal{L}}$, $S'_0 = S_{\mathcal{L}}$ and $V = \text{VAR}$ and applying Corollaries 5.8.32 and 5.8.29. □

Corollary 5.8.35. *Let $\mathcal{L}, \mathcal{L}'$ be two first-order languages and let Γ be a finite set of atomic formulas that is both a set of $\mathcal{L}$-atomic formulas and a set of $\mathcal{L}'$-atomic formulas. Then a substitution s is a most general $\mathcal{L}$-unifier of Γ if and only if s is a most general $\mathcal{L}'$-unifier of Γ.*

Proof. This statement follows from a double application of Theorem 5.8.34. □

Note that the above corollary shows that the set of most general unifiers of a finite set of $\mathcal{L}$-atomic formulas Γ does not depend on the first-order language $\mathcal{L}$.

Definition 5.8.36. Let $\mathcal{L}$ be a first-order language and let C_0, C_1 be two $\mathcal{L}$-clauses. A *standardization* of (C_0, C_1) is a pair (s_0, s_1) such that s_0 is a renaming of C_0, s_1 is a renaming of C_1 and $\mathtt{V}(\bar{s}_0(C_0)) \cap \mathtt{V}(\bar{s}_1(C_1)) = \emptyset$.

An $\mathcal{L}$-clause R is an *$\mathcal{L}$-resolvent* of two $\mathcal{L}$-clauses C_0, C_1, if there is a standardization (s_0, s_1) of (C_0, C_1), a nonempty set of positive literals $L \subseteq \bar{s}_0(C_0)$, a nonempty set of negative literals $K \subseteq \bar{s}_1(C_1)$ and an $\mathcal{L}$-unifier s of $L \cup \overline{K}$ such that

$$R = \bar{s}\left((\bar{s}_0(C_0) - L) \cup (\bar{s}_1(C_1) - K)\right).$$

The clauses C_0, C_1 are referred to as *premises* of R.

- If s is a most general $\mathcal{L}$-unifier of $L \cup \overline{K}$, then we say that R is a *most general $\mathcal{L}$-resolvent of C_0 and C_1* or *mgu $\mathcal{L}$-resolvent of C_0 and C_1*.
- If both s_0 and s_1 are the identity mappings on their respective domains, then we refer to R as a *simple $\mathcal{L}$-resolvent.*
- If s_0 is a renaming of C_0 and s_1 is a renaming of C_1, but (s_0, s_1) is not necessarily a standardization of (C_0, C_1), then we refer to R as a *weak $\mathcal{L}$-resolvent* of C_0 and C_1.
- If $|L| = |K| = 1$, then we refer to R as a *binary $\mathcal{L}$-resolvent.*

- If $\overline{s}(\overline{s_0}(C_0) - L) \cap \overline{s}(L) = \emptyset$ and $\overline{s}(\overline{s_1}(C_1) - K) \cap \overline{s}(K) = \emptyset$, then we say that R is a *full $\mathcal{L}$-resolvent.*

$\square$

Let $\mathcal{L}, \mathcal{L}'$ be two first-order languages and let C_0, C_1 be both $\mathcal{L}$- and $\mathcal{L}'$-clauses. Then, by Corollary 5.8.35, a clause R is a most general $\mathcal{L}$-resolvent of C_0 and C_1 if and only if it is a most general $\mathcal{L}'$-resolvent of C_0 and C_1 and hence is both an $\mathcal{L}$- and an $\mathcal{L}'$-clause. In view of this, we may refer to R as a *most general resolvent* of C_0 and C_1.

Example 5.8.37. Let $\mathcal{L} = \{P, a, b\}$ be a first-order language where P is a binary relation symbol and a, b are constant symbols. The $\mathcal{L}$-clauses C_0, C_1 are given by

$$C_0 = \{P(x, b), P(x, y), P(a, b)\} \text{ and } C_1 = \{(\neg P(w, v)), (\neg P(a, b))\},$$

where x, y, w, v are distinct variables.

The clause $R = \{P(a, b)\}$ is a simple mgu resolvent of C_0 and C_1, obtained by taking the substitutions s_0, s_1 as the identity mappings, the sets L and K to be $\{P(x, y), P(a, b)\}$, C_1, respectively, and s to be $s\begin{smallmatrix} x\,y\,w\,v \\ a\,b\,a\,b \end{smallmatrix}$. Note however that this argument does not show that R is a full resolvent of C_0 and C_1 because $\overline{s}(\overline{s_0}(C_0) - L) \cap \overline{s}(L) = \{P(a, b)\}$.

If we replace L with C_0, then we obtain $\square$ as a simple, full mgu resolvent of C_0 and C_1. $\square$

Observe that if C_0 and C_1 are ground $\mathcal{L}$-clauses and R is an $\mathcal{L}$-resolvent of C_0, C_1, then there is a positive literal ℓ such that $L = \{\ell\}$, $K = \{\bar{\ell}\}$ and $R = (C_0 - \{\ell\}) \cup (C_1 - \{\bar{\ell}\})$. In this case, we denote R by $\mathrm{res}_\ell(C_0, C_1)$. Further, R is in fact a full most general resolvent of C_0 and C_1.

Lemma 5.8.38. *Let $\mathcal{L}$ be a first-order language. If R is a weak $\mathcal{L}$-resolvent of the $\mathcal{L}$-clauses C_0, C_1, then R is an $\mathcal{L}$-resolvent of these clauses. Moreover, if R is a weak binary $\mathcal{L}$-resolvent of C_0, C_1, the R is a binary $\mathcal{L}$-resolvent of C_0, C_1.*

Proof. By definition of weak resolvent, there exist two renaming s_0, s_1 of C_0, C_1 respectively, a set of positive literals $L \subseteq \overline{s}_0(C_0)$, a set of negative literals $K \subseteq \overline{s}_1(C_1)$ and an $\mathcal{L}$-unifier s of $L \cup \overline{K}$ such that

$$R = \overline{s}(\overline{s}_0(C_0) - L) \cup \overline{s}(\overline{s}_1(C_1) - K).$$

Let $s_2 : \mathtt{V}(\overline{s}_1(C_1)) \longrightarrow \text{VAR}$ be a renaming of $\overline{s}_1(C_1)$ such that $\text{Ran}(s_2) \cap \mathtt{V}(\overline{s}_0(C_0)) = \emptyset$.

We claim that $s_{12} = (s_2 {*} s_1) \restriction \mathtt{V}(C_1)$ is a renaming of C_1. The range of s_{12} is clearly a set of variables and the injectivity is also immediate from the injectivity of s_1 and s_2. Furthermore, the pair (s_0, s_{12}) is a standardization of (C_0, C_1). Indeed, since $\text{Ran}(s_{12}) \subseteq \text{Ran}(s_2)$, we have $\text{Ran}(s_{12}) \cap \text{Ran}(s_0) = \emptyset$.

We have $L \subseteq \overline{s}_0(C_0)$ and $\overline{s}_2(K) \subseteq \overline{s}_2(\overline{s}_1(C_1)) = \overline{s_2 * s_1}(C_1) = \overline{s}_{12}(C_1)$, where $\overline{s}_2(K)$ is a set of negative literals. Next, we prove that $L \cup \overline{\overline{s}_2(K)}$ is unifiable. Define the substitution z_2 as

$$z_2(x) = \begin{cases} y & \text{if } s_2(y) = x \\ x & \text{if } x \notin \text{Ran}(s_2). \end{cases}$$

The substitution z_2 is well-defined because s_2 is injective.

We shall prove that the $\mathcal{L}$-substitution $s * z_2$ is a unifier of $L \cup \overline{\overline{s}_2(K)}$. We have $\overline{s * z_2}(L) = \overline{s}(\overline{z}_2(L)) = \overline{s}(L)$ because z_2 leaves unchanged the variables of L. On the other hand, $\overline{s * z_2}(\overline{\overline{s}_2(K)}) = \overline{s * z_2}(\overline{s}_2(\overline{K})) = \overline{s}(\overline{z}_2(\overline{s}_2(\overline{K}))) = \overline{s}(\overline{z_2 * s_2}(\overline{K}))$. Observe that $\overline{z_2 * s_2}(\overline{K}) = \overline{K}$ because for each variable $x \in \mathtt{V}(\overline{K})$, we have $z_2 * s_2(x) = x$. Thus, we have $\overline{s * z_2}(\overline{\overline{s}_2(K)}) = \overline{s}(\overline{K})$. Since s is a unifier of $L \cup \overline{K}$, $s * z_2$ is a unifier of $L \cup \overline{\overline{s}_2(K)}$. This allows us to conclude that

$$\begin{aligned} R' &= \overline{s * z_2}(\overline{s_0}(C_0) - L)) \cup \overline{s * z_2}(\overline{s}_{12}(C_1) - \overline{s}_2(K)) \\ &= \overline{s}(\overline{z_2}(\overline{s_0}(C_0) - L)) \cup \overline{s}(\overline{z}_2(\overline{s}_2(\overline{s}_1(C_1)) - \overline{s}_2(K))), \end{aligned}$$

is an $\mathcal{L}$-resolvent of C_0 and C_1, where in the last equality we used the fact that $\overline{s}_{12}(C_1) = \overline{s}_2(\overline{s}_1(C_1))$. Observe that if R is a binary resolvent (i.e., $|L| = |K| = 1$), then R' is also a binary resolvent. We continue to transform R' as follows.

$$\begin{aligned} R' &= \overline{s}(\overline{s_0}(C_0) - L) \cup \overline{s}(\overline{z}_2(\overline{s}_2(\overline{s}_1(C_1)) - \overline{s}_2(K)_{)}) \\ &\qquad \text{(because } z_2 \text{ leaves unchanged the variables of } \overline{s}_0(C_0)) \\ &= \overline{s}(\overline{s_0}(C_0) - L) \cup \overline{s}(\overline{z}'_2(\overline{s}_2(\overline{s}_1(C_1)) - \overline{s}_2(K))), \end{aligned}$$

where z_2' is the restriction of z_2 to the range of s_2. Note that z_2' is injective and hence, by Exercise 21 of Chapter 4, $\overline{z}_2'$ is injective. Because of this property, we can further write

$$\begin{aligned} R' &= \overline{s}(\overline{s_0}(C_0) - L) \cup \overline{s}(\overline{z}_2'(\overline{s}_2(\overline{s}_1(C_1))) - \overline{z}_2'(\overline{s}_2(K))) \\ &\qquad \text{(because of the injectivity of } \overline{z}_2'\text{)} \\ &= \overline{s}(\overline{s_0}(C_0) - L) \cup \overline{s}(\overline{z_2' * s_2}(\overline{s}_1(C_1)) - \overline{z_2' * s_2}(K)) \\ &= \overline{s}(\overline{s_0}(C_0) - L) \cup \overline{s}(\overline{s}_1(C_1) - K). \end{aligned}$$

The last equality is justified by the fact that $z_2' * s_2$ fixes every variable in $\overline{s}_1(C_1)$. Now we can conclude that $R = R'$. □

The following theorem shows that in forming mgu resolvents of clauses we may consider only *finite* mgus.

Theorem 5.8.39. *If R is an mgu resolvent of two clauses C_0 and C_1 then there is a standardization (s_0, s_1) of (C_0, C_1), a nonempty set of positive literals $L \subseteq \bar{s}_0(C_0)$, a nonempty set of negative literals $K \subseteq \bar{s}_1(C_1)$ and a finite most general unifier s of $L \cup \overline{K}$ such that*

$$R = \bar{s}\left((\bar{s}_0(C_0) - L) \cup (\bar{s}_1(C_1) - K)\right).$$

Proof. The definition of mgu resolvent implies the existence of all the objects mentioned in the body of the theorem, except that the mgu s need not be finite. However, choosing $V_0 = \mathtt{V}(C_0) \cup \mathtt{V}(C_1)$ in Supplement 83 of Chapter 1, we obtain a finite mgu s' of $L \cup \overline{K}$ that agrees with s on V_0, which we can use in place of s to obtain the same R. □

Note that in the case of an arbitrary unifier s of $L \cup \overline{K}$, the existence of a finite unifier s' that produces the same resolvent is easily obtained as $s'(x) = s(x)$ if $x \in \mathtt{V}(C_0) \cup \mathtt{V}(C_1)$ and $s'(x) = x$, otherwise.

Theorem 5.8.40. *Let $\mathcal{L}$ be a first-order language and let C_0, C_1 be two $\mathcal{L}$-clauses. If R is an $\mathcal{L}$-resolvent of C_0 and C_1 and $\mathcal{A}$ is an $\mathcal{L}$-structure, then $\mathcal{A} \models C_0$ and $\mathcal{A} \models C_1$ implies $\mathcal{A} \models R$.*

Proof. Let s_0, s_1, L, K, s be as in the definition of resolvent, so

$$R = \bar{s}\left((\bar{s}_0(C_0) - L) \cup (\bar{s}_1(C_1) - K)\right).$$

Since $\mathcal{A} \models C_0$ and $\mathcal{A} \models C_1$, by Corollary 5.8.24, $\mathcal{A} \models \bar{s}_0(C_0)$ and $\mathcal{A} \models \bar{s}_1(C_1)$. Then, by Theorem 5.8.23, we have $\mathcal{A} \models \bar{s}(\bar{s}_0(C_0))$ and $\mathcal{A} \models \bar{s}(\bar{s}_1(C_1))$. Observe that since s is a unifier of $L \cup \overline{K}$, there is a positive literal ℓ such that $\bar{s}(L \cup \overline{K}) = \{\ell\}$.

Let $\sigma \in \mathrm{ASSIGN}_{\mathcal{A}}$. We must show that $(\mathcal{A}, \sigma) \models R$. We consider two cases.

`Case 1:` $(\mathcal{A}, \sigma) \models \bar{s}(\bar{s}_0(C_0) - L)$. By the definition of R, we obtain $(\mathcal{A}, \sigma) \models R$.

`Case 2:` $(\mathcal{A}, \sigma) \not\models \bar{s}(\bar{s}_0(C_0) - L)$. Since $(\mathcal{A}, \sigma) \models \bar{s}(\bar{s}_0(C_0))$, it follows that $(\mathcal{A}, \sigma) \models \bar{s}(L) = \{\ell\}$. Therefore, $(\mathcal{A}, \sigma) \not\models \{\bar{\ell}\} = \bar{s}(K)$. Since $(\mathcal{A}, \sigma) \models \bar{s}(\bar{s}_1(C_1))$, it follows that $(\mathcal{A}, \sigma) \models \bar{s}(\bar{s}_1(C_1) - K)$, which implies that $(\mathcal{A}, \sigma) \models R$. □

Corollary 5.8.41. *Let $\mathcal{L}$ be a first-order language, $\mathcal{C}$ be a set of $\mathcal{L}$-clauses and let R be an $\mathcal{L}$-resolvent of two clauses in $\mathcal{C}$. Then, for every $\mathcal{L}$-structure $\mathcal{A}$, $\mathcal{A} \models \mathcal{C}$ if and only if $\mathcal{A} \models \mathcal{C} \cup \{R\}$.*

Proof. By Theorem 5.8.10, if $\mathcal{A} \models \mathcal{C} \cup \{R\}$, then $\mathcal{A} \models \mathcal{C}$. Conversely, suppose $\mathcal{A} \models \mathcal{C}$ and R is an $\mathcal{L}$-resolvent of C_0 and C_1 in $\mathcal{C}$. Then, $\mathcal{A} \models C_0$ and $\mathcal{A} \models C_1$, so by Theorem 5.8.40, $\mathcal{A} \models R$, so $\mathcal{A} \models \mathcal{C} \cup \{R\}$. □

Note that Theorem 5.8.40 holds even if the pair (s_0, s_1) consists of any two substitutions (not necessarily a standardization).

Definition 5.8.42. Let $\mathcal{C}$ be a set of $\mathcal{L}$-clauses, where $\mathcal{L}$ is a first-order language. Then we define

$$\mathrm{Res}_{\mathcal{L}}(\mathcal{C}) = \mathcal{C} \cup \{R \mid R \text{ is an } \mathcal{L}\text{-resolvent of two clauses in } \mathcal{C}\}$$

and

$$\mathrm{Res}^{mgu}_{\mathcal{L}}(\mathcal{C}) = \mathcal{C} \cup \{R \mid R \text{ is a most general resolvent of two clauses in } \mathcal{C}\}.$$

The counterparts of $\text{Res}_{\mathcal{L}}$ and $\text{Res}_{\mathcal{L}}^{mgu}$ which make use only of full resolvents will be denoted by $\text{fRes}_{\mathcal{L}}$ and $\text{fRes}_{\mathcal{L}}^{mgu}$, respectively. ▯

Observe that if $\mathcal{C}$ is an admissible set of $\mathcal{L}$-clauses and $\mathcal{L}'$-clauses, then $\text{Res}_{\mathcal{L}}^{mgu}(\mathcal{C}) = \text{Res}_{\mathcal{L}'}^{mgu}(\mathcal{C})$ and $\text{fRes}_{\mathcal{L}}^{mgu}(\mathcal{C}) = \text{fRes}_{\mathcal{L}'}^{mgu}(\mathcal{C})$. This allows us to use the simpler notations $\text{Res}^{mgu}(\mathcal{C})$ and $\text{fRes}^{mgu}(\mathcal{C})$.

First-order resolution is analytical in the sense that for every set of $\mathcal{L}$-clauses $\mathcal{C}$, we have $\text{Res}_{\mathcal{L}}(\mathcal{C}) \subseteq \bigcup\{W^*_{\mathcal{L},\text{VAR}}(C) \mid C \in \mathcal{C}\}$.

Theorem 5.8.43. *Let $\mathcal{C}$ and $\mathcal{D}$ be a sets of $\mathcal{L}$-clauses and let $\mathcal{A}$ be an $\mathcal{L}$-structure. Then, if $\mathcal{R}$ is Res, Res^{mgu}, fRes, or fRes^{mgu} we have*:

(1) $\mathcal{C} \subseteq \mathcal{R}_{\mathcal{L}}(\mathcal{C})$;
(2) *if $\mathcal{C} \subseteq \mathcal{D}$, then $\mathcal{R}_{\mathcal{L}}(\mathcal{C}) \subseteq \mathcal{R}_{\mathcal{L}}(\mathcal{D})$; and*
(3) *$\mathcal{A} \models \mathcal{R}_{\mathcal{L}}(\mathcal{C})$ if and only if $\mathcal{A} \models \mathcal{C}$.*

Proof. The first two parts of the theorem are immediate consequences of the Definition 5.8.42.

By Part 1 of Theorem 5.8.10 and Part 1 of the current theorem, if $\mathcal{A} \models \mathcal{R}_{\mathcal{L}}(\mathcal{C})$, then $\mathcal{A} \models \mathcal{C}$. Conversely, if $\mathcal{A} \models \mathcal{C}$, then, by Theorem 5.8.40, $\mathcal{A} \models R$ for every $\mathcal{L}$-resolvent R of two clauses in $\mathcal{C}$, and therefore, $\mathcal{A} \models \mathcal{R}_{\mathcal{L}}(\mathcal{C})$. □

Let $\mathcal{L}$ be a first-order language, $\mathcal{R}$ be Res, Res^{mgu}, fRes, or fRes^{mgu}, and let $\mathcal{S}_{\mathcal{L}}$ be the set of all $\mathcal{L}$-clauses. Then, $\mathcal{R}_{\mathcal{L}} : \mathcal{P}(\mathcal{S}_{\mathcal{L}}) \longrightarrow \mathcal{P}(\mathcal{S}_{\mathcal{L}})$, so we can consider its iterations $\mathcal{R}^n_{\mathcal{L}}$ for $n \in \mathbf{N}$, following the standard definition:

$$\mathcal{R}^0_{\mathcal{L}}(\mathcal{C}) = \mathcal{C},$$
$$\mathcal{R}^{n+1}_{\mathcal{L}}(\mathcal{C}) = \mathcal{R}_{\mathcal{L}}(\mathcal{R}^n_{\mathcal{L}}(\mathcal{C}))$$

for every set of clauses $\mathcal{C} \in \mathcal{P}(\mathcal{S}_{\mathcal{L}})$.

Note that, as a consequence of Part (1) of Theorem 5.8.43, we have the increasing chain of sets

$$\mathcal{C} = \mathcal{R}^0_{\mathcal{L}}(\mathcal{C}) \subseteq \mathcal{R}^1_{\mathcal{L}}(\mathcal{C}) \subseteq \cdots \subseteq \mathcal{R}^n_{\mathcal{L}}(\mathcal{C}) \subseteq \cdots . \tag{5.5}$$

Definition 5.8.44. Let $\mathcal{C}$ be a set of $\mathcal{L}$-clauses and let $\mathcal{R}$ be Res, Res^{mgu}, fRes, or fRes^{mgu}. Define $\mathcal{R}^*_{\mathcal{L}}(\mathcal{C})$ as by

$$\mathcal{R}^*_{\mathcal{L}}(\mathcal{C}) = \bigcup_{n\geq 0} \mathcal{R}^n_{\mathcal{L}}(\mathcal{C}).$$

▯

As before, if $\mathcal{C}$ is a set of $\mathcal{L}$-clauses and $\mathcal{L}'$-clauses and $\mathcal{R}$ is either Res^{mgu} or fRes^{mgu}, then $\mathcal{R}^*_{\mathcal{L}}(\mathcal{C}) = \mathcal{R}^*_{\mathcal{L}'}(\mathcal{C})$. This allows us to use the simpler notations $(\mathrm{Res}^{mgu})^*(\mathcal{C})$ and $(\mathrm{fRes}^{mgu})^*(\mathcal{C})$.

Theorem 5.8.45. *Let $\mathcal{C}$ and $\mathcal{D}$ be sets of $\mathcal{L}$-clauses and let $\mathcal{A}$ be an $\mathcal{L}$-structure. If $\mathcal{R}$ is Res, Res^{mgu}, fRes, or $fRes^{mgu}$, then*

(1) *$\mathcal{C} \subseteq \mathcal{R}^n_{\mathcal{L}}(\mathcal{C})$ for all $n \in \mathbf{N}$ and $\mathcal{C} \subseteq \mathcal{R}^*_{\mathcal{L}}(\mathcal{C})$;*
(2) *for all $n \in \mathbf{N}$, $\mathcal{A} \models \mathcal{R}^n_{\mathcal{L}}(\mathcal{C})$ if and only if $\mathcal{A} \models \mathcal{C}$;*
(3) *$\mathcal{A} \models \mathcal{R}^*_{\mathcal{L}}(\mathcal{C})$ if and only if $\mathcal{A} \models \mathcal{C}$;*
(4) *$\mathcal{C}$ has a model if and only if $\mathcal{R}^*_{\mathcal{L}}(\mathcal{C})$ has a model; and*
(5) *if $\mathcal{C} \subseteq \mathcal{D}$, then $\mathcal{R}^n_{\mathcal{L}}(\mathcal{C}) \subseteq \mathcal{R}^n_{\mathcal{L}}(\mathcal{D})$ for all $n \in \mathbf{N}$ and $\mathcal{R}^*_{\mathcal{L}}(\mathcal{C}) \subseteq \mathcal{R}^*_{\mathcal{L}}(\mathcal{D})$.*

Proof. The first part of the theorem follows from Equation (5.5).

For the second part of the theorem, the argument is by induction on n. The basis step, $n = 0$, is trivial. Suppose that for every $\mathcal{L}$-structure $\mathcal{A}$, $\mathcal{A} \models \mathcal{C}$ if and only if $\mathcal{A} \models \mathcal{R}^n_{\mathcal{L}}(\mathcal{C})$. Using the last part of Theorem 5.8.43, applied to $\mathcal{R}^n_{\mathcal{L}}(\mathcal{C})$, we obtain the desired conclusion.

The third part is an easy consequence of the second and the fourth part follows immediately from the third. The fifth part is a simple proof by induction, which uses Theorem 5.8.43, Part (2). □

Definition 5.8.46. Let $\mathcal{C}$ be a set of $\mathcal{L}$-clauses. An *$\mathcal{L}$-resolution proof* over $\mathcal{C}$ is a finite sequence $(C_0, C_1, \ldots, C_{n-1})$ of $\mathcal{L}$-clauses such that $n \geq 1$ and for each i, $0 \leq i \leq n-1$ either $C_i \in \mathcal{C}$ or else $C_i \notin \mathcal{C}$ and there are $j, k < i$ such C_i is an $\mathcal{L}$-resolvent of C_j and C_k. In the first case, C_i is an *input step* of the proof; in the second case, C_i is a *resolution step.*

An *$\mathcal{L}$-resolution proof of a clause C* over $\mathcal{C}$ is an $\mathcal{L}$-resolution proof over $\mathcal{C}$ whose last entry is C.

If at each resolution step, C_i is a full resolvent, or a most general resolvent or a full most general resolvent, then the sequence $(C_0, C_1, \ldots, C_{n-1})$ is a *full $\mathcal{L}$-resolution proof*, or a *most general resolution proof* or still a *full most general resolution proof*, respectively. ▯

Theorem 5.8.47. *Let $\mathcal{C}$ be a set of $\mathcal{L}$-clauses. Then,*

- $\mathrm{Res}^*_{\mathcal{L}}(\mathcal{C})$ *is the set of clauses which have $\mathcal{L}$-resolution proofs over $\mathcal{C}$;*

- $f\mathrm{Res}^*_{\mathcal{L}}(\mathcal{C})$ *is the set of clauses which have full $\mathcal{L}$-resolution proofs over $\mathcal{C}$;*
- $(\mathrm{Res}^{mgu})^*(\mathcal{C})$ *is the set of clauses which have most general resolution proofs over $\mathcal{C}$;*
- $(f\mathrm{Res}^{mgu})^*(\mathcal{C})$ *is the set of clauses which have most general full resolution proofs over $\mathcal{C}$.*

Proof. The argument is similar to the proof of the corresponding propositional logic counterpart, Theorem 3.8.19. $\square$

Next, we present two lemmas which help us prove (in Theorem 5.8.50) that we can transform an $\mathcal{L}$-resolution proof into an $\mathcal{L}'$-resolution proof, where $\mathcal{L}'$ is obtained from $\mathcal{L}$ by replacing a relation symbol P with a symbol with the same arity.

Lemma 5.8.48. *Let $\mathcal{L}$ be a first-order language and let P and R be two relation symbols of equal arity such that $P \in \mathcal{L}$ and $R \notin \mathcal{L}$. Define the first-order language $\mathcal{L}' = (\mathcal{L} - \{P\}) \cup \{R\}$. Let Γ be a set of $\mathcal{L}$-atomic formulas. If s is an $(S_{\mathcal{L}}, VAR)$-substitution, then s is an $\mathcal{L}$-unifier of Γ if and only if s is an $\mathcal{L}'$-unifier of $\mathsf{s}^P_R(\Gamma)$. Furthermore, s is a most general unifier of Γ if and only if s is a most general unifier of $\mathsf{s}^P_R(\Gamma)$.*

Proof. We begin by observing that $S_{\mathcal{L}} = S_{\mathcal{L}'}$ because the sets of function symbols of $\mathcal{L}$ and $\mathcal{L}'$ coincide. Also note that since $R \notin \mathcal{L}$,

$$\mathsf{s}^R_P(\mathsf{s}^P_R(\Gamma)) = \Gamma. \tag{5.6}$$

Suppose that s is an $\mathcal{L}$-unifier of Γ, that is, $|s(\Gamma)| \leq 1$. Then, by Theorem 1.2.23, we have $|s(\mathsf{s}^P_R(\Gamma))| = |\mathsf{s}^P_R(s(\Gamma))| \leq 1$, so s is an $\mathcal{L}'$-unifier of $\mathsf{s}^P_R(\Gamma)$.

The reverse implication can be obtained by replacing Γ with $\mathsf{s}^P_R(\Gamma)$ and exchanging the roles of P and R and of $\mathcal{L}$ and $\mathcal{L}'$, taking into account Equality (5.6). This concludes the proof of the first part of the lemma.

To prove the second part, suppose that s is a most general $\mathcal{L}$-unifier of Γ. By the first part of the lemma, s is an $\mathcal{L}'$-unifier of $\mathsf{s}^P_R(\Gamma)$. Let z be an $\mathcal{L}'$-unifier of $\mathsf{s}^P_R(\Gamma)$. Again, by the first part of the lemma, z is an $\mathcal{L}$-unifier of Γ and therefore, we have the factorization $z = z_1 * s$, where z_1 is an $\mathcal{L}$-substitution and hence an $\mathcal{L}'$-substitution.

This shows that s is a most general $\mathcal{L}'$-unifier of $\mathsf{s}_R^P(\Gamma)$. The converse implication can be shown by using the same argument as for the first part of the theorem. $\square$

Lemma 5.8.49. *Let $\mathcal{L}$ be a first-order language and let P and R be two relation symbols of equal arity such that $P \in \mathcal{L}$ and $R \notin \mathcal{L}$. Define the first-order language $\mathcal{L}' = (\mathcal{L} - \{P\}) \cup \{R\}$. Let C_0, C_1 be two $\mathcal{L}$-clauses and C be an $\mathcal{L}$-resolvent (most general resolvent, full $\mathcal{L}$-resolvent, full most general resolvent) of C_0 and C_1. Then, $\mathsf{s}_R^P(C)$ is the same type of $\mathcal{L}'$-resolvent of $\mathsf{s}_R^P(C_0)$ and $\mathsf{s}_R^P(C_1)$.*

Proof. Since C is an $\mathcal{L}$-resolvent of C_0, C_1, there is a standardization (s_0, s_1) of (C_0, C_1), a nonempty set of positive literals $L \subseteq \bar{s}_0(C_0)$, a nonempty set of negative literals $K \subseteq \bar{s}_1(C_1)$ and an $\mathcal{L}$-unifier s of $L \cup \overline{K}$ such that

$$C = \bar{s}\left((\bar{s}_0(C_0) - L) \cup (\bar{s}_1(C_1) - K)\right).$$

Since $\mathsf{s}_R^P(C_0), \mathsf{s}_R^P(C_1)$ have the same variables as C_0, C_1, respectively, (s_0, s_1) is a standardization of $(\mathsf{s}_R^P(C_0), \mathsf{s}_R^P(C_1))$. Also, $\mathsf{s}_R^P(L)$ is a subset of positive literals of $\mathsf{s}_R^P(C_0)$ and $\mathsf{s}_R^P(K)$ is a subset of negative literals of $\mathsf{s}_R^P(C_1)$.

By Lemma 5.8.48, s is an $\mathcal{L}'$-unifier of $\mathsf{s}_R^P(L) \cup \overline{\mathsf{s}_R^P(K)} = \mathsf{s}_R^P(L \cup \overline{K})$. By definition of resolvent, the clause

$$C' = \bar{s}((\bar{s}_0(\mathsf{s}_R^P(C_0)) - \mathsf{s}_R^P(L)) \cup (\bar{s}_1(\mathsf{s}_R^P(C_1)) - \mathsf{s}_R^P(K)))$$

is a resolvent of $\mathsf{s}_R^P(C_0)$ and $\mathsf{s}_R^P(C_1)$. We show now that $C' = \mathsf{s}_R^P(C)$. Note firstly that s_R^P restricted to $\mathcal{L}$-literals is an injection ranging over $\mathcal{L}'$-literals because R is not in $\mathcal{L}$. Now, we have

$$\begin{aligned}
\mathsf{s}_R^P(C) &= \mathsf{s}_R^P(\bar{s}\left((\bar{s}_0(C_0) - L) \cup (\bar{s}_1(C_1) - K)\right)) \\
&= \bar{s}(\mathsf{s}_R^P\left((\bar{s}_0(C_0) - L) \cup (\bar{s}_1(C_1) - K)\right)) \\
&\quad \text{(by Theorem 1.2.23)} \\
&= \bar{s}(\mathsf{s}_R^P(\bar{s}_0(C_0) - L) \cup \mathsf{s}_R^P(\bar{s}_1(C_1) - K)) \\
&= \bar{s}((\mathsf{s}_R^P(\bar{s}_0(C_0)) - \mathsf{s}_R^P(L)) \cup (\mathsf{s}_R^P(\bar{s}_1(C_1)) - \mathsf{s}_R^P(K))) \\
&\quad \text{(because of the injectivity of } \mathsf{s}_R^P\text{)} \\
&= C'.
\end{aligned}$$

This completes the argument for the first claim of the lemma.

Combining Lemma 5.8.48 with the argument used for the first part of the current lemma, gives us the second part of the lemma. We leave to the reader the proof of the last two parts. □

Theorem 5.8.50. *Let $\mathcal{C}$ be a set of $\mathcal{L}$-clauses and let P and R be two relation symbols of equal arity such that $P \in \mathcal{L}$ and $R \notin \mathcal{L}$. Define the first-order language $\mathcal{L}' = (\mathcal{L} - \{P\}) \cup \{R\}$. If $(C_0, \ldots, C_{n-1})$ is an $\mathcal{L}$-resolution (most general $\mathcal{L}$-resolution, full $\mathcal{L}$-resolution, full most general $\mathcal{L}$-resolution) proof of C over $\mathcal{C}$, then $(\mathsf{s}_R^P(C_0), \ldots, \mathsf{s}_R^P(C_{n-1}))$ is the same type of proof of $\mathsf{s}_R^P(C)$ over $\mathsf{s}_R^P(\mathcal{C})$ with $\mathcal{L}$ replaced by $\mathcal{L}'$.*

Proof. Let i be a number such that $0 \leq i \leq n-1$. If $C_i \in \mathcal{C}$, then $\mathsf{s}_R^P(C_i) \in \mathsf{s}_R^P(\mathcal{C})$. Otherwise, there are $j, k < i$ such that C_i is an $\mathcal{L}$-resolvent of C_j and C_k. By Lemma 5.8.49, $\mathsf{s}_R^P(C_i)$ is an $\mathcal{L}'$-resolvent of $\mathsf{s}_R^P(C_j)$ and $\mathsf{s}_R^P(C_k)$. Thus, the sequence $(\mathsf{s}_R^P(C_0), \ldots, \mathsf{s}_R^P(C_{n-1}))$ is an $\mathcal{L}'$-resolution proof.

By the second part of Lemma 5.8.49, if $(C_0, \ldots, C_{n-1})$ is a most general $\mathcal{L}$-resolution proof, then $(\mathsf{s}_R^P(C_0), \ldots, \mathsf{s}_R^P(C_{n-1}))$ is a most general $\mathcal{L}'$-resolution proof.

We leave the arguments for the remaining parts to the reader. □

Corollary 5.8.51. *Let $\mathcal{C}$ be a set of $\mathcal{L}$-clauses, C be an $\mathcal{L}$-clause, and let P and R be two relation symbols of equal arity such that $P \in \mathcal{L}$ and $R \notin \mathcal{L}$. Define the first-order language $\mathcal{L}' = (\mathcal{L} - \{P\}) \cup \{R\}$. If $\mathcal{R}$ is either* Res, *Res^{mgu}, fRes or $fRes^{mgu}$, we have $C \in \mathcal{R}_{\mathcal{L}}^*(\mathcal{C})$ if and only if $\mathsf{s}_R^P(C) \in \mathcal{R}_{\mathcal{L}'}^*(\mathsf{s}_R^P(\mathcal{C}))$.*

Proof. This is a direct consequence of Theorem 5.8.50. □

The notion of $\mathcal{L}$-resolution proof can be viewed in the framework of the formal system introduced next.

Definition 5.8.52. Let $\mathcal{L}$ be a first-order language. The formal system $\mathcal{FRES}^{\mathcal{L}}$ is

$$\mathcal{FRES}^{\mathcal{L}} = (\mathcal{P}_{fin}(\mathrm{LIT}_{\mathcal{L}}), \emptyset, \{\mathsf{R}\}),$$

where the set of objects $\mathcal{P}_{fin}(\mathrm{LIT}_{\mathcal{L}})$ consists of finite sets of literals of $\mathcal{L}$ ($\mathcal{L}$-clauses) and the rule R consists of all pairs $((C, D), E)$ where E is an $\mathcal{L}$-resolvent of C and D.

If rule R is replaced by rule R^f consisting of all pairs $((C,D),E)$ where E is a full $\mathcal{L}$-resolvent of C and D, we obtain the formal system $\mathcal{FFRES}^{\mathcal{L}}$.

Let R_{mgu} be the rule that consists of the pairs $((C,D),E)$ such that E is a most general resolvent of C and D. If the rule R is replaced in $\mathcal{FRES}^{\mathcal{L}}$ by the rule R_{mgu}, we obtain the formal system $\mathcal{FRES}^{\mathcal{L}}_{mgu}$.

The formal system $\mathcal{FFRES}^{\mathcal{L}}_{mgu}$ is obtained by replacing the rule R_{mgu} with R^f_{mgu} using full most general resolvents. ▯

Note that if $\mathcal{C}$ is a set of $\mathcal{L}$-clauses, then an $\mathcal{L}$-resolution proof over $\mathcal{C}$ is the same thing as a proof in the formal system $\mathcal{FRES}^{\mathcal{L}}_{\mathcal{C}}$, so Theorem 5.8.47 implies that $\mathrm{Thm}(\mathcal{FRES}^{\mathcal{L}}_{\mathcal{C}}) = \mathrm{Res}^*_{\mathcal{L}}(\mathcal{C})$. Similarly, $\mathrm{Thm}((\mathcal{FRES}^{\mathcal{L}}_{mgu})_{\mathcal{C}}) = (\mathrm{Res}^{mgu}_{\mathcal{L}})^*(\mathcal{C})$. The introduction of a formal system allows us to make use of the idea of proof tree.

Definition 5.8.53. Let $\mathcal{C}$ be a set of $\mathcal{L}$-clauses. An *$\mathcal{L}$-resolution tree over C* is an $\mathcal{FRES}^{\mathcal{L}}_{\mathcal{C}}$-proof tree. ▯

In other words, an $\mathcal{L}$-resolution tree over $\mathcal{C}$ is a lot such that its leaves are labeled with clauses from $\mathcal{C}$ and each interior node is labeled with a clause that is an $\mathcal{L}$-resolvent of the clauses which are labels of its two immediate descendents. Theorem 1.8.23 and our previous discussion allow us to conclude that a clause C is in $\mathrm{Res}^*_{\mathcal{L}}(\mathcal{C})$ if and only if there is an $\mathcal{L}$-resolution tree over $\mathcal{C}$ such that C is the label of its root.

Theorem 5.8.54 (Soundness of First-Order Resolution). *Let $\mathcal{C}$ be a set of $\mathcal{L}$-clauses, where $\mathcal{L}$ is a first-order language. If $\square \in \mathrm{Res}^*_{\mathcal{L}}(\mathcal{C})$, then $\mathcal{C}$ has no model.*

Proof. By the third part of Theorem 5.8.45, any model of $\mathcal{C}$ would be a model of $\mathrm{Res}^*_{\mathcal{L}}(\mathcal{C})$. Since $\square \in \mathrm{Res}^*_{\mathcal{L}}(\mathcal{C})$, it follows that there is no model for $\mathrm{Res}^*_{\mathcal{L}}(\mathcal{C})$, and, therefore, there is no model for $\mathcal{C}$. □

Note that the soundness theorem holds even if we use arbitrary substitutions rather than standardizations in computing "resolvents."

In terms of the formal system $\mathcal{FRES}^{\mathcal{L}}_{\mathcal{C}}$, Theorem 5.8.54 amounts to saying that if $\square$ is a theorem of $\mathcal{FRES}^{\mathcal{L}}_{\mathcal{C}}$, then $\mathcal{C}$ has no model.

Corollary 5.8.55 (Soundness of MGU-Resolution). *Let $\mathcal{C}$ be a set of $\mathcal{L}$-clauses, where $\mathcal{L}$ is a first-order language. If $\Box$ belongs to $(\mathrm{Res}^{mgu})^*(\mathcal{C})$, then $\mathcal{C}$ has no model.*

Proof. The corollary follows from Theorem 5.8.54. $\Box$

Example 5.8.56. In Example 5.8.18, we showed that the logical validity of the formula $\varphi = (\exists x)(P(x) \to (\forall x)P(x))$ is equivalent to the nonexistence of a model for the set of clauses $\mathcal{C} = \{\{P(x)\}, \{(\neg P(f(x)))\}\}$. We now show that $\Box \in \mathrm{Res}^*_{\mathcal{L}}(\mathcal{C})$, which, by the Soundness Theorem, will establish the nonexistence of a model for $\mathcal{C}$. Let $C_0 = \{P(x)\}$ and $C_1 = \{(\neg P(f(x)))\}$ and let $s_0(x) = y$ and $s_1(x) = x$. Thus, $\bar{s}_0(C_0) = \{P(y)\}$ and $\bar{s}_1(C_1) = \{(\neg P(f(x)))\}$. Let $L = \bar{s}_0(C_0)$ and $K = \bar{s}_1(C_1)$, where we are using the notations of Definition 5.8.36. The unifier s of $L \cup \overline{K}$ is defined by $s(x) = x$, $s(y) = f(x)$ and $s(z) = z$ for $z \notin \{x, y\}$. The resolvent R is $\Box$, which shows that $\Box \in \mathrm{Res}^*_{\mathcal{L}}(\mathcal{C})$.

Observe that the renamings s_0 and s_1 make possible the resolution process, since otherwise, $P(x)$ and $P(f(x))$ would not be directly unifiable.

Our previous examples (Examples 5.8.16, 5.8.19 and 5.8.20) also involve sets of clauses that can be resolved to $\Box$ in one step and therefore have no model. $\Box$

Lemma 5.8.57. *Let s be an $\mathcal{L}$-inter-substitution, where $\mathcal{L}$ is a first-order language, $\mathcal{A}$ be an $\mathcal{L}$-structure and σ be an assignment in $\mathrm{ASSIGN}_{\mathcal{A}}$. Define a truth assignment v by:*

$$v(p) = \begin{cases} \mathbf{T} & \textit{if } (\mathcal{A}, \sigma) \models s(p) \\ \mathbf{F} & \textit{otherwise.} \end{cases}$$

Then, for every clause C of propositional logic such that $s(C)$ is a first-order logic clause, v satisfies C if and only if $(\mathcal{A}, \sigma) \models s(C)$.

Proof. The result follows immediately from Lemma 4.8.11. $\Box$

Theorem 5.8.58. *Let $\mathcal{C}$ be a set of $\mathcal{L}$-clauses, where $\mathcal{L}$ is a first-order language and let $\mathcal{C}_0$ be a propositional form for $\mathcal{C}$. If $\mathcal{C}$ is satisfiable, then $\mathcal{C}_0$ is satisfiable.*

Proof. Suppose that $\mathcal{C} = s(\mathcal{C}_0)$, where s is an inter-substitution. We may assume that s is an $\mathcal{L}$-substitution by redefining s on the variables that do not occur in the clauses of $\mathcal{C}_0$, if necessary. Since $\mathcal{C}$ is satisfiable, there is an $\mathcal{L}$-structure $\mathcal{A}$ and $\sigma \in \mathrm{ASSIGN}_{\mathcal{A}}$ such that $(\mathcal{A}, \sigma) \models C$ for every $C \in \mathcal{C}$. Define v as in Lemma 5.8.57. Then, by the same Lemma, v satisfies $\mathcal{C}_0$. $\square$

Theorem 5.8.59. *Let $\mathcal{L}$ be a first-order language without equality, $\mathcal{C}$ be a set of $\mathcal{L}$-clauses, and $\mathcal{C}_0$ be a fundamental propositional form for $\mathcal{C}$. Then, $\mathcal{C}$ is satisfiable if and only if $\mathcal{C}_0$ is satisfiable.*

Proof. One half of the statement was already shown in Theorem 5.8.58. Suppose now that $\mathcal{C}_0$ is satisfiable and let v be a truth assignment that satisfies $\mathcal{C}_0$. Since $\mathcal{C}_0$ is a fundamental propositional form for $\mathcal{C}$, there is an injective, prime inter-substitution s such that $\mathcal{C} = s(\mathcal{C}_0)$. Note that if $p \in SV(\mathcal{C}_0)$, then $s(p)$ is an atomic $\mathcal{L}$-formula because $s(p)$ occurs in a clause of $\mathcal{C}$. Consider the set of atomic formulas $S = \{s(p) \mid p \in SV(\mathcal{C}_0) \text{ and } v(p) = \mathbf{T}\}$, the structure $\mathcal{A} = \mathsf{STR}_{\mathcal{L},\mathrm{VAR}}(S)$, and the assignment σ defined by $\sigma(x) = x$ for every $x \in \mathrm{VAR}$.

We now show that $(\mathcal{C}, \sigma) \models \mathcal{C}$. To this end, let $C \in \mathcal{C}$ and let $D \in \mathcal{C}_0$ be such that $s(D) = C$. Since v satisfies $\mathcal{C}_0$, it follows that v satisfies D, say $v(\ell) = \mathbf{T}$ for some literal $\ell \in D$. If $\ell = p$, we have $v(p) = \mathbf{T}$, so $s(p) \in S$. By Lemma 4.10.11, $(\mathcal{A}, \sigma) \models s(p) \in s(D) = C$ and therefore $(\mathcal{A}, \sigma) \models C$. If $\ell = (\neg p)$, we have $v(p) = \mathbf{F}$, so by the injectivity of s, $s(p) \notin S$, hence, by the same lemma, $(\mathcal{A}, \sigma) \not\models s(p)$, so $(\mathcal{A}, \sigma) \models s((\neg p))$, which again implies that $(\mathcal{A}, \sigma) \models C$. Since C was an arbitrary clause of $\mathcal{C}$, $(\mathcal{A}, \sigma) \models \mathcal{C}$. $\square$

The following preliminary result is needed in the completeness proof for ground clauses.

Lemma 5.8.60. *Let $\mathcal{L}$ be a first-order language, C_0, C_1 be two propositional logic clauses and $R = res_\ell(C_0, C_1)$ be a resolvent of C_0 and C_1. If s is an injective, atomic $\mathcal{L}$-inter-substitution, then $R' = s(R)$ is an $\mathcal{L}$-resolvent of the first-order clauses $C_0' = s(C_0)$ and $C_1' = s(C_1)$.*

Proof. By the definition of propositional resolvent, we can assume that ℓ is a positive literal such that $\ell \in C_0$, $\bar{\ell} \in C_1$, and

$R = (C_0 - \{\ell\}) \cup (C_1 - \{\bar{\ell}\})$. We have

$$\begin{aligned} R' = s(R) &= s(C_0 - \{\ell\}) \cup s(C_1 - \{\bar{\ell}\}) \\ &= (s(C_0) - s(\{\ell\})) \cup (s(C_1) - s(\{\bar{\ell}\})) \\ &\quad \text{(because } s \text{ is injective and atomic)} \\ &= (C_0' - \{s(\ell)\})) \cup (C_1' - \{\overline{s(\ell)}\}), \end{aligned}$$

taking into account Theorem 4.8.5. The clause R' is a weak $\mathcal{L}$-resolvent of the clauses C_0' and C_1' because, using the notation of Definition 5.8.36, we can take $L = \{s(\ell)\}$, $K = \{\overline{s(\ell)}\}$ and s_0, s_1 and s to be the identity mappings on their respective domains. Lemma 5.8.38 implies that R' is an $\mathcal{L}$-resolvent of C_0' and C_1'. □

Lemma 5.8.61 (First Lifting Lemma). *Let $\mathcal{L}$ be a first-order language and let s be an injective, atomic $\mathcal{L}$-inter-substitution.*

If $(C_0, \ldots, C_{n-1})$ is a propositional resolution proof over a set of propositional clauses $\mathcal{C}$, then $(s(C_0), \ldots, s(C_{n-1}))$ is an $\mathcal{L}$-resolution proof over $s(\mathcal{C})$. Further, i is an input step of the proof $(C_0, \ldots, C_{n-1})$ if and only if it is an input step of the proof $(s(C_0), \ldots, s(C_{n-1}))$, for $0 \leq i \leq n-1$.

Proof. Let $0 \leq i \leq n-1$. If $C_i \in \mathcal{C}$, then $s(C_i) \in s(\mathcal{C})$. Otherwise, $C_i \notin \mathcal{C}$, so by Theorem 4.8.5, $s(C_i) \notin s(\mathcal{C})$, and there are $j, k < i$ such that $C_i = \text{res}_\ell(C_j, C_k)$ which implies that $s(C_i)$ is an $\mathcal{L}$-resolvent of $s(C_j)$ and $s(C_k)$ by Lemma 5.8.60. □

The next statement allows "lifting" a resolution proof involving fundamental propositional forms of clauses to a resolution proof involving these clauses.

Theorem 5.8.62. *Let $\mathcal{L}$ be a first-order language, $\mathcal{C}$ be a set of $\mathcal{L}$-clauses, $\mathcal{C}_0$ be a fundamental propositional form for $\mathcal{C}$ and let s be a prime, injective inter-substitution with $s(\mathcal{C}_0) = \mathcal{C}$. Then, for all propositional clauses $C \in \text{Res}^*(\mathcal{C}_0)$, we have $s(C) \in \text{Res}^*_{\mathcal{L}}(\mathcal{C})$.*

Proof. Note that we can assume that s is an atomic $\mathcal{L}$-inter-substitution by redefining it on the variables that do not occur in $\mathcal{C}_0$. Let $(C_0, \ldots, C_{n-1})$ be a resolution proof of C over $\mathcal{C}_0$. By Lemma 5.8.61, $(s(C_0), \ldots, s(C_{n-1}))$ is an $\mathcal{L}$-resolution proof of

$s(C_{n-1}) = s(C)$ over $s(\mathcal{C}_0) = \mathcal{C}$. The current statement follows immediately from Theorem 5.8.47 and the corresponding theorem from propositional logic. $\square$

We prove now a restricted form of completeness of resolution which is limited to ground clauses and equality-free first-order languages.

Theorem 5.8.63 (Completeness of Resolution for Ground Clauses and Languages without Equality). *Let $\mathcal{L}$ be a first-order language without equality and let $\mathcal{C}$ be a set of ground $\mathcal{L}$-clauses. If $\mathcal{C}$ has no model, then $\square \in \mathrm{Res}^*_{\mathcal{L}}(\mathcal{C})$.*

Proof. By Theorem 5.8.7, there is a fundamental propositional form $\mathcal{C}_0$ for $\mathcal{C}$. Since $\mathcal{C}$ consists of ground clauses and has no model, it follows from Corollary 5.8.14 that $\mathcal{C}$ is unsatisfiable. This in turn implies that $\mathcal{C}_0$ is unsatisfiable, by Theorem 5.8.59. By the completeness of propositional resolution (Theorem 3.8.32), we have $\square \in \mathrm{Res}^*(\mathcal{C}_0)$. By Theorem 5.8.62, $\square \in \mathrm{Res}^*_{\mathcal{L}}(\mathcal{C})$. $\square$

To reduce model existence for arbitrary sets of $\mathcal{L}$-clauses to model existence for sets of ground $\mathcal{L}$-clauses, we need to introduce the notion of ground instance of a set of clauses as a special case of the notion of instance of a set of clauses.

Definition 5.8.64. If V is a set of variables and C is a clause with $\mathtt{V}(C) = \{y_0, \ldots, y_{n-1}\}$ with $y_0, \ldots, y_{n-1}$ in the standard order of variables, then an *$(\mathcal{L}, V)$-instance of C* is a clause of the form

$$\{(\ell)_{y_0,\ldots,y_{n-1}:=t_0,\ldots,t_{n-1}} \mid \ell \in C\},$$

where $t_0, \ldots, t_{n-1}$ are $(\mathcal{L}, V)$-terms. If $V = \emptyset$, then we refer to an $(\mathcal{L}, V)$-instance of C as a *ground instance of C*.

The set of $(\mathcal{L}, V)$-instances of a clause is denoted by $\mathrm{INST}_{\mathcal{L},V}(C)$, while the set of ground instances of C is denoted by $\mathrm{GINST}_{\mathcal{L}}(C)$. These notations are extended to sets of clauses $\mathcal{C}$ by

$$\mathrm{INST}_{\mathcal{L},V}(\mathcal{C}) = \bigcup_{C \in \mathcal{C}} \mathrm{INST}_{\mathcal{L},V}(C),$$

$$\mathrm{GINST}_{\mathcal{L}}(\mathcal{C}) = \bigcup_{C \in \mathcal{C}} \mathrm{GINST}_{\mathcal{L}}(C).$$

$\square$

Let $\mathcal{C}$ be a set of $\mathcal{L}$-clauses that does not contain $\Box$. Recall that $\Gamma_{\mathcal{C}}$ is a set of formulas each of which is a disjunction of literals. Starting from this set of formulas, we construct the set of ground instances of the universal closure $\Gamma_{\mathcal{C}}^{\forall}$. The set $\mathrm{GINST}_{\mathcal{L}}(\Gamma_{\mathcal{C}}^{\forall})$ consists of disjunctions of ground literals. The next statement relates the set of clauses which correspond to $\mathrm{GINST}_{\mathcal{L}}(\Gamma_{\mathcal{C}}^{\forall})$ to the set of ground instances of $\mathcal{C}$.

Theorem 5.8.65. *Let $\mathcal{L}$ be a first-order language. If $\mathcal{C}$ is a set of $\mathcal{L}$-clauses that does not contain $\Box$, we have*

$$\mathcal{C}_{\mathrm{GINST}_{\mathcal{L}}(\Gamma_{\mathcal{C}}^{\forall})} = \mathrm{GINST}_{\mathcal{L}}(\mathcal{C}).$$

If $\Box \in \mathcal{C}$, then we have

$$\mathcal{C}_{\mathrm{GINST}_{\mathcal{L}}(\Gamma_{\mathcal{C}-\{\Box\}}^{\forall})} \cup \{\Box\} = \mathrm{GINST}_{\mathcal{L}}(\mathcal{C})$$

Proof. Suppose initially that $\Box \notin \mathcal{C}$. Let $C \in \mathcal{C}_{\mathrm{GINST}_{\mathcal{L}}(\Gamma_{\mathcal{C}}^{\forall})}$. There is a formula $\varphi \in \mathrm{GINST}_{\mathcal{L}}(\Gamma_{\mathcal{C}}^{\forall})$ such that $C \in \mathcal{C}_{\varphi}$. In turn, this implies the existence of a formula $\psi \in \Gamma_{\mathcal{C}}$ such that $\mathtt{V}(\psi) = \{y_0, \ldots, y_{n-1}\}$ (where the variables appear in standard order) and $\varphi = (\psi)_{y_0,\ldots,y_{n-1}:=t_0,\ldots,t_{n-1}}$, with $t_0, \ldots, t_{n-1} \in \mathrm{GTERM}_{\mathcal{L}}$. Since $\psi \in \Gamma_{\mathcal{C}}$, we have $\psi = (\ell_0 \vee \cdots \vee \ell_{m-1})$, where $\{\ell_0, \ldots, \ell_{m-1}\} \in \mathcal{C}$. Since $\varphi = ((\ell_0 \vee \cdots \vee \ell_{m-1}))_{y_0,\ldots,y_{n-1}:=t_0,\ldots,t_{n-1}}$, it follows that

$$C \in \mathcal{C}_{\varphi} = \{\{(\ell_0)_{y_0,\ldots,y_{n-1}:=t_0,\ldots,t_{n-1}}, \ldots, (\ell_{m-1})_{y_0,\ldots,y_{n-1}:=t_0,\ldots,t_{n-1}}\}\}.$$

Therefore, $C \in \mathrm{GINST}_{\mathcal{L}}(\mathcal{C})$.

The converse can be proven similarly.

If $\Box \in \mathcal{C}$, we have

$$\mathrm{GINST}_{\mathcal{L}}(\mathcal{C}) = \mathrm{GINST}_{\mathcal{L}}(\mathcal{C} - \{\Box\}) \cup \{\Box\}.$$

Now, the first part of the theorem implies the second part. $\Box$

The following easy set-theoretical observation is relevant for the next lemma. Let A, B be two sets, $f : A \longrightarrow B$ be a function, and let U be a subset of B. We leave to the reader to verify that

$$f(A - f^{-1}(U)) = f(A) - U. \tag{5.7}$$

Next, we present another lifting result which allows us to transfer the resolvent of ground instances of two clauses to a resolvent of these clauses.

Lemma 5.8.66 (Second Disjoint Lifting Lemma). *Let C_0, C_1 be two ground $\mathcal{L}$-clauses such that $R = res_\ell(C_0, C_1)$, where $\mathcal{L}$ is a first-order language. Suppose that C_0', C_1' are two $\mathcal{L}$-clauses whose sets of variables are disjoint and s_0, s_1 are $\mathcal{L}$-substitutions such that $s_0(C_0') = C_0$ and $s_1(C_1') = C_1$. Then, there is a simple, full most general resolvent R' of C_0' and C_1' and an $\mathcal{L}$-substitution s such that $s(R') = R$.*

Proof. By definition, $\ell \in C_0$ and $\bar{\ell} \in C_1$ and $R = (C_0 - \{\ell\}) \cup (C_1 - \{\bar{\ell}\})$. Define the nonempty sets $L = s_0^{-1}(\ell) \cap C_0'$ and $K = s_1^{-1}(\bar{\ell}) \cap C_1'$. Since C_0' and C_1' have no variables in common, it is possible to extend s_0 and s_1 to a common $\mathcal{L}$-substitution s' defined on $\mathtt{V}(C_0') \cup \mathtt{V}(C_1')$. We have $s'(L) = s_0(L) = \{\ell\}$ and $s'(\overline{K}) = s_1(\overline{K}) = \{\ell\}$, so $L \cup \overline{K}$ is unifiable. Let s_m be a most general unifier of $L \cup \overline{K}$ and let $R' = s_m\left((C_0' - L) \cup (C_1' - K)\right)$. Observe that R' is a simple, most general resolvent of C_0' and C_1' (see Figure 5.38).

Moreover, we claim that R' is a full resolvent of C_0' and C_1'. Indeed, if $\ell' \in L$ and $\ell'' \in C_0' - L$, then $s_0(\ell') = \ell$ and $s_0(\ell'') \neq \ell$, so $s_0(L) \cap s_0(C_0 - L) = \emptyset$. Since s' agrees with s_0 on $\mathtt{V}(C_0)$, we have $s'(L) \cap s'(C_0 - L) = \emptyset$. Since s_m is a most general unifier and s' is an $\mathcal{L}$-unifier of $L \cup \overline{K}$, it is possible to factor s' as $s' = s * s_m$, for some $\mathcal{L}$-substitution s. Then, $s_m(L) \cap s_m(C_0 - L) = \emptyset$. Similarly, $s_m(K) \cap s_m(C_1 - K) = \emptyset$, which means that R' is a full resolvent of C_0' and C_1'.

We claim that $s(R') = R$. Indeed, we have

$$\begin{aligned} s(R') &= s\left(s_m\left((C_0' - L) \cup (C_1' - K)\right)\right) \\ &= s'\left((C_0' - L) \cup (C_1' - K)\right) \end{aligned}$$

$$\begin{tikzcd}
C_0' \arrow[dd, "s_0"'] \arrow[dr, dash] & & C_1' \arrow[dd, "s_1"] \arrow[dl, dash] \\
& R' \arrow[dd, "s"] & \\
C_0 \arrow[dr, dash] & & C_1 \arrow[dl, dash] \\
& R &
\end{tikzcd}$$

Fig. 5.38. Clauses, resolvents and substitutions in Lemma (5.8.66).

$$\begin{aligned}
&= s'(C'_0 - L) \cup s'(C'_1 - K)\\
&= (s'(C'_0) - s'(L)) \cup (s'(C'_1) - s'(K))\\
&\quad \text{(by Equality (5.7))}\\
&= (C_0 - \{\ell\}) \cup (C_1 - \{\bar{\ell}\}) = R.
\end{aligned}$$

□

Lemma 5.8.67 (Second Lifting Lemma). *Let C_0, C_1 be two ground $\mathcal{L}$-clauses such that $R = res_\ell(C_0, C_1)$, where $\mathcal{L}$ is a first-order language. Suppose that C'_0, C'_1 are two $\mathcal{L}$-clauses, and s'_0, s'_1 are $\mathcal{L}$-substitutions such that $s'_0(C'_0) = C_0$ and $s'_1(C'_1) = C_1$. Then, there is a full, most general resolvent R' of C'_0 and C'_1 and an $\mathcal{L}$-substitution s such that $s(R') = R$.*

Proof. The sets of variables of C'_0 and C'_1 may not be disjoint. However, there are renamings s''_0 and s''_1 of the clauses C'_0 and C'_1, respectively, such that $C''_0 = s''_0(C'_0)$ and $C''_1 = s''_1(C'_1)$ have disjoint sets of variables (see Figure 5.39).

By Theorem 5.8.22, there are $\mathcal{L}$-substitutions s_0, s_1 such that $s_0(C''_0) = C'_0$ and $s_1(C''_1) = C'_1$. Note that $s'_0 * s_0(C''_0) = C_0$ and $s'_1 * s_1(C''_1) = C_1$, so by Lemma 5.8.66, there is a simple full most general resolvent R' of C''_0 and C''_1 and an $\mathcal{L}$-substitution s such that $s(R') = R$. Because (s''_0, s''_1) is a standardization of (C'_0, C'_1), the clause R' is a full most general resolvent of C'_0 and C'_1. □

Theorem 5.8.68. *Let $\mathcal{L}$ be a first-order language, $\mathcal{C}$ be a set of $\mathcal{L}$-clauses, and $\varpi = (C_0, \ldots, C_{n-1})$ be a resolution proof over* GINST$_{\mathcal{L}}(\mathcal{C})$. *Let $K \subseteq \{0, \ldots, n-1\}$ be such that K includes all resolution steps of ϖ and for each $k \in K$, let C_k be a resolvent of*

$$\begin{tikzcd}
C'_0 \arrow[r, shift right, "s''_0"'] \arrow[ddr, "s'_0"'] & C''_0 \arrow[l, shift right, "s_0"'] \arrow[dr, dash] & & C''_1 \arrow[r, shift left, "s_1"] \arrow[dl, dash] & C'_1 \arrow[l, shift left, "s''_1"] \arrow[ddl, "s'_1"] \\
 & & R' \arrow[dd, "s"] & & \\
 & C_0 \arrow[dr, dash] & & C_1 \arrow[dl, dash] & \\
 & & R & &
\end{tikzcd}$$

Fig. 5.39. Clauses, resolvents and substitutions in Theorem 5.8.68.

C_{i_k} and C_{j_k} with $i_k, j_k < k$. Then, there is a full most general resolution proof $\varpi' = (C'_0, \ldots, C'_{n-1})$ over $\mathcal{C}$ and $\mathcal{L}$-substitutions s'_i such that $s'_i(C'_i) = C_i$ for $0 \le i \le n-1$. Further, if $k \in K$ then C'_k is a full mgu resolvent of C'_{i_k} and C'_{j_k} and if $k \in \{0, \ldots, n-1\} - K$, then $C'_k \in \mathcal{C}$.

Proof. We construct the proof $(C'_0, \ldots, C'_{n-1})$ recursively. Suppose that $0 \le k \le n-1$ and we have constructed $C'_0, \ldots, C'_{k-1}$. If $k \notin K$, then k is an input step of ϖ, so a $\mathcal{C}$-clause C'_k and an $\mathcal{L}$-substitution s'_k with $s'_k(C'_k) = C_k$ exist by definition of ground instance. If $k \in K$ then we have $\mathcal{C}$-clauses C'_{i_k} and C'_{j_k} and $\mathcal{L}$-substitutions s'_{i_k} and s'_{j_k} with $s'_{i_k}(C'_{i_k}) = C_{i_k}$ and $s'_{j_k}(C'_{j_k}) = C_{j_k}$. By Lemma 5.8.67, there is a clause C'_k and a substitution s'_k such that $s'_k(C'_k) = C_k$ and C'_k is a full most general resolvent of C'_{i_k} and C'_{j_k}. $\square$

Corollary 5.8.69. *Let $\mathcal{L}$ be a first-order language and $\mathcal{C}$ be a set of $\mathcal{L}$-clauses such that $\square \in \mathrm{Res}^*(\mathrm{GINST}_{\mathcal{L}}(\mathcal{C}))$. Then, $\square \in (fRes^{mgu})^*(\mathcal{C})$.*

Proof. This is an immediate consequence of Theorem 5.8.68. $\square$

Theorem 5.8.70 (Resolution Completeness for Languages without Equality). *Let $\mathcal{L}$ be a first-order language without equality and let $\mathcal{C}$ be a set of $\mathcal{L}$-clauses. If $\mathcal{C}$ has no model, then $\square \in (fRes^{mgu})^*(\mathcal{C})$.*

Proof. If $\square \in \mathcal{C}$, the conclusion is immediate. Therefore, suppose that $\square \notin \mathcal{C}$. Recall that we introduced the Herbrand extension $\mathcal{H}(\mathcal{L})$ of $\mathcal{L}$ in Definition 4.10.1. Denote this extension by $\mathcal{L}'$.

Since $\mathcal{C}$ has no model, by Corollary 5.8.13, the set $\Gamma_{\mathcal{C}}$ has no model. Therefore, ${\Gamma_{\mathcal{C}}}^{\forall}$ has no model by Corollary 4.5.60. This, in turn, implies that $\mathrm{GINST}_{\mathcal{L}'}((\Gamma_{\mathcal{C}})^{\forall})$ has no model, by Corollary 4.10.20. Thus, the set of clauses $\mathcal{C}_{\mathrm{GINST}_{\mathcal{L}'}((\Gamma_{\mathcal{C}})^{\forall})}$ has no model as established in Corollary 5.8.13. An application of Theorem 5.8.65 allows us to conclude that $\mathrm{GINST}_{\mathcal{L}'}(\mathcal{C})$ has no model. Consequently, by Theorem 5.8.63, $\square \in \mathrm{Res}^*_{\mathcal{L}'}(\mathrm{GINST}_{\mathcal{L}'}(\mathcal{C}))$. Finally, by Corollary 5.8.69, $\square \in (\mathrm{fRes}^{mgu})^*(\mathcal{C})$. $\square$

In each of the statements ranging from Lemma 5.8.66 to Theorem 5.8.70, we could have used idempotent most general unifiers as provided by the Unification Algorithm (Algorithm 1.6.18). The adjective "idempotent" will be applied to any resolvent obtained using an idempotent unifier.

The next corollary combines soundness and completeness for languages without equality.

Corollary 5.8.71. *Let $\mathcal{L}$ be a first-order language without equality and let $\mathcal{C}$ be a set of $\mathcal{L}$-clauses. The following five statements are equivalent:*

(1) *$\mathcal{C}$ has no model*;
(2) $\Box \in (\mathit{fRes}^{mgu})^*(\mathcal{C})$;
(3) $\Box \in (\mathrm{Res}^{mgu})^*(\mathcal{C})$;
(4) $\Box \in \mathit{fRes}^*_{\mathcal{L}}(\mathcal{C})$;
(5) $\Box \in \mathrm{Res}^*_{\mathcal{L}}(\mathcal{C})$.

Proof. (1) implies (2) follows from the Completeness Theorem (Theorem 5.8.70). The implications (2) implies (3), (2) implies (4), (3) implies (5), and (4) implies (5) are immediate. Finally, (5) implies (1) follows from the Soundness Theorem (Theorem 5.8.54). $\Box$

Example 5.8.72. In this example, we make the point that standardization is essential in the definition of resolvent of two clauses because resolution would not be complete without it. Indeed, consider the set of clauses $\mathcal{C} = \{\{R(x)\}, \{(\neg R(f(x)))\}\}$. Note that the unification algorithm applied to the atomic formulas $R(x)$ and $R(f(x))$ fails because the occurrence check fails (since x occurs in the term $f(x)$). Therefore, to resolve $\{R(x)\}$ and $\{(\neg R(f(x)))\}$ to $\Box$, we rename $\{R(x)\}$ to $\{R(y)\}$ with $y \neq x$ and then unify $\{R(y), R(f(x))\}$ with the substitution $\mathsf{s}^{y}_{f(x)}$. ▯

A resolution step involves, in general, unifying several literals of the clauses involved in this process. If we limit the number of literals to one per clause, the resulting "binary" resolution is not complete when the most general unifier requirement is maintained. However, if this requirement is dropped, then binary resolution is complete.

Definition 5.8.73. Let $\mathrm{Res}_{2\mathcal{L}}$ and Res_2^{mgu} be the analogues of $\mathrm{Res}_{\mathcal{L}}$ and Res^{mgu} when the usual resolution is replaced by binary resolution. If $\mathcal{R}$ is either of these functions, we use the notations $\mathcal{R}^n$ and $\mathcal{R}^*$ with their obvious meanings.

The formal system $\mathcal{FRES}^{2\mathcal{L}}$ is

$$\mathcal{FRES}^{2\mathcal{L}} = (\mathcal{P}_{fin}(\mathrm{LIT}_{\mathcal{L}}), \emptyset, \{\mathsf{R}_2\}),$$

where the set of objects $\mathcal{P}_{fin}(\mathrm{LIT}_{\mathcal{L}})$ consists of finite sets of literals of $\mathcal{L}$ ($\mathcal{L}$-clauses) and the rule R_2 consists of all pairs $((C, D), E)$ where E is a binary $\mathcal{L}$-resolvent of C and D.

Let R_{2mgu} be the analogue of R_{mgu} obtained by replacing resolution by binary resolution and let $\mathcal{FRES}^{2\mathcal{L}}_{mgu}$ be the corresponding resolution formal system. $\square$

Example 5.8.74. Let $\mathcal{L}$ be a first-order language without equality that contains a relation symbol R of positive arity. We claim that $\mathcal{FRES}^{2\mathcal{L}}_{mgu}$ is not complete in the sense of Theorem 5.8.70.

We discuss the case when R is a unary relation symbol. The argument can easily be extended by the reader to the n-ary case. Let $\mathcal{C} = \{\{R(x_0), R(x_1)\}, \{(\neg R(x_2)), (\neg R(x_3))\}\}$. It is clear the $\square$ can be obtained from $\mathcal{C}$ by a single (nonbinary) mgu-resolution, hence $\mathcal{C}$ has no model.

We claim that if $C \in (\mathrm{Res}_2^{mgu})^n(\mathcal{C})$, then C has one of the forms

$$\{R(y_0), R(y_1)\}, \{R(y_0), (\neg R(y_1))\}, \{(\neg R(y_0)), (\neg R(y_1))\},$$

where y_0, y_1 are two distinct variables. The argument is by induction on n. The basis step, $n = 0$, is immediate. Suppose that the statement holds for n and let $C \in (\mathrm{Res}_2^{mgu})^{n+1}(\mathcal{C})$. If $C \in (\mathrm{Res}_2^{mgu})^n(\mathcal{C})$, the property holds. Otherwise, C is a resolvent of two clauses D and E in $(\mathrm{Res}_2^{mgu})^n(\mathcal{C})$. By inductive hypothesis, D and E have one of the prescribed forms. Note that to apply resolution to D and E, D must contain a positive literal and E must contain a negative literal. Therefore, we have four essentially different cases. Suppose, for example, that $D = \{R(y_0), R(y_1)\}$ and $E = \{R(z_0), (\neg R(z_1))\}$, where $y_0 \neq y_1$ and $z_0 \neq z_1$. By the definition of resolution, there is a standardization (s_0, s_1) of (D, E), where $s_0(D) = \{R(y_0'), R(y_1')\}$ and $s_1(E) = \{R(z_0'), (\neg R(z_1'))\}$. Observe that y_0', y_1' and z_0', z_1' remain distinct because s_0 and s_1 are renamings. Also, $\{y_0', y_1'\}$ and $\{z_0', z_1'\}$ are disjoint because (s_0, s_1) is a standardization. Let ℓ_0, ℓ_1 be the positive and negative literals involved in the binary resolution. The literal ℓ_1 must be $(\neg R(z_1'))$ while ℓ_0 can be either $R(y_0')$ or $R(y_1')$. The resolvent C has the form $\{R(s(z)), R(s(z'))\}$ where s is an mgu of $\{\ell_0, \overline{\ell_1}\}$ and z, z' are distinct variables that do not occur in $\{\ell_0, \overline{\ell_1}\}$. By Supplement 80 of Chapter 1, $s(z)$ and $s(z')$ are distinct variables, so C has one of the prescribed forms. We leave the other cases to the reader. $\square$

Lemma 5.8.75. *Let C_0, C_1 be two $\mathcal{L}$-clauses, where $\mathcal{L}$ is a first-order language. If R is an $\mathcal{L}$-resolvent of C_0, C_1 obtained using an idempotent unifier, then $R \in \mathrm{Res}^*_{2\mathcal{L}}(\{C_0, C_1\})$.*

Proof. By definition of resolvent, there are a standardization (s_0, s_1) of (C_0, C_1), a nonempty set of positive literals $L \subseteq \bar{s}_0(C_0)$, $L = \{\ell_0, \ldots, \ell_{p-1}\}$, a nonempty set of negative literals $K \subseteq \bar{s}_1(C_1)$, $K = \{(\neg k_0), \ldots, (\neg k_{r-1})\}$ and an idempotent $\mathcal{L}$-unifier s of $L \cup \overline{K}$ such that

$$R = \bar{s}\left((\bar{s}_0(C_0) - L) \cup (\bar{s}_1(C_1) - K)\right).$$

Since $\bar{s}$ is a unifier of $L \cup \overline{K}$, there is a literal ℓ such that $\bar{s}(L \cup \overline{K}) = \{\ell\}$. Because $\bar{s}$ is idempotent, we also have $\bar{s}(\ell) = \ell$ and $\bar{s}(l) = l$, for every $l \in R$.

We consider four cases.

`Case 1:` $p = 1$ `and` $r = 1$. In this case, there is nothing to prove because the resolution is binary.

`Case 2:` $p > 1$ `and` $r > 1$. Since $\ell_0 \in \bar{s}_0(C_0)$, $(\neg k_0) \in \bar{s}_1(C_1)$ and s is an $\mathcal{L}$-unifier (although not necessarily a most general unifier) of $\{\ell_0, k_0\}$, we have the binary $\mathcal{L}$-resolvent

$$R' = \bar{s}(\bar{s}_0(C_0) - \{\ell_0\}) \cup \bar{s}(\bar{s}_1(C_1) - \{(\neg k_0)\}).$$

We claim that $R' = R \cup \{\ell, (\neg \ell)\}$. It is clear that $R \subseteq R'$. In addition, $\ell = \bar{s}(\ell_1) \in R'$ because $p > 1$; also, $(\neg \ell) = (\neg \bar{s}(k_1)) \in R'$ because $k > 1$. Thus, $R' \supseteq R \cup \{\ell, (\neg \ell)\}$. To prove the converse inclusion, let $l \in \bar{s}(\bar{s}_0(C_0) - \{\ell_0\})$. If $l \in \bar{s}(\bar{s}_0(C_0) - \{\ell_0, \ldots, \ell_{p-1}\})$, then $l \in R$. Otherwise, $l = \bar{s}(\ell_i) = \ell$, for some i, $1 \leq i \leq p-1$, so $l \in R \cup \{\ell, (\neg \ell)\}$. The case when $l \in \bar{s}(\bar{s}_0(C_1) - \{(\neg k_0)\})$ is similar, which proves the converse inclusion.

We now consider two subcases.

`Case 2.1:` $\ell \notin R$. We now apply weak binary $\mathcal{L}$-resolution to R' and C_1, by using the identity renaming ι for R' and the renaming s_1

for C_1. We have $\ell \in R'$ and $(\neg k_1) \in \bar{s}_1(C_1)$ and due to the idempotency of $\bar{s}$, we have that s is an $\mathcal{L}$-unifier of $\{\ell, k_1\}$, so we can apply weak binary $\mathcal{L}$-resolution to obtain

$$\begin{aligned} R'' &= \bar{s}(R' - \{\ell\}) \cup \bar{s}(\bar{s}_1(C_1) - \{(\neg k_1)\}) \\ &= \bar{s}((R \cup \{\ell, (\neg\ell)\}) - \{\ell\}) \cup \bar{s}(\bar{s}_1(C_1) - \{(\neg k_1)\}) \\ &= \bar{s}(R \cup \{(\neg\ell)\}) \cup \bar{s}(\bar{s}_1(C_1) - \{(\neg k_1)\}) \\ &= R \cup \{(\neg\ell)\} \cup \bar{s}(\bar{s}_1(C_1) - \{(\neg k_1)\}) \end{aligned}$$

We claim that $R'' = R \cup \{(\neg\ell)\}$. To justify this claim, we need to show only that

$$R \cup \{(\neg\ell)\} \supseteq \bar{s}(\bar{s}_1(C_1) - \{(\neg k_1)\}).$$

Let $l \in \bar{s}(\bar{s}_1(C_1) - \{(\neg k_1)\})$, say $l = \bar{s}(l')$. If $l' \notin \{(\neg k_0), \ldots, (\neg k_r)\}$, then $l \in R$. Otherwise, $l' = (\neg\ell)$. Note that by Lemma 5.8.38, R'' can be obtained from R' and C_1 by binary $\mathcal{L}$-resolution.

If $(\neg\ell) \in R$, we have $R'' = R$ and we are done. Otherwise, we apply weak binary $\mathcal{L}$-resolution to C_0 and R', by using the identity renaming ι for R' and the renaming s_0 for C_0. We have $(\neg\ell) \in R'$ and $\ell_1 \in \bar{s}_0(C_0)$ and due to the idempotency of $\bar{s}$, we have that s is an $\mathcal{L}$-unifier of $\{\ell, \ell_1\}$, so we can apply weak binary $\mathcal{L}$-resolution to obtain

$$\begin{aligned} R''' &= \bar{s}(R' - \{(\neg\ell)\}) \cup \bar{s}(\bar{s}_0(C_0) - \{\ell_1\}) \\ &= \bar{s}((R \cup \{\ell, (\neg\ell)\}) - \{(\neg\ell)\}) \cup \bar{s}(\bar{s}_0(C_0) - \{\ell_1\}) \\ &= \bar{s}(R \cup \{\ell\}) \cup \bar{s}(\bar{s}_0(C_0) - \{\ell_1\}) \\ &= R \cup \{\ell\} \cup \bar{s}(\bar{s}_0(C_0) - \{\ell_1\}) \end{aligned}$$

We claim that $R''' = R \cup \{\ell\}$. To justify this claim, we need to show only that

$$R \cup \{\ell\} \supseteq \bar{s}(\bar{s}_0(C_0) - \{\ell_1\}).$$

Let $l \in \bar{s}(\bar{s}_0(C_0) - \{\ell_1\})$, say $l = \bar{s}(l')$. If $l' \notin \{\ell_0, \ldots, \ell_p\}$, then $l \in R$. Otherwise, $l' = \ell$. Again, by Lemma 5.8.38, R''' can be obtained from C_0 and R' by binary $\mathcal{L}$-resolution.

Since $\ell \notin R$ and $(\neg \ell) \notin R$, a final weak binary $\mathcal{L}$-resolution step involving R''' and R'' yields R. A final application of Lemma 5.8.38 shows that R is a binary $\mathcal{L}$-resolvent of R''' and R''.

`Case 2.2:` $\ell \in R$. If $(\neg \ell) \in R$, then $R' = R$ and we are finished. If $(\neg \ell) \notin R$, then we obtain $R''' = R \cup \{\ell\}$ as in Case 2.1 and since $\ell \in R$, $R''' = R$.

`Case 3:` $p = 1$ `and` $r > 1$. Since $\ell_0 \in \bar{s}_0(C_0)$, $(\neg k_0) \in \bar{s}_1(C_1)$ and s is an $\mathcal{L}$-unifier (although not necessarily a most general unifier) of $\{\ell_0, k_0\}$, we have the binary $\mathcal{L}$-resolvent

$$R' = \bar{s}(\bar{s}_0(C_0) - \{\ell_0\}) \cup \bar{s}(\bar{s}_1(C_1) - \{(\neg k_0)\}).$$

By an argument similar to the one used in Case 2, we have $R' = R \cup \{(\neg \ell)\}$. If $(\neg \ell) \in R$, we have $R' = R$ and we are done.

Suppose $(\neg \ell) \notin R$. We apply weak binary $\mathcal{L}$-resolution to C_0 and R', by using the identity renaming ι for R' and the renaming s_0 for C_0. We have $(\neg \ell) \in R'$ and $\ell_0 \in \bar{s}_0(C_0)$ and due to the idempotency of $\bar{s}$, we have that s is an $\mathcal{L}$-unifier of $\{\ell, \ell_0\}$, so we can apply weak binary $\mathcal{L}$-resolution to obtain

$$\begin{aligned} R'' &= \bar{s}(R' - \{(\neg \ell)\}) \cup \bar{s}(\bar{s}_0(C_0) - \{\ell_0\}) \\ &= \bar{s}((R \cup \{(\neg \ell)\}) - \{(\neg \ell)\}) \cup \bar{s}(\bar{s}_0(C_0) - \{\ell_0\}) \\ &= \bar{s}(R) \cup \bar{s}(\bar{s}_0(C_0) - \{\ell_0\}) \\ &= R \cup \bar{s}(\bar{s}_0(C_0) - \{\ell_0\}). \end{aligned}$$

We have $R'' = R$ because $\bar{s}(\bar{s}_0(C_0) - \{\ell_0\}) \subseteq R$. By Lemma 5.8.38, $R'' = R$ is a binary $\mathcal{L}$-resolvent of C_0 and R'.

`Case 4:` $p > 1$ `and` $r = 1$. This last case is similar to case 3 and is left to the reader. □

Lemma 5.8.76. *Let $\mathcal{L}$ be a first-order language and $\mathcal{C}$ be a set of $\mathcal{L}$-clauses. If there is an $\mathcal{L}$-resolution proof of C over $\mathcal{C}$ that uses idempotent unifiers at every resolution step, then $C \in \mathrm{Res}^*_{2\mathcal{L}}(\mathcal{C})$.*

Proof. We are going to show that if there is an $\mathcal{L}$-resolution proof of C over $\mathcal{C}$ of length n that uses idempotent unifiers at every resolution step, then $C \in \mathrm{Res}^*_{2\mathcal{L}}(\mathcal{C})$.

We use course-of-values induction on the length n of the proof. Suppose the result is true for all m with $1 \leq m < n$ and let C be a clause that is obtained by a proof over $\mathcal{C}$ using idempotent unifiers and having length n. If $C \in \mathcal{C}$, the result is immediate. Otherwise, C is obtained as an $\mathcal{L}$-resolvent of C', C'' which have proofs of length less than n using idempotent unifiers. By the inductive hypothesis, $C', C'' \in \mathrm{Res}^*_{2\mathcal{L}}(\mathcal{C})$, and by Lemma 5.8.75, $C \in \mathrm{Res}^*_{2\mathcal{L}}(\{C', C''\})$. Thus, $C \in \mathrm{Res}^*_{2\mathcal{L}}(\mathcal{C})$. $\square$

The previous lemma is a preliminary for a completeness result for binary resolution.

Theorem 5.8.77. *Let $\mathcal{L}$ be a first-order language without equality and $\mathcal{C}$ be a set of $\mathcal{L}$-clauses. If $\mathcal{C}$ has no model, then $\square \in \mathrm{Res}^*_{2\mathcal{L}}(\mathcal{C})$.*

Proof. By the proof of Theorem 5.8.70, since $\mathcal{C}$ has no model, $\square \in \mathrm{Res}^*_{\mathcal{L}}(\mathcal{C})$ by a proof using idempotent unifiers at every step. By Lemma 5.8.76, $\square \in \mathrm{Res}^*_{2\mathcal{L}}(\mathcal{C})$. $\square$

5.9 Variations of First-Order Resolution

We now examine restrictions on clauses and on proof formats similar to the ones considered in propositional logic (see Section 3.9) and examine their completeness.

Definition 5.9.1. Let $\mathcal{L}$ be a first-order language and $\mathcal{C}$ be a set of $\mathcal{L}$-clauses.

- A *positive $\mathcal{L}$-resolution proof* (*negative $\mathcal{L}$-resolution proof*) over $\mathcal{C}$ is an $\mathcal{L}$-resolution proof over $\mathcal{C}$ such that for every resolution step, there is a pair of premises such that one of the premises is positive (negative).
- A *linear $\mathcal{L}$-resolution proof* is an $\mathcal{L}$-resolution proof $(C_0, \ldots, C_{n-1})$ over $\mathcal{C}$, such that for some k, $0 \leq k \leq n-1$, $C_0, \ldots, C_k$ belong to $\mathcal{C}$ and for every j such that $k < j \leq n$, C_j is obtained as a resolvent of two predecessors one of which is C_{j-1}.
  If for each j such that $k < j \leq n-1$, C_j is a resolvent of C_{j-1} and some C_h with $0 \leq h \leq k$, then $(C_0, \ldots, C_n)$ is an *input $\mathcal{L}$-resolution proof* over $\mathcal{C}$.

▯

Next we consider several other variations of first-order resolution which correspond to previously considered resolution methods of propositional logic.

Definition 5.9.2. Let $\mathcal{L}$ be a first-order language and $\mathcal{C}$ be a set of $\mathcal{L}$-clauses.

- Let $\mathcal{A}$ be an $\mathcal{L}$-structure. An *$\mathcal{A}$-semantic $\mathcal{L}$-resolution proof over $\mathcal{C}$* is an $\mathcal{L}$-resolution proof over $\mathcal{C}$ such that if i is a resolution step, then there are premises C_j, C_k for step i such that $\mathcal{A} \not\models C_j$ or $\mathcal{A} \not\models C_k$.
- A subset $\mathcal{D}$ of $\mathcal{C}$ is a *set-of-support* for $\mathcal{C}$ if $\mathcal{C} - \mathcal{D}$ has a model. A $\mathcal{D}$-set-of-support $\mathcal{L}$-resolution proof over $\mathcal{C}$ is an $\mathcal{L}$-resolution proof over $\mathcal{C}$ such that for every resolution step of the proof there is a pair of premises at least one of which does not belong to $\mathcal{C} - \mathcal{D}$.

$\square$

Next we discuss a sharpening of the First Lifting Lemma (Lemma 5.8.61).

Lemma 5.9.3. *Let $\mathcal{L}$ be a first-order language and let s be an injective, atomic $\mathcal{L}$-inter-substitution.*

If $(C_0, \ldots, C_{n-1})$ is a positive (negative, linear, input) propositional resolution proof over a set of propositional clauses $\mathcal{C}$, then $(s(C_0), \ldots, s(C_{n-1}))$ is an $\mathcal{L}$-resolution proof over $s(\mathcal{C})$ of the same type. Further, i is an input step of the proof $(C_0, \ldots, C_{n-1})$ if and only if it is an input step of the proof $(s(C_0), \ldots, s(C_{n-1}))$, for $0 \leq i \leq n-1$.

Proof. Lemma 5.8.61 implies that $(s(C_0), \ldots, s(C_{n-1}))$ is an $\mathcal{L}$-resolution proof over $s(\mathcal{C})$ preserving input steps and by Lemma 5.8.60, if $0 \leq i, j < k \leq n-1$ and C_k is a resolvent of C_i and C_j, then $s(C_k)$ is a resolvent of $s(C_i)$ and $s(C_j)$ which implies that the type of the lifted proof is the same as the type of the original propositional proof. $\square$

Yet another lifting result is the following.

Lemma 5.9.4. *Let $\mathcal{L}$ be a first-order language, $\mathcal{A}$ be an $\mathcal{L}$-structure, $\mathcal{C}$ be a set of propositional clauses and let s be an injective, atomic*

$\mathcal{L}$-inter-substitution. Define a truth assignment v as

$$v(p) = \begin{cases} \mathbf{T} & \textit{if } \mathcal{A} \models s(p), \\ \mathbf{F} & \textit{if } \mathcal{A} \not\models s(p). \end{cases}$$

If $s(C_0), \ldots, s(C_{n-1})$ are all ground clauses and $(C_0, \ldots, C_{n-1})$ is a v-semantic resolution proof over $\mathcal{C}$, then

$$(s(C_0), \ldots, s(C_{n-1}))$$

is an $\mathcal{A}$-semantic $\mathcal{L}$-resolution proof over $s(\mathcal{C})$.

Proof. By Lemma 5.8.61, $(s(C_0), \ldots, s(C_{n-1}))$ is an $\mathcal{L}$-resolution proof over $s(\mathcal{C})$. To verify that the lifted proof $(s(C_0), \ldots, s(C_{n-1}))$ is $\mathcal{A}$-semantic, let $\sigma \in \text{ASSIGN}_{\mathcal{A}}$ and let v' be the truth assignment defined as

$$v'(p) = \begin{cases} \mathbf{T} & \text{if } (\mathcal{A}, \sigma) \models s(p), \\ \mathbf{F} & \text{if } (\mathcal{A}, \sigma) \not\models s(p). \end{cases}$$

If a variable p occurs in a propositional clause C_i, then $s(p)$ is a ground atomic formula because $s(C_i)$ is a ground clause. Thus, $v'(p) = v(p)$ for every variable in C_i. Therefore, we have the following equivalent statements:

(1) v satisfies C_i;
(2) v' satisfies C_i;
(3) $(\mathcal{A}, \sigma) \models s(C_i)$ (by Lemma 5.8.57);
(4) $\mathcal{A} \models s(C_i)$ (because $s(C_i)$ is a ground clause).

The result follows immediately. $\square$

The sharpening of the Completeness of Resolution for Ground Clauses and Languages without Equality (Theorem 5.8.63) is discussed next.

Theorem 5.9.5. *Let $\mathcal{L}$ be a first-order language without equality and let $\mathcal{C}$ be a set of ground $\mathcal{L}$-clauses that has no model. The following statements hold:*

- *There is a positive (negative, linear) $\mathcal{L}$-resolution proof of $\square$ over $\mathcal{C}$.*
- *If $\mathcal{C}$ consists of Horn clauses, then there is an input $\mathcal{L}$-resolution proof of $\square$ over $\mathcal{C}$.*

- *For each $\mathcal{L}$-structure $\mathcal{A}$, there is an $\mathcal{A}$-semantic $\mathcal{L}$-resolution proof of $\Box$ over $\mathcal{C}$.*

Proof. The argument proceeds as in Theorem 5.8.63. As we have shown, there is a fundamental propositional form $\mathcal{C}_0$ for $\mathcal{C}$ by Theorem 5.8.7. Let s be an injective atomic $\mathcal{L}$-intersubstitution such that $s(\mathcal{C}_0) = \mathcal{C}$. Since $\mathcal{C}$ consists of ground clauses and has no model, it follows from Corollary 5.8.14 that $\mathcal{C}$ is unsatisfiable. This in turn implies that $\mathcal{C}_0$ is unsatisfiable, by Theorem 5.8.59.

By Corollary 3.9.3, there is a positive (negative) propositional resolution proof of $\Box$ over $\mathcal{C}_0$. By Lemma 5.9.3, the image of this proof under s is a positive (negative) $\mathcal{L}$-resolution proof of $\Box$ over $s(\mathcal{C}_0) = \mathcal{C}$.

For the linear $\mathcal{L}$-resolution proof, the argument is similar except for the use of Corollary 3.9.14 in place of Corollary 3.9.3.

For the second part of the theorem, note that if $\mathcal{C}$ consists of Horn clauses, then so does $\mathcal{C}_0$. The statement follows by the previous argument by using Corollary 3.9.19.

For the third part of the theorem, let $\mathcal{A}$ be an $\mathcal{L}$-structure and define a truth assignment v as

$$v(p) = \begin{cases} \mathbf{T} & \text{if } \mathcal{A} \models s(p), \\ \mathbf{F} & \text{if } \mathcal{A} \not\models s(p). \end{cases}$$

By Corollary 3.9.22 there is a v-semantic resolution proof of $\Box$ over $\mathcal{C}_0$. By Lemma 5.9.4, the image of this proof under s is an $\mathcal{A}$-semantic $\mathcal{L}$-resolution proof of $\Box$ over $\mathcal{C}$. $\Box$

Theorem 5.9.6. *Let $\mathcal{L}$ be a first-order language, $\mathcal{C}$ be a set of $\mathcal{L}$-clauses and let $\mathcal{A}$ be an $\mathcal{L}$-structure. If $\Box$ has a positive (negative, linear, input, $\mathcal{A}$-semantic) $\mathcal{L}$-resolution proof over* $\mathrm{GINST}_{\mathcal{L}}(\mathcal{C})$*, then $\Box$ has a positive (negative, linear, input, $\mathcal{A}$-semantic) full, most general resolution proof over $\mathcal{C}$.*

Proof. In the case of positive (negative, linear, input) $\mathcal{L}$-resolution proofs, the statement follows from Theorem 5.8.68. The case of $\mathcal{A}$-semantic $\mathcal{L}$-resolution proofs follows from Theorems 5.8.68 and 5.8.23. $\Box$

Theorem 5.9.7. *Let $\mathcal{L}$ be a first-order language without equality, $\mathcal{C}$ be a set of $\mathcal{L}$-clauses that has no model and $\mathcal{A}$ be an $\mathcal{L}$-structure. Then* *there is a positive (negative, linear, $\mathcal{A}$-semantic) full most general resolution proof of $\Box$ over $\mathcal{C}$.*

If $\mathcal{C}$ consists of Horn clauses, then there is an input full most general resolution proof of $\Box$ over $\mathcal{C}$.

Proof. Our argument follows the same lines as the proof of Theorem 5.8.70. If $\Box \in \mathcal{C}$, the conclusion is immediate. Therefore, suppose that $\Box \notin \mathcal{C}$. As before we denote the Herbrand extension of $\mathcal{L}$ by $\mathcal{L}'$. Let $\mathcal{A}'$ be an arbitrary expansion of $\mathcal{A}$ to $\mathcal{L}'$.

We can conclude that $\mathrm{GINST}_{\mathcal{L}'}(\mathcal{C})$ has no model. By Theorem 5.9.5, there is a positive (negative, linear, $\mathcal{A}'$-semantic) $\mathcal{L}'$-resolution proof of $\Box$ over $\mathrm{GINST}_{\mathcal{L}'}(\mathcal{C})$. If $\mathcal{C}$ consists of Horn clauses, then the ground instances of $\mathcal{C}$ are also Horn clauses and hence there is an input $\mathcal{L}'$-resolution of $\Box$ over $\mathrm{GINST}_{\mathcal{L}'}(\mathcal{C})$.

The result now follows from Theorem 5.9.6, noting that $\mathcal{A}$ and $\mathcal{A}'$ agree on all symbols occurring in a most general resolution proof over a set of $\mathcal{L}$-clauses. $\Box$

Theorem 5.9.8. *Let $\mathcal{L}$ be a first-order language without equality, $\mathcal{C}$ be a set of $\mathcal{L}$-clauses and $\mathcal{D} \subseteq \mathcal{C}$ be a set of support for $\mathcal{C}$. If $\mathcal{C}$ has no model, then there is a $\mathcal{D}$-set-of-support full most general resolution proof of $\Box$ over $\mathcal{C}$.*

Proof. Since $\mathcal{D}$ is a set-of-support for $\mathcal{C}$, $\mathcal{C} - \mathcal{D}$ has a model. Thus, there exists an $\mathcal{L}$-structure $\mathcal{A}$ such that $\mathcal{A}$ is a model of $\mathcal{C} - \mathcal{D}$. By Theorem 5.9.7, there exists an $\mathcal{A}$-semantic full most general resolution proof of $\Box$ over $\mathcal{C}$. Note that in every resolution step of this proof, there is a premise C such that $\mathcal{A}$ is not a model for C, that is, a clause not from $\mathcal{C} - \mathcal{D}$. Therefore, this proof is a full most general $\mathcal{D}$-set-of-support proof of $\Box$ over $\mathcal{C}$. $\Box$

We introduce now the notion of hyperresolution which is similar to the corresponding notion from propositional logic.

Definition 5.9.9. Let $\mathcal{L}$ be a first-order language. A positive $\mathcal{L}$-clause C is an *$\mathcal{L}$-hyperresolvent* (*full $\mathcal{L}$-hyperresolvent*, *most general $\mathcal{L}$-hyperresolvent*) of a sequence of $\mathcal{L}$-clauses $C_1, \ldots, C_n, D$, where $n \geq 1$ and $C_1, \ldots, C_n$ are positive, if there is a sequence of $\mathcal{L}$-clauses $E_0, \ldots, E_n$ such that the following conditions are satisfied:

(1) $E_0 = C$ and $E_n = D$;

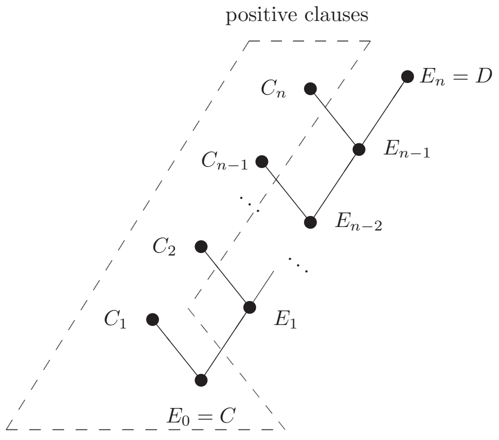


Fig. 5.40. Construction of an $\mathcal{L}$-hyperresolvent of a sequence of $\mathcal{L}$-clauses.

(2) for $0 \le i \le n-1$, E_i is an $\mathcal{L}$-resolvent (full $\mathcal{L}$-resolvent. most general resolvent) of C_{i+1} and E_{i+1}.

Figure 5.40 shows the construction of an $\mathcal{L}$-hyperresolvent of a sequence of $\mathcal{L}$-clauses.

Let $\mathcal{C}$ be a set of $\mathcal{L}$-clauses. An *$\mathcal{L}$-hyperresolution proof* over $\mathcal{C}$ is a finite sequence $(C_0, C_1, \ldots, C_{n-1})$ of $\mathcal{L}$-clauses such that $n \ge 1$ and for each step i, $0 \le i \le n-1$ either $C_i \in \mathcal{C}$ or else $C_i \notin \mathcal{C}$ and there is a sequence $j_0, \ldots, j_{p-1} < i$ such that C_i is an $\mathcal{L}$-hyperresolvent of $C_{j_0}, \ldots, C_{j_{p-1}}$. In the first case, i is an *input step* of the proof; in the second case, i is a *hyperresolution step*.

If at each hyperresolution step, the clause C_i is a full (most general) $\mathcal{L}$-hyperresolvent, then the sequence $(C_0, C_1, \ldots, C_{n-1})$ is a *full $\mathcal{L}$-hyperresolution proof over $\mathcal{C}$* (*most general $\mathcal{L}$-hyperresolution proof over $\mathcal{C}$*).

An *$\mathcal{L}$-hyperresolution proof of a clause C* over $\mathcal{C}$ is an $\mathcal{L}$-hyperresolution proof over $\mathcal{C}$ whose last entry is C. ☐

Lemma 5.9.10. *Let $\mathcal{L}, \mathcal{L}'$ be first-order languages and let $C_1, \ldots, C_n, D$ be $\mathcal{L} \cap \mathcal{L}'$-clauses. Then C is a most general $\mathcal{L}$-hyperresolvent of $C_1, \ldots, C_n, D$ if and only if C is a most general $\mathcal{L}'$-hyperresolvent of the same clauses.*

Proof. Suppose that C is a most general $\mathcal{L}$-hyperresolvent of $C_1, \ldots, C_n, D$. By the definition of hyperresolvent, we have $\mathcal{L}$-clauses

$$C = E_0, E_1, \ldots, E_n = D$$

such that for $0 \leq i \leq n-1$, E_i is a most general $\mathcal{L}$-resolvent of C_{i+1} and E_{i+1}. We prove by induction on k with $1 \leq k \leq n$, that E_{n-k} is a an $\mathcal{L}'$-clause. Since C_n and $E_n = D$ are $\mathcal{L}'$-clauses, by Corollary 5.8.35, E_{n-1} is a most general $\mathcal{L}'$-resolvent of C_n and E_n and hence is an $\mathcal{L}'$-clause.

Suppose now that the result holds for $k < n$. Since E_{n-k} is an $\mathcal{L}'$-clause, it follows again from Corollary 5.8.35 that E_{n-k-1} is a most general $\mathcal{L}'$-resolvent of C_{n-k} and E_{n-k} and hence is an $\mathcal{L}'$-clause. We can conclude that C is a most general $\mathcal{L}'$-hyperresolvent of $C_1, \ldots, C_n, D$. The argument for the reverse inclusion can be obtained by swapping $\mathcal{L}$ and $\mathcal{L}'$. $\square$

In view of Lemma 5.9.10, we can refer to a clause C as being a most general hyperresolvent of a sequence of clauses without reference to a language.

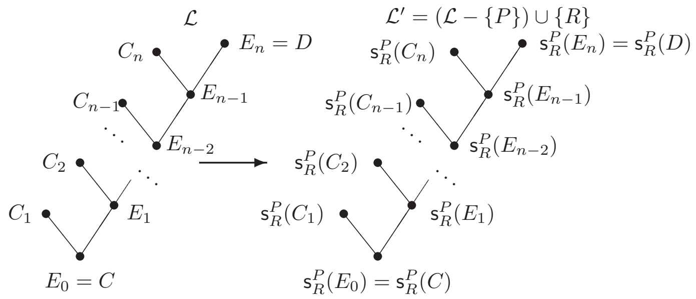
Theorem 5.9.11. *Let $\mathcal{L}, \mathcal{L}'$ be two first-order languages and let $\mathcal{C}$ be a set of $\mathcal{L} \cap \mathcal{L}'$-clauses. Then, a sequence of clauses $(C_0, \ldots, C_{n-1})$ is a most general $\mathcal{L}$-hyperresolution proof over $\mathcal{C}$ if and only if it is a most general $\mathcal{L}'$-hyperresolution proof over $\mathcal{C}$.*

Proof. Suppose that $(C_0, \ldots, C_{n-1})$ is a most general $\mathcal{L}$-hyperesolution proof over $\mathcal{C}$. We show by course-of-values induction that each C_i is also an $\mathcal{L}'$-clause and therefore the same sequence $(C_0, \ldots, C_{n-1})$ is a most general $\mathcal{L}'$-hyperresolution proof over $\mathcal{C}$.

Suppose that for $0 \leq j < i \leq n-1$, each clause C_j is an $\mathcal{L}'$-clause. If $C_i \in \mathcal{C}$, then by hypothesis, C_i is an $\mathcal{L}'$-clause. If C_i is a most general hyperresolvent of $C_{j_0}, \ldots, C_{j_{p-1}}$ with $j_0, \ldots, j_{p-1} < i$, then, by inductive hypothesis, $C_{j_0}, \ldots, C_{j_{p-1}}$ are all $\mathcal{L}'$-clauses, so by Lemma 5.9.10, C_i is an $\mathcal{L}'$-clause. $\square$

In view of Theorem 5.9.11, if $\mathcal{C}$ is an admissible set of clauses, we can refer to a most general hyperresolution proof over $\mathcal{C}$ without reference to a specific language.

Lemma 5.9.12. *Let $\mathcal{L}$ be a first-order language and let P and R be two relation symbols of equal arity such that $P \in \mathcal{L}$ and*



$\mathcal{L}$
C_n
$E_n = D$
E_{n-1}
C_{n-1}
E_{n-2}
C_2
C_1
E_1
$E_0 = C$
$\mathcal{L}' = (\mathcal{L} - \{P\}) \cup \{R\}$
$\mathsf{s}_R^P(C_n)$
$\mathsf{s}_R^P(E_n) = \mathsf{s}_R^P(D)$
$\mathsf{s}_R^P(E_{n-1})$
$\mathsf{s}_R^P(C_{n-1})$
$\mathsf{s}_R^P(E_{n-2})$
$\mathsf{s}_R^P(C_2)$
$\mathsf{s}_R^P(C_1)$
$\mathsf{s}_R^P(E_1)$
$\mathsf{s}_R^P(E_0) = \mathsf{s}_R^P(C)$

Fig. 5.41. Preservation of hyperresolvents under relation symbol substitution.

$R \notin \mathcal{L}$. Define the first-order language $\mathcal{L}' = (\mathcal{L} - \{P\}) \cup \{R\}$. Let $C_1, \ldots, C_n, D$ be $\mathcal{L}$-clauses and C be an $\mathcal{L}$-hyperresolvent (full $\mathcal{L}$-hyperresolvent, most general hyperresolvent) of $C_1, \ldots, C_n, D$. Then, $\mathsf{s}_R^P(C)$ is an $\mathcal{L}'$-hyperresolvent (full $\mathcal{L}'$-hyperresolvent, most general hyperresolvent) of the sequence

$$(\mathsf{s}_R^P(C_1), \ldots, \mathsf{s}_R^P(C_n))$$

and $\mathsf{s}_R^P(D)$ (see Figure 5.41).

Proof. First note that $\mathsf{s}_R^P(C_1), \ldots, \mathsf{s}_R^P(C_n), \mathsf{s}_R^P(C)$ are positive since the clauses $C_1, \ldots, C_n, C$ are positive.

Further, there are $\mathcal{L}$-clauses $E_0, \ldots, E_n$ with $E_0 = C$, $E_n = D$ and E_{i-1} is an $\mathcal{L}$-resolvent (full $\mathcal{L}$-resolvent, most general resolvent) of C_i and E_i for $1 \leq i \leq n$. Then, $\mathsf{s}_R^P(E_i)$ is an $\mathcal{L}'$-clause for $0 \leq i \leq n$, $\mathsf{s}_R^P(E_0) = \mathsf{s}_R^P(C)$, $\mathsf{s}_R^P(E_n) = \mathsf{s}_R^P(D)$, and by Lemma 5.8.49, $\mathsf{s}_R^P(E_{i-1})$ is a (full, most general) $\mathcal{L}'$-resolvent of $\mathsf{s}_R^P(C_i)$ and $\mathsf{s}_R^P(E_i)$ for $1 \leq i \leq n$. □

Theorem 5.9.13. *Let $\mathcal{C}$ be a set of $\mathcal{L}$-clauses and let P and R be two relation symbols of equal arity such that $P \in \mathcal{L}$ and $R \notin \mathcal{L}$. Define the first-order language $\mathcal{L}' = (\mathcal{L} - \{P\}) \cup \{R\}$. If $(C_0, \ldots, C_{n-1})$ is an $\mathcal{L}$-hyperresolution (full $\mathcal{L}$-hyperresolution, most general hyperresolution) proof of C over $\mathcal{C}$, then $(\mathsf{s}_R^P(C_0), \ldots, \mathsf{s}_R^P(C_{n-1}))$ is an $\mathcal{L}'$-hyperresolution (full $\mathcal{L}'$-hyperresolution, most general hyperresolution) proof of $\mathsf{s}_R^P(C)$ over $\mathsf{s}_R^P(\mathcal{C})$.*

Proof. The argument is based on Lemma 5.9.12 and it is entirely similar to the proof of Theorem 5.8.50. □

Corollary 5.9.14. *Let $\mathcal{C}$ be a set of $\mathcal{L}$-clauses, C be an $\mathcal{L}$-clause, and let P and R be two relation symbols of equal arity such that $P \in \mathcal{L}$ and $R \notin \mathcal{L}$. Define the first-order language $\mathcal{L}' = (\mathcal{L} - \{P\}) \cup \{R\}$. There is a (full, most general) $\mathcal{L}$-hyperresolution proof of C over $\mathcal{C}$ if and only if there is a (full, most general) $\mathcal{L}'$-hyperresolution proof of $\mathsf{s}_R^P(C)$ over $\mathsf{s}_R^P(\mathcal{C})$.*

Proof. This follows directly from Theorem 5.9.13. □

Lemma 5.9.15. *Let $\mathcal{L}$ be a first-order language and let C_0, C_1, C be $\mathcal{L}$-clauses such that C is an $\mathcal{L}$-resolvent (a full $\mathcal{L}$-resolvent, a most general resolvent) of C_0, C_1. If s_0, s_1 are renamings of C_0, C_1, respectively, and $C_i' = s_i(C_i)$ for $i \in \{0,1\}$, then C is an $\mathcal{L}$-resolvent (a full $\mathcal{L}$-resolvent, a most general resolvent) of C_0', C_1'.*

Proof. By the definition of resolvent, we have a standardization (s_0', s_1') of (C_0, C_1), a set of positive literals $L \subseteq s_0'(C_0)$, and a set of negative literals $K \subseteq s_1'(C_1)$ with $C = s(s_0'(C_0) - L) \cup s(s_1'(C_1) - K)$, where s is an $\mathcal{L}$-unifier of $L \cup \overline{K}$. By Theorem 5.8.22, C_i is a renaming of $s_i(C_i)$ for $i \in \{0,1\}$ and by the same theorem, $s_i'(C_i)$ is a renaming of $s_i(C_i)$, for $i \in \{0,1\}$. Denote by s_i'' a renaming of $s_i(C_i)$ with $s_i''(s_i(C_i)) = s_i'(C_i)$ for $i \in \{0,1\}$, and observe that (s_0'', s_1'') is a standardization of $(s_0(C_0), s_1(C_1))$, because (s_0', s_1') is a standardization of (C_0, C_1). This shows that C is also an $\mathcal{L}$-resolvent of $s_0(C_0)$ and $s_1(C_1)$.

The same argument works for full resolvents and most general resolvents. □

Theorem 5.9.16. *Let $\mathcal{L}$ be a first-order language and let $C, C_1, \ldots, C_n, D$ be $\mathcal{L}$-clauses such that C is an $\mathcal{L}$-hyperresolvent (a full $\mathcal{L}$-hyperresolvent, a most general hyperresolvent) of $C_1, \ldots, C_n, D$. If $s_1, \ldots, s_n, z$ are renamings of $C_1, \ldots, C_n, D$, respectively, $C_i' = s_i(C_i)$ for $1 \leq i \leq n$ and $D' = z(D)$, then C is an $\mathcal{L}$-hyperresolvent (a full $\mathcal{L}$-hyperresolvent, a most general hyperresolvent) of $C_1', \ldots, C_n', D'$.*

Proof. The result follows from the definition of hyperresolvent and repeated application of Lemma 5.9.15. □

We aim now to prove the soundness and completeness of hyperresolution for first-order logic.

Lemma 5.9.17. *Let $\mathcal{L}$ be a first-order language and let $C_1, \dots, C_n, D$ be $\mathcal{L}$-clauses. If E is an $\mathcal{L}$-hyperresolvent of $(C_1, \dots, C_n, D)$, then there is an $\mathcal{L}$-resolution proof of E over $\{C_1, \dots, C_n, D\}$.*

Proof. The proof is virtually the same as the proof of Lemma 3.9.27. $\square$

Lemma 5.9.18. *Let $\mathcal{L}$ be a first-order language. If there is an $\mathcal{L}$-hyperresolution proof of an $\mathcal{L}$-clause C over a set of $\mathcal{L}$-clauses $\mathcal{C}$, then there is an $\mathcal{L}$-resolution proof of C over $\mathcal{C}$.*

Proof. Again, the proof is similar to the proof of Lemma 3.9.28. $\square$

Theorem 5.9.19 (Soundness of First-Order Hyperresolution). *If $\mathcal{C}$ is a set of $\mathcal{L}$-clauses and there is an $\mathcal{L}$-hyperresolution proof of $\square$ over $\mathcal{C}$, then $\mathcal{C}$ is unsatisfiable.*

Proof. By Lemma 5.9.18, if there is an $\mathcal{L}$-hyperresolution proof of $\square$ over $\mathcal{C}$, then, there is an $\mathcal{L}$-resolution proof of $\square$ over $\mathcal{C}$, which implies that $\mathcal{C}$ is unsatisfiable, by Theorem 5.8.54. $\square$

We now prepare the ground for proving the completeness of first-order hyperresolution.

Lemma 5.9.20. *Let $\mathcal{L}$ be a first-order language, $C_1, \dots, C_{n-1}$ be positive propositional logic clauses, D be a propositional logic clause, and R be a hyperresolvent of $C_1, \dots, C_n, D$.*

If s is an injective, atomic $\mathcal{L}$-inter-substitution, then $R' = s(R)$ is an $\mathcal{L}$-hyperresolvent of the first-order clauses $C'_1 = s(C_1), \dots, C'_n = s(C_n), D' = s(D)$.

Proof. It is immediate that the clauses $C'_1, \dots, C'_n$ are positive. By the definition of propositional hyperresolvent, there is a sequence of clauses $E_0, \dots, E_n$ such that the following conditions are satisfied:

(1) $E_0 = R$ and $E_n = D$;
(2) for $0 \leq i \leq n-1$, E_i is a resolvent of C_{i+1} and E_{i+1}.

We then have the sequence of $\mathcal{L}$-clauses $E'_0 = s(E_0), \dots, E'_n = s(E_n)$, with $E'_0 = s(E_0) = s(R) = R'$ and $E'_n = s(E_n) = s(D) = D'$. By Lemma 5.8.60, for $0 \leq i \leq n-1$, $E'_i = s(E_i)$ is an $\mathcal{L}$-resolvent of

$C'_{i+1} = s(C_{i+1})$ and $E'_{i+1} = s(E_{i+1})$. Thus, R' is an $\mathcal{L}$-hyperresolvent of $C'_0, \ldots, C'_{n-1}$ and D'. $\square$

An analogue of the First Lifting Lemma (Lemma 5.8.61) for hyperresolution is given next.

Lemma 5.9.21 (First Hyperresolution Lifting Lemma). *Let $\mathcal{L}$ be a first-order language and let s be an injective, atomic $\mathcal{L}$-inter-substitution. If $(C_0, \ldots, C_{n-1})$ is a propositional hyperresolution proof over a set of clauses $\mathcal{C}$, then $(s(C_0), \ldots, s(C_{n-1}))$ is an $\mathcal{L}$-hyperresolution proof over $s(\mathcal{C})$. Further, for $0 \le i \le n-1$, i is an input step of the proof $(C_0, \ldots, C_{n-1})$ if and only if it is an input step of the proof $(s(C_0), \ldots, s(C_{n-1}))$.*

Proof. Let $0 \le i \le n-1$. If $C_i \in \mathcal{C}$, then $s(C_i) \in s(\mathcal{C})$. Otherwise, $C_i \notin \mathcal{C}$ and there is a sequence $j_0, \ldots, j_{p-1} < i$ such C_i is a hyperresolvent of $C_{j_0}, \ldots, C_{j_{p-1}}$. By Theorem 4.8.5, $s(C_i) \notin s(\mathcal{C})$ and by Lemma 5.9.20, $s(C_i)$ is a hyperresolvent of $s(C_{j_0}), \ldots, s(C_{j_{p-1}})$. $\square$

Theorem 5.9.22. *Let $\mathcal{L}$ be a first-order language, $\mathcal{C}$ be a set of $\mathcal{L}$-clauses, $\mathcal{C}_0$ be a fundamental propositional form for $\mathcal{C}$ and let s be a prime, injective inter-substitution with $s(\mathcal{C}_0) = \mathcal{C}$. If C is a propositional clause that has a hyperresolution proof over $\mathcal{C}_0$, then $s(C)$ has an $\mathcal{L}$-hyperresolution proof over $\mathcal{C}$.*

Proof. The argument is analogous with that of Theorem 5.8.62 using Lemma 5.9.21. $\square$

Theorem 5.9.23 (Completeness of Hyperresolution for Ground Clauses and Languages without Equality). *Let $\mathcal{L}$ be a first-order language without equality and let $\mathcal{C}$ be a set of ground $\mathcal{L}$-clauses. If $\mathcal{C}$ has no model, then there is a hyperresolution proof of $\square$ over $\mathcal{C}$.*

Proof. The argument is similar to the proof of Theorem 5.8.63, but using the completeness of propositional hyperresolution (Theorem 3.9.32) in place of Theorem 3.8.32. $\square$

A counterpart for hyperresolution of Lemma 5.8.66 is given next.

Lemma 5.9.24. *Let $\mathcal{L}$ be a first-order language, $C_1, \ldots, C_n, D$ be ground $\mathcal{L}$-clauses such that R is an $\mathcal{L}$-hyperresolvent of $C_1, \ldots, C_n, D$. Suppose that $C'_1, \ldots, C'_n, D'$ are $\mathcal{L}$-clauses whose sets* *of variables are pairwise disjoint and* $s_1, \ldots, s_n, z$ *are* $\mathcal{L}$*-substitutions such that* $s_i(C'_i) = C_i$ *for* $1 \leq i \leq n$ *and* $z(D') = D$*. Then, there is a full most general hyperresolvent* R' *of* $C'_1, \ldots, C'_n, D'$ *and a substitution* s *such that* $s(R') = R$*.*

Proof. Note that since C_i is positive for $1 \leq i \leq n$, the clauses C'_i must also be positive because each of the clauses C_i is obtained from C'_i by the substitution s_i. By definition of hyperresolvent, there is a sequence of $\mathcal{L}$-clauses $E_0, \ldots, E_n$ such that the following conditions are satisfied:

(1) $E_0 = R$ and $E_n = D$;
(2) for $0 \leq i \leq n-1$, E_i is a resolvent of C_{i+1} and E_{i+1}.

It is easy to see by induction on k, where $0 \leq k \leq n$, that E_{n-k} is a ground clause because a resolvent of two ground clauses must be a ground clause.

We demonstrate the existence of $\mathcal{L}$-clauses $E'_0, \ldots, E'_n$ and of substitutions $s'_0, \ldots, s'_n$ such that for $0 \leq k \leq n$, we have

(1) $s'_{n-k}(E'_{n-k}) = E_{n-k}$;
(2) if $k \geq 1$, E'_{n-k} is a resolvent of C'_{n-k+1} and E'_{n-k+1};
(3) $\mathtt{V}(E'_{n-k}) \subseteq \mathtt{V}(C'_{n-k+1}) \cup \cdots \cup \mathtt{V}(C'_n) \cup \mathtt{V}(D')$.

The argument is by induction on k. For $k = 0$, we set $E'_n = D'$ and $s'_n = z$ and the conditions are clearly satisfied. Suppose the statements hold for $k < n$. By inductive hypothesis, $s'_{n-k}(E'_{n-k}) = E_{n-k}$ and $\mathtt{V}(C'_{n-k}) \cap \mathtt{V}(E'_{n-k}) = \emptyset$ because $\mathtt{V}(E'_{n-k}) \subseteq \mathtt{V}(C'_{n-k+1}) \cup \cdots \cup \mathtt{V}(C'_n) \cup \mathtt{V}(D')$ and the sets of variables

$$\mathtt{V}(C'_{n-k}), \ldots, \mathtt{V}(C'_n), \mathtt{V}(D')$$

are pairwise disjoint. We further have $s_{n-k}(C'_{n-k}) = C_{n-k}$ and E_{n-k-1} is a resolvent of C_{n-k} and E_{n-k}. By the Second Disjoint Lifting Lemma (Lemma 5.8.66), there are an $\mathcal{L}$-clause E'_{n-k-1} which is a simple full most general resolvent of C'_{n-k} and E'_{n-k} and a substitution s'_{n-k-1} with

$$s'_{n-k-1}(E'_{n-k-1}) = E_{n-k-1}.$$

Because E'_{n-k-1} is a simple resolvent, we have $\mathtt{V}(E'_{n-k-1}) \subseteq \mathtt{V}(C'_{n-k}) \cup \mathtt{V}(E'_{n-k}) \subseteq \mathtt{V}(C'_{n-k}) \cup \mathtt{V}(C'_{n-k+1}) \cup \cdots \cup \mathtt{V}(C'_n) \cup \mathtt{V}(D')$, which concludes the inductive argument.

Finally, we set $R' = E'_0$ and $s = s'_0$ to obtain the desired full most general hyperresolvent with $s(R') = R$. (Note that the last equality implies that R' is positive by the positivity of R.) $\square$

Lemma 5.9.25 (Second Hyperresolution Lifting Lemma). *Let $\mathcal{L}$ be a first-order language, $\mathcal{C}$ be a set of $\mathcal{L}$-clauses, and let C be an $\mathcal{L}$-clause that has an $\mathcal{L}$-hyperresolution proof over* $\mathrm{GINST}_{\mathcal{L}}(\mathcal{C})$. *Then, there is an $\mathcal{L}$-clause C' that has a full most general hyperresolution proof over $\mathcal{C}$ and a substitution s' such that $s'(C') = C$.*

Proof. We prove by course-of-values induction on n, the length of the $\mathcal{L}$-hyperresolution proof of C that there is a clause C' that has a most general hyperresolution proof over $\mathcal{C}$ and a substitution s' such that $s'(C') = C$. Suppose that $n \geq 1$ and the result holds for all m with $1 \leq m < n$.

If C has an $\mathcal{L}$-hyperresolution proof over $\mathrm{GINST}_{\mathcal{L}}(\mathcal{C})$ of length n, there are two cases to consider. If $C \in \mathrm{GINST}_{\mathcal{L}}(\mathcal{C})$, the result is immediate. Else, there are $C_1, \ldots, C_p, D$ with $\mathcal{L}$-hyperresolution proofs over $\mathrm{GINST}_{\mathcal{L}}(\mathcal{C})$ of length less than n such that C is an $\mathcal{L}$-hyperresolvent of $C_1, \ldots, C_p, D$. By the inductive hypothesis, there exist $\mathcal{L}$-clauses $C'_1, \ldots, C'_p, D'$ with full most general hyperresolution proofs over $\mathcal{C}$ and substitutions $s_1, \ldots, s_p, z$ with $s_i(C'_i) = C_i$, for $1 \leq i \leq p$ and $z(D') = D$. Let $s'_1, \ldots, s'_p, z'$ be renamings of $C'_1, \ldots, C'_p, D'$, respectively, such that the sets of variables of $s'_1(C'_1), \ldots, s'_p(C'_p), z'(D')$ are pairwise disjoint. By Theorem 5.8.22, there are renamings s''_i of $s'_i(C'_i)$ for $1 \leq i \leq p$, and a renaming z'' of $z'(D')$ with $s''_i(s'_i(C'_i)) = C'_i$ and $z''(z'(D')) = D'$. Thus, for $1 \leq i \leq p$, we have $(s_i * s''_i)(s'_i(C'_i)) = C_i$ and $(z * z'')(z'(D')) = D$. By Lemma 5.9.24, there is a full most general hyperresolvent C' of $s'_1(C'_1), \ldots, s'_p(C'_p), z'(D')$ and a substitution s with $s(C') = C$. By Theorem 5.9.16, C' is also a most general hyperresolvent of $C'_1, \ldots, C'_p, D'$ and thus C' has a full most general hyperresolution proof over $\mathcal{C}$.

(See Figure 5.42.) $\square$

Theorem 5.9.26. *Let $\mathcal{L}$ be a first-order language and let $\mathcal{C}$ be a set of $\mathcal{L}$-clauses such that $\square$ has an $\mathcal{L}$-hyperresolution proof over* $\mathrm{GINST}_{\mathcal{L}}(\mathcal{C})$. *Then, $\square$ has a full most general hyperresolution proof over $\mathcal{C}$.*

Proof. This is an immediate consequence of Lemma 5.9.25. $\square$

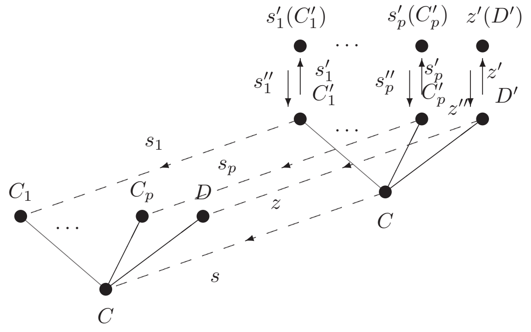


Fig. 5.42. Inductive step in the proof.

Theorem 5.9.27 (Hyperresolution Completeness for Languages without Equality). *Let $\mathcal{L}$ be a first-order language without equality and let $\mathcal{C}$ be a set of $\mathcal{L}$-clauses. If $\mathcal{C}$ has no model, then there is a full most-general hyperresolution proof of $\Box$ over $\mathcal{C}$.*

Proof. The proof follows the same course as the proof of Theorem 5.8.70. $\Box$

Corollary 5.9.28. *Let $\mathcal{L}$ be a first-order language without equality and let $\mathcal{C}$ be a set of $\mathcal{L}$-clauses. The following three statements are equivalent.*

(1) *$\mathcal{C}$ has no model;*
(2) *there is a full most general hyperresolution proof of $\Box$ over $\mathcal{C}$;*
(3) *there is an $\mathcal{L}$-hyperresolution proof of $\Box$ over $\mathcal{C}$.*

Proof. (1) implies (2): This follows from the Completeness Theorem (Theorem 5.9.27).

(2) implies (3): This implication is immediate.

(3) implies (1): This follows from the soundness of hyperresolution. $\Box$

5.10 First-Order Resolution with Equality

5.10.1 *Equality Axioms and Resolution*

The presence of equality in the language poses special problems for the completeness proofs, as we saw with other formal systems.

Recall that $\mathrm{MEq}^{\dagger}_{=,\mathcal{L}}$ is the set of matrices of the equality axioms for the language $\mathcal{L}$, in conjunctive normal form as given in Definition 4.5.64. The next theorem makes use of the set of clauses $\mathcal{C}_{\mathrm{MEq}^{\dagger}_{=,\mathcal{L}}}$ which consists of:

- $\{x_0 = x_0\}$;
- $\{x_0 \neq x_1, x_1 = x_0\}$;
- $\{x_0 \neq x_1, x_1 \neq x_2, x_0 = x_2\}$;
- for every n-ary function symbol $f \in \mathcal{L}$ with $n > 0$,

  $$\{x_0 \neq x_n, \ldots, x_{n-1} \neq x_{2n-1}, f(x_0, \ldots, x_{n-1}) = f(x_n, \ldots, x_{2n-1})\};$$

- for every n-ary relation symbol $P \in \mathcal{L}$ which is not a relational constant and is distinct from $=$,

  $$\{x_0 \neq x_n, \ldots, x_{n-1} \neq x_{2n-1}, (\neg P(x_0, \ldots, x_{n-1})), P(x_n, \ldots, x_{2n-1})\}$$

  and

  $$\{x_0 \neq x_n, \ldots, x_{n-1} \neq x_{2n-1}, (\neg P(x_n, \ldots, x_{2n-1})), P(x_0, \ldots, x_{n-1})\}.$$

To simplify notation, we will denote the set of clauses $\mathcal{C}_{\mathrm{MEq}^{\dagger}_{=,\mathcal{L}}}$ by $\mathcal{C}_{=,\mathcal{L}}$.

Theorem 5.10.1 (Soundness and Completeness of Resolution for Languages with Equality). *Let $\mathcal{L}$ be a first-order language with equality and let $\mathcal{C}$ be a set of $\mathcal{L}$-clauses. If $\mathcal{R}$ is one of* Res, *Res^{mgu}, fRes or $fRes^{mgu}$, then $\mathcal{C}$ has no model if and only if $\Box \in \mathcal{R}^*_{\mathcal{L}}(\mathcal{C} \cup \mathcal{C}_{=,\mathcal{L}})$.*

Proof. We may assume that $\Box \notin \mathcal{C}$, for if $\Box \in \mathcal{C}$, the conclusion follows immediately.

Let R be a binary relation symbol not in $\mathcal{L}$ and let $\mathcal{L}'$ be the language $(\mathcal{L} - \{=\}) \cup \{R\}$.

We will prove the theorem by showing that the following statements are equivalent.

(1) $\mathcal{C}$ has no model;
(2) $\Gamma_{\mathcal{C}}$ has no model;
(3) $\mathsf{s}^{=}_{R}(\Gamma_{\mathcal{C}}) \cup \mathrm{Eq}_{R,\mathcal{L}'}$ has no model;
(4) $\mathsf{s}^{=}_{R}(\Gamma_{\mathcal{C}}) \cup \mathrm{MEq}^{\dagger}_{R,\mathcal{L}'}$ has no model;

(5) $\mathcal{C}_{\mathsf{s}^{=}_{R}(\Gamma_{\mathcal{C}})} \cup \mathcal{C}_{\mathrm{MEq}^{\dagger}_{R,\mathcal{L}'}}$ has no model;
(6) $\mathsf{s}^{=}_{R}(\mathcal{C}) \cup \mathcal{C}_{\mathrm{MEq}^{\dagger}_{R,\mathcal{L}'}}$ has no model;
(7) $\square \in \mathcal{R}^{*}_{\mathcal{L}'}(\mathsf{s}^{=}_{R}(\mathcal{C}) \cup \mathcal{C}_{\mathrm{MEq}^{\dagger}_{R,\mathcal{L}'}})$;
(8) $\square \in \mathcal{R}^{*}_{\mathcal{L}}(\mathcal{C} \cup \mathcal{C}_{=,\mathcal{L}})$.

Statement (1) is equivalent to (2) by Corollary 5.8.13. Next, (2) is equivalent to (3) by Corollary 4.10.37.

The equivalence of (3) and (4) follows from the equivalence of the first and third statements of Theorem 4.5.65.

Statement (4) is equivalent to (5) by Corollary 5.8.13. Next, (5) is equivalent to (6) by Supplement 80.

The equivalence between (6) and (7) follows by Corollary 5.8.71.

Finally, (7) is equivalent to (8). Indeed, observe that

$$\mathsf{s}^{=}_{R}(\mathcal{C}_{=,\mathcal{L}}) = \mathsf{s}^{=}_{R}(\mathcal{C}_{\mathrm{MEq}^{\dagger}_{=,\mathcal{L}}}) = \mathcal{C}_{\mathsf{s}^{=}_{R}(\mathrm{MEq}^{\dagger}_{=,\mathcal{L}})} = \mathcal{C}_{\mathrm{MEq}^{\dagger}_{R,\mathcal{L}'}}. \tag{5.8}$$

The equivalence now follows from Corollary 5.8.51. □

Example 5.10.2. We show that the formula

$$\begin{aligned}\varphi = (\forall x_0)(\forall x_1)(((\forall x_2)(f(x_0, x_2) = x_2)\\ \wedge(\forall x_3)(f(x_3, x_1) = x_3)) \rightarrow (x_0 = x_1)),\end{aligned}$$

introduced in Example 5.8.17 is logically valid by applying Theorem 5.10.1 to the set of clauses $\mathcal{C}$ given in the example mentioned above. The resolution tree shown in Figure 5.43 shows $\square \in \mathrm{Res}^{*}_{\mathcal{L}}(\mathcal{C} \cup \mathcal{C}_{=,\mathcal{L}})$. In this figure, we denote substitutions of the form $\mathsf{s}^{y_0 \cdots y_{k-1}}_{t_0 \cdots t_{k-1}}$ by a list of the form $y_0 \mapsto t_0, \ldots, y_{k-1} \mapsto t_{k-1}$. ▯

We give now a soundness and completeness result for hyperresolution involving clauses over languages with equality.

Theorem 5.10.3 (Soundness and Completeness of Hyperresolution for Languages with Equality). *Let $\mathcal{L}$ be a first-order language with equality and let $\mathcal{C}$ be a set of $\mathcal{L}$-clauses. Then, $\mathcal{C}$ has no model if and only if there is an $\mathcal{L}$-hyperresolution proof of $\square$ over $\mathcal{C} \cup \mathcal{C}_{=,\mathcal{L}}$.*

Proof. Our argument is a modification of the proof of Theorem 5.10.1. Let R be a binary relation symbol not in $\mathcal{L}$ and let $\mathcal{L}'$ be the language $(\mathcal{L} - \{=\}) \cup \{R\}$.

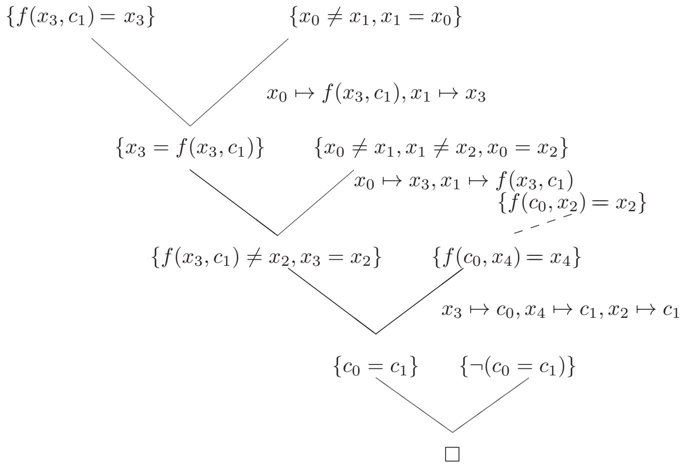

$\{f(x_3,c_1) = x_3\}$
$\{x_0 \neq x_1, x_1 = x_0\}$
$x_0 \mapsto f(x_3,c_1), x_1 \mapsto x_3$
$\{x_3 = f(x_3,c_1)\}$
$\{x_0 \neq x_1, x_1 \neq x_2, x_0 = x_2\}$
$x_0 \mapsto x_3, x_1 \mapsto f(x_3,c_1)$
$\{f(c_0,x_2) = x_2\}$
$\{f(x_3,c_1) \neq x_2, x_3 = x_2\}$
$\{f(c_0,x_4) = x_4\}$
$x_3 \mapsto c_0, x_4 \mapsto c_1, x_2 \mapsto c_1$
$\{c_0 = c_1\}$
$\{\neg(c_0 = c_1)\}$
$\square$

Fig. 5.43. Resolution tree of $\square$ for the set of clauses defined in Example 5.8.17.

We will prove the theorem by showing that the following statements are equivalent.

(1) $\mathcal{C}$ has no model;
(2) $\mathsf{s}_R^=(\mathcal{C}) \cup \mathcal{C}_{\mathrm{MEq}^\dagger_{R,\mathcal{L}'}}$ has no model;
(3) there is an $\mathcal{L}'$-hyperresolution proof of $\square$ over $\mathsf{s}_R^=(\mathcal{C}) \cup \mathcal{C}_{\mathrm{MEq}^\dagger_{R,\mathcal{L}'}}$;
(4) there is an $\mathcal{L}$-hyperresolution proof of $\square$ over $\mathcal{C} \cup \mathcal{C}_{=,\mathcal{L}}$.

The equivalence of (1) and (2) was shown in the proof of Theorem 5.10.1.

The equivalence between (2) and (3) follows from the Completeness of First-Order Hyperresolution for languages without equality as stated in Theorem 5.9.27.

The equivalence of (3) and (4) can be shown in a similar manner to the equivalence between (7) and (8) in the proof of Theorem 5.10.1, using Corollary 5.9.14 instead of Corollary 5.8.51. $\square$

Theorem 5.10.4 (Soundness and Completeness of Full MGU-hyperresolution for Languages with Equality). *Let $\mathcal{L}$ be a first-order language with equality and let $\mathcal{C}$ be a set of $\mathcal{L}$-clauses. Then, $\mathcal{C}$ has no model if and only if there is a full most general hyperresolution proof of $\square$ over $\mathcal{C} \cup \mathcal{C}_{=,\mathcal{L}}$.*

Proof. The argument is similar to the one used in Theorem 5.10.3 except that we use now the completeness of full most general hyper-resolution for languages without equality. $\square$

5.10.2 *Brand's Modification Method and Resolution*

At this point, we have dealt with presence of equality by adding a large number of special new equality clauses. Resolution proofs involving equality are usually lengthy and difficult to put together because of the large search space of axioms. The method we are about to discuss adapts Brand's technique discussed in Section 4.11 to clauses and transforms a set of clauses such that only the reflexivity clause has to be added to the modified set of clauses.

Theorem 5.10.5. *Let $\mathcal{L}$ be a first-order language with equality and let $\mathcal{C}$ be a finite set of $\mathcal{L}$-clauses that does not contain $\square$. If Γ has the form $\mathsf{CF}_{=}((\Gamma_{\mathcal{C}})^{\langle = \rangle})$, then the following three statements are equivalent:*

(1) *$\mathcal{C}$ has no model;*
(2) $\square \in \mathrm{Res}^*_{\mathcal{L}}(\mathcal{C}_\Gamma \cup \{x = x\})$*;*
(3) $\square \in (\mathit{fRes}^{mgu}_{\mathcal{L}})^*(\mathcal{C}_\Gamma \cup \{x = x\})$*.*

Proof. We prove only the equivalence of the first two statements. The equivalence between the first and third statements is proven similarly. Let R be a binary relation symbol not in $\mathcal{L}$ and let $\mathcal{L}'$ be the first-order language $(\mathcal{L} - \{=\}) \cup \{R\}$.

By Theorem 4.11.12, there is an R-flattening, $(\mathsf{s}^{=}_{R}(\Gamma_{\mathcal{C}}))^{\langle R \rangle}$ of $\mathsf{s}^{=}_{R}(\Gamma_{\mathcal{C}})$ such that $\mathsf{s}^{R}_{=}(\mathsf{s}^{=}_{R}(\Gamma_{\mathcal{C}})^{\langle R \rangle}) = \Gamma^{\langle = \rangle}_{\mathcal{C}}$. Let $\Gamma' = \bigcup\{\mathsf{CF}_R(\varphi) \mid \varphi \in (\mathsf{s}^{=}_{R}(\Gamma_{\mathcal{C}}))^{\langle R \rangle}\} \cup \{R(x,x)\}$

We claim that the following statements are equivalent:

(1) the set of $\mathcal{L}$-clauses $\mathcal{C}$ has no model;
(2) the set of $\mathcal{L}$-formulas $\Gamma_{\mathcal{C}}$ has no model;
(3) $\mathsf{s}^{=}_{R}(\Gamma_{\mathcal{C}}) \cup \mathrm{Eq}_{R,\mathcal{L}'}$ has no model;
(4) $\mathsf{s}^{=}_{R}(\Gamma_{\mathcal{C}})$ has no model $\mathcal{A}$ with $R^{\mathcal{A}}$ a congruence;
(5) $(\mathsf{s}^{=}_{R}(\Gamma_{\mathcal{C}}))^{\langle R \rangle}$ has no model $\mathcal{A}'$ where $R^{\mathcal{A}'}$ is an equivalence relation;

(6) $\mathsf{PEEXP}_R((\mathsf{s}_R^{=}(\Gamma_{\mathcal{C}}))^{\langle R\rangle})$, has no model $\mathcal{B}$ where $R^{\mathcal{B}}$ is a reflexive relation;
(7) $\bigcup\{\mathsf{CF}_R(\varphi) \mid \varphi \in (\mathsf{s}_R^{=}(\Gamma_{\mathcal{C}}))^{\langle R\rangle}\}$ has no model $\tilde{\mathcal{A}}$ with $R^{\tilde{\mathcal{A}}}$ reflexive;
(8) Γ' has no model;
(9) the set of $\mathcal{L}'$-clauses $\mathcal{C}_{\Gamma'}$ has no model;
(10) $\Box \in \mathrm{Res}^*_{\mathcal{L}'}(\mathcal{C}_{\Gamma'})$;
(11) $\Box \in \mathrm{Res}^*_{\mathcal{L}}\left(\mathsf{s}_{=}^{R}(\mathcal{C}_{\Gamma'})\right)$;
(12) $\Box \in \mathrm{Res}^*_{\mathcal{L}}\left(\mathcal{C}_{\mathsf{s}_{=}^{R}(\Gamma')}\right)$;
(13) $\Box \in \mathrm{Res}^*_{\mathcal{L}}(\mathcal{C}_{\Gamma} \cup \{x = x\})$.

As we saw in the proof of Theorem 5.10.1, statement (1) is equivalent to (2) by Corollary 5.8.13. Similarly, statement (2) is equivalent to (3) by Corollary 4.10.37. The equivalence of statements (3) and (4) follows from Theorem 4.5.63.

Note
that the set $\mathsf{s}_R^{=}(\Gamma_{\mathcal{C}})$ consists of quantifier-free formulas. Therefore the equivalence of (4) and (5) follows from Theorem 4.11.10. The equivalence of (5) and (6) follows from Theorem 4.11.23 because the formulas of the set $(\mathsf{s}_R^{=}(\Gamma_{\mathcal{C}}))^{\langle R\rangle}$ do not contain $\leftrightarrow$. The equivalence of (6) and (7) follows from the correctness of Algorithm 4.11.26 because the formulas in $(\mathsf{s}_R^{=}(\Gamma_{\mathcal{C}}))^{\langle R\rangle}$ are clausal. The equivalence of (7) and (8) is immediate.

The equivalence of (8) and (9) follows from Corollary 5.8.13 and the fact that $\Box \not\in \Gamma'$. Statements (9) and (10) are equivalent due to Theorem 5.8.70. Corollary 5.8.51 implies the equivalence of Statements (10) and (11). The equivalence of (11) and (12) follows from the immediate observation that $\mathsf{s}_{=}^{R}(\mathcal{C}_{\Gamma'})$ is the same as $\mathcal{C}_{\mathsf{s}_{=}^{R}(\Gamma')}$.

Finally, to prove the equivalence of (12) and (13) it will suffice to show that $\mathsf{s}_{=}^{R}(\Gamma') = \Gamma \cup \{x = x\}$. Indeed, we can write

$$\begin{aligned}
\mathsf{s}_{=}^{R}(\Gamma') &= \mathsf{s}_{=}^{R}(\bigcup\{\mathsf{CF}_R(\varphi) \mid \varphi \in (\mathsf{s}_R^{=}(\Gamma_{\mathcal{C}}))^{\langle R\rangle}\} \cup \{R(x,x)\})\\
&= \bigcup\{\mathsf{s}_{=}^{R}(\mathsf{CF}_R(\varphi)) \mid \varphi \in (\mathsf{s}_R^{=}(\Gamma_{\mathcal{C}}))^{\langle R\rangle}\} \cup \{(x = x)\}\\
&= \bigcup\{\mathsf{CF}_{=}(\mathsf{s}_{=}^{R}(\varphi))) \mid \varphi \in (\mathsf{s}_R^{=}(\Gamma_{\mathcal{C}}))^{\langle R\rangle}\} \cup \{(x = x)\}\\
&\quad \text{(by Theorem 4.11.30)}
\end{aligned}$$

$$= \bigcup \{\mathsf{CF}_=(\varphi) \mid \varphi \in \Gamma_{\mathcal{C}}^{\langle = \rangle}\} \cup \{(x = x)\}$$

(by the choice of the R-flattening)

$$= \Gamma \cup \{(x = x)\}$$

□

Example 5.10.6. We reprove the logical validity of the formula

$$\varphi = (\forall x_0)(\forall x_1)(((\forall x_2)(f(x_0, x_2) = x_2) \\ \wedge(\forall x_3)(f(x_3, x_1) = x_3)) \to (x_0 = x_1)),$$

which we already proved in Example 5.10.2. This time, the argument is based on Theorem 5.10.5. As before, we prove that the corresponding set of clauses obtained in Example 5.8.17,

$$\mathcal{C} = \{\{(f(c_0, x_2) = x_2)\}, \{(f(x_3, c_1) = x_3)\}, \{(\neg(c_0 = c_1))\}\},$$

has no model. The set of formulas $\Gamma_{\mathcal{C}}$ is:

$$\Gamma_{\mathcal{C}} = \{(f(c_0, x_2) = x_2), (f(x_3, c_1) = x_3), (\neg(c_0 = c_1))\}.$$

Assuming that y_0, y_1 are new variables, an $=$-flattening of $\Gamma_{\mathcal{C}}$ is

$$(\Gamma_{\mathcal{C}})^{\langle = \rangle} = \{((f(y_0, x_2) = x_2) \vee (\neg(y_0 = c_0))), \\ ((f(x_3, y_1) = x_3) \vee (\neg(y_1 = c_1))), (\neg(c_0 = c_1))\}.$$

Let w_0, w_1 be the first variables that do not occur in $(\Gamma_{\mathcal{C}})^{\langle = \rangle}$. The set of clausal formulas $\Gamma = \mathsf{CF}_=((\Gamma_{\mathcal{C}})^{\langle = \rangle})$ is:

$$\Gamma = \{((\neg(f(y_0, x_2) = w_0)) \vee (x_2 = w_0) \vee (\neg(y_0 = c_0))), \\ ((f(y_0, x_2) = w_0) \vee (\neg(x_2 = w_0)) \vee (\neg(y_0 = c_0))), \\ ((\neg(f(x_3, y_1) = w_1)) \vee (x_3 = w_1) \vee (\neg(y_1 = c_1))), \\ ((f(x_3, y_1) = w_1) \vee (\neg(x_3 = w_1)) \vee (\neg(y_1 = c_1))), (\neg(c_0 = c_1))\}.$$

Now, $\mathcal{C}_\Gamma \cup \{x = x\}$ is

$$\mathcal{C}_\Gamma \cup \{x = x\} = \{\{(\neg(f(y_0, x_2) = w_0)), (x_2 = w_0), (\neg(y_0 = c_0))\}, \\ \{(f(y_0, x_2) = w_0), (\neg(x_2 = w_0)), (\neg(y_0 = c_0))\}, \\ \{(\neg(f(x_3, y_1) = w_1)), (x_3 = w_1), (\neg(y_1 = c_1))\}, \\ \{(f(x_3, y_1) = w_1), (\neg(x_3 = w_1)), (\neg(y_1 = c_1))), \\ (\neg(c_0 = c_1)\}, \{(x = x)\}\}.$$

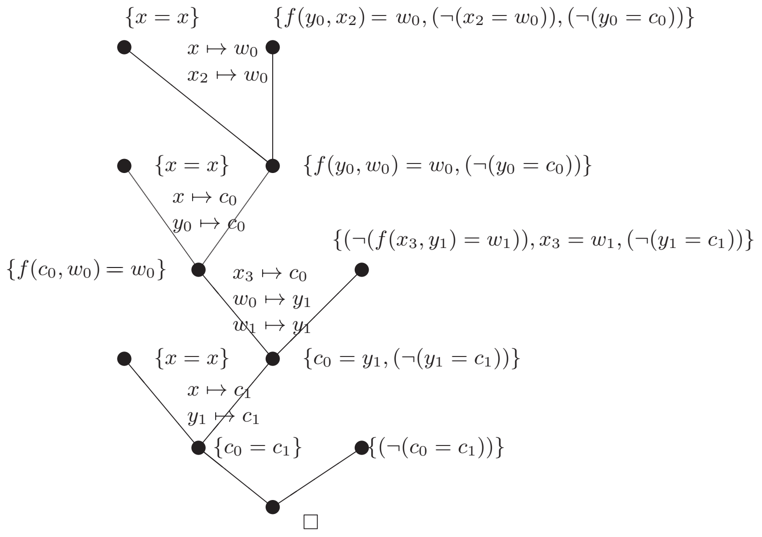


Fig. 5.44. Resolution tree for $\mathcal{C}_\Gamma \cup \{x = x\}$.

The resolution tree shown in Figure 5.44 shows that $\Box \in \text{Res}^*(\mathcal{C}_\Gamma \cup \{x = x\})$. Note that the resolution tree only makes use of the reflexivity of equality while the proof in Example 5.10.2 uses both transitivity and symmetry.

$\Box$

5.10.3 *Paramodulation*

An alternative approach to handling equality in resolution adds a new rule, called *paramodulation*, producing shorter and simpler proofs.

Theorem 5.10.7. *Let $\mathcal{L}$ be a first-order language, ℓ be an $\mathcal{L}$-literal and $\zeta = (t, i)$ be an occurrence of an $\mathcal{L}$-term t in ℓ.*

If ℓ is a positive literal $R(t_0, \ldots, t_{n-1})$, then ζ is part of one of the occurrences of the form $(t_i, 2 + i + \sum_{j=0}^{i-1} |t_j|)$.

If ℓ is a negative literal $(\neg R(t_0, \ldots, t_{n-1}))$, then ζ is part of one of the occurrences of the form $(t_i, 4 + i + \sum_{j=0}^{i-1} |t_j|)$.

Proof. When ℓ is a positive literal, by regarding ℓ as a term in the extended signature $S_{\mathcal{L}}^{ext}$ and taking account that t must be a proper subterm of ℓ, the result follows by Theorem 1.5.27.

Let now $\ell = (\neg R(t_0, \ldots, t_{n-1}))$ be a negative literal. Observe that for $\zeta = (t, i)$, we have $i \geq 4$, because t cannot start with (, $\neg$, or R. Furthermore, t cannot start with a comma, so ζ starts inside some term t_j, at the same position where a subterm t' of t_j begins (by Supplement 50 of Chapter 1). Since no term can be a proper prefix of another term, it follows that $t = t'$, which establishes the result. $\square$

Corollary 5.10.8. *Let $\mathcal{L}$ be a first-order language, ℓ be an $\mathcal{L}$-literal, $\zeta = (t, i)$ be an occurrence of an $\mathcal{L}$-term t in ℓ and let w be an $\mathcal{L}$-term. Then,* $\texttt{replace}\,(\ell, \zeta, w)$ *is an $\mathcal{L}$-literal of the same nature (positive or negative) as ℓ.*

Proof. This statement follows from Theorems 5.10.7 and 1.5.30. $\square$

Definition 5.10.9. Let $\mathcal{L}$ be a first-order language. An $\mathcal{L}$-clause P is an *$\mathcal{L}$-paramodulant* of two $\mathcal{L}$-clauses C_0, C_1, if there are:

- a standardization (s_0, s_1) of (C_0, C_1);
- a nonempty subset L of $\bar{s}_0(C_0)$ consisting of positive equality literals;
- a nonempty subset K of $\bar{s}_1(C_1)$;
- an $\mathcal{L}$-unifier s' of L such that $\overline{s'}(L) = \{u = v\}$;
- an $\mathcal{L}$-unifier s'' of K such that $\overline{s''}(K) = \{\ell\}$;
- an occurrence $\zeta = (t, i) \in \texttt{OCC}_t(\ell)$ of a term t in ℓ,

and one of the following two situations occur:

(1) there exists an $\mathcal{L}$-unifier s of $\{u, t\}$ such that

$$P = \overline{s}\left(\overline{s'}(\overline{s_0}(C_0) - L) \cup \overline{s''}(\overline{s_1}(C_1) - K) \cup \{\texttt{replace}\,(\ell, \zeta, v)\}\right).$$

(2) there exists an $\mathcal{L}$-unifier s of $\{v, t\}$ such that

$$P = \overline{s}\left(\overline{s'}(\overline{s_0}(C_0) - L) \cup \overline{s''}(\overline{s_1}(C_1) - K) \cup \{\texttt{replace}\,(\ell, \zeta, u)\}\right).$$

If one of the two previous cases holds, we say that P was obtained from C_0 and C_1 by *paramodulating* the equality $(u = v)$ into the term occurrence (t, i). The first case of the above definition is represented in Figure 5.45.

If s', s'' are most general unifiers of L and K, respectively, and s is a most general unifier of $\{u, t\}$ in the first case or of $\{v, t\}$ in

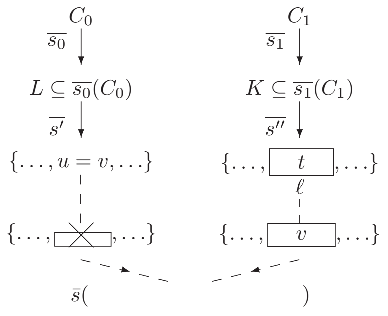


Fig. 5.45. The construction of a paramodulant.

the second, then we say that P is a *most general paramodulant of* C_0 *and* C_1.

If s_0, s_1 are both the identity map on their respective domains, then we say that P is a *simple* $\mathcal{L}$*-paramodulant* of C_0 and C_1.

If we have both $\overline{s'}(\overline{s_0}(C_0) - L) \cap \overline{s'}(L) = \emptyset$ and $\overline{s''}(\overline{s_1}(C_1) - K) \cap \overline{s''}(K) = \emptyset$, then we say that P is a *full* $\mathcal{L}$*-paramodulant* of C_0 and C_1. □

In either of the cases mentioned in Definition 5.10.9, P is a clause due to Corollary 5.10.8. The construction shown in Figure 5.45 represents a full paramodulant.

Example 5.10.10. Let $C_0 = \{(f(x_3, c_1) = x_3)\}$ and $C_1 = \{(f(c_0, x_2) = x_2)\}$. The clause $\{c_0 = c_1\}$ is a paramodulant of C_0 and C_1 obtained by choosing s_0 and s_1 to be the identity substitutions, the set of equality literals L as $\{f(x_3, c_1) = x_3\}$, the set of literals K as $\{(f(c_0, x_2) = x_2)\}$, the unifiers s', s'' as the identity substitution, and the occurrence ζ as $(f(c_0, x_2), 2)$[2]. The unifier s of $f(x_3, c_1)$ and $f(c_0, x_2)$ maps the variables x_3 and x_2 into c_0 and c_1, respectively, so the paramodulant equals

$$s(\texttt{replace}\,(f(c_0, x_2) = x_2, (f(c_0, x_2), 2), x_3)),$$

which equals $\{c_0 = c_1\}$. □

[2]Observe that the formula $(f(c_0, x_2) = x_2)$ is actually $= (f(c_0, x_2), x_2)$ which means that f is the third symbol of this formula.

Note that if P is an $\mathcal{L}$-paramodulant of two ground clauses C_0 and C_1, and the first of the two alternatives in the definition of paramodulant holds, then there is an equality $(u = v)$ in C_0, a literal ℓ in C_1, and an occurrence (u, i) of u in ℓ such that

$$P = (C_0 - \{(u = v)\}) \cup (C_1 - \{\ell\}) \cup \{\texttt{replace}\,(\ell, (u, i), v)\}$$

and similarly if the second alternative holds. Also, P is a full, most general paramodulant of C_0 and C_1.

Theorem 5.10.11. *Let $\mathcal{L}$ be a first-order language and let C_0, C_1 be two $\mathcal{L}$-clauses. If P is an $\mathcal{L}$-paramodulant of C_0 and C_1 and $\mathcal{A}$ is an $\mathcal{L}$-structure, then $\mathcal{A} \models C_0$ and $\mathcal{A} \models C_1$ implies $\mathcal{A} \models P$.*

Proof. Let $s_0, s_1, L, K, s', s'', u, v, \ell, \zeta, s$ be as in the definition of paramodulant, so we have either

(1) there exists a unifier s of $\{u, t\}$ such that

$$\begin{aligned} P &= \overline{s}\left(\overline{s'}(\overline{s_0}(C_0) - L) \cup \overline{s''}(\overline{s_1}(C_1) - K) \cup \{\texttt{replace}\,(\ell, \zeta, v)\}\right) \\ &= \overline{s}\left(\overline{s'}(\overline{s_0}(C_0) - L)\right) \cup \overline{s}\left(\overline{s''}(\overline{s_1}(C_1) - K)\right) \\ &\quad \cup \overline{s}\left(\{\texttt{replace}\,(\ell, \zeta, v)\}\right) \end{aligned}$$

or

(2) there exists a unifier s of $\{v, t\}$ such that

$$P = \overline{s}\left(\overline{s'}(\overline{s_0}(C_0) - L) \cup \overline{s''}(\overline{s_1}(C_1) - K) \cup \{\texttt{replace}\,(\ell, \zeta, u)\}\right).$$

We will treat the first case only. Since $\mathcal{A} \models C_0$ and $\mathcal{A} \models C_1$, by Corollary 5.8.24, $\mathcal{A} \models \overline{s}_0(C_0)$ and $\mathcal{A} \models \overline{s}_1(C_1)$. Then, by Theorem 5.8.23, we have $\mathcal{A} \models \overline{s}(\overline{s'}(\overline{s_0}(C_0)))$ and $\mathcal{A} \models \overline{s}(\overline{s''}(\overline{s_1}(C_1)))$.

Suppose $\sigma \in \mathrm{ASSIGN}_{\mathcal{A}}$. To prove that $(\mathcal{A}, \sigma) \models P$, we may assume that $(\mathcal{A}, \sigma) \not\models \overline{s}\left(\overline{s'}(\overline{s_0}(C_0) - L)\right)$ and $(\mathcal{A}, \sigma) \not\models \overline{s}\left(\overline{s''}(\overline{s_1}(C_1) - K)\right)$ and prove that $(\mathcal{A}, \sigma) \models \overline{s}\left(\{\texttt{replace}\,(\ell, \zeta, v)\}\right)$. Since $(\mathcal{A}, \sigma) \models \overline{s}(\overline{s'}(\overline{s_0}(C_0)))$ and $(\mathcal{A}, \sigma) \models \overline{s}(\overline{s''}(\overline{s_1}(C_1)))$, it follows that

$$(\mathcal{A}, \sigma) \models \overline{s}(u = v) \tag{5.9}$$

$$(\mathcal{A}, \sigma) \models \overline{s}(\ell). \tag{5.10}$$

Assume now that $\ell = R(t_0, \ldots, t_{n-1})$. We have

$$\begin{aligned}
&\bar{s}(\texttt{replace}\,(\ell, \zeta, v)) \\
&\quad = \texttt{replace}\,(\bar{s}(\ell), \zeta', \bar{s}(v)) \\
&\qquad \text{(by Theorem 1.2.13, where } \zeta' \text{ is an occurrence of } \bar{s}(t)) \\
&\quad = \texttt{replace}\,(R(\bar{s}(t_0), \ldots, \bar{s}(t_{n-1})), \zeta', \bar{s}(v)) \\
&\quad = R(\bar{s}(t_0), \ldots, \texttt{replace}\,(\bar{s}(t_j), \zeta'', \bar{s}(v)), \ldots, \bar{s}(t_{n-1})) \\
&\qquad \text{(by Theorem 5.10.7, where } \zeta'' \text{ is an occurrence of } \bar{s}(t) \text{ in } t_j).
\end{aligned}$$

Thus, $(\mathcal{A}, \sigma) \models \bar{s}(\texttt{replace}\,(\ell, \zeta, v))$ if and only if

$$(\sigma^{\mathcal{A}}(\bar{s}(t_0)), \ldots, \sigma^{\mathcal{A}}(\texttt{replace}\,(\bar{s}(t_j), \zeta'', \bar{s}(v))), \ldots, \sigma^{\mathcal{A}}(\bar{s}(t_{n-1}))) \in R^{\mathcal{A}}$$

By (5.10), we have

$$(\sigma^{\mathcal{A}}(\bar{s}(t_0)), \ldots, \sigma^{\mathcal{A}}(\bar{s}(t_{n-1}))) \in R^{\mathcal{A}}. \tag{5.11}$$

By (5.9), we have $\sigma^{\mathcal{A}}(\bar{s}(u)) = \sigma^{\mathcal{A}}(\bar{s}(v))$. Taking into account that $\bar{s}(u) = \bar{s}(t)$, we have $\sigma^{\mathcal{A}}(\bar{s}(t)) = \sigma^{\mathcal{A}}(\bar{s}(v))$. Thus, by Supplement 50 of Chapter 4, we have

$$\sigma^{\mathcal{A}}(\texttt{replace}\,(\bar{s}(t_j), \zeta'', \bar{s}(v))) = \sigma^{\mathcal{A}}(\bar{s}(t_j)).$$

This equality, together with (5.11) establishes the result.

The case when ℓ is a negative literal can be treated similarly. □

Corollary 5.10.12. *Let $\mathcal{L}$ be a first-order language, $\mathcal{C}$ be a set of $\mathcal{L}$-clauses and let P be an $\mathcal{L}$-paramodulant of two clauses in $\mathcal{C}$. Then, for every $\mathcal{L}$-structure $\mathcal{A}$, $\mathcal{A} \models \mathcal{C}$ if and only if $\mathcal{A} \models \mathcal{C} \cup \{P\}$.*

Proof. The proof is similar to that of Corollary 5.8.41. □

Definition 5.10.13. Let $\mathcal{C}$ be a set of $\mathcal{L}$-clauses, where $\mathcal{L}$ is a first-order language with equality. Then we define

$$\begin{aligned}&\mathrm{Respar}_{\mathcal{L}}(\mathcal{C})\\ &\quad = \mathcal{C} \cup \{R \mid R \text{ is an } \mathcal{L}\text{-resolvent of two clauses in } \mathcal{C}\}\\ &\qquad \cup \{P \mid P \text{ is an } \mathcal{L}\text{-paramodulant of two clauses in } \mathcal{C}\},\end{aligned}$$

and

$$\begin{aligned}&\mathrm{Respar}_{\mathcal{L}}^{mgu}(\mathcal{C})\\ &\ = \mathcal{C} \cup \{R \mid R \text{ is a most general } \mathcal{L}\text{-resolvent of two clauses in } \mathcal{C}\}\\ &\quad \cup \{P \mid P \text{ is a most general } \mathcal{L}\text{-paramodulant of two clauses in } \mathcal{C}\}.\end{aligned}$$

The mappings $\mathrm{fRespar}_{\mathcal{L}}$ and $\mathrm{fRespar}_{\mathcal{L}}^{mgu}$ are defined similarly using full resolvents and paramodulants. □

The results of applying the mappings $\mathrm{Respar}_{\mathcal{L}}^{mgu}$ and $\mathrm{fRespar}_{\mathcal{L}}^{mgu}$ to sets of $\mathcal{L}$-clauses do not depend on $\mathcal{L}$, so we may use the notations Respar^{mgu} and $\mathrm{fRespar}^{mgu}$, when convenient.

Theorem 5.10.14. *Let $\mathcal{L}$ be a first-order language with equality, $\mathcal{C}$ and $\mathcal{D}$ be a sets of $\mathcal{L}$-clauses and let $\mathcal{A}$ be an $\mathcal{L}$-structure. Then, if $\mathcal{R}$ is Respar, Respar^{mgu}, fRespar, or fRespar^{mgu} we have:*

(1) $\mathcal{C} \subseteq \mathcal{R}_{\mathcal{L}}(\mathcal{C})$;
(2) *if* $\mathcal{C} \subseteq \mathcal{D}$, *then* $\mathcal{R}_{\mathcal{L}}(\mathcal{C}) \subseteq \mathcal{R}_{\mathcal{L}}(\mathcal{D})$; *and*
(3) $\mathcal{A} \models \mathcal{R}_{\mathcal{L}}(\mathcal{C})$ *if and only if* $\mathcal{A} \models \mathcal{C}$.

Proof. The proof is similar to that of Theorem 5.8.43. Observe that to prove the third part, in addition to Theorem 5.8.40, we have to use Theorem 5.10.11. □

Let $\mathcal{L}$ be a first-order language with equality and let $\mathcal{S}_{\mathcal{L}}$ be the set of all $\mathcal{L}$-clauses. Then, $\mathrm{Respar}_{\mathcal{L}} : \mathcal{P}(\mathcal{S}_{\mathcal{L}}) \longrightarrow \mathcal{P}(\mathcal{S}_{\mathcal{L}})$, so we can consider its iterations $\mathrm{Respar}_{\mathcal{L}}^{n}$ for $n \in \mathbf{N}$, following the standard definition:

$$\begin{aligned}\mathrm{Respar}_{\mathcal{L}}^{0}(\mathcal{C}) &= \mathcal{C},\\ \mathrm{Respar}_{\mathcal{L}}^{n+1}(\mathcal{C}) &= \mathrm{Respar}_{\mathcal{L}}(\mathrm{Respar}_{\mathcal{L}}^{n}(\mathcal{C}))\end{aligned}$$

for every set of clauses $\mathcal{C} \in \mathcal{P}(\mathcal{S}_{\mathcal{L}})$.

Note that, as a consequence of Part (1) of Theorem 5.10.14, we have the increasing chain of sets

$$\mathcal{C} = \mathrm{Respar}^0_{\mathcal{L}}(\mathcal{C}) \subseteq \mathrm{Respar}^1_{\mathcal{L}}(\mathcal{C}) \subseteq \cdots \subseteq \mathrm{Respar}^n_{\mathcal{L}}(\mathcal{C}) \subseteq \cdots . \quad (5.12)$$

Definition 5.10.15. Let $\mathcal{C}$ be a set of $\mathcal{L}$-clauses, where $\mathcal{L}$ is a first-order language with equality. The set $\mathrm{Respar}^*_{\mathcal{L}}(\mathcal{C})$, the *resolution-paramodulation closure* of $\mathcal{C}$, is defined by

$$\mathrm{Respar}^*_{\mathcal{L}}(\mathcal{C}) = \bigcup_{n \geq 0} \mathrm{Respar}^n_{\mathcal{L}}(\mathcal{C}).$$

$\square$

The mappings

$$(\mathrm{Respar}^{mgu}_{\mathcal{L}})^n, (\mathrm{Respar}^{mgu}_{\mathcal{L}})^*,$$
$$\mathrm{fRespar}^n_{\mathcal{L}}, \mathrm{fRespar}^*_{\mathcal{L}},$$
$$(\mathrm{fRespar}^{mgu}_{\mathcal{L}})^n, (\mathrm{fRespar}^{mgu}_{\mathcal{L}})^*$$

are defined similarly.

Theorem 5.10.16. *Let $\mathcal{C}$ and $\mathcal{D}$ be sets of $\mathcal{L}$-clauses and let $\mathcal{A}$ be an $\mathcal{L}$-structure, where $\mathcal{L}$ is a first-order language with equality. If $\mathcal{R}$ is Respar, Respar^{mgu}, fRespar, or fRespar^{mgu} we have:*

(1) *$\mathcal{C} \subseteq \mathcal{R}^n_{\mathcal{L}}(\mathcal{C})$ for all $n \in \mathbf{N}$ and $\mathcal{C} \subseteq \mathcal{R}^*_{\mathcal{L}}(\mathcal{C})$;*
(2) *for all $n \in \mathbf{N}$, $\mathcal{A} \models \mathcal{R}^n_{\mathcal{L}}(\mathcal{C})$ if and only if $\mathcal{A} \models \mathcal{C}$;*
(3) *$\mathcal{A} \models \mathcal{R}^*_{\mathcal{L}}(\mathcal{C})$ if and only if $\mathcal{A} \models \mathcal{C}$;*
(4) *$\mathcal{C}$ has a model if and only if $\mathcal{R}^*_{\mathcal{L}}(\mathcal{C})$ has a model; and*
(5) *if $\mathcal{C} \subseteq \mathcal{D}$, then $\mathcal{R}^n_{\mathcal{L}}(\mathcal{C}) \subseteq \mathcal{R}^n_{\mathcal{L}}(\mathcal{D})$ for all $n \in \mathbf{N}$ and $\mathcal{R}^*_{\mathcal{L}}(\mathcal{C}) \subseteq \mathcal{R}^*_{\mathcal{L}}(\mathcal{D})$.*

Proof. The proof is similar to that of Theorem 5.8.45. $\square$

Definition 5.10.17. Let $\mathcal{C}$ be a set of $\mathcal{L}$-clauses, where $\mathcal{L}$ is a first-order language with equality. An *$\mathcal{L}$-resolution-paramodulation proof* over $\mathcal{C}$ is a finite sequence $(C_0, C_1, \ldots, C_{n-1})$ of $\mathcal{L}$-clauses such that $n \geq 1$ and for each i, $0 \leq i \leq n-1$ one of the following cases occurs:

(1) $C_i \in \mathcal{C}$;
(2) $C_i \notin \mathcal{C}$ and there are $j, k < i$ such C_i is an $\mathcal{L}$-resolvent of C_j and C_k;

(3) neither of the above cases holds and there are $j, k < i$ such C_i is an $\mathcal{L}$-paramodulant of C_j and C_k.

In the first case, i is an *input step* of the proof; in the second, i is a *resolution step*, and in the third case, i is a *paramodulation step*.

An *$\mathcal{L}$-resolution-paramodulation proof of a clause C* over $\mathcal{C}$, where $\mathcal{L}$ is a first-order language with equality is an $\mathcal{L}$-resolution-paramodulation proof over $\mathcal{C}$ whose last entry is C.

If at each resolution step i, C_i is a full $\mathcal{L}$-resolvent (most general resolvent) and at each paramodulation step i, C_i is a full $\mathcal{L}$-paramodulant (most general paramodulant), then the sequence $(C_0, C_1, \ldots, C_{n-1})$ is a *full $\mathcal{L}$-resolution-paramodulation proof* (*most general $\mathcal{L}$-resolution-paramodulation proof*). ∎

Note that if $\mathcal{C}$ is a set of $\mathcal{L}$-clauses, then the notion of most general $\mathcal{L}$-resolution-paramodulation proof is independent of $\mathcal{L}$ and therefore $\mathcal{L}$ may be omitted.

Theorem 5.10.18. *Let $\mathcal{C}$ be a set of $\mathcal{L}$-clauses, where $\mathcal{L}$ is a first-order language with equality. Then, $Respar^*_{\mathcal{L}}(\mathcal{C})$ is the set of clauses which have $\mathcal{L}$-resolution-paramodulation proofs over $\mathcal{C}$, $(Respar^{mgu}_{\mathcal{L}})^*$ is the set of clauses which have most general resolution-paramodulation proofs over $\mathcal{C}$, $fRespar^*_{\mathcal{L}}$ is the set of clauses which have full $\mathcal{L}$-resolution-paramodulation proofs over $\mathcal{C}$, and $(fRespar^{mgu}_{\mathcal{L}})^*$ is the set of clauses which have full most general resolution-paramodulation proofs over $\mathcal{C}$.*

Proof. The argument is similar to the proof of Theorem 5.8.47. □

The notion of $\mathcal{L}$-resolution-paramodulation proof can be viewed in the framework of the formal system introduced next.

Definition 5.10.19. Let $\mathcal{L}$ be a first-order language with equality. The formal system $\mathcal{FRESPAR}^{\mathcal{L}}$ is

$$\mathcal{FRESPAR}^{\mathcal{L}} = (\mathcal{P}_{fin}(\mathrm{LIT}_{\mathcal{L}}), \emptyset, \{\mathsf{R}, \mathsf{P}\}),$$

where the set of objects $\mathcal{P}_{fin}(\mathrm{LIT}_{\mathcal{L}})$ consists of finite sets of literals of $\mathcal{L}$ ($\mathcal{L}$-clauses), the binary rule R consists of all pairs $((C, D), E)$ where E is an $\mathcal{L}$-resolvent of C and D, and the binary rule P consists of all pairs $((C, D), E)$ where E is an $\mathcal{L}$-paramodulant of C and D.

If rules R and P are replaced by the rules R^f consisting of all pairs $((C, D), E)$ where E is a full $\mathcal{L}$-resolvent of C and D and P^f consisting of all pairs $((C, D), E)$ where E is a full $\mathcal{L}$-paramodulant of C and D , we obtain the formal system $\mathcal{FFRESPAR}^{\mathcal{L}}$.

If the rule R is replaced by the rule R_{mgu} that consists of all pairs $((C, D), E)$, where E is a most general resolvent of C and D and the rule P is replaced by the rule P_{mgu} that consists of similar pairs, where E is a most general paramodulant of C and D, the corresponding formal system $\mathcal{FRESPAR}^{\mathcal{L}}_{mgu}$ will be referred to as the *most general $\mathcal{L}$-resolution-paramodulation formal system.*

The formal system $\mathcal{FFRESPAR}^{\mathcal{L}}_{mgu}$ is obtained by replacing the rules $\mathsf{R}_{mgu}, \mathsf{P}_{mgu}$ with R^f_{mgu} and P^f_{mgu} using full most general resolvents and full most general paramodulants, respectively. $\square$

Note that if $\mathcal{C}$ is a set of $\mathcal{L}$-clauses, then an $\mathcal{L}$-resolution-paramodulation proof (most general resolution-paramodulation proof) over $\mathcal{C}$ is the same thing as a proof in the formal system $\mathcal{FRESPAR}^{\mathcal{L}}_{\mathcal{C}}$ $((\mathcal{FRESPAR}^{\mathcal{L}}_{mgu})_{\mathcal{C}})$. Thus, Theorem 5.10.18 amounts to saying that $\text{Thm}(\mathcal{FRESPAR}^{\mathcal{L}}_{\mathcal{C}}) = \text{Respar}^*_{\mathcal{L}}(\mathcal{C})$ and $\text{Thm}((\mathcal{FRESPAR}^{\mathcal{L}}_{mgu})_{\mathcal{C}}) = (\text{Respar}^{mgu}_{\mathcal{L}})^*(\mathcal{C})$.

Similar comments work for $\mathcal{FFRESPAR}^{\mathcal{L}}$ and $\mathcal{FFRESPAR}^{\mathcal{L}}_{mgu}$.

The introduction of a formal system allows us to make use of the idea of proof tree.

Definition 5.10.20. Let $\mathcal{C}$ be a set of $\mathcal{L}$-clauses, where $\mathcal{L}$ is a first-order language with equality. An *$\mathcal{L}$-resolution-paramodulation tree over C* is an $\mathcal{FRESPAR}^{\mathcal{L}}_{\mathcal{C}}$-proof tree. $\square$

In other words, an $\mathcal{L}$-resolution-paramodulation tree over $\mathcal{C}$ is a lot such that its leaves are labeled with clauses from $\mathcal{C}$ and each interior node is labeled with a clause that is an $\mathcal{L}$-resolvent or an $\mathcal{L}$-paramodulant of the clauses which are labels of its two immediate descendents. Theorem 1.8.23 and our previous discussion allow us to conclude that a clause C is in $\text{Respar}^*_{\mathcal{L}}(\mathcal{C})$ if and only if there is an $\mathcal{L}$-resolution-paramodulation tree over $\mathcal{C}$ such that C is the label of its root.

Theorem 5.10.21 (Soundness of Resolution-Paramodulation). *Let $\mathcal{C}$ be a set of $\mathcal{L}$-clauses, where $\mathcal{L}$ is a first-order language with equality. If $\square \in \text{Respar}^*_{\mathcal{L}}(\mathcal{C})$, then $\mathcal{C}$ has no model.*

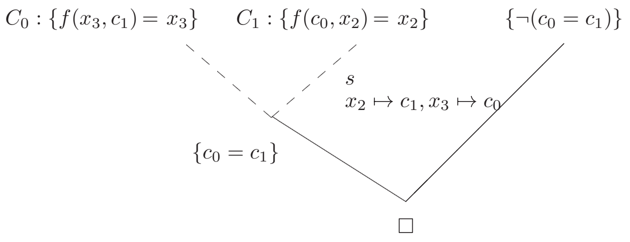

$C_0 : \{f(x_3, c_1) = x_3\}$
$C_1 : \{f(c_0, x_2) = x_2\}$
$\{\neg(c_0 = c_1)\}$
s
$x_2 \mapsto c_1, x_3 \mapsto c_0$
$\{c_0 = c_1\}$
$\square$

Fig. 5.46. Resolution-paramodulation tree.

Proof. By the third part of Theorem 5.10.16, any model of $\mathcal{C}$ would be a model of $\mathrm{Respar}^*_{\mathcal{L}}(\mathcal{C})$. Since $\square \in \mathrm{Respar}^*_{\mathcal{L}}(\mathcal{C})$, it follows that there is no model for $\mathrm{Respar}^*_{\mathcal{L}}(\mathcal{C})$, and, therefore, there is no model for $\mathcal{C}$. $\square$

In terms of the formal system $\mathcal{FRESPAR}^{\mathcal{L}}_{\mathcal{C}}$ Theorem 5.10.21 amounts to saying that if $\square$ is a theorem of $\mathcal{FRESPAR}^{\mathcal{L}}_{\mathcal{C}}$, then $\mathcal{C}$ has no model.

Example 5.10.22. Consider again the set of clauses

$$\mathcal{C} = \{\{(f(c_0, x_2) = x_2)\}, \{(f(x_3, c_1) = x_3)\}, \{(\neg(c_0 = c_1))\}\}$$

introduced in Example 5.8.17. Figure 5.46 contains a resolution-paramodulation tree showing that $\square \in \mathrm{Respar}^*_{\mathcal{L}}(\mathcal{C})$ and hence, by Theorem 5.10.21, $\mathcal{C}$ has no model. Note that the paramodulation step is represented by dashed lines. This tree contains a paramodulation step involving the clauses

$$C_0 : \{f(x_3, c_1) = x_3\}, C_1 : \{f(c_0, x_2) = x_2\}$$

which results in the clause $\{c_0 = c_1\}$. This step was justified in Example 5.10.10. The resolvent of the clauses $\{c_0 = c_1\}$ and $\{(\neg c_0 = c_1)\}$ is $\square$.

Note that the current proof of $\square$ using paramodulation involves only two steps, whereas the previous proof given in Example 5.10.2 involves four steps. $\square$

5.10.4 *Semantic Trees for Languages with Equality*

We begin by discussing an ordering of ground terms and ground atomic formulas in order to define semantic trees for first-order languages with equality. This will be useful in presenting Peterson's[3] approach to proving the completeness of paramodulation.

Definition 5.10.23. Let $\mathcal{L}$ be a first-order language. A *ground $\mathcal{L}$-word* is a sequence that is either a ground $\mathcal{L}$-term or a ground $\mathcal{L}$-atomic formula. The set of ground $\mathcal{L}$-words is denoted by $\mathrm{GWORD}_{\mathcal{L}}$.

Finally, the set $\mathrm{GWORDNE}_{\mathcal{L}}$ is $\mathrm{GWORD}_{\mathcal{L}}$ if $\mathcal{L}$ does not contain the equality symbol and is $\mathrm{GWORD}_{\mathcal{L}} - \mathrm{GEQ}_{\mathcal{L}}$ if $\mathcal{L}$ contains the equality symbol. $\square$

Let $\mathcal{L}$ be a first-order language. For each symbol $s \in \mathcal{L} - \{=\}$, let $s^\sharp$ be a unique code number in $\mathbf{P}$. In other words the function which maps s into $s^\sharp$ is injective.

Define recursively the function $G_{\mathcal{L}} : \mathrm{GTERM}_{\mathcal{L}} \longrightarrow \mathbf{P}$ by

- $G_{\mathcal{L}}(c) = 2^{(c^\sharp)}$ if c is a constant symbol of $\mathcal{L}$;
- $G_{\mathcal{L}}(f(t_0, \ldots, t_{n-1})) = 2^{(f^\sharp)} \prod_{i=0}^{n-1} p_{i+1}^{G_{\mathcal{L}}(t_i)}$, if f is an n-ary function symbol in $\mathcal{L}$ with $n > 0$, where $p_0 < p_1 < \cdots$ is the sequence of prime numbers.

We extend the function $G_{\mathcal{L}}$ to $\mathrm{GAFORMNE}_{\mathcal{L}}$ when $\mathcal{L}$ is not an algebraic language by:

- $G_{\mathcal{L}}(R) = 2^{(R^\sharp)}$ if R is a 0-ary relation symbol of $\mathcal{L}$;
- $G_{\mathcal{L}}(R(t_0, \ldots, t_{n-1})) = 2^{(R^\sharp)} \prod_{i=0}^{n-1} p_{i+1}^{G_{\mathcal{L}}(t_i)}$, where R is an n-ary relation symbol of $\mathcal{L} - \{=\}$ and $n > 0$.

The next statement presents essential properties of the function $G_{\mathcal{L}}$.

[3]Gerald E. Peterson was born on August 9, 1938 and died on October 12, 1999 in St. Louis. He was a graduate of the University of Utah where he received a PhD in 1965. Peterson taught mathematics and computer science at the University of Utah, Brigham Young University, University of Missouri at St. Louis, and University of Illinois-Edwardsville and worked as a research engineer at Boeing.

Theorem 5.10.24. *Let $\mathcal{L}$ be a first-order language. The function $G_{\mathcal{L}} : \mathrm{GWORDNE}_{\mathcal{L}} \longrightarrow \mathbf{P}$ satisfies the following properties:*

G1. *If $w \in \mathrm{GWORDNE}_{\mathcal{L}}$, $w = s(t_0, \ldots, t_{n-1})$ where $t_0, \ldots, t_{n-1}$ belong to $\mathrm{GTERM}_{\mathcal{L}}$, then $G_{\mathcal{L}}(t_i) < G_{\mathcal{L}}(w)$, for $0 \leq i \leq n-1$.*

G2. *The function $G_{\mathcal{L}}$ is one-to-one on $\mathrm{GWORDNE}_{\mathcal{L}}$.*

G3. *If $w \in \mathrm{GWORDNE}_{\mathcal{L}}$ and $t \in \mathrm{GTERM}_{\mathcal{L}}$ occurs in w, then $G_{\mathcal{L}}(t) \leq G_{\mathcal{L}}(w)$. (Note that by G2, if $t \neq w$, then $G_{\mathcal{L}}(t) < G_{\mathcal{L}}(w)$.)*

G4. *If $w \in \mathrm{GWORDNE}_{\mathcal{L}}$, $t, u \in \mathrm{GTERM}_{\mathcal{L}}$, $(t, i) \in \mathtt{OCC}_t(w)$, and we have $G_{\mathcal{L}}(u) < G_{\mathcal{L}}(t)$, then*

$$G_{\mathcal{L}}(w[(t,i) \to u]) < G_{\mathcal{L}}(w).$$

Proof.

- *Proof of G1*: For $0 \leq i \leq n-1$, we have $G_{\mathcal{L}}(t_i) < p_{i+1}^{G_{\mathcal{L}}(t_i)} < G_{\mathcal{L}}(w)$.
- *Proof of G2*: We show by course-of-values induction on n that if $w, w' \in \mathrm{GWORDNE}_{\mathcal{L}}$ and $G_{\mathcal{L}}(w) = G_{\mathcal{L}}(w') = n$, then $w = w'$.
  Suppose that the result is true for all $n' < n$ and $G_{\mathcal{L}}(w) = G_{\mathcal{L}}(w') = n$. If m is the largest number such that 2^m divides n, then m is the code number of the first symbol of w and w', so both w and w' begin with the same symbol s. If s is 0-ary, then $w = s = w'$. If s is n-ary with $n > 0$, then for some $t_0, \ldots, t_{n-1}, t'_0, \ldots, t'_{n-1} \in \mathrm{GTERM}_{\mathcal{L}}$, we have $w = s(t_0, \ldots, t_{n-1})$ and $w' = s(t'_0, \ldots, t'_{n-1})$. For $0 \leq i \leq n-1$, if m_i is the largest number such that $p_{i+1}^{m_i}$ divides n, then $G_{\mathcal{L}}(t_i) = m_i = G_{\mathcal{L}}(t'_i)$, and $m_i < n$, so by the inductive hypothesis, $t_i = t'_i$. Thus, $w = w'$.
- *Proof of G3*: We first show the result for $w \in \mathrm{GTERM}_{\mathcal{L}}$ by induction. If $t = w$, the result is immediate. If w is a constant symbol, then $t = w$.
  Suppose that $w = f(t_0, \ldots, t_{n-1})$, where $n > 0$. If t is different from w, then t occurs in one of the terms t_i and by the inductive hypothesis, $G_{\mathcal{L}}(t) \leq G_{\mathcal{L}}(t_i) < G_{\mathcal{L}}(w)$ due to (G1).
  Now suppose that $w = R(t_0, \ldots, t_{n-1})$, where R is an n-ary relation symbol other than $=$. Then t occurs in one of the terms t_i, so by the above, $G_{\mathcal{L}}(t) \leq G_{\mathcal{L}}(t_i) < G_{\mathcal{L}}(w)$ by (G1).
- *Proof of G4*: We begin by showing the result for $w \in \mathrm{GTERM}_{\mathcal{L}}$ by induction on w.
  If $t = w$, then $w[(t,i) \to u] = u$ and the result is immediate. If w is a constant symbol, then $t = w$. Suppose that $w = f(t_0, \ldots, t_{n-1})$,

where $n > 0$ and the result is true for each t_i. If $t \neq w$, then t occurs in some t_j, say (t, i') is the corresponding occurrence in t_j, and by the inductive hypothesis, $G_{\mathcal{L}}(t_j[(t,i') \to u]) < G_{\mathcal{L}}(t_j)$, which allows us to write:

$$\begin{aligned} G_{\mathcal{L}}(w[(t,i) \to u]) &= G_{\mathcal{L}}(f(t_0, \ldots, t_j[(t,i') \to u], \ldots, t_{n-1}) \\ &= 2^{f^{\#}} p_1^{G_{\mathcal{L}}(t_0)} \cdots p_{j+1}^{G_{\mathcal{L}}(t_j[(t,i')\to u])} \cdots p_n^{G_{\mathcal{L}}(t_{n-1})} \\ &< 2^{f^{\#}} p_1^{G_{\mathcal{L}}(t_0)} \cdots p_{j+1}^{G_{\mathcal{L}}(t_j)} \cdots p_n^{G_{\mathcal{L}}(t_{n-1})} = G_{\mathcal{L}}(w). \end{aligned}$$

Now suppose that $w = R(t_0, \ldots, t_{n-1})$, where R is an n-ary relation symbol of $\mathcal{L} - \{=\}$. Then, t occurs in one of the terms t_j, say (t_j, i') is the corresponding occurrence in t_j. As just shown $G_{\mathcal{L}}(t_j[(t,i') \to u]) < G_{\mathcal{L}}(t_j)$. Then, $G_{\mathcal{L}}(w[(t,i) \to u]) < G_{\mathcal{L}}(w)$ as above (replacing f by R).

$\square$

Definition 5.10.25. Let $\mathcal{L}$ be a first-order language. The relation $\prec_{\mathcal{L}}$ on GWORDNE$_{\mathcal{L}}$ is given by $w_0 \prec_{\mathcal{L}} w_1$ if $G_{\mathcal{L}}(w_0) < G_{\mathcal{L}}(w_1)$. $\square$

Theorem 5.10.26. *The relation $\prec_{\mathcal{L}}$ has the following properties:*

W1. The relation $\prec_{\mathcal{L}}$ is a linear order on GWORDNE$_{\mathcal{L}}$.

W2. Each $w \in$ GWORDNE$_{\mathcal{L}}$ *has a finite number of predecessors under $\prec_{\mathcal{L}}$.*

W3. If $w \in$ GWORDNE$_{\mathcal{L}}$ *and $t \in$* GTERM$_{\mathcal{L}}$ *occurs in w, then $t \preceq_{\mathcal{L}} w$. Thus, if $t \neq w$, then $t \prec_{\mathcal{L}} w$.*

W4. If $w \in$ GWORDNE$_{\mathcal{L}}$, *$t, u \in$* GTERM$_{\mathcal{L}}$, *$(t,i) \in \mathtt{OCC}_t(w)$, and $u \prec_{\mathcal{L}} t$, then $w[(t,i) \to u] \prec_{\mathcal{L}} w$.*

Proof.

- *Proof of W1*: Since $G_{\mathcal{L}}(w) \not< G_{\mathcal{L}}(w)$, we have $w \not\prec_{\mathcal{L}} w$, for $w \in$ GWORDNE$_{\mathcal{L}}$.
  If $w_0 \prec_{\mathcal{L}} w_1 \prec_{\mathcal{L}} w_2$, then $G_{\mathcal{L}}(w_0) < G_{\mathcal{L}}(w_1) < G_{\mathcal{L}}(w_2)$, so $G_{\mathcal{L}}(w_0) < G_{\mathcal{L}}(w_2)$ and hence $w_0 \prec_{\mathcal{L}} w_2$.
  If $w \neq w'$ taking into account the injectivity of $G_{\mathcal{L}}$ we have either $G_{\mathcal{L}}(w) < G_{\mathcal{L}}(w')$ or $G_{\mathcal{L}}(w') < G_{\mathcal{L}}(w)$, so either $w \prec_{\mathcal{L}} w'$ or $w' \prec_{\mathcal{L}} w$.

- *Proof of W2*: For $w \in \mathrm{GWORDNE}_{\mathcal{L}}$ we have

$$\begin{aligned}&\{w' \in \mathrm{GWORDNE}_{\mathcal{L}} \mid w' \prec_{\mathcal{L}} w\}\\ &= \{w' \in \mathrm{GWORDNE}_{\mathcal{L}} \mid G_{\mathcal{L}}(w') < G_{\mathcal{L}}(w)\}.\end{aligned}$$

  Since $G_{\mathcal{L}}$ is injective, the last set is finite.
- *Proofs of W3 and W4*: These statements follow from G3 and G4 of Theorem 5.10.24, respectively.

□

When $\mathcal{L}$ contains the equality symbol, we extend the linear order $\prec_{\mathcal{L}}$ introduced above to all of $\mathrm{GWORD}_{\mathcal{L}}$ as follows.

For $t = u \in \mathrm{GEQ}_{\mathcal{L}}$, define

$$\max(t = u) = \begin{cases} t & \text{if } u \preceq_{\mathcal{L}} t \\ u & \text{if } t \prec_{\mathcal{L}} u \end{cases}$$

$$\min(t = u) = \begin{cases} t & \text{if } t \preceq_{\mathcal{L}} u \\ u & \text{if } u \prec_{\mathcal{L}} t. \end{cases}$$

Note that $\max(t = t) = \min(t = t) = t$.

For $w \in \mathrm{GWORDNE}_{\mathcal{L}}$, $t, u \in \mathrm{GTERM}_{\mathcal{L}}$, define

$$\begin{aligned}(t = u) \prec w \ &\text{ if } \max(t = u) \preceq w\\ w \prec (t = u) \ &\text{ if } w \prec \max(t = u).\end{aligned}$$

For $t_0, t_1, u_0, u_1 \in \mathrm{GTERM}_{\mathcal{L}}$ define $(t_0 = u_0) \prec (t_1 = u_1)$ if one of the following conditions is satisfied:

(1) $\max(t_0 = u_0) \prec_{\mathcal{L}} \max(t_1 = u_1)$,
(2) $\max(t_0 = u_0) = \max(t_1 = u_1)$ and $\min(t_0 = u_0) \prec_{\mathcal{L}} \min(t_1 = u_1)$,
(3) $\max(t_0 = u_0) = \max(t_1 = u_1)$, $\min(t_0 = u_0) = \min(t_1 = u_1)$, and $t_1 \prec_{\mathcal{L}} t_0$.

The following theorem contains the basic properties of the partial order $\prec_{\mathcal{L}}$ on $\mathrm{GWORD}_{\mathcal{L}}$.

To prove the theorem, we need the following preliminary result.

Lemma 5.10.27. *Let $\mathcal{L}$ be a first-order language with equality and let $\ell : \mathrm{GWORD}_{\mathcal{L}} \longrightarrow \mathrm{GWORDNE}_{\mathcal{L}}$ be the function defined by*

$$\ell(w) = w \text{ if } w \in \mathrm{GWORDNE}_{\mathcal{L}},$$

$$\ell(t = u) = \max(t = u) \text{ if } t, u \in \mathrm{GTERM}_{\mathcal{L}}.$$

The following properties hold for $w, w' \in \mathrm{GWORD}_{\mathcal{L}}$:

L1. If $w \prec_{\mathcal{L}} w'$, then $\ell(w) \preceq_{\mathcal{L}} \ell(w')$.
L2. If $w \prec_{\mathcal{L}} w'$ and $\ell(w) = \ell(w')$, then $w \in \mathrm{GEQ}_{\mathcal{L}}$.
L3. If $\ell(w) \prec_{\mathcal{L}} \ell(w')$, then $w \prec_{\mathcal{L}} w'$.

Proof. These properties follow immediately from the definition of the relation $\prec_{\mathcal{L}}$. □

Theorem 5.10.28. *Let $\mathcal{L}$ be a first-order language with equality. The following statements hold:*

O1. $\prec_{\mathcal{L}}$ is a linear order on $\mathrm{GWORD}_{\mathcal{L}}$.
O2. Each element of $\mathrm{GWORD}_{\mathcal{L}}$ has only finitely many predecessors under $\prec_{\mathcal{L}}$.
O3. If $t \in \mathrm{GTERM}_{\mathcal{L}}$, $w \in \mathrm{GWORDNE}_{\mathcal{L}}$ and t occurs in w, then $t \preceq_{\mathcal{L}} w$.
O4. If $w \in \mathrm{GWORD}_{\mathcal{L}}$, $t, u \in \mathrm{GTERM}_{\mathcal{L}}$, $(t, i) \in \mathtt{OCC}_t(w)$ and $u \prec_{\mathcal{L}} t$, then $w[(t, i) \to u] \prec_{\mathcal{L}} w$.
O5. If $t, u \in \mathrm{GTERM}_{\mathcal{L}}$ and $u \prec_{\mathcal{L}} t$, then $(t = u)$ and $(u = t)$ are adjacent in $\prec_{\mathcal{L}}$ and $(t = u) \prec_{\mathcal{L}} (u = t)$.
O6. If $w \in \mathrm{GWORD}_{\mathcal{L}}$, $t, u \in \mathrm{GTERM}_{\mathcal{L}}$, $u \preceq_{\mathcal{L}} t$, t occurs in w, and w has neither of the forms $w = (t = t')$ nor $w = (t' = t)$ with $t' \in \mathrm{GTERM}_{\mathcal{L}}$ and $t' \preceq_{\mathcal{L}} u$, then $(u = t) \prec_{\mathcal{L}} w$.

Proof.

- *Proof of O1*: If $w \in \mathrm{GWORDNE}_{\mathcal{L}}$, then by W1, $w \not\prec_{\mathcal{L}} w$. If $w = (t = u) \in \mathrm{GEQ}_{\mathcal{L}}$ and $w \prec_{\mathcal{L}} w$, then by the definition of $\prec_{\mathcal{L}}$ one of the following would hold:

$$\max(t = u) \prec_{\mathcal{L}} \max(t = u),$$
$$\min(t = u) \prec_{\mathcal{L}} \min(t = u), \text{ or}$$
$$t \prec_{\mathcal{L}} t.$$

  None of the above cases is possible by the irreflexivity of $\prec_{\mathcal{L}}$ on $\mathrm{GWORDNE}_{\mathcal{L}}$.

Suppose that $w_0 \prec_{\mathcal{L}} w_1$ and $w_1 \prec_{\mathcal{L}} w_2$. Then, by L1, $\ell(w_0) \preceq_{\mathcal{L}} \ell(w_1)$ and $\ell(w_1) \preceq_{\mathcal{L}} \ell(w_2)$. If either of these inequalities is strict, then using the transitivity of $\prec_{\mathcal{L}}$ on $\mathrm{GWORDNE}_{\mathcal{L}}$, we have $\ell(w_0) \prec_{\mathcal{L}} \ell(w_2)$ so $w_0 \prec_{\mathcal{L}} w_2$ by L3.
If $\ell(w_0) = \ell(w_1) = \ell(w_2)$, then by L2 w_0 and w_1 are equalities, say $w_0 = (t_0 = u_0)$ and $w_1 = (t_1 = u_1)$. If $w_2 \in \mathrm{GWORDNE}_{\mathcal{L}}$, then $\max(w_0) = \max(w_1) = w_2$, so $w_0 \prec_{\mathcal{L}} w_2$. Suppose that $w_2 = (t_2 = u_2)$. Then,

$$\max(t_0 = u_0) = \max(t_1 = u_1) = \max(t_2 = u_2).$$

Let t be this common value. We have

$$\min(t_0 = u_0) \preceq_{\mathcal{L}} \min(t_1 = u_1) \preceq_{\mathcal{L}} \min(t_2 = u_2).$$

If either inequality is strict, then using transitivity of $\prec_{\mathcal{L}}$ on the set $\mathrm{GWORDNE}_{\mathcal{L}}$, we have $\min(t_0 = u_0) \prec_{\mathcal{L}} \min(t_2 = u_2)$, so $w_0 \prec_{\mathcal{L}} w_2$.
If

$$\min(t_0 = u_0) = \min(t_1 = u_1) = \min(t_2 = u_2) = u,$$

say, then $t_2 \prec_{\mathcal{L}} t_1 \prec_{\mathcal{L}} t_0$. Since each of w_0, w_1, w_2 belongs to $\{(t = u), (u = t)\}$, at least two of the values t_0, t_1, t_2 are the same. If $t_0 = t_1$ or $t_1 = t_2$, then this contradicts the irreflexivity of $\prec_{\mathcal{L}}$. If $t_0 = t_2$ then by transitivity of $\prec_{\mathcal{L}}$ on $\mathrm{GWORDNE}_{\mathcal{L}}$, we obtain $t_0 = t_2 \prec_{\mathcal{L}} t_0$ which violates irreflexivity.
Next we show that $\prec_{\mathcal{L}}$ is a total order on $\mathrm{GWORD}_{\mathcal{L}}$. If $w, w' \in \mathrm{GWORDNE}_{\mathcal{L}}$ and $w' \neq w$, then either $w \prec_{\mathcal{L}} w'$ or $w' \prec_{\mathcal{L}} w$ by W1. If $w \in \mathrm{GWORDNE}_{\mathcal{L}}$ and $w' = (t' = u')$, then by W1 either $w \prec_{\mathcal{L}} \max(t' = u')$ or $\max(t' = u') \preceq_{\mathcal{L}} w$. Thus, either $w \prec_{\mathcal{L}} w'$ or $w' \prec_{\mathcal{L}} w$.
If $w = (t = u)$ and $w' = (t' = u')$, $w \neq w'$, then by W1, $\max(w)$ and $\max(w')$ are comparable and $\min(w)$ and $\min(w')$ are comparable. If $\max(w) \neq \max(w')$, or $\max(w) = \max(w')$ and $\min(w) \neq \min(w')$, then w, w' are comparable.
If $\max(w) = \max(w')$ and $\min(w) = \min(w')$ then, we have $t \neq t'$ because $w \neq w'$. By W1, either $t \prec_{\mathcal{L}} t'$ or $t' \prec_{\mathcal{L}} t$, so w and w' are comparable.

- *Proof of O2*: If $w_0 \prec_{\mathcal{L}} w_1$, then $\ell(w_0) \preceq_{\mathcal{L}} \ell(w_1)$ by L1. By W2, $\ell(w_1)$ has only finitely many predecessors in $\mathrm{GWORDNE}_{\mathcal{L}}$. Therefore, if we show that for each $w \in \mathrm{GWORDNE}_{\mathcal{L}}$ there are only finitely many $w' \in \mathrm{GWORD}_{\mathcal{L}}$ with $\ell(w') = w$, O2 will follow. If $w \in \mathrm{GAFORMNE}_{\mathcal{L}}$, then $\ell^{-1}(w) = \{w\}$. If $w \in \mathrm{GTERM}_{\mathcal{L}}$, then

$$\begin{aligned}\ell^{-1}(w) &= \{w\} \cup \{e \in \mathrm{GEQ}_{\mathcal{L}} \mid \max(e) = w\} \\ &= \{w\} \cup \{(w = u) \mid u \preceq_{\mathcal{L}} w\} \cup \{(u = w) \mid u \preceq_{\mathcal{L}} w\}.\end{aligned}$$

  By W2, $\ell^{-1}(w)$ is finite.
- *Proof of O3*: O3 coincides with W3.
- *Proof of O4*: If $w \in \mathrm{GWORDNE}_{\mathcal{L}}$, then O4 follows from W4. Therefore, suppose that $w = (t_0 = t_1)$. Without loss of generality, suppose that (t, i) is part of t_0 and correspondingly (t, i') is the occurrence in t_0. By W4, $t_0[(t, i') \to u] \prec_{\mathcal{L}} t_0$. We need to consider two cases.
  If $t_1 \prec_{\mathcal{L}} t_0$, then

$$\max(t_0[(t, i') \to u] = t_1) \prec_{\mathcal{L}} t_0 = \max(t_0 = t_1).$$

  Therefore, $(t_0 = t_1)[(t, i) \to u] \prec_{\mathcal{L}} (t_0 = t_1)$.
  If $t_0 \preceq_{\mathcal{L}} t_1$, then

$$\max(t_0[(t, i') \to u] = t_1) = t_1 = \max(t_0 = t_1)$$

  and

$$\min(t_0[(t, i') \to u] = t_1) = t_0[(t, i') \to u] \prec_{\mathcal{L}} t_0 = \min(t_0 = t_1).$$

  Thus, $(t_0 = t_1)[(t, i) \to u] \prec_{\mathcal{L}} (t_0 = t_1)$.
- *Proof of O5*: Suppose that $t, u \in \mathrm{GTERM}_{\mathcal{L}}$ and $u \prec_{\mathcal{L}} t$. We have $\max(t = u) = \max(u = t)$, $\min(t = u) = \min(u = t)$ and $u \prec_{\mathcal{L}} t$, so $(t = u) \prec_{\mathcal{L}} (u = t)$. Suppose that there is $w \in \mathrm{GWORD}_{\mathcal{L}}$ such that $(t = u) \prec_{\mathcal{L}} w \prec_{\mathcal{L}} (u = t)$. We consider two cases.
  In the first case, suppose that $w \in \mathrm{GWORDNE}_{\mathcal{L}}$. Then,

$$\max(t = u) \preceq_{\mathcal{L}} w \prec_{\mathcal{L}} \max(u = t) = \max(t = u).$$

  This leads to $\max(t = u) \prec_{\mathcal{L}} \max(t = u)$ which is a contradiction.

In the second case, $w \in \mathrm{GEQ}_{\mathcal{L}}$, say $w = (t_0 = u_0)$. Then

$$\max(t = u) \preceq_{\mathcal{L}} \max(t_0 = u_0) \preceq_{\mathcal{L}} \max(u = t) = \max(t = u),$$

so

$$\max(t = u) = \max(t_0 = u_0) = \max(u = t),$$

and therefore

$$\min(t = u) \preceq_{\mathcal{L}} \min(t_0 = u_0) \preceq_{\mathcal{L}} \min(u = t) = \min(t = u).$$

It follows that

$$\min(t = u) = \min(t_0 = u_0) = \min(u = t),$$

so $u \prec_{\mathcal{L}} t_0 \prec_{\mathcal{L}} t$. However, t_0 is either $\min(t = u) = u$ or $\max(t = u) = t$, which contradicts the irreflexivity of $\prec_{\mathcal{L}}$.

- *Proof of O6:* Suppose that $w \in \mathrm{GWORD}_{\mathcal{L}}$, $t, u \in \mathrm{GTERM}_{\mathcal{L}}$, $u \preceq_{\mathcal{L}} t$, t occurs in w and w has neither of the forms $w = (t = t')$ nor $w = (t' = t)$ with $t' \in \mathrm{GTERM}_{\mathcal{L}}$ and $t' \preceq_{\mathcal{L}} u$. We need to consider two cases.

  Case 1: Suppose that $w \in \mathrm{GWORDNE}_{\mathcal{L}}$. Then, by O3, $\max(u = t) = t \preceq_{\mathcal{L}} w$, so $(u = t) \prec_{\mathcal{L}} w$.

  Case 2: Suppose that $w \in \mathrm{GEQ}_{\mathcal{L}}$, say $w = (t_0 = t_1)$. There are several subcases to consider:

  Case 2.1: Assume that t is a proper subterm of either t_0 or t_1. Without loss of generality, we may assume that t is a proper subterm of t_0. Then, by O3, we have

$$\max(u = t) = t \prec_{\mathcal{L}} t_0 \preceq_{\mathcal{L}} \max(t_0 = t_1),$$

  so $(u = t) \prec_{\mathcal{L}} (t_0 = t_1)$.

  Case 2.2: Assume that either $t = t_0$ or $t = t_1$. Without loss of generality, let $t = t_0$. Since w has neither of the forms $w = (t = t')$ nor $w = (t' = t)$ with $t' \in \mathrm{GTERM}_{\mathcal{L}}$ and $t' \preceq_{\mathcal{L}} u$ and by O1, we have $u \prec_{\mathcal{L}} t_1$. This case has two further subcases:

  Case 2.2.1: This occurs when $t \prec_{\mathcal{L}} t_1$. Then,

$$\max(u = t) = t \prec_{\mathcal{L}} t_1 = \max(t = t_1),$$

  so $(u = t) \prec_{\mathcal{L}} (t = t_1) = w$.

Case 2.2.2: This sub-case occurs when $t_1 \preceq_{\mathcal{L}} t$. We have $\max(u = t) = t = \max(t = t_1)$ and $\min(u = t) = u \prec_{\mathcal{L}} t_1 = \min(t = t_1)$, so $(u = t) \prec_{\mathcal{L}} (t = t_1) = w$. □

E-interpretations are special $\mathcal{LH}$-interpretations suitable for first-order languages with equality. (Recall that $\mathcal{LH}$-interpretation was introduced immediately before Exercise 86.)

Definition 5.10.29. Let $\mathcal{L}$ be a first-order language with equality that contains at least one constant symbol and let $\Gamma \subseteq \mathrm{GAFORM}_{\mathcal{L}}$. An *E-interpretation* on Γ is an $\mathcal{LH}$-interpretation I on Γ that satisfies the following conditions:

E1. $I(t = t) = \mathbf{T}$ if $(t = t) \in \Gamma$;
E2. $I(t = u) = I(u = t)$ if $(t = u), (u = t) \in \Gamma$;
E3. If $\alpha \in \mathrm{GAFORM}_{\mathcal{L}}$, $(t, i) \in \mathtt{OCC}_t(\alpha)$ for some term $t \in \mathrm{GTERM}_{\mathcal{L}}$, $(t = u), \alpha, \alpha[(t, i) \to u] \in \Gamma$ and $I(t = u) = \mathbf{T}$, then $I(\alpha) = I(\alpha[(t, i) \to u])$.

An *$\mathcal{L}$-E-interpretation* is an E-interpretation on $\mathrm{GAFORM}_{\mathcal{L}}$. ▯

Theorem 5.10.30. *Let $\mathcal{L}$ be a first-order language with equality that contains at least one constant symbol. If an $\mathcal{L}$-interpretation satisfies E1 and E3, then it satisfies E2. In other words, an $\mathcal{L}$-interpretation that satisfies E1 and E3 is an $\mathcal{L}$-E-interpretation.*

Proof. Let $t, u \in \mathrm{GTERM}_{\mathcal{L}}$ and suppose that $I(t = u) = \mathbf{T}$. By E1, $I(t = t) = \mathbf{T}$, so by E3, $\mathbf{T} = I(t = t) = I(u = t)$. Similarly, if $I(u = t) = \mathbf{T}$, then $I(t = u) = \mathbf{T}$, so $I(t = u) = I(u = t)$. □

Theorem 5.10.31. *Let $\mathcal{L}$ be a first-order language with equality that contains at least one constant symbol and let I be an E-interpretation on Γ, where $\Gamma \subseteq \mathrm{GAFORM}_{\mathcal{L}}$. If $(t = u), (u = z), (t = z) \in \Gamma$ and $I(t = u) = I(u = z) = \mathbf{T}$, then $I(t = z) = \mathbf{T}$, where $t, u, z \in \mathrm{GTERM}_{\mathcal{L}}$.*

Proof.
Since $I(u = z) = \mathbf{T}$, by E3, we have $\mathbf{T} = I(t = u) = I(t = z)$. □

Definition 5.10.32. Let $\mathcal{L}$ be a first-order language with equality that contains at least one constant symbol and let Γ be a set of closed quantifier-free $\mathcal{L}$-formulas. An *$\mathcal{L}$-E-model* for Γ is an $\mathcal{L}$-E-interpretation I such that $I(\varphi) = \mathbf{T}$ for all $\varphi \in \Gamma$ (that is, $I \models \Gamma$). $\square$

Theorem 5.10.33. *Let $\mathcal{L}$ be a first-order language with equality that contains at least one constant symbol and let Γ be a set of closed quantifier-free $\mathcal{L}$-formulas. Then, Γ has a model if and only if Γ has an $\mathcal{L}$-E-model.*

Proof. Suppose that the $\mathcal{L}$-structure $\mathcal{A}$ is a model of Γ. We define an $\mathcal{L}$-interpretation I by

$$I(\alpha) = \begin{cases} \mathbf{T} & \text{if } \mathcal{A} \models \alpha, \\ \mathbf{F} & \text{if } \mathcal{A} \not\models \alpha \end{cases}$$

for $\alpha \in \mathrm{GAFORM}_{\mathcal{L}}$. By Theorem 4.5.16, for all closed quantifier-free $\mathcal{L}$-formulas φ, $I(\varphi) = \mathbf{T}$ if and only if $\mathcal{A} \models \varphi$. Since $\mathcal{A}$ is a model of Γ, $I(\varphi) = \mathbf{T}$ for all $\varphi \in \Gamma$. We need to show that I is an $\mathcal{L}$-E-interpretation.

To show that E1 holds, let $t \in \mathrm{GTERM}_{\mathcal{L}}$. We have $(t^{\mathcal{A}}, t^{\mathcal{A}}) \in =^{\mathcal{A}}$, so $\mathcal{A} \models (t = t)$ and therefore $I(t = t) = \mathbf{T}$.

To prove E3, let $\alpha \in \mathrm{GAFORM}_{\mathcal{L}}$, $t, u \in \mathrm{GTERM}_{\mathcal{L}}$, $(t, i) \in \mathtt{OCC}_t(\alpha)$ and $I(t = u) = \mathbf{T}$, that is $\mathcal{A} \models (t = u)$. Thus, $t^{\mathcal{A}} = u^{\mathcal{A}}$ and $t \equiv_{\mathcal{A}} u$. By Theorem 4.6.17, $\alpha \equiv_{\mathcal{A}} \alpha[(t, i) \to u]$, so $\mathcal{A} \models \alpha$ if and only if $\mathcal{A} \models \alpha[(t, i) \to u]$. This implies that $I(\alpha) = \mathbf{T}$ if and only if $I(\alpha[(t, i) \to u]) = \mathbf{T}$, so $I(\alpha) = I(\alpha[(t, i) \to u])$.

As noted previously, E2 follows from E1 and E3.

Conversely, suppose that I is an $\mathcal{L}$-E-model of Γ. Define the relation ρ on $\mathrm{GTERM}_{\mathcal{L}}$ by $t\rho u$ if and only if $I(t = u) = \mathbf{T}$. We begin by verifying that ρ is an equivalence on $\mathrm{GTERM}_{\mathcal{L}}$.

ρ is reflexive because for all $t \in \mathrm{GTERM}_{\mathcal{L}}$, $I(t = t) = \mathbf{T}$ by E1.

ρ is symmetric because for all $t, u \in \mathrm{GTERM}_{\mathcal{L}}$, if $t\rho u$, then $I(t = u) = \mathbf{T}$. By E2, $I(u = t) = I(t = u) = \mathbf{T}$, so we have $u\rho t$.

ρ is transitive. Indeed, suppose that $t, u, z \in \mathrm{GTERM}_{\mathcal{L}}$, $t\rho u$, and $u\rho z$, that is $I(t = u) = \mathbf{T} = I(u = z)$. By Theorem 5.10.31, $I(t = z) = \mathbf{T}$, so $t\rho z$.

Let $f \in \mathcal{L}$ be an n-ary function symbol with $n > 0$ and let $t_i \rho u_i$ for $0 \leq i \leq n - 1$, where $t_0, \ldots, t_{n-1}, u_0, \ldots, u_{n-1}$ are terms in $\mathrm{GTERM}_{\mathcal{L}}$. For $0 \leq i \leq n - 1$, $I(t_i = u_i) = \mathbf{T}$ because $t_i \rho u_i$. By E1, $I(f(t_0, \ldots, t_{n-1}) = f(t_0, \ldots, t_{n-1})) = \mathbf{T}$. By E3,

$I(f(t_0,\dots,t_{n-1}) = f(u_0,\dots,t_{n-1})) = \mathbf{T}$. Another application of E3 yields $I(f(t_0,t_1,\dots,t_{n-1}) = f(u_0,u_1,\dots,t_{n-1})) = \mathbf{T}$, etc. Finally, by another application of E3, we have $I(f(t_0,\dots,t_{n-1}) = f(u_0,\dots,u_{n-1})) = \mathbf{T}$. Thus, $f(t_0,\dots,t_{n-1})\rho f(u_0,\dots,u_{n-1})$.

Now let $R \in \mathcal{L} - \{=\}$ be an n-ary relation symbol with positive arity, $t_0,\dots,t_{n-1},u_0,\dots,u_{n-1} \in \mathrm{GTERM}_{\mathcal{L}}$ and $t_i\rho u_i$ for $0 \leq i \leq n-1$. Then by repeated application of E3, $I(R(t_0,\dots,t_{n-1})) = I(R(u_0,\dots,u_{n-1}))$ and therefore $R(t_0,\dots,t_{n-1})\rho R(u_0,\dots,u_{n-1})$.

Let $S = \{\alpha \in \mathrm{GAFORMNE}_{\mathcal{L}} \mid I(\alpha) = \mathbf{T}\}$ and define $\mathcal{B} = \mathsf{STR}_{\mathcal{L}}(S)$ as in Definition 4.10.5. If $\alpha \in \mathrm{GAFORMNE}_{\mathcal{L}}$, then, by Lemma 4.10.11,

$$\mathcal{B} \models \alpha \text{ if and only if } \alpha \in S \text{ if and only if } I(\alpha) = \mathbf{T}.$$

We saw that ρ is an equivalence on $|\mathcal{B}| = \mathrm{GTERM}_{\mathcal{L}}$. We will show that ρ is in fact a congruence on $\mathcal{B}$, that is, it is compatible with the functions and the relations defined on $\mathcal{B}$.

If f is an n-ary function symbol of $\mathcal{L}$ with $n > 0$ and $t_0\rho u_0, \dots t_{n-1}\rho u_{n-1}$, where $t_0,\dots,t_{n-1},u_0,\dots,u_{n-1} \in \mathrm{GTERM}_{\mathcal{L}}$, then as shown

$$f(t_0,\dots,t_{n-1})\rho f(u_0,\dots,u_{n-1}),$$

that is, $f^{\mathcal{B}}(t_0,\dots,t_{n-1})\rho f^{\mathcal{B}}(u_0,\dots,u_{n-1})$.

If $R \in \mathcal{L} - \{=\}$ is an n-ary relation symbol of $\mathcal{L}$ with $n > 0$ and $t_i\rho u_i$ for $0 \leq i \leq n-1$, then

$$\begin{aligned}
&(t_0,\dots,t_{n-1}) \in R^{\mathcal{B}}\\
&\quad \text{if and only if } \ R(t_0,\dots,t_{n-1}) \in S\\
&\qquad\qquad\qquad\quad (\text{by the definition of } \mathsf{STR}_{\mathcal{L}}(S))\\
&\quad \text{if and only if } \ I(R(t_0,\dots,t_{n-1})) = \mathbf{T}\\
&\qquad\qquad\qquad\quad (\text{by the definition of } S)\\
&\quad \text{if and only if } \ I(R(u_0,\dots,u_{n-1})) = \mathbf{T}\\
&\quad \text{if and only if } \ R(u_0,\dots,u_{n-1}) \in S\\
&\quad \text{if and only if } \ (u_0,\dots,u_{n-1}) \in R^{\mathcal{B}}.
\end{aligned}$$

Thus, ρ is a congruence on $\mathcal{B}$. Let $\mathcal{A} = \mathcal{B}/\rho$ and let h be the canonical homomorphism defined by $h(t) = [t]_\rho$ for $t \in \mathrm{GTERM}_{\mathcal{L}}$.

If $\alpha \in \mathrm{GAFORMNE}_{\mathcal{L}}$, by Theorem 4.5.68, we have

$$\mathcal{A} \models \alpha \text{ if and only if } \mathcal{B} \models \alpha \text{ if and only if } I(\alpha) = \mathbf{T}.$$

If $t, u \in \mathrm{GTERM}_{\mathcal{L}}$, we have the following equivalent statements:

$$\begin{aligned}
&\mathcal{A} \models (t = u)\\
&t^{\mathcal{A}} = u^{\mathcal{A}}\\
&\ \text{(by Theorem 4.5.16)}\\
&h(t^{\mathcal{B}}) = h(u^{\mathcal{B}})\\
&\ \text{(by Theorem 4.5.6)}\\
&h(t) = h(u)\\
&\ \text{(by Corollary 4.10.9)}\\
&[t]_\rho = [u]_\rho\\
&t \rho u\\
&I(t = u) = \mathbf{T}.
\end{aligned}$$

Thus, for all $\alpha \in \mathrm{GAFORM}_{\mathcal{L}}$, we have $\mathcal{A} \models \alpha$ if and only if $I(\alpha) = \mathbf{T}$. By a simple induction argument, we can prove that $\mathcal{A} \models \varphi$ if and only if $I(\varphi) = \mathbf{T}$ for all closed quantifier-free $\mathcal{L}$-formulas φ. Since I is an $\mathcal{L}$-E-model of Γ, $\mathcal{A}$ is a model of Γ. $\square$

Theorem 5.10.34. *Let $\mathcal{L}$ be a first-order language with equality that contains at least one constant symbol and let Γ be a set of closed $\mathcal{L}$-formulas in conjunction normal form. Then, Γ has an $\mathcal{L}$-E-model if and only if $\mathcal{C}_\Gamma$ has an $\mathcal{L}$-E-model.*

Proof. This is a direct consequence of Part (a) of Exercise 86. $\square$

Theorem 5.10.35. *Let $\mathcal{L}$ be a first-order language with equality that contains at least one constant symbol and let $\mathcal{C}$ be a set of $\mathcal{L}$-clauses. Then, $\mathcal{C}$ has a model if and only if $\mathrm{GINST}_{\mathcal{L}}(\mathcal{C})$ has an $\mathcal{L}$-E-model.*

Proof. If $\square \in \mathcal{C}$, then $\square \in \mathrm{GINST}_{\mathcal{L}}(\mathcal{C})$, so $\mathcal{C}$ has no model and $\mathrm{GINST}_{\mathcal{L}}(\mathcal{C})$ has no $\mathcal{L}$-E-model.

If $\square \notin \mathcal{C}$, the following statements are equivalent:

$\mathcal{C}$ has a model
$\Gamma_{\mathcal{C}}$ has a model
(by Corollary 5.8.13)
$(\Gamma_{\mathcal{C}})^{\forall}$ has a model
(by Corollary 4.5.60)
$\mathrm{GINST}_{\mathcal{L}}((\Gamma_{\mathcal{C}})^{\forall})$ has a model
(by Theorem 4.10.41)
$\mathrm{GINST}_{\mathcal{L}}((\Gamma_{\mathcal{C}})^{\forall})$ has an $\mathcal{L}$-E-model
(by Theorem 5.10.33)
$\mathcal{C}_{\mathrm{GINST}_{\mathcal{L}}((\Gamma_{\mathcal{C}})^{\forall})}$ has an $\mathcal{L}$-E-model
(by Theorem 5.10.34)
$\mathrm{GINST}_{\mathcal{L}}(\mathcal{C})$ has an $\mathcal{L}$-E-model
(by Theorem 5.8.65).

$\square$

Let $\mathcal{L}$ be a first-order language with equality that contains at least one constant symbol. By O1 and O2, $(\mathrm{GAFORM}_{\mathcal{L}}, \preceq_{\mathcal{L}})$ is a linearly ordered set and every element of $\mathrm{GAFORM}_{\mathcal{L}}$ has only finitely many predecessors under $\preceq_{\mathcal{L}}$. By Theorems 1.9.10 and 1.9.11, this poset is isomorphic either to a finite poset $(\{0, \ldots, n-1\}, \leq)$ or to $(\mathbf{N}, \leq)$, so the elements of $\mathrm{GAFORM}_{\mathcal{L}}$ can be listed in order by $\preceq_{\mathcal{L}}$ as $\{\alpha_0, \alpha_1, \ldots\}$, a finite or infinite list depending on $\mathcal{L}$. If $0 \leq k \leq |\mathrm{GAFORM}_{\mathcal{L}}|$ for $k \in \mathbf{N}$ (we consider $k < |\mathrm{GAFORM}_{\mathcal{L}}|$ to be true for all $k \in \mathbf{N}$ if $\mathrm{GAFORM}_{\mathcal{L}}$ is infinite), we define $\Gamma_k^{\mathcal{L}}$ as $\{\alpha_0, \ldots, \alpha_{k-1}\}$. A subset Γ of $\mathrm{GAFORM}_{\mathcal{L}}$ is *left segment* of $\mathrm{GAFORM}_{\mathcal{L}}$ if $\Gamma = \Gamma_k^{\mathcal{L}}$ for some $k \in \mathbf{N}$, $0 \leq k \leq |\mathrm{GAFORM}_{\mathcal{L}}|$ or $\Gamma = \mathrm{GAFORM}_{\mathcal{L}}$.

Theorem 5.10.36. *Let $\mathcal{L}$ be a first-order language with equality that contains at least one constant symbol and let $I : \mathrm{GAFORM}_{\mathcal{L}} \longrightarrow$* **Bool**. *Then, I is an $\mathcal{L}$-E-interpretation if and only if $I \restriction \Gamma_k^{\mathcal{L}}$ is an E-interpretation on $\Gamma_k^{\mathcal{L}}$ for all k, $0 \leq k \leq |\mathrm{GAFORM}_{\mathcal{L}}|$.*

Proof. The argument is immediate and it is left to the reader. $\square$

Definition 5.10.37. Let $\mathcal{L}$ be a first-order language with equality that contains at least one constant symbol and let I be an interpretation on a left segment Γ of $\mathrm{GAFORM}_{\mathcal{L}}$. The function $f_I : \mathrm{GWORD}_{\mathcal{L}} \longrightarrow \mathcal{P}(\mathrm{GWORD}_{\mathcal{L}})$ is given by recursion as follows.

Suppose that $w \in \mathrm{GWORD}_{\mathcal{L}}$ and $f_I(v)$ is defined for all $v \prec_{\mathcal{L}} w$. Let $f_I(w)$ be the set of all $v \in \mathrm{GWORD}_{\mathcal{L}}$ satisfying either F1 or F2 where:

F1. w is $(u = t)$, $u \prec_{\mathcal{L}} t$, $f_I(t = u) = \emptyset$ and v is $(t = u)$.[4]

F2. There is an occurrence $(t, i) \in \mathtt{OCC}_t(w)$ and a ground $\mathcal{L}$-term u, where $u \prec_{\mathcal{L}} t$, with $(t = u) \prec_{\mathcal{L}} w$, $(t = u) \in \Gamma$, $I(t = u) = \mathbf{T}$, $f_I(t = u) = \emptyset$ and $v = w[(t, i) \to u]$. In this case we say that $v \in f_I(w)$ via F2 using $(t = u)$.

The relation $\to_I$ on $\mathrm{GWORD}_{\mathcal{L}}$ is the set of pairs $\{(w, v) \mid v \in f_I(w)\}$. We say that w *I-reduces* to v if $w \to_I v$, and w is *I-irreducible* if $f_I(w) = \emptyset$.

If $v \in f_I(w)$ via F2 using $(t = u)$, we write $w \to_I v$ using $(t = u)$. ∎

Note that if $v \in f_I(w)$ via F1, then $v \prec_{\mathcal{L}} w$ by O5 and if $v \in f_I(w)$ via F2, then $v \prec_{\mathcal{L}} w$ by O4. Hence, if $w \to_I v$, then $v \prec_{\mathcal{L}} w$. Thus, there is no infinite sequence $v_0, v_1, \ldots$ with $v_0 \to_I v_1 \to_I \cdots$. In other words, the poset $(\mathrm{GWORD}_{\mathcal{L}}, \to_I)$ is well-founded. In particular, $\to_I$ is irreflexive.

Theorem 5.10.38. *Let $\mathcal{L}$ be a first-order language with equality that contains at least one constant symbol. Suppose that I is an $\mathcal{H}$-interpretation on a left segment Γ of* $\mathrm{GAFORM}_{\mathcal{L}}$*, $\alpha = R(u_0, \ldots, u_{n-1})$ is a ground atomic $\mathcal{L}$-formula, and α I-reduces to β via F2 using $(t = u)$. Then, there is an i, $0 \leq i \leq n-1$ and a term u'_i such that u_i I-reduces to u'_i via F2 using $(t = u)$ and $\beta = R(u_0, \ldots, u_{i-1}, u'_i, u_{i+1}, \ldots, u_{n-1})$.*

Proof. By the definition of F2, for some j, we have:

- $(t, j) \in \mathtt{OCC}_t(\alpha)$;
- $u \prec_{\mathcal{L}} t$;
- $(t = u) \prec_{\mathcal{L}} \alpha$;
- $(t = u) \in \Gamma$;
- $I(t = u) = \mathbf{T}$;

[4]Note that $(t = u) \prec_{\mathcal{L}} (u = t)$ by O5 so $f_I(t = u)$ is defined.

- $f_I(t = u) = \emptyset$.

The occurrence (t, j) in α must be part of one of the u_is and if the corresponding occurrence of t in u_i is (t, j') and $u'_i = u_i[(t, j') \to u]$, then $\beta = R(u_0, \ldots, u_{i-1}, u'_i, u_{i+1}, \ldots, u_{n-1})$. By O6, we have $(u = t) \prec_{\mathcal{L}} u_i$. Since $(t = u) \prec_{\mathcal{L}} (u = t)$ by O5, we have $(t = u) \prec_{\mathcal{L}} u_i$, so $u_i \to_I u'_i$ by F2 using $(t = u)$. $\square$

Corollary 5.10.39. *Let $\mathcal{L}$ be a first-order language with equality that contains at least one constant symbol. Suppose that I is an $\mathcal{H}$-interpretation on a left segment Γ of* GAFORM$_{\mathcal{L}}$*, and α is a ground atomic $\mathcal{L}$-formula $R(u_0, \ldots, u_{n-1})$. If each u_i, $0 \le i \le n-1$, is I-irreducible, then α is not I-reducible via F2.*

Proof. This follows immediately from Theorem 5.10.38. $\square$

Theorem 5.10.40. *Let $\mathcal{L}$ be a first-order language with equality that contains at least one constant symbol. Suppose that I is an $\mathcal{H}$-interpretation on a left segment Γ of* GAFORM$_{\mathcal{L}}$*, $\alpha = R(u_0, \ldots, u_{n-1})$ is a ground atomic $\mathcal{L}$-formula, and $\beta = R(u_0, \ldots, u_{i-1}, u'_i, u_{i+1}, \ldots, u_{n-1})$, where $0 \le i \le n-1$ and $u_i \to_I u'_i$. Then, if the following condition:*

$$\boxed{\begin{array}{l}\textit{the reduction } u_i \to_I u'_i \textit{ uses } (t = u),\ u_i = t \textit{ and either} \\ \alpha = (t' = t) \textit{ with } t' \prec_{\mathcal{L}} u \textit{ or } \alpha = (t = t') \textit{ with } t' \preceq_{\mathcal{L}} u\end{array}}$$

does not hold, we have $\alpha \to_I \beta$.

Proof. Suppose that $u_i \to_I u'_i$ using $(t = u)$. Then, for some j we have:

- $(t, j) \in \mathtt{OCC}_t(u_i)$;
- $u \prec_{\mathcal{L}} t$;
- $(t = u) \prec_{\mathcal{L}} u_i$;
- $(t = u) \in \Gamma$;
- $I(t = u) = \mathbf{T}$;
- $f_I(t = u) = \emptyset$;
- $u'_i = u_i[(t, j) \to u]$.

Let (t, j') be the occurrence of t in α corresponding to the occurrence (t, j) of t in u_i. Then, $\beta = \alpha[(t, j') \to u]$. To show that $\alpha \to_I \beta$ using $(t = u)$ it suffices to show that $(t = u) \prec_{\mathcal{L}} \alpha$.

If α is not an equality, or t is a proper subterm of u_i, or $t = u_i$ and α is $(t = t')$ or $(t' = t)$ with $u \prec_{\mathcal{L}} t'$, then, by O6, we have $(u = t) \prec_{\mathcal{L}} \alpha$, so by O5, $(t = u) \prec_{\mathcal{L}} (u = t) \prec_{\mathcal{L}} \alpha$. Since the condition shown in the box does not hold, the only remaining case is $\alpha = (u = t)$ and then $(t = u) \prec_{\mathcal{L}} (u = t) = \alpha$, by O5. □

Theorem 5.10.41. *Let $\mathcal{L}$ be a first-order language with equality that contains at least one constant symbol. Suppose that I is an $\mathcal{H}$-interpretation on a left segment Γ of* $\mathrm{GAFORM}_{\mathcal{L}}$*. Let u_0, u_1 be ground $\mathcal{L}$-terms with $u_0 \preceq_{\mathcal{L}} u_1$ and assume that $u_0 \rightarrow_I u_0'$ for some ground $\mathcal{L}$-term u_0'. Then, $(u_0 = u_1) \rightarrow_I (u_0' = u_1)$ and $(u_1 = u_0) \rightarrow_I (u_1 = u_0')$.*

Proof. Suppose $u_0 \rightarrow_I u_0'$ using $(t = u)$. Then, $u \prec_{\mathcal{L}} t$, while $t \preceq_{\mathcal{L}} u_0$ by O3, so $u \prec_{\mathcal{L}} u_0 \preceq_{\mathcal{L}} u_1$. The result follows by Theorem 5.10.40 because the boxed condition does not hold. □

Theorem 5.10.42. *Let $\mathcal{L}$ be a first-order language with equality that contains at least one constant symbol. Suppose that I is an E-interpretation on a left segment Γ of* $\mathrm{GAFORM}_{\mathcal{L}}$*, $\alpha \in \Gamma$ and $\alpha \rightarrow_I \beta$. Then, $I(\alpha) = I(\beta)$.*

Proof. As noted previously, $\alpha \rightarrow_I \beta$ implies $\beta \prec_{\mathcal{L}} \alpha$, so $\beta \in \Gamma$ and $I(\beta)$ is defined.

If $\alpha \rightarrow_I \beta$ via F1, then for some ground $\mathcal{L}$-terms t and u, we have $\alpha = (t = u)$ and $\beta = (u = t)$, so by E2, $I(\alpha) = I(\beta)$.

If $\alpha \rightarrow_I \beta$ via F2 using $(t = u)$, say $\beta = \alpha[(t, i) \rightarrow u]$, then, by F2, $(t = u) \in \Gamma$ and $I(t = u) = \mathbf{T}$, so $I(\alpha) = I(\beta)$ by E3. □

Theorem 5.10.43. *Let $\mathcal{L}$ be a first-order language with equality that contains at least one constant symbol. Suppose that I' is an $\mathcal{H}$-interpretation on a left segment $\Gamma_k^{\mathcal{L}}$ for some $k <$* $|\mathrm{GAFORM}_{\mathcal{L}}|$*, I is an $\mathcal{H}$-interpretation on Γ where either $\Gamma = \Gamma_r^{\mathcal{L}}$ for some r, $k < r \leq$* $|\mathrm{GAFORM}_{\mathcal{L}}|$ *or* $\Gamma = \mathrm{GAFORM}_{\mathcal{L}}$ *and $I' = I \lceil \Gamma_k^{\mathcal{L}}$. Then, we have:*

a. for all $\beta \in \Gamma_{k+1}^{\mathcal{L}}$, $f_{I'}(\beta) = f_I(\beta)$;
b. for all $\beta \in$ $\mathrm{GAFORM}_{\mathcal{L}}$*, if $\gamma \in f_{I'}(\beta)$ via F2 using $(t = u)$, then $\gamma \in f_I(\beta)$ via F2 using $(t = u)$;*
c. for all $\beta \in$ $\mathrm{GAFORM}_{\mathcal{L}}$*, if $\gamma \in f_I(\beta)$ via F1, then $\gamma \in f_{I'}(\beta)$ via F1.*

Proof. (a:) Suppose that $\beta \in \Gamma_{k+1}^{\mathcal{L}}$ and the result holds for all β' with $\beta' \prec_{\mathcal{L}} \beta$. For any ground $\mathcal{L}$-equality $(t = u)$ with $(t = u) \prec_{\mathcal{L}} \beta$, we have $(t = u) \in \Gamma_k^{\mathcal{L}} \subseteq \Gamma$ and $I'(t = u) = I(t = u)$. Further, $f_{I'}(t = u) = f_I(t = u)$ by inductive hypothesis. It follows from the definitions of F1 and F2 that $f_{I'}(\beta) = f_I(\beta)$.

(b:) Suppose that $\gamma \in f_{I'}(\beta)$ via F2 using $(t = u)$. Then, $(t = u) \in \Gamma_k^{\mathcal{L}} \subseteq \Gamma$, $I(t = u) = I'(t = u) = \mathbf{T}$ and by Part (a), $f_{I'}(t = u) = f_I(t = u) = \emptyset$. Thus, $\gamma \in f_I(\beta)$ via F2 using $(t = u)$.

(c:) Suppose that $\gamma \in f_I(\beta)$ via F1. Then, for some ground $\mathcal{L}$-terms t and u with $u \prec_{\mathcal{L}} t$, $\beta = (u = t)$, $f_I(t = u) = \emptyset$ and $\gamma = (t = u)$. Since $(t = u)$ cannot be reduced via F1, it follows from Part (b) that $f_{I'}(t = u) \subseteq f_I(t = u) = \emptyset$, hence $\gamma \in f_{I'}(\beta)$ via F1. □

Theorem 5.10.44. *Let $\mathcal{L}$ be a first-order language with equality that contains at least one constant symbol. Suppose that I is an E-interpretation on a left segment $\Gamma_k^{\mathcal{L}}$ for some $k < |\mathrm{GAFORM}_{\mathcal{L}}|$, and $\alpha_k \rightarrow_I \beta_0$ and $\alpha_k \rightarrow_I \beta_1$. Then, we have $I(\beta_0) = I(\beta_1)$.*

Proof. Note that $\alpha_k \rightarrow_I \beta_0$ and $\alpha_k \rightarrow_I \beta_1$ imply $\beta_0, \beta_1 \prec_{\mathcal{L}} \alpha_k$, so $\beta_0, \beta_1 \in \Gamma_k^{\mathcal{L}}$ and $I(\beta_0)$ and $I(\beta_1)$ are defined. We may assume that $\beta_0 \neq \beta_1$.

Suppose that one of β_0 or β_1 is placed into $f_I(\alpha_k)$ via F1, say β_0. Then, for some ground $\mathcal{L}$-terms t, u we have:

- $\alpha_k = (u = t)$;
- $u \prec_{\mathcal{L}} t$;
- $f_I(t = u) = \emptyset$;
- $\beta_0 = (t = u) = \alpha_{k-1}$ (by O5).

Since only one element can be put into $f_I(\alpha_k)$ via F1, β_1 must be placed into $f_I(\alpha_k)$ via F2, so for some ground $\mathcal{L}$-terms t', u' and some index i', we have:

- $(t', i') \in \mathtt{OCC}_{t'}(u = t)$;
- $u' \prec_{\mathcal{L}} t'$;
- $(t' = u') \prec_{\mathcal{L}} \alpha_k = (u = t)$;
- $I(t' = u') = \mathbf{T}$;
- $f_I(t' = u') = \emptyset$;
- $\beta_1 = (u = t)[(t', i) \rightarrow u']$.

If $(t' = u') \prec_{\mathcal{L}} (t = u)$, then $(t = u)$ would be I-reducible via F2 using $(t' = u')$, contradicting $f_I(t = u) = \emptyset$, so $(t = u) \preceq_{\mathcal{L}} (t' = u') \prec_{\mathcal{L}} (u = t)$. By O5, this implies $(t' = u') = (t = u)$, so $I(\beta_0) = I(t = u) = I(t' = u') = \mathbf{T}$. Since $u \prec_{\mathcal{L}} t$, t cannot occur in u by O3, so the only occurrence of t in $(u = t)$ is the one at the right of the $=$ symbol, and $\beta_1 = (u = t)[(t, i') \to u] = (u = u)$. Since I is an E-interpretation on $\Gamma_k^{\mathcal{L}}$, we have $I(\beta_1) = I(u = u) = \mathbf{T} = I(\beta_0)$.

Now suppose that β_0 and β_1 are both placed into $f_I(\alpha_k)$ via F2. Then, for some ground $\mathcal{L}$-terms t_0, u_0, t_1, u_1 and some indices i_0, i_1, we have for $j \in \{0, 1\}$:

- $(t_j, i_j) \in \mathtt{OCC}_{t_j}(\alpha_k)$;
- $u_j \prec_{\mathcal{L}} t_j$;
- $(t_j = u_j) \in \Gamma_k^{\mathcal{L}}$;
- $I(t_j = u_j) = \mathbf{T}$;
- $f_I(t_j = u_j) = \emptyset$;
- $\beta_j = \alpha_k[(t_j, i_j) \to u_j]$.

First suppose that the occurrences (t_0, i_0) and (t_1, i_1) do not overlap. Without loss of generality, we may assume that $i_0 < i_1$. Then, making repeated use of E3, we have for an appropriate value of i_1':

$$\begin{aligned}
I(\beta_1) &= I(\alpha_k[(t_1, i_1) \to u_1]) \\
&= I(\alpha_k[(t_1, i_1) \to u_1][(t_0, i_0) \to u_0]) \\
&= I(\alpha_k[(t_0, i_0) \to u_0][(t_1, i_1') \to u_1]) \\
&= I(\alpha_k[(t_0, i_0) \to u_0] = I(\beta_0).
\end{aligned}$$

Consider now the case when the occurrences (t_0, i_0) and (t_1, i_1) overlap. Since no proper prefix of a term is a suffix of another term by Exercise 53 of Chapter 1, one occurrence is part of the other, say (t_1, i_1) is part of (t_0, i_0). We claim that $t_1 = t_0$, so $i_1 = i_0$. Suppose that this is not the case, that is t_1 is a proper subterm of t_0. By O3, $u_1 \prec_{\mathcal{L}} t_1 \prec_{\mathcal{L}} t_0$, so by O6, $(u_1 = t_1) \prec_{\mathcal{L}} (t_0 = u_0)$ and therefore by O5

$$(t_1 = u_1) \prec_{\mathcal{L}} (u_1 = t_1) \prec_{\mathcal{L}} (t_0 = u_0).$$

Thus, $(t_0 = u_0)$ is I-reducible via F2 using $(t_1 = u_1)$. This contradicts $f_I(t_0 = u_0) = \emptyset$. We have established that $t_1 = t_0$ and $i_1 = i_0$ and, without loss of generality, we may assume that $u_0 \prec_{\mathcal{L}} u_1$. Thus we have:

$$\begin{aligned}\beta_1 &= \alpha_k[(t_1, i_1) \to u_1]\\ &= \alpha_k[(t_0, i_0) \to u_1]\\ &= \alpha_k[(t_0, i_0) \to u_0][(u_0, i_0) \to u_1]\\ &= \beta_0[(u_0, i_0) \to u_1].\end{aligned}$$

We have $u_0 \prec_{\mathcal{L}} u_1 \prec_{\mathcal{L}} t_1$, so by O6, $(u_0 = u_1) \prec_{\mathcal{L}} (t_1 = u_1)$ and hence $(u_0 = u_1) \in \Gamma^{\mathcal{L}}_k$. Since $I(t_1 = u_0) = I(t_0 = u_0) = \mathbf{T}$, by E3, $I(u_0 = u_1) = I(t_1 = u_1) = \mathbf{T}$. Again, by E3, $I(\beta_1) = I(\beta_0[(u_0, i_0) \to u_1]) = I(\beta_0)$. □

Theorem 5.10.45. *Let $\mathcal{L}$ be a first-order language with equality that contains at least one constant symbol. Suppose that $0 \leq k < |\text{GAFORM}_{\mathcal{L}}|$. I' is an E-interpretation on $\Gamma^{\mathcal{L}}_k$ and I is an $\mathcal{H}$-interpretation on $\Gamma^{\mathcal{L}}_{k+1}$ such that $I{\upharpoonright}\Gamma^{\mathcal{L}}_k = I'$. Then, I is an E-interpretation on $\Gamma^{\mathcal{L}}_{k+1}$ if and only if one of the following conditions holds:*

1. *α_k is I'-reducible and for all β such that $\alpha_k \to_{I'} \beta$, $I(\alpha_k) = I(\beta)$;*
2. *α_k is I'-irreducible and of the form $(t = t)$ and $I(t = t) = \mathbf{T}$;*
3. *α_k is I'-irreducible and not of the form $(t = t)$.*

Proof. Suppose that (1), (2) and (3) are all false. If α_k is I'-irreducible, then, since (3) is false, $\alpha_k = (t = t)$ for some ground $\mathcal{L}$-term t. Since (2) is false, $I(t = t) = \mathbf{F}$. Thus, I is not an E-interpretation on $\Gamma^{\mathcal{L}}_{k+1}$ because E1 is violated.

If α_k is I'-reducible, then, since (1) is false, there is a formula β with $\alpha_k \to_{I'} \beta$ and $I(\alpha_k) \neq I(\beta)$. Since by Theorem 5.10.43 $\alpha_k \to_I \beta$, I is not an E-interpretation on $\Gamma^{\mathcal{L}}_{k+1}$, by Theorem 5.10.42.

Conversely, suppose that one of the conditions (1), (2) or (3) is true. To show that I is an E-interpretation on $\Gamma^{\mathcal{L}}_{k+1}$, we must show the following five statements hold:

P1. If $\alpha_k = (t = t)$ for some ground $\mathcal{L}$-term t, then $I(t = t) = \mathbf{T}$.
P2. If $\alpha_k = (u = t)$ for some ground $\mathcal{L}$-terms t and u with $u \prec_{\mathcal{L}} t$, then $I(\alpha_k) = I(t = u)$.
P3. If $\alpha_k = (t = u)$ for some ground $\mathcal{L}$-terms t, u, $\beta \preceq_{\mathcal{L}} \alpha_k$, $(t, i) \in \texttt{OCC}_t(\beta)$, $\beta[(t, i) \to u] \preceq_{\mathcal{L}} \alpha_k$, and $I(t = u) = \mathbf{T}$, then $I(\beta) = I(\beta[(t, i) \to u])$.

P4. If for the ground $\mathcal{L}$-terms t and u, $(t, i) \in \mathtt{OCC}_t(\alpha_k)$, $(t = u) \prec_{\mathcal{L}} \alpha_k$, $\alpha_k[(t, i) \to u] \preceq \alpha_k$, and $I(t = u) = \mathbf{T}$, then $I(\alpha_k) = I(\alpha_k[(t, i) \to u])$.

P5. If for the ground $\mathcal{L}$-terms t and u and $\beta \prec_{\mathcal{L}} \alpha_k$, $(t, i) \in \mathtt{OCC}_t(\beta)$, $(t = u) \prec_{\mathcal{L}} \alpha_k$, $\alpha_k = \beta[(t, i) \to u]$, and $I(t = u) = \mathbf{T}$, then $I(\alpha_k) = I(\beta)$.

Proof of P1: Suppose that $\alpha_k = (t = t)$ for some ground $\mathcal{L}$-term t. If α_k is I'-irreducible, then (1) and (3) are false, so (2) is true and $I(t = t) = \mathbf{T}$.

Suppose that α_k is I'-reducible, say $\alpha_k \to_{I'} \beta$. Then, (1) must be true and $I(\alpha_k) = I(\beta) = I'(\beta)$, since $\beta \prec_{\mathcal{L}} \alpha_k$. Because α_k cannot be reduced via F1, the reduction must be via F2, so for some ground $\mathcal{L}$-terms t', u' and index i, we have:

- $(t', i) \in \mathtt{OCC}_{t'}(t = t)$;
- $u' \prec_{\mathcal{L}} t'$;
- $(t' = u') \prec_{\mathcal{L}} (t = t)$;
- $I'(t' = u') = \mathbf{T}$;
- $\beta = (t = t)[(t', i) \to u']$.

Let t'' be the result of replacing the occurrence of t' in t by u'. We assume that this occurrence is to the left of the equality symbol. so $\beta = (t'' = t)$. (A similar argument applies if the occurrence is located to the right of the equality symbol.) By O4, $t'' \prec_{\mathcal{L}} t$, so by O6, $(t'' = t'') \prec_{\mathcal{L}} (t'' = t) = \beta$. We have:

$$\begin{aligned} I(t = t) &= I'(\beta) \\ &= I'(t'' = t'') \text{ (by E3, since } I' \text{ is an E-interpretation on } \Gamma_k^{\mathcal{L}}) \\ &= \mathbf{T} \text{ (by E1).} \end{aligned}$$

Proof of P2: Suppose that $\alpha_k = (u = t)$ for the ground $\mathcal{L}$-terms t, u with $u \prec_{\mathcal{L}} t$. If $(t = u)$ is I'-irreducible, then $(u = t) \to_{I'} (t = u)$ via F1 and since (1) must be true, $I(u = t) = I(t = u)$.

Suppose that $(t = u)$ is I'-reducible. The reduction must be via F2, say $(t = u) \to_{I'} (t' = u')$ via F2 using $(t'' = u'')$. Since $(t'' = u'') \prec_{\mathcal{L}} (t = u) \prec_{\mathcal{L}} (u = t)$, we have $(u = t) \to_{I'} (u' = t')$ via F2 using $(t'' = u'')$. Thus (1) must hold and we have:

$$\begin{aligned}
I(u = t) &= I(u' = t') \text{ (by (1))} \\
&= I'(u' = t') \text{ (since } (u' = t') \in \Gamma_k^{\mathcal{L}}) \\
&= I'(t' = u') \text{ (by E2 since } (t' = u') \in \Gamma_k^{\mathcal{L}} \text{ and} \\
&\qquad I' \text{ is an E-interpretation on } \Gamma_k^{\mathcal{L}}) \\
&= I'(t = u) \text{ (by Theorem 5.10.42 since } (t = u) \in \Gamma_k^{\mathcal{L}}) \\
&= I(t = u).
\end{aligned}$$

Proof of P3: Suppose that $\alpha_k = (t = u)$ for some ground $\mathcal{L}$-terms t and u, $\beta \preceq_{\mathcal{L}} \alpha_k$, $(t, i) \in \mathtt{OCC}_t(\beta)$, $\beta[(t, i) \to u] \preceq_{\mathcal{L}} \alpha_k$ and $I(t = u) = \mathbf{T}$. We must show that $I(\beta) = I(\beta[(t, i) \to u])$.

We consider cases based on the relationship between t and u.

Case 1: $t = u$. Then, $\beta[(t, i) \to u] = \beta$ and the result is immediate.

Case 2: $t \prec_{\mathcal{L}} u$. Then, $(u = t) \in \Gamma_k^{\mathcal{L}}$ and by P2, $I'(u = t) = I(u = t) = I(t = u) = \mathbf{T}$. We have three subcases:

Case 2.1: We have both $\beta \prec_{\mathcal{L}} \alpha_k$ and $\beta[(t, i) \to u] \prec_{\mathcal{L}} \alpha_k$. Since I' is an E-interpretation on $\Gamma_k^{\mathcal{L}}$, we have:

$$\begin{aligned}
I(\beta[(t, i) \to u]) &= I'(\beta[(t, i) \to u]) \\
&= I'(\beta[(t, i) \to u][(u, i) \to t]) \text{ by E3} \\
&= I'(\beta) = I(\beta).
\end{aligned}$$

Case 2.2: $\beta = \alpha_k$. Then, by O4, we have

$$\alpha_k = \beta = \beta[(t, i) \to u][(u, i) \to t] \prec_{\mathcal{L}} \beta[(t, i) \to u],$$

thus contradicting $\beta[(t, i) \to u] \preceq_{\mathcal{L}} \alpha_k$. Hence, this case cannot occur.

Case 2.3: $\beta[(t, i) \to u] = \alpha_k = (t = u)$. Then, β must have the form $(t' = u')$. If the replacement of t with u within β to produce $(t = u)$ took place to the left of the $=$, then t would have u as a subterm, which is impossible by O3 since we are assuming $t \prec_{\mathcal{L}} u$. Thus, $\beta = (t = t)$. Since $\beta \preceq_{\mathcal{L}} \alpha_k$ and $\beta \neq \alpha_k$, we must have $\beta \in \Gamma_k^{\mathcal{L}}$. Using E1, we have

$$I(\beta) = I(t = t) = I'(t = t) = \mathbf{T} = I(\alpha_k) = I(\beta[(t, i) \to u]).$$

Case 3: $u \prec_{\mathcal{L}} t$. We have $\beta \preceq_{\mathcal{L}} \alpha_k = (t = u) \preceq_{\mathcal{L}} (u = t)$ and t occurs in β. Since $u \prec_{\mathcal{L}} t$, by O6, β must have one of the forms

$(t = t')$ or $(t' = t)$ with $t' \preceq_{\mathcal{L}} u$. Note that if β has the form $(t' = t)$, then in fact $t' \prec_{\mathcal{L}} u$ since $\beta \prec_{\mathcal{L}} (u = t)$. Based on the I'-reducibility of α_k we have two subcases:

Case 3.1: α_k is I'-irreducible. Depending on the form of β, we have three subsubcases.

Case 3.1.1: $\beta = (t' = t)$ with $t' \prec_{\mathcal{L}} u$. Let $\overset{*}{\rightarrow}_{I'}$ denote the reflexive, transitive closure of $\rightarrow_{I'}$. Since $\rightarrow_{I'}$ is well-founded, there is a ground $\mathcal{L}$-term t'' such that $t' \overset{*}{\rightarrow}_{I'} t''$ and t'' is I'-irreducible. We have $t' \prec_{\mathcal{L}} u \prec_{\mathcal{L}} t$, so by repeated use of Theorem 5.10.41, $\beta = (t' = t) \overset{*}{\rightarrow}_{I'} (t'' = t)$.

We claim that $(t = t'')$ is I'-irreducible. Suppose that this is not the case, that is, $(t = t'')$ is I'-reducible. Since $t'' \preceq_{\mathcal{L}} t' \prec_{\mathcal{L}} t$, $(t = t'')$ is not I'-reducible via F1, so $(t = t'')$ must be I'-reducible via F2, say using $(t_0 = u_0)$. Since t'' is I'-irreducible, by Theorem 5.10.38, t is I'-reducible via F2 using $(t_0 = u_0)$. In particular, t_0 occurs in t. We have $u_0 \prec_{\mathcal{L}} t_0$, $(t_0 = u_0) \prec_{\mathcal{L}} (t = t'') \prec_{\mathcal{L}} (t'' = t) \prec_{\mathcal{L}} (t = u)$, where the latter inequality holds by O6 because $t'' \preceq_{\mathcal{L}} t' \prec_{\mathcal{L}} u$, $I'(t_0 = u_0) = \mathbf{T}$, and $f_{I'}(t_0 = u_0) = \emptyset$. Thus, $\alpha_k = (t = u)$ is I'-reducible via F2 using $(t_0 = u_0)$ contradicting the case assumption. This establishes that $(t = t'')$ is I'-irreducible, so $(t'' = t) \rightarrow_{I'} (t = t'')$ via F1 and therefore, $\beta \overset{*}{\rightarrow}_{I'} (t = t'')$. Since $\beta \neq (t = u) = \alpha_k$, $\beta \in \Gamma_k^{\mathcal{L}}$ so by repeated use of Theorem 5.10.42, $I'(\beta) = I'(t = t'')$. If $I'(t = t'') = \mathbf{T}$, then we have $t'' \prec_{\mathcal{L}} t$, $(t = t'') \prec_{\mathcal{L}} (t = u)$, and $(t = t'')$ is I'-irreducible, so $\alpha_k = (t = u)$ would be I'-reducible by F2 using $(t = t'')$, contradicting the case assumption. Thus, $I'(\beta) = I'(t = t'') = \mathbf{F}$.

By a similar argument, we will show that $I'(\beta[(t,i) \rightarrow u]) = \mathbf{F}$, giving $I(\beta) = I(\beta[(t,i) \rightarrow u])$. Note that since $t' \prec_{\mathcal{L}} t$, by O3, t cannot occur in t', so $\beta[(t,i) \rightarrow u] = (t' = t)[(t,i) \rightarrow u] = (t' = u)$. Again, letting t'' be an I'-irreducible ground term with $t' \overset{*}{\rightarrow}_{I'} t''$, we reach the following conclusions:

- $(t' = u) \overset{*}{\rightarrow}_{I'} (t'' = u)$ by Theorem 5.10.41 since $t' \prec_{\mathcal{L}} u$;
- $(u = t'')$ is I'-irreducible because else $\alpha_k = (t = u)$ would be I'-reducible;
- $\beta[(t,i) \rightarrow u] = (t' = u) \overset{*}{\rightarrow}_{I'} (u = t'')$;
- $I'(\beta[(t,i) \rightarrow u]) = I'(u = t'')$;
- $I'(u = t'') = \mathbf{F}$ because else α_k would be I'-reducible;

- $I'(\beta[(t,i) \to u]) = \mathbf{F}$.

Case 3.1.2: $\beta = (t = u)$. Since $u \prec_{\mathcal{L}} t$, t cannot occur in u by O3, so $\beta[(t,i) \to u] = (u = u)$. Since $\beta[(t,i) \to u] \prec_{\mathcal{L}} \beta = \alpha_k$, $(u = u) \in \Gamma_k^{\mathcal{L}}$. Therefore, by E1 we have:

$$I(\beta[(t,i) \to u]) = I'(u = u) = \mathbf{T} = I(t = u) = I(\beta).$$

Case 3.1.3: $\beta = (t = t')$ with $t' \prec_{\mathcal{L}} u$. The argument is similar to that of Case 3.1.1.

Case 3.2: $\alpha_k = (t = u)$ is I'-reducible. Then, Condition (1) of the theorem must hold. Since $u \prec_{\mathcal{L}} t$, α_k cannot be I'-reduced via F1, so α_k is I'-reducible via F2 say by using $(t_0 = u_0)$. Then, $u_0 \prec_{\mathcal{L}} t_0$, $(t_0 = u_0) \prec_{\mathcal{L}} \alpha_k$, $I'(t_0 = u_0) = \mathbf{T}$ and t_0 occurs in $\alpha_k = (t = u)$. Based on the form of β we have three subsubcases:

Case 3.2.1: $\beta = (t' = t)$ with $t' \prec_{\mathcal{L}} u$. Since $t' \prec_{\mathcal{L}} u \prec_{\mathcal{L}} t$, t cannot occur in t', so $\beta[(t,i) \to u] = (t' = t)[(t,i) \to u] = (t' = u)$. Now we encounter the following situations:

Case 3.2.1.1: the occurrence of t_0 reduced in α_k is to the left of $=$, say (t_0, j) is the occurrence in t. Let $t'' = t[(t_0, j) \to u_0]$. Then, $\alpha_k = (t = u) \to_{I'} (t'' = u)$ and by Condition (1), $\mathbf{T} = I(\alpha_k) = I(t'' = u) = I'(t'' = u)$. We have:

$$\begin{aligned} I(\beta) &= I'(t' = t) \\ &= I'(t' = t'') \text{ (by E3 since } I'(t_0 = u_0) = \mathbf{T}) \\ &= I'(t' = u) \text{ (by E3 since } I'(t'' = u) = \mathbf{T}) \\ &= I'(\beta[(t,i) \to u]) = I(\beta[(t,i) \to u]). \end{aligned}$$

Case 3.2.1.2: The occurrence of t_0 reduced in α_k is to the right of $=$, say (t_0, j) is the occurrence in u. Let $u'' = u[(t_0, j) \to u_0]$. Then, $\alpha_k = (t = u) \to_{I'} (t = u'')$ and by Condition (1), $\mathbf{T} = I(\alpha_k) = I(t = u'') = I'(t = u'')$. We have:

$$\begin{aligned} I(\beta) &= I'(t' = t) \\ &= I'(t' = u'') \text{ (by E3 since } I'(t = u'') = \mathbf{T}) \\ &= I'(t' = u) \text{ (by E3 since } I'(t_0 = u_0) = \mathbf{T}) \\ &= I'(\beta[(t,i) \to u]) = I(\beta[(t,i) \to u]). \end{aligned}$$

Case 3.2.2: $\beta = (t = u)$. The argument of Case 3.1.2 does not depend on the I'-reducibility of α_k so it applies in this case as well.

Case 3.2.3: $\beta = (t = t')$ with $t' \prec_{\mathcal{L}} u$. The argument of Case 3.2.1 can be easily modified to apply in this case.

Proof of P4: Suppose that t and u are ground $\mathcal{L}$-terms, $(t, i) \in \mathtt{OCC}_t(\alpha_k)$, $(t = u) \prec_{\mathcal{L}} \alpha_k$, $\alpha_k[(t,i) \to u] \preceq_{\mathcal{L}} \alpha_k$ and $I(t = u) = \mathbf{T}$. We must show that $I(\alpha_k) = I(\alpha_k[(t,i) \to u])$.

Case 1: $\alpha_k = \alpha_k[(t,i) \to u]$. The result is immediate.

Case 2: $\alpha_k[(t,i) \to u] \prec_{\mathcal{L}} \alpha_k$. Then, by O4, $u \prec_{\mathcal{L}} t$. We distinguish two subcases based on the I'-reducibility of t.

Case 2.1: t is I'-irreducible. Let u' be a ground $\mathcal{L}$-term such that $u \overset{*}{\to}_{I'} u'$ and u' is I'-irreducible. Since $u \prec_{\mathcal{L}} t$, by Theorem 5.10.41 $(t = u) \overset{*}{\to}_{I'} (t = u')$, so we have:

$$\begin{aligned}
\mathbf{T} &= I(t = u)\\
&= I'(t = u)\\
&= I'(t = u') \text{ (by Theorem 5.10.42 because } (t = u) \prec_{\mathcal{L}} \alpha_k)\\
&= I'(u = u') \text{ (by E3 since } I'(t = u) = \mathbf{T}).
\end{aligned}$$

By Theorem 5.10.38, $(t = u')$ is not I'-reducible via F2 and, since $u' \preceq_{\mathcal{L}} u \prec_{\mathcal{L}} t$, $(t = u')$ is not I'-reducible via F1. Therefore $(t = u')$ is I'-irreducible and $I'(t = u') = \mathbf{T}$. Thus, $\alpha_k \to_{I'} \alpha_k[(t,i) \to u']$ via F2, so α_k is I'-reducible and Condition (1) holds. We have:

$$\begin{aligned}
I(\alpha_k) &= I(\alpha_k[(t,i) \to u']) \text{ (by Condition (1))}\\
&= I'(\alpha_k[(t,i) \to u'])\\
&= I'(\alpha_k[(t,i) \to u'][(u',i) \to u]) \text{ (by E3 since } I'(u = u') = \mathbf{T})\\
&= I'(\alpha_k[(t,i) \to u]) = I(\alpha_k[(t,i) \to u]).
\end{aligned}$$

Case 2.2: t is I'-reducible. Then, for some ground $\mathcal{L}$-terms t_0, u_0 and index j, $(t_0, j) \in \mathtt{OCC}_{t_0}(t)$, $u_0 \prec_{\mathcal{L}} t_0$, $(t_0 = u_0) \prec_{\mathcal{L}} t$, $(t_0 = u_0) \in \Gamma_k^{\mathcal{L}}$, $I'(t_0 = u_0) = \mathbf{T}$ and $f_{I'}(t_0 = u_0) = \emptyset$. Let (t_0, j') be the corresponding occurrence of t_0 in α_k, that is, $j' = i + j$. Then, by E3,

$$\mathbf{T} = I'(t = u) = I'(t[(t_0, j) \to u_0] = u).$$

We have $\alpha_k \to_{I'} \alpha_k[(t_0, j') \to u_0]$ via F2, so α_k is I'-reducible and Condition (1) holds. Furthermore,

$$\begin{aligned}
I(\alpha_k) &= I(\alpha_k[(t_0, j') \to u_0]) \text{ (by Condition (1))}\\
&= I'(\alpha_k[(t_0, j') \to u_0])\\
&= I'(\alpha_k[(t, i) \to t[(t_0, j) \to u_0]])\\
&= I'(\alpha_k[(t, i) \to t[(t_0, j) \to u_0]][(t[(t_0, j) \to u_0], i) \to u])\\
&\quad \text{(by E3 because } I'(t[(t_0, j) \to u_0] = u) = \mathbf{T})\\
&= I'(\alpha_k[(t, i) \to u]) = I(\alpha_k[(t, i) \to u]).
\end{aligned}$$

Proof of P5: Suppose that for the ground $\mathcal{L}$-terms t and u, and $\beta \preceq_{\mathcal{L}} \alpha_k$, we have $(t, i) \in \mathtt{OCC}_t(\beta)$, $(t = u) \prec_{\mathcal{L}} \alpha_k$, $\alpha_k = \beta[(t, i) \to u]$, and $I(t = u) = \mathbf{T}$. We must show that $I(\alpha_k) = I(\beta)$.

If $\beta = \alpha_k$, the result is immediate, so we can assume that $\beta \prec_{\mathcal{L}} \alpha_k$. Then, by O4, we must have $t \prec_{\mathcal{L}} u$ and by O5, $(u = t) \prec_{\mathcal{L}} (t = u) \prec_{\mathcal{L}} \alpha_k$. By E2, we have $I'(u = t) = I'(t = u) = \mathbf{T}$. By P4,

$$\begin{aligned}
I(\alpha_k) &= I(\alpha_k[(u, i) \to t])\\
&= I(\beta[(t, i) \to u][(u, i) \to t]) = I(\beta).
\end{aligned}$$

□

Definition 5.10.46. Let $\mathcal{L}$ be a first-order language with equality that contains at least one constant symbol. An *$\mathcal{LH}$-interpretation tree* is a lot $\mathtt{T}$ such that

- $q \in \mathrm{Dom}(\mathtt{T})$ implies that $|q| \leq |\mathrm{GAFORM}_{\mathcal{L}}|$ and $\mathtt{T}(q)$ is an $\mathcal{H}$-interpretation on $\Gamma^{\mathcal{L}}_{|q|}$ denoted by $I^{\mathtt{T}}_q$;
- if $qi \in \mathrm{Dom}(\mathtt{T})$ for $i \in \{0, 1\}$, then $I^{\mathtt{T}}_{qi} \restriction \Gamma^{\mathcal{L}}_{|q|} = I^{\mathtt{T}}_q$.

If $\mathtt{B}$ is a branch of an $\mathcal{LH}$-interpretation tree $\mathtt{T}$, we denote $\bigcup_{q \in \mathtt{B}} I^{\mathtt{T}}_q$ by $I^{\mathtt{T}}_{\mathtt{B}}$.

An $\mathcal{LH}$-interpretation tree $\mathtt{T}$ is an *$\mathcal{L}$-semantic tree* if the following conditions hold for every $q \in \mathrm{Dom}(\mathtt{T})$ with $k = |q| < |\mathrm{GAFORM}_{\mathcal{L}}|$:

- T1. If α_k is $I^{\mathtt{T}}_q$-reducible, then for some β with $\alpha_k \to_{I^{\mathtt{T}}_q} \beta$, q has one immediate descendant $q0$ in $\mathtt{T}$ and $I^{\mathtt{T}}_{q0}(\alpha_k) = I^{\mathtt{T}}_q(\beta)$. Note that $\beta \prec_{\mathcal{L}} \alpha_k$ so $I^{\mathtt{T}}_q(\beta)$ is defined.
- T2. If α_k is $I^{\mathtt{T}}_q$-irreducible and has the form $(t = t)$, then q has one immediate descendant $q0$ in $\mathtt{T}$ and $I^{\mathtt{T}}_{q0}(\alpha_k) = \mathbf{T}$.

- T3. If α_k is $I_q^{\mathtt{T}}$ irreducible and not of the form $(t = t)$, then q has two immediate descendants $q0$ and $q1$ in $\mathtt{T}$, $I_{q0}^{\mathtt{T}}(\alpha_k) = \mathbf{F}$ and $I_{q1}^{\mathtt{T}}(\alpha_k) = \mathbf{T}$.

□

Note that if $\mathtt{T}$ is an $\mathcal{L}$-semantic tree and $|\text{GAFORM}_{\mathcal{L}}| = n$, then every branch of $\mathtt{T}$ has length n, and if $\text{GAFORM}_{\mathcal{L}}$ is infinite, then every branch of $\mathtt{T}$ is infinite.

Theorem 5.10.47. *Let $\mathcal{L}$ be a first-order language with equality that contains at least one constant symbol and let $\mathtt{T}$ be an $\mathcal{L}$-semantic tree. Then, for all $q \in \text{Dom}(\mathtt{T})$, $I_q^{\mathtt{T}}$ is an E-interpretation on $\Gamma_{|q|}^{\mathcal{L}}$.*

Proof. The argument is by induction on $|q|$. For the basis step, $q = \lambda$, $\Gamma_{|q|}^{\mathcal{L}} = \Gamma_0^{\mathcal{L}} = \emptyset$ and $I_\lambda^{\mathtt{T}}$ is the empty function which is vacuously an E-interpretation on $\emptyset$.

For the inductive step, suppose that $|q| < |\text{GAFORM}_{\mathcal{L}}|$ and $I_q^{\mathtt{T}}$ is an E-interpretation on $\Gamma_k^{\mathcal{L}}$ where $k = |q|$. We must show that $I_{q0}^{\mathtt{T}}$ and $I_{q1}^{\mathtt{T}}$, if defined, are E-interpretations on $\Gamma_{k+1}^{\mathcal{L}}$. We need to consider three cases.

- *Case 1*: If α_k is $I_q^{\mathtt{T}}$-reducible, then by T1, $I_{q0}^{\mathtt{T}}(\alpha_k) = I_q^{\mathtt{T}}(\beta)$ for some β such that $\alpha_k \rightarrow_{I_q^{\mathtt{T}}} \beta$ and $q1 \notin \text{Dom}(\mathtt{T})$. Since $I_q^{\mathtt{T}}$ is an E-interpretation on $\Gamma_k^{\mathcal{L}}$, by Theorem 5.10.44, $I_{q0}^{\mathtt{T}}(\alpha_k) = I_q^{\mathtt{T}}(\beta')$ for all β' such that $\alpha_k \rightarrow_{I_q^{\mathtt{T}}} \beta'$. By Theorem 5.10.45, $I_{q0}^{\mathtt{T}}$ is an E-interpretation on $\Gamma_{k+1}^{\mathcal{L}}$.
- *Case 2*: α_k is $I_q^{\mathtt{T}}$-irreducible and of the form $(t = t)$. Then, by T2, $I_{q0}^{\mathtt{T}}(\alpha_k) = \mathbf{T}$ and $q1 \notin \text{Dom}(\mathtt{T})$. By Theorem 5.10.45, $I_{q0}^{\mathtt{T}}$ is an E-interpretation on $\Gamma_{k+1}^{\mathcal{L}}$.
- *Case 3*: α_k is $I_q^{\mathtt{T}}$-irreducible and not of the form $(t = t)$. Then, by Theorem 5.10.45, any extension of $I_q^{\mathtt{T}}$ to $\Gamma_{k+1}^{\mathcal{L}}$ is an E-interpretation on $\Gamma_{k+1}^{\mathcal{L}}$, so $I_{q0}^{\mathtt{T}}$ and $I_{q1}^{\mathtt{T}}$ are E-interpretations.

□

Theorem 5.10.48. *Let $\mathcal{L}$ be a first-order language with equality that contains at least one constant symbol. If $\mathtt{T}_0$ and $\mathtt{T}_1$ are $\mathcal{L}$-semantic trees, then $\mathtt{T}_0 = \mathtt{T}_1$.*

Proof. Let $\mathtt{T}_0$ and $\mathtt{T}_1$ be $\mathcal{L}$-semantic trees. To show that $\mathtt{T}_0 = \mathtt{T}_1$, we prove that for $q \in \{0,1\}^*$, $q \in \mathrm{Dom}(\mathtt{T}_0)$ if and only if $q \in \mathrm{Dom}(\mathtt{T}_1)$, and if $q \in \mathrm{Dom}(\mathtt{T}_0)$, then $I_q^{\mathtt{T}_0} = I_q^{\mathtt{T}_1}$. The proof is by induction on $|q|$.

For the basis step, $q = \lambda$, hence $q \in \mathrm{Dom}(\mathtt{T}_0) \cap \mathrm{Dom}(\mathtt{T}_1)$ and $I_q^{\mathtt{T}_0}$ and $I_q^{\mathtt{T}_1}$ are both the empty function.

For the inductive step, suppose that the result holds for q with $|q| = k$. If $q \notin \mathrm{Dom}(\mathtt{T}_0)$. then, by inductive hypothesis, $q \notin \mathrm{Dom}(\mathtt{T}_1)$ and therefore $q0$ and $q1$ do not belong to $\mathrm{Dom}(\mathtt{T}_0) \cup \mathrm{Dom}(\mathtt{T}_1)$.

If $q \in \mathrm{Dom}(\mathtt{T}_0)$ and $|q| = |\mathrm{GAFORM}_{\mathcal{L}}|$, then $q0, q_1$ do not belong to $\mathrm{Dom}(\mathtt{T}_0) \cup \mathrm{Dom}(\mathtt{T}_1)$.

Suppose $q \in \mathrm{Dom}(\mathtt{T}_0)$ and $|q| < |\mathrm{GAFORM}_{\mathcal{L}}|$. Then, $q \in \mathrm{Dom}(\mathtt{T}_1)$ and $I_q^{\mathtt{T}_0} = I_q^{\mathtt{T}_1}$. We denote this common value by I_q. We need to consider the following cases:

- *Case 1*: α_k is I_q-reducible. We have $q0 \in \mathrm{Dom}(\mathtt{T}_0) \cap \mathrm{Dom}(\mathtt{T}_1)$ and $q1 \notin \mathrm{Dom}(\mathtt{T}_0) \cup \mathrm{Dom}(\mathtt{T}_1)$. For some β_0, β_1 with $\alpha_k \rightarrow_{I_q} \beta_0$, $\alpha_k \rightarrow_{I_q} \beta_1$, we have $I_{q0}^{\mathtt{T}_0}(\alpha_k) = I_q(\beta_0)$ and $I_{q0}^{\mathtt{T}_1}(\alpha_k) = I_q(\beta_1)$. By Theorem, 5.10.47, I_q is an E-interpretation on $\Gamma_k^{\mathcal{L}}$, so by Theorem 5.10.44, $I_q(\beta_0) = I_q(\beta_1)$, hence $I_{q0}^{\mathtt{T}_0}(\alpha_k) = I_{q0}^{\mathtt{T}_1}(\alpha_k)$, which implies $I_{q0}^{\mathtt{T}_0} = I_{q0}^{\mathtt{T}_1}$.
- *Case 2*: α_k is I_q-irreducible and of the form $(t = t)$. Now we have $q0 \in \mathrm{Dom}(\mathtt{T}_0) \cap \mathrm{Dom}(\mathtt{T}_1)$ and $q1 \notin \mathrm{Dom}(\mathtt{T}_0) \cup \mathrm{Dom}(\mathtt{T}_1)$ and $I_{q0}^{\mathtt{T}_0}(\alpha_k) = \mathbf{T} = I_{q0}^{\mathtt{T}_1}(\alpha_k)$, so $I_{q0}^{\mathtt{T}_0} = I_{q0}^{\mathtt{T}_1}$.
- *Case 3*: α_k is I_q-irreducible and not of the form $(t = t)$. In this case, we have $q0, q1 \in \mathrm{Dom}(\mathtt{T}_0) \cap \mathrm{Dom}(\mathtt{T}_1)$, $I_{q0}^{\mathtt{T}_0}(\alpha_k) = \mathbf{F} = I_{q0}^{\mathtt{T}_1}(\alpha_k)$, $I_{q1}^{\mathtt{T}_0}(\alpha_k) = \mathbf{T} = I_{q1}^{\mathtt{T}_1}(\alpha_k)$, so $I_{q0}^{\mathtt{T}_0} = I_{q0}^{\mathtt{T}_1}$ and $I_{q1}^{\mathtt{T}_0} = I_{q1}^{\mathtt{T}_1}$.

$\square$

The uniqueness result of the previous theorem applies once an ordering of $\mathrm{GWORD}_{\mathcal{L}}$ satisfying O1 to O6 is fixed. We write $\mathtt{S}_{\mathcal{L}}$ for the unique $\mathcal{L}$-semantic tree.

Theorem 5.10.49. *Let $\mathcal{L}$ be a first-order language with equality that contains at least one constant symbol. The following statements hold:*

a. *If $\mathtt{B}$ is a branch of $\mathtt{S}_{\mathcal{L}}$, then $I_{\mathtt{B}}^{\mathtt{S}_{\mathcal{L}}}$ is an $\mathcal{L}$-E-interpretation.*
b. *If I is an $\mathcal{L}$-E-interpretation, then there is a branch $\mathtt{B}$ of $\mathtt{S}_{\mathcal{L}}$ with $I_{\mathtt{B}}^{\mathtt{S}_{\mathcal{L}}} = I$.*

Proof. For Part (a), let $\mathtt{B}$ be a branch of $\mathtt{S}_{\mathcal{L}}$. If $\mathtt{B}$ is finite, then $I_{\mathtt{B}}^{\mathtt{S}_{\mathcal{L}}} = I_q^{\mathtt{S}_{\mathcal{L}}}$ for a leaf q of $\mathtt{S}_{\mathcal{L}}$. Thus, by Theorem 5.10.47, $I_{\mathtt{B}}^{\mathtt{S}_{\mathcal{L}}}$ is an E-interpretation on $\Gamma_{|q|}^{\mathcal{L}} = \text{GAFORM}_{\mathcal{L}}$.

If $\mathtt{B}$ is infinite, then for each $k \geq 0$, $I_{\mathtt{B}}^{\mathtt{S}_{\mathcal{L}}} \restriction \Gamma_k^{\mathcal{L}} = I_q^{\mathtt{S}_{\mathcal{L}}}$ where q is the element of $\mathtt{B}$ of length k. By Theorem 5.10.47, $I_q^{\mathtt{S}_{\mathcal{L}}}$ is an E-interpretation on $\Gamma_k^{\mathcal{L}}$ so by Theorem 5.10.36, $I_{\mathtt{B}}^{\mathtt{S}_{\mathcal{L}}}$ is an $\mathcal{L}$-E-interpretation.

For Part (b), let I be an $\mathcal{L}$-E-interpretation. We define the strings q_k for $0 \leq k \leq |\text{GAFORM}_{\mathcal{L}}|$ such that $|q_k| = k$ and $\mathtt{B} = \{q_0, q_1, \ldots\}$ is a branch of $\mathtt{S}_{\mathcal{L}}$ with $I_{q_k}^{\mathtt{S}_{\mathcal{L}}} = I \restriction \Gamma_k^{\mathcal{L}}$ for all k. This implies that $I_{\mathtt{B}}^{\mathtt{S}_{\mathcal{L}}} = I$.

We set $q_0 = \lambda$. Given k with $0 \leq k < |\text{GAFORM}_{\mathcal{L}}|$ and q_k with $|q_k| = k$, and $I_{q_k}^{\mathtt{S}_{\mathcal{L}}} = I \restriction \Gamma_k^{\mathcal{L}}$, by Theorem 5.10.45 one of the following cases holds:

- *Case 1*: α_k is $I_{q_k}^{\mathtt{S}_{\mathcal{L}}}$-reducible and for all β such that $\alpha_k \rightarrow_{I_{q_k}^{\mathtt{S}_{\mathcal{L}}}} \beta$, $I(\alpha_k) = I(\beta)$. By Part (T1) of Definition 5.10.46, for some β such that $\alpha_k \rightarrow_{I_{q_k}^{\mathtt{S}_{\mathcal{L}}}} \beta$, we have:
$$I_{q_k 0}^{\mathtt{S}_{\mathcal{L}}}(\alpha_k) = I_{q_k}^{\mathtt{S}_{\mathcal{L}}}(\beta) = I(\beta) = I(\alpha_k).$$
Setting $q_{k+1} = q_k 0$, we have $I_{q_{k+1}}^{\mathtt{S}_{\mathcal{L}}} = I \restriction \Gamma_{k+1}^{\mathcal{L}}$.
- *Case 2*: α_k is $I_{q_k}^{\mathtt{S}_{\mathcal{L}}}$-irreducible and of the form $(t = t)$ and $I(\alpha_k) = \mathbf{T}$. By Part (T2) of Definition 5.10.46, $I_{q0}^{\mathtt{S}_{\mathcal{L}}}(\alpha_k) = \mathbf{T}$ so setting $q_{k+1} = q_k 0$ we have $I_{q_{k+1}}^{\mathtt{S}_{\mathcal{L}}} = I \restriction \Gamma_{k+1}^{\mathcal{L}}$.
- *Case 3*: α_k is $I_{q_k}^{\mathtt{S}_{\mathcal{L}}}$-irreducible and not of the form $(t = t)$. If $I(\alpha_k) = \mathbf{F}$ then by Part (T3) of the same definition, $I_{q_k 0}^{\mathtt{S}_{\mathcal{L}}}(\alpha_k) = \mathbf{F}$, so we let $q_{k+1} = q_k 0$. If $I(\alpha_k) = \mathbf{T}$ then by the same Part (T3), $I_{q_k 1}^{\mathtt{S}_{\mathcal{L}}}(\alpha_k) = \mathbf{T}$ so we let $q_{k+1} = q_k 1$.

□

Definition 5.10.50. Let $\mathcal{L}$ be a first-order language with equality that contains at least one constant symbol and let $\mathcal{C}$ be a set of ground $\mathcal{L}$-clauses. A node q of $\mathtt{S}_{\mathcal{L}}$ is a *failure node* for $\mathcal{C}$ if there is a clause $C \in \mathcal{C}$ with $I_q^{\mathtt{S}_{\mathcal{L}}}(C) = \mathbf{F}$. ▯

Theorem 5.10.51. *Let $\mathcal{L}$ be a first-order language with equality that contains at least one constant symbol and let $\mathcal{C}$ be a set of ground $\mathcal{L}$-clauses. Then, $\mathcal{C}$ has an $\mathcal{L}$-E-model if and only if there is a branch* $\mathtt{B}$ *of* $\mathtt{S}_{\mathcal{L}}$ *such that* $\mathtt{B}$ *does not contain a failure node for $\mathcal{C}$.*

Proof. Suppose that $\mathcal{C}$ has an $\mathcal{L}$-E-model I. By Theorem 5.10.49 there is a branch $\mathtt{B}$ of $\mathtt{S}_{\mathcal{L}}$ with $I_{\mathtt{B}}^{\mathtt{S}_{\mathcal{L}}} = I$. If $\mathtt{B}$ contained a failure node q for $\mathcal{C}$, then for some $C \in \mathcal{C}$, $I(C) = I_{\mathtt{B}}^{\mathtt{S}_{\mathcal{L}}}(C) = I_q^{\mathtt{S}_{\mathcal{L}}}(C) = \mathbf{F}$, which contradicts the fact that I is an $\mathcal{L}$-E-model of $\mathcal{C}$.

Conversely, suppose that $\mathtt{B}$ is a branch of $\mathtt{S}_{\mathcal{L}}$ that does not contain a failure node for $\mathcal{C}$. Then, $I_{\mathtt{B}}^{\mathtt{S}_{\mathcal{L}}}(C) = \mathbf{T}$ for every $C \in \mathcal{C}$. By Theorem 5.10.49, $I_{\mathtt{B}}^{\mathtt{S}_{\mathcal{L}}}$ is an $\mathcal{L}$-E-interpretation, so $\mathcal{C}$ has an $\mathcal{L}$-E-model. □

Definition 5.10.52. Let $\mathcal{L}$ be a first-order language with equality that contains at least one constant symbol and let $\mathcal{C}$ be a set of ground $\mathcal{L}$-clauses. The sublot of $\mathtt{S}_{\mathcal{L}}$ with domain

$$\{q \in \mathrm{Dom}(\mathtt{S}_{\mathcal{L}}) \mid \text{no proper prefix of } q \text{ is a failure node of } \mathcal{C}\}$$

is denoted by $\mathtt{S}^*_{\mathcal{L},\mathcal{C}}$. ▯

Theorem 5.10.53. *Let $\mathcal{L}$ be a first-order language with equality that contains at least one constant symbol and let $\mathcal{C}, \mathcal{C}'$ be two sets of ground $\mathcal{L}$-clauses. If $\mathcal{C} \subseteq \mathcal{C}'$, then $\mathtt{S}^*_{\mathcal{L},\mathcal{C}'}$ is a sublot of $\mathtt{S}^*_{\mathcal{L},\mathcal{C}}$.*

Proof. This result follows directly from the definitions. □

Theorem 5.10.54. *Let $\mathcal{L}$ be a first-order language with equality that contains at least one constant symbol, $\mathcal{C}$ be a set of ground $\mathcal{L}$-clauses, and let q be a node in* $\mathrm{Dom}(\mathtt{S}_{\mathcal{L}})$ *that is not a failure node for $\mathcal{C}$ and $|q| = k$. If $C \in \mathcal{C}$ is such that $I_{qi}^{\mathtt{S}_{\mathcal{L}}}(C) = \mathbf{F}$, the following statements hold:*

a. *if $I_{qi}^{\mathtt{S}_{\mathcal{L}}}(\alpha_k) = \mathbf{T}$, then $(\neg\alpha_k) \in C$ and $I_q^{\mathtt{S}_{\mathcal{L}}}(C - \{(\neg\alpha_k)\}) = \mathbf{F}$;*
b. *if $I_{qi}^{\mathtt{S}_{\mathcal{L}}}(\alpha_k) = \mathbf{F}$, then $\alpha_k \in C$ and $I_q^{\mathtt{S}_{\mathcal{L}}}(C - \{\alpha_k\}) = \mathbf{F}$.*

Proof. Let $C \in \mathcal{C}$ be such that $I_{qi}^{\mathtt{S}_{\mathcal{L}}}(C) = \mathbf{F}$. Since $I_{qi}^{\mathtt{S}_{\mathcal{L}}}(C)$ is defined, for all i such that either $\alpha_i \in C$ or $(\neg\alpha_i) \in C$, we must have $i \leq k$. If $i < k$ for all such i, then $I_q^{\mathtt{S}_{\mathcal{L}}}(C) = I_{qi}^{\mathtt{S}_{\mathcal{L}}}(C) = \mathbf{F}$, contradicting the assumption that q is not a failure node for $\mathcal{C}$. Thus, either $\alpha_k \in C$ or $(\neg\alpha_k) \in C$. If $I_{qi}^{\mathtt{S}_{\mathcal{L}}}(\alpha_k) = \mathbf{T}$, then since $I_{qi}^{\mathtt{S}_{\mathcal{L}}}(C) = \mathbf{F}$, we must have $\alpha_k \notin C$ and $(\neg\alpha_k) \in C$, and $I_q^{\mathtt{S}_{\mathcal{L}}}(C - \{(\neg\alpha_k)\}) = \mathbf{F}$. If $I_{qi}^{\mathtt{S}_{\mathcal{L}}}(\alpha_k) = \mathbf{F}$, then $(\neg\alpha_k) \notin C$, so $\alpha_k \in C$, and $I_q^{\mathtt{S}_{\mathcal{L}}}(C - \{\alpha_k\}) = \mathbf{F}$. □

Theorem 5.10.55. *Let $\mathcal{L}$ be a first-order language with equality that contains at least one constant symbol and let $\mathcal{C}$ be a set of ground $\mathcal{L}$-clauses that has no $\mathcal{L}$-E-model. Then we have:*

a. $\mathtt{S}^*_{\mathcal{L},\mathcal{C}}$ *is a finite lot*;
b. *every leaf of* $\mathtt{S}^*_{\mathcal{L},\mathcal{C}}$ *is a failure node for* $\mathcal{C}$.

Proof. The argument follows the outline of Supplement 92. □

5.10.5 *Completeness of Paramodulation*

Lemma 5.10.56 (Disjoint Paramodulation Lifting). *Let $\mathcal{L}$ be a first-order language with equality, C_0, C_1 be two ground $\mathcal{L}$-clauses and let P be an $\mathcal{L}$-paramodulant of C_0, C_1 obtained by paramodulating an equality $(u = v) \in C_0$ into a term occurrence (t, i) in a literal ℓ in C_1. Suppose that C_0', C_1' are two $\mathcal{L}$-clauses whose sets of variables are disjoint and s_0, s_1 are $\mathcal{L}$-substitutions such that $s_0(C_0') = C_0$ and $s_1(C_1') = C_1$. If there is a literal $\ell_1 \in s_1^{-1}(\ell) \cap C_1$ such that the occurrence (t, i) in ℓ is visible in ℓ_1, then there is simple, full, most general paramodulant P' of C_0' and C_1' and an $\mathcal{L}$-substitution s such that $s(P') = P$.*

Proof. We assume that $t = u$. The case $t = v$ is similar. Since C_0, C_1 are ground clauses, we have

$$P = (C_0 - \{(u = v)\}) \cup (C_1 - \{\ell\}) \cup \{\ell[(u, i) \to v]\}.$$

Since $\mathtt{V}(C_0') \cap V(C_1') = \emptyset$, we can let s' be an $\mathcal{L}$-substitution with $s' \upharpoonright \mathtt{V}(C_0') = s_0 \upharpoonright \mathtt{V}(C_0')$ and $s' \upharpoonright \mathtt{V}(C_1') = s_1 \upharpoonright \mathtt{V}(C_1')$. Let $L = s_0^{-1}(u = v) \cap C_0'$ and $K = s_1^{-1}(\ell) \cap C_1'$. Then L and K are nonempty and

$$s'(L) = s_0(L) = \{(u = v)\}, s'(K) = s_1(K) = \{\ell\},$$

so L and K are unifiable, say s_0^m is a most general unifier of L and s_1^m is a most general unifier of K. By using most general unifiers produced by the unification algorithm, we can assume that $\mathtt{V}(s_0^m(C_0')) \subseteq \mathtt{V}(C_0')$ and $\mathtt{V}(s_1^m(C_1')) \subseteq \mathtt{V}(C_1')$. Since s' unifies both L and K, there are $\mathcal{L}$-substitutions s_0' and s_1' with $s' = s_0' * s_0^m$ and $s' = s_1' * s_1^m$. Since $\mathtt{V}(C_0') \cap \mathtt{V}(C_1') = \emptyset$, we can let $\hat{s}$ be an $\mathcal{L}$-substitution that agrees with s_0' on $\mathtt{V}(C_0')$ and with s_1' on $\mathtt{V}(C_1')$. Since

$$\mathtt{V}(s_0^m(C_0')) \subseteq \mathtt{V}(C_0') \text{ and } \mathtt{V}(s_1^m(C_1')) \subseteq \mathtt{V}(C_1'),$$

we have

$$\begin{gathered}\hat{s} * s_0^m(C_0') = \hat{s}(s_0^m(C_0')) = s_0'(s_0^m(C_0')) = \\ s_0' * s_0^m(C_0') = s'(C_0') = s_0(C_0') = C_0,\end{gathered}$$

and similarly $\hat{s} * s_1^m(C_1') = C_1$.

Let $s_0^m(L) = \{(u' = v')\}$ and $s_1^m(K) = \{\ell'\}$. The occurrence $(t, i) = (u, i)$ in ℓ is visible in $\ell_1 \in s_1^{-1}(\ell) \cap C_1 = K$, and $s_1' * s_1^m(\ell_1) = s_1(\ell_1) = \ell$, so by Theorem 4.3.62, (u, i) is visible in $s_1^m(\ell_1) \in s_1^m(K) = \{\ell'\}$, say (t', i') is an occurrence of a term t' in ℓ' with $\ell' = q_0' t' q_1'$, where $|q_0'| = i'$, $\ell = q_0 u q_1$, where $|q_0| = i$, and

$$\hat{s}(q_0') = s_1'(q_0') = q_0,$$
$$\hat{s}(t') = s_1'(t') = u,$$
$$\hat{s}(q_1') = s_1'(q_1') = q_1.$$

Since

$$\begin{aligned}\hat{s}(u' = v') &= \hat{s}(s_0^m(L) = s_0'(s_0^m(L))\\ &= s_0' * s_0^m(L) = s'(L) = s_0(L) = \{(u = v)\},\end{aligned}$$

we have $\hat{s}(v') = v$ and $\hat{s}(u') = u = \hat{s}(t')$, so $\{u', t'\}$ is unifiable. Let s_{01}^m be a most general unifier of u' and t'. Then,

$$P' = s_{01}^m(s_0^m(C_0' - L) \cup s_1^m(C_1' - K) \cup \{\ell'[(t', i') \to v']\})$$

is a simple most general paramodulant of C_0' and C_1'. Suppose that $s_0^m(C_0' - L) \cap s_0^m(L) \neq \emptyset$, say $s_0^m(\ell_0) = s_0^m(\ell_0')$ with $\ell_0 \in C_0' - L$ and $\ell_0' \in L$. Then,

$$\begin{aligned}s_0(\ell_0) = s'(\ell_0) &= s_0' * s_0^m(\ell_0) = s_0'(s_0^m(\ell_0))\\ &= s_0'(s_0^m(\ell_0')) = s_0(\ell_0') = (u = v)\end{aligned}$$

since $\ell_0' \in L \subseteq s_0^{-1}(u = v)$. Thus, $\ell_0 \in s_0^{-1}((u = v)) \cap C_0' = L$, contradicting hypothesis, so $s_0^m(C_0' - L) \cap s_0^m(L) = \emptyset$. Similarly, $s_1^m(C_1' - K) \cap s_1^m(K) = \emptyset$, and P' is a full $\mathcal{L}$-paramodulant of C_0 and C_1.

Since $\hat{s}$ unifies $\{u', t'\}$, there is an $\mathcal{L}$-substitution s with $\hat{s} = s * s_{01}^m$. We claim that $s(P') = P$ which will finish the proof.

First note that $\ell'[(t',i') \to v'] = q_0'v'q_1'$, so

$$\hat{s}(\ell'[(t',i') \to v'] = \hat{s}(q_0')\hat{s}(v')\hat{s}(q_1') = q_0vq_1 = \ell[(u,i) \to v].$$

Thus we can write:

$$\begin{aligned}
s(P') &= s(s_{01}^m(s_0^m(C_0' - L) \cup s_1^m(C_1' - K) \cup \{\ell'[(t',i') \to v']\}) \\
&= s * s_{01}^m(s_0^m(C_0' - L) \cup s_1^m(C_1' - K) \cup \{\ell'[(t',i') \to v']\}) \\
&= \hat{s}(s_0^m(C_0' - L) \cup s_1^m(C_1' - K) \cup \{\ell'[(t',i') \to v']\}) \\
&= \hat{s} * s_0^m(C_0' - L) \cup \hat{s} * s_1^m(C_1' - K) \cup \{\hat{s}(\ell'[(t',i') \to v'])\} \\
&= s_0(C_0' - L) \cup s_1(C_1' - K) \cup \{\ell[(u,i) \to v]\} \\
&= (s_0(C_0') - s_0(L)) \cup (s_1(C_1') - s_1(K)) \cup \{\ell[(u,i) \to v]\} \\
&\quad \text{(by Equality (5.7))} \\
&= (C_0 - \{(u = v)\}) \cup (C_1 - \{\ell\}) \cup \{\ell[(u,i) \to v]\} = P.
\end{aligned}$$

□

Lemma 5.10.57 (Paramodulation Lifting). *Let $\mathcal{L}$ be a first-order language with equality, C_0, C_1 be two ground $\mathcal{L}$-clauses and let P be an $\mathcal{L}$-paramodulant of C_0, C_1 obtained by paramodulating an equality $(u = v) \in C_0$ into a term occurrence (t,i) in a literal ℓ in C_1. Suppose that C_0', C_1' are two $\mathcal{L}$-clauses and s_0, s_1 are $\mathcal{L}$-substitutions such that $s_0(C_0') = C_0$ and $s_1(C_1') = C_1$. If there is a literal $\ell_1 \in s_1^{-1}(\ell) \cap C_1$ such that the occurrence (t,i) in ℓ is visible in ℓ_1, then there is a full, most general paramodulant P' of C_0' and C_1' and an $\mathcal{L}$-substitution s such that $s(P') = P$.*

Proof. Let (s_0'', s_1'') be a standardization of (C_0', C_1') and let $C_0'' = s_0''(C_0'), C_1'' = s_1''(C_1')$. By the injectivity of s_0'' and s_1'' on $V(C_0'), V(C_1')$, respectively, we may let s_0', s_1' be $\mathcal{L}$-substitutions such that for $x \in V(C_0')$, $s_0'(s_0''(x)) = x$ and for $x \in V(C_1')$, $s_1'(s_1''(x)) = x$. We then have $s_0 * s_0'(C_0'') = s_0(C_0') = C_0$ and $s_1 * s_1'(C_1'') = s_1(C_1') = C_1$ and also $s_1' * s_1'' \restriction V(C_1') = \iota \restriction V(C_1')$, where ι is the identity substitution.

Since the occurrence (t,i) in ℓ is visible in $\ell_1 \in C_1'$, there is an occurrence of a term (t',i') in ℓ_1 such that if $\ell = q_0tq_1$ with $|q_0| = i$ and $\ell_1 = q_0't'q_1'$ with $|q_0'| = i'$, we have $s_1(q_0') = q_0$. Then, $s_1''(\ell_1) = s_1''(q_0')s_1''(t')s_1''(q_1')$ is a literal in C_1'', $(s_1''(t'), |s_1''(q_0')|)$ is an occurrence of a term in $s_1''(\ell_1)$ and $s_1 * s_1'(s_1''(q_0')) = s_1 * (s_1' * s_1''(q_0')) = s_1(q_0') = q_0$.

Also, we have $s_1 * s'_1(s''_1(\ell_1)) = s_1(s'_1 * s''_1(\ell_1)) = s_1(\ell_1) = \ell$. Therefore, $s''_1(\ell_1) \in (s_1 * s'_1)^{-1}(\ell)$ and (t, i) is visible in $s''_1(\ell_1)$.

Since $V(C''_0) \cap V(C''_1) = \emptyset$, by Lemma 5.10.56, there is a simple, most general paramodulant P' of C''_0 and C''_1 and an $\mathcal{L}$-substitution s such that $s(P') = P$. Since (C''_0, C''_1) is obtained by a standardization of (C'_0, C'_1) and P' is a simple paramodulant of C''_0 and C''_1, P' is a paramodulant of C'_0 and C'_1. □

Definition 5.10.58. Let $\mathcal{L}$ be a first-order language with equality containing at least one constant symbol and let I be an $\mathcal{H}$-interpretation on a left segment Γ of $\mathrm{GAFORM}_{\mathcal{L}}$. The binary relation $\rightsquigarrow_I$ on the set of ground $\mathcal{L}$-substitutions is defined by $s \rightsquigarrow_I s'$ if there is a variable x such that $s(x) \rightarrow_I s'(x)$ and $s'(y) = s(y)$ for all $y \neq x$. We denote the reflexive and transitive closure of $\rightsquigarrow_I$ by $\rightsquigarrow_I^*$. ▯

Theorem 5.10.59. *Let $\mathcal{L}$ be a first-order language with equality containing at least one constant symbol, let I be an E-interpretation on a left segment Γ of* $\mathrm{GAFORM}_{\mathcal{L}}$*, and let s, s' be ground $\mathcal{L}$-substitutions with $s \rightsquigarrow_I^* s'$. If C is an $\mathcal{L}$-clause such that $I(s(C))$ is defined* (*that is, every atomic formula that appears in $s(C)$ belongs to* Γ)*, then $I(s'(C))$ is defined and $I(s(C)) = I(s'(C))$.*

Proof. It suffices to show the result when $s \rightsquigarrow_I s'$, so suppose that $s(x) \rightarrow_I s'(x)$ and $s'(y) = s(y)$ for $y \neq x$. Let α be an atomic formula that appears in C. Since $s(x) \rightarrow_I s'(x)$, we have $s'(x) \preceq_{\mathcal{L}} s(x)$ and since $s'(\alpha)$ is obtained from $s(\alpha)$ by replacing zero or more occurrences of $s(x)$ by $s'(x)$, by O4, $s'(\alpha) \preceq_{\mathcal{L}} s(\alpha)$, so $s'(\alpha) \in \Gamma$ and $I(s'(\alpha))$ is defined. We will show that $I(s'(\alpha)) = I(s(\alpha))$. Since this is true for all atomic formulas that appear in C, this will establish that $I(s'(C))$ is defined and equal to $I(s(C))$.

If x does not occur in α, then $s'(\alpha) = s(\alpha)$ and the result is immediate, so suppose that x does occur in α. Let $s(x) = t$ and $s'(x) = t'$ and say that $t \rightarrow_I t'$ via $t_0 = u_0$. We consider two cases based on the relationship between t_0 and t.

- *Case 1*: $t_0 = t$. Then, $t' = u_0$. By F2, $I(t = t') = I(t_0 = u_0) = \mathbf{T}$. Since $s'(\alpha)$ is obtained from $s(\alpha)$ by replacing one or more occurrences of t by t', $I(t = t') = \mathbf{T}$ and I is an E-interpretation on Γ, $I(s'(\alpha)) = I(s(\alpha))$ by E3.
- *Case 2*: $t_0 \neq t$. We consider three subcases based on the form of $s(\alpha)$.

- *Case 2.1*: $s(\alpha)$ does not have the form $(t = u')$ or $(u' = t)$ with $u' \preceq_{\mathcal{L}} t'$. Then, since t occurs in $s(\alpha)$ and $t' \prec_{\mathcal{L}} t$, by O6, $(t' = t) \prec_{\mathcal{L}} s(\alpha)$, so $(t' = t) \in \Gamma$. By O5, $(t = t') \prec_{\mathcal{L}} (t' = t)$, so $(t = t') \in \Gamma$.
  Since $t \rightarrow_I t'$ via $(t_0 = u_0)$ and t_0 is a proper subtern of t, the boxed condition of Theorem 5.10.40 does not hold when α in that condition is replaced by $(t' = t)$ and therefore, $(t' = t) \rightarrow_I (t' = t')$. By E2, Theorem 5.10.42 and E1, $I(t = t') = I(t' = t) = I(t' = t') = \mathbf{T}$. As in Case 1, we obtain $I(s'(\alpha)) = I(s(\alpha))$ by E3.
- *Case 2.2*: $s(\alpha) = (t = u')$ with $u' \preceq_{\mathcal{L}} t'$. Since $u' \preceq_{\mathcal{L}} t' \prec_{\mathcal{L}} t$, t cannot occur in u', so $s'(\alpha) = (t' = u')$. Since $t \rightarrow_I t'$ via $(t_0 = u_0)$ and t_0 is a proper subterm of t, the boxed condition of Theorem 5.10.40 does not apply when α is replaced by $(t = u')$ and thus $(t = u') \rightarrow_I (t' = u')$. By Theorem 5.10.42, $I(s(\alpha)) = I(t = u') = I(t' = u') = I(s'(\alpha))$.
- *Case 2.3*: $s(\alpha) = (u' = t)$ with $u' \preceq_{\mathcal{L}} t'$. This case is handled similarly to Case 2.2.

$\square$

Definition 5.10.60. Let $\mathcal{L}$ be a first-order language with equality containing at least one constant symbol, let I be an E-interpretation on a left segment Γ of GAFORM$_{\mathcal{L}}$, and C be an $\mathcal{L}$-clause. An $\mathcal{L}$-substitution s is *I-irreducible on C* if $s(x)$ is an I-irreducible term for all $x \in \mathtt{V}(C)$. ▯

Corollary 5.10.61. *Let $\mathcal{L}$ be a first-order language with equality containing at least one constant symbol, let I be an E-interpretation on a left segment Γ of* GAFORM$_{\mathcal{L}}$*, s be a ground $\mathcal{L}$-substitution and C be an $\mathcal{L}$-clause such that $I(s(C))$ is defined. Then, there is a ground $\mathcal{L}$-substitution s' such that s' is I-irreducible on C, $I(s'(C))$ is defined and $I(s'(C)) = I(s(C))$.*

Proof. Since $\rightarrow_I$ is well-founded, there is a ground $\mathcal{L}$-substitution s' such that $s \rightsquigarrow_I^* s'$ and s' is I-irreducible on C. By Theorem 5.10.59, $I(s'(C))$ is defined and $I(s'(C)) = I(s(C))$. $\square$

Definition 5.10.62. Let $\mathcal{L}$ be a first-order language with equality that contains at least one constant symbol and let $\mathcal{C}$ be a set of ground $\mathcal{L}$-clauses. A node q of $\mathtt{S}^*_{\mathcal{L},\mathcal{C}}$ is a *resolution inference node* if the following conditions are satisfied:

(1) q has depth 1 in $\mathtt{S}^*_{\mathcal{L},\mathcal{C}}$;
(2) $\alpha_{|q|}$ is $I_q^{\mathtt{S}^*_{\mathcal{L},\mathcal{C}}}$-irreducible.

A node q of $\mathtt{S}^*_{\mathcal{L},\mathcal{C}}$ is a *paramodulation inference node* if the following conditions are satisfied:

(1) q has depth 1 in $\mathtt{S}^*_{\mathcal{L},\mathcal{C}}$;
(2) $\alpha_{|q|}$ is $I_q^{\mathtt{S}^*_{\mathcal{L},\mathcal{C}}}$-reducible;
(3) if $r1$ is a prefix of q, then $r0$ is a failure node for $\mathcal{C}$.

∎

Theorem 5.10.63. *Let $\mathcal{L}$ be a first-order language with equality that contains at least one constant symbol and let $\mathcal{C}$ be a set of ground $\mathcal{L}$-clauses. If $\mathcal{C}$ has no $\mathcal{L}$-E-model and $\Box \notin \mathcal{C}$, then $\mathtt{S}^*_{\mathcal{L},\mathcal{C}}$ contains either a resolution inference node or a paramodulation inference node.*

Proof. Suppose that $\mathtt{S}^*_{\mathcal{L},\mathcal{C}}$ does not contain a resolution inference node. This means that for all nodes q with depth 1 in $\mathtt{S}^*_{\mathcal{L},\mathcal{C}}$, $\alpha_{|q|}$ is $I_q^{\mathtt{S}^*_{\mathcal{L},\mathcal{C}}}$-reducible. We define a path $(q_0, q_1, \ldots)$ of $\mathtt{S}^*_{\mathcal{L},\mathcal{C}}$ ending in a paramodulation inference node q. Since $\Box \notin \mathcal{C}$, λ is an interior node of $\mathtt{S}^*_{\mathcal{L},\mathcal{C}}$. Define $q_0 = \lambda$. If q_i is defined and has depth 1, halt with $q = q_i$. If q_i is defined and has depth greater than 1, define q_{i+1} as follows. If q_i has one immediate descendant in $\mathtt{S}^*_{\mathcal{L},\mathcal{C}}$, define $q_{i+1} = q_i0$. If q_i has two immediate descendants, define

$$q_{i+1} = \begin{cases} q_i0 & \text{if } q_i0 \text{ is not a leaf of } \mathtt{S}^*_{\mathcal{L},\mathcal{C}}, \\ q_i1 & \text{otherwise.} \end{cases}$$

By Theorem 5.10.55, $\mathtt{S}^*_{\mathcal{L},\mathcal{C}}$ is finite, so the construction halts with a node q of depth 1. Since there is no resolution inference node, $\alpha_{|q|}$ is $I_q^{\mathtt{S}^*_{\mathcal{L},\mathcal{C}}}$-reducible and by construction, if $r1$ is a prefix of q then $r0$ is a leaf of $\mathtt{S}^*_{\mathcal{L},\mathcal{C}}$ and hence, by Theorem 5.10.55, $r0$ is failure node for $\mathcal{C}$. Thus, q is a paramodulation inference node of $\mathtt{S}^*_{\mathcal{L},\mathcal{C}}$. $\Box$

For a set of $\mathcal{L}$-clauses $\mathcal{C}$, we use the notation $\mathtt{SG}_{\mathcal{L},\mathcal{C}}$ for $\mathtt{S}^*_{\mathcal{L},\mathrm{GINST}_{\mathcal{L}}(\mathcal{C})}$.

Theorem 5.10.64. *Let $\mathcal{L}$ be a first-order language with equality that contains at least one constant symbol and let $\mathcal{C}$ be a set of $\mathcal{L}$-clauses that has no model. If $\mathtt{SG}_{\mathcal{L},\mathcal{C}}$ contains a resolution inference node q,* *then there is a full mgu resolvent R of two clauses in $\mathcal{C} \cup \{(x = x)\}$ such that*

$$\mathrm{Dom}(\mathtt{SG}_{\mathcal{L},\mathcal{C}\cup\{R\}}) \subset \mathrm{Dom}(\mathtt{SG}_{\mathcal{L},\mathcal{C}}).$$

Proof. Since $\mathcal{C}$ has no model, by Theorem 5.10.35, $\mathrm{GINST}_{\mathcal{L}}(\mathcal{C})$ has no $\mathcal{L}$-E-model. By Theorem 5.10.55, every leaf of $\mathtt{SG}_{\mathcal{L},\mathcal{C}}$ is a failure node for $\mathrm{GINST}_{\mathcal{L}}(\mathcal{C})$. We consider two cases depending on how many immediate descendants the resolution inference node q has. We let $k = |q|$.

- *Case 1*: q has two immediate descendants in $\mathtt{SG}_{\mathcal{L},\mathcal{C}}$. Since $q0$ and $q1$ are leaves of $\mathtt{SG}_{\mathcal{L},\mathcal{C}}$, they are failure nodes for $\mathrm{GINST}_{\mathcal{L}}(\mathcal{C})$, so, by Theorem 5.10.54, there are clauses $C_{q0}, C_{q1} \in \mathcal{C}$ and ground $\mathcal{L}$-substitutions s_{q0}, s_{q1} such that $\alpha_k \in s_{q0}(C_{q0})$ and $(\neg\alpha_k) \in s_{q1}(C_{q1})$ and

$$I_q^{\mathtt{S}_{\mathcal{L}}}((s_{q0}(C_{q0}) - \{\alpha_k\}) \cup (s_{q1}(C_{q1}) - \{(\neg\alpha_k)\})) = \mathbf{F}.$$

  Letting $R' = (s_{q0}(C_{q0}) - \{\alpha_k\}) \cup (s_{q1}(C_{q1}) - \{(\neg\alpha_k)\})$, we have that R' is a resolvent of $s_{q0}(C_{q0})$ and $s_{q1}(C_{q1})$ and $I_q^{\mathtt{S}_{\mathcal{L}}}(R') = \mathbf{F}$. By Lemma 5.8.67, there is a full, most general resolvent R of C_{q0} and C_{q1} and a ground $\mathcal{L}$-substitution s_q with $s_q(R) = R'$. Since q is a failure node for $\mathrm{GINST}_{\mathcal{L}}(\mathcal{C} \cup \{R\})$, $q0, q1 \notin \mathrm{Dom}(\mathtt{SG}_{\mathcal{L},\mathcal{C}\cup\{R\}})$. Together with Theorem 5.10.53, this implies $\mathrm{Dom}(\mathtt{SG}_{\mathcal{L},\mathcal{C}\cup\{R\}}) \subset \mathrm{Dom}(\mathtt{SG}_{\mathcal{L},\mathcal{C}})$.
- *Case 2*: q has one immediate descendant in $\mathtt{SG}_{\mathcal{L},\mathcal{C}}$. Since α_k is $I_q^{\mathtt{SG}_{\mathcal{L},\mathcal{C}}}$-irreducible but q has only one immediate descendant in $\mathtt{SG}_{\mathcal{L},\mathcal{C}}$, α_k must be $(t = t)$ for some ground $\mathcal{L}$-term t and $I_{q0}^{\mathtt{SG}_{\mathcal{L},\mathcal{C}}}(\alpha_k) = \mathbf{T}$. By Theorem 5.10.54, there is clause $C_{q0} \in \mathcal{C}$ and a ground $\mathcal{L}$-substitution s_{q0} such that $(\neg(t = t)) \in s_{q0}(C_{q0})$ and

$$I_q^{\mathtt{S}_{\mathcal{L}}}(s_{q0}(C_{q0}) - \{(\neg(t = t))\}) = \mathbf{F}.$$

  The clause $R' = s_{q0}(C_{q0}) - \{(\neg(t = t))\}$ is a resolvent of $s_{q0}(C_{q0}))$ and $\{(t = t)\}$. Since $\{(t = t)\}$ is a ground instance of $\{(x = x)\}$, by Lemma 5.8.67, there is full most general resolvent R of C_{q0} and $\{(x = x)\}$ and a ground $\mathcal{L}$-substitution s_q with $s_q(R) = R'$. Since $I_q^{\mathtt{S}_{\mathcal{L}}}(R) = \mathbf{F}$, $q0 \notin \mathrm{Dom}(\mathtt{SG}_{\mathcal{L},\mathcal{C}\cup\{R\}})$, so $\mathrm{Dom}(\mathtt{SG}_{\mathcal{L},\mathcal{C}\cup\{R\}}) \subset \mathrm{Dom}(\mathtt{SG}_{\mathcal{L},\mathcal{C}})$.

□

Theorem 5.10.65. *Let $\mathcal{L}$ be a first-order language with equality that contains at least one constant symbol and let $\mathcal{C}$ be a set of $\mathcal{L}$-clauses that has no model. If $\mathtt{SG}_{\mathcal{L},\mathcal{C}}$ contains a paramodulation inference node, then there is a full, most general paramodulant P of two clauses in $\mathcal{C} \cup \{(x = x)\}$ such that*

$$\mathrm{Dom}(\mathtt{SG}_{\mathcal{L},\mathcal{C}\cup\{P\}}) \subset \mathrm{Dom}(\mathtt{SG}_{\mathcal{L},\mathcal{C}}).$$

Proof. Since $\mathcal{C}$ has no model, by Theorem 5.10.35, $\mathrm{GINST}_{\mathcal{L}}(\mathcal{C})$ has no $\mathcal{L}$-E-model. By Theorem 5.10.55, every leaf of $\mathtt{SG}_{\mathcal{L},\mathcal{C}}$ is a failure node for $\mathrm{GINST}_{\mathcal{L}}(\mathcal{C})$. Let q be a paramodulation inference node of $\mathtt{SG}_{\mathcal{L},\mathcal{C}}$ and let $k = |q|$. By definition, α_k is $I_q^{\mathtt{S}_{\mathcal{L}}}$-reducible because $I_q^{\mathtt{SG}_{\mathcal{L},\mathcal{C}}} = I_q^{\mathtt{S}_{\mathcal{L}}}$, so q has only one immediate descendant $q0$ in $\mathtt{SG}_{\mathcal{L},\mathcal{C}}$. Since $q0$ is a failure node for $\mathrm{GINST}_{\mathcal{L}}(\mathcal{C})$, there is a clause $C_{q0} \in \mathcal{C}$ and a ground $\mathcal{L}$-substitution s_{q0} with $I_{q0}^{\mathtt{S}_{\mathcal{L}}}(s_{q0}(C_{q0})) = \mathbf{F}$. By Corollary 5.10.61, we may assume that s_{q0} is $I_{q0}^{\mathtt{S}_{\mathcal{L}}}$-irreducible on C_{q0}. Let

$$\ell = \begin{cases} \alpha_k & \text{if } I_{q0}^{\mathtt{S}_{\mathcal{L}}}(\alpha_k) = \mathbf{F}, \\ (\neg\alpha_k) & \text{if } I_{q0}^{\mathtt{S}_{\mathcal{L}}}(\alpha_k) = \mathbf{T}. \end{cases}$$

By Theorem 5.10.54, $\ell \in s_{q0}(C_{q0})$ and $I_q^{\mathtt{S}_{\mathcal{L}}}(s_{q0}(C_{q0}) - \{\ell\}) = \mathbf{F}$. We consider two cases depending on how α_k is $I_q^{\mathtt{S}_{\mathcal{L}}}$-reducible.

- *Case 1*: α_k is $I_q^{\mathtt{S}_{\mathcal{L}}}$-reducible by F2.
- *Case 2*: α_k is $I_q^{\mathtt{S}_{\mathcal{L}}}$-reducible by F1 but not by F2.

In Case 1, let ℓ' be any literal in C_{q0} with $s_{q0}(\ell') = \ell$ and define α' to be the atomic $\mathcal{L}$-formula appearing in ℓ', so $s_{q0}(\alpha') = \alpha_k$. Let α_k be $I_q^{\mathtt{S}_{\mathcal{L}}}$-reducible via $(t = u)$. Then, we have:

∗ there is an occurrence (t, i) in α_k;
∗ $u \prec_{\mathcal{L}} t$;
∗ $(t = u) \prec_{\mathcal{L}} \alpha_k$;
∗ $I_q^{\mathtt{S}_{\mathcal{L}}}(t = u) = \mathbf{T}$;
∗ $(t = u)$ is $I_q^{\mathtt{S}_{\mathcal{L}}}$-irreducible.

Using the notation of Theorem 4.3.64, let

$$\alpha' = q_0 y_0 q_1 y_1 \cdots q_{n-1} y_{n-1} q_n$$

and $\alpha_k = s_{q0}(\alpha') = q_0 u_0 q_1 u_1 \cdots q_{n-1} u_{n-1} q_n$, where $s_{q0}(y_i) = u_i$ for $0 \leq i \leq n-1$. If (t, i) begins in one of the u_js, say u_{k_0}, then, since by

Exercise 53 of Chapter 1, no proper prefix of an $\mathcal{L}$-term is a suffix of an $\mathcal{L}$-term, (t, i) ends in u_{k_0}. Thus, by O5 and O6,

$$(t = u) \prec_{\mathcal{L}} (u = t) \prec_{\mathcal{L}} u_{k_0}.$$

Since $I_{q0}^{\mathtt{S}_{\mathcal{L}}}$ extends $I_{q}^{\mathtt{S}_{\mathcal{L}}}$, we have

$$I_{q0}^{\mathtt{S}_{\mathcal{L}}}(t = u) = I_{q}^{\mathtt{S}_{\mathcal{L}}}(t = u) = \mathbf{T},$$

and by Part (a) of Theorem 5.10.43 the $I_{q}^{\mathtt{S}_{\mathcal{L}}}$-irreducibility of $(t = u)$ implies that $(t = u)$ is $I_{q0}^{\mathtt{S}_{\mathcal{L}}}$-irreducible. It follows that u_{k_0} is $I_{q0}^{\mathtt{S}_{\mathcal{L}}}$-reducible by F2 via $(t = u)$. This contradicts the assumption that s_{q0} is $I_{q0}^{\mathtt{S}_{\mathcal{L}}}$-irreducible on C_{q0}. Thus, the occurrence (t, i) in α_k starts in one of the q_js, so by Theorem 4.3.64, (t, i) is visible in α'.

It follows that if (t, i') is the occurrence in ℓ corresponding to the occurrence (t, i) in α_k (that is, either (t, i) or $(t, i+2)$), then (t, i') is visible in any literal $\ell' \in s_{q0}^{-1}(\ell)$.

Since $(t = u) \prec_{\mathcal{L}} \alpha_k$, we have $(t = u) = \alpha_j$ for some $j < k$. Let r be the prefix of q of length j. Since $(t = u)$ is $I_{q}^{\mathtt{S}_{\mathcal{L}}}$-irreducible, by Theorem 5.10.43, $(t = u)$ is $I_{r}^{\mathtt{S}_{\mathcal{L}}}$-irreducible, so r has two immediate descendants in $\mathtt{S}_{\mathcal{L}}$ with $I_{r0}^{\mathtt{S}_{\mathcal{L}}}(t = u) = \mathbf{F}$ and $I_{r1}^{\mathtt{S}_{\mathcal{L}}}(t = u) = \mathbf{T}$. Since $I_{q}^{\mathtt{S}_{\mathcal{L}}}(t = u) = \mathbf{T}$, $r1$ is a prefix of q and hence, since q is a paramodulation inference node of $\mathtt{SG}_{\mathcal{L},\mathcal{C}}$, $r0$ is a failure node for $\mathrm{GINST}_{\mathcal{L}}(\mathcal{C})$. This means that there is a clause $C_{r0} \in \mathcal{C}$ and a ground $\mathcal{L}$-substitution s_{r0} such that $I_{r0}^{\mathtt{S}_{\mathcal{L}}}(s_{r0}(C_{r0})) = \mathbf{F}$. By Theorem 5.10.54, $(t = u) \in s_{r0}(C_{r0})$ and $I_{r}^{\mathtt{S}_{\mathcal{L}}}(s_{r0}(C_{r0}) - \{(t = u)\}) = \mathbf{F}$.

Thus,

$$P' = (s_{r0}(C_{r0}) - \{(t = u)\}) \cup (s_{q0}(C_{q0}) - \{\ell\}) \cup \{\ell[(t, i') \to u]\}$$

is a paramodulant of $s_{r0}(C_{r0})$ and $s_{q0}(C_{q0})$. Since (t, i') is visible in any $\ell' \in s_{q0}^{-1}(\ell)$, by the Paramodulation Lifting Lemma (Lemma 5.10.57), there is a full, most general paramodulant P of C_{r0} and C_{q0} and a ground $\mathcal{L}$-substitution s with $s(P) = P'$.

We claim that $I_{q}^{\mathtt{S}_{\mathcal{L}}}(P') = \mathbf{F}$. We prove this by showing in turn that $I_{q}^{\mathtt{S}_{\mathcal{L}}}$ falsifies each of the three sets of literals whose union is P'.

First, we have

$$I_{q}^{\mathtt{S}_{\mathcal{L}}}(s_{r0}(C_{r0}) - \{(t = u)\}) = I_{r}^{\mathtt{S}_{\mathcal{L}}}(s_{r0}(C_{r0}) - \{(t = u)\}) = \mathbf{F}.$$

Second, $I_{q}^{\mathtt{S}_{\mathcal{L}}}(s_{q0}(C_{q0}) - \{\ell\}) = \mathbf{F}$ has already been established.

Finally, to show that $I_q^{\mathtt{S}_\mathcal{L}}(\ell[(t,i') \to u]) = \mathbf{F}$, we begin by noting that $\alpha_k[(t,i) \to u] \prec_\mathcal{L} \alpha_k$ by O4, so $I_q^{\mathtt{S}_\mathcal{L}}(\alpha_k[(t,i) \to u])$ is defined and equal to $I_{q0}^{\mathtt{S}_\mathcal{L}}(\alpha_k[(t,i) \to u])$. By Part (a) of Theorem 5.10.43, since $(t = u) \in \Gamma_k^\mathcal{L}$ and is $I_q^{\mathtt{S}_\mathcal{L}}$-irreducible, $(t = u)$ is $I_{q0}^{\mathtt{S}_\mathcal{L}}$-irreducible and we also have $I_{q0}^{\mathtt{S}_\mathcal{L}}(t = u) = I_q^{\mathtt{S}_\mathcal{L}}(t = u) = \mathbf{T}$. Thus, $\alpha_k \to_{I_{q0}^{\mathtt{S}_\mathcal{L}}} \alpha_k[(t,i) \to u]$ and by Theorem 5.10.42, $I_{q0}^{\mathtt{S}_\mathcal{L}}(\alpha_k) = I_{q0}^{\mathtt{S}_\mathcal{L}}(\alpha_k[(t,i) \to u])$, so

$$I_q^{\mathtt{S}_\mathcal{L}}(\alpha_k[(t,i) \to u]) = I_{q0}^{\mathtt{S}_\mathcal{L}}(\alpha_k).$$

It follows that $I_q^{\mathtt{S}_\mathcal{L}}(\ell[(t,i') \to u]) = I_{q0}^{\mathtt{S}_\mathcal{L}}(\ell) = \mathbf{F}$, as desired.

Thus, q is a failure node for $\mathrm{GINST}_\mathcal{L}(\mathcal{C} \cup \{P\})$, so $q0 \notin \mathrm{Dom}(\mathtt{SG}_{\mathcal{L},\mathcal{C}\cup\{P\}})$ and $\mathrm{Dom}(\mathtt{SG}_{\mathcal{L},\mathcal{C}\cup\{P\}}) \subset \mathrm{Dom}(\mathtt{SG}_{\mathcal{L},\mathcal{C}})$.

Now we tackle Case 2 which occurs when α_k is $I_q^{\mathtt{S}_\mathcal{L}}$-reducible by F1 but not by F2. Then, $\alpha_k = (u = t)$ for some ground $\mathcal{L}$-terms u and t with $u \prec_\mathcal{L} t$ and $\alpha_{k-1} = (t = u)$ is $I_q^{\mathtt{S}_\mathcal{L}}$-irreducible. Suppose that $(\neg(u = t)) \in s_{q0}(C_{q0})$. Then, because $I_{q0}^{\mathtt{S}_\mathcal{L}}(s_{q0}(C_{q0})) = \mathbf{F}$, $I_{q0}^{\mathtt{S}_\mathcal{L}}(u = t) = \mathbf{T}$, and since $I_{q0}^{\mathtt{S}_\mathcal{L}}$ is an E-interpretation on $\Gamma_{k+1}^\mathcal{L}$, $I_q^{\mathtt{S}_\mathcal{L}}(t = u) = I_{q0}^{\mathtt{S}_\mathcal{L}}(t = u) = I_{q0}^{\mathtt{S}_\mathcal{L}}(u = t) = \mathbf{T}$. The fact that $(t = u)$ is $I_q^{\mathtt{S}_\mathcal{L}}$-irreducible and $(t = u) \prec_\mathcal{L} (u = t)$ implies that $\alpha_k = (u = t)$ is $I_q^{\mathtt{S}_\mathcal{L}}$-reducible by F2 via $(t = u)$ contradicting the case assumption. Thus, $(u = t) \in s_{q0}(C_{q0})$, $I_q^{\mathtt{S}_\mathcal{L}}(s_{q0}(C_{q0}) - \{(u = t)\}) = \mathbf{F}$ and $I_{q0}^{\mathtt{S}_\mathcal{L}}(u = t) = \mathbf{F}$, so

$$I_q^{\mathtt{S}_\mathcal{L}}(t = u) = I_{q0}^{\mathtt{S}_\mathcal{L}}(t = u) = I_{q0}^{\mathtt{S}_\mathcal{L}}(u = t) = \mathbf{F}.$$

Let P' be the paramodulant

$$\begin{aligned}&(s_{q0}(C_{q0}) - \{(u = t)\}) \cup (\{(u = u)\} - \{(u = u)\}) \cup \{(t = u)\}\\&= (s_{q0}(C_{q0}) - \{(u = t)\}) \cup \{(t = u)\}\end{aligned}$$

of $s_{q0}(C_{q0})$ and $\{(u = u)\}$. Then, $I_q^{\mathtt{S}_\mathcal{L}}(P') = \mathbf{F}$. The clause $\{(u = u)\}$ is obtained from the $\mathcal{L}$-clause $\{(x = x)\}$ by a ground $\mathcal{L}$-substitution and the paramodulated occurrence of u in $(u = u)\}$ is visible in $\{(x = x)\}$. By the Paramodulation Lifting Lemma, there is a full, most general paramodulant P of C_{q0} and $\{(x = x)\}$ and a ground $\mathcal{L}$-substitution s with $s(P) = P'$. Then, q is a failure node for $\mathrm{GINST}_\mathcal{L}(\mathcal{C} \cup \{P\})$, so $q0 \notin \mathrm{Dom}(\mathtt{SG}_{\mathcal{L},\mathcal{C}\cup\{P\}})$ and $\mathrm{Dom}(\mathtt{SG}_{\mathcal{L},\mathcal{C}\cup\{P\}}) \subset \mathrm{Dom}(\mathtt{SG}_{\mathcal{L},\mathcal{C}})$. □

Theorem 5.10.66. *Let $\mathcal{L}$ be first-order language with equality and let $\mathcal{C}$ be a set of $\mathcal{L}$-clauses. If $\mathcal{C}$ has no model, then $\Box \in (fRespar^{mgu})^*(\mathcal{C} \cup \{(x = x)\})$.*

Proof. Let $\mathcal{L}' = \mathcal{H}(\mathcal{L})$ be the Herbrand extension of $\mathcal{L}$. We will prove by strong induction on $n \geq 1$ that if $\mathcal{C}$ is a set of $\mathcal{L}'$-clauses that has no model and $|\mathrm{Dom}(\mathtt{SG}_{\mathcal{L}',\mathcal{C}})| = n$, then $\Box \in (\mathrm{fRespar}^{mgu})^*(\mathcal{C} \cup \{(x = x)\}$. This suffices to prove the theorem because if $\mathcal{C}$ is a set of $\mathcal{L}$-clauses that does not have a model, then it is a set of $\mathcal{L}'$-clauses that has no model, so by Theorem 5.10.35, $\mathrm{GINST}_{\mathcal{L}'}(\mathcal{C})$ does not have an $\mathcal{L}'$-E-model, and hence by Theorem 5.10.55, $\mathtt{SG}_{\mathcal{L}',\mathcal{C}} = \mathtt{S}^*_{\mathcal{L}',\mathrm{GINST}_{\mathcal{L}'}(\mathcal{C})}$ is finite.

For the basis step, suppose that $|\mathrm{Dom}(\mathtt{SG}_{\mathcal{L}',\mathcal{C}})| = 1$. Therefore, $\Box \in \mathrm{GINST}_{\mathcal{L}'}(\mathcal{C})$, so $\Box \in \mathcal{C} \subseteq (\mathrm{fRespar}^{mgu})^*(\mathcal{C} \cup \{(x = x)\})$.

For the inductive step, suppose that $\mathcal{C}$ has no model, $|\mathrm{Dom}(\mathtt{SG}_{\mathcal{L}',\mathcal{C}})| = n > 1$ and the result is true for all $n' < n$. Since $|\mathrm{Dom}(\mathtt{SG}_{\mathcal{L}',\mathcal{C}})| > 1$, $\Box \notin \mathrm{GINST}_{\mathcal{L}'}(\mathcal{C})$ and so by Theorem 5.10.63, since $\mathrm{GINST}_{\mathcal{L}'}(\mathcal{C})$ has no $\mathcal{L}'$-E-model, $\mathtt{SG}_{\mathcal{L}',\mathcal{C}}$ contains either a resolution inference node or a paramodulation inference node.

If $\mathtt{SG}_{\mathcal{L}',\mathcal{C}}$ contains a resolution inference node, then by Theorem 5.10.64, there is a full, most general resolvent R of two clauses in $\mathcal{C} \cup \{(x = x)\}$ such that $|\mathrm{Dom}(\mathtt{SG}_{\mathcal{L}',\mathcal{C}\cup\{R\}})| < |\mathrm{Dom}(\mathtt{SG}_{\mathcal{L}',\mathcal{C}})| = n$. Since $\mathcal{C}$ has no model, $\mathcal{C} \cup \{R\}$ has no model, so by inductive hypothesis,

$$\begin{aligned}\Box &\in (\mathrm{fRespar}^{mgu})^*(\mathcal{C} \cup \{R\} \cup \{(x = x)\}) \\ &= (\mathrm{fRespar}^{mgu})^*(\mathcal{C} \cup \{(x = x)\}).\end{aligned}$$

If $\mathtt{SG}_{\mathcal{L}',\mathcal{C}}$ contains a paramodulation inference node, then we reach the desired conclusion by a similar argument using Theorem 5.10.65 instead of Theorem 5.10.64. □

5.11 Exercises and Supplements

A Hilbert/Frege-Style Formal System for First-Order Logic

(1) Verify that the formulas of the groups 1–18 of the formal system $\mathcal{HF}^{\mathcal{L}}$ are first-order tautologies.

(2) Let $\mathcal{L}$ and $\mathcal{L}'$ be first-order languages and let $\Gamma \subseteq \text{FORM}_{\mathcal{L}} \cap \text{FORM}_{\mathcal{L}'}$. Prove that Γ is $\mathcal{L}$-consistent if and only if Γ is $\mathcal{L}'$-consistent.
Hint. This follows from Corollary 5.2.40.

(3) Prove that the following statements:

   (a) the Soundness Theorem;
   (b) for a first-order language $\mathcal{L}$, if a set of $\mathcal{L}$-formulas Γ is satisfiable, then Γ is $\mathcal{L}$-consistent,

can be directly derived from each other.
Hint. The argument is similar to the one of Supplement 6 of Chapter 3.

(4) Prove that the following statements:

   (a) the Completeness Theorem;
   (b) for a first-order language $\mathcal{L}$, if a set of formulas Γ is consistent, then Γ is satisfiable,

can be directly derived from each other.
Hint. The argument is similar to the one of Supplement 7 of Chapter 3.

(5) Without using the Completeness Theorem, prove that for every $\mathcal{L}$-formula φ, we have $\vdash_{\mathcal{HF}^{\mathcal{L}}} (\exists x)(\varphi \to (\forall x)\varphi)$ and $\vdash_{\mathcal{HF}^{\mathcal{L}}} ((\exists x)(\forall y)\varphi \to (\forall y)(\exists x)\varphi)$.
Again, without using the Completeness Theorem, prove that there is an $\mathcal{L}$-formula φ such that $\vdash_{\mathcal{HF}^{\mathcal{L}}} ((\forall x)(\exists y)\varphi \to (\exists y)(\forall x)\varphi)$ does not hold.

(6) Without using the Completeness Theorem, show that the following pairs of formulas are provably equivalent in $\mathcal{HF}^{\mathcal{L}}$:

   (a) $(\neg(\forall x)\varphi), (\exists x)(\neg\varphi)$ and $(\neg(\exists x)\varphi), (\forall x)(\neg\varphi)$;
   (b) $((\exists x)\varphi \vee \psi), (\exists x)(\varphi \vee \psi)$ and $((\forall x)\varphi \vee \psi), (\forall x)(\varphi \vee \psi)$, where $x \notin \texttt{FV}(\psi)$;
   (c) $(\varphi \vee (\exists x)\psi), (\exists x)(\varphi \vee \psi)$ and $(\varphi \vee (\forall x)\psi), (\forall x)(\varphi \vee \psi)$, where $x \notin \texttt{FV}(\varphi)$;
   (d) $((\exists x)\varphi \wedge \psi), (\exists x)(\varphi \wedge \psi)$ and $((\forall x)\varphi \wedge \psi), (\forall x)(\varphi \wedge \psi)$, where $x \notin \texttt{FV}(\psi)$;
   (e) $(\varphi \wedge (\exists x)\psi), (\exists x)(\varphi \wedge \psi)$ and $(\varphi \wedge (\forall x)\psi), (\forall x)(\varphi \wedge \psi)$, where $x \notin \texttt{FV}(\varphi)$;
   (f) $((\exists x)\varphi \to \psi), (\forall x)(\varphi \to \psi)$ and $((\forall x)\varphi \to \psi), (\exists x)(\varphi \to \psi)$, where $x \notin \texttt{FV}(\psi)$;

(g) $(\varphi \to (\exists x)\psi), (\exists x)(\varphi \to \psi)$ and $(\varphi \to (\forall x)\psi), (\forall x)(\varphi \to \psi)$, where $x \notin \mathtt{FV}(\varphi)$.

(7) Using Theorems 5.2.43, 5.2.49, and 5.2.51, and Exercise 6, give a syntactic argument that for every $\mathcal{L}$-formula φ there is an $\mathcal{L}$-formula φ' in prenex normal form such that φ and φ' are provably equivalent in $\mathcal{HF}^{\mathcal{L}}$.

(8) Let $\mathcal{L}$ be a first-order language. The formal system $\mathcal{HF}'^{\mathcal{L}}$ is defined starting from the formal system $\mathcal{HF}^{\mathcal{L}}$ by replacing the first 18 formula groups with the formula groups obtained from the axioms of the propositional formal system $\mathcal{HF}'$ (defined in Exercise 5 of Chapter 3) by substituting $\mathcal{L}$-formulas for α, β, γ. Prove that for any set of $\mathcal{L}$-formulas Γ and $\mathcal{L}$-formula φ, we have $\Gamma \models \varphi$ if and only if $\Gamma \vdash_{\mathcal{HF}'^{\mathcal{L}}} \varphi$.
Hint. The argument parallels that of Section 5.2 for $\mathcal{HF}^{\mathcal{L}}$. The analogue of Theorem 5.2.3 follows from Exercise 5 of Chapter 3.

First-Order Tableaux

(9) Prove that if $b\varphi$ is a signed formula that occurs in a node of a $(\Delta, \mathcal{L}, V)$-tableau, then $\mathtt{FV}(b\varphi) \subseteq V$.

(10) Let Δ be a set of signed propositional formulas and s be an atomic $\mathcal{L}$-substitution, where $\mathcal{L}$ is a first-order language without equality. Prove that for every $(s(\Delta), \mathcal{L}, V)$-tableau $\mathtt{T}'$ without cut, there is a Δ-tableau $\mathtt{T}$ (of propositional logic) such that $\mathtt{T}' = s \circ \mathtt{T}$. Further, prove that for an injective inter-substitution s, if $\mathtt{T}'$ is closed, then so is $\mathtt{T}$.
Hint. Use Exercise 152 of Chapter 4. For the second part, use Exercise 107 of Chapter 4.

(11) Let Δ be a set of signed $(\mathcal{L}, V)$-formulas, where V is an $\mathcal{L}$-suitable set of variables. Suppose that Δ consists of formulas of the form $\mathbf{T}\varphi, \mathbf{T}\psi, \mathbf{F}\alpha$, where φ is a Π_1-formula and ψ, α are Π_0-formulas, that is, quantifier-free formulas. Prove that if $\mathtt{T}$ is a $(\Delta, \mathcal{L}, V)$-tableau, then there are no $\boldsymbol{\delta}$-formulas in any node of $\mathtt{T}$.
Hint. Use induction on the length of the path that joins a node to the root of the tableau.

(12) Let $\mathcal{L}$ be a first-order language that contains an infinite set of constant symbols, V be an $\mathcal{L}$-suitable set of variables and

Δ be a finite set of signed $(\mathcal{L}, V)$-formulas. Prove that Δ is unsatisfiable if and only if it is a theorem of $\mathcal{F}^{\text{tabl,cons}}_{\mathcal{L},V}$.

(13) Show that the analogue of Exercise 12 of Chapter 3 for first-order tableaux is false.
Hint. Consider the tableau shown in Figure 5.7.

(14) Recall that the notion of inconsistency property was introduced in Definition 4.12.22. Let $\mathcal{L}$ be a first-order language and V be a set of variables.

(a) Prove that the collection $\mathcal{I}$ of all sets of signed $(\mathcal{L}^c, V)$-formulas Δ such that either Δ contains infinitely many constant symbols of $\mathcal{L}^c - \mathcal{L}$ or Δ contains only finitely many constant symbols of $\mathcal{L}^c - \mathcal{L}$ and there is a finite closed $(\Delta, \mathcal{L}^c, V)$-tableau is an $(\mathcal{L}^c, V)$-inconsistency property.
(b) Prove that the collection $\mathcal{I}_s$ of all sets of signed $(\mathcal{L}^c, V)$-formulas Δ such that either Δ contains infinitely many constant symbols of $\mathcal{L}^c - \mathcal{L}$ or Δ contains only finitely many constant symbols of $\mathcal{L}^c - \mathcal{L}$ and there is a strongly closed $(\Delta, \mathcal{L}^c, V)$-tableau is an $(\mathcal{L}^c, V)$-inconsistency property.
(c) Use the first part to give an alternative proof of the Completeness Theorem for Tableaux of First-Order Logic. Use the second part to give an alternative proof of the Strong Completeness Theorem for Tableaux of First-Order Logic.

(15) Let Δ be an unsatisfiable set of $(\mathcal{L}, V)$-formulas. Using a finite conservative closed $(\Delta, \mathcal{L}^c, V)$-tableau, give an explicit nonrecursive construction of a finite unsatisfiable subset of Δ.
Hint. See Supplement 24 of Chapter 3.

(16) Reprove Exercise 169 of Chapter 4 using tableaux.

(17) Let $\mathcal{L}$ be a first-order language, V be an $\mathcal{L}$-suitable set of variables and Γ be a set of $(\mathcal{L}, V)$-formulas. Prove an analogue of Supplement 26 of Chapter 3 by giving a construction whose input is a strongly closed unsigned $(\Gamma, \mathcal{L}, V)$-tableau $\mathtt{T}$ and whose output is a strongly closed $(\mathsf{s}(\Gamma), \mathcal{L}, V)$-tableau $\mathtt{T}'$.
Solution: The case when $\mathtt{T}$ is a one-node tree is identical to the corresponding case in Supplement 26 of Chapter 3.
Consider now the case when $\mathtt{T}$ has more than one node. If we used variantization at the root, that is $\mathtt{T}(\lambda) = \Gamma$ and there is a finite-to-one function $f : \mathtt{T}(0) \longrightarrow \mathtt{T}(\lambda)$ such that for every formula $\varphi \in \mathtt{T}(0)$, φ is a variant of $f(\varphi)$, apply the construction

recursively to $\mathtt{T}_{[0]}$ to produce a strongly closed $(\mathsf{s}(\mathtt{T}(0)), \mathcal{L}, V)$-tableau $\mathtt{T}'_0$ and return $\mathtt{T}' = (\mathtt{T}'_0; \mathsf{s}(\Gamma))$. Note that if we define $f' : \mathsf{s}(\mathtt{T}(0)) \longrightarrow \mathsf{s}(\Gamma)$ by $f'(\mathsf{s}(\varphi)) = \mathsf{s}(f(\varphi))$, then f' is well-defined and finite-to-one because s is one-to-one and f is finite-to-one, and for all $\mathsf{s}(\varphi) \in \mathsf{s}(\mathtt{T}(0))$, $\mathsf{s}(\varphi)$ is a variant of $f'(\mathsf{s}(\varphi))$, so $\mathtt{T}'$ is an $(\mathsf{s}(\Gamma), \mathcal{L}, V)$-tableau using variantization at the root. If propositional expansion is used at the root of $\mathtt{T}$, the construction is the same as in Supplement 26 of Chapter 3.
Suppose that $\boldsymbol{\gamma}$-expansion is used at the root of $\mathtt{T}$. If $\Gamma = \mathtt{T}(\lambda) = \Gamma' \cup \{(\forall x)\psi\}$ and $\mathtt{T}(0) = \Gamma' \cup \{(\forall x)\psi, (\psi')_{x:=t}\}$, where ψ' is a variant of the positive formula ψ such that t is substitutable for x in ψ', then we apply the construction to $\mathtt{T}_{[0]}$ to obtain $\mathtt{T}'_0$, an $(\mathsf{s}(\Gamma') \cup \{\mathbf{T}(\forall x)\psi, \mathbf{T}(\psi')_{x:=t}\})$-tableau. Return $(\mathtt{T}'_0; \mathsf{s}(\Gamma))$.
If

$$\begin{aligned} \Gamma = \mathtt{T}(\lambda) &= \Gamma' \cup \{(\forall x)(\neg\psi)\}, \text{ and} \\ \mathtt{T}(0) &= \Gamma' \cup \{(\forall x)(\neg\psi), ((\neg\psi'))_{x:=t}\}, \end{aligned}$$

where ψ' is a variant of the formula ψ such that t is substitutable for x in ψ', then we apply the construction to $\mathtt{T}_{[0]}$ to obtain $\mathtt{T}'_0$, an $(\mathsf{s}(\Gamma') \cup \{\mathbf{T}(\forall x)(\neg\psi), \mathbf{F}(\psi')_{x:=t}\})$-tableau. Return

$$((\mathtt{T}'_0; \mathsf{s}(\Gamma') \cup \{\mathbf{T}(\forall x)(\neg\psi), \mathbf{T}((\neg\psi'))_{x:=t}\}); \mathsf{s}(\Gamma)).$$

We leave the remaining $\boldsymbol{\gamma}$-expansion cases, as well as the $\boldsymbol{\delta}$-expansion and equality expansion cases to the reader.

(18) Let $\mathcal{L}$ be a first-order language, V be an $\mathcal{L}$-suitable set of variables and Γ be a set of $(\mathcal{L}, V)$-formulas. Prove that for every node q of a $(\Gamma, \mathcal{L}, V)$-tableau $\mathtt{T}$, we have

$$\mathtt{T}(q) \subseteq \begin{cases} W^*_{\mathcal{L},V}(\Gamma) & \text{if } = \notin \mathcal{L} \\ W^*_{\mathcal{L},V}(\Gamma \cup \mathrm{INST}_{\mathcal{L},V}(\mathrm{Eq}_{=,\mathcal{L}})) & \text{if } = \in \mathcal{L}. \end{cases}$$

(19) Formulate and prove analogues of Theorem 5.3.14, Corollaries 5.3.15 and 5.3.16, Theorem 5.3.18, and Theorem 5.3.19 for unsigned tableaux.

(20) Let $\mathcal{L}$ be a first-order language, V be an $\mathcal{L}$-suitable set of variables and Γ be a set of $(\mathcal{L}, V)$-formulas.

(a) Let $\mathtt{P}$ be a path of a conservative $(\Gamma, \mathcal{L}, V)$-tableau $\mathtt{T}$ ending in the node q. Prove that:

i. If $\varphi \in \mathtt{T}(\mathtt{P})$ is a $\boldsymbol{\gamma}$-formula then $\varphi \in \mathtt{T}(q)$.
ii. If $\varphi \in \mathtt{T}(\mathtt{P}) - \mathtt{T}(q)$, then there is an $(\mathcal{L}, V)$-constituent K of φ such that $K \subseteq \mathtt{T}(\mathtt{P})$.

(b) Use a reformulated Definition 5.3.24 to give a construction of a completed $(\Gamma, \mathcal{L}^c, V)$-tableau.

(21) Let $\mathcal{L}$ be a first-order language, V be a set of variables and Γ be a set of $(\mathcal{L}, V)$-formulas.

(a) Prove that if Γ is unsatisfiable, then there exists a finite, conservative, closed $(\Gamma, \mathcal{L}^c, V)$-tableau. (This is the Completeness Theorem for Unsigned Tableaux of First-Order Logic.)
(b) Prove that the following statements are equivalent:

i. Γ is unsatisfiable.
ii. There exists a finite conservative, closed $(\Gamma, \mathcal{L}^c, V)$-tableau.
iii. There is a conservative, closed $(\Gamma, \mathcal{L}^c, V)$-tableau.

(22) Use Part (b) of Exercise 20 to reprove Exercise 154 of Chapter 4.
(23) Let $\mathcal{L}$ be a first-order language, V be a set of variables and Γ be a set of $(\mathcal{L}, V)$-formulas.

(a) Prove constructively that if Γ is unsatisfiable, then there is a conservative, strongly closed $(\Gamma, \mathcal{L}^c, V)$-tableaux. (This is the Strong Completeness Theorem for Unsigned Tableaux of First-Order Logic.)
(b) Show that any of the statements included in Part b of Exercise 21 is equivalent to the existence of a conservative, strongly closed $(\Gamma, \mathcal{L}^c, V)$-tableau.

(24) Use Exercise 23 to obtain an another proof of the Compactness Theorem for First-Order Logic.
(25) Construct an analogue of the formal system $\mathcal{F}^{\text{tabl}}_{\mathcal{L},V}$ for unsigned formulas, and prove its soundness and partial completeness.
(26) Construct an analogue of the formal system $\mathcal{F}^{\text{tabl,cons}}_{\mathcal{L},V}$ for unsigned formulas, and prove its soundness and partial completeness.
(27) State and prove an analogue of Exercise 14 for sets of unsigned formulas.

(28) In Supplement 66 of Chapter 4 we proved that the set of signed formulas Δ_n is unsatisfiable. Show that every closed $(\Delta_n, \mathcal{L}^c)$-tableau has at least 2^{2^n} nodes. (Thus, by Exercise 9 of Chapter 4, the number of nodes of any closed $(\Delta_n, \mathcal{L}^c)$-tableau is exponential in the size of Δ_n.)

Solution: Let $\mathtt{T}$ be a closed $(\Delta_n, \mathcal{L}^c)$-tableau. We introduce the $\mathcal{L}^c$-structure $\mathcal{A}_0$ by $|\mathcal{A}_0| = \mathbf{P}$, $L^{\mathcal{A}_0} = \mathbf{P}$, $c_1^{\mathcal{A}_0} = 1$, $d^{\mathcal{A}_0}(n) = 2n$, $s^{\mathcal{A}_0}(n,m) = n + m$, for $n, m \in \mathbf{P}$, and letting $c^{\mathcal{A}_0}$ be arbitrary positive number for c a constant symbol different from c_1. A positive number i is $\mathtt{T}$-instanced if there is an $\mathcal{L}^c$-ground term t such that $i = t^{\mathcal{A}_0}$ and the formula $\mathbf{T}(L(t) \to L(s(t, c_1)))$ occurs in $\mathtt{T}$.

Define the $\mathcal{L}^c$-structure $\mathcal{A}_{\mathtt{T}}$ by

$$L^{\mathcal{A}_{\mathtt{T}}} = \{i \in \mathbf{P} \mid j \text{ is } \mathtt{T}\text{-instanced for all positive } j < i\}.$$

The remaining symbols of $\mathcal{L}^c$ have the same interpretation in $\mathcal{A}_{\mathtt{T}}$ as in $\mathcal{A}_0$.

We claim that if a formula $\mathbf{T}(L(t) \to L(s(t, c_1)))$ occurs in a node of $\mathtt{T}$, then $\mathcal{A}_{\mathtt{T}} \models \mathbf{T}(L(t) \to L(s(t, c_1)))$. Suppose that $\mathbf{T}(L(t) \to L(s(t, c_1)))$ occurs in $\mathtt{T}$ and that $\mathcal{A}_{\mathtt{T}} \models L(t)$. We must show that $\mathcal{A}_{\mathtt{T}} \models L(s(t, c_1))$. Since $\mathcal{A}_{\mathtt{T}} \models L(t)$, we have $t^{\mathcal{A}_{\mathtt{T}}} \in L^{\mathcal{A}_{\mathtt{T}}}$. Therefore, for all $j < t^{\mathcal{A}_{\mathtt{T}}}$, j is $\mathtt{T}$-instanced. Since $t^{\mathcal{A}_{\mathtt{T}}} = t^{\mathcal{A}_0}$ and the signed formula $\mathbf{T}(L(t) \to L(s(t, c_1)))$ occurs in a node of $\mathtt{T}$, it follows that $t^{\mathcal{A}_{\mathtt{T}}}$ is $\mathtt{T}$-instanced. So, every $j < t^{\mathcal{A}_{\mathtt{T}}} + 1 = s(t, c_1)^{\mathcal{A}_{\mathtt{T}}}$ is $\mathtt{T}$-instanced. Consequently, $s(t, c_1)^{\mathcal{A}_{\mathtt{T}}} \in L^{\mathcal{A}_{\mathtt{T}}}$, that is $\mathcal{A}_{\mathtt{T}} \models L(s(t, c_1))$, as desired.

Now we show that every number less than 2^{2^n} is $\mathtt{T}$-instanced. Suppose that $i < 2^{2^n}$ were not $\mathtt{T}$-instanced.

Let $\Delta' = \{\mathbf{T}\theta' \mid \mathbf{T}\theta' \text{ is a variant of } \mathbf{T}\beta\}$. We claim that there is a branch $\mathtt{B}$ in $\mathtt{T}$ such that $\mathcal{A}_{\mathtt{T}} \models \mathtt{T}(\mathtt{B}) - \Delta'$. We verify first that $\mathcal{A}_{\mathtt{T}} \models \mathtt{T}(\lambda) - \Delta' = \mathtt{T}(\lambda) - \{\mathbf{T}\beta\}$, where $\mathtt{T}(\lambda) = \Delta_n$. Since $s^{\mathcal{A}_{\mathtt{T}}}$ is an associative binary operation on $\mathbf{P}$, it is clear that $\mathcal{A}_{\mathtt{T}} \models \mathbf{T}\varphi$. We have $\mathcal{A} \models \mathbf{T}\psi$ because of the definitions of $d^{\mathcal{A}_{\mathtt{T}}}$ and $s^{\mathcal{A}_{\mathtt{T}}}$. Also, $\mathcal{A}_{\mathtt{T}} \models \mathbf{T}L(c_1)$ because there are no positive integers less than 1. Finally, $\mathcal{A}_{\mathtt{T}}$ is a model of $\mathbf{F}\theta_n$ because

$$\underbrace{(d(\cdots d}_{2^n}(c_1)\cdots))^{\mathcal{A}_{\mathtt{T}}} = 2^{2^n}$$

and because of our assumption that there is an $i < 2^{2^n}$ that is not $\mathtt{T}$-instanced.

Now suppose that q is an interior node of $\mathtt{T}$ and that $\mathcal{A}_{\mathtt{T}} \models \mathtt{T}(q) - \Delta'$. We wish to prove that there is an immediate descendant q' of q such that $\mathcal{A}_{\mathtt{T}} \models \mathtt{T}(q') - \Delta'$. We need to consider three cases depending on the type of expansion used at q:

(a) If equality expansion was used, then there is only one immediate descendant $q' = q0$ of q and $\mathtt{T}(q')$ is obtained from $\mathtt{T}(q)$ by adding a logically valid formula. Thus, by the inductive hypothesis, $\mathcal{A}_{\mathtt{T}} \models \mathtt{T}(q') - \Delta'$.

(b) If regular expansion was used at q, it does not involve a $\boldsymbol{\delta}$-formula because no such formula exists in $\mathtt{T}$ by Exercise 11. Suppose initially that the formula expanded at q is not in Δ'. Then, by Theorem 4.12.48, there is a constituent K_i of the expanded formula such that $\mathcal{A}_{\mathtt{T}} \models K_i$ and therefore $\mathcal{A}_{\mathtt{T}} \models \mathtt{T}(qi) - \Delta'$. If the formula expanded at q is in Δ', then q has only one immediate descendant $q' = q0$, $\mathtt{T}(q') = \mathtt{T}(q) \cup \{\mathbf{T}(L(t) \to L(s(t, c_1)))\}$ for some $\mathcal{L}^c$-ground term t and $\mathcal{A}_{\mathtt{T}} \models \mathtt{T}(q') - \Delta'$ because of the inductive hypothesis and the fact that $\mathcal{A}_{\mathtt{T}} \models \mathbf{T}(L(t) \to L(s(t, c_1)))$.

(c) If variantizing was used at q, then q has one immediate descendant $q0$ in $\mathtt{T}$ and if $b\theta' \in \mathtt{T}(q0) - \Delta'$, then $b\theta'$ is variant of a formula $b\theta \in \mathtt{T}(q)$. We claim that $b\theta \notin \Delta'$. Indeed, if $b\theta \in \Delta'$, this would imply that $b\theta'$ would also belong to Δ' because a variant of a variant of $\mathbf{T}\beta$ is again a variant of $\mathbf{T}\beta$. Thus, $b\theta \in \mathtt{T}(q) - \Delta'$, so $\mathcal{A}_{\mathtt{T}} \models b\theta$. Since variants are logically equivalent to each other, it follows that $\mathcal{A}_{\mathtt{T}} \models b\theta'$. Thus, $\mathcal{A}_{\mathtt{T}} \models \mathtt{T}(q0) - \Delta'$.

Thus, we have proved that there is a branch $\mathtt{B}$ in $\mathtt{T}$ such that $\mathcal{A}_{\mathtt{T}} \models \mathtt{T}(\mathtt{B}) - \Delta'$, which implies that there is no formula γ which is not a variant of β such that both $\mathbf{T}\gamma$ and $\mathbf{F}\gamma$ are in $\mathtt{T}(\mathtt{B})$. Since $\mathtt{T}$ is a closed $(\Delta, \mathcal{L}^c)$-tableau, $\mathtt{T}(\mathtt{B})$ must be closed, so it must contain $\mathbf{F}\beta'$ for some variant β' of β. However, $\mathbf{F}\beta'$ is a $\boldsymbol{\delta}$-formula and $\mathtt{T}$ contains no such formulas. This contradiction shows that every number less than 2^{2^n} must be $\mathtt{T}$-instanced.

We leave it to the reader to prove by induction on $|q|$ that for every node q of $\mathtt{T}$, the formulas that occur in q have one of the following forms

1. $b\varphi$, where φ does not contain the relation symbol L;
2. $\mathbf{T}\beta'$, where β' is a variant of β;
3. $\mathbf{T}(L(t) \rightarrow L(s(t, c_1)))$ for some $\mathcal{L}^c$-ground term t;
4. $\mathbf{T}((t = s) \rightarrow (L(t) \leftrightarrow L(s)))$ for some $\mathcal{L}^c$-ground terms t and s;
5. $\mathbf{T}(L(t) \leftrightarrow L(s))$ for some $\mathcal{L}^c$-ground terms t and s;
6. $bL(t)$ for some $\mathcal{L}^c$-ground term t,

and if q is an immediate descendant of q', then there is at most one formula of the third type in $\mathtt{T}(q) - \mathtt{T}(q')$. Since the root contains no formula of type three, there are more nodes in $\mathtt{T}$ than there are formulas of type three. Since every $i < 2^{2^n}$ is $\mathtt{T}$-instanced, $\mathtt{T}$ must have at least 2^{2^n} nodes.

Cut Rule for First-Order Tableaux

(29) Let Δ be a finite set of signed $\mathcal{L}$-sentences, where $\mathcal{L}$ is a first-order language with infinitely many constant symbols. Suppose that $\mathtt{U}$ is a strongly closed $(\Delta \cup \{\mathbf{T}(\exists x)\varphi\}, \mathcal{L})$-tableau and $\mathtt{V}$ is a strongly closed $(\Delta \cup \{\mathbf{F}(\varphi)_{x:=t}\}, \mathcal{L})$-tableau, where t is an $\mathcal{L}$-ground term. Give an effective, syntactic transformation that produces a strongly closed $(\Delta, \mathcal{L})$-tableau.
Hint. Modify the argument of Example 5.4.14.

(30) Let Δ be a finite set of signed $\mathcal{L}$-sentences, where $\mathcal{L}$ is a first-order language with infinitely many constant symbols, φ be an $\mathcal{L}$-sentence and c be a constant symbol in $\mathcal{L}$ that does not occur in either Δ or φ.

(a) Suppose that $\mathtt{U}$ is a strongly-closed $(\Delta \cup \{\mathbf{F}(\exists x)\varphi\}, \mathcal{L})$-tableau and $\mathtt{V}$ is a strongly-closed $(\Delta \cup \{\mathbf{T}(\varphi)_{x:=c}\}, \mathcal{L})$-tableau. Give an effective, syntactic transformation that produces a strongly closed $(\Delta, \mathcal{L})$-tableau.

(b) Suppose that $\mathtt{U}$ is a strongly-closed $(\Delta \cup \{\mathbf{T}(\forall x)\varphi\}, \mathcal{L})$-tableau and $\mathtt{V}$ is a strongly-closed $(\Delta \cup \{\mathbf{F}(\varphi)_{x:=c}\}, \mathcal{L})$-tableau. Give an effective, syntactic transformation that produces a strongly closed $(\Delta, \mathcal{L})$-tableau.

(31) Let Δ be a finite set of $\mathcal{L}$-sentences, where $\mathcal{L}$ is a first-order language with infinitely many constant symbols. Further, let $b\varphi$ be a signed formula, where φ is either a $\boldsymbol{\delta}$ formula, a formula of the form $(\neg\alpha)$, or a formula of the form $(\alpha C \beta)$ with C a binary connective symbol. Suppose that K is an $\mathcal{L}$-constituent

of $b\varphi$, and $\mathtt{T}$ is a strongly closed $(\Delta \cup \{b\varphi\}, \mathcal{L})$-tableau. Using cut elimination, give an effective, syntactic way of constructing a strongly closed $(\Delta \cup K, \mathcal{L})$-tableau.

Solution: We discuss only the case of $\boldsymbol{\delta}$-formulas. The remaining cases are similar and we refer the reader to Supplement 45 of Chapter 3.

Let $b\varphi = b(Qx)\psi$ be a $\boldsymbol{\delta}$-formula and let $K = \{b(\psi)_{x:=t}\}$ be one of its $(\mathcal{L}, \emptyset)$-constituents. Define $\mathtt{U} = \mathtt{T} \uplus K$, which is a strongly closed $(\Delta \cup K \cup \{b\varphi\}, \mathcal{L})$-tableau. Define the tableau $\mathtt{V}$, where $\mathrm{Dom}(\mathtt{V}) = \{\lambda, 0\}$, $\mathtt{V}(\lambda) = \Delta \cup K \cup \{\bar{b}\varphi\}$, and $\mathtt{V}(0) = \Delta \cup K \cup \{\bar{b}\varphi, \bar{b}(\varphi)_{x:=t}\}$. It is easy to see that $\mathtt{V}$ is a strongly closed $(\Delta \cup K \cup \{\bar{b}\varphi\})$-tableau. The desired tableau is produced by cut elimination from the tableaux $\mathtt{U}$ and $\mathtt{V}$.

(32) Let Δ be a finite set of $\mathcal{L}$-sentences, where $\mathcal{L}$ is a first-order language with infinitely many constant symbols, φ be an $\mathcal{L}$-formula such that $\mathtt{FV}(\varphi) \subseteq \{x\}$ and t be a ground term of $\mathcal{L}$.

Prove that if $\mathtt{T}$ is a strongly closed $(\Delta \cup \{\mathbf{T}((\forall x)\varphi \to (\varphi)_{x:=t})\}, \mathcal{L})$-tableau or a strongly closed $(\Delta \cup \{\mathbf{T}((\varphi)_{x:=t}) \to (\exists x)\varphi)\}, \mathcal{L})$-tableau, then there is an effective syntactic construction of a strongly closed $(\Delta, \mathcal{L})$-tableau starting from $\mathtt{T}$.

Solution: Suppose that $\mathtt{T}$ is a strongly closed $(\Delta \cup \{\mathbf{T}((\forall x)\varphi \to (\varphi)_{x:=t})\}, \mathcal{L})$-tableau. By Supplement 31, we can construct effectively a strongly closed $(\Delta \cup \{\mathbf{F}(\forall x)\varphi\}, \mathcal{L})$-tableau and a strongly closed $(\Delta \cup \{\mathbf{T}(\varphi)_{x:=t}\}, \mathcal{L})$-tableau. The desired $(\Delta, \mathcal{L})$-tableau is obtained by applying the construction in Example 5.4.14.

The argument for the other case is similar, but uses Part (a) of Supplement 29 in place of Example 5.4.14.

(33) Let Δ be a finite set of $\mathcal{L}$-sentences, where $\mathcal{L}$ is a first-order language with infinitely many constant symbols, φ be an $\mathcal{L}$-formula such that $\mathtt{FV}(\varphi) \subseteq \{x\}$ and c be a constant symbol of $\mathcal{L}$ that does not occur in Δ or in φ.

Prove that if $\mathtt{T}$ is a strongly closed $(\Delta \cup \{\mathbf{T}((\exists x)\varphi \to (\varphi)_{x:=c})\}, \mathcal{L})$-tableau or a strongly closed $(\Delta \cup \{\mathbf{T}((\varphi)_{x:=c}) \to (\forall x)\varphi)\}, \mathcal{L})$-tableau, then there is an effective syntactic construction of a strongly closed $(\Delta, \mathcal{L})$-tableau starting from $\mathtt{T}$.

(34) Let $\mathcal{L}$ be a first-order language, V be a set of variables, Δ be a set of signed propositional formulas and s be an $\mathrm{FORM}_{\mathcal{L}}(V)$-inter-substitution. Prove that if $\mathtt{T}$ is a propositional Δ-tableau

(with cut), then $s \circ \mathtt{T}$ is an $(s(\Delta), \mathcal{L}, V)$-tableau (with cut). Furthermore, if $\mathtt{T}$ is (strongly) closed, then so is $s \circ \mathtt{T}$.

(35) Give an example of a first-order language $\mathcal{L}$ and a sequence of unsatisfiable sets of signed $\mathcal{L}$-sentences $(\Delta'_1, \Delta'_2, \ldots)$ with the following two properties:

- for every sequence $(\mathtt{T}'_1, \mathtt{T}'_2, \ldots)$ such that $\mathtt{T}'_n$ is a closed $(\Delta'_n, \mathcal{L})$-tableau (without cut) for $n \geq 1$, we have $\mathtt{size}(\mathtt{T}'_n)$ is super-polynomial in $\mathtt{size}(\Delta'_n)$;
- there is a sequence $(\mathtt{T}^\dagger_1, \mathtt{T}^\dagger_2, \ldots)$ such that $\mathtt{T}^\dagger_n$ is a closed $(\Delta'_n, \mathcal{L})$-tableau with cut and $\mathtt{size}(\mathtt{T}^\dagger_n)$ is polynomial in $\mathtt{size}(\Delta'_n)$.

Hint. Take $\mathcal{L}$ to be the language introduced in the hint of Exercise 108 of Chapter 4 and s be the inter-substitution defined there. Then take $\Delta'_n = s(\Delta_n)$ for $n \geq 1$, where Δ_n is the set of formulas introduced in Supplement 40 of Chapter 3. Use Supplements 40 and 47 of Chapter 3, and Exercises 10 and 34 to complete the argument.

(36) Prove that if K is an $(\mathcal{L}, V)$-constituent of a signed $(\mathcal{L}, V)$-formula φ with $\varphi = (\alpha C \beta)$, for some binary connective symbol C, or $\varphi = (\neg\alpha)$, then using the cut rule, there is an effective way of constructing a (strongly) closed $(\Delta \cup K, \mathcal{L}, V)$-tableau with cut from a (strongly) closed $(\Delta \cup \{b\varphi\}, \mathcal{L}, V)$-tableau with cut.
Hint. The argument is identical to the one provided in Supplement 45 of Chapter 3.

(37) In Exercise 67 of Chapter 4, we proved that the set of signed formulas

$$\widehat{\Delta} = \{\mathbf{T}\varphi, \mathbf{T}\psi, \mathbf{T}(\gamma)_{y:=a}, \mathbf{F}(\gamma)_{y:=d(a)})\}$$

is unsatisfiable, where

$$\begin{aligned} \varphi &= (\forall x)(\forall y)(\forall z)(s(x, s(y, z)) = s(s(x, y), z)) \\ \psi &= (\forall x)(d(x) = s(x, x)) \\ \gamma &= (L(y) \wedge (\forall x)(L(x) \to L(s(x, y)))). \end{aligned}$$

Also, the set Δ_n and the language $\mathcal{L}$ were defined in Exercise 9 of Chapter 4.

(a) Prove that there is a strongly closed $(\widehat{\Delta}, \mathcal{L}^c)$-tableau $\widehat{\mathtt{T}}$ with no more than 46 nodes which never uses in a $\boldsymbol{\delta}$-expansion either a or c_1, where c_1 is the constant symbol introduced in the definition of Δ_n.

(b) Using the first part, prove that there is a sequence of tableaux $\widehat{\mathtt{T}}_0, \ldots, \widehat{\mathtt{T}}_n, \ldots$ such that each $\widehat{\mathtt{T}}_i$ is a strongly closed $(\widehat{\Delta}_i, \mathcal{L}^c)$-tableau, where $\widehat{\Delta}_i = \{\mathbf{T}\varphi, \mathbf{T}\psi, \mathbf{T}(\gamma)_{y:=a}, \mathbf{F}(\gamma)_{y:=d^{2^i}(a)}\}$, neither a nor c_1 is used in a $\boldsymbol{\delta}$-expansion in $\widehat{\mathtt{T}}_i$, and $\mathtt{size}(\widehat{\mathtt{T}}_i) = O((2^i)^2)$. Here $d^k(a)$ is the term obtained by applying k times the function symbol d to a.

Solution: We leave the construction of the tableau required by the first part to the reader.

Define $\widehat{\mathtt{T}}_0 = \widehat{\mathtt{T}}$. Suppose that we have obtained $\widehat{\mathtt{T}}_i$, a strongly closed $(\widehat{\Delta}_i, \mathcal{L}^c)$-tableau that never uses in a $\boldsymbol{\delta}$-expansion either a or c_1. Since the single constant symbol in the term $d^{2^i}(a)$ is a, by Theorem 5.4.8, we obtain the strongly closed $(\Delta_i', \mathcal{L})$-tableau $\mathtt{T}_i' = \mathsf{s}^a_{d^{2^i}(a)}(\widehat{\mathtt{T}}_i)$, where

$$\Delta_i' = \{\mathbf{T}\varphi, \mathbf{T}\psi, \mathbf{T}(\gamma)_{y:=d^{2^i}(a)}, \mathbf{F}(\gamma)_{y:=d^{2^{i+1}}(a)}\}.$$

This allows us to define the tableaux:

$$\begin{aligned} \mathtt{T}_i^\dagger &= \widehat{\mathtt{T}}_i \uplus \mathbf{F}(\gamma)_{y:=d^{2^{i+1}}(a)} \\ \mathtt{T}_i^\star &= \mathtt{T}_i' \uplus \mathbf{T}(\gamma)_{y:=a}, \end{aligned}$$

where $\mathtt{T}_i^\dagger$ is a strongly closed $(\Delta_i^\dagger, \mathcal{L})$-tableau, $\mathtt{T}_i^\star$ is a strongly closed $(\Delta_i^\star, \mathcal{L})$-tableau, and

$$\begin{aligned} \Delta_i^\dagger &= \{\mathbf{T}\varphi, \mathbf{T}\psi, \mathbf{T}(\gamma)_{y:=a}, \mathbf{F}(\gamma)_{y:=d^{2^i}(a)}, \mathbf{F}(\gamma)_{y:=d^{2^{i+1}}(a)}\}, \\ \Delta_i^\star &= \{\mathbf{T}\varphi, \mathbf{T}\psi, \mathbf{T}(\gamma)_{y:=a}, \mathbf{T}(\gamma)_{y:=d^{2^i}(a)}, \mathbf{F}(\gamma)_{y:=d^{2^{i+1}}(a)}\}. \end{aligned}$$

Applying the cut rule to the tableaux $\mathtt{T}_i^\star$ and $\mathtt{T}_i^\dagger$ yields the strongly closed $(\widehat{\Delta}_{i+1}, \mathcal{L})$-tableau

$$\widehat{\mathtt{T}}_{i+1} = (\mathtt{T}_i^\star, \mathtt{T}_i^\dagger; \{\mathbf{T}\varphi, \mathbf{T}\psi, \mathbf{T}(\gamma)_{y:=a}, \mathbf{F}(\gamma)_{y:=d^{2^{i+1}}(a)}\}).$$

Observe that for the number of nodes $|\widehat{\mathtt{T}}_i|$ of $\widehat{\mathtt{T}}_i$, we have the recurrence $|\widehat{\mathtt{T}}_{i+1}| = 2|\widehat{\mathtt{T}}_i| + 3$, which implies that $|\widehat{\mathtt{T}}_i| = \Theta(2^i)$.

Let A_i be the number of occurrences of the constant symbol a in the labels of the nodes of $\widehat{\mathtt{T}}_i$. The definitions of $\mathtt{T}_i^\dagger$ and $\mathtt{T}_i^\star$ imply that the symbol a occurs in the labels of each of these tableaux $A_i + 6$ times. Therefore, we have $A_{i+1} = 2A_i + 16$, which yields $A_i = \Theta(2^i)$.

The definition of $\mathtt{T}_i^\dagger$ implies that

$$\begin{aligned}\mathtt{size}(\mathtt{T}_i^\dagger) &= \mathtt{size}(\widehat{\mathtt{T}}_i) + \mathtt{size}(\Delta_i^\dagger)\\ &= \mathtt{size}(\widehat{\mathtt{T}}_i) + \Theta(2^i).\end{aligned}$$

Since $\mathtt{T}_i' = \mathsf{s}^a_{d^{2^i}(a)}(\widehat{\mathtt{T}}_i)$, we have

$$\begin{aligned}\mathtt{size}(\mathtt{T}_i') &= \mathtt{size}(\widehat{\mathtt{T}}_i) + A_i \cdot \Theta(2^i)\\ &= \mathtt{size}(\widehat{\mathtt{T}}_i) + \Theta(2^i) \cdot \Theta(2^i)\\ &= \mathtt{size}(\widehat{\mathtt{T}}_i) + \Theta(2^{2i}).\end{aligned}$$

In turn, this implies

$$\begin{aligned}\mathtt{size}(\mathtt{T}_i^\star) &= \mathtt{size}(\mathtt{T}_i') + \mathtt{size}(\Delta_i^\star)\\ &= \mathtt{size}(\widehat{\mathtt{T}}_i) + \Theta(2^{2i}) + \Theta(2^i)\\ &= \mathtt{size}(\widehat{\mathtt{T}}_i) + \Theta(2^{2i}).\end{aligned}$$

Thus,

$$\begin{aligned}\mathtt{size}(\widehat{\mathtt{T}}_{i+1}) &= \mathtt{size}(\mathtt{T}_i^\star) + \mathtt{size}(\mathtt{T}_i^\dagger) + \Theta(2^i)\\ &= \mathtt{size}(\widehat{\mathtt{T}}_i) + \Theta(2^{2i}) + \mathtt{size}(\widehat{\mathtt{T}}_i) + \Theta(2^i) + \Theta(2^i)\\ &= 2\mathtt{size}(\widehat{\mathtt{T}}_i) + \Theta(2^{2i}).\end{aligned}$$

It is easy to verify from this inequality that $\mathtt{size}(\widehat{\mathtt{T}}_i) = \Theta(2^{2i})$.

The next supplement demonstrates again the power of the cut rule. In Supplement 28, we saw that any closed $(\Delta_n, \mathcal{L}^c)$-tableau has a number of nodes that is exponential in the size of Δ_n. Therefore, any such tableau is of size exponential in the size of Δ_n. By contrast, we prove that the use of the cut rule allows the construction of strongly closed tableaux with cut for Δ_n whose size is polynomial in the size of Δ_n.

(38) The set Δ_n and the language $\mathcal{L}$ were defined in Exercise 9 of Chapter 4. Show that there are tableaux with cut $\mathtt{T}_0, \mathtt{T}_1, \ldots$ such that $\mathtt{T}_n$ is a strongly closed $(\Delta_n, \mathcal{L}^c)$-tableau and $\mathtt{size}(\mathtt{T}_n) = O(\mathtt{size}(\Delta_n)^2)$.

Solution: Let $\widehat{\mathtt{T}}_n$ be the strongly closed $(\widehat{\Delta}_n, \mathcal{L}^c)$-tableau from Supplement 37 of size $\Theta((2^n)^2)$. Define the tableau $\widetilde{\mathtt{T}}_n = \mathsf{s}^a_{c_1}(\widehat{\mathtt{T}}_n)$; this is a strongly closed $(\widetilde{\Delta}_n, \mathcal{L}^c)$-tableau with cut of the same asymptotic size as $\widehat{\mathtt{T}}_n$, where

$$\begin{aligned}\widetilde{\Delta}_n &= \mathsf{s}^a_{c_1}(\widehat{\Delta}_n)\\ &= \{\mathbf{T}\varphi, \mathbf{T}\psi, \mathbf{T}(\alpha \wedge \beta), \mathbf{F}(\theta_n \wedge \beta)\}.\end{aligned}$$

(We are using here the notations introduced in Exercise 9 of Chapter 4.) By Exercise 36, we can construct a strongly closed $(\Delta^{\ddagger}_n, \mathcal{L}^c)$-tableau with cut $\mathtt{T}^{\ddagger}_n$, where $\Delta^{\ddagger}_n = \{\mathbf{T}\varphi, \mathbf{T}\psi, \mathbf{T}\alpha, \mathbf{T}\beta, \mathbf{F}(\theta_n \wedge \beta)\}$, from $\widetilde{\mathtt{T}}_n$. We leave to the reader to show that $\mathtt{size}(\mathtt{T}^{\ddagger}_n) = \Theta((2^n)^2)$. By applying again the same construction to $\mathtt{T}^{\ddagger}_n$, we obtain a strongly closed $(\Delta_n, \mathcal{L}^c)$-tableau with cut $\mathtt{T}_n$ of size $\Theta((2^n)^2) = \Theta(\mathtt{size}(\Delta_n)^2)$.

Let $\mathcal{L}$ be a first-order language, V be an $\mathcal{L}$-suitable set of variables and Γ be a set of $(\mathcal{L}, V)$-formulas. A $(\Gamma, \mathcal{L}, V)$*-tableau with cut* is an unsigned tableau $\mathtt{T}$ that satisfies the following conditions:

- The root of $\mathtt{T}$ is labeled by Γ, i.e., $\mathtt{T}(\lambda) = \Gamma$.
- If q is an interior node of $\mathtt{T}$, then one of the following cases occurs:
  (1) regular expansion is used at q, or
  (2) variantizing is used at q, or
  (3) $= \in \mathcal{L}$ and equality expansion is used at q, or
  (4) q has two immediate descendants and there is an $(\mathcal{L}, V)$-formula φ such that $\mathtt{T}(q0) = \mathtt{T}(q) \cup \{\varphi\}$ and $\mathtt{T}(q1) = \mathtt{T}(q) \cup \{(\neg\varphi)\}$.

(39) Let $\mathcal{L}$ be a first-order language, V be an $\mathcal{L}$-suitable set of variables, Γ be a set of $(\mathcal{L}, V)$-formulas and $\mathtt{T}$ be a completed $(\Gamma, \mathcal{L}, V)$-tableau with cut. Prove that:

(a) If $\mathtt{T}$ is conservative, then Γ is satisfiable if and only if $\mathtt{T}$ is not closed.

(b) If T is strongly completed, then Γ is satisfiable if and only if T is not closed.

Hint. Modify the proof of Theorem 5.4.2 using the analogues for unsigned tableaux with cut of the results asked for in Exercise 19.

(40) Let $\mathcal{L}$ be a first-order language, V be an $\mathcal{L}$-suitable set of variables, Γ be a set of $(\mathcal{L}, V)$-formulas. Prove that Γ is unsatisfiable if and only if there exists a strongly closed $(\Gamma, \mathcal{L}^c, V)$-tableau with cut.
Hint. The proof parallels the one developed in Corollary 5.4.3.

(41) Let $\mathcal{L}$ be a first-order language and V be an $\mathcal{L}$-suitable set of variables.
Give a construction that starts with a strongly closed $(\Delta, \mathcal{L}, V)$-tableau with cut T and yields a strongly closed unsigned $(\mathsf{u}(\Delta), \mathcal{L}, V)$-tableau with cut.
Hint. The construction proceeds along the lines of the construction for the propositional case given in Supplement 51 of Chapter 3.

(42) Let $\mathcal{L}$ be a first-order language and V be an $\mathcal{L}$-suitable set of variables. Give a construction that starts with a strongly closed unsigned $(\Gamma, \mathcal{L}, V)$-tableau with cut T and produces a strongly closed (signed) $(\mathsf{s}(\Gamma), \mathcal{L}, V)$-tableau with cut.
Hint. The construction is similar to the construction discussed in Supplement 52 of Chapter 3.

(43) Let $\mathcal{L}$ be a first-order language and V be an $\mathcal{L}$-suitable set of variables. Give a construction that starts with a strongly closed unsigned $(\Gamma, \mathcal{L}, V)$-tableau with cut T and produces a strongly closed unsigned $(\Gamma, \mathcal{L}, V)$-tableau (without cut).
Hint. Apply the method of Supplement 53 of Chapter 3.

First-Order Sequents

(44) Let $\mathcal{L}, \mathcal{L}'$ be two first-order languages and κ be both an $(\mathcal{L}, V)$-sequent and an $(\mathcal{L}', V)$-sequent. Prove that κ is satisfiable (valid) as an $(\mathcal{L}, V)$-sequent if and only if it is satisfiable (valid) as an $(\mathcal{L}', V)$-sequent.

(45) Let x, y be two variables, $\mathcal{L}$ be a first-order language and let α be an $\mathcal{L}$-formula such that y is substitutable for x in α and y does not occur free in α.

$$\frac{((\varphi)_{x:=c} \vee \alpha_0 \vee \cdots \alpha_{n-1})}{((\forall x)\varphi \vee \alpha_0 \vee \cdots \alpha_{n-1})}\mathsf{R}_{\forall,p,(nondegenerate)}$$
where c does not occur in $\varphi, \alpha_0, \ldots, \alpha_{n-1}$ and $x \in \mathtt{FV}(\varphi)$.

$$\frac{(\varphi \vee \alpha_0 \vee \cdots \alpha_{n-1})}{((\forall x)\varphi \vee \alpha_0 \vee \cdots \alpha_{n-1})}\mathsf{R}_{\forall,p,(degenerate)}$$
where $x \notin \mathtt{FV}(\varphi)$.

$$\frac{((\neg(\forall x)\varphi) \vee (\neg(\varphi')_{x:=t}) \vee \alpha_0 \vee \cdots \alpha_{n-1})}{((\neg(\forall x)\varphi) \vee \alpha_0 \vee \cdots \alpha_{n-1})}\mathsf{R}_{\forall,n}$$
where φ' is a variant of φ such that t is substitutable for x in φ'.

$$\frac{((\exists x)\varphi \vee (\varphi')_{x:=t} \vee \alpha_0 \vee \cdots \alpha_{n-1})}{((\exists x)\varphi \vee \alpha_0 \vee \cdots \alpha_{n-1})}\mathsf{R}_{\exists,p}$$
where φ' is a variant of φ such that t is substitutable for x in φ'.

$$\frac{((\neg(\varphi)_{x:=c}) \vee \alpha_0 \vee \cdots \alpha_{n-1})}{((\neg(\exists x)\varphi) \vee \alpha_0 \vee \cdots \alpha_{n-1})}\mathsf{R}_{\exists,n,(nondegenerate)}$$
where c does not occur in $\varphi, \alpha_0, \ldots, \alpha_{n-1}$ and $x \in \mathtt{FV}(\varphi)$.

$$\frac{((\neg\varphi) \vee \alpha_0 \vee \cdots \alpha_{n-1})}{((\neg(\exists x)\varphi) \vee \alpha_0 \vee \cdots \alpha_{n-1})}\mathsf{R}_{\exists,n,(degenerate)}$$
where $x \notin \mathtt{FV}(\varphi)$.

If $=$ is in $\mathcal{L}$, include the rule
$$\frac{((\neg\alpha) \vee \alpha_0 \vee \cdots \alpha_{n-1})}{(\alpha_0 \vee \cdots \alpha_{n-1})}\mathsf{R}_{=}$$
where $n > 0$ and $\alpha \in \mathrm{INST}_{\mathcal{L},V}(\mathrm{Eq}_{=,\mathcal{L}})$.

Fig. 5.47. Set of Additional Rules of the Formal System $\mathcal{HG}_{\mathcal{L},V}$.

Give proofs in the formal system $\mathcal{F}^{\text{seq,cons}}_{\mathcal{L}^c,\text{VAR}}$ for the sequents:

(a) $(\forall y)(\alpha)_{x:=y} \Rightarrow (\forall x)\alpha$;
(b) $(\exists x)\alpha \Rightarrow (\exists y)(\alpha)_{x:=y}$.

Let $\mathcal{L}$ be a first-order language and V be an $\mathcal{L}$-suitable set of variables. We introduce the formal system $\mathcal{HG}_{\mathcal{L},V}$ as a first-order analogue of the formal system $\mathcal{HG}$ defined in propositional logic. The set of objects of this new system is $\mathrm{FORM}_{\mathcal{L}}(V)$. Its set of axioms is $\{(\varphi \vee (\neg\varphi)) \mid \varphi \in \mathrm{FORM}_{\mathcal{L}}(V)\}$, and the set of rules is obtained from the set of rules of $\mathcal{HG}$ given in Figure 3.37, excluding the rule $\mathsf{R}_{mp'}$, by replacing the propositional formulas with $\mathrm{FORM}_{\mathcal{L}}(V)$ formulas. In addition, we include the rules shown in Figure 5.47.

(46) Let $\mathcal{L}$ be a first-order language and let V be an $\mathcal{L}$-suitable set of variables. Show that the following *structural rule*

$$\frac{(\varphi_0 \vee \cdots \vee \varphi_{n-1})}{(\psi_0 \vee \cdots \vee \psi_{m-1})}\ \mathsf{R}_{\text{struc}}$$

where $\{\varphi_0, \ldots, \varphi_{n-1}\} \subseteq \{\psi_0, \ldots, \psi_{m-1}\}$, is a derived rule of $\mathcal{HG}_{\mathcal{L},V}$.

(47) Show that there is a first-order language $\mathcal{L}$ and an $\mathcal{L}$-suitable set of variables V such that $\Gamma \vdash_{\mathcal{HG}_{\mathcal{L},V}} \varphi$ but $\Gamma \not\models \varphi$, for some set of $(\mathcal{L}, V)$ formulas Γ and $(\mathcal{L}, V)$-formula φ.
Solution: Consider $\mathcal{L} = \{R, c\}$, where R is a unary relation symbol and c is a constant symbol, $V = \emptyset$, $\Gamma = \{R(c)\}$ and $\varphi = (\forall x)R(x)$. Observe that $R(c) \vdash_{\mathcal{HG}_{\mathcal{L},V}} (\forall x)R(x)$ by $\mathsf{R}_{\forall,p,(nondegenerate)}$. However, $R(c) \not\models (\forall x)R(x)$.

(48) Show that in spite of the previous supplement, the following limited version of soundness holds: if $\vdash_{\mathcal{HG}_{\mathcal{L},V}} \varphi$, then $\models \varphi$.
Solution: It is clear that the axioms of $\mathcal{HG}_{\mathcal{L},V}$ are logically valid, so it suffices to show that if the hypotheses of an instance of a rule of $\mathcal{HG}_{\mathcal{L},V}$ are logically valid, then so is the conclusion. For the rules inherited from the propositional formal system $\mathcal{HG}$, the argument of Supplement 60 of Chapter 3 carries over to show the stronger result that the hypotheses of an instance of a rule logically imply the conclusion of that instance. Thus, we need to consider only the quantifier and the equality rules. Let

$$\frac{((\varphi)_{x:=c} \vee \alpha_0 \vee \cdots \alpha_{n-1})}{(\forall x)\varphi \vee \alpha_0 \vee \cdots \alpha_{n-1})}$$

be an instance of $\mathsf{R}_{\forall,p,(nondegenerate)}$. Proceeding by contrapositive, assume that the formula $((\forall x)\varphi \vee \alpha_0 \vee \cdots \alpha_{n-1})$ is not logically valid. By Theorem 4.5.55, $(\neg((\forall x)\varphi \vee \alpha_0 \vee \cdots \vee \alpha_{n-1})) \equiv ((\neg(\forall x)\varphi) \wedge (\neg\alpha_0) \wedge \cdots \wedge (\neg\alpha_{n-1}))$ is satisfiable and hence the set

$$\{(\neg\alpha_0), \ldots, (\neg\alpha_{n-1})\} \cup \{(\neg(\forall x)\varphi)\}$$

is satisfiable. Since c does not occur in $\{\alpha_0, \cdots, \alpha_{n-1}, \varphi\}$, by Part 2 of Theorem 4.6.12, $\{(\neg\alpha_0), \ldots, (\neg\alpha_{n-1})\} \cup \{(\neg(\varphi)_{x:=c})\}$

is satisfiable, which means that $(\neg((\varphi)_{x:=c} \vee \alpha_0 \vee \cdots \vee \alpha_{n-1}))$ is satisfiable, hence $((\varphi)_{x:=c} \vee \alpha_0 \vee \cdots \alpha_{n-1})$ is not logically valid.
Let

$$\frac{(\varphi \vee \alpha_0 \vee \cdots \alpha_{n-1})}{((\forall x)\varphi \vee \alpha_0 \vee \cdots \alpha_{n-1})}$$

be an instance of $\mathsf{R}_{\forall,p,(degenerate)}$.
Since $x \notin \mathtt{FV}(\varphi)$, by Corollary 4.5.46, we have the logical equivalence $(\forall x)\varphi \equiv \varphi$, so the hypothesis of the instance logically implies the conclusion.
Let

$$\frac{((\neg(\forall x)\varphi) \vee (\neg(\varphi')_{x:=t}) \vee \alpha_0 \vee \cdots \alpha_{n-1})}{(\neg(\forall x)\varphi) \vee \alpha_0 \vee \cdots \alpha_{n-1})}$$

be an instance of $\mathsf{R}_{\forall,n}$, where φ' is a variant of φ such that t is substitutable for x in φ'. By Supplement 93 of Chapter 4, $(\forall x)\varphi \models (\varphi')_{x:=t}$, hence $(\neg(\varphi')_{x:=t}) \models (\neg(\forall x)\varphi)$. It follows that the hypothesis of the instance logically implies the conclusion.
We leave to the reader to treat the case of rules involving the existential quantifier.
If $=$ is in $\mathcal{L}$, let

$$\frac{((\neg\alpha) \vee \alpha_0 \vee \cdots \alpha_{n-1})}{(\alpha_0 \vee \cdots \alpha_{n-1})}$$

be an instance of $\mathsf{R}_{=}$, where $n > 0$ and $\alpha \in \mathrm{INST}_{\mathcal{L},V}(\mathrm{Eq}_{=,\mathcal{L}})$. By Corollary 4.6.10, α is logically valid, so the hypothesis of the instance logically implies the conclusion.

(49) Let $\mathcal{L}$ be a first-order language and V be an $\mathcal{L}$-suitable set of variables. Consider the mapping $\Lambda_{\mathcal{L},V} : \mathsf{SQT}_{\mathcal{L},V} \longrightarrow \mathcal{P}(\mathrm{FORM}_{\mathcal{L}}(V))$ defined by

$$\Lambda_{\mathcal{L},V}(\kappa) = \mathsf{u}(\overline{\mathtt{sf}_{\mathcal{L},V}(\kappa)})$$

for every $\kappa \in \mathsf{SQT}_{\mathcal{L},V}$. In other words, we have

$$\Lambda_{\mathcal{L},V}(\Gamma \Rightarrow \Gamma') = \{(\neg\varphi) \mid \varphi \in \Gamma\} \cup \Gamma'.$$

Show, by induction on the definition of proof trees, that for every proof tree $\mathtt{T} \in \mathcal{PT}_{\mathcal{F}^{\mathrm{seq,cons}}_{\mathcal{L},V}}$, if $\mathtt{T}(\lambda) = \kappa$, then there exist

$\alpha_0, \ldots, \alpha_{n-1} \in \Lambda_{\mathcal{L},V}(\kappa)$ with $n > 0$ such that

$$\vdash_{\mathcal{HG}_{\mathcal{L},V}} (\alpha_0 \vee \cdots \vee \alpha_{n-1})$$

Solution: The case when $\mathtt{T}$ is a one-node proof tree is straightforward. Let $\mathtt{T}_0, \ldots, \mathtt{T}_{n-1}$ be $\mathcal{F}^{\text{seq,cons}}_{\mathcal{L},V}$-proof trees that satisfy the condition and let $((\mathtt{T}_0(\lambda), \ldots, \mathtt{T}_{n-1}(\lambda)), \kappa)$ be an instance of a rule of $\mathcal{F}^{\text{seq,cons}}_{\mathcal{L},V}$ for some sequent κ. We intend to show that the proof tree $\mathtt{T} = (\mathtt{T}_0, \ldots, \mathtt{T}_{n-1}; \kappa)$ also satisfies the condition. If $((\mathtt{T}_0(\lambda), \ldots, \mathtt{T}_{n-1}(\lambda)), \kappa)$ is an instance of a propositional rule, the proof of Supplement 61 of Chapter 3 carries over. Suppose that

$$((\mathtt{T}_0(\lambda), \ldots, \mathtt{T}_{n-1}(\lambda)), \kappa)$$

is an instance of $\mathsf{R}_{\forall,l}$. Then, $n = 1$, $\mathtt{T}_0(\lambda)$ is $\Gamma, (\forall x)\varphi, (\varphi')_{x:=t} \Rightarrow \Gamma'$, where φ' is a variant of φ with t substitutable for x in φ', and κ is the sequent $\Gamma, (\forall x)\varphi \Rightarrow \Gamma'$. Applying the inductive hypothesis and the structural rule, we have $\vdash_{\mathcal{HG}_{\mathcal{L},V}} ((\neg(\forall x)\varphi) \vee (\neg(\varphi')_{x:=t}) \vee \alpha_0 \vee \cdots \vee \alpha_{n-1})$ for some $\alpha_0, \ldots, \alpha_{n-1}$ in $\Lambda_{\mathcal{L},V}(\Gamma \Rightarrow \Gamma')$. By Rule $\mathsf{R}_{\forall,n}$, we have

$$\vdash_{\mathcal{HG}_{\mathcal{L},V}} ((\neg(\forall x)\varphi) \vee \alpha_0 \vee \cdots \vee \alpha_{n-1}),$$

where $\{(\neg(\forall x)\varphi), \alpha_0, \ldots, \alpha_{n-1}\} \subseteq \Lambda_{\mathcal{L},V}(\kappa)$.

The remaining rules are left to the reader.

(50) Let $\mathcal{L}$ be a first-order language and V be an $\mathcal{L}$-suitable set of variables. Prove that if $\varphi \in \text{FORM}_{\mathcal{L}}(V)$ is logically valid, then $\vdash_{\mathcal{HG}_{\mathcal{L}^c,V}} \varphi$.

Solution: Since φ is logically valid, the $(\mathcal{L}, V)$ sequent $\Rightarrow \varphi$ is valid, so has an $\mathcal{F}^{\text{seq,cons}}_{\mathcal{L}^c,V}$-proof tree by Theorem 5.5.20. By Supplement 49, we have

$$\vdash_{\mathcal{HG}_{\mathcal{L}^c,V}} (\underbrace{\varphi \vee \cdots \vee \varphi}_{n \geq 1}).$$

If $n > 1$, by the structural rule, we obtain $\vdash_{\mathcal{HG}_{\mathcal{L}^c,V}} \varphi$.

First-Order Natural Deduction

(51) Let $\mathcal{L}$ be a first-order language and φ, ψ be two $\mathcal{L}$-formulas such that the variable x does not occur free in ψ.

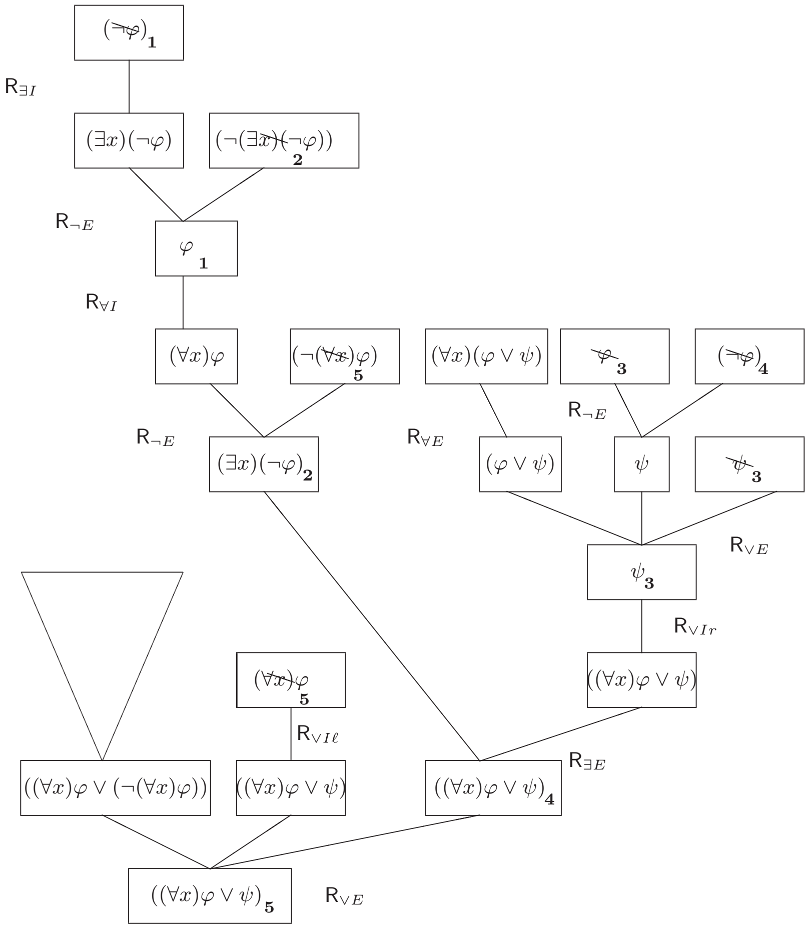

$(\neg\varphi)_1$
$R_{\exists I}$
$(\exists x)(\neg\varphi)$
$(\neg(\exists x)(\neg\varphi))_2$
$R_{\neg E}$
φ_1
$R_{\forall I}$
$(\forall x)\varphi$
$(\neg(\forall x)\varphi)_5$
$(\forall x)(\varphi \vee \psi)$
φ_3
$(\neg\varphi)_4$
$R_{\neg E}$
$R_{\neg E}$
$R_{\forall E}$
$(\exists x)(\neg\varphi)_2$
$(\varphi \vee \psi)$
ψ
ψ_3
$R_{\vee E}$
ψ_3
$R_{\vee Ir}$
$((\forall x)\varphi \vee \psi)$
$(\forall x)\varphi_5$
$R_{\vee I\ell}$
$R_{\exists E}$
$((\forall x)\varphi \vee (\neg(\forall x)\varphi))$
$((\forall x)\varphi \vee \psi)$
$((\forall x)\varphi \vee \psi)_4$
$((\forall x)\varphi \vee \psi)_5$
$R_{\vee E}$

Fig. 5.48. Natural Deduction Tree Showing $(\forall x)(\varphi \vee \psi) \overset{\bullet}{\vdash}_{\text{fond}^{\mathcal{L}}} ((\forall x)\varphi \vee \psi)$.

(a) Show that $(\forall x)(\varphi \vee \psi) \overset{\bullet}{\vdash}_{\text{fond}^{\mathcal{L}}} ((\forall x)\varphi \vee \psi)$.
(b) Show that $(\exists x)(\varphi \vee \psi) \overset{\bullet}{\vdash}_{\text{fond}^{\mathcal{L}}} ((\exists x)\varphi \vee \psi)$.

Solution: The required tree for Part (a) is given in Figure 5.48. Note that we make use of the natural deduction tree for $((\forall x)\varphi \vee (\neg(\forall x)\varphi))$ obtained in Example 5.6.6. We leave the construction of the tree for Part (b) to the reader.

(52) Let $\mathcal{L}$ be a first-order language, $\alpha_0, \ldots, \alpha_{n-1}$ be $\mathcal{L}$-formulas (for $n \geq 1$) and let ℓ be such that $0 \leq \ell \leq n-1$. Prove that

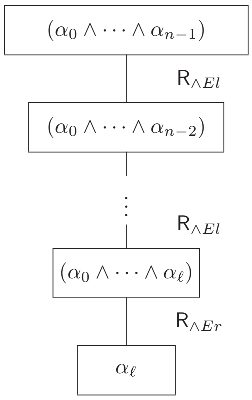

$(\alpha_0 \wedge \cdots \wedge \alpha_{n-1})$
$\mathsf{R}_{\wedge El}$
$(\alpha_0 \wedge \cdots \wedge \alpha_{n-2})$
$\mathsf{R}_{\wedge El}$
$(\alpha_0 \wedge \cdots \wedge \alpha_\ell)$
$\mathsf{R}_{\wedge Er}$
α_ℓ

Fig. 5.49. Natural deduction tree to prove that $(\alpha_0 \wedge \cdots \wedge \alpha_{n-1}) \overset{\bullet}{\vdash}_{\text{fond}^{\mathcal{L}}} \alpha_i$.

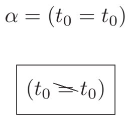


Fig. 5.50. Natural deduction tree for $(t_0 = t_0)$.

there is an effective construction of an $\mathcal{L}$-natural deduction tree $\mathcal{T} = (\mathtt{T}, M)$ with $\mathcal{UNC}(\mathcal{T}) = \{(\alpha_0 \wedge \cdots \wedge \alpha_{n-1})\}$ and $\mathtt{T}(\lambda) = \alpha_\ell$.
Solution: The desired natural deduction tree is given in Figure 5.49.

(53) Let $\mathcal{L}$ be a first-order language with equality. For each $\mathcal{L}$-instance α of an $\mathcal{L}$-equality axiom, construct a natural deduction tree $\mathcal{T} = (\mathtt{T}, M)$ for $\mathcal{L}$ with $\mathcal{UNC}(\mathcal{T}) = \emptyset$ and $\mathtt{T}(\lambda) = \alpha$, thus showing that $\emptyset \overset{\bullet}{\vdash}_{\text{fond}^{\mathcal{L}}} \alpha$.
Solution: If α is an instance $(t_0 = t_0)$ of the reflexivity axiom, the natural deduction tree is shown in Figure 5.50.
To prove the instance $((t_0 = t_1) \rightarrow (t_1 = t_0))$ of the symmetry axiom, we use the natural deduction tree from Figure 5.51.
The instance $(((t_0 = t_1) \wedge (t_1 = t_2)) \rightarrow (t_0 = t_2))$ of the transitivity axiom is proven by the natural deduction tree from Figure 5.52.
An instance of the function compatibility axiom for the function symbol f is proven as shown in Figure 5.53.

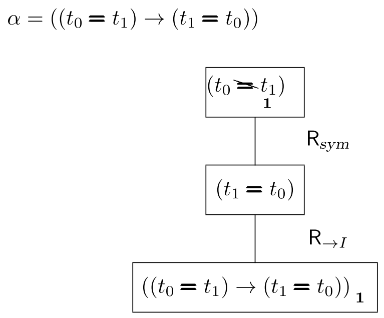

$\alpha = ((t_0 = t_1) \rightarrow (t_1 = t_0))$
$(t_0 = t_1)$ 1
R_{sym}
$(t_1 = t_0)$
$\mathsf{R}_{\rightarrow I}$
$((t_0 = t_1) \rightarrow (t_1 = t_0))$ 1

Fig. 5.51. Natural deduction tree for $((t_0 = t_1) \rightarrow (t_1 = t_0))$.

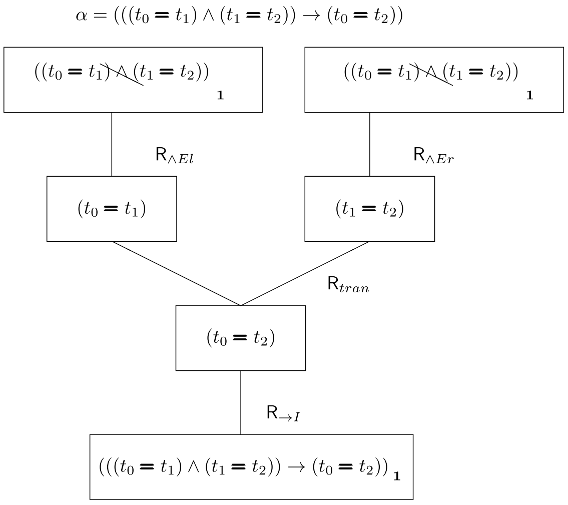

$\alpha = (((t_0 = t_1) \wedge (t_1 = t_2)) \rightarrow (t_0 = t_2))$
$((t_0 = t_1) \wedge (t_1 = t_2))$ 1
$((t_0 = t_1) \wedge (t_1 = t_2))$ 1
$\mathsf{R}_{\wedge El}$
$\mathsf{R}_{\wedge Er}$
$(t_0 = t_1)$
$(t_1 = t_2)$
R_{tran}
$(t_0 = t_2)$
$\mathsf{R}_{\rightarrow I}$
$(((t_0 = t_1) \wedge (t_1 = t_2)) \rightarrow (t_0 = t_2))$ 1

Fig. 5.52. Proof in natural deduction of an instance of the transitivity axiom of equality.

A relation symbol compatibility instance for a relation symbol R is proven by the first-order natural deduction tree shown in Figure 5.54.

(54) Let $\mathcal{L}$ be a first-order language. Give a constructive proof showing that for any $\mathcal{L}$-formulas α and β and for any set of $\mathcal{L}$-formulas Γ, if $\Gamma \overset{\bullet}{\vdash}_{\text{fond}^{\mathcal{L}}} \alpha$, then $\Gamma \cup \{((\neg\alpha) \vee \beta)\} \overset{\bullet}{\vdash}_{\text{fond}^{\mathcal{L}}} \beta$.
Solution: Let $\mathcal{T} = (\mathtt{T}, M)$ be an $\mathcal{L}$-natural deduction tree with $\mathcal{UNC}(\mathcal{T}) \subseteq \Gamma$ and $\mathtt{T}(\lambda) = \alpha$. Then, we obtain an $\mathcal{L}$-natural

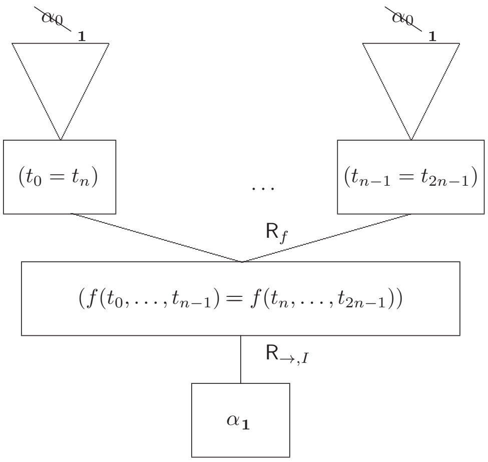


Fig. 5.53. First-order natural deduction tree for an instance of the function compatibility axiom for the function symbol f.

deduction tree $\mathcal{T}' = (\mathtt{T}', M')$ with $\mathcal{UNC}(\mathcal{T}') \subseteq \Gamma \cup \{((\neg\alpha) \vee \beta)\}$ and $\mathtt{T}'(\lambda) = \beta$ as shown in Figure 5.55.

(55) Let $\mathcal{L}$ be a first-order language, α be an instance of an equality axiom for $\mathcal{L}$ and β be an $\mathcal{L}$-formula. Prove that there is an effective construction of a first-order natural deduction tree $\mathcal{T} = (\mathtt{T}, M)$ such that $\mathcal{UNC}(\mathcal{T}) = \{((\neg\alpha) \vee \beta)\}$ and $\mathtt{T}(\lambda) = \beta$.
Hint. Use Supplements 53 and 54.

(56) Let $\mathcal{T}_0, \ldots, \mathcal{T}_{n-1}$ be marked $\mathcal{F}^{\mathcal{L}}_{\text{fond}}$-deduction trees, where $\mathcal{L}$ is a first-order language, θ be an $\mathcal{L}$-formula, and r be a node of $\mathcal{T}_i$. Show that for $\mathcal{T} = (\mathcal{T}_0, \ldots, \mathcal{T}_{n-1}; \theta)$, we have $\mathcal{UNC}(\mathcal{T}_{[ir]}) = \mathcal{UNC}((\mathcal{T}_i)_{[r]})$.
Hint. Use Exercise 94 of Chapter 1.

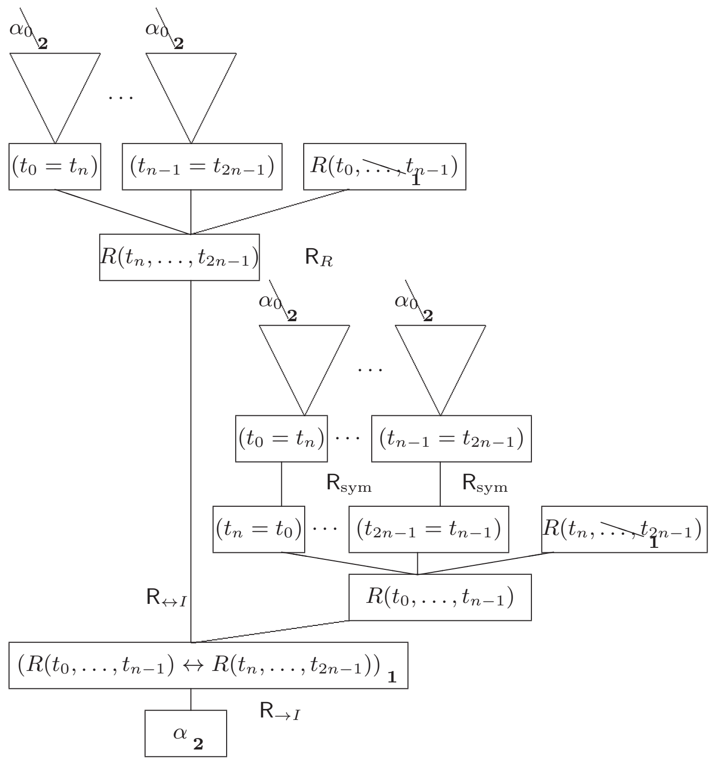
(57) Let $\mathcal{L}$ be a first-order language, $\mathcal{T}, \mathcal{T}'$ be marked $\mathcal{F}^{\mathcal{L}}_{\text{fond}}$-deduction trees, and r be a node of $\mathcal{T}$. Prove that

$$\mathcal{UNC}(\mathcal{T}[r \to \mathcal{T}']) \subseteq \mathcal{UNC}(\mathcal{T}) \cup \mathcal{UNC}(\mathcal{T}').$$

(58) Let $\mathcal{L}$ be a first-order language, $\mathcal{T}$ be a marked $\mathcal{F}^{\mathcal{L}}_{\text{fond}}$-deduction tree, r be a node of $\mathcal{T}$, and let φ be an $\mathcal{L}$-formula. Prove that

$$\mathcal{UNC}((L_\varphi(\mathcal{T}))_{[r]}) = \mathcal{UNC}(\mathcal{T}_{[r]}) - \{\varphi\}.$$

(59) Let $\mathcal{L}$ be a first-order language and let $\mathcal{T} = (\mathtt{T}, M)$ and $\mathcal{T}' = (\mathtt{T}', M')$ be two natural deduction trees of $\mathcal{L}$ and r be a node of



α_0 2
α_0 2
. . .
$(t_0 = t_n)$
$(t_{n-1} = t_{2n-1})$
$R(t_0, \ldots, t_{n-1})$ 1
$R(t_n, \ldots, t_{2n-1})$
R_R
α_0 2
α_0 2
. . .
$(t_0 = t_n)$ · · · $(t_{n-1} = t_{2n-1})$
R_{sym}
R_{sym}
$(t_n = t_0)$ · · · $(t_{2n-1} = t_{n-1})$
$R(t_n, \ldots, t_{2n-1})$ 1
$\mathsf{R}_{\leftrightarrow I}$
$R(t_0, \ldots, t_{n-1})$
$(R(t_0, \ldots, t_{n-1}) \leftrightarrow R(t_n, \ldots, t_{2n-1}))$ 1
$\mathsf{R}_{\to I}$
α 2

Fig. 5.54. First-order natural deduction tree that proves an instance of the relation compatibility axiom for R.

$\mathcal{T}$ such that $\mathtt{T}(r) = \mathtt{T}'(\lambda)$ and $\mathtt{FV}(\mathcal{UNC}(\mathcal{T}')) \subseteq \mathtt{FV}(\mathcal{UNC}(\mathcal{T}_{[r]}))$. Show that one can effectively find a marked lot $\mathcal{T}''$ that is at least as marked as $\mathcal{T}'$ such that $\mathcal{T}[r \to \mathcal{T}''] \in \text{FONDT}^{\mathcal{L}}$.

Solution: If $r = \lambda$, then $\mathcal{T}[r \to \mathcal{T}''] = \mathcal{T}''$, so we can take $\mathcal{T}'' = \mathcal{T}'$.

We proceed by induction on $\mathcal{T}$. The basis step follows immediately from the previous observation.

Suppose that $\mathsf{R}_{\forall,I}$ was used at the root of $\mathcal{T}$, so $\mathcal{T} = (\mathcal{T}_0; (\forall x)\varphi)$, where $\mathcal{T}_0 = (\mathtt{T}_0, M_0)$, $\mathtt{T}_0(\lambda) = \varphi$, and $x \notin \mathtt{FV}(\mathcal{UNC}(\mathcal{T}_0))$. The case when $r = \lambda$ is covered by the initial observation, so we may suppose that $r = 0r'$. We have $\mathtt{T}_0(r') = \mathtt{T}(0r') = \mathtt{T}(r) = \mathtt{T}'(\lambda)$ and $\mathcal{UNC}((\mathcal{T}_0)_{[r']}) = \mathcal{UNC}(\mathcal{T}_{[r]})$

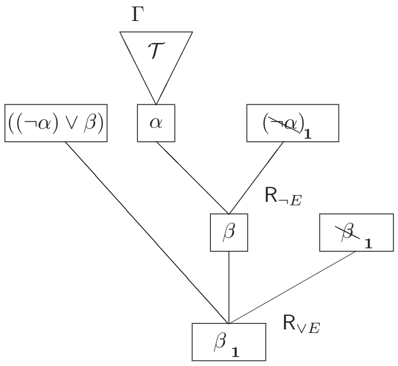


Fig. 5.55. First-order natural deduction tree $\mathcal{T}'$.

by Exercise 56. Therefore,

$$\texttt{FV}(\mathcal{UNC}(\mathcal{T}')) \subseteq \texttt{FV}(\mathcal{UNC}(\mathcal{T}_{[r]})) = \texttt{FV}(\mathcal{UNC}((\mathcal{T}_0)_{[r']})).$$

By inductive hypothesis, there is a marked lot $\mathcal{T}''$ at least as marked as $\mathcal{T}'$ such that $\mathcal{T}_0[r' \to \mathcal{T}'']$ is in FONDT$^{\mathcal{L}}$. Note that by Exercise 102 of Chapter 1, we have $\mathcal{T}[r \to \mathcal{T}''] = (\mathcal{T}_0[r' \to \mathcal{T}'']; (\forall x)\varphi)$.
By Exercise 57, we have

$$\begin{aligned}\mathcal{UNC}(\mathcal{T}_0[r' \to \mathcal{T}'']) &\subseteq \mathcal{UNC}(\mathcal{T}_0) \cup \mathcal{UNC}(\mathcal{T}'')\\ &\subseteq \mathcal{UNC}(\mathcal{T}_0) \cup \mathcal{UNC}(\mathcal{T}').\end{aligned}$$

Therefore,

$$\begin{aligned}\texttt{FV}(\mathcal{UNC}(\mathcal{T}_0[r' \to \mathcal{T}''])) &\subseteq \texttt{FV}(\mathcal{UNC}(\mathcal{T}_0)) \cup \texttt{FV}(\mathcal{UNC}(\mathcal{T}'))\\ &\subseteq \texttt{FV}(\mathcal{UNC}(\mathcal{T}_0)) \cup \texttt{FV}(\mathcal{UNC}(\mathcal{T}_{[r]}))\\ &= \texttt{FV}(\mathcal{UNC}(\mathcal{T}_0)) \cup \texttt{FV}(\mathcal{UNC}((\mathcal{T}_0)_{[r']}))\\ &= \texttt{FV}(\mathcal{UNC}(\mathcal{T}_0)).\end{aligned}$$

Thus, $x \notin \mathcal{UNC}(\mathcal{T}_0[r' \to \mathcal{T}''])$, so $\mathcal{T}[r \to \mathcal{T}''] \in$ FONDT$^{\mathcal{L}}$.
Suppose now that $\mathsf{R}_{\exists E}$ is used at the root $\mathcal{T}$, so $\mathcal{T} = (\mathcal{T}_0, L_\varphi(\mathcal{T}_1); \psi)$, where $\mathcal{T}_0 = (\texttt{T}_0, M_0)$, $\mathcal{T}_1 = (\texttt{T}_1, M_1)$, $\texttt{T}_0(\lambda) = (\exists x)\varphi$, $\texttt{T}_1(\lambda) = \psi$, $x \notin \texttt{FV}(\psi)$, and $x \notin \texttt{FV}(\mathcal{UNC}(\mathcal{T}_1) - \{\varphi\})$. We assume that $r \neq \lambda$. We deal with two cases: $r = 0r'$ and $r = 1r'$.

In the first case, $r = 0r'$, we have $\mathtt{T}_0(r') = \mathtt{T}(0r') = \mathtt{T}(r) = \mathtt{T}'(\lambda)$ and $\mathcal{UNC}((\mathcal{T}_0)_{[r']}) = \mathcal{UNC}(\mathcal{T}_{[r]})$, so

$$\mathtt{FV}(\mathcal{UNC}(\mathcal{T}')) \subseteq \mathtt{FV}(\mathcal{UNC}(\mathcal{T}_{[r]})) = \mathtt{FV}(\mathcal{UNC}((\mathcal{T}_0)_{[r']})).$$

By inductive hypothesis, there a marked lot $\mathcal{T}''$ at least as marked as $\mathcal{T}'$ such that $\mathcal{T}_0[r' \to \mathcal{T}''] \in \mathrm{FONDT}^{\mathcal{L}}$. We have $\mathcal{T}[r \to \mathcal{T}''] = (\mathcal{T}_0[r' \to \mathcal{T}''], L_\varphi(\mathcal{T}_1); \psi)$. Since $\mathtt{T}_0[r' \to \mathtt{T}''](\lambda) = (\exists x)\varphi$, we have $\mathcal{T}[r \to \mathcal{T}''] \in \mathrm{FONDT}^{\mathcal{L}}$.
In the second case, $r = 1r'$, we have $\mathtt{T}_1(r') = \mathtt{T}(1r') = \mathtt{T}(r) = \mathtt{T}'(\lambda)$. Also,

$$\begin{aligned}\mathcal{UNC}(\mathcal{T}_{[r]}) &= \mathcal{UNC}((L_\varphi(\mathcal{T}_1))_{[r']})\\ &= \mathcal{UNC}((\mathcal{T}_1)_{[r']}) - \{\varphi\}\\ &\quad \text{(by Exercise 58)}\\ &\subseteq \mathcal{UNC}((\mathcal{T}_1)_{[r']}).\end{aligned}$$

Therefore, $\mathtt{FV}(\mathcal{UNC}(\mathcal{T}')) \subseteq \mathtt{FV}(\mathcal{UNC}(\mathcal{T}_{[r]})) \subseteq \mathtt{FV}(\mathcal{UNC}((\mathcal{T}_1)_{[r']}))$.
By inductive hypothesis, there is a marked lot $\mathcal{T}''' = (\mathtt{T}''', M''')$ at least as marked as $\mathcal{T}'$ such that $\mathcal{T}_1[r' \to \mathcal{T}'''] \in \mathrm{FONDT}^{\mathcal{L}}$. Let $\mathcal{T}'' = L_\varphi(\mathcal{T}''')$. Then, $\mathcal{T}''$ is at least as marked as $\mathcal{T}'$ and we have

$$\begin{aligned}\mathcal{T}[r \to \mathcal{T}''] &= (\mathcal{T}_0, L_\varphi(\mathcal{T}_1)[r' \to \mathcal{T}''];\psi)\\ &= (\mathcal{T}_0, L_\varphi(\mathcal{T}_1)[r' \to L_\varphi(\mathcal{T}''')];\psi)\\ &= (\mathcal{T}_0, L_\varphi(\mathcal{T}_1[r' \to \mathcal{T}''']);\psi)\\ &\quad \text{(by Exercise 101 of Chapter 1).}\end{aligned}$$

Additionally $\mathtt{T}_1[r' \to \mathtt{T}'''](\lambda) = \psi$ because $T'''(\lambda) = \mathtt{T}'(\lambda)$, and

$$\begin{aligned}\mathcal{UNC}(\mathcal{T}_1[r' \to \mathcal{T}''']) - \{\varphi\} &\subseteq (\mathcal{UNC}(\mathcal{T}_1) \cup \mathcal{UNC}(\mathcal{T}''')) - \{\varphi\}\\ &\subseteq (\mathcal{UNC}(\mathcal{T}_1) \cup \mathcal{UNC}(\mathcal{T}')) - \{\varphi\}\\ &\subseteq (\mathcal{UNC}(\mathcal{T}_1) - \{\varphi\}) \cup \mathcal{UNC}(\mathcal{T}').\end{aligned}$$

Thus, we have

$$\begin{aligned}&\mathtt{FV}(\mathcal{UNC}(\mathcal{T}_1[r' \to \mathcal{T}''']) - \{\varphi\})\\ &\quad \subseteq \mathtt{FV}(\mathcal{UNC}(\mathcal{T}_1) - \{\varphi\}) \cup \mathtt{FV}(\mathcal{UNC}(\mathcal{T}')).\end{aligned}$$

We also have

$$\begin{aligned}\mathtt{FV}(\mathcal{UNC}(\mathcal{T}')) &\subseteq \mathtt{FV}(\mathcal{UNC}(\mathcal{T}_{[r]}))\\ &= \mathtt{FV}(\mathcal{UNC}((L_\varphi(\mathcal{T}_1))_{[r']}))\\ &= \mathtt{FV}(\mathcal{UNC}((\mathcal{T}_1)_{[r']}) - \{\varphi\})\\ &\subseteq \mathtt{FV}(\mathcal{UNC}(\mathcal{T}_1) - \{\varphi\}).\end{aligned}$$

Since $x \notin \mathtt{FV}(\mathcal{UNC}(\mathcal{T}_1) - \{\varphi\})$, it follows that $x \notin \mathtt{FV}(\mathcal{UNC}(\mathcal{T}_1[r' \to \mathcal{T}''']) - \{\varphi\})$, so $\mathcal{T}[r \to \mathcal{T}''] \in \mathrm{FONDT}^{\mathcal{L}}$.
We leave the other cases to the reader.

(60) Let $\mathcal{L}$ be a first-order language and let $\mathcal{T} = (\mathtt{T}, M)$ and $\mathcal{T}' = (\mathtt{T}', M')$ be two natural deduction trees of $\mathcal{L}$ and r be a node of $\mathcal{T}$ such that $\mathtt{T}(r) = \mathtt{T}'(\lambda)$ and $\mathtt{FV}(\mathcal{UNC}(\mathcal{T}')) = \emptyset$. Prove that $\mathcal{T}[r \to \mathcal{T}'] \in \mathrm{FONDT}^{\mathcal{L}}$.

(61) Let $\mathcal{L}$ be a first-order language and let φ, ψ and θ be $\mathcal{L}$-formulas such that $\varphi \overset{\bullet}{\vdash}_{\mathrm{fond}^{\mathcal{L}}} \psi$, $\psi \overset{\bullet}{\vdash}_{\mathrm{fond}^{\mathcal{L}}} \theta$ and $\mathtt{FV}(\varphi) \subseteq \mathtt{FV}(\psi)$. Prove syntactically that $\varphi \overset{\bullet}{\vdash}_{\mathrm{fond}^{\mathcal{L}}} \theta$.
Hint. Use Supplement 59.

Let $\mathcal{L}$ be a first-order language and let $\mathrm{FONDT}'^{\mathcal{L}}$ be the subset of $\mathrm{FONDT}^{\mathcal{L}}$ obtained by removing the $\neg$-introduction rule. For a set of $\mathcal{L}$-formulas Γ and $\mathcal{L}$-formula φ, we write $\Gamma \overset{\bullet}{\vdash}_{\mathrm{fond}'^{\mathcal{L}}} \varphi$ if there is a natural deduction tree $(\mathtt{T}, M) \in \mathrm{FONDT}'^{\mathcal{L}}$ such that $\mathtt{T}(\lambda) = \varphi$ and $\mathcal{UNC}(\mathtt{T}, M) \subseteq \Gamma$.

(62) Show that Supplement 59 remains valid when the natural deduction trees mentioned in the statement belong to $\mathrm{FONDT}'^{\mathcal{L}}$.

(63) Let $\mathcal{L}$ be a first-order language. Prove that for any set of $\mathcal{L}$-formulas Γ and $\mathcal{L}$-formula we have $\Gamma \overset{\bullet}{\vdash}_{\mathrm{fond}^{\mathcal{L}}} \varphi$ if and only if $\Gamma \overset{\bullet}{\vdash}_{\mathrm{fond}'^{\mathcal{L}}} \varphi$.
Hint. The argument is similar to the one given in Supplement 70 of Chapter 3.

(64) Let $\mathcal{L}$ be a first-order language. Give a recursive definition of a partial function $\Omega_0^{\mathcal{L}}$ from the set

$$\mathrm{FONDT}^{\mathcal{L}} \times \mathrm{Seq}(\mathrm{FORM}_{\mathcal{L}}) \times \mathrm{FORM}_{\mathcal{L}} \times \mathrm{FORM}_{\mathcal{L}} \times \mathbf{N} \times \mathbf{N}$$

to $\mathrm{FONDT}^{\mathcal{L}}$ such that $\Omega_0^{\mathcal{L}}((\mathtt{T}, M), \vec{\gamma}, \delta, \alpha, m, i)$ is defined if and only if $\vec{\gamma} = (\gamma_0, \ldots, \gamma_{m-1})$, $0 \leq i \leq m$, $\mathcal{UNC}(\mathtt{T}, M)$

$\subseteq \{\gamma_0, \dots, \gamma_{m-1}\}$ and $\mathtt{T}(\lambda) = \delta$, and, when defined, $\Omega_0^{\mathcal{L}}((\mathtt{T}, M), \vec{\gamma}, \delta, \alpha, m, i) = (\mathtt{T}', M')$, where $\mathcal{UNC}(\mathtt{T}', M') \subseteq \{(\gamma_0 \vee \alpha), \dots, (\gamma_{i-1} \vee \alpha), \gamma_i, \dots, \gamma_{m-1}\}$ and $\mathtt{T}'(\lambda) = (\delta \vee \alpha)$.
Hint. The definition is virtually the same as in Supplement 71 of Chapter 3.

(65) Let $\mathcal{L}$ be a first-order language. Using the partial function $\Omega_0^{\mathcal{L}}$ from Supplement 64, give a recursive definition of a partial function $\Omega_1^{\mathcal{L}}$ from the set

$$\mathrm{FONDT}^{\mathcal{L}} \times \mathrm{Seq}(\mathrm{FORM}_{\mathcal{L}}) \times \mathrm{FORM}_{\mathcal{L}} \times \mathrm{Seq}(\mathrm{FORM}_{\mathcal{L}}) \times \mathbf{N} \times \mathbf{N}$$

to $\mathrm{FONDT}^{\mathcal{L}}$ such that $\Omega_1^{\mathcal{L}}((\mathtt{T}, M), \vec{\theta}, \delta, \vec{\alpha}, m, n)$ is defined if and only if we have $\vec{\theta} = (\theta_0, \dots, \theta_{m-1})$, $\vec{\alpha} = (\alpha_0, \dots, \alpha_{n-1})$, $\mathcal{UNC}(\mathtt{T}, M) \subseteq \{\theta_0, \dots, \theta_{m-1}\}$ and $\mathtt{T}(\lambda) = \delta$, (so that $\{\theta_0, \dots, \theta_{m-1}\} \overset{\bullet}{\vdash}_{\mathrm{fond}^{\mathcal{L}}} \delta$). If defined, $\Omega_1^{\mathcal{L}}((\mathtt{T}, M), \vec{\theta}, \delta, \vec{\alpha}, m, n) = (\mathtt{T}', M')$, where

$$\mathcal{UNC}(\mathtt{T}', M') \subseteq \{(\theta_0 \vee \alpha_0 \vee \cdots \vee \alpha_{n-1}), \dots, (\theta_{m-1} \vee \alpha_0 \vee \cdots \vee \alpha_{n-1})\}$$

and $\mathtt{T}'(\lambda) = (\delta \vee \alpha_0 \vee \cdots \vee \alpha_{n-1})$ (which means that $\{(\theta_0 \vee \alpha_0 \vee \cdots \vee \alpha_{n-1}), \dots, (\theta_{m-1} \vee \alpha_0 \vee \cdots \vee \alpha_{n-1})\} \overset{\bullet}{\vdash}_{\mathrm{fond}^{\mathcal{L}}} (\delta \vee \alpha_0 \vee \cdots \vee \alpha_{n-1})$).
Hint. The solution is similar to the one of Supplement 72 of Chapter 3.

(66) Let $\mathcal{L}$ be a first-order language, $\mathcal{T} = (\mathtt{T}, M) \in \mathrm{FONDT}^{\mathcal{L}}$, c be a constant symbol in $\mathcal{L}$ and y be a variable that does not occur in $\mathtt{T}$. Define a marked lot $\mathcal{T}' = \mathsf{s}_y^c(\mathcal{T})$ as $(\mathtt{T}', M)$, where $\mathtt{T}' = \mathsf{s}_y^c \circ \mathtt{T}$. Show that $\mathsf{s}_y^c(\mathcal{T})$ is an $\mathcal{L}$-natural deduction tree.
Solution: The argument is by induction on $\mathcal{T}$. If $\mathcal{T}$ consists of one node, the statement obviously holds.
Suppose now that the natural deduction tree $\mathcal{T}$ is obtained by applying the rule $\mathsf{R}_{\wedge I}$ to the trees $\mathcal{T}_0 = (\mathtt{T}_0, M_0)$ with $\mathtt{T}(\lambda) = \varphi$ and $\mathcal{T}_1 = (\mathtt{T}_1, M_1)$ with $\mathtt{T}_1(\lambda) = \psi$ and the result holds for $\mathcal{T}_0$ and $\mathcal{T}_1$. Since $\mathcal{T} = (\mathcal{T}_0, \mathcal{T}_1; (\varphi \wedge \psi))$, we have

$$\mathsf{s}_y^c(\mathcal{T}) = (\mathsf{s}_y^c(\mathcal{T}_0), \mathsf{s}_y^c(\mathcal{T}_1); (\mathsf{s}_y^c(\varphi) \wedge \mathsf{s}_y^c(\psi))).$$

The new natural deduction tree is obtained by applying again $\mathsf{R}_{\wedge I}$ to the natural deduction trees $\mathsf{s}_y^c(\mathcal{T}_0)$ and $\mathsf{s}_y^c(\mathcal{T}_1)$.
The cases when $\mathcal{T}$ is obtained by applying $\mathsf{R}_{\wedge El}$, $\mathsf{R}_{\wedge Er}$, $\mathsf{R}_{\vee Il}$, $\mathsf{R}_{\vee Ir}$, $\mathsf{R}_{\rightarrow E}$, $\mathsf{R}_{\leftrightarrow El}$ or $\mathsf{R}_{\leftrightarrow Er}$ at the root are similar.

If $\mathsf{R}_{\forall E}$ or $\mathsf{R}_{\exists I}$ is applied at the root, the argument is still similar to the previous case, but takes into account Corollary 4.6.55 which applies because y does not occur in $\mathtt{T}$.
Suppose $\mathsf{R}_{\forall I}$ is used at the root of $\mathcal{T}$, say $\mathtt{T}(0) = \varphi$, $\mathtt{T}(\lambda) = (\forall x)\varphi$, where $x \notin \mathtt{FV}(\mathcal{UNC}(\mathcal{T}_{[0]}))$. Recall that $\mathsf{s}_y^c(\mathcal{T})$ is $\mathcal{T}' = (\mathtt{T}', M)$, so $\mathcal{T}' = (\mathsf{s}_y^c(\mathcal{T}_{[0]}); (\forall x)\mathsf{s}_y^c(\varphi))$. Then, we have $\mathtt{T}'(0) = \mathsf{s}_y^c(\varphi)$, $\mathtt{T}'(\lambda) = (\forall x)\mathsf{s}_y^c(\varphi)$. Since x occurs in $\mathtt{T}$, y is different from x, so $x \notin \mathtt{FV}(\mathcal{UNC}(\mathsf{s}_y^c(\mathcal{T}_{[0]})))$. By the inductive hypothesis, $\mathcal{T}'$ is a natural deduction tree with $\mathsf{R}_{\forall I}$ used at the root.
Suppose now that $\mathsf{R}_{\vee E}$ is used at the root of $\mathcal{T}$, say

$$\mathcal{T} = (\mathcal{T}_0, L_\varphi(\mathcal{T}_1), L_\psi(\mathcal{T}_2); \theta),$$

where $\mathtt{T}_0(\lambda) = (\varphi \vee \psi)$, $\mathtt{T}_1(\lambda) = \mathtt{T}_2(\lambda) = \theta$. Then, $\mathtt{T}'(0) = (\mathsf{s}_y^c(\varphi) \vee \mathsf{s}_y^c(\psi))$, $\mathtt{T}'(1) = \mathtt{T}'(2) = \mathtt{T}'(\lambda) = \mathsf{s}_y^c(\theta)$ and

$$\mathcal{T}' = (\mathsf{s}_y^c(\mathcal{T}_0), \mathsf{s}_y^c(L_\varphi(\mathcal{T}_1)), \mathsf{s}_y^c(L_\psi(\mathcal{T}_2)); \mathsf{s}_y^c(\theta)).$$

Since y does not occur in $\mathcal{T}$, it does not occur in LEAVES$(\mathcal{T}_1) \cup \{\varphi\}$, so by Supplement 11 of Chapter 1, s_y^c is injective on this set, which by Supplement 97 of the same chapter implies $\mathsf{s}_y^c(L_\varphi(\mathcal{T}_1)) = L_{\mathsf{s}_y^c(\varphi)}(\mathsf{s}_y^c(\mathcal{T}_1))$. Similarly, $\mathsf{s}_y^c(L_\psi(\mathcal{T}_2)) = L_{\mathsf{s}_y^c(\psi)}(\mathsf{s}_y^c(\mathcal{T}_2))$. Thus,

$$\mathcal{T}' = (\mathsf{s}_y^c(\mathcal{T}_0), L_{\mathsf{s}_y^c(\varphi)}(\mathsf{s}_y^c(\mathcal{T}_1)), L_{\mathsf{s}_y^c(\psi)}(\mathsf{s}_y^c(\mathcal{T}_2)); \mathsf{s}_y^c(\theta)),$$

and by the inductive hypothesis, $\mathcal{T}'$ is a first-order natural deduction tree with $\mathsf{R}_{\vee E}$ applied at the root.
The arguments for $\mathsf{R}_{\to I}$, $\mathsf{R}_{\leftrightarrow I}$, $\mathsf{R}_{\neg I}$ and $\mathsf{R}_{\neg E}$ are similar. The argument for $\mathsf{R}_{\exists E}$ combines techniques from $\mathsf{R}_{\forall I}$ and $\mathsf{R}_{\vee E}$.

(67) Let $\mathcal{L}$ be a first-order language, Γ be a set of $\mathcal{L}$-formulas and φ be an $\mathcal{L}$-formula. Prove constructively that if $\Gamma \overset{\bullet}{\vdash}_{\mathrm{fond}^{\mathcal{L}^c}} \varphi$, then $\Gamma \overset{\bullet}{\vdash}_{\mathrm{fond}^{\mathcal{L}}} \varphi$.
Hint. Make repeated use of Supplement 66.

Let φ, ψ be two $\mathcal{L}$-formulas. We use the notation $\varphi \equiv_{nd}^{\mathcal{L}} \psi$ to mean that $\varphi \overset{\bullet}{\vdash}_{\mathrm{fond}^{\mathcal{L}}} \psi$ and $\psi \overset{\bullet}{\vdash}_{\mathrm{fond}^{\mathcal{L}}} \varphi$.

(68) Prove that $\varphi \equiv_{nd}^{\mathcal{L}} \psi$ if and only if $\overset{\bullet}{\vdash}_{\mathrm{fond}^{\mathcal{L}}} (\varphi \leftrightarrow \psi)$.

(69) Let $\mathcal{L}$ be a first-order language. Suppose that $\varphi \equiv_{nd}^{\mathcal{L}} \varphi'$ and $\psi \equiv_{nd}^{\mathcal{L}} \psi'$. Prove that

(a) $(\neg\varphi) \equiv_{nd}^{\mathcal{L}} (\neg\varphi')$;
(b) $(\varphi C \psi) \equiv_{nd}^{\mathcal{L}} (\varphi' C \psi)$ and $(\varphi C \psi) \equiv_{nd}^{\mathcal{L}} (\varphi C \psi')$, where C is one of the binary connective symbols $\vee, \wedge, \rightarrow, \leftrightarrow$;
(c) $(Qx)\varphi \equiv_{nd}^{\mathcal{L}} (Qx)\varphi'$, where Q is one of the quantifier symbols $\forall, \exists$ and x is a variable.

Solution: By symmetry, it suffices to show just one of the two natural deduction proofs involved in each case.
For Part (a), the proof of $(\neg\varphi) \overset{\bullet}{\vdash}_{\text{fond}^{\mathcal{L}}} (\neg\varphi')$ is achieved by the first-order natural deduction tree shown in Figure 5.56.
For Part (b), we discuss only the case when C is $\rightarrow$. Figures 5.57 and 5.58 give natural deduction trees showing that

$$(\varphi \rightarrow \psi) \overset{\bullet}{\vdash}_{\text{fond}^{\mathcal{L}}} (\varphi' \longrightarrow \psi) \text{ and } (\varphi \rightarrow \psi) \overset{\bullet}{\vdash}_{\text{fond}^{\mathcal{L}}} (\varphi \longrightarrow \psi').$$

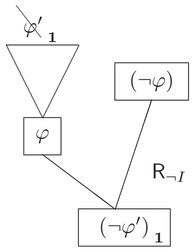


Fig. 5.56. Natural deduction tree for Part (a) of Supplement 69.

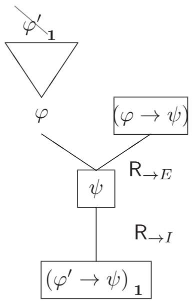


Fig. 5.57. Natural deduction tree for $(\varphi \rightarrow \psi) \overset{\bullet}{\vdash}_{\text{fond}^{\mathcal{L}}} (\varphi' \longrightarrow \psi)$.

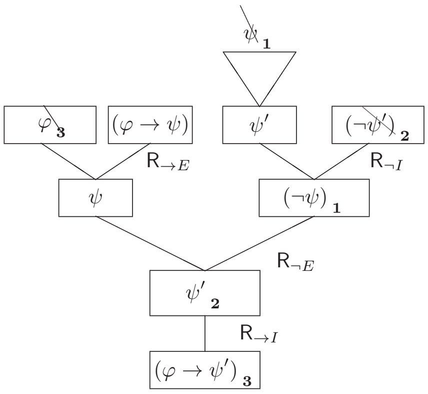


Fig. 5.58. Natural deduction tree for $(\varphi \to \psi) \overset{\bullet}{\vdash}_{\text{fond}\mathcal{L}} (\varphi \longrightarrow \psi')$.

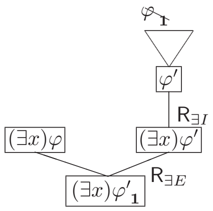


Fig. 5.59. Natural deduction tree for $(\exists x)\varphi \overset{\bullet}{\vdash}_{\text{fond}\mathcal{L}} (\exists x)\varphi'$.

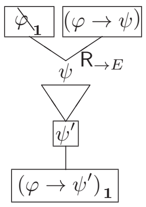


Fig. 5.60. Incorrect tree for $(\varphi \to \psi) \overset{\bullet}{\vdash}_{\text{fond}\mathcal{L}} (\varphi \longrightarrow \psi')$.

For the last part, we discuss the case when Q is $\exists$. Figure 5.59 gives a natural deduction tree showing that $(\exists x)\varphi \overset{\bullet}{\vdash}_{\text{fond}\mathcal{L}} (\exists x)\varphi'$.

(70) Why is the following tree shown in Figure 5.60 not usable to prove that $(\varphi \to \psi) \overset{\bullet}{\vdash}_{\text{fond}\mathcal{L}} (\varphi \longrightarrow \psi')$ in the previous supplement?

Solution: Supplement 59 may not apply because we may not have $\texttt{FV}(\{\varphi, (\varphi \to \psi)\}) \subseteq \texttt{FV}(\psi)$.

(71) Let $\mathcal{L}$ be a first-order language and α, β be two $\mathcal{L}$-formulas. Prove that if $\alpha \equiv_{nd}^{\mathcal{L}} \beta$ and the $\mathcal{L}$-formula ψ is obtained from the $\mathcal{L}$-formula φ by replacing an occurrence of α by β, then $\varphi \equiv_{nd}^{\mathcal{L}} \psi$.
Hint. Use induction on φ and Supplement 69.

(72) Let $\mathcal{L}$ be a first-order language and φ, ψ be two $\mathcal{L}$-formulas such that ψ is an immediate variant of φ. Prove that $\varphi \equiv_{nd}^{\mathcal{L}} \psi$ and show that natural deduction trees establishing this can be effectively constructed.
Solution: The formula ψ is obtained from the formula φ by replacing an occurrence of a subformula $(Qx)\alpha$ of φ by $(Qy)(\alpha)_{x:=y}$, where y is substitutable for x in α and does not occur free in α. If $x = y$, then $\psi = \varphi$ and the result is immediate, so we assume that $x \neq y$. By Exercise 71, it suffices to show that $(Qx)\alpha \equiv_{nd}^{\mathcal{L}} (Qy)(\alpha)_{x:=y}$.
Suppose initially that Q is $\forall$. In Figure 5.61, we show that

$$(\forall x)\alpha \overset{\bullet}{\vdash}_{\text{fond}^{\mathcal{L}}} (\forall y)(\alpha)_{x:=y}.$$

By Corollary 4.3.84, $x \notin \mathtt{FV}((\alpha)_{x:=y})$ and by Corollary 4.3.78, x is substitutable for y in $(\alpha)_{x:=y}$, so by the same argument,

$$(\forall y)(\alpha)_{x:=y} \overset{\bullet}{\vdash}_{\text{fond}^{\mathcal{L}}} (\forall x)((\alpha)_{x:=y})_{y:=x} = (\forall x)\alpha$$

where the last equality follows from Corollary 4.3.87.
In the case that Q is $\exists$, in Figure 5.62, we show that

$$(\exists x)\alpha \overset{\bullet}{\vdash}_{\text{fond}^{\mathcal{L}}} (\exists y)(\alpha)_{x:=y}.$$

The argument for the reverse direction is virtually the same as the argument for $\forall$.

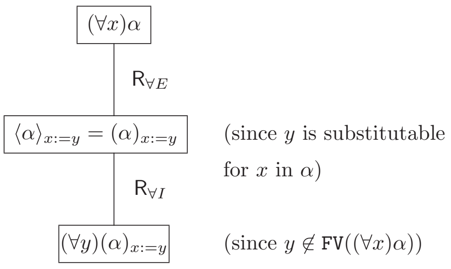

$(\forall x)\alpha$
$\mathsf{R}_{\forall E}$
$\langle\alpha\rangle_{x:=y} = (\alpha)_{x:=y}$
(since y is substitutable for x in α)
$\mathsf{R}_{\forall I}$
$(\forall y)(\alpha)_{x:=y}$
(since $y \notin \mathtt{FV}((\forall x)\alpha)$)

Fig. 5.61. Natural deduction tree for $(\forall x)\alpha \overset{\bullet}{\vdash}_{\text{fond}^{\mathcal{L}}} (\forall y)(\alpha)_{x:=y}$.

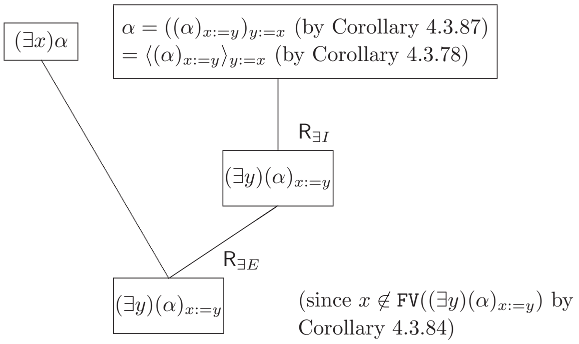

$(\exists x)\alpha$
$\alpha = ((\alpha)_{x:=y})_{y:=x}$ (by Corollary 4.3.87)
$= \langle(\alpha)_{x:=y}\rangle_{y:=x}$ (by Corollary 4.3.78)
$\mathsf{R}_{\exists I}$
$(\exists y)(\alpha)_{x:=y}$
$\mathsf{R}_{\exists E}$
$(\exists y)(\alpha)_{x:=y}$
(since $x \notin \mathtt{FV}((\exists y)(\alpha)_{x:=y})$ by Corollary 4.3.84)

Fig. 5.62. Natural deduction tree for $(\exists x)\alpha \overset{\bullet}{\vdash}_{\text{fond}^{\mathcal{L}}} (\exists y)(\alpha)_{x:=y}$.

(73) Let $\mathcal{L}$ be a first-order language and φ, ψ be two $\mathcal{L}$-formulas such that ψ is a variant of φ. Prove that $\varphi \equiv_{nd}^{\mathcal{L}} \psi$ and show that a natural deduction tree establishing this can be effectively constructed.
Solution: By the definition of variant, there is a sequence $\varphi = \varphi_0, \varphi_1, \ldots, \varphi_{n-1} = \psi$ such that for $0 \leq i \leq n-2$, φ_{i+1} is an immediate variant of φ_i, so by Supplement 72, $\varphi_i \equiv_{nd}^{\mathcal{L}} \varphi_{i+1}$. By Theorem 4.6.22, all formulas φ_i share same free variables, so by repeated use of Exercise 61, we have $\varphi \equiv_{nd}^{\mathcal{L}} \psi$.

The set of objects and the axioms of the formal system $\mathcal{FOND}'^{\mathcal{L}}$ are the same as for $\mathcal{FOND}^{\mathcal{L}}$.

The *rules for introducing connective symbols* remain the same as the rules of the propositional logic natural deduction formal system $\mathcal{ND}'$ where propositional logic objects (sets of formulas and formulas) are replaced by their first-order logic counterparts.

The *rules for introducing and eliminating quantifier symbols as well as the expansion rule in $\mathcal{FOND}'^{\mathcal{L}}$* are the same as the rules in $\mathcal{FOND}^{\mathcal{L}}$.

In addition, if $= \in \mathcal{L}$, then we include in $\mathcal{FOND}'^{\mathcal{L}}$ the following *equality rules*:

$\dfrac{(\Gamma, t_0 = t_1)}{(\Gamma, t_1 = t_0)}$	symmetry of equality
$\dfrac{(\Gamma, t_0 = t_1), (\Gamma, t_1 = t_2)}{(\Gamma, t_0 = t_2)}$	transitivity of equality
$\dfrac{(\Gamma, t_0 = t_n), \ldots, (\Gamma, t_{n-1} = t_{2n-1})}{(\Gamma, f(t_0, \ldots, t_{n-1}) = f(t_n, \ldots, t_{2n-1}))}$	f-congru- ence rule
$\dfrac{(\Gamma, t_0 = t_n), \ldots, (\Gamma, t_{n-1} = t_{2n-1}), (\Gamma, R(t_0, \ldots, t_{n-1}))}{(\Gamma, R(t_n, \ldots, t_{2n-1})}$	R-congru- ence rule

for every function symbol f of positive arity and every relation symbol R of positive arity.

(74) Prove that the theorems of $\mathcal{FOND}'^{\mathcal{L}}$ and $\mathcal{FOND}^{\mathcal{L}}$ are the same.

Transformations Between Formal Systems for First-Order Logic

(75) Formulate and prove an analogue of Theorem 5.7.11 for $\mathcal{HF}'^{\mathcal{L}}$. **Hint.** Follow the devlopments in Section 5.7.1 with $\mathcal{HF}^{\mathcal{L}}$ replaced by $\mathcal{HF}'^{\mathcal{L}}$. The analogue of Lemma 5.7.3 follows by an argument similar to that used in Supplement 76 of Chapter 3.

The formal system $\mathcal{HG}_{\mathcal{L},VAR}$ will be denoted by $\mathcal{HG}_{\mathcal{L}}$.

(76) Let $\mathcal{L}$ be a first-order language and let φ be an $\mathcal{L}$-formula. Prove that if $\vdash_{\mathcal{HG}_{\mathcal{L}}} \varphi$, then $\overset{\bullet}{\vdash}_{\text{fond}^{\mathcal{L}}} \varphi$.
Solution: The argument consists of showing that we can effectively convert an $\mathcal{HG}_{\mathcal{L}}$-proof tree `T` for φ into a natural deduction tree $\widehat{\mathcal{T}} = (\widehat{\texttt{T}}, \widehat{M})$ for φ. We proceed by induction on `T`.
For the basis step, `T` is the one-node tree labelled with $\varphi = (\theta \vee (\neg\theta))$, for some $\mathcal{L}$-formula θ. By Example 5.6.6, $\overset{\bullet}{\vdash}_{\text{fond}^{\mathcal{L}}} \varphi$.
For the inductive steps when a rule mimics a propositional logic rule, we proceed as in Supplement 78 of Chapter 3.

Suppose that $\mathsf{R}_{\forall,p,(nondegenerate)}$ is used at the root of $\mathtt{T}$, say

$$\mathtt{T}(0) = ((\varphi)_{x:=c} \vee \alpha_0 \vee \cdots \vee \alpha_{n-1})$$
$$\mathtt{T}(\lambda) = ((\forall x)\varphi \vee \alpha_0 \vee \cdots \vee \alpha_{n-1}),$$

where c does not occur in φ or in $\alpha_0, \ldots, \alpha_{n-1}$. Let $\widehat{\mathcal{T}_0}$ be the result of applying the algorithm to the subtree $\mathcal{T}_{[0]}$, so $\mathcal{UNC}(\widehat{\mathcal{T}_0}) = \emptyset$ and $\widehat{\mathtt{T}_0}(\lambda) = ((\varphi)_{x:=c} \vee \alpha_0 \vee \cdots \vee \alpha_{n-1})$. Let y be a variable that does not occur in $\widehat{\mathtt{T}_0}$ and is different from x and let $\mathcal{T}' = \mathsf{s}^c_y(\widehat{\mathcal{T}_0})$. By Supplement 66, $\mathcal{T}' \in \mathrm{FONDT}^{\mathcal{L}}$ with $\mathcal{UNC}(\mathcal{T}') = \emptyset$ and

$$\begin{aligned}\mathtt{T}'(\lambda) &= \mathsf{s}^c_y(((\varphi)_{x:=c} \vee \alpha_0 \vee \cdots \vee \alpha_{n-1}))\\ &= (\mathsf{s}^c_y((\varphi)_{x:=c}) \vee \alpha_0 \vee \cdots \vee \alpha_{n-1})\\ &\quad \text{(since } c \text{ does not occur in } \alpha_0, \ldots, \alpha_{n-1})\\ &= ((\varphi)_{x:=y} \vee \alpha_0 \vee \cdots \vee \alpha_{n-1})\\ &\quad \text{(as we argued in Theorem 5.2.22).}\end{aligned}$$

Since $y \notin \mathtt{FV}(\mathcal{UNC}(\mathcal{T}'))$, we can apply $\mathsf{R}_{\forall I}$ to obtain

$$\mathcal{T}_n = (\mathcal{T}'; (\forall y)((\varphi)_{x:=y} \vee \alpha_0 \vee \cdots \vee \alpha_{n-1}))$$

in $\mathrm{FONDT}^{\mathcal{L}}$ with $\mathcal{UNC}(\mathcal{T}_n) = \emptyset$ and $\mathtt{T}_n(\lambda) = (\forall y)((\varphi)_{x:=y} \vee \alpha_0 \vee \cdots \vee \alpha_{n-1})$.

Suppose we have $\mathcal{T}_k$ where $0 < k \leq n$ in $\mathrm{FONDT}^{\mathcal{L}}$ with $\mathcal{UNC}(\mathcal{T}_k) = \emptyset$ and $\mathtt{T}_k(\lambda) = ((\forall y)((\varphi)_{x:=y} \vee \alpha_0 \vee \cdots \vee \alpha_{k-1}) \vee \alpha_k \vee \cdots \vee \alpha_{n-1})$. Since y does not occur in $\widehat{\mathcal{T}_0}$, it follows that $y \notin \mathtt{FV}(\alpha_{k-1})$, so by Supplement 51 there is a tree $\mathcal{T}'_{k-1} \in \mathrm{FONDT}^{\mathcal{L}}$ such that $\mathcal{UNC}(\mathcal{T}'_{k-1}) = \{(\forall y)((\varphi)_{x:=y} \vee \alpha_0 \vee \cdots \vee \alpha_{k-1})\}$ and $\mathtt{T}'_{k-1}(\lambda) = ((\forall y)((\varphi)_{x:=y} \vee \alpha_0 \vee \cdots \vee \alpha_{k-2}) \vee \alpha_{k-1})$.

Using the function $\Omega^{\mathcal{L}}_1$ of Exercise 65, we define

$$\begin{aligned}\mathcal{T}''_{k-1} = \Omega^{\mathcal{L}}_1(&\mathcal{T}'_{k-1}, ((\forall y)((\varphi)_{x:=y} \vee \alpha_0 \vee \cdots \vee \alpha_{k-1})),\\ &((\forall y)((\varphi)_{x:=y} \vee \alpha_0 \vee \cdots \vee \alpha_{k-2}) \vee \alpha_{k-1}),\\ &(\alpha_k, \ldots, \alpha_{n-1}), 1, n-k),\end{aligned}$$

so

$$\mathcal{UNC}(\mathcal{T}''_{k-1}) \subseteq \{((\forall y)((\varphi)_{x:=y} \vee \alpha_0 \vee \cdots \vee \alpha_{k-1}) \vee \alpha_k \vee \cdots \vee \alpha_{n-1})\},$$

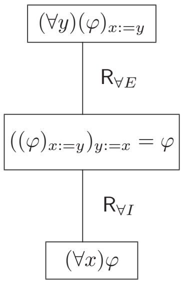


Fig. 5.63. Natural deduction tree $\mathcal{T}^*$.

and

$\mathtt{T}''_{k-1}(\lambda) = ((\forall y)((\varphi)_{x:=y} \vee \alpha_0 \vee \cdots \vee \alpha_{k-2}) \vee \alpha_{k-1} \vee \cdots \vee \alpha_{n-1})$.

Obtain $\mathcal{T}_{k-1}$ from $\mathcal{T}''_{k-1}$ by replacing each uncancelled leaf of $\mathcal{T}''_{k-1}$ with $\mathcal{T}_k$. By Exercise 60, $\mathcal{T}_{k-1} \in \mathrm{FONDT}^{\mathcal{L}}$ and we have $\mathcal{UNC}(\mathcal{T}_{k-1}) = \emptyset$ and $\mathtt{T}_{k-1}(\lambda) = ((\forall y)((\varphi)_{x:=y} \vee \alpha_0 \vee \cdots \vee \alpha_{k-2}) \vee \alpha_{k-1} \vee \cdots \vee \alpha_{n-1})$.

The end result of this process is a natural deduction tree $\mathcal{T}_0$ with $\mathcal{UNC}(\mathcal{T}_0) = \emptyset$ and $\mathtt{T}_0(\lambda) = ((\forall y)(\varphi)_{x:=y} \vee \alpha_0 \vee \cdots \vee \alpha_{n-1})$.

Since y does not occur in φ, x is substitutable for y in $(\varphi)_{x:=y}$ and $((\varphi)_{x:=y})_{y:=x} = \varphi$ by Corollaries 4.3.78 and 4.3.87. Thus, we have the natural deduction tree $\mathcal{T}^*$ given in Figure 5.63.

Define $\mathcal{T}^{**}$ as

$$\Omega_1^{\mathcal{L}}(\mathcal{T}^*, ((\forall y)(\varphi)_{x:=y}), (\forall x)\varphi, (\alpha_0, \ldots, \alpha_{n-1}), 1, n).$$

Then, $\mathcal{T}^{**} \in \mathrm{FONDT}^{\mathcal{L}}$, $\mathcal{UNC}(\mathcal{T}^{**}) \subseteq \{((\forall y)(\varphi)_{x:=y} \vee \alpha_0 \vee \cdots \alpha_{n-1})\}$ and $\mathtt{T}^{**}(\lambda) = ((\forall x)\varphi \vee \alpha_0 \vee \cdots \vee \alpha_{n-1})$. Finally, we obtain $\widehat{\mathcal{T}}$ from $\mathcal{T}^{**}$ by replacing all uncancelled leaves by $\mathcal{T}_0$. Then, $\mathcal{UNC}(\widehat{\mathcal{T}}) = \emptyset$ and $\widehat{\mathtt{T}}(\lambda) = ((\forall x)\varphi \vee \alpha_0 \vee \cdots \vee \alpha_{n-1})$.

Suppose that $\mathsf{R}_{\forall,p,(degenerate)}$ is used at the root of $\mathtt{T}$, say

$$\mathtt{T}(0) = (\varphi \vee \alpha_0 \vee \cdots \vee \alpha_{n-1})$$

$$\mathtt{T}(\lambda) = ((\forall x)\varphi \vee \alpha_0 \vee \cdots \vee \alpha_{n-1}),$$

where $x \notin \mathtt{FV}(\varphi)$.

Let $\widehat{\mathcal{T}}_0$ be the result of applying the algorithm to $\mathcal{T}_{[0]}$, so

$$\mathcal{UNC}(\widehat{\mathcal{T}}_0) = \emptyset \text{ and } \widehat{\mathtt{T}}_0(\lambda) = (\varphi \vee \alpha_0 \vee \cdots \vee \alpha_{n-1}).$$

Since $x \notin \mathtt{FV}(\varphi)$, we have $\mathcal{T}^* \in \mathrm{FONDT}^{\mathcal{L}}$ shown in Figure 5.64.

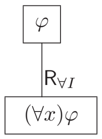


Fig. 5.64. Natural deduction tree $\mathcal{T}^*$.

Let $\mathcal{T}^{**} = \Omega_1^{\mathcal{L}}(\mathcal{T}^*, (\varphi), (\forall x)\varphi, (\alpha_0, \ldots, \alpha_{n-1}), 1, n)$. Then, it follows that we have $\mathcal{T}^{**} \in \text{FONDT}^{\mathcal{L}}$,

$$\mathcal{UNC}(\mathcal{T}^{**}) \subseteq \{(\varphi \vee \alpha_0 \vee \cdots \vee \alpha_{n-1})\},$$

and

$$\mathtt{T}^{**}(\lambda) = ((\forall x)\varphi \vee \alpha_0 \vee \cdots \vee \alpha_{n-1}).$$

The natural deduction tree $\widehat{\mathcal{T}}$ is obtained by replacing each uncancelled leaf of $\mathcal{T}^{**}$ with $\widehat{\mathcal{T}}_0$.
Suppose that $\mathsf{R}_{\forall,n}$ was used at the root of $\mathcal{T}$, say

$$\mathtt{T}(0) = ((\neg(\forall x)\varphi) \vee (\neg(\varphi')_{x:=t}) \vee \alpha_0 \vee \cdots \vee \alpha_{n-1})$$
$$\mathtt{T}(\lambda) = ((\neg(\forall x)\varphi) \vee \alpha_0 \vee \cdots \vee \alpha_{n-1}),$$

where φ' is a variant of φ such that t is substitutable for x in φ. Let $\widehat{\mathcal{T}}_0$ be the result of applying the algorithm recursively to $\mathcal{T}_{[0]}$, so $\mathcal{UNC}(\widehat{\mathcal{T}}_0) = \emptyset$ and

$$\widehat{\mathtt{T}}_0(\lambda) = ((\neg(\forall x)\varphi) \vee (\neg(\varphi')_{x:=t}) \vee \alpha_0 \vee \cdots \vee \alpha_{n-1}).$$

Let $\mathcal{T}'$ be the natural deduction tree shown in Figure 5.65.
By definition, we have $\langle\varphi\rangle_{x:=t} = (\mathsf{variant}(\varphi, x, t))_{x:=t}$. Both formulas $\mathsf{variant}(\varphi, x, t)$ and φ' are variants of φ such that t is substitutable for x in the variant. By Supplement 83 of Chapter 4, the formulas $\langle\varphi\rangle_{x:=t}$ and $(\varphi')_{x:=t}$ are variants of each other, so the formulas $((\neg(\forall x)\varphi) \vee (\neg\langle\varphi\rangle_{x:=t}))$ and $((\neg(\forall x)\varphi) \vee (\neg(\varphi')_{x:=t}))$ are variants of each other. By Supplement 73, we can find $\mathcal{T}'' \in \text{FONDT}^{\mathcal{L}}$ with $\mathcal{UNC}(\mathcal{T}'') = ((\neg(\forall x)\varphi) \vee (\neg(\varphi')_{x:=t}))$ and $\mathtt{T}''(\lambda) = ((\neg(\forall x)\varphi) \vee (\neg\langle\varphi\rangle_{x:=t}))$.
Let $\mathcal{T}'''$ be the natural deduction tree $\mathcal{T}'[0 \to \mathcal{T}'']$, so $\mathcal{UNC}(\mathcal{T}''') = \{((\neg(\forall x)\varphi) \vee (\neg(\varphi')_{x:=t}))\}$ and $\mathtt{T}''(\lambda) = (\neg(\forall x)\varphi)$.

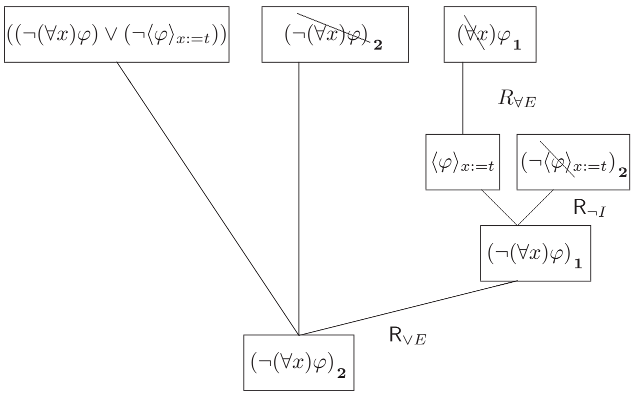

$((\neg(\forall x)\varphi) \vee (\neg\langle\varphi\rangle_{x:=t}))$
$(\neg(\forall x)\varphi)_2$
$(\forall x)\varphi_1$
$R_{\forall E}$
$\langle\varphi\rangle_{x:=t}$
$(\neg\langle\varphi\rangle_{x:=t})_2$
$R_{\neg I}$
$(\neg(\forall x)\varphi)_1$
$R_{\vee E}$
$(\neg(\forall x)\varphi)_2$

Fig. 5.65. Natural deduction tree $\mathcal{T}'$.

Let

$$\mathcal{T}^* = \Omega_1^{\mathcal{L}}(\mathcal{T}''', (((\neg(\forall x)\varphi) \vee (\neg(\varphi')_{x:=t}))),$$
$$(\neg(\forall x)\varphi), (\alpha_0, \dots, \alpha_{n-1}), 1, n).$$

Then $\widehat{\mathcal{T}}$ is obtained from $\mathcal{T}^*$ by replacing all uncancelled leaves by $\widehat{\mathcal{T}_0}$.
If one of the rules involving the existential quantifier is used at the root of $\mathtt{T}$, then the arguments are similar to the ones for the universal quantifier: the treatments of the rules $\mathsf{R}_{\exists,p}$, $\mathsf{R}_{\exists,n(nondegenerate)}$ and $\mathsf{R}_{\exists,n,(degenerate)}$ match the treatments of $\mathsf{R}_{\forall,n}$, $\mathsf{R}_{\forall,p(nondegenerate)}$ and $\mathsf{R}_{\forall,p,(degenerate)}$. However, in the case of $\mathsf{R}_{\exists,n,(nondegenerate)}$, the role of the natural deduction tree $\mathcal{T}^*$ is played by the tree shown in Figure 5.66.
Suppose that $\mathcal{L}$ contains the equality symbol $=$ and the rule $\mathsf{R}_=$ was used at the root of $\mathtt{T}$, say

$$\mathtt{T}(0) = ((\neg\alpha) \vee \alpha_0 \vee \cdots \vee \alpha_{n-1}),$$
$$\mathtt{T}(\lambda) = (\alpha_0 \vee \cdots \vee \alpha_{n-1}),$$

where $n > 0$ and $\alpha \in \mathrm{INST}_{\mathcal{L},V}(\mathrm{Eq}_{=,\mathcal{L}})$. Apply the algorithm recursively to $\mathtt{T}_{[0]}$ to obtain $\widehat{\mathcal{T}_0} \in \mathrm{FONDT}^{\mathcal{L}}$ with $\mathcal{UNC}(\widehat{\mathcal{T}_0}) = \emptyset$ and $\widehat{\mathtt{T}}_0(\lambda) = ((\neg\alpha) \vee \alpha_0 \vee \cdots \vee \alpha_{n-1})$. By Exercise 55, there is a natural deduction tree $\mathcal{T}'$ with $\mathcal{UNC}(\mathcal{T}') \subseteq \{((\neg\alpha) \vee \alpha_0)\}$ and

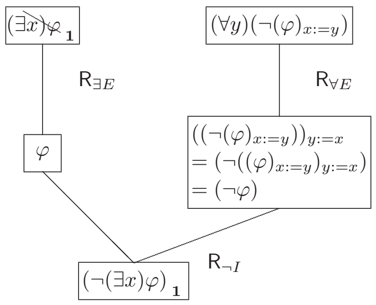

$\cancel{(\exists x)\varphi}_{\mathbf{1}}$
$(\forall y)(\neg(\varphi)_{x:=y})$
$\mathsf{R}_{\exists E}$
$\mathsf{R}_{\forall E}$
φ
$((\neg(\varphi)_{x:=y}))_{y:=x}$
$= (\neg((\varphi)_{x:=y})_{y:=x})$
$= (\neg\varphi)$
$\mathsf{R}_{\neg I}$
$(\neg(\exists x)\varphi)_{\mathbf{1}}$

Fig. 5.66. Natural deduction tree $\mathcal{T}^*$.

$\mathtt{T}'(\lambda) = \alpha_0$. Let

$$\mathcal{T}^* = \Omega_1^{\mathcal{L}}(\mathcal{T}', (((\neg\alpha) \vee \alpha_0)), \alpha_0, (\alpha_1, \dots, \alpha_{n-1}), 1, n-1),$$

so $\mathcal{UNC}(\mathcal{T}^*) = \{((\neg\alpha) \vee \alpha_0 \vee \cdots \vee \alpha_{n-1})\}$ and $\mathtt{T}^*(\lambda) = \alpha_0 \vee \cdots \vee \alpha_{n-1}$. Obtain $\widehat{\mathcal{T}}$ from $\mathcal{T}^*$ by replacing all uncancelled leaves with $\widehat{\mathcal{T}_0}$.

The goal of the next supplement is to give an alternative proof of completeness of natural deduction using the previous result, completeness of the formal system $\mathcal{HG}_{\mathcal{L}}$ and the Compactness Theorem for first-order logic.Note that usually we prove compactness starting from completeness. Here we offer an inverse approach which initially produces a somewhat weaker result because it involves the extension $\mathcal{L}^c$ of the first-order language $\mathcal{L}$.

(77) Let $\mathcal{L}$ be a first-order language, φ be an $\mathcal{L}$-formula and Γ be a set of $\mathcal{L}$-formulas. Use Supplements 50 and 76, and the Compactness Theorem to prove that if $\Gamma \models \varphi$, then $\Gamma \overset{\bullet}{\vdash}_{\text{fond}^{\mathcal{L}^c}} \varphi$.
Solution: Suppose that $\Gamma \models \varphi$. Then, by compactness, there is a finite subset $\Gamma_0 = \{\psi_0, \dots, \psi_{n-1}\}$ of Γ with $\Gamma_0 \models \varphi$. By repeated use of Part (3) of Theorem 4.5.52, we have $\models (\psi_0 \to (\psi_1 \to \cdots \to (\psi_{n-1} \to \varphi)\cdots)$. By Supplement 50, $\vdash_{\mathcal{HG}_{\mathcal{L}^c}} (\psi_0 \to (\psi_1 \to \cdots \to (\psi_{n-1} \to \varphi)\cdots)$. By Supplement 76, $\overset{\bullet}{\vdash}_{\text{fond}^{\mathcal{L}^c}} (\psi_0 \to (\psi_1 \to \cdots \to (\psi_{n-1} \to \varphi)\cdots)$. Thus, we have the natural deduction tree $\mathcal{T}$ shown in Figure 5.67 with $\mathcal{UNC}(\mathcal{T}) = \{\psi_0, \dots, \psi_{n-1}\}$ and $\mathtt{T}(\lambda) = \varphi$. Thus, $\Gamma \overset{\bullet}{\vdash}_{\text{fond}^{\mathcal{L}^c}} \varphi$.

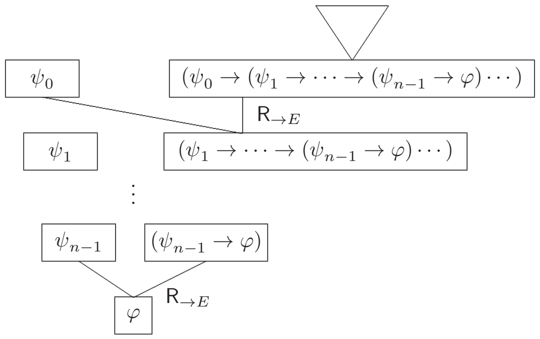

ψ_0
$(\psi_0 \to (\psi_1 \to \cdots \to (\psi_{n-1} \to \varphi)\cdots)$
$\mathsf{R}_{\to E}$
ψ_1
$(\psi_1 \to \cdots \to (\psi_{n-1} \to \varphi)\cdots)$
ψ_{n-1}
$(\psi_{n-1} \to \varphi)$
$\mathsf{R}_{\to E}$
φ

Fig. 5.67. Natural deduction tree $\mathcal{T}$.

(78) Let $\mathcal{L}$ be a first-order language, φ be an $\mathcal{L}$-formula and Γ be a set of $\mathcal{L}$-formulas. Use Supplements 77 and 67, to prove that if $\Gamma \models \varphi$, then $\Gamma \overset{\bullet}{\vdash}_{\text{fond}^{\mathcal{L}}} \varphi$.

First-Order Resolution

(79) Let C be a clause, s be a renaming of C, s' be a variable-pure bijective extension of s to VAR, and L be a subset of $s(C)$ that consists of positive literals (negative literals). If z is an mgu of L ($\overline{L}$), prove that $z * s'$ is an mgu of $\{\ell \in C \mid s(\ell) \in L\}$ (an mgu of $\overline{\{\ell \in C \mid s(\ell) \in L\}}$).
Solution: We discuss only the case when L consists of positive literals. The case for negative literals is similar. Let $H = \{\ell \in C \mid s(\ell) \in L\}$. Since $s'(H) = L$, we have $|z * s'(H)| = |z(s'(H))| = |z(L)| = 1$, so $z * s'$ is a unifier of H. Now let z_1 be an arbitrary unifier of H. Observe that $z_1 * (s')^{-1}(L) = z_1(H)$, so $z_1 * (s')^{-1}$ is a unifier of L. This implies that there is a substitution z_2 with $z_1 * (s')^{-1} = z_2 * z$, which in turn implies that $z_1 = (z_2 * z) * s' = z_2 * (z * s')$. Thus, $z * s'$ is an mgu of H.

(80) Let P, R be two relation symbols of the same arity and $\mathcal{C}$ be a set of clauses that does not contain $\square$. Prove that:

(a) $\mathsf{s}_R^P(\Gamma_{\mathcal{C}}) = \Gamma_{\mathsf{s}_R^P(\mathcal{C})}$.

(b) $\mathsf{s}_R^P(\mathcal{C}) = \mathcal{C}_{\mathsf{s}_R^P(\Gamma_{\mathcal{C}})}$.

Solution: We leave the first part to the reader. For the second part, note that $\mathcal{C}_{\mathsf{s}_R^P(\Gamma_{\mathcal{C}})} = \mathcal{C}_{\Gamma_{\mathsf{s}_R^P(\mathcal{C})}}$ by the first part. Applying Equality (5.4), we obtain the second equality.

(81) Let P, P' be two unary relation symbols and let C_0 and C_1 be the clauses $C_0 = \{P(x), P'(y)\}$, $C_1 = \{(\neg P(x)), P'(z)\}$, where x, y, z are distinct variables. Show that the clause $R = \{P'(z)\}$ is a weak most general resolvent of C_0 and C_1 and a resolvent of the same clauses, but not a most general resolvent of C_0 and C_1.

Let $\mathcal{L}$ be a first-order language and C be an $\mathcal{L}$-clause. An $\mathcal{L}$-clause C' is an *$\mathcal{L}$-factor* of C if there is a subset D of C with at least two elements such that one of the following cases occurs:

(1) D consists of positive literals and there is a $\mathcal{L}$-unifier s of D such that $C' = s(C)$;
(2) D consists of negative literals and there is a $\mathcal{L}$-unifier s of $\overline{D}$ such that $C' = s(c)$.

C' is an *mgu-factor* of C if s is an mgu of D or $\overline{D}$, for the previous two cases, respectively. We will say that C' is obtained from C by *factoring* (*mgu factoring*), respectively.

Recall that binary resolution with arbitrary unifiers was shown to be complete in Theorem 5.8.77, while binary resolution with most general unifiers is not complete as we saw in Example 5.8.74. Next we are going to show that binary resolution supplemented by factoring is complete when most general unifiers are applied. Supplement 82 is a step in this direction.

(82) Let $\mathcal{L}$ be a first-order language and let C_0, C_1 be two $\mathcal{L}$-clauses. If R is a full mgu resolvent of C_0 and C_1, then R can be obtained from C_0, C_1 by binary mgu resolution and mgu factoring.

Solution: By Definition 5.8.36, there are a standardization (s_0, s_1) of (C_0, C_1), a nonempty set of positive literals $L \subseteq \bar{s}_0(C_0)$, a nonempty set of negative literals $K \subseteq \bar{s}_1(C_1)$ and a most general unifier s of $L \cup \overline{K}$ such that

$$R = \bar{s}\left((\bar{s}_0(C_0) - L) \cup (\bar{s}_1(C_1) - K)\right).$$

By Theorem 5.8.39, we can assume that s is finite.

If both L and K consist of one literal, then R is a binary mgu resolvent of C_0, C_1. Suppose that $|L|, |K| > 1$. Since $L \cup \overline{K}$ is unifiable, both L and $\overline{K}$ are unifiable. Let z_0 be an mgu of L and z_1 be an mgu of $\overline{K}$ such that they are finite and introduce no new variables when unifying L and $\overline{K}$ respectively.

(Note that such mgus exist because the unification algorithm produces mgus having these properties.)
We claim that $z_0(L) \cup z_1(\overline{K})$ is unifiable. Since s is a unifier of L, we have $s = z_0' * z_0$, for some z_0'; similarly, since s is a unifier of $\overline{K}$, we have $s = z_1' * z_1$, for some z_1'. Since $\mathtt{V}(s_0(C_0)) \cap \mathtt{V}(s_1(C_1)) = \emptyset$, we can define

$$z(x) = \begin{cases} z_0'(x) & \text{if } x \in \mathtt{V}(s_0(C_0)) \\ z_1'(x) & \text{if } x \in \mathtt{V}(s_1(C_1)) \\ x & \text{otherwise.} \end{cases}$$

Then,

$$\begin{aligned} |z(z_0(L) \cup z_1(\overline{K}))| &= |z * z_0(L) \cup z * z_1(\overline{K})| \\ &= |z_0' * z_0(L) \cup z_1' * z_1(\overline{K})| = |s(L) \cup s(\overline{K})| \\ &= |s(L \cup \overline{K})| = 1. \end{aligned}$$

Let $\hat{z}$ be a finite mgu of $z_0(L) \cup z_1(\overline{K})$ and let

$$\tilde{z}(x) = \begin{cases} z_0(x) & \text{if } x \in \mathtt{V}(s_0(C_0)) \\ z_1(x) & \text{if } x \in \mathtt{V}(s_1(C_1)) \\ x & \text{otherwise.} \end{cases}$$

By Supplement 84 of Chapter 1, $\hat{s} = \hat{z} * \tilde{z}$ is a most general unifier of $L \cup \overline{K}$ and $\hat{s}$ is finite. By Theorem 1.6.6, $\hat{s} \equiv s$. By Part (f) of Supplement 68 of Chapter 1, we have $s = \tilde{s} * \hat{s}$, for some variable-pure finite bijection $\tilde{s}$, and $\tilde{s} * \hat{z}$ is also an mgu of $z_0(L) \cup z_1(\overline{K})$.
Let $\hat{L} = s_0^{-1}(L)$ and $\hat{K} = s_1^{-1}(K)$. Since $|L|, |K| > 1$, we have $|\hat{L}|, |\hat{K}| > 1$. Also, let s_0', s_1' be variable-pure bijective extensions of s_0, s_1, respectively, to VAR. By Supplement 79, $z_0 * s_0'$ is an mgu of $\hat{L}$ and $z_1 * s_1'$ is an mgu of $\overline{\hat{K}}$. Therefore, $z_0 * s_0(C_0) = z_0 * s_0'(C_0)$ is an mgu-factor of C_0 and $z_1 * s_1(C_1) = z_1 * s_1'(C_1)$ is an mgu-factor of C_1. We have, $z_0(L) \subseteq z_0 * s_0(C_0)$, $z_1(K) \subseteq z_1 * s_1(C_1)$, $|z_0(L)| = 1$ and $|z_1(K)| = 1$. Since $\tilde{s} * \hat{z}$ is an mgu of $z_0(L) \cup z_1(\overline{K})$, $R' = \tilde{s} * \hat{z}(z_0 * s_0(C_0) - z_0(L)) \cup \tilde{s} * \hat{z}(z_1 * s_1(C_1) - z_1(K))$

is a binary mgu resolvent of $z_0 * s_0(C_0)$ and $z_1 * s_1(C_1)$. We claim the $R' = R$.

We will prove both $R' \subseteq R$ and $R \subseteq R'$. For the first inclusion, suppose that $\ell' \in R'$. Without loss of generality, we have $\ell' = \tilde{s} * \hat{z}(z_0 * s_0(\ell_0))$ for some $\ell_0 \in C_0$ with $z_0 * s_0(\ell_0) \notin z_0(L)$, which implies that $s_0(\ell_0) \notin L$. Since z_0 and $\tilde{z}$ agree on the variables of $s_0(C_0)$, the literal ℓ' can be written as

$$\begin{aligned}\ell' &= \tilde{s} * \hat{z}(\tilde{z} * s_0(\ell_0))\\ &= \tilde{s}(\hat{z} * \tilde{z}(s_0(\ell_0)))\\ &= \tilde{s}(\hat{s}(s_0(\ell_0)))\\ &= \tilde{s} * \hat{s}(s_0(\ell_0))\\ &= s(s_0(\ell_0)).\end{aligned}$$

Since $s_0(\ell_0) \notin L$, it follows that $\ell' \in s(s_0(C_0) - L) \subseteq R$.

For the second inclusion, let $\ell \in R$. Again, without loss of generality, we can write $\ell = s(s_0(\ell_0))$, where $\ell_0 \in C_0$ and $s_0(\ell_0) \notin L$. Then,

$$\begin{aligned}\ell &= s(s_0(\ell_0))\\ &= \tilde{s} * \hat{s}(s_0(\ell_0))\\ &= \tilde{s} * \hat{z} * \tilde{z}(s_0(\ell_0))\\ &= \tilde{s} * \hat{z}(\tilde{z}(s_0(\ell_0)))\\ &= \tilde{s} * \hat{z}(z_0(s_0(\ell_0)))\\ &\quad \text{(since } \tilde{z} \text{ agrees with } z_0 \text{ on } \mathtt{V}(s_0(C_0))\\ &= \tilde{s} * \hat{z}(z_0 * s_0(\ell_0)).\end{aligned}$$

Now, if $z_0 * s_0(\ell_0) \in z_0(L)$, we would have

$$\begin{aligned}s(s_0(\ell_0)) &= z_0' * z_0(s_0(\ell_0))\\ &= z_0'(z_0 * s_0(\ell_0))\\ &\in z_0'(z_0(L))\\ &= z_0' * z_0(L) = s(L).\end{aligned}$$

Since $s_0(\ell_0) \notin L$, this would imply

$$s(s_0(C_0) - L) \cap s(L) \neq \emptyset$$

which contradicts the assumption that R is a full resolvent. Thus, $z_0 * s_0(\ell_0) \in z_0 * s_0(C_0) - z_0(L)$ and $\ell \in \tilde{s} * \hat{z}(z_0 * s_0(C_0) - z_0(L)) \subseteq R$.

We leave to the reader the cases when only one of the sets L, K consists of more than one literal.

(83) Let $\mathcal{L}$ be a first-order language and let C_0, C_1 be two $\mathcal{L}$-clauses. If R is a full mgu resolvent of C_0 and C_1, then R can be obtained from C_0, C_1 by full binary mgu resolution and mgu factoring. (Note that this statement generalizes Supplement 82.)

Solution: We use the same notation as in Supplement 82. We need to prove that $\tilde{s} * \hat{z}(z_0 * s_0(C_0) - z_0(L)) \cap \tilde{s} * \hat{z}(z_0(L)) = \emptyset$ and a similar equality for z_1, s_1 and K.

Suppose that $\ell \in \tilde{s} * \hat{z}(z_0 * s_0(C_0) - z_0(L)) \cap \tilde{s} * \hat{z}(z_0(L))$. Then, $\ell = \tilde{s} * \hat{z}(\ell_0)$ for some $\ell_0 \in z_0 * s_0(C_0) - z_0(L)$ and $\ell = \tilde{s} * \hat{z}(\ell_1)$ for some $\ell_1 \in z_0(L)$. We have

$$\ell_0 = z_0(\ell_0') \text{ for some } \ell_0' \in s_0(C_0) - L,$$

and

$$\ell_1 = z_0(\ell_1') \text{ for some } \ell_1' \in L.$$

Then, we have:

$$\begin{aligned} s(\ell_0') &= \tilde{s} * \hat{s}(\ell_0') = \tilde{s} * \hat{z} * \tilde{z}(\ell_0') \\ &= \tilde{s} * \hat{z} * z_0(\ell_0') = \tilde{s} * \hat{z}(\ell_0) \\ &= \tilde{s} * \hat{z}(\ell_1) = \tilde{s} * \hat{z} * z_0(\ell_1') \\ &= \tilde{s} * \hat{z} * \tilde{z}(\ell_1') = \tilde{s} * \hat{s}(\ell_1') = s(\ell_1'). \end{aligned}$$

This contradicts the assumption that R is a full resolvent of C_0 and C_1.

Let $\mathcal{C}$ be a set of $\mathcal{L}$-clauses, where $\mathcal{L}$ is a first-order language. Then we define

$$\begin{aligned} \mathrm{fRes}^{mgu}_{2,fact\ \mathcal{L}}(\mathcal{C}) = \mathcal{C} \cup \{R \mid R &\text{ is a full most general binary resolvent} \\ &\text{of two clauses in } \mathcal{C}\} \cup \{F \mid F \text{ is an mgu factor} \\ &\text{of a clause in } \mathcal{C}\}. \end{aligned}$$

We also define $(\mathrm{fRes}^{mgu}_{2,fact\ \mathcal{L}}(\mathcal{C}))^n$ and $(\mathrm{fRes}^{mgu}_{2,fact\ \mathcal{L}}(\mathcal{C}))^*$ in the usual way.

(84) Let $\mathcal{L}$ be a first-order language without equality and $\mathcal{C}$ be a set of $\mathcal{L}$-clauses. Prove that $\mathcal{C}$ has no model if and only if

$$\square \in (\mathrm{fRes}^{mgu}_{2,fact\ \mathcal{L}})^*(\mathcal{C}).$$

Hint. Use Theorems 5.8.23, 5.8.40, 5.8.70 and Supplement 83.

(85) Let $\mathcal{L}$ be a first-order language with equality and $\mathcal{C}$ be a set of $\mathcal{L}$-clauses. Prove that $\mathcal{C}$ has no model if and only if

$$\square \in (\mathrm{fRes}^{mgu}_{2,fact\ \mathcal{L}})^*(\mathcal{C} \cup \mathcal{C}_{=,\mathcal{L}}).$$

Hint. Use Theorems 5.8.23 and 5.8.40 to prove that if

$$\square \in (\mathrm{fRes}^{mgu}_{2,fact\ \mathcal{L}})^*(\mathcal{C} \cup \mathcal{C}_{=,\mathcal{L}}),$$

then $\mathcal{C} \cup \mathcal{C}_{=,\mathcal{L}}$ has no model, and therefore $\mathcal{C}$ has no model. Use Theorem 5.10.1 and Supplement 83 to show the converse.

Let $\mathcal{L}$ be a first-order language that contains a constant symbol. An *$\mathcal{LH}$-interpretation* (a *partial $\mathcal{LH}$-interpretation*) is a (partial) function from $\mathrm{GAFORM}_{\mathcal{L}}$ to $\{\mathbf{T}, \mathbf{F}\}$. (Note that the letter $\mathcal{H}$ is suggested by the relationship between these interpretations and Herbrand structures.)

If $\Gamma \subseteq \mathrm{GAFORM}_{\mathcal{L}}$, a partial $\mathcal{LH}$-interpretation with domain Γ is called an $\mathcal{H}$-interpretation on Γ.

If I is a partial $\mathcal{LH}$-interpretation, define Σ_I to be the set of all closed, quantifier-free $\mathcal{L}$-formulas φ such that every atomic formula that appears in φ is in $\mathrm{Dom}(I)$. We extend I to Σ_I by

$$I((\neg\psi)) = f_{\neg}(I(\psi)),$$
$$I((\psi_0 C \psi_1)) = f_C(I(\psi_0), I(\psi_1)),$$

where C is a binary connective symbol.

If $\varphi \in \Sigma_I$ and $I(\varphi) = \mathbf{T}$, we write $I \models \varphi$; if $\Gamma \subseteq \Sigma_I$, we write $I \models \Gamma$ if $I \models \varphi$ for all $\varphi \in \Gamma$.

If K is ground $\mathcal{L}$-clause and every atomic formula appearing in K belongs to $\mathrm{Dom}(I)$, define

$$I(K) = \begin{cases} \mathbf{T} & \text{if } I(\ell) = \mathbf{T} \text{ for some } \ell \in K, \\ \mathbf{F} & \text{if } I(\ell) = \mathbf{F} \text{ for all } \ell \in K. \end{cases}$$

If $I(K) = \mathbf{T}$, we write $I \models K$. If $\mathcal{K}$ is a set of ground $\mathcal{L}$-clauses such that every atomic formula that appears in $\mathcal{K}$ is in $\mathrm{Dom}(I)$, then we write $I \models \mathcal{K}$ if $I \models K$ for every $K \in \mathcal{K}$.

(86) Let $\mathcal{L}$ be a first-order language that contains at least one constant symbol and let I be a partial $\mathcal{LH}$-interpretation.

(a) Prove that if $\Gamma \subseteq \Sigma_I$ is a set of formulas in conjunctive normal form, then $I \models \Gamma$ if and only if $I \models \mathcal{C}_\Gamma$.

(b) Prove that if $\mathcal{C}$ is a set of ground $\mathcal{L}$-clauses such that $\Box \notin \mathcal{C}$ and every atomic formula that appears in any clause in $\mathcal{C}$ is in $\mathrm{Dom}(I)$, then $I \models \mathcal{C}$ if and only if $I \models \Gamma_{\mathcal{C}}$.

Let $\mathcal{L}$ be a first-order language that does not contain $=$ but contains at least one constant symbol.

The set of ground atomic $\mathcal{L}$-formulas $\mathrm{GAFORM}_{\mathcal{L}} = \{\alpha_0, \alpha_1, \ldots\}$ may be finite or infinite depending on $\mathcal{L}$. The *$\mathcal{L}$-semantic tree* $\mathtt{T}^{sem}_{\mathcal{L}}$ is the lot with domain $\{0,1\}^*$ if $\mathrm{GAFORM}_{\mathcal{L}}$ is infinite, or $\{0,1\}^{\leq n}$ if $|\mathrm{GAFORM}_{\mathcal{L}}| = n$ and for $q \in \mathrm{Dom}(\mathtt{T}^{sem}_{\mathcal{L}})$, $\mathtt{T}^{sem}_{\mathcal{L}}(q) = I_q$, where $I_q : \{\alpha_0, \ldots, \alpha_{|q|-1}\} \longrightarrow \{\mathbf{T}, \mathbf{F}\}$ is given by

$$I_q(\alpha_i) = \begin{cases} \mathbf{T} & \text{if } q(i) = 1, \\ \mathbf{F} & \text{if } q(i) = 0. \end{cases}$$

If r is a prefix of $q \in \mathrm{Dom}(\mathtt{T}^{sem}_{\mathcal{L}})$, then $I_r \subseteq I_q$, so if $\mathtt{B}$ is a branch of $\mathtt{T}^{sem}_{\mathcal{L}}$, define

$$I_{\mathtt{B}} = \bigcup_{q \in \mathtt{B}} I_q.$$

For a branch $\mathtt{B}$ of $\mathtt{T}^{sem}_{\mathcal{L}}$, define

$$S_{\mathtt{B}} = \{\alpha \in \mathrm{GAFORM}_{\mathcal{L}} \mid I_{\mathtt{B}}(\alpha) = \mathbf{T}\},$$

and let $\mathcal{A}_{\mathtt{B}}$ be the $\mathcal{L}$-Herbrand structure $\mathsf{STR}_{\mathcal{L}}(S_{\mathtt{B}})$ (using the notation from Definition 4.10.5).

(87) Let $\mathcal{L}$ be a first-order language without equality which contains a constant symbol. Prove that an $\mathcal{L}$-structure $\mathcal{A}$ is an $\mathcal{L}$-Herbrand structure if and only if there is a branch $\mathtt{B}$ of $\mathtt{T}^{sem}_{\mathcal{L}}$ such that $\mathcal{A} = \mathcal{A}_{\mathtt{B}}$.

(88) Let $\mathcal{L}$ be a first-order language without equality which contains a constant symbol. Prove that if $\mathtt{B}$ is a branch of $\mathtt{T}^{sem}_{\mathcal{L}}$ then for every ground $\mathcal{L}$-clause C, we have $\mathcal{A}_{\mathtt{B}} \models C$ if and only if $I_{\mathtt{B}}(C) = \mathbf{T}$.

Let $\mathcal{L}$ be a first-order language without equality which contains a constant symbol and let $\mathcal{C}$ be a set of ground $\mathcal{L}$-clauses. A node q of $\mathtt{T}_{\mathcal{L}}^{sem}$ is a *failure node for* $\mathcal{C}$ if there is a clause $C \in \mathcal{C}$ with $I_q(C) = \mathbf{F}$.

(89) Let $\mathcal{L}$ be a first-order language without equality which contains a constant symbol. Prove that a set of ground $\mathcal{L}$-clauses $\mathcal{C}$ has an $\mathcal{L}$-Herbrand model if and only if there is a branch $\mathtt{B}$ of $\mathtt{T}_{\mathcal{L}}^{sem}$ that does not contain a failure node for $\mathcal{C}$.

Let $\mathcal{L}$ be a first-order language without equality which contains a constant symbol and $\mathcal{C}$ be a set of ground $\mathcal{L}$-clauses. Define $\mathtt{T}^*_{\mathcal{L},\mathcal{C}}$ to be the sublot of $\mathtt{T}_{\mathcal{L}}^{sem}$ with domain

$$\{q \in \mathrm{Dom}(\mathtt{T}_{\mathcal{L}}^{sem}) \mid \text{ no proper prefix of } q \text{ is a failure node of } \mathcal{C}\}.$$

(90) Let $\mathcal{L}$ be a first-order language without equality which contains a constant symbol and let $\mathcal{C}, \mathcal{C}'$ be two sets of ground $\mathcal{L}$-clauses. Prove that:

(a) if $\mathcal{C} \subseteq \mathcal{C}'$, then $\mathtt{T}^*_{\mathcal{L},\mathcal{C}'}$ is a sublot of $\mathtt{T}^*_{\mathcal{L},\mathcal{C}}$;
(b) every interior node of $\mathtt{T}^*_{\mathcal{L},\mathcal{C}}$ has two immediate descendants.

(91) Let $\mathcal{L}$ be a first-order language without equality which contains a constant symbol, $\mathcal{C}$ be a set of ground $\mathcal{L}$-clauses and let q be a node in $\mathrm{Dom}(\mathtt{T}_{\mathcal{L}}^{sem})$ that is not a failure node for $\mathcal{C}$ such that $|q| = k$. Prove that:

(a) if $q0$ is a failure node of $\mathcal{C}$, then $\mathcal{C}$ contains a clause C such that $\alpha_k \in C$ and $I_q(C - \{\alpha_k\}) = \mathbf{F}$;
(b) if $q1$ is a failure node of $\mathcal{C}$, then $\mathcal{C}$ contains a clause C such that $(\neg\alpha_k) \in C$ and $I_q(C - \{(\neg\alpha_k)\}) = \mathbf{F}$.

(92) Let $\mathcal{L}$ be a first-order language without equality which contains a constant symbol and $\mathcal{C}$ be a set of ground $\mathcal{L}$-clauses that has no $\mathcal{L}$-Herbrand model. Prove that:

(a) $\mathtt{T}^*_{\mathcal{L},\mathcal{C}}$ is a finite lot;
(b) every leaf of $\mathtt{T}^*_{\mathcal{L},\mathcal{C}}$ is failure node for $\mathcal{C}$.

Solution: For Part (a), suppose that $\mathtt{T}^*_{\mathcal{L},\mathcal{C}}$ is infinite. By König's Lemma, $\mathtt{T}^*_{\mathcal{L},\mathcal{C}}$ has an infinite branch $\mathtt{B}$ which is also a branch of $\mathtt{T}_{\mathcal{L}}^{sem}$. By the definition of $\mathtt{T}^*_{\mathcal{L},\mathcal{C}}$, $\mathtt{B}$ does not contain a failure node for $\mathcal{C}$, so by Exercise 89 $\mathcal{C}$ has an $\mathcal{L}$-Herbrand model, which is a contradiction.

For Part (b), suppose that q is a leaf of $\mathtt{T}^*_{\mathcal{L},\mathcal{C}}$ that is not a failure node for $\mathcal{C}$. By the definition of $\mathtt{T}^*_{\mathcal{L},\mathcal{C}}$, no proper prefix of q is a failure node for $\mathcal{C}$. If q is not a leaf of $\mathtt{T}^{sem}_{\mathcal{L}}$, then $q0$ and $q1$ are nodes of $\mathtt{T}^*_{\mathcal{L},\mathcal{C}}$ contradicting the assumption that q is a leaf of $\mathtt{T}^*_{\mathcal{L},\mathcal{C}}$. If q is a leaf of $\mathtt{T}^{sem}_{\mathcal{L}}$, then the branch of $\mathtt{T}^{sem}_{\mathcal{L}}$ ending in q contains no failure node, so by Exercise 89 $\mathcal{C}$ has an $\mathcal{L}$-Herbrand model, which again is a contradiction.

(93) Let $\mathcal{L}$ be a first-order language without equality. Prove that if $\mathcal{C}$ is a set of $\mathcal{L}$-clauses that has no model, then $\Box \in (\mathrm{fRes}^{mgu})^*(\mathcal{C})$.
Solution: Let $\mathcal{L}' = \mathcal{H}(\mathcal{L})$, the Herbrand extension of $\mathcal{L}$. As shown in the proof of Theorem 5.8.70, $\mathrm{GINST}_{\mathcal{L}'}(\mathcal{C})$ has no model, if $\Box \notin \mathcal{C}$, and clearly the same holds if $\Box \in \mathcal{C}$. In particular no $\mathcal{L}'$-Herbrand model exists for this set, so $\mathtt{T}^*_{\mathcal{L}',\mathrm{GINST}_{\mathcal{L}'}(\mathcal{C})}$ is finite by Supplement 92.
We proceed by induction on $n = |\mathtt{T}^*_{\mathcal{L}',\mathrm{GINST}_{\mathcal{L}'}(\mathcal{C})}|$ to prove that $\Box \in (\mathrm{fRes}^{mgu})^*(\mathcal{C})$.
The basis step, $n = 1$ is immediate because $\Box \in \mathrm{GINST}_{\mathcal{L}'}(\mathcal{C})$, so $\Box \in \mathcal{C}$.
For the inductive step, suppose that $n > 1$ and the result holds for all $n' < n$. Let q be a node of depth 1 of $\mathtt{T}^*_{\mathcal{L}',\mathrm{GINST}_{\mathcal{L}'}(\mathcal{C})}$ and $|q| = k$. Then by Supplement 92, the leaves $q0$ and $q1$ are both failure nodes for $\mathcal{C}$ and q is not, so there are clauses $C_0, C_1 \in \mathcal{C}$ and ground $\mathcal{L}'$-substitutions s_0, s_1 such that $\alpha_k \in s_0(C_0)$, $(\neg\alpha_k) \in s_1(C_1)$ and

$$I_q((s_0(C_0) - \{\alpha_k\}) \cup (s_1(C_1) - \{(\neg\alpha_k)\})) = \mathbf{F}.$$

Note that q has two immediate descendants by Exercise 90. The existence of the clauses and substitutions mentioned above follows from Exercise 91.
Let $R' = (s_0(C_0) - \{\alpha_k\}) \cup (s_1(C_1) - \{(\neg\alpha_k)\})$. Then, R' is a resolvent of $s_0(C_0)$ and $s_1(C_1)$ and $I_q(R') = \mathbf{F}$. By Lemma 5.8.67, there is a full mgu resolvent R of C_0 and C_1 and a ground $\mathcal{L}'$-substitution s such that $R' = s(R)$. Since $\mathrm{GINST}_{\mathcal{L}'}(\mathcal{C}) \subseteq \mathrm{GINST}_{\mathcal{L}'}(\mathcal{C} \cup \{R\})$, by Exercise 90 $\mathtt{T}^*_{\mathcal{L}',\mathcal{C}\cup\{R\}}$ is a sublot of $\mathtt{T}^*_{\mathcal{L}',\mathcal{C}}$. Since $R' = s(R)$, q is a failure node for $\mathcal{C}\cup\{R\}$, so $q0, q1 \notin \mathrm{Dom}(\mathtt{T}^*_{\mathcal{L}',\mathrm{GINST}_{\mathcal{L}'}(\mathcal{C}\cup\{R\})})$ and $|\mathtt{T}^*_{\mathcal{L}',\mathrm{GINST}_{\mathcal{L}'}(\mathcal{C}\cup\{R\})}| < n$. By inductive hypothesis,

$$\Box \in (\mathrm{fRes}^{mgu})^*(\mathcal{C} \cup \{R\}) = (\mathrm{fRes}^{mgu})^*(\mathcal{C}).$$

Variations of First-Order Resolution

(94) Let $\mathcal{C}$ be a set of $\mathcal{L}$-clauses and let P and R be two relation symbols of equal arity such that $P \in \mathcal{L}$ and $R \notin \mathcal{L}$. Define the first-order language $\mathcal{L}' = (\mathcal{L} - \{P\}) \cup \{R\}$. Prove that if $(C_0, \ldots, C_{n-1})$ is a positive (negative, linear, input) $\mathcal{L}$-resolution (most general $\mathcal{L}$-resolution, full $\mathcal{L}$-resolution, full most general $\mathcal{L}$-resolution) proof of C over $\mathcal{C}$, then $(\mathsf{s}^P_R(C_0), \ldots, \mathsf{s}^P_R(C_{n-1}))$ is the same type of proof of $\mathsf{s}^P_R(C)$ over $\mathsf{s}^P_R(\mathcal{C})$ with $\mathcal{L}$ replaced by $\mathcal{L}'$.
Hint. Use the proof of Theorem 5.8.50.

(95) Let $\mathcal{C}$ be a set of $\mathcal{L}$-clauses and let P and R be two relation symbols of equal arity such that $P \in \mathcal{L}$ and $R \notin \mathcal{L}$. Let $\mathcal{A}$ be an $\mathcal{L}$-structure and assume that if R is $=$ then $P^{\mathcal{A}} = \{(a,a) \mid a \in |\mathcal{A}|\}$. Define the first-order language $\mathcal{L}' = (\mathcal{L} - \{P\}) \cup \{R\}$. Prove that if $(C_0, \ldots, C_{n-1})$ is an $\mathcal{A}$-semantic $\mathcal{L}$-resolution (most general $\mathcal{L}$-resolution, full $\mathcal{L}$-resolution, full most general $\mathcal{L}$-resolution) proof of C over $\mathcal{C}$, then $(\mathsf{s}^P_R(C_0), \ldots, \mathsf{s}^P_R(C_{n-1}))$ is an $\mathcal{A}_{(P\to R)}$-semantic resolution proof of the same type of $\mathsf{s}^P_R(C)$ over $\mathsf{s}^P_R(\mathcal{C})$ with $\mathcal{L}$ replaced by $\mathcal{L}'$.
Hint. Use the proof of Theorem 5.8.50 and a clausal version of Theorem 4.6.1.

First-Order Resolution with Equality

Here we extend the Soundness and Completeness of Resolution for Languages with Equality (Theorem 5.10.1) to include positive, negative, and linear resolution proofs.

(96) Let $\mathcal{L}$ be a first-order language with equality and let $\mathcal{C}$ be a set of $\mathcal{L}$-clauses. Prove that if $\mathcal{R}$ is one of $\mathrm{Res}, \mathrm{Res}^{mgu}, \mathrm{fRes}$ or fRes^{mgu}, then $\mathcal{C}$ has no model if and only if the membership $\Box \in \mathcal{R}^*_{\mathcal{L}}(\mathcal{C} \cup \mathcal{C}_{=,\mathcal{L}})$ can be established using a positive (negative, linear) resolution proof.
Solution: We may assume that $\Box \notin \mathcal{C}$, for if $\Box \in \mathcal{C}$, the conclusion follows immediately.
Let R be a binary relation symbol not in $\mathcal{L}$ and let $\mathcal{L}'$ be the language $(\mathcal{L} - \{=\}) \cup \{R\}$.
The chain of equivalent statements 1 to 6 established in the proof of Theorem 5.10.1 remains the same. Statement 6 is equivalent to

7′. there exists a positive (negative, linear) resolution proof showing $\square \in \mathcal{R}^*_{\mathcal{L}'}(\mathsf{s}^{=}_{R}(\mathcal{C}) \cup \mathcal{C}_{\mathrm{MEq}^{\dagger}_{R,\mathcal{L}'}})$,

by soundness and by Theorem 5.9.6. In turn, statement 7′ is equivalent to

8′. there exists a positive (negative, linear) resolution proof showing that $\square \in \mathcal{R}^*_{\mathcal{L}}(\mathcal{C} \cup \mathcal{C}_{=,\mathcal{L}})$,

by Exercise 94 and the Equalities (5.8).

(97) Let $\mathcal{L}$ be a first-order language with equality and let $\mathcal{C}$ be a set of Horn $\mathcal{L}$-clauses. Prove that if $\mathcal{R}$ is one of $\mathrm{Res}, \mathrm{Res}^{mgu}, \mathrm{fRes}$ or fRes^{mgu}, then $\mathcal{C}$ has no model if and only if the membership $\square \in \mathcal{R}^*_{\mathcal{L}}(\mathcal{C} \cup \mathcal{C}_{=,\mathcal{L}})$ can be established using an input resolution proof.

Hint. Apply the previous proof noting that $\mathsf{s}^{=}_{R}(\mathcal{C}) \cup \mathcal{C}_{\mathrm{MEq}^{\dagger}_{R,\mathcal{L}'}}$ consists of Horn clauses.

(98) Let $\mathcal{L}$ be a first-order language with equality, $\mathcal{C}$ be a set of $\mathcal{L}$-clauses, and $\mathcal{A}$ be an $\mathcal{L}$-structure. Prove that if $\mathcal{R}$ is one of $\mathrm{Res}, \mathrm{Res}^{mgu}, \mathrm{fRes}$ or fRes^{mgu}, then $\mathcal{C}$ has no model if and only if the membership $\square \in \mathcal{R}^*_{\mathcal{L}}(\mathcal{C} \cup \mathcal{C}_{=,\mathcal{L}})$ can be established using an $\mathcal{A}$-semantic resolution proof.

Solution: Begin by observing that $(\mathcal{A}_{=\to R})_{R\to =} = \mathcal{A}$. The statement

7″ there exists an $\mathcal{A}_{(=\to R)}$-semantic resolution proof showing $\square \in \mathcal{R}^*_{\mathcal{L}'}(\mathsf{s}^{=}_{R}(\mathcal{C}) \cup \mathcal{C}_{\mathrm{MEq}^{\dagger}_{R,\mathcal{L}'}})$,

is equivalent to Statement 6 of Theorem 5.10.1 by soundness and by Theorem 5.9.6. In turn, statement 7″ is equivalent to

8″ there exists an $\mathcal{A}$-semantic resolution proof showing that $\square \in \mathcal{R}^*_{\mathcal{L}}(\mathcal{C} \cup \mathcal{C}_{=,\mathcal{L}})$,

by Exercise 95, the initial observation, and the Equalities (5.8).

Let $\mathcal{L}$ be a language with equality and C be an $\mathcal{L}$-clause. An $\mathcal{L}$-clause D is obtained from C by *$\mathcal{L}$-irreflexivity removal* if there is a nonempty set $L \subseteq C$ of inequalities, $L = \{(\neg(t_i = u_i)) \mid 0 \leq i \leq n-1\}$ and an $\mathcal{L}$-unifier s of $\{t_i \mid 0 \leq i \leq n-1\} \cup \{u_i \mid 0 \leq i \leq n-1\}$ such that $D = s(C - L)$. If $s(C-L) \cap s(L) = \emptyset$, we say that D is obtained from C by *full $\mathcal{L}$-irreflexivity removal*, and if s is a most general unifier of

$\{t_i \mid 0 \leq i \leq n-1\} \cup \{u_i \mid 0 \leq i \leq n-1\}$, then we say that D is obtained from C by *most general irreflexivity removal.*

(99) Let $\mathcal{L}$ be a first-order language with equality and C be an $\mathcal{L}$-clause. If D is obtained from C by $\mathcal{L}$-irreflexivity removal and $\mathcal{A}$ is an $\mathcal{L}$-structure, then prove that $\mathcal{A} \models C$ implies $\mathcal{A} \models D$.
Solution: Let $D = s(C - L)$, where L is as above. Since $\mathcal{A} \models C$, by Theorem 5.8.23, we have $\mathcal{A} \models s(C)$. Let $\sigma \in \text{ASSIGN}_{\mathcal{A}}$. Since $\mathcal{A} \models s(C)$, for some $\ell' \in C$, $(\mathcal{A}, \sigma) \models \ell$ where $\ell = s(\ell')$. If $\ell' \in L$, then $\ell = s(\ell')$ has the form $(\neg(t = t))$ and $(\mathcal{A}, \sigma) \models \ell$ is impossible, so $\ell' \in C - L$ and $\ell = s(\ell') \in s(C - L) = D$. Thus, $(\mathcal{C}, \sigma) \models D$. Since this is true for all $\sigma \in \text{ASSIGN}_{\mathcal{A}}$, it follows that $\mathcal{A} \models D$.

(100) Let $\mathcal{L}$ be a first-order language with equality, C_0 be a ground $\mathcal{L}$-clause and D be obtained from C_0 by $\mathcal{L}$-inequality removal. Suppose that C_0' is an $\mathcal{L}$-clause and s_0 is a ground $\mathcal{L}$-substitution with $s_0(C_0') = C_0$. Prove that there is an $\mathcal{L}'$-clause D' obtained from C_0' by full, most general irreflexivitiy removal and a ground $\mathcal{L}$-substitution s such that $s(D') = D$.
Solution: Since C_0 is a ground clause, there is a ground $\mathcal{L}$-term t with $(\neg(t = t)) \in C_0$ and $D = C_0 - \{(\neg(t = t))\}$. Let $L = s_0^{-1}(\neg(t = t)) \cap C_0'$. Then, L is a nonempty set of $\mathcal{L}$-inequalities $L = \{(\neg(t_i = u_i)) \mid 0 \leq i \leq n-1\}$ for some $n > 0$, and s_0 is an $\mathcal{L}$-unifier of

$$\{t_i \mid 0 \leq i \leq n-1\} \cup \{u_i \mid 0 \leq i \leq n-1\}.$$

Let s_0^m be a most general unifier of this set. There is an $\mathcal{L}$-substitution s with $s_0 = s{*}s_0^m$. Let $D' = s_0^m(C_0' - L)$. Then, D' is obtained from C_0' by most general irreflexivity removal. If $s_0^m(\ell_0) = s_0^m(\ell_1)$ with $\ell_0 \in C_0' - L$ and $\ell_1 \in L$, then we have

$$s_0(\ell_0) = s * s_0^m(\ell_0) = s * s_0^m(\ell_1) = s_0(\ell_1) = (\neg(t = t)),$$

contradicting the fact that $\ell_0 \notin L$. Thus, D' is obtained by full, most general irreflexivity removal from C_0'. Finally, we have

$$\begin{aligned} s(D') &= s(s_0^m(C_0' - L)) = s_0(C_0' - L) \\ &= s_0(C_0') - s_0(L) = C_0 - \{(\neg(t = t))\} = D, \end{aligned}$$

where the third equality is implied by (5.7).

Let $\mathcal{L}$ be a first-order language with equality and let C be an $\mathcal{L}$-clause. An $\mathcal{L}$-clause D is obtained from C by *equality reversal* if there is an equality $(t = u) \in C$ and $D = (C - \{(t = u)\}) \cup \{(u = t)\}$.

(101) Let $\mathcal{L}$ be a first-order language with equality and C be an $\mathcal{L}$-clause. If D is obtained from C by equality reversal and $\mathcal{A}$ is an $\mathcal{L}$-structure, prove that $\mathcal{A} \models C$ implies $\mathcal{A} \models D$.

(102) Let $\mathcal{L}$ be a first-order language with equality, C_0 be a ground $\mathcal{L}$-clause and let D be obtained from C by equality reversal. Suppose that C_0' is an $\mathcal{L}$-clause and s_0 is a ground $\mathcal{L}$-substitution with $s_0(C_0') = C_0$. Prove that there is clause D' obtained from C_0' by one or more applications of equality reversal such that $s_0(D') = D$.
Solution: Let $D = (C_0 - \{(t = u)\}) \cup \{(u = t)\}$ and $L = s_0^{-1}(t = u) \cap C_0'$. Then, L is a set of one or more equalities and $D' = (C_0' - L) \cup \{(u' = t') \mid (t' = u') \in L\}$ is obtained from C_0' by one or more equality reversals. We have

$$\begin{aligned} s_0(D') &= (s_0(C_0') - s_0(L)) \cup s_0(\{(u' = t') \mid (t' = u') \in L\}) \\ &= (C_0 - \{(t = u)\}) \cup \{(u = t)\} = D. \end{aligned}$$

Let $\mathcal{L}$ be a first-order language with equality and let $\mathcal{C}$ be a set of $\mathcal{L}$-clauses. Define

$$\begin{aligned} &\text{fRespar-ir-er}^{mgu}(\mathcal{C}) \\ &\quad = \text{fRespar}^{mgu}(\mathcal{C}) \cup \{D \mid D \text{ is obtained by full, most general} \\ &\qquad \text{irreflexivity removal from some clause } C \in \mathcal{C}\} \cup \{D \mid D \text{ is} \\ &\qquad \text{obtained by equality reversal from some clause } C \in \mathcal{C}\}. \end{aligned}$$

Define $(\text{fRespar-ir-er}^{mgu})^n(\mathcal{C})$ and $(\text{fRespar-ir-er}^{mgu})^*(\mathcal{C})$ as usual, that is similarly to $(\text{fRespar}^{mgu})^n(\mathcal{C})$ and $(\text{fRespar}^{mgu})^*(\mathcal{C})$.

(103) Let $\mathcal{L}$ be a first-order language with equality. Define a formal system $\mathcal{FRESPAR} - \mathcal{IRER}^{\mathcal{L}}_{mgu}$ and prove that for a set of $\mathcal{L}$-clauses $\mathcal{C}$, we have:

$$(\text{fRespar-ir-er}^{mgu})^*(\mathcal{C}) = \text{Thm}(\text{fRespar-ir-er}^{mgu}_{\mathcal{C}}).$$

(104) Let $\mathcal{L}$ be a first-order language with equality and let $\mathcal{C}$ be a set of $\mathcal{L}$-clauses. Prove that if $D \in (\text{fRespar-ir-er}^{mgu})^*(\mathcal{C})$ and $\mathcal{A} \models \mathcal{C}$ for some $\mathcal{L}$-structure $\mathcal{A}$, then $\mathcal{A} \models D$.

Furthermore, prove that if $\Box \in (\text{fRespar-ir-er}^{mgu})^*(\mathcal{C})$, then $\mathcal{C}$ has no model.

(105) Let $\mathcal{L}$ be a first-order language with equality and let $\mathcal{C}$ be a set of $\mathcal{L}$-clauses. Prove that if $\mathcal{C}$ has no model, then

$$\Box \in (\text{fRespar-ir-er}^{mgu})^*(\mathcal{C}).$$

Hint. Examine the proofs of Theorems 5.10.64 and 5.10.65 and verify that the uses of the clause $\{(x = x)\}$ can be replaced by applications of the full, most general irreflexivity removal and equality reversal.

(106) Let $\mathcal{L}$ be a first-order language and C_0, C_0', C_1, C_1' be ground $\mathcal{L}$-clauses with $C_0 \subseteq C_0'$ and $C_1 \subseteq C_1'$. Prove that:

(a) if R is a resolvent of C_0 and C_1, then there is a resolvent R' of C_0' and C_1' such that $R \subseteq R'$ and $R' - R \subseteq (C_0' - C_0) \cup (C_1' - C_1)$;

(b) if $\mathcal{L}$ is a language with equality and P is a paramodulant of C_0 and C_1, then there is a paramodulant P' of C_0' and C_1' such that $P \subseteq P'$ and $P' - P \subseteq (C_0' - C_0) \cup (C_1' - C_1)$.

(107) Let $\mathcal{L}$ be a first-order language with equality and $\mathcal{C}$ be a set of ground $\mathcal{L}$-clauses. We say that $\mathcal{C}$ is *RP-refutation consistent* if $\Box \notin \text{Respar}^*_{\mathcal{L}}(\mathcal{C})$.
Note that since $\text{Respar}^*_{\mathcal{L}}(\mathcal{C}) = (\text{fRespar}^{mgu})^*(\mathcal{C})$, when $\mathcal{C}$ is a set of ground clauses, RP-refutation consistency is language independent.
Prove that:

(a) RP-refutation consistency is a property of finite character of the subsets of the set of ground $\mathcal{L}$-clauses $\mathsf{GCLAUSES}_{\mathcal{L}}$;

(b) if $\mathcal{C}$ is a maximal RP-refutation consistent subet of $\mathsf{GCLAUSES}_{\mathcal{L}}$ and $C \in \text{Respar}^*_{\mathcal{L}}(\mathcal{C})$, then $C \in \mathcal{C}$ (i.e., $\text{Respar}^*_{\mathcal{L}}(\mathcal{C}) = \mathcal{C}$).

(108) Let $\mathcal{L}$ be a first-order language with equality and let $\mathcal{C}$ be a maximal RP-refutation consistent subset of $\mathsf{GCLAUSES}_{\mathcal{L}}$. Prove that for every $C \in \mathcal{C}$, there is a literal $\ell \in \mathcal{C}$ such that $\{\ell\} \in \mathcal{C}$.
Solution: Suppose for a contradiction that there is a clause $C \in \mathcal{C}$ such that for no $\ell' \in C$ do we have $\{\ell'\} \in \mathcal{C}$. Let C be such a clause of minimal size. Then, no proper subset of C

can belong to $\mathcal{C}$. Since $\mathcal{C}$ is RP-refutation consistent, $C \neq \Box$, so there is a literal $\ell \in C$ and by the previous assumption, $\{\ell\} \notin \mathcal{C}$. Thus, $\mathcal{C} \subset \mathcal{C} \cup \{\ell\}$ and by the maximality of $\mathcal{C}$, $\mathcal{C} \cup \{\ell\}$ is not RP-refution consistent, so $\Box \in \mathrm{Respar}^*_{\mathcal{L}}(\mathcal{C} \cup \{\ell\})$. Thus, there is a sequence $(C_0, \ldots, C_{n-1})$ which is an $\mathcal{L}$-resolution paramodulation proof over $\mathcal{C} \cup \{\ell\}$ of $\Box$. We define recursively a resolution paramodulation proof $(C'_0, \ldots, C'_{n-1})$ over $\mathcal{C}$ with these two properties:

- $C_i \subseteq C'_i$,
- $C'_i - C_i \subseteq C - \{\ell\}$

for $0 \leq i \leq n-1$.
Suppose that $0 \leq k \leq n-1$ and $C_{k'}$ is defined for $0 \leq k' \leq k$ with the desired properties specified above. Define C'_k as follows:

- If $C_k \in \mathcal{C}$, define $C'_k = C_k$. In this case, the two properties clearly hold.
- If $C_k = \{\ell\}$, define $C'_k = C$. Again, the two properties hold.
- If neither of the previous two cases holds, there are $i, j < k$ such that C_k is either a resolvent or a paramodulant of C_i and C_j. By Exercise 106, there is a resolvent or a paramodulant C'_k of C'_i and C'_j with $C_k \subseteq C'_k$ and $C'_k - C_k \subseteq (C'_i - C_i) \cup (C'_j - C_j) \subseteq C - \{\ell\}$.

Since $C'_{n-1} \in \mathrm{Respar}^*_{\mathcal{L}}(\mathcal{C})$, by Exercise 107, $C'_{n-1} \in \mathcal{C}$. Since $C_{n-1} = \Box$, $C'_{n-1} \subseteq C - \{\ell\}$, C'_{n-1} is a proper subset of C, which is a contradiction. Thus, C does not exist.

(109) Let $\mathcal{L}$ be a first-order language with equality that contains at least one constant symbol and let $\mathcal{C}$ be a maximal RP-refutation consistent subset of $\mathsf{GCLAUSES}_{\mathcal{L}}$ that contains $\{\{(t = t)\} \mid t \in \mathrm{GTERM}_{\mathcal{L}}\}$. Prove that $\mathcal{C}$ has an $\mathcal{L}$-E-model.
Solution: Define an $\mathcal{LH}$-interpretation I by

$$I(\alpha) = \begin{cases} \mathbf{T} & \text{if } \{\alpha\} \in \mathcal{C}, \\ \mathbf{F} & \text{if } \{\alpha\} \notin \mathcal{C}, \end{cases}$$

for $\alpha \in \mathrm{GAFORM}_{\mathcal{L}}$. If $C \in \mathcal{C}$, then by Supplement 108, there is a literal ℓ such that $\{\ell\} \in \mathcal{C}$. If ℓ is positive, then $I(\ell) = \mathbf{T}$, so

$I \models C$. If ℓ is negative, then since $\mathcal{C}$ is RP-refutation consistent $\{\bar{\ell}\} \notin \mathcal{C}$, so $I(\bar{\ell}) = \mathbf{F}$, hence $I(\ell) = \mathbf{T}$ and $I \models C$. Thus, $I \models \mathcal{C}$. E1 holds because $\{(t = t)\} \in \mathcal{C}$ for all $t \in \mathrm{GTERM}_{\mathcal{L}}$. To see that E3 holds, let $\alpha \in \mathrm{GAFORM}_{\mathcal{L}}$, $(t, i) \in \mathtt{OCC}_t(\alpha)$, and $I(t = u) = \mathbf{T}$, that is, $(t = u) \in \mathcal{C}$. If $I(\alpha) = \mathbf{T}$, then $\{\alpha\} \in \mathcal{C}$ and paramodulating $(t = u)$ into α, we obtain $\{\alpha[(t, i) \to u]\} \in \mathrm{Respar}_{\mathcal{L}}(\mathcal{C}) = \mathcal{C}$, so $I(\alpha[(t, i) \to u]) = \mathbf{T} = I(\alpha)$. Similarly, if $I(\alpha[(t, i) \to u]) = \mathbf{T}$, that is $\{\alpha[(t, i) \to u]\} \in \mathcal{C}$, we paramodulate $(t = u)$ into $\{\alpha[(t, i) \to u]\}$ in reverse, replacing u with t, and obtain $\{\alpha\} \in \mathrm{Respar}_{\mathcal{L}}(\mathcal{C}) = \mathcal{C}$, so $I(\alpha) = \mathbf{T} = I(\alpha[(t, i) \to u])$. Thus, by Theorem 5.10.30, I is an $\mathcal{L}$-E-interpretation and hence an $\mathcal{L}$-E-model of $\mathcal{C}$.

(110) Let $\mathcal{L}$ be a first-order language with equality that contains at least one constant symbol and let $\mathcal{C} \subseteq \mathsf{GCLAUSES}_{\mathcal{L}}$ be such that $\mathcal{C}$ has no $\mathcal{L}$-E-model and $\{\{(t = t)\} \mid t \in \mathrm{GTERM}_{\mathcal{L}}\} \subseteq \mathcal{C}$. Prove that $\square \in \mathrm{Respar}^*_{\mathcal{L}}(\mathcal{C})$.
Solution: To prove the contrapositive, suppose that $\square \notin \mathrm{Respar}^*_{\mathcal{L}}(\mathcal{C})$, that is, $\mathcal{C}$ is RP-refutation consistent. By Theorem 1.3.3, there is a maximal RP-refutation consistent subset $\mathcal{C}'$ of $\mathsf{GCLAUSES}_{\mathcal{L}}$ with $\mathcal{C} \subseteq \mathcal{C}'$ and thus $\{\{(t = t)\} \mid t \in \mathrm{GTERM}_{\mathcal{L}}\} \subseteq \mathcal{C}'$. By Supplement 109, $\mathcal{C}'$ has an $\mathcal{L}$-E-model, so $\mathcal{C}$ has an $\mathcal{L}$-E-model.

Let $\mathcal{L}$ be a first-order language with equality. Define

$$\begin{aligned}\mathsf{FEQ}_{\mathcal{L}} = &\{\{(a = a)\} \mid a \text{ is a constant symbol of } \mathcal{L}\}\\ &\cup\{\{(f(x_0, \ldots, x_{n-1}) = f(x_0, \ldots, x_{n-1}))\}\\ &\mid f \text{ is a function symbol of } \mathcal{L} \text{ with positive arity } n\}.\end{aligned}$$

(111) Let $\mathcal{L}$ be a first-order language with equality. Prove that $\{\{(t = t)\} \mid t \in \mathrm{GTERM}_{\mathcal{L}}\} \subseteq (\mathrm{fRespar}^{mgu})^*(\mathsf{FEQ}_{\mathcal{L}})$.
Solution: The argument is by induction on t. If t is a constant symbol, then $\{(t = t)\} \in \mathsf{FEQ}_{\mathcal{L}} \subseteq (\mathrm{fRespar}^{mgu})^*(\mathsf{FEQ}_{\mathcal{L}})$. Suppose that $t = f(t_0, \ldots, t_{n-1})$ and by the inductive hypothesis, $\{(t_i = t_i)\} \in (\mathrm{fRespar}^{mgu})^*(\mathsf{FEQ}_{\mathcal{L}})$ for $0 \leq i \leq n - 1$. We have $\{(f(x_0, \ldots, x_{n-1}) = f(x_0, \ldots, x_{n-1}))\} \in \mathsf{FEQ}_{\mathcal{L}}$ and $\{(t_0 = t_0)\} \in (\mathrm{fRespar}^{mgu})^*(\mathsf{FEQ}_{\mathcal{L}})$. Mapping x_0 to t_0 is an mgu of the set $\{x_0, t\}$. Paramodulating $(t_0 = t_0)$ into $(f(x_0, \ldots x_{n-1}) = f(x_0, \ldots, x_{n-1}))$,

we obtain $\{(f(t_0, x_1, \ldots, x_{n-1}) = f(t_0, x_1, \ldots, x_{n-1}))\}$ as a full, most general paramodulant of the two clauses. We now paramodulate $(t_1 = t_1)$ into $\{(f(t_0, x_1, \ldots, x_{n-1}) = f(t_0, x_1, \ldots, x_{n-1}))\}$ and continue until we obtain $\{(t = t)\} \in (\text{fRespar}^{mgu})^*(\mathsf{FEQ}_{\mathcal{L}})$.

(112) Let $\mathcal{L}$ be a first-order language with equality and let $\mathcal{C}$ be a set of $\mathcal{L}$-clauses. Prove that $\text{GINST}_{\mathcal{L}}(\mathcal{C}) \subseteq (\text{fRespar}^{mgu})^*(\mathcal{C} \cup \mathsf{FEQ}_{\mathcal{L}})$.

Solution: Let $C \in \mathcal{C}$ and let C' be a ground instance of C, say $C' = \mathsf{s}^{x_0 \cdots x_{n-1}}_{t_0 \cdots t_{n-1}}(C)$, where each x_i occurs in some literal in C. By Supplement 111, we have $\{(t_i = t_i)\} \in (\text{fRespar}^{mgu})^*(\mathcal{C} \cup \mathsf{FEQ}_{\mathcal{L}})$ for $0 \leq i \leq n-1$. If $n = 0$, then $C' = C \in (\text{fRespar}^{mgu})^*(\mathcal{C} \cup \mathsf{FEQ}_{\mathcal{L}})$. If $n > 0$, then we paramodulate $(t_0 = t_0)$ into some occurrence of x_0 in a literal of C, obtaining $\mathsf{s}^{x_0}_{t_0}(C)$ as a full, mgu paramodulant of C and $\{(t_0 = t_0)\}$. Continuing in the same manner, we obtain $C' \in (\text{fRespar}^{mgu})^*(\mathcal{C} \cup \mathsf{FEQ}_{\mathcal{L}})$.

(113) Let $\mathcal{L}$ be a first-order language with equality that contains at least one constant symbol and let $\mathcal{C}$ be a set of $\mathcal{L}$-clauses that has no model. Then, $\Box \in (\text{fRespar}^{mgu})^*(\mathcal{C} \cup \mathsf{FEQ}_{\mathcal{L}})$.

Solution: Let $\mathcal{D} = \text{GINST}_{\mathcal{L}}(\mathcal{C}) \cup \{\{(t = t)\} \mid t \in \text{GTERM}_{\mathcal{L}}\}$. By the previous two supplements, we have $\mathcal{D} \subseteq (\text{fRespar}^{mgu})^*(\mathcal{C} \cup \mathsf{FEQ}_{\mathcal{L}})$ and hence

$$\text{Respar}^*_{\mathcal{L}}(\mathcal{D}) = (\text{fRespar}^{mgu})^*(\mathcal{D}) \subseteq (\text{fRespar}^{mgu})^*(\mathcal{C} \cup \mathsf{FEQ}_{\mathcal{L}}). \tag{5.13}$$

By Theorem 5.10.35, $\text{GINST}_{\mathcal{L}}(\mathcal{C})$ has no $\mathcal{L}$-E-model, so $\mathcal{D}$ has no $\mathcal{L}$-E-model. By Supplement 110, $\Box \in \text{Respar}^*_{\mathcal{L}}(\mathcal{D})$. This fact combined with (5.13) leads to the desired conclusion.

5.12 Bibliographical Comments

As mentioned in Chapter 3, the initial reference for tableaux is E. W. Beth [4] continued by the work of Raymond Smullyan [31].

For sequents see Gentzen in [16] and for natural deduction, the initial reference is [17] where the cut rule and cut elimination were also introduced.

First-order resolution was introduced by Robinson in [27]. Hyper-resolution was also introduced by Robinson in [26]. Paramodulation appears first in [2] where they proved the completeness result of Supplement 113. The alternative proof presented in this supplement is due to [28]. The other completeness result of Theorem 5.10.66 was first proven by Brand as outlined in [6]. Our proof of this stronger completeness result follows Peterson's approach in [24].

The results of Supplement 28 were obtained in [5].

This page intentionally left blank

Chapter 6

Program Verification

6.1 Introduction

In this chapter we present an introduction to the area of program verification, an area of intersection between Computer Science and Logic that originates in the works of Floyd [15][1] and Hoare [20].[2]

Computer Science and Logic have a common tradition of rigorous and separate presentation of syntax and semantics of the realms they investigate. This chapter presents a class of programs and studies its properties in this spirit.

[1]Robert W. Floyd (1936–2001) was born on June 8, 1936 in New York. Floyd obtained two bachelor's degrees from the University of Chicago, in 1953 and 1958, and taught at Carnegie-Mellon and Stanford Universities. Floyd pioneered systematic methods of program verification and had a strong influence on Hoare's work. Floyd won the 1978 ACM Turing Award, the highest honor in Computer Science "for having a clear influence on methodologies for the creation of efficient and reliable software, and for helping to found the following important subfields of computer science: the theory of parsing, the semantics of programming languages, automatic program verification, automatic program synthesis, and analysis of algorithms". He died on September 25, 2001.

[2]Sir Charles Anthony Richard Hoare was born on January 11, 1934. C.A.R. Hoare is Professor Emeritus at Oxford University, the winner of the 1980 Turing Award of the Association for Computing Machinery, and is currently a Principal Researcher with the Microsoft Research Center in the United Kingdom. His main contributions are in the area of structured programming, formal methods for program verification and communicating sequential processes.

We define inductively a class of programs known as $\mathcal{L}$-programs and then we proceed to redefine the same class of programs using a definition that satisfies the unique readability condition. This later definition is essential because in the subsequent text we use numerous inductive definitions of functions on programs.

Next, in Section 6.3, we introduce the semantics of $\mathcal{L}$-programs by using the notion of $\mathcal{A}$-state. The semantics of a $\mathcal{L}$-program $\mathfrak{p}$ is defined with the help of two functions $\mathcal{S}_{\mathcal{A}}^{pr}(\mathfrak{p})$ and $\mathcal{S}_{\mathcal{A}}^{at\text{-}pr}(\mathfrak{p})$ that are partial transformations on the set of states $\mathrm{STATES}_{\mathcal{A}}$. In the same section we prove two technically important results: the state agreement theorem and the time agreement theorem for programs.

The use of $\mathcal{L}$-programs to compute functions over $\mathcal{L}$-structures is discussed in Section 6.4. This section has mostly a technical character and is a prelude for the rest of the chapter.

Section 6.5 introduces partial and total correctness Hoare triples, which are composite objects formed by two special formulas called assertions and an $\mathcal{L}$-program. The intuitive notions of partial and total correctness in an $\mathcal{L}$-structure are captured by the satisfaction of a triple by a pair $(\mathcal{A}, \sigma)$, where $\mathcal{A}$ is an $\mathcal{L}$-structure and σ is an $\mathcal{A}$-state. We argue that total correctness is not a first-order property, that is, total correctness depends not only on the theory of a structure, but on the structure itself. In contrast, partial correctness is shown to be a first-order property. This is proven using the notion of weakest liberal precondition which was introduced by Dijkstra[3] [12].

Section 6.6 is dedicated to the study of the sets of partial and total correctness triples that are valid in an $\mathcal{L}$-structure $\mathcal{A}$ and we show that the set of all partial correctness triples that are valid in arithmetic is not semidecidable and, thus, it is undecidable. In this

[3]Edsger Wybe Dijkstra (1930–2002) was born on May 11, 1930 in Rotterdam, Netherlands. Dijkstra studied theoretical physics at the University of Leiden and obtained his Ph.D. in Mathematics from the University of Amsterdam in 1959. He worked as a programmer at the Mathematisch Centrum in Amsterdam between 1952 and 1962, was a professor of Mathematics at the Eindhoven University of Technology and taught at the University of Texas at Austin between 1984 and 1999. Dijkstra was the 1972 recipient of the ACM Turing Award. He highlighted the deep links between mathematical logic and program development, and made fundamental contributions in algorithms and operating systems. He was the author of the first ALGOL compiler. Dijkstra died on August 6, 2002 in Neuen, the Netherlands.

section we introduce the notion of expressive structure and we prove that for these structures the decidability of their theories implies the decidability of their partial correctness theories. We show the expressiveness of finite structures and, also, the expressiveness of arithmetic. In the last part of this section we prove that the Hoare partial correctness of Pressburger Arithmetic is not semidecidable.

The last section of the chapter presents a formal system for Hoare triples. After arguing that the introduction of a complete formal system that would prove Hoare triples is not possible we introduce a formal system that is sound and, in a limited sense, complete for expressive structures. Special notations for proofs in the formal system referred to as annotated programs are also introduced and we demonstrate the role of loop invariants in constructing annotated programs.

6.2 The $\texttt{WHILE}_{\mathcal{L}}$ Programming Language — Syntax

Let $\mathcal{L}$ be a first-order language. The syntactic components of the $\texttt{WHILE}_{\mathcal{L}}$ programming language that we are about to introduce are at the core of many programming languages. We assume that VAR is partitioned into two infinite sets PVAR and SVAR, referred to as the set of *program variables* and the set of *specification variables*, respectively. Further, we assume that we have a fixed bijection between PVAR and SVAR; if $y \in \text{PVAR}$, we denote its image under this bijection as $y_{\star}$.

We use several *program symbols* that we denote (using multi-letter words for readability) as

$$\textbf{while} \quad \textbf{do} \quad \textbf{endwhile} \; \textbf{if} \quad \textbf{then} \quad \textbf{else} \quad \textbf{endif} \;\; ; \;\; \leftarrow$$

The set of *basic symbols* consists of the program variables, program symbols and punctuation signs.

Definition 6.2.1. Let $\mathcal{L}$ be a first-order language. The programming language $\texttt{WHILE}_{\mathcal{L}}$ consists of the set of sequences of basic symbols and symbols of $\mathcal{L}$ defined recursively as follows:

(1) A sequence of the form $v \leftarrow t$, where v is a program variable and t is an $(\mathcal{L}, \text{PVAR})$-term belongs to $\texttt{WHILE}_{\mathcal{L}}$ and is called an *assignment statement of* $\mathcal{L}$.

(2) If β is a quantifier-free $(\mathcal{L}, \mathrm{PVAR})$-formula and $\mathfrak{p} \in \mathtt{WHILE}_{\mathcal{L}}$, then

$$\textbf{while } \beta \textbf{ do } \mathfrak{p} \textbf{ endwhile}$$

is in $\mathtt{WHILE}_{\mathcal{L}}$.

(3) If β is a quantifier-free $(\mathcal{L}, \mathrm{PVAR})$-formula, and $\mathfrak{p}, \mathfrak{p}' \in \mathtt{WHILE}_{\mathcal{L}}$, then

$$\textbf{if } \beta \textbf{ then } \mathfrak{p} \textbf{ else } \mathfrak{p}' \textbf{ endif}$$

belongs to $\mathtt{WHILE}_{\mathcal{L}}$.

(4) If $\mathfrak{p}, \mathfrak{p}' \in \mathtt{WHILE}_{\mathcal{L}}$, then $\mathfrak{p}; \mathfrak{p}'$ belongs to $\mathtt{WHILE}_{\mathcal{L}}$.

We refer to members of $\mathtt{WHILE}_{\mathcal{L}}$ as *$\mathcal{L}$-programs*.

If $\mathfrak{p}_0$ is the program **while** β **do** $\mathfrak{p}$ **endwhile**, then we refer to $\mathfrak{p}$ as the *body* of $\mathfrak{p}_0$.

If a string of symbols is an $\mathcal{L}$-program for some first-order language $\mathcal{L}$, then we say that string is a *program*. We denote the set of all programs as `WHILE`. ▯

Theorem 6.2.2. `WHILE` *equals the set W given by the following inductive definition*:

(1) *A sequence of the form $v \leftarrow t$, where v is a program variable and t is a term of first-order logic with $\mathtt{V}(t) \subseteq PVAR$ belongs to W.*

(2) *If β is a quantifier-free formula of first-order logic whose variables are included in $PVAR$ and $\mathfrak{p} \in W$, then*

$$\textbf{\textit{while}}\ \beta\ \textbf{\textit{do}}\ \mathfrak{p}\ \textbf{\textit{endwhile}}$$

is in W.

(3) *If β is a quantifier-free formula whose variables belong to $PVAR$, and $\mathfrak{p}, \mathfrak{p}' \in W$, then*

$$\textbf{\textit{if}}\ \beta\ \textbf{\textit{then}}\ \mathfrak{p}\ \textbf{\textit{else}}\ \mathfrak{p}'\ \textbf{\textit{endif}}$$

belongs to W.

(4) *If $\mathfrak{p}, \mathfrak{p}' \in W$, then $\mathfrak{p}; \mathfrak{p}'$ belongs to W.*

Proof. The proof is similar to previous results concerning terms and formulas. (See Theorem 4.3.6 and Theorem 4.3.15.) □

Note that Definition 6.2.1 is not uniquely readable. For example, if $\mathfrak{p}, \mathfrak{p}', \mathfrak{p}''$ are $\mathcal{L}$-programs, then we can build the program $\mathfrak{p};\mathfrak{p}';\mathfrak{p}''$ either by concatenating program $\mathfrak{p};\mathfrak{p}'$ with $\mathfrak{p}''$ or by concatenating the program $\mathfrak{p}$ with the program $\mathfrak{p}';\mathfrak{p}''$. This lack of unique readability can create difficulties for functions defined inductively on $\mathtt{WHILE}_{\mathcal{L}}$.

An equivalent definition that has the unique readability property involves the simultaneous introduction of the set of atomic programs and the set of programs. This definition is given next:

Definition 6.2.3. Let $\mathcal{L}$ be a first-order language. We define inductively the sets $\mathtt{WHILE}^{at}_{\mathcal{L}}$ and $\mathtt{WHILE}'_{\mathcal{L}}$ of sequences of basic symbols and symbols of $\mathcal{L}$ as follows:

(1) Any assignment statement of the form $v \leftarrow t$, where v is a program variable and t is an $(\mathcal{L}, \mathrm{PVAR})$-term belongs to $\mathtt{WHILE}^{at}_{\mathcal{L}}$.
(2) If β is a quantifier-free $(\mathcal{L}, \mathrm{PVAR})$-formula and $\mathfrak{p}$ is in $\mathtt{WHILE}'_{\mathcal{L}}$, then

$$\textbf{while } \beta \textbf{ do } \mathfrak{p} \textbf{ endwhile}$$

belongs to $\mathtt{WHILE}^{at}_{\mathcal{L}}$.
(3) If β is a quantifier-free $(\mathcal{L}, \mathrm{PVAR})$-formula, and $\mathfrak{p}, \mathfrak{p}' \in \mathtt{WHILE}'_{\mathcal{L}}$, then

$$\textbf{if } \beta \textbf{ then } \mathfrak{p} \textbf{ else } \mathfrak{p}' \textbf{ endif}$$

belongs to $\mathtt{WHILE}^{at}_{\mathcal{L}}$.
(4) Every sequence in $\mathtt{WHILE}^{at}_{\mathcal{L}}$ belongs to $\mathtt{WHILE}'_{\mathcal{L}}$.
(5) If $\mathfrak{p} \in \mathtt{WHILE}^{at}_{\mathcal{L}}$ and $\mathfrak{p}' \in \mathtt{WHILE}'_{\mathcal{L}}$, then $\mathfrak{p};\mathfrak{p}'$ belongs to $\mathtt{WHILE}'_{\mathcal{L}}$.

A member of $\mathtt{WHILE}^{at}_{\mathcal{L}}$ is called an *atomic program of* $\mathcal{L}$. or *atomic* $\mathcal{L}$*-program* $\square$

To prove the equivalence of Definitions 6.2.1 and 6.2.3 we prove the following technical statement:

Lemma 6.2.4. *Let* $\mathcal{L}$ *be a first-order language. If* $\mathfrak{p}, \mathfrak{p}' \in \mathtt{WHILE}'_{\mathcal{L}}$, *then* $\mathfrak{p};\mathfrak{p}' \in \mathtt{WHILE}'_{\mathcal{L}}$.

Proof. The argument is by induction on $\mathfrak{p}$. For the basis step suppose that $\mathfrak{p}$ is an atomic program of $\mathcal{L}$. Then, $\mathfrak{p};\mathfrak{p}'$ belongs to $\mathtt{WHILE}'_{\mathcal{L}}$ by Rule 5 of Definition 6.2.3.

For the inductive step let $\mathfrak{p}$ be a sequence in $\mathtt{WHILE}'_{\mathcal{L}}$ that has the form $\mathfrak{p} = \mathfrak{p}_0;\mathfrak{p}_1$, where $\mathfrak{p}_0 \in \mathtt{WHILE}^{at}_{\mathcal{L}}$ and $\mathfrak{p}_1$ is a sequence in $\mathtt{WHILE}'_{\mathcal{L}}$ such that the above statement holds. We have $\mathfrak{p};\mathfrak{p}' = \mathfrak{p}_0;\mathfrak{p}_1;\mathfrak{p}'$. By the inductive hypothesis, $\mathfrak{p}_1;\mathfrak{p}' \in \mathtt{WHILE}'_{\mathcal{L}}$, so $\mathfrak{p}_0;\mathfrak{p}_1;\mathfrak{p}'$ belongs to $\mathtt{WHILE}'_{\mathcal{L}}$ by Rule 5 of Definition 6.2.3. □

Theorem 6.2.5. *We have* $\mathtt{WHILE}_{\mathcal{L}} = \mathtt{WHILE}'_{\mathcal{L}}$ *for every first-order language* $\mathcal{L}$.

Proof. It is easy to prove by simultaneous induction on the definitions of the sets $\mathtt{WHILE}^{at}_{\mathcal{L}}$ and $\mathtt{WHILE}'_{\mathcal{L}}$ that they are both included in $\mathtt{WHILE}_{\mathcal{L}}$. Thus, we need to prove only the reverse inclusion, $\mathtt{WHILE}_{\mathcal{L}} \subseteq \mathtt{WHILE}'_{\mathcal{L}}$.

Let $\mathfrak{p}$ be a sequence in $\mathtt{WHILE}_{\mathcal{L}}$. The argument is by induction on $\mathfrak{p}$.

If $\mathfrak{p}$ is an assignment statement of $\mathcal{L}$, then $\mathfrak{p}$ is an atomic program of $\mathcal{L}$ by Rule 1 of Definition 6.2.3 and therefore it belongs to $\mathtt{WHILE}'_{\mathcal{L}}$ by Rule 4 of the same definition.

If $\mathfrak{p}$ was put in $\mathtt{WHILE}_{\mathcal{L}}$ by Rule 2 of Definition 6.2.1, then

$$\mathfrak{p} = \textbf{ while } \beta \textbf{ do } \hat{\mathfrak{p}} \textbf{ endwhile},$$

where $\hat{\mathfrak{p}} \in \mathtt{WHILE}'_{\mathcal{L}}$ by the inductive hypothesis. Then, $\mathfrak{p} \in \mathtt{WHILE}^{at}_{\mathcal{L}}$ by Rule 2 of Definition 6.2.3, so it belongs to $\mathtt{WHILE}'_{\mathcal{L}}$ by Rule 4 of the same definition.

The case when $\mathfrak{p}$ is put in $\mathtt{WHILE}_{\mathcal{L}}$ by Rule 3 of Definition 6.2.1 is similar and is left to the reader.

If $\mathfrak{p}$ belongs to $\mathtt{WHILE}_{\mathcal{L}}$ by Rule 4 of Definition 6.2.1, then $\mathfrak{p} = \mathfrak{p}';\mathfrak{p}''$, where $\mathfrak{p}', \mathfrak{p}'' \in \mathtt{WHILE}'_{\mathcal{L}}$ by the inductive hypothesis. By Lemma 6.2.4 we have $\mathfrak{p} \in \mathtt{WHILE}'_{\mathcal{L}}$. □

To prepare for the proof of the unique readability of Definition 6.2.3, we need the following definition.

Definition 6.2.6. Let $\mathcal{L}$ be a first-order language and let $\mathfrak{p} \in \mathtt{WHILE}_{\mathcal{L}}$. If (s, i) is an occurrence of a symbol s in $\mathfrak{p}$, then the *level* of (s, i) in $\mathfrak{p}$ is

$$\mathrm{lev}_{\mathfrak{p}}(s, i) = |q|_{\textbf{while}} - |q|_{\textbf{endwhile}} + |q|_{\textbf{if}} - |q|_{\textbf{endif}},$$

where $q = \mathrm{pref}(\mathfrak{p}, i)$. ▯

Lemma 6.2.7. *Let $\mathcal{L}$ be a first-order language. For every program $\mathfrak{p} \in \texttt{WHILE}_{\mathcal{L}}$, we have $|\mathfrak{p}|_{\textbf{while}} = |\mathfrak{p}|_{\textbf{endwhile}}$ and $|\mathfrak{p}|_{\textbf{if}} = |\mathfrak{p}|_{\textbf{then}} = |\mathfrak{p}|_{\textbf{else}} = |\mathfrak{p}|_{\textbf{endif}}$.*

Proof. The argument is by induction based on Definition 6.2.1 and is left to the reader. □

Lemma 6.2.8. *Let $\mathcal{L}$ be a first-order language and $\mathfrak{p}$ be a $\mathcal{L}$-program. For any occurrence of a semicolon in $\mathfrak{p}$, the level of the occurrence is nonnegative. Further, if $\mathfrak{p}$ is an atomic program, the level of any occurrence of a semicolon in $\mathfrak{p}$ is positive.*

Proof. We prove simultaneously both statements of the lemma by induction based on Definition 6.2.3. The basis step is vacuous because there are no semicolons in an assignment statement.

Now suppose that $\mathfrak{p}_0$ is **while** β **do** $\mathfrak{p}$ **endwhile**, where the level of every occurrence of a semicolon in $\mathfrak{p}$ is nonnegative. Then, clearly the level of the corresponding occurrence in $\mathfrak{p}_0$ is larger by one than the level in $\mathfrak{p}$, so is positive. Since the only semicolon occurrences in $\mathfrak{p}_0$ are the ones in $\mathfrak{p}$, we obtain the desired statement for the atomic program $\mathfrak{p}_0$.

Let now $\mathfrak{p}_0$ be **if** β **then** $\mathfrak{p}$ **else** $\mathfrak{p}'$ **endif**, where every occurrence of a semicolon in $\mathfrak{p}$ or $\mathfrak{p}'$ has a nonnegative level. There are two types of occurrences of semicolons in $\mathfrak{p}_0$: those corresponding to occurrences in $\mathfrak{p}$ and those corresponding to occurrences in $\mathfrak{p}'$. In the first case, the level in $\mathfrak{p}_0$ is larger by one than the level in $\mathfrak{p}$ and so it is positive. In the second case, the same property holds because of Lemma 6.2.7 applied to $\mathfrak{p}$.

For programs that enter $\texttt{WHILE}_{\mathcal{L}}$ by Rule 4 of Definition 6.2.3, the positivity of the levels of occurrences of semicolons in atomic programs implies the nonnegativity of the levels of occurrences of semicolons in the same program considered as a member of $\texttt{WHILE}_{\mathcal{L}}$.

Finally, let $\mathfrak{p}_0 = \mathfrak{p}; \mathfrak{p}'$, where $\mathfrak{p}$ is an atomic program such that every occurrence of a semicolon has positive level and $\mathfrak{p}'$ is a program such that every occurrence of a semicolon has nonnegative level. The occurrences of semicolons in $\mathfrak{p}_0$ which are the same as occurrences of semicolons in $\mathfrak{p}$ have the same level in both programs and therefore have positive levels. The occurrence $(;, |\mathfrak{p}|)$ has level 0 by Lemma 6.2.7. The occurrences of semicolons in $\mathfrak{p}_0$ that correspond to occurrences in $\mathfrak{p}'$ have the same level in both programs because of

Lemma 6.2.7, which allows us to conclude that any occurrence of a semicolon in $\mathfrak{p}_0$ has nonnegative level. □

Lemma 6.2.9. *Let $\mathcal{L}$ be a first-order language and let $\mathfrak{p} \in \texttt{WHILE}_{\mathcal{L}}$. The level of any occurrence of* **else** *in $\mathfrak{p}$ is positive.*

Proof. The argument is by induction on Definition 6.2.1 and it is left to the reader. □

Theorem 6.2.10. *Definition* 6.2.3 *meets the unique readability condition.*

Proof. We need to prove:

- Every atomic program is put in the set $\texttt{WHILE}^{at}_{\mathcal{L}}$ by only one of Rules 1-3 of the definition.
- If an atomic program enters $\texttt{WHILE}^{at}_{\mathcal{L}}$ by either Rule 2 or 3, then $\mathfrak{p}$, or $\mathfrak{p}$ and $\mathfrak{p}'$, respectively, are uniquely determined.
- Every program is put in the set $\texttt{WHILE}_{\mathcal{L}}$ by only one of Rules 4 and 5 of the definition.
- If a program enters $\texttt{WHILE}_{\mathcal{L}}$ by Rule 5, then the atomic program $\mathfrak{p}$ and the program $\mathfrak{p}'$ are uniquely determined.

Note that in principle a uniqueness condition has to be proved for Rule 4. However, this condition is trivially satisfied.

If $\mathfrak{p}_0$ is an atomic program, its first symbol (a variable, **while**, or **if**) determines the rule that was applied.

If $\mathfrak{p}_0 =$ **while** β **do** $\mathfrak{p}$ **endwhile**, then $\mathfrak{p}$ is the subsequence of $\mathfrak{p}_0$ located between the first **do** and the last **endwhile**. (In addition, note that β is also uniquely determined as the subsequence between the first **while** and the first **do**.)

If $\mathfrak{p}_0 =$ **if** β **then** $\mathfrak{p}$ **else** $\mathfrak{p}'$ **endif**, then $\mathfrak{p}_0$ contains exactly one occurrence of **else** at level 1. Indeed, note that any occurrence of **else** within $\mathfrak{p}$ or $\mathfrak{p}'$ is at least at level 2 due to Lemma 6.2.9 and to the presence of the initial **if** symbol. Therefore, $\mathfrak{p}$ is the subsequence of $\mathfrak{p}_0$ located between the first **then** and the level one occurrence of **else** and $\mathfrak{p}'$ is the subsequence of $\mathfrak{p}_0$ located between the level one occurrence of **else** and the last occurrence of **endif**.

Let $\mathfrak{p}_1$ be a program. If $\mathfrak{p}_1$ was put in $\texttt{WHILE}_{\mathcal{L}}$ by Rule 4, then by Lemma 6.2.8, $\mathfrak{p}_1$ contains no occurrence of a semicolon at level 0. Otherwise, that is if $\mathfrak{p}_1$ was put in by Rule 5, then it contains an occurrence of a semicolon at level 0 by Lemma 6.2.7. Thus, $\mathfrak{p}_1$ is put in $\mathtt{WHILE}_{\mathcal{L}}$ by only one rule.

If $\mathfrak{p}_1 = \mathfrak{p};\mathfrak{p}'$, where $\mathfrak{p} \in \mathtt{WHILE}^{at}_{\mathcal{L}}$ and $\mathfrak{p}' \in \mathtt{WHILE}_{\mathcal{L}}$, then $\mathfrak{p}$ is the prefix of $\mathfrak{p}_1$ which ends just before the first level zero occurrence of a semicolon, while $\mathfrak{p}'$ is the suffix of $\mathfrak{p}_1$ that follows this occurrence. □

For a program $\mathfrak{p}$ of $\mathcal{L}$, we denote by $\mathrm{PVAR}(\mathfrak{p})$ the set of program variables that occur in $\mathfrak{p}$.

Example 6.2.11. Let $\mathcal{L} = \{+, -\}$, where $+, -$ are binary function symbols. We write terms $+(t_0, t_1)$ as $t_0 + t_1$ and similarly for $-$. The sequence $\mathfrak{p}$, given by

$$x \leftarrow x + y;\ y \leftarrow x - y;\ x \leftarrow x - y$$

is a program in $\mathtt{WHILE}_{\mathcal{L}}$. This can be justified by observing that $x \leftarrow x + y$, $y \leftarrow x - y$, and $x \leftarrow x - y$ are assignment statements and therefore are programs and then by a double application of the last rule of Definition 6.2.1, $\mathfrak{p} \in \mathtt{WHILE}_{\mathcal{L}}$. To increase readability, we will write this program as

```
$x \leftarrow x + y$;
$y \leftarrow x - y$;
$x \leftarrow x - y$
```

and we will continue to apply this method of displaying programs from now on. Note that $\mathrm{PVAR}(\mathfrak{p}) = \{x, y\}$. □

Example 6.2.12. Let $\mathcal{L} = \{0, -, =, >\}$, where 0 is a constant symbol, $-$ is a binary function symbol, and $>$ is a binary relation symbol. The following sequence is in $\mathtt{WHILE}_{\mathcal{L}}$:

```
while ($x \neq 0 \wedge y \neq 0$)
    do  if  $x > y$
                then $x \leftarrow x - y$
                else $y \leftarrow y - x$
            endif
endwhile;
if $x = 0$
    then $z \leftarrow y$
    else $z \leftarrow x$
endif
```

We wrote, as usual, $x - y$ in place of $-(x, y)$. ▯

Example 6.2.13. Let

$$\mathcal{L} = \{0, 1, 2, \text{mod}, \text{div}, *, =\}$$

be a first-order language, where $0, 1, 2$ are constant symbols, and $\text{mod}, \text{div}, *$ are binary function symbols. The following sequence is in $\texttt{WHILE}_{\mathcal{L}}$:

```
$z \leftarrow 1$;
 while $(y \neq 0)$
     do $d \leftarrow y$ mod 2;
         $y \leftarrow y$ div 2
          if $d \neq 0$
              then $z \leftarrow z * x$
              else $z \leftarrow z$
          endif
         $x \leftarrow x * x$
 endwhile
```

▯

Theorem 6.2.14. *Let $\mathcal{L}, \mathcal{L}'$ be two first-order languages such that $\mathcal{L} \subseteq \mathcal{L}'$. Then, $\texttt{WHILE}_{\mathcal{L}} \subseteq \texttt{WHILE}_{\mathcal{L}'}$.*

Proof. An easy induction argument on programs in $\texttt{WHILE}_{\mathcal{L}}$ shows that every such program is a program of $\texttt{WHILE}_{\mathcal{L}'}$. □

The definition of WHILE given previously (in Theorem 6.2.2) does meet the unique readability condition. Now we are aiming to provide an alternative definition of this set which meets this condition. Our approach is similar to the definition of $\texttt{WHILE}_{\mathcal{L}}$ that meets unique readability.

Definition 6.2.15. We define inductively the sets $\texttt{WHILE}^{at}$ and $\texttt{WHILE}'$ of sequences of basic symbols and symbols of first-order logic as follows:

(1) Any assignment statement of the form $v \leftarrow t$, where v is a program variable and t is a term such that $\texttt{V}(t) \subseteq \text{PVAR}$ belongs to $\texttt{WHILE}^{at}$.

(2) If β is a quantifier-free formula whose variables are included in PVAR and $\mathfrak{p}$ is in $\mathtt{WHILE}'$, then

$$\textbf{while } \beta \textbf{ do } \mathfrak{p} \textbf{ endwhile}$$

belongs to $\mathtt{WHILE}^{at}$.

(3) If β is a quantifier-free formula whose variables are included in PVAR, and $\mathfrak{p}, \mathfrak{p}' \in \mathtt{WHILE}'$, then

$$\textbf{if } \beta \textbf{ then } \mathfrak{p} \textbf{ else } \mathfrak{p}' \textbf{ endif}$$

belongs to $\mathtt{WHILE}^{at}$.

(4) Every sequence in $\mathtt{WHILE}^{at}$ belongs to $\mathtt{WHILE}'$.

(5) If $\mathfrak{p} \in \mathtt{WHILE}^{at}$ and $\mathfrak{p}' \in \mathtt{WHILE}'$, then $\mathfrak{p}; \mathfrak{p}'$ belongs to $\mathtt{WHILE}'$.

▯

Theorem 6.2.16. *We have* $\mathtt{WHILE} = \mathtt{WHILE}'$.

Proof. The proof is similar to that of Theorem 6.2.5. □

Theorem 6.2.17. *Definition* 6.2.15 *meets the unique readability condition.*

Proof. The proof follows the same pattern as the proof of Theorem 6.2.10. □

6.3 The $\mathtt{WHILE}_{\mathcal{L}}$ Programming Language — Semantics

We want to describe the semantics of running an $\mathcal{L}$-program $\mathfrak{p}$ in an $\mathcal{L}$-structure $\mathcal{A}$. To this end, we need to follow the evolution of the program state, that is, the evolution of the values assigned to variables as the program executes. Recall that in Section 4.5, we defined an assignment over $\mathcal{A}$ to be a function from VAR to $|\mathcal{A}|$, and therefore, we can use the notion of assignment as a formal counterpart of the notion of state of a program. This justifies the next definition.

Definition 6.3.1. Let $\mathcal{A} = (A, \mathcal{I})$ be a $\mathcal{L}$-structure. An $\mathcal{A}$-*state* is an assignment σ in $\mathrm{ASSIGN}_{\mathcal{A}}$. We will denote the set of $\mathcal{A}$-states by $\mathrm{STATES}_{\mathcal{A}}$. ▯

To assign meaning or semantics to $\mathcal{L}$-programs in $\mathcal{L}$-structures, we need to re-examine the meaning of terms and formulas, since they are basic constituents of programs. We can regard the meaning of an $\mathcal{L}$-term in a $\mathcal{L}$-structure $\mathcal{A}$ as a function from states to $|\mathcal{A}|$. We can denote this meaning for a term t by $\mathcal{S}^{term}_{\mathcal{A}}(t)$, where $\mathcal{S}^{term}_{\mathcal{A}} : \mathrm{TERM}_{\mathcal{L}}(\mathrm{PVAR}) \longrightarrow (\mathrm{STATES}_{\mathcal{A}} \longrightarrow A)$ is given by $\mathcal{S}^{term}_{\mathcal{A}}(t)(\sigma) = \sigma^{\mathcal{A}}(t)$.

In Definition 4.5.9, we defined the meaning of an $\mathcal{L}$-formula φ in an $\mathcal{L}$-structure $\mathcal{A}$ as a function $\mathcal{S}_{\mathcal{A}}(\varphi)$ from $\mathcal{A}$-states to **Bool**. In this context, we will denote the meaning function $\mathcal{S}_{\mathcal{A}}$ as $\mathcal{S}^{for}_{\mathcal{A}}$.

We will now define the meaning of an $\mathcal{L}$-program $\mathfrak{p}$ in an $\mathcal{L}$-structure $\mathcal{A}$ as a partial transformation $\mathcal{S}^{pr}_{\mathcal{A}}(\mathfrak{p})$ of the set of $\mathcal{A}$-states into itself. The intention is that for an initial state σ_0, $\mathcal{S}^{pr}_{\mathcal{A}}(\mathfrak{p})(\sigma_0)$ is defined if and only if $\mathfrak{p}$ halts eventually when started in state σ_0. If it halts, then $\mathcal{S}^{pr}_{\mathcal{A}}(\mathfrak{p})(\sigma_0)$ is the state reached when the program halts.

We remind the reader that the composition of two partial functions $f, g : M \rightsquigarrow M$ is again a partial function $f \circ g : M \rightsquigarrow M$. Note that $f \circ g(x)$ is defined if and only if $g(x)$ is defined and $f(g(x))$ is defined. Further, we introduce the iterations of a partial function $f : M \rightsquigarrow M$ as being the functions $f^{(k)}$ for $k \in \mathbf{N}$, defined inductively as follows:

$$f^{(0)} = 1_M(\text{the identity function on } M)$$
$$f^{(k+1)} = f \circ f^{(k)},$$

for $k \in \mathbf{N}$.

To define the function $\mathcal{S}^{pr}_{\mathcal{A}}$, we cannot use Definition 6.2.1 because it does not satisfy the unique readability condition. Thus, we use Definition 6.2.3, which requires us to define simultaneously two partial functions $\mathcal{S}^{at\text{-}pr}_{\mathcal{A}}$ and $\mathcal{S}^{pr}_{\mathcal{A}}$, which give the formal semantics for atomic $\mathcal{L}$-programs and for $\mathcal{L}$-programs, respectively.

Definition 6.3.2. Let $\mathcal{L}$ be a first-order language and let $\mathcal{A}$ be an $\mathcal{L}$-structure. The functions

$$\mathcal{S}^{at\text{-}pr}_{\mathcal{A}} : \mathtt{WHILE}_{\mathcal{L}} \longrightarrow (\mathrm{STATES}_{\mathcal{A}} \rightsquigarrow \mathrm{STATES}_{\mathcal{A}})$$

and

$$\mathcal{S}^{pr}_{\mathcal{A}} : \mathtt{WHILE}_{\mathcal{L}} \longrightarrow (\mathrm{STATES}_{\mathcal{A}} \rightsquigarrow \mathrm{STATES}_{\mathcal{A}}).$$

are given by:

(1) For assignment statements of $\mathcal{L}$ we have

$$\mathcal{S}^{at\text{-}pr}_{\mathcal{A}}(v \leftarrow t)(\sigma) = [v \rightarrow \sigma^{\mathcal{A}}(t)]\sigma.$$

(2) If $\mathfrak{p}_0 =$ **if** β **then** $\mathfrak{p}$ **else** $\mathfrak{p}'$ **endif**, where $\mathfrak{p}, \mathfrak{p}'$ are $\mathcal{L}$-programs and β is a quantifier-free $(\mathcal{L}, \text{PVAR})$-formula, then

$$\mathcal{S}^{at\text{-}pr}_{\mathcal{A}}(\mathfrak{p}_0)(\sigma) = \begin{cases} \mathcal{S}^{pr}_{\mathcal{A}}(\mathfrak{p})(\sigma) & \text{if } \mathcal{S}^{for}_{\mathcal{A}}(\beta)(\sigma) = \mathbf{T} \\ \mathcal{S}^{pr}_{\mathcal{A}}(\mathfrak{p}')(\sigma) & \text{if } \mathcal{S}^{for}_{\mathcal{A}}(\beta)(\sigma) = \mathbf{F}. \end{cases}$$

(3) If $\mathfrak{p}_0 =$ **while** β **do** $\mathfrak{p}$ **endwhile**, where $\mathfrak{p}$ is an $\mathcal{L}$-program, β is a quantifier-free $(\mathcal{L}, \text{PVAR})$-formula, and $\sigma \in \text{STATES}_{\mathcal{A}}$, let $k \in \mathbf{N}$ be the least number such that there is a state $\sigma_k = \left(\mathcal{S}^{pr}_{\mathcal{A}}(\mathfrak{p})\right)^{(k)}(\sigma)$ and $\mathcal{S}^{for}_{\mathcal{A}}(\beta)(\sigma_k) = \mathbf{F}$, if such a number exists. When the number k exists, $\mathcal{S}^{at\text{-}pr}_{\mathcal{A}}(\mathfrak{p}_0)(\sigma) = \sigma_k$; otherwise, $\mathcal{S}^{at\text{-}pr}_{\mathcal{A}}(\mathfrak{p}_0)(\sigma)$ is undefined.

(4) For every atomic $\mathcal{L}$-program $\mathfrak{p}$ we have:

$$\mathcal{S}^{pr}_{\mathcal{A}}(\mathfrak{p}) = \mathcal{S}^{at\text{-}pr}_{\mathcal{A}}(\mathfrak{p}).$$

(5) If $\mathfrak{p}$ is an atomic $\mathcal{L}$-program and $\mathfrak{p}'$ is an $\mathcal{L}$-program, then

$$\mathcal{S}^{pr}_{\mathcal{A}}(\mathfrak{p};\mathfrak{p}') = \mathcal{S}^{pr}_{\mathcal{A}}(\mathfrak{p}') \circ \mathcal{S}^{at\text{-}pr}_{\mathcal{A}}(\mathfrak{p}).$$

If $\mathcal{S}^{pr}_{\mathcal{A}}(\mathfrak{p})(\sigma)$ is defined, then we say that program $\mathfrak{p}$ *halts when started in $\mathcal{A}$-state σ.* □

For an atomic program $\mathfrak{p}_0 =$ **while** β **do** $\mathfrak{p}$ **endwhile**, $\mathcal{S}^{at\text{-}pr}_{\mathcal{A}}(\mathfrak{p}_0)(\sigma)$ may be undefined for two reasons: either $\sigma_k = \left(\mathcal{S}^{at\text{-}pr}_{\mathcal{A}}(\mathfrak{p})\right)^{(k)}(\sigma)$ is defined and $\mathcal{S}^{for}_{\mathcal{A}}(\beta)(\sigma_k) = \mathbf{T}$ for all k, or there is a least k such that $\left(\mathcal{S}^{at\text{-}pr}_{\mathcal{A}}(\mathfrak{p})\right)^{(k)}(\sigma)$ is undefined and for all $i < k$, if $\sigma_i = \left(\mathcal{S}^{at\text{-}pr}_{\mathcal{A}}(\mathfrak{p})\right)^{(i)}(\sigma)$, then $\mathcal{S}^{for}_{\mathcal{A}}(\beta)(\sigma_i) = \mathbf{T}$. In the first case, the infinite looping is caused by the topmost **while** loop in $\mathfrak{p}_0$; in the second case, the infinite looping is due to infinite looping that occurs in $\mathfrak{p}$.

Theorem 6.3.3. *Let $\mathcal{L}$ be a first-order language and let $\mathfrak{p}, \mathfrak{p}'$ be $\mathcal{L}$-programs. Then, $\mathcal{S}^{pr}_{\mathcal{A}}(\mathfrak{p};\mathfrak{p}') = \mathcal{S}^{pr}_{\mathcal{A}}(\mathfrak{p}') \circ \mathcal{S}^{pr}_{\mathcal{A}}(\mathfrak{p})$.*

Proof. The argument is by induction on the $\mathcal{L}$-program $\mathfrak{p}$ using Definition 6.2.3. The basis step when $\mathfrak{p}$ is atomic, follows immediately from the last two parts of Definition 6.3.2.

Suppose that $\mathfrak{p} = \mathfrak{p}_1; \mathfrak{p}''$ where $\mathfrak{p}_1$ is atomic and the statement holds for $\mathfrak{p}''$. Then,

$$\begin{aligned}
\mathcal{S}^{pr}_{\mathcal{A}}(\mathfrak{p}; \mathfrak{p}') &= \mathcal{S}^{pr}_{\mathcal{A}}(\mathfrak{p}_1; \mathfrak{p}''; \mathfrak{p}') \\
&= \mathcal{S}^{pr}_{\mathcal{A}}(\mathfrak{p}''; \mathfrak{p}') \circ \mathcal{S}^{at\text{-}pr}_{\mathcal{A}}(\mathfrak{p}_1) \\
&= (\mathcal{S}^{pr}_{\mathcal{A}}(\mathfrak{p}') \circ \mathcal{S}^{pr}_{\mathcal{A}}(\mathfrak{p}'')) \circ \mathcal{S}^{at\text{-}pr}_{\mathcal{A}}(\mathfrak{p}_1) \\
&= \mathcal{S}^{pr}_{\mathcal{A}}(\mathfrak{p}') \circ (\mathcal{S}^{pr}_{\mathcal{A}}(\mathfrak{p}'') \circ \mathcal{S}^{at\text{-}pr}_{\mathcal{A}}(\mathfrak{p}_1)) \\
&= \mathcal{S}^{pr}_{\mathcal{A}}(\mathfrak{p}') \circ \mathcal{S}^{pr}_{\mathcal{A}}(\mathfrak{p}_1; \mathfrak{p}'') \\
&= \mathcal{S}^{pr}_{\mathcal{A}}(\mathfrak{p}') \circ \mathcal{S}^{pr}_{\mathcal{A}}(\mathfrak{p}),
\end{aligned}$$

which completes the argument. □

Corollary 6.3.4. *Let $\mathcal{L}$ be a first-order language and let $\mathfrak{p}_0, \mathfrak{p}_1, \ldots, \mathfrak{p}_{n-1}$ be $\mathcal{L}$-programs, for $n \geq 1$. Then,*

$$\mathcal{S}^{pr}_{\mathcal{A}}(\mathfrak{p}_0; \mathfrak{p}_1; \cdots ; \mathfrak{p}_{n-1}) = \mathcal{S}^{pr}_{\mathcal{A}}(\mathfrak{p}_{n-1}) \circ \cdots \circ \mathcal{S}^{pr}_{\mathcal{A}}(\mathfrak{p}_1) \circ \mathcal{S}^{pr}_{\mathcal{A}}(\mathfrak{p}_0).$$

Proof. The corollary can be proven immediately by induction on n. □

Theorem 6.3.5. *Let $\mathcal{L}$ be a first-order language and let $\mathfrak{p}$ be an $\mathcal{L}$-program. If x is a variable such that $x \notin PVAR(\mathfrak{p})$, $\mathcal{A}$ is an $\mathcal{L}$-structure, and $\sigma, \sigma' \in \mathrm{STATES}_{\mathcal{A}}$ are such that $\sigma' = \mathcal{S}^{pr}_{\mathcal{A}}(\mathfrak{p})(\sigma)$, then $\sigma'(x) = \sigma(x)$.*

Proof. The argument is by induction on $\mathcal{L}$-programs and is left to the reader. □

Corollary 6.3.6. *Let $\mathcal{L}$ be a first-order language and let $\mathfrak{p}$ be a $\mathcal{L}$-program. If x is a specification variable, $\mathcal{A}$ is an $\mathcal{L}$-structure, and $\sigma, \sigma' \in \mathrm{STATES}_{\mathcal{A}}$ are such that $\sigma' = \mathcal{S}^{pr}_{\mathcal{A}}(\mathfrak{p})(\sigma)$, then $\sigma'(x) = \sigma(x)$.*

Proof. This is a direct consequence of Theorem 6.3.5. □

From a semantical point of view, two states that agree on all the variables that occur in a program can be considered indiscernible. This is justified by the following theorem.

Theorem 6.3.7 (State Agreement Theorem for Programs). *Let $\mathcal{L}$ be a first-order language and let $\mathfrak{p}_0$ be an $\mathcal{L}$-program. If $\sigma, \sigma' \in$*

STATES$_{\mathcal{A}}$ *and* $\sigma(x) = \sigma'(x)$ *for all* $x \in PVAR(\mathfrak{p}_0)$*, then* $\mathfrak{p}_0$ *halts when started in state* σ *if and only if it halts when started in state* σ' *and if* $\mathfrak{p}_0$ *halts when started in these states, then* $\mathcal{S}^{pr}_{\mathcal{A}}(\mathfrak{p}_0)(\sigma)(x) = \mathcal{S}^{pr}_{\mathcal{A}}(\mathfrak{p}_0)(\sigma')(x)$ *for all variables* x *such that* $\sigma(x) = \sigma'(x)$.

Proof. The argument is by induction on the program $\mathfrak{p}_0$ using Definition 6.2.3. In all steps, we assume that σ, σ' are as in the statement and that x is a variable such that $\sigma(x) = \sigma'(x)$.

For the basis step, let $\mathfrak{p}_0$ be the assignment statement $v{\leftarrow}t$, where v is a program variable and t is an $(\mathcal{L}, \mathrm{PVAR})$-term. Since $\mathfrak{p}_0$ halts when started in any state, the first part of the statement is immediate. Note that $\sigma^{\mathcal{A}}(t) = \sigma'^{\mathcal{A}}(t)$ by Theorem 4.5.3. Thus, we can write

$$\begin{aligned}
\mathcal{S}^{pr}_{\mathcal{A}}(\mathfrak{p}_0)(\sigma)(x) &= ([v \to \sigma^{\mathcal{A}}(t)]\sigma)^{\mathcal{A}}(x) \\
&= \begin{cases} \sigma(x) & \text{if } x \neq v \\ \sigma^{\mathcal{A}}(t) & \text{if } x = v \end{cases} \\
&= \begin{cases} \sigma'(x) & \text{if } x \neq v \\ \sigma'^{\mathcal{A}}(t) & \text{if } x = v \end{cases} \\
&= ([v \to \sigma'^{\mathcal{A}}(t)]\sigma')^{\mathcal{A}}(x) \\
&= \mathcal{S}^{pr}_{\mathcal{A}}(\mathfrak{p}_0)(\sigma')(x).
\end{aligned}$$

There are three inductive steps. For the first one, let

$$\mathfrak{p}_0 = \textbf{ if } \beta \textbf{ then } \mathfrak{p}_1 \textbf{ else } \mathfrak{p}_2 \textbf{ endif}$$

where the statement holds for the programs $\mathfrak{p}_1$ and $\mathfrak{p}_2$. There are two subcases to consider. For the first subcase, suppose that $\mathcal{S}^{for}_{\mathcal{A}}(\beta)(\sigma) = \mathbf{T}$. Then, $\mathcal{S}^{for}_{\mathcal{A}}(\beta)(\sigma') = \mathbf{T}$ because σ, σ' agree on all the variables of β. Thus, by the definition of $\mathcal{S}^{pr}_{\mathcal{A}}$, we have $\mathcal{S}^{pr}_{\mathcal{A}}(\mathfrak{p}_0)(\sigma) = \mathcal{S}^{pr}_{\mathcal{A}}(\mathfrak{p}_1)(\sigma)$ and $\mathcal{S}^{pr}_{\mathcal{A}}(\mathfrak{p}_0)(\sigma') = \mathcal{S}^{pr}_{\mathcal{A}}(\mathfrak{p}_1)(\sigma')$. The states σ and σ' agree on all variables of PVAR$(\mathfrak{p}_1)$ because they agree on all variables in PVAR$(\mathfrak{p}_0)$. By inductive hypothesis, $\mathfrak{p}_0$ halts when started in state σ if and only if it halts when started in σ'. Moreover, if the program halts, by the same hypothesis, we have

$$\mathcal{S}^{pr}_{\mathcal{A}}(\mathfrak{p}_0)(\sigma)(x) = \mathcal{S}^{pr}_{\mathcal{A}}(\mathfrak{p}_1)(\sigma)(x) = \mathcal{S}^{pr}_{\mathcal{A}}(\mathfrak{p}_1)(\sigma')(x) = \mathcal{S}^{pr}_{\mathcal{A}}(\mathfrak{p}_0)(\sigma')(x).$$

The second subcase, when $\mathcal{S}^{for}_{\mathcal{A}}(\beta)(\sigma) = \mathbf{F}$ is similar and left to the reader.

Suppose now that $\mathfrak{p}_0 =$ **while** β **do** $\mathfrak{p}$ **endwhile**, where the statement holds for $\mathfrak{p}$. By induction on $k \in \mathbf{N}$, one can easily prove that for all $\sigma, \sigma' \in \text{STATES}_{\mathcal{A}}$ such that $\sigma(x) = \sigma'(x)$ for $x \in \text{PVAR}(\mathfrak{p})$, $(\mathcal{S}^{pr}_{\mathcal{A}}(\mathfrak{p}))^{(k)}(\sigma)$ is defined if and only if $(\mathcal{S}^{pr}_{\mathcal{A}}(\mathfrak{p}))^{(k)}(\sigma')$ is defined and, if defined these states agree on all variables on which σ and σ' agree.

Suppose that $\mathcal{S}^{pr}_{\mathcal{A}}(\mathfrak{p}_0)(\sigma)$ is defined. Let k be the least number such that $\sigma_k = (\mathcal{S}^{pr}_{\mathcal{A}}(\mathfrak{p}))^{(k)}(\sigma)$ is defined and $\mathcal{S}^{for}_{\mathcal{A}}(\beta)(\sigma_k) = \mathbf{F}$. Then, by the previous paragraph, $\sigma'_k = (\mathcal{S}^{pr}_{\mathcal{A}}(\mathfrak{p}))^{(k)}(\sigma')$ is defined and σ_k, σ'_k agree on all variables on which σ, σ' agree and those include all variables of β. Therefore, by the Agreement Theorem for First-Order Logic (Theorem 4.5.12), $\mathcal{S}^{for}_{\mathcal{A}}(\beta)(\sigma'_k) = \mathbf{F}$. Note that if $\ell < k$, then $\sigma_\ell = (\mathcal{S}^{pr}_{\mathcal{A}}(\mathfrak{p}))^{(\ell)}(\sigma)$ is defined and $\mathcal{S}^{for}_{\mathcal{A}}(\beta)(\sigma_\ell) = \mathbf{T}$. So for $\ell < k$, $\sigma'_\ell = (\mathcal{S}^{pr}_{\mathcal{A}}(\mathfrak{p}))^{(\ell)}(\sigma')$ is defined and $\mathcal{S}^{for}_{\mathcal{A}}(\beta)(\sigma'_\ell) = \mathbf{T}$. Thus, $\mathcal{S}^{pr}_{\mathcal{A}}(\mathfrak{p}_0)(\sigma') = \sigma'_k$ and σ_k, σ'_k agree on all variables on which σ, σ' agree. Similarly, if $\mathcal{S}^{pr}_{\mathcal{A}}(\mathfrak{p}_0)(\sigma')$ is defined, the corresponding conclusion follows for $\mathcal{S}^{pr}_{\mathcal{A}}(\mathfrak{p}_0)(\sigma)$.

We leave to the reader the case when $\mathfrak{p}_0 = \mathfrak{p}_1; \mathfrak{p}_2$, where the statement holds for $\mathfrak{p}_1, \mathfrak{p}_2$. □

Theorem 6.3.8. *Let $\mathcal{L}, \mathcal{L}'$ be two first-order languages such that $\mathcal{L} \subseteq \mathcal{L}'$, let $\mathfrak{p}$ be an $\mathcal{L}$-program, and let $\mathcal{A}'$ be an $\mathcal{L}'$-structure. If $\mathcal{A}$ is the reduct of $\mathcal{A}'$ to $\mathcal{L}$, then $\mathcal{S}^{pr}_{\mathcal{A}}(\mathfrak{p}) = \mathcal{S}^{pr}_{\mathcal{A}'}(\mathfrak{p})$.*

Proof. The argument is by induction on $\mathcal{L}$-programs and makes use of Theorem 4.5.24. □

A companion to the semantics of programs is the *running time function* $\mathsf{time}^{pr}_{\mathcal{A}} : \mathtt{WHILE}_{\mathcal{L}} \longrightarrow (\text{STATES}_{\mathcal{A}} \rightsquigarrow \mathbf{N})$, where $\mathsf{time}^{pr}_{\mathcal{A}}(\mathfrak{p})(\sigma)$ is the running time of the program $\wp$ when started in the state $\sigma \in \text{STATES}_{\mathcal{A}}$. This function is defined by induction on programs.

Definition 6.3.9. Let $\mathcal{L}$ be a first-order language and let $\mathcal{A}$ be an $\mathcal{L}$-structure. The functions $\mathsf{time}^{at\text{-}pr}_{\mathcal{A}}$ and $\mathsf{time}^{pr}_{\mathcal{A}}$ are given by:

(1) For assignment statements of $\mathcal{L}$, we have:

$$\mathsf{time}^{at\text{-}pr}_{\mathcal{A}}\mathbf{(}v{\leftarrow}t\mathbf{)}(\sigma) = 1.$$

(2) If $\mathfrak{p}_0 =$ **if** β **then** $\mathfrak{p}$ **else** $\mathfrak{p}'$ **endif**, where $\mathfrak{p}, \mathfrak{p}'$ are $\mathcal{L}$-programs, then

$$\mathsf{time}^{at\text{-}pr}_{\mathcal{A}}\mathbf{(}\mathfrak{p}_0\mathbf{)}(\sigma) = \begin{cases} 1 + \mathsf{time}^{pr}_{\mathcal{A}}\mathbf{(}\mathfrak{p}\mathbf{)}(\sigma) & \text{if } \mathcal{S}^{for}_{\mathcal{A}}\mathbf{(}\beta\mathbf{)}(\sigma) = \mathbf{T} \\ 1 + \mathsf{time}^{pr}_{\mathcal{A}}\mathbf{(}\mathfrak{p}'\mathbf{)}(\sigma) & \text{if } \mathcal{S}^{for}_{\mathcal{A}}\mathbf{(}\beta\mathbf{)}(\sigma) = \mathbf{F}. \end{cases}$$

(3) Let $\mathfrak{p}_0 =$ **while** β **do** $\mathfrak{p}$ **endwhile**, where $\mathfrak{p}$ is an $\mathcal{L}$-program, β is a quantifier-free $(\mathcal{L}, \mathrm{PVAR})$-formula, and $\sigma \in \mathrm{STATES}_{\mathcal{A}}$. Let $k \in \mathbf{N}$ be the least number such that there is a state $\sigma_k = \left(\mathcal{S}^{pr}_{\mathcal{A}}\mathbf{(}\mathfrak{p}\mathbf{)}\right)^{(k)}(\sigma)$ and $\mathcal{S}^{for}_{\mathcal{A}}\mathbf{(}\beta\mathbf{)}(\sigma_k) = \mathbf{F}$, if such a number exists. When the number k exists, $\mathsf{time}^{at\text{-}pr}_{\mathcal{A}}\mathbf{(}\mathfrak{p}_0\mathbf{)}(\sigma) = k + 1 + \sum_{i=0}^{k-1} \mathsf{time}^{pr}_{\mathcal{A}}\mathbf{(}\mathfrak{p}\mathbf{)}(\sigma_i)$; otherwise, $\mathsf{time}^{at\text{-}pr}_{\mathcal{A}}\mathbf{(}\mathfrak{p}_0\mathbf{)}(\sigma)$ is undefined.

(4) For every atomic $\mathcal{L}$-program $\mathfrak{p}$ we have:

$$\mathsf{time}^{pr}_{\mathcal{A}}\mathbf{(}\mathfrak{p}\mathbf{)} = \mathsf{time}^{at\text{-}pr}_{\mathcal{A}}\mathbf{(}\mathfrak{p}\mathbf{)}.$$

(5) If $\mathfrak{p}$ is an atomic $\mathcal{L}$-program and $\mathfrak{p}'$ is an $\mathcal{L}$-program, then

$$\mathsf{time}^{pr}_{\mathcal{A}}\mathbf{(}\mathfrak{p};\mathfrak{p}'\mathbf{)}(\sigma) = \mathsf{time}^{at\text{-}pr}_{\mathcal{A}}\mathbf{(}\mathfrak{p}\mathbf{)}(\sigma) + \mathsf{time}^{pr}_{\mathcal{A}}\mathbf{(}\mathfrak{p}'\mathbf{)}(\mathcal{S}^{at\text{-}pr}_{\mathcal{A}}\mathbf{(}\mathfrak{p}\mathbf{)}(\sigma)).$$

□

Theorem 6.3.10. *Let $\mathcal{L}$ be a first-order language and $\mathcal{A}$ be an $\mathcal{L}$-structure. For all atomic $\mathcal{L}$-programs $\mathfrak{p}$, $\mathcal{L}$-programs $\mathfrak{q}$ and states σ, the functions $\mathsf{time}^{at\text{-}pr}_{\mathcal{A}}\mathbf{(}\mathfrak{p}\mathbf{)}(\sigma)$ and $\mathsf{time}^{pr}_{\mathcal{A}}\mathbf{(}\mathfrak{q}\mathbf{)}(\sigma)$ are defined if and only if $\mathcal{S}^{at\text{-}pr}_{\mathcal{A}}\mathbf{(}\mathfrak{p}\mathbf{)}(\sigma)$ and $\mathcal{S}^{pr}_{\mathcal{A}}\mathbf{(}\mathfrak{q}\mathbf{)}(\sigma)$ are defined, respectively.*

Proof. The argument is by induction on programs and is left to the reader. □

Theorem 6.3.11 (Time Agreement Theorem for Programs). *Let $\mathcal{L}$ be a first-order language and let $\mathfrak{p}_0$ be an $\mathcal{L}$-program. If $\sigma, \sigma' \in \mathrm{STATES}_{\mathcal{A}}$ and $\sigma(x) = \sigma'(x)$ for all $x \in PVAR(\mathfrak{p}_0)$, then $\mathsf{time}^{pr}_{\mathcal{A}}\mathbf{(}\mathfrak{p}_0\mathbf{)}(\sigma)$ is defined if and only if $\mathsf{time}^{pr}_{\mathcal{A}}\mathbf{(}\mathfrak{p}_0\mathbf{)}(\sigma')$ is defined and if both are defined, they are equal.*

Proof. The fact that $\mathsf{time}^{pr}_{\mathcal{A}}(\mathfrak{p}_0)(\sigma)$ is defined if and only if $\mathsf{time}^{pr}_{\mathcal{A}}(\mathfrak{p}_0)(\sigma')$ is defined follows from Theorems 6.3.7 and 6.3.10. The equality

$$\mathsf{time}^{pr}_{\mathcal{A}}(\mathfrak{p}_0)(\sigma) = \mathsf{time}^{pr}_{\mathcal{A}}(\mathfrak{p}_0)(\sigma')$$

can be shown by induction on $\mathfrak{p}_0$ and is similar to the argument of Theorem 6.3.7. □

Example 6.3.12. Let $\mathcal{L} = \{+, -\}$ be the first-order language introduced in Example 6.2.11 and let $\mathcal{A}$ be the $\mathcal{L}$-structure where $|\mathcal{A}| = \mathbf{Z}$ and $+^{\mathcal{A}}, -^{\mathcal{A}}$ are the addition and subtraction functions on $\mathbf{Z}$. Consider the program $\mathfrak{p}$ mentioned in the same example and let $\sigma \in \mathrm{STATES}_{\mathcal{A}}$ be such that $\sigma(x) = m$ and $\sigma(y) = n$.

By Corollary 6.3.4,

$$\mathcal{S}^{pr}_{\mathcal{A}}(\mathfrak{p})(\sigma) = \mathcal{S}^{at\text{-}pr}_{\mathcal{A}}(x\leftarrow x - y)(\mathcal{S}^{at\text{-}pr}_{\mathcal{A}}(y\leftarrow x - y)(\mathcal{S}^{at\text{-}pr}_{\mathcal{A}}(x\leftarrow x + y)(\sigma)))$$

Consider the states

$$\begin{aligned}
\sigma_1 &= \mathcal{S}^{at\text{-}pr}_{\mathcal{A}}(x\leftarrow x + y)(\sigma) \ = [x \to \sigma^{\mathcal{A}}(x + y)]\sigma,\\
\sigma_2 &= \mathcal{S}^{at\text{-}pr}_{\mathcal{A}}(y\leftarrow x - y)(\sigma_1) = [y \to \sigma_1^{\mathcal{A}}(x - y)]\sigma_1,\\
\sigma_3 &= \mathcal{S}^{at\text{-}pr}_{\mathcal{A}}(x\leftarrow x - y)(\sigma_2) = [x \to \sigma_2^{\mathcal{A}}(x - y)]\sigma_2.
\end{aligned}$$

Since $\sigma_1 = [x \to m + n]\sigma$, we have $\sigma_1(x) = m + n$ and $\sigma_1(y) = n$. Further, we have $\sigma_2 = [y \to m]\sigma_1$ and this gives $\sigma_2(x) = m + n$ and $\sigma_2(y) = m$. Finally, we have $\sigma_3 = [x \to n]\sigma_2$, which implies $\sigma_3(x) = n$ and $\sigma_3(y) = m$. This shows that the effect of the program is to swap the values of the variables x and y.

For the time function, we have $\mathsf{time}^{pr}_{\mathcal{A}}(\mathfrak{p})(\sigma) = 3$, for every state σ, as the reader can easily verify. □

Example 6.3.13. Let $\mathcal{L}$ be the first-order language introduced in Example 6.2.12 and let $\mathfrak{p}$ be the program introduced in the same example. We can write $\mathfrak{p} = \mathfrak{p}_0; \mathfrak{p}_1$, where $\mathfrak{p}_0$ is the program:

```
while ($x \neq 0 \wedge y \neq 0$)
    do  if  $x > y$
              then $x\leftarrow x - y$
              else $y\leftarrow y - x$
        endif
endwhile
```

and $\mathfrak{p}_1$ is the program:

```
if $x = 0$
    then $z \leftarrow y$
    else $z \leftarrow x$
endif
```

Consider the structure $\mathcal{A}$, where $|\mathcal{A}| = \mathbf{Z}$, $0^{\mathcal{A}} = 0$, $-^{\mathcal{A}}$ is the subtraction operation on $\mathbf{Z}$ and $>^{\mathcal{A}}$ is the "greater than" relation on $\mathbf{Z}$. If $\mathfrak{p}'$ is **if** $x > y$ **then** $x \leftarrow x - y$ **else** $y \leftarrow y - x$ **endif**, then

$$\mathcal{S}^{pr}_{\mathcal{A}}(\mathfrak{p}')(\sigma) = \begin{cases} [x \to \sigma(x) - \sigma(y)]\sigma & \text{if } \sigma(x) > \sigma(y) \\ [y \to \sigma(y) - \sigma(x)]\sigma & \text{if } \sigma(x) \leq \sigma(y). \end{cases}$$

Let $\sigma_k = \left(\mathcal{S}^{pr}_{\mathcal{A}}(\mathfrak{p}')\right)^{(k)}(\sigma)$ for $k \geq 0$. If, say, $\sigma(x) = 6$ and $\sigma(y) = 8$, then $\mathfrak{p}_0$ goes through the states shown in the next table.

k	$\sigma_k(x)$	$\sigma_k(y)$	$\mathcal{S}^{for}_{\mathcal{A}}(x \neq 0 \wedge y \neq 0)(\sigma_k)$
0	6	8	**T**
1	6	2	**T**
2	4	2	**T**
3	2	2	**T**
4	2	0	**F**

Therefore, we have $\sigma_4 = \mathcal{S}^{pr}_{\mathcal{A}}(\mathfrak{p}_0)(\sigma)$. At completion, the program $\mathfrak{p}$ reaches the state

$$\sigma' = \mathcal{S}^{pr}_{\mathcal{A}}(\mathfrak{p})(\sigma) = \mathcal{S}^{pr}_{\mathcal{A}}(\mathfrak{p}_1)(\mathcal{S}^{pr}_{\mathcal{A}}(\mathfrak{p}_0)(\sigma)) = \mathcal{S}^{pr}_{\mathcal{A}}(\mathfrak{p}_1)(\sigma_4).$$

Since $\mathcal{S}^{for}_{\mathcal{A}}(x = 0)(\sigma_4) = \mathbf{F}$, we have

$$\begin{aligned} \sigma'(z) &= \sigma_4(x) = 2 \\ \sigma'(x) &= \sigma_4(x) = 2 \\ \sigma'(y) &= \sigma_4(y) = 0. \end{aligned}$$

Suppose now that $\bar{\sigma}$ is an $\mathcal{A}$-state such that $\bar{\sigma}(x) = 4$ and $\bar{\sigma}(y) = -2$. It is easy to verify that in this case $\bar{\sigma}_k(x) = 4 + 2k$ and $\bar{\sigma}_k(y) = -2$, where $\bar{\sigma}_k = \left(\mathcal{S}^{pr}_{\mathcal{A}}(\mathfrak{p}')\right)^{(k)}(\bar{\sigma})$.

Therefore, $\mathcal{S}^{for}_{\mathcal{A}}(x \neq 0 \wedge y \neq 0)(\bar{\sigma}_k) = \mathbf{T}$ for $k \in \mathbf{N}$ so $\mathcal{S}^{pr}_{\mathcal{A}}(\mathfrak{p}_0)(\bar{\sigma})$ is undefined, which means that $\mathcal{S}^{pr}_{\mathcal{A}}(\mathfrak{p})(\bar{\sigma})$ is undefined.

Informally, the program $\mathfrak{p}$, when run in the structure $\mathcal{A}$ with the initial values of x and y limited to natural numbers, is a “slow” implementation of Euclid’s Algorithm for finding the greatest common divisor of two natural numbers when defined (that is, when they are not both equal to 0). The basic idea is that at each execution of the body of the **while** loop the greatest common divisor of x and y remains the same and the sum of x and y decreases. Thus, when the smaller of the two numbers reaches 0 (and one can prove that this happens eventually), the larger equals the greatest common divisor.

For the running time, function, we can write

$$\mathsf{time}^{pr}_{\mathcal{A}}(\mathfrak{p}_0;\mathfrak{p}_1)(\sigma) = \mathsf{time}^{at\text{-}pr}_{\mathcal{A}}(\mathfrak{p}_0)(\sigma) + \mathsf{time}^{pr}_{\mathcal{A}}(\mathfrak{p}_1)(\mathcal{S}^{at\text{-}pr}_{\mathcal{A}}(\mathfrak{p}_0)(\sigma)).$$

The running time for the program $\mathfrak{p}_0$ and the state σ is given by

$$\mathsf{time}^{at\text{-}pr}_{\mathcal{A}}(\mathfrak{p}_0)(\sigma) = 5 + \sum_{k=0}^{3} \mathsf{time}^{pr}_{\mathcal{A}}(\mathfrak{p}')(\sigma_k).$$

By the third part of Definition 6.3.9, $\mathsf{time}^{pr}_{\mathcal{A}}(\mathfrak{p}')(\sigma_k) = 2$, for $0 \leq k \leq 3$, so $\mathsf{time}^{at\text{-}pr}_{\mathcal{A}}(\mathfrak{p}_0)(\sigma) = 13$. Further, for the same reason, $\mathsf{time}^{at\text{-}pr}_{\mathcal{A}}(\mathfrak{p}_1)(\sigma_4) = 2$, so $\mathsf{time}^{pr}_{\mathcal{A}}(\mathfrak{p})(\sigma) = 15$. ∎

Example 6.3.14. Let $\mathcal{L}$ be the first-order language introduced in Example 6.2.13. The program $\mathfrak{p}$ introduced in that example can be written as $\mathfrak{p} = z{\leftarrow}1;\mathfrak{p}_1$, where $\mathfrak{p}_1$ is

```
while ($y \neq 0$)
    do $d \leftarrow y$ mod 2;
       $y \leftarrow y$ div 2
        if $d \neq 0$
            then $z \leftarrow z * x$
            else $z \leftarrow z$
        endif
       $x \leftarrow x * x$
endwhile
```

Define the $\mathcal{L}$-structure $\mathcal{A}$ by $|\mathcal{A}| = \mathbf{N}$, $0^{\mathcal{A}} = 0, 1^{\mathcal{A}} = 1, 2^{\mathcal{A}} = 2$,

$$m \ \mathrm{div}^{\mathcal{A}} \ n = \begin{cases} \lfloor \frac{m}{n} \rfloor & \text{if } n > 0 \\ 0 & \text{otherwise,} \end{cases}$$

$$m \ \mathrm{mod}^{\mathcal{A}} \ n = m - (m \ \mathrm{div}^{\mathcal{A}} \ n) \cdot n$$

and $*^{\mathcal{A}}$ is the usual multiplication of natural numbers.

Suppose that $\mathfrak{p}$ starts with a state σ where $\sigma(x) = 3$ and $\sigma(y) = 6$ and let $\sigma_0 = \mathcal{S}^{pr}_{\mathcal{A}}(z{\leftarrow}1)(\sigma) = [z \to 1]\sigma$. Let $\mathfrak{p}'$ be the body of $\mathfrak{p}_1$. Define $\sigma_k = \left(\mathcal{S}^{pr}_{\mathcal{A}}(\mathfrak{p}')\right)^{(k)}(\sigma_0)$ for $k \geq 1$. The successive σ_k are shown below.

k	$\sigma_k(x)$	$\sigma_k(y)$	$\sigma_k(z)$	$\sigma_k(d)$	$\mathcal{S}^{for}_{\mathcal{A}}(y \neq 0)(\sigma_k)$
0	3	6	1	$\sigma(d)$	**T**
1	9	3	1	0	**T**
2	81	1	9	1	**T**
3	6561	0	729	1	**F**

Therefore, we have $\sigma_3 = \mathcal{S}^{pr}_{\mathcal{A}}(\mathfrak{p})(\sigma)$, where $\sigma_3(z) = 729 = 3^6$.

Informally, suppose that $a = \sigma(x)$ and $b = \sigma(y)$. Let $d_n \cdots d_1 d_0$ be the binary expansion of b, that is $b = 2^n d_n + \cdots 2^1 d_1 + d_0$. This allows us to write $a^b = \left(a^{2^n}\right)^{d_n} \cdot \cdots \cdot \left(a^{2^1}\right)^{d_1} \cdot a^{d_0}$, so only those a^{2^j} where $d_j = 1$ contribute to a^b. Note also that $a^{2^{j+1}} = (a^{2^j})^2$. Thus the successive values of x contain the numbers $a, a^2, \ldots, a^{2^j}, \ldots$. Those a^{2^j} for which $d_j = 1$ are multiplied into z, so z will contain eventually a^b. □

6.4 Functions Computable by Programs

Programs in $\mathtt{WHILE}_{\mathcal{L}}$ can be used to compute functions over $\mathcal{L}$-structures.

Definition 6.4.1. Let $\mathcal{A}$ be an $\mathcal{L}$-structure.

A partial function $g : |\mathcal{A}|^n \rightsquigarrow |\mathcal{A}|$ is *computable by an $\mathcal{L}$-program $\mathfrak{p}$ with the sequence of distinct variables* $(y_0, \ldots, y_n)$ if for every state $\sigma \in \mathrm{STATES}_{\mathcal{A}}$, $\mathcal{S}^{pr}_{\mathcal{A}}(\mathfrak{p})(\sigma)$ is defined if and only if

$g(\sigma(y_0),\ldots,\sigma(y_{n-1}))$ is defined and when $\mathcal{S}^{pr}_{\mathcal{A}}\mathbf{(}\mathfrak{p}\mathbf{)}(\sigma)$ is defined, we have

$$g(\sigma(y_0),\ldots,\sigma(y_{n-1})) = \mathcal{S}^{pr}_{\mathcal{A}}\mathbf{(}\mathfrak{p}\mathbf{)}(\sigma)(y_n).$$

If in addition when $\mathcal{S}^{pr}_{\mathcal{A}}\mathbf{(}\mathfrak{p}\mathbf{)}(\sigma)$ is defined we have

$$\mathcal{S}^{pr}_{\mathcal{A}}\mathbf{(}\mathfrak{p}\mathbf{)}(\sigma)(y_i) = \sigma(y_i)$$

for $0 \leq i \leq n-1$, then we say that $\mathfrak{p}$ *computes g preserving inputs.* □

Example 6.4.2. Let $\mathcal{L} = \{0, 1, 2, \mathrm{mod}, \mathrm{div}, *, =\}$ be the first-order language introduced in Example 6.2.13 and let $\mathcal{A}$ be the $\mathcal{L}$-structure introduced in Example 6.3.14. The $\mathcal{L}$-program introduced in the first example, together with the sequence of variables (x, y, z) computes the function $g : \mathbf{N}^2 \longrightarrow \mathbf{N}$ given by $g(m, n) = m^n$, as we will show in Example 6.5.8. □

Example 6.4.3. Let $\mathcal{L}_{pra} = \{=, <, 0, s, +\}$ be the first-order language of Presburger arithmetic. The $\mathcal{L}_{pra}$-program $\mathfrak{p}$ is defined as

```
$z \leftarrow 0$;
$w \leftarrow 0$;
 **while** $(w \neq y)$
      **do** $z \leftarrow z + x$;
           $w \leftarrow s(w)$
 **endwhile**
```

We will show that $\mathfrak{p}$ and the sequence of variables (x, y, z) computes the multiplication function $h : \mathbf{N}^2 \longrightarrow \mathbf{N}$ given by $h(m, n) = mn$.

Let σ be an $\mathcal{A}_{pra}$-state, $\sigma' = \mathcal{S}^{pr}_{\mathcal{A}_{pra}}\mathbf{(}z \leftarrow 0\mathbf{)}(\sigma)$, and $\sigma'' = \mathcal{S}^{pr}_{\mathcal{A}_{pra}}\mathbf{(}w \leftarrow 0\mathbf{)}(\sigma')$. Also, denote by $\mathfrak{p}_0$ the program $z \leftarrow z + x; w \leftarrow s(w)$ and by $\mathfrak{p}_1$ the program **while** $(w \neq y)$ **do** $\mathfrak{p}_0$ **endwhile**. The states $\sigma_k = (\mathcal{S}^{pr}_{\mathcal{A}_{pra}}\mathbf{(}\mathfrak{p}_0\mathbf{)})^{(k)}(\sigma'')$ are defined for all $k \in \mathbf{N}$. Observe that $\sigma' = [z \to 0]\sigma$ and $\sigma'' = [w \to 0][z \to 0]\sigma$. We claim that for all

$k \in \mathbf{N}$, we have

$$\begin{aligned}\sigma_k(z) &= k\sigma(x)\\ \sigma_k(w) &= k\\ \sigma_k(x) &= \sigma(x)\\ \sigma_k(y) &= \sigma(y).\end{aligned}$$

The proof is by induction on k. The basis step, $k = 0$, is immediate because $\sigma_0 = \sigma''$. For the inductive step, suppose that these statements hold for k. If $\sigma'_{k+1} = [z \to \sigma_k^{\mathcal{A}_{pra}}(z + x)]\sigma_k$, then, by inductive hypothesis, we have $\sigma'_{k+1} = [z \to \sigma_k(z) + \sigma_k(x)]\sigma_k = [z \to (k+1)\sigma_k(x)]\sigma_k = [z \to (k+1)\sigma(x)]\sigma_k$. Further,

$$\begin{aligned}\sigma_{k+1} &= [w \to \sigma'^{\mathcal{A}_{pra}}_{k+1}(s(w))]\sigma'_{k+1}\\ &= [w \to s(\sigma'_{k+1}(w))][z \to (k+1)\sigma(x)]\sigma_k\\ &= [w \to k+1][z \to (k+1)\sigma(x)]\sigma_k,\end{aligned}$$

which shows that the above equalities hold for σ_{k+1}.

It follows that the least number k such that $(\mathcal{A}_{pra}, \sigma_k) \not\models (w \neq y)$ is the number $k = \sigma(y)$ and thus $\mathcal{S}^{pr}_{\mathcal{A}_{pra}}(\mathfrak{p}_1)(\sigma'') = \sigma_{\sigma(y)}$. Now, the final state of the program will be

$$\begin{aligned}&\mathcal{S}^{pr}_{\mathcal{A}_{pra}}(\mathfrak{p}_1)(\mathcal{S}^{pr}_{\mathcal{A}_{pra}}(w\leftarrow 0)(\mathcal{S}^{pr}_{\mathcal{A}_{pra}}(z\leftarrow 0)(\sigma)))\\ &= \mathcal{S}^{pr}_{\mathcal{A}_{pra}}(\mathfrak{p}_1)(\mathcal{S}^{pr}_{\mathcal{A}_{pra}}(w\leftarrow 0)(\sigma'))\\ &= \mathcal{S}^{pr}_{\mathcal{A}_{pra}}(\mathfrak{p}_1)(\sigma'')\\ &= \sigma_{\sigma(y)}.\end{aligned}$$

Thus $\mathcal{S}^{pr}_{\mathcal{A}_{pra}}(\mathfrak{p})(\sigma)(z) = \sigma_{\sigma(y)}(z) = \sigma(y)\sigma(z)$, as desired. □

Example 6.4.4. Let $\mathcal{L}_{ar} = \{=, <, 0, s, +, \cdot\}$ be the first-order language of arithmetic. The $\mathcal{L}_{ar}$-program

```
$z \leftarrow s(0)$;
$w \leftarrow 0$;
 while $(w \neq y)$
      do $z \leftarrow z \cdot x$;
          $w \leftarrow s(w)$
 endwhile
```

and the sequence of variables (x, y, z) computes the same function g as the one in Example 6.4.2, as the reader can easily verify. This latest way of computing is less efficient than the one of Example 6.4.2, but it is easier to justify. □

Example 6.4.5. Let $\mathsf{b} : \mathbf{N}^3 \longrightarrow \mathbf{N}$ be the function given by $\mathsf{b}(c, d, i) = r$ if $c = r \mod ((i+1)d+1)$; in other words, we have $\mathsf{b}(c, d, i) = r$ if and only if $(c, d, i, r) \in B$, where B is the Gödel relation introduced in Theorem 4.7.7.

The following $\mathcal{L}_{ar}$-program

```
$z \leftarrow s(s(x_i) \cdot x_d)$;
$w \leftarrow 0$;
 while $(s(w) \cdot z \leq x_c)$
       do $w \leftarrow s(w)$
 endwhile;
$x_r \leftarrow 0$;
 while $(w \cdot z + x_r < x_c)$
       do $x_r \leftarrow s(x_r)$
 endwhile
```

computes the function b with the sequence of variables (x_c, x_d, x_i, x_r), as the reader can easily verify. □

We want to show that if a program $\mathfrak{p}$ computes an n-ary function g with a sequence of variables $(y_0, \ldots, y_n)$, then the program $s(\mathfrak{p})$, where s is an appropriate substitution which renames the program variables, computes the same function with the sequence of variables $(s(y_0), \ldots, s(y_n))$. To be more specific about the type of substitution we are interested in, we give the following definition.

Definition 6.4.6. Let $\mathcal{L}$ be a first-order language. An *$\mathcal{L}$-program substitution* is an injective $\mathcal{L}$-substitution s such that $s(\mathrm{VAR}) \subseteq \mathrm{VAR}$ and $s(\mathrm{PVAR}) \subseteq \mathrm{PVAR}$. □

Note that if σ is an $\mathcal{A}$-state and s is an $\mathcal{L}$-program substitution, then σs is also an $\mathcal{A}$-state. Also, we have $\sigma s = \sigma^{\mathcal{A}} s$, because the range of s consists of variables.

Theorem 6.4.7. *If $\mathfrak{p}$ is an $\mathcal{L}$-program and s is a $\mathcal{L}$-program substitution, then $\overline{s}(\mathfrak{p})$ is also an $\mathcal{L}$-program.*

Proof. The proof is by induction on programs, and we leave it to the reader. □

To justify formally the claim made before Definition 6.4.6, we need to prove the following result.

Theorem 6.4.8. *Let $\mathcal{L}$ be a first-order language, $\mathfrak{p}$ be an $\mathcal{L}$-program, and s be an $\mathcal{L}$-program substitution. For every $\mathcal{L}$-structure $\mathcal{A}$ and $\mathcal{A}$-state σ, we have*

$$\mathcal{S}_{\mathcal{A}}^{pr}(\mathfrak{p})(\sigma s) = (\mathcal{S}_{\mathcal{A}}^{pr}(\overline{s}(\mathfrak{p}))(\sigma))s.$$

Proof. The argument is by induction on programs. For the basis step, suppose that $\mathfrak{p}$ is $v \leftarrow t$. By Definition 6.3.2, we have

$$\begin{aligned}
\mathcal{S}_{\mathcal{A}}^{at\text{-}pr}(\mathfrak{p})(\sigma s) &= [v \rightarrow (\sigma s)^{\mathcal{A}}(t)](\sigma s) \\
(\mathcal{S}_{\mathcal{A}}^{at\text{-}pr}(\overline{s}(\mathfrak{p}))(\sigma))s &= ([\overline{s}(v) \rightarrow \sigma^{\mathcal{A}}(\overline{s}(t))]\sigma)s \\
&= ([s(v) \rightarrow \sigma^{\mathcal{A}}(\overline{s}(t))]\sigma)s.
\end{aligned}$$

By Lemma 4.6.2, we have $\sigma^{\mathcal{A}}\overline{s} = (\sigma^{\mathcal{A}} s)^{\mathcal{A}}$. Since s is an $\mathcal{L}$-program substitution, we have $\sigma^{\mathcal{A}} s = \sigma s$, when we regard s as a transformation of variables. Thus, $\sigma^{\mathcal{A}}\overline{s}(t) = (\sigma s)^{\mathcal{A}}(t)$ and we denote their common value by a. We need to show that $[v \rightarrow a](\sigma s) = ([s(v) \rightarrow a]\sigma)s$. If w is a variable distinct from v, then $[v \rightarrow a](\sigma s)(w) = \sigma(s(w))$ and $([s(v) \rightarrow a]\sigma)s(w) = \sigma(s(w))$ because $s(v) \neq s(w)$. On the other hand, $[v \rightarrow a](\sigma s)(v) = a$ and $(([s(v) \rightarrow a]\sigma)s)(v) = a$.

We discuss only the inductive step when $\mathfrak{p}$ is **while** β **do** $\mathfrak{q}$ **endwhile**, where the inductive hypothesis holds for $\mathfrak{q}$. In other words, we have

$$\mathcal{S}_{\mathcal{A}}^{pr}(\mathfrak{q})(\sigma s) = (\mathcal{S}_{\mathcal{A}}^{pr}(\overline{s}(\mathfrak{q}))(\sigma))s, \tag{6.1}$$

for all $\mathcal{A}$-states σ. We have $\overline{s}(\mathfrak{p}) =$ **while** $\overline{s}(\beta)$ **do** $\overline{s}(\mathfrak{q})$ **endwhile**. We claim first that for all $k \geq 0$, $(\mathcal{S}_{\mathcal{A}}^{pr}(\mathfrak{q}))^{(k)}(\sigma s)$ is defined if and only if $(\mathcal{S}_{\mathcal{A}}^{pr}(\overline{s}(\mathfrak{q})))^{(k)}(\sigma)$ is defined and, if both are defined, then $(\mathcal{S}_{\mathcal{A}}^{pr}(\mathfrak{q}))^{(k)}(\sigma s) = ((\mathcal{S}_{\mathcal{A}}^{pr}(\overline{s}(\mathfrak{q})))^{(k)}(\sigma))s$. We show this by induction on k. The basis step, $k = 0$ is immediate.

Suppose that $(\mathcal{S}^{pr}_{\mathcal{A}}(\mathfrak{q}))^{(k+1)}(\sigma s)$ is defined. We have

$$\begin{aligned}(\mathcal{S}^{pr}_{\mathcal{A}}(\mathfrak{q}))^{(k+1)}(\sigma s) &= \mathcal{S}^{pr}_{\mathcal{A}}(\mathfrak{q})((\mathcal{S}^{pr}_{\mathcal{A}}(\mathfrak{q}))^{(k)}(\sigma s))\\ &= \mathcal{S}^{pr}_{\mathcal{A}}(\mathfrak{q})(((\mathcal{S}^{pr}_{\mathcal{A}}(\overline{s}(\mathfrak{q})))^{(k)}(\sigma))s)\\ &\quad \text{(by inductive hypothesis on } k)\\ &= (\mathcal{S}^{pr}_{\mathcal{A}}(\overline{s}(\mathfrak{q}))((\mathcal{S}^{pr}_{\mathcal{A}}(\overline{s}(\mathfrak{q})))^{(k)}(\sigma)))s\\ &\quad \text{(by Equality (6.1))}\\ &= ((\mathcal{S}^{pr}_{\mathcal{A}}(\overline{s}(\mathfrak{q})))^{(k+1)}(\sigma))s,\end{aligned}$$

which shows that $\mathcal{S}^{pr}_{\mathcal{A}}(\overline{s}(\mathfrak{q})))^{(k+1)}(\sigma)$ is defined and the desired equality holds for $k+1$.

The same equality can be reached assuming that $(\mathcal{S}^{pr}_{\mathcal{A}}(\overline{s}(\mathfrak{q})))^{(k+1)}$ (σ) is defined and in the process of proving the equality, the definedness of $(\mathcal{S}^{pr}_{\mathcal{A}}(\mathfrak{q}))^{(k+1)}(\sigma s)$ would follow.

We need to justify a second claim, namely, that if $(\mathcal{S}^{pr}_{\mathcal{A}}(\mathfrak{q}))^{(k)}(\sigma s)$ and $(\mathcal{S}^{pr}_{\mathcal{A}}(\overline{s}(\mathfrak{q})))^{(k)}(\sigma)$ are defined, then $(\mathcal{A}, (\mathcal{S}^{pr}_{\mathcal{A}}(\mathfrak{q}))^{(k)}(\sigma s)) \models \beta$ if and only if $(\mathcal{A}, (\mathcal{S}^{pr}_{\mathcal{A}}(\overline{s}(\mathfrak{q})))^{(k)}(\sigma)) \models \overline{s}(\beta)$.

The following four statements are equivalent:

(i) $(\mathcal{A}, (\mathcal{S}^{pr}_{\mathcal{A}}(\mathfrak{q}))^{(k)}(\sigma s)) \models \beta$;
(ii) $(\mathcal{A}, ((\mathcal{S}^{pr}_{\mathcal{A}}(\overline{s}(\mathfrak{q})))^{(k)}(\sigma))s) \models \beta$;
(iii) $(\mathcal{A}, ((\mathcal{S}^{pr}_{\mathcal{A}}(\overline{s}(\mathfrak{q})))^{(k)}(\sigma))^{\mathcal{A}}s) \models \beta$;
(iv) $(\mathcal{A}, (\mathcal{S}^{pr}_{\mathcal{A}}(\overline{s}(\mathfrak{q})))^{(k)}(\sigma)) \models \overline{s}(\beta)$.

The equivalence between (i) and (ii) follows from the first claim of this proof. The equivalence between (ii) and (iii) is a consequence of the fact that the range of s is a set of variables. Finally, the equivalence between (iii) and (iv) follows from Corollary 4.6.5 and from the fact that $\text{FVSubst}(s, \beta) = \overline{s}(\beta)$ because β is a quantifier-free formula.

Suppose that $\mathcal{S}^{pr}_{\mathcal{A}}(\mathfrak{p})(\sigma s)$ is defined. Let k_0 be the least number k such that $\sigma_k = (\mathcal{S}^{pr}_{\mathcal{A}}(\mathfrak{q}))^{(k)}(\sigma s)$ is defined and $(\mathcal{A}, \sigma_k) \not\models \beta$, in other words, $\mathcal{S}^{pr}_{\mathcal{A}}(\mathfrak{p})(\sigma s) = \sigma_{k_0}$. By the previous claims, k_0 is also the least number k such that $\sigma'_k = (\mathcal{S}^{pr}_{\mathcal{A}}(\overline{s}(\mathfrak{q})))^{(k)}(\sigma)$ is defined and $(\mathcal{A}, \sigma'_k) \not\models \overline{s}(\beta)$. Thus, $\mathcal{S}^{pr}_{\mathcal{A}}(\overline{s}(\mathfrak{p}))(\sigma)$ is defined and equals σ'_{k_0}. By the first claim, we have $\sigma_{k_0} = \sigma'_{k_0}s$, which is the desired conclusion. Notice that the same conclusion can be reached in reverse if we assume that $\mathcal{S}^{pr}_{\mathcal{A}}(\overline{s}(\mathfrak{p}))(\sigma)$ is defined. □

Corollary 6.4.9. *If the $\mathcal{L}$-program $\mathfrak{p}$ computes the n-ary function g on an $\mathcal{L}$-structure $\mathcal{A}$ with the sequence of variables $(y_0, \ldots, y_n)$ and s is an $\mathcal{L}$-program substitution, then $\overline{s}(\mathfrak{p})$ computes the same function g with the sequence of variables $(s(y_0), \ldots, s(y_n))$.*

Proof. Since $\mathfrak{p}$ computes g in $\mathcal{A}$, we have

$$\mathcal{S}^{pr}_{\mathcal{A}}(\mathfrak{p})(\sigma)(y_n) = g(\sigma(y_0), \ldots, \sigma(y_{n-1})) \tag{6.2}$$

for every $\mathcal{A}$-state σ. By Theorem 6.4.8, we have $\mathcal{S}^{pr}_{\mathcal{A}}(\overline{s}(\mathfrak{p}))(\sigma)(s(y_n)) = \mathcal{S}^{pr}_{\mathcal{A}}(\mathfrak{p})(\sigma s)(y_n)$. Replacing σ with σs in Equality (6.2), we have

$$\mathcal{S}^{pr}_{\mathcal{A}}(\mathfrak{p})(\sigma s)(y_n) = g(\sigma(s(y_0)), \ldots, \sigma(s(y_{n-1})))$$

which gives the desired result. □

The next corollary shows that for a function computable in a structure, we can specify the sequence of variables involved.

Corollary 6.4.10. *Let g be an n-ary function on an $\mathcal{L}$-structure $\mathcal{A}$ that is computable by an $\mathcal{L}$-program with some sequence of variables. For any sequence of distinct program variables $z = (z_0, \ldots, z_{n-1}, z_n)$, there is an $\mathcal{L}$-program $\mathfrak{p}_z$ that computes g with the sequence z.*

Proof. Let $\mathfrak{p}$ be an $\mathcal{L}$-program that computes g with the sequence of variables $(y_0, \ldots, y_{n-1}, y_n)$. Consider an $\mathcal{L}$-program substitution s such that $s(y_i) = z_i$ for $0 \leq i \leq n$. Such a substitution exists because the members of the sequence z are pairwise distinct program variables. Then, by Corollary 6.4.9, the program $\mathfrak{p}_z = s(\mathfrak{p})$ computes the function g with z. □

A stronger version of Corollary 6.4.10 is given next.

Corollary 6.4.11. *Let g be an n-ary function on an $\mathcal{L}$-structure $\mathcal{A}$ that is computable by an $\mathcal{L}$-program with some sequence of variables. For any sequence of distinct program variables $z = (z_0, \ldots, z_{n-1}, z_n)$, there is an $\mathcal{L}$-program $\mathfrak{p}_z$ that computes g with the sequence z preserving inputs.*

Proof. By Corollary 6.4.10, there is an $\mathcal{L}$-program $\mathfrak{p}_z{}'$ that computes g with the sequence z. If $n = 0$, then $\mathfrak{p}_z{}'$ computes g with inputs preserved because there are no inputs, so we may assume that $n > 0$.

Consider a program substitution s such $s(x) \notin \{z_0, \ldots, z_{n-1}\}$ for $x \in \mathrm{PVAR}(\mathfrak{p}_z{}') \cup \{z_0, \ldots, z_{n-1}\}$. By Corollary 6.4.9, $s(\mathfrak{p}_z{}')$ computes the same function g with the sequence of variables $(s(z_0), \ldots, s(z_n))$. Define the program $\mathfrak{p}_z$ as

$$s(z_0) \leftarrow z_0; \cdots ; s(z_{n-1}) \leftarrow z_{n-1}; s(\mathfrak{p}_z{}'); z_n \leftarrow s(z_n)$$

Let $\sigma \in \mathrm{ASSIGN}_{\mathcal{A}}$ and

$$\sigma_1 = \mathcal{S}^{pr}_{\mathcal{A}} \boldsymbol{(} s(z_0) \leftarrow z_0; \cdots ; s(z_{n-1}) \leftarrow z_{n-1} \boldsymbol{)} (\sigma).$$

Since $\{z_0, \ldots, z_{n-1}\} \cap \{s(z_0), \ldots, s(z_{n-1})\} = \emptyset$, we have

$$(\sigma_1(s(z_0)), \ldots, \sigma_1(s(z_{n-1}))) = (\sigma(z_0), \ldots, \sigma(z_{n-1})).$$

If $g(\sigma(z_0), \ldots, \sigma(z_{n-1}))$ is defined, then $\mathcal{S}^{pr}_{\mathcal{A}} \boldsymbol{(} s(\mathfrak{p}_z{}') \boldsymbol{)} (\sigma_1) = \sigma_2$ is defined, because $s(\mathfrak{p}_z{}')$ computes g with $(s(z_0), \ldots, s(z_{n-1}))$, and

$$\sigma_2(s(z_n)) = g(\sigma_1(s(z_0)), \ldots, \sigma_1(s(z_{n-1}))) = g(\sigma(z_0), \ldots, \sigma(z_{n-1})).$$

Thus,

$$\sigma_3 = \mathcal{S}^{pr}_{\mathcal{A}} \boldsymbol{(} z_n \leftarrow s(z_n) \boldsymbol{)} (\sigma_2) = \mathcal{S}^{pr}_{\mathcal{A}} \boldsymbol{(} s(\mathfrak{p}_z) \boldsymbol{)} (\sigma)$$

is defined and $\sigma_3(z_n) = \sigma_2(s(z_n)) = g(\sigma(z_0), \ldots, \sigma(z_{n-1}))$.

If $g(\sigma(z_0), \ldots, \sigma(z_{n-1})) = g(\sigma_1(s(z_0)), \ldots, \sigma_1(s(z_{n-1})))$ is undefined, it follows that $\mathcal{S}^{pr}_{\mathcal{A}} \boldsymbol{(} s(\mathfrak{p}_z{}') \boldsymbol{)} (\sigma_1)$ is undefined, so $\mathcal{S}^{pr}_{\mathcal{A}} \boldsymbol{(} s(\mathfrak{p}_z) \boldsymbol{)} (\sigma)$ is undefined. Thus, $\mathfrak{p}_z$ defines g with $(z_0, \ldots, z_n)$.

Note that $\mathfrak{p}_z$ is input preserving. Indeed, observe that for $0 \leq i \leq n-1$, $z_i \notin \{s(z_0), \ldots, s(z_{n-1}), z_n\}$. Also, z_i does not appear in $s(\mathfrak{p}_z{}')$ and hence, by Theorem 6.3.5, if $\mathcal{S}^{pr}_{\mathcal{A}} \boldsymbol{(} \mathfrak{p}_z \boldsymbol{)} (\sigma)$ is defined, then $\mathcal{S}^{pr}_{\mathcal{A}} \boldsymbol{(} \mathfrak{p}_z \boldsymbol{)} (\sigma)(z_i) = \sigma(z_i)$. □

Let $\mathcal{L}$ be a first-order language and let $\mathcal{A}$ be an $\mathcal{L}$-structure. Once we have shown that a (total) n-ary function $g : |\mathcal{A}|^n \longrightarrow |\mathcal{A}|$ can be computed by an $\mathcal{L}$-program $\mathfrak{r}_g$ with a sequence $(y_0, \ldots, y_n)$, we can use the function as a "macro". This is done by adding an n-ary function symbol f_g to $\mathcal{L}$ and then using f_g to write programs in this expanded language. Moreover, we can translate effectively programs in the expanded language into programs in the original language by using a technique that emulates macro expansion in standard programming languages.

Let $\mathcal{L}' = \mathcal{L} \cup \{f_g\}$ be the first-order language obtained by adding f_g to $\mathcal{L}$. Define $\mathcal{A}'$ to be the expansion of $\mathcal{A}$ to $\mathcal{L}'$ defined by $f_g^{\mathcal{A}'} = g$. For an $\mathcal{L}'$-program $\mathfrak{p}$, let S be a finite set of variables such that $\mathrm{PVAR}(\mathfrak{p}) \subseteq S$. We seek to define an $\mathcal{L}$-program $\mathfrak{q}(\mathfrak{p}, S)$ such that for every $\sigma \in \mathrm{STATES}_{\mathcal{A}}$, $\mathcal{S}^{pr}_{\mathcal{A}'}(\mathfrak{p})(\sigma)$ is defined if and only if $\mathcal{S}^{pr}_{\mathcal{A}}(\mathfrak{q}(\mathfrak{p}, S))(\sigma)$ is defined and if both are defined, then these states agree on all the variables of S. In the remainder of this section, we use the notations introduced in this paragraph.

Theorem 6.4.12. *Let t be an $(\mathcal{L}', PVAR)$-term, S be a finite set of program variables such that $VAR(t) \subseteq S$ and let v be a program variable not in S. Then, one can construct effectively an $\mathcal{L}$-program $\mathfrak{p}_{t,S,v}$ such that for every $\sigma \in \mathrm{STATES}_{\mathcal{A}}$, $\sigma_1 = \mathcal{S}^{pr}_{\mathcal{A}}(\mathfrak{p}_{t,S,v})(\sigma)$ is defined, $\sigma_1(v) = \sigma^{\mathcal{A}'}(t)$, and $\sigma_1(x) = \sigma(x)$ for every $x \in S$.*

Proof. The argument is by induction on $(\mathcal{L}', \mathrm{PVAR})$-terms. For the basis step, let t be the program variable y. The program $\mathfrak{p}_{t,S,v}$ is $v \leftarrow y$. It is clear by Definition 6.3.2 that this program satisfies the conditions of the theorem.

For the inductive step, suppose that t begins with the m-ary function symbol f, that is, $t = f$ if $m = 0$ and $t = f(t_0, \ldots, t_{m-1})$ if $m > 0$. For the latter case, assume that the statement holds for $t_0, \ldots, t_{m-1}$.

Initially, suppose that $m > 0$. Since S is a finite set of variables, we can select m program variables $v_0, \ldots, v_{m-1}$ not in S and distinct from v. The set $S \cup \{v_0, \ldots, v_{i-1}\}$ is denoted by S_i for $0 \leq i \leq m-1$. By the inductive hypothesis, there are programs $\mathfrak{p}_{t_i,S_i,v_i}$ for $0 \leq i \leq m-1$ that satisfy the conditions of the statement. Let $\mathfrak{r}_{t,S}$ be the program

$$\begin{array}{l}
\mathfrak{p}_{t_0,S_0,v_0};\\
\mathfrak{p}_{t_1,S_1,v_1};\\
\vdots\\
\mathfrak{p}_{t_{m-1},S_{m-1},v_{m-1}}
\end{array}$$

By Corollary 6.3.4, we can write

$$\mathcal{S}^{pr}_{\mathcal{A}}(\mathfrak{r}_{t,S}) = \mathcal{S}^{pr}_{\mathcal{A}}(\mathfrak{p}_{t_{m-1},S_{m-1},v_{m-1}}) \circ \cdots \circ \mathcal{S}^{pr}_{\mathcal{A}}(\mathfrak{p}_{t_0,S_0,v_0}).$$

Let τ_i be the $\mathcal{A}$-state given by

$$\tau_i = \mathcal{S}^{pr}_{\mathcal{A}}\boldsymbol{(}\mathfrak{p}_{t_i,S_i,v_i}\boldsymbol{)}(\cdots \mathcal{S}^{pr}_{\mathcal{A}}\boldsymbol{(}\mathfrak{p}_{t_0,S_0,v_0}\boldsymbol{)}(\sigma)\cdots)$$

for $0 \leq i \leq m-1$. By the inductive hypothesis, the states τ_i are defined. Further, we can show by induction on i, left to the reader, that we have $\tau_i(v_j) = \sigma^{\mathcal{A}'}(t_j)$ for $0 \leq j \leq i \leq m-1$ and $\tau_i(x) = \sigma(x)$ for $x \in S$. In particular, we have $\tau_{m-1}(v_j) = \sigma^{\mathcal{A}'}(t_j)$ for $0 \leq j \leq m-1$ and $\tau_{m-1}(x) = \sigma(x)$ for $x \in S$. Also recall that $\tau_{m-1} = \mathcal{S}^{pr}_{\mathcal{A}}\boldsymbol{(}\mathfrak{r}_{t,S}\boldsymbol{)}(\sigma)$.

We now continue the argument for all values of m including $m = 0$. First, let us assume that $f \neq f_g$. The program $\mathfrak{p}_{t,S,v}$ can be defined as $\mathfrak{r}_{t,S}; v\leftarrow f(v_0,\ldots,v_{m-1})$, if $m > 0$ and as $v\leftarrow f$, if $m = 0$. The desired conclusion is obvious when $m = 0$. Suppose now that $m > 0$. When we start with the state σ, the state which is obtained after the execution of the program is $\sigma_1 = [v \to \tau^{\mathcal{A}}_{m-1}(f(v_0,\ldots,v_{m-1}))]\tau_{m-1}$. Since

$$\begin{aligned}
\tau^{\mathcal{A}}_{m-1}(f(v_0,\ldots,v_{m-1})) &= f^{\mathcal{A}}(\tau_{m-1}(v_0),\ldots,\tau_{m-1}(v_{m-1})) \\
&= f^{\mathcal{A}}(\sigma^{\mathcal{A}'}(t_0),\ldots,\sigma^{\mathcal{A}'}(t_{m-1})) \\
&= \sigma^{\mathcal{A}'}(f(t_0,\ldots,t_{m-1})) \\
&= \sigma^{\mathcal{A}'}(t),
\end{aligned}$$

we obtain $\sigma_1 = [v \to \sigma^{\mathcal{A}'}(t)]\tau_{m-1}$. The state σ_1 meets the requirements of the theorem in view of the properties of τ_{m-1} that we have shown above.

Now assume that $f = f_g$, so $m = n$. Recall that we assumed that the function g is computed by the program $\mathfrak{r}_g$ with the sequence $(y_0,\ldots,y_n)$. Consider a program substitution s such that $s(y_i) = v_i$ for $0 \leq i \leq n-1$, $s(y_n) = v$, and for all $x \in \mathrm{PVAR}(\mathfrak{r}_g)$, $s(x) \notin S$. The program $\mathfrak{p}_{t,S,v}$ is defined as $\mathfrak{r}_{t,S}; \overline{s}(\mathfrak{r}_g)$, when $m = n > 0$ and as $\overline{s}(\mathfrak{r}_g)$, if $m = n = 0$. As before, when $m = n > 0$, let $\tau_{m-1} = \mathcal{S}^{pr}_{\mathcal{A}}\boldsymbol{(}\mathfrak{r}_{t,S}\boldsymbol{)}(\sigma)$. By Corollary 6.4.9, the program $\overline{s}(\mathfrak{r}_g)$ computes the function g with $(s(y_0),\ldots,s(y_{n-1}),s(y_n)) = (v_0,\ldots,v_{n-1},v)$. Since g is a total function, $\sigma_1 = \mathcal{S}^{pr}_{\mathcal{A}}\boldsymbol{(}\overline{s}(\mathfrak{r}_g)\boldsymbol{)}(\tau_{m-1})$ is defined. This is the state that the program $\mathfrak{p}_{t,S,v}$ ends in when started with the state σ. Therefore, we have

$$\begin{aligned}\sigma_1(v) &= g(\tau_{m-1}(v_0), \ldots, \tau_{m-1}(v_{n-1}))\\ &= g(\sigma^{\mathcal{A}'}(t_0), \ldots, \sigma^{\mathcal{A}'}(t_{m-1}))\\ &= \sigma^{\mathcal{A}'}(f_g(t_0, \ldots, t_{m-1}))\\ &= \sigma^{\mathcal{A}'}(t).\end{aligned}$$

Also, we have $\sigma_1(x) = \sigma(x)$ for all $x \in S$ due to Theorem 6.3.5 and the fact that $\tau_{m-1}(x) = \sigma(x)$ as we proved above.

If $m = n = 0$, again, let σ_1 be the state that the program $\mathfrak{p}_{t,S,v}$ ends in when started with the state σ. Now we have $\sigma_1 = \mathcal{S}^{pr}_{\mathcal{A}}(\overline{s}(\mathfrak{r}_g))(\sigma)$, hence, $\sigma_1(v) = g() = \sigma^{\mathcal{A}'}(f_g) = \sigma^{\mathcal{A}'}(t)$. Theorem 6.3.5 allows us again to conclude that $\sigma_1(x) = \sigma(x)$ for all $x \in S$. □

Using the notations introduced before Theorem 6.4.12, we have the following statement.

Theorem 6.4.13. *Let β be a quantifier-free $(\mathcal{L}', PVAR)$-formula and S be a finite set of program variables such that $\mathtt{FV}(\beta) \subseteq S$. We can effectively construct an $\mathcal{L}$-program $\mathfrak{p}_{\beta,S}$ and a quantifier-free $(\mathcal{L}, PVAR)$-formula $\gamma(\beta, S)$ such that for every state $\sigma \in \mathrm{STATES}_{\mathcal{A}}$, the state $\sigma_1 = \mathcal{S}^{pr}_{\mathcal{A}}(\mathfrak{p}_{\beta,S})(\sigma)$ is defined, $\sigma_1(x) = \sigma(x)$ for every $x \in S$, and $(\mathcal{A}', \sigma) \models \beta$ if and only if $(\mathcal{A}, \sigma_1) \models \gamma(\beta, S)$.*

Proof. The proof is by induction on the formula β. Suppose that β is R, where R is a 0-ary relation symbol. Since f_g does not appear in β, we can define $\gamma(\beta, S) = \beta$ and $\mathfrak{p}_{\beta,S}$ to be $v \leftarrow v$ for some variable v, that is, a do-nothing program. If β is an atomic formula $R(t_0, \ldots, t_{n-1})$ with $n > 0$, then let $v_0, \ldots, v_{n-1}$ be n distinct program variables that do not belong to S and let $\gamma(\beta, S) = R(v_0, \ldots, v_{n-1})$. The program $\mathfrak{p}_{\beta,S}$ consists of the sequence of programs

$$\begin{aligned}&\mathfrak{p}_{t_0,S_0,v_0};\\ &\mathfrak{p}_{t_1,S_1,v_1};\\ &\vdots\\ &\mathfrak{p}_{t_{n-1},S_{n-1},v_{n-1}}\end{aligned}$$

Here, $S_i = S \cup \{v_0, \ldots, v_{i-1}\}$, for $0 \leq i \leq n-1$ and $\mathfrak{p}_{t_i,S_i,v_i}$ is the program defined in Theorem 6.4.12. As in the proof of that theorem, the state σ_1 is always defined, $\sigma_1(x) = \sigma(x)$ for $x \in S$ and $\sigma_1(v_i) = \sigma^{\mathcal{A}'}(t_i)$ for $0 \leq i \leq n-1$. Further, the following statements are equivalent:

(1) $(\mathcal{A}, \sigma_1) \models R(v_0, \ldots, v_{n-1}) = \gamma(\beta, S)$;
(2) $(\sigma_1(v_0), \ldots, \sigma_1(v_{n-1})) \in R^{\mathcal{A}}$;
(3) $(\sigma^{\mathcal{A}'}(t_0), \ldots, \sigma^{\mathcal{A}'}(t_{n-1})) \in R^{\mathcal{A}'}$;
(4) $(\mathcal{A}', \sigma) \models R(t_0, \ldots, t_{n-1}) = \beta$.

We discuss here only one of the inductive steps, namely, when $\beta = (\beta_0 C \beta_1)$ for some binary connective symbol C, and the conclusion holds for β_0 and β_1. If $\mathtt{FV}(\beta) \subseteq S$, then we have $\mathtt{FV}(\beta_0) \subseteq S$, so we obtain by the inductive hypothesis, a program $\mathfrak{p}_{\beta_0,S}$ and a quantifier-free formula $\gamma(\beta_0, S)$. Letting $S' = S \cup \mathtt{FV}(\gamma(\beta_0, S))$, we have $\mathtt{FV}(\beta_1) \subseteq S'$, so by applying the inductive hypothesis we obtain the existence of a program $\mathfrak{p}_{\beta_1,S'}$ and a quantifier-free formula $\gamma(\beta_1, S')$.

Define the program $\mathfrak{p}_{\beta,S} = \mathfrak{p}_{\beta_1,S'}; \mathfrak{p}_{\beta_0,S}$ and the quantifier-free formula $\gamma(\beta, S) = (\gamma(\beta_0, S) C \gamma(\beta_1, S'))$. By Theorem 6.3.3 and the inductive hypothesis, $\mathcal{S}^{pr}_{\mathcal{A}}(\mathfrak{p}_{\beta,S})(\sigma)$ is defined for σ. For $\sigma \in \mathrm{STATES}_{\mathcal{A}}$, let $\sigma'_1 = \mathcal{S}^{pr}_{\mathcal{A}}(\mathfrak{p}_{\beta_0,S})(\sigma)$ and $\sigma_1 = \mathcal{S}^{pr}_{\mathcal{A}}(\mathfrak{p}_{\beta_1,S'})(\sigma'_1) = \mathcal{S}^{pr}_{\mathcal{A}}(\mathfrak{p}_{\beta,S})(\sigma)$. By inductive hypothesis, for all $x \in S$, we have $\sigma_1(x) = \sigma'_1(x) = \sigma(x)$, so in particular, σ'_1 and σ agree on $\mathtt{FV}(\beta_1)$. Also by inductive hypothesis, σ_1 and σ'_1 agree on $\mathtt{FV}(\gamma(\beta_0, S))$.

For any $\sigma \in \mathrm{STATES}_{\mathcal{A}}$, we have $(\mathcal{A}', \sigma) \models \beta_0$ if and only if $(\mathcal{A}, \sigma'_1) \models \gamma(\beta_0, S)$ (by inductive hypothesis) which is equivalent to $(\mathcal{A}, \sigma_1) \models \gamma(\beta_0, S)$ (by Theorem 4.5.12). Also, $(\mathcal{A}', \sigma) \models \beta_1$ if and only if $(\mathcal{A}', \sigma'_1) \models \beta_1$ which is equivalent to $(\mathcal{A}, \sigma_1) \models \gamma(\beta_1, S')$. It follows that $(\mathcal{A}', \sigma) \models \beta = (\beta_0 C \beta_1)$ if and only if $(\mathcal{A}, \sigma_1) \models \gamma(\beta, S) = (\gamma(\beta_0, S) C \gamma(\beta_1, S))$.

We leave the remaining inductive step to the reader. □

Theorem 6.4.14. *Let $\mathfrak{p}$ be an $\mathcal{L}'$-program and S be a finite set of variables that contains $PVAR(\mathfrak{p})$. There is an effective construction that yields an $\mathcal{L}$-program $\mathfrak{q}(\mathfrak{p}, S)$ such that for all $\sigma, \tau \in \mathrm{STATES}_{\mathcal{A}}$ with $\sigma(x) = \tau(x)$ for every $x \in S$, $\mathcal{S}^{pr}_{\mathcal{A}'}(\mathfrak{p})(\sigma)$ is defined if and only if $\mathcal{S}^{pr}_{\mathcal{A}}(\mathfrak{q}(\mathfrak{p}, S))(\tau)$ is defined and if both are defined, then these states agree on all the variables of S.*

Proof. The proof is by simultaneous induction on atomic $\mathcal{L}'$-programs and on $\mathcal{L}'$-programs. We will prove the existence of two functions $\mathfrak{q}^{at}$ and $\mathfrak{q}$ such that $\mathfrak{q}$ is an extension of $\mathfrak{q}^{at}$.

Suppose that $\mathfrak{p}$ is the atomic $\mathcal{L}'$-program $v \leftarrow t$, where $\{v\} \cup \mathrm{VAR}(t) \subseteq S$. Then, the program $\mathfrak{q}^{at}(\mathfrak{p}, S)$ is $\mathfrak{p}_{t,S,v'}; v \leftarrow v'$, where $\mathfrak{p}_{t,S,v'}$ is the $\mathcal{L}$-program introduced in Theorem 6.4.12 and v' is a variable not in S. Note that $\mathcal{S}^{pr}_{\mathcal{A}'}(\mathfrak{p})(\sigma)$ is defined if and only if $\mathcal{S}^{pr}_{\mathcal{A}}(\mathfrak{q}^{at}(\mathfrak{p}, S))(\tau)$ is defined because those states always exist. Further, if $z \in S$, then, because σ and τ agree on S and therefore on $\mathrm{VAR}(t)$, the value of z in both these states is $\sigma^{\mathcal{A}'}(t)$ if $z = v$. If $z \neq v$, then the value is $\sigma(z)$.

For the first inductive step, suppose that

$$\mathfrak{p} = \textbf{ while } \beta \textbf{ do } \mathfrak{p}_0 \textbf{ endwhile},$$

where the statement holds for $\mathfrak{q}(\mathfrak{p}_0, S)$. Define $\mathfrak{q}^{at}(\mathfrak{p}, S)$ as

$$\mathfrak{p}_{\beta,S}; \textbf{ while } \gamma(\beta, S) \textbf{ do } \mathfrak{q}(\mathfrak{p}_0, S); \mathfrak{p}_{\beta,S} \textbf{ endwhile}.$$

In the above program, $\mathfrak{p}_{\beta,S}$ and $\gamma(\beta, S)$ are the program and the formula defined in Theorem 6.4.13. Assume that $\mathcal{S}^{at\text{-}pr}_{\mathcal{A}'}(\mathfrak{p})(\sigma)$ is defined, that is there exists $k \in \mathbf{N}$ such that $\sigma_j = (\mathcal{S}^{pr}_{\mathcal{A}'}(\mathfrak{p}_0))^{(j)}(\sigma)$ exists for $j \leq k$, $(\mathcal{A}', \sigma_j) \models \beta$ for $j < k$, and $(\mathcal{A}', \sigma_k) \not\models \beta$. Notice that the state $\tau_0 = \mathcal{S}^{pr}_{\mathcal{A}}(\mathfrak{p}_{\beta,S})(\tau)$ exists by Theorem 6.4.13.

We prove by induction on j that for every $j \leq k$,

$$\tau_j = (\mathcal{S}^{pr}_{\mathcal{A}}(\mathfrak{q}(\mathfrak{p}_0, S); \mathfrak{p}_{\beta,S}))^{(j)}(\tau_0)$$

is defined, $\tau_j(x) = \sigma_j(x)$ for all $x \in S$, and $(\mathcal{A}', \sigma_j) \models \beta$ if and only if $(\mathcal{A}, \tau_j) \models \gamma(\beta, S)$. The basis step, $j = 0$, follows immediately from Theorem 6.4.13. Suppose that the claim holds for $j < k$. The state $\xi_j = \mathcal{S}^{pr}_{\mathcal{A}}(\mathfrak{q}(\mathfrak{p}_0, S))(\tau_j)$ is defined and $\xi_j(x) = \sigma_{j+1}(x)$ for all $x \in S$ by inductive hypothesis, because $\sigma_j(x) = \tau_j(x)$ for $x \in S$ and σ_{j+1} is defined. Further, $\tau_{j+1} = \mathcal{S}^{pr}_{\mathcal{A}}(\mathfrak{p}_{\beta,S})(\xi_j)$ is defined and $\tau_{j+1}(x) = \xi_j(x) = \sigma_{j+1}(x)$ for $x \in S$, by Theorem 6.4.13. Note that the following three statements are equivalent: (i) $(\mathcal{A}', \sigma_{j+1}) \models \beta$, (ii) $(\mathcal{A}', \xi_j) \models \beta$, and (iii) $(\mathcal{A}, \tau_{j+1}) \models \gamma(\beta, S)$. The equivalence between (i) and (ii) follows form the fact that σ_{j+1} and ξ_j agree on the variables of S and these variables include $\mathtt{FV}(\beta)$, the variables of the quantifier-free formula β. The equivalence between the last two statements is a consequence of Theorem 6.4.13.

The previous argument shows that $\tau_k = \mathcal{S}^{pr}_{\mathcal{A}}(\mathfrak{q}^{at}(\mathfrak{p}, S))(\tau)$ and that $\tau_k(x) = \sigma_k(x) = \mathcal{S}^{pr}_{\mathcal{A}'}(\mathfrak{p})(\sigma)(x)$ for every $x \in S$.

The converse statement, namely that the definedness of $\mathcal{S}^{pr}_{\mathcal{A}}(\mathfrak{q}^{at}(\mathfrak{p}, S))(\tau)$ implies that $\mathcal{S}^{pr}_{\mathcal{A}'}(\mathfrak{p})(\sigma)$ is defined and the two states agree on the variables of S has a similar proof which is left to the reader.

For the second inductive step, let $\mathfrak{p} =$ **if** β **then** $\mathfrak{p}_0$ **else** $\mathfrak{p}_1$ **endif**, where the inductive hypothesis holds for $\mathfrak{p}_0$ and $\mathfrak{p}_1$. Define $\mathfrak{q}^{at}(\mathfrak{p}, S)$ as

$$\mathfrak{p}_{\beta,S};\ \textbf{if}\ \gamma(\beta, S)\ \textbf{then}\ \mathfrak{q}(\mathfrak{p}_0, S)\ \textbf{else}\ \mathfrak{q}(\mathfrak{p}_1, S)\ \textbf{endif}.$$

Suppose that $(\mathcal{A}', \sigma) \models \beta$. Then, $\mathcal{S}^{at\text{-}pr}_{\mathcal{A}'}(\mathfrak{p})(\sigma) = \mathcal{S}^{pr}_{\mathcal{A}'}(\mathfrak{p}_0)(\sigma)$. Let $\xi = \mathcal{S}^{pr}_{\mathcal{A}}(\mathfrak{p}_{\beta,S})(\tau)$. Since $\sigma(x) = \tau(x)$ for every $x \in S$ and the variables of β are included in S, it follows that $(\mathcal{A}', \tau) \models \beta$. By Theorem 6.4.13, we have $(\mathcal{A}, \xi) \models \gamma(\beta, S)$ and $\xi(x) = \sigma(x)$ for all $x \in S$. Further,

$$\mathcal{S}^{at\text{-}pr}_{\mathcal{A}}(\mathfrak{q}^{at}(\mathfrak{p}, S))(\tau) = \mathcal{S}^{pr}_{\mathcal{A}}(\mathfrak{q}(\mathfrak{p}_0, S))(\xi).$$

The inductive hypothesis applied to $\mathfrak{p}_0$ and the states σ and ξ gives the desired result. We leave to the reader the entirely similar case when $(\mathcal{A}', \sigma) \not\models \beta$.

For an atomic program $\mathfrak{p}$, we define $\mathfrak{q}(\mathfrak{p}, S) = \mathfrak{q}^{at}(\mathfrak{p}, S)$. Finally, let $\mathfrak{p} = \mathfrak{p}'; \mathfrak{p}_0$, where $\mathfrak{p}'$ is atomic. We leave to the reader to prove that the program $\mathfrak{q}(\mathfrak{p}, S)$ defined as $\mathfrak{q}^{at}(\mathfrak{p}', S); \mathfrak{q}(\mathfrak{p}_0, S)$ satisfies the conditions of the theorem. □

Theorem 6.4.15. *Let $\mathfrak{p}$ be an $\mathcal{L}'$-program and S be a finite set of variables that contains $PVAR(\mathfrak{p})$. There is an effective construction that yields an $\mathcal{L}$-program $\mathfrak{q}(\mathfrak{p}, S)$ such that for all $\sigma \in \mathrm{STATES}_{\mathcal{A}}$, $\mathcal{S}^{pr}_{\mathcal{A}'}(\mathfrak{p})(\sigma)$ is defined if and only if $\mathcal{S}^{pr}_{\mathcal{A}}(\mathfrak{q}(\mathfrak{p}, S))(\sigma)$ is defined and if both are defined, then these states agree on all the variables of S.*

Proof. The statement follows from Theorem 6.4.14 by taking $\tau = \sigma$. □

Theorem 6.4.16. *Let $\mathfrak{p}$ be an $\mathcal{L}'$-program. There is an effective construction yielding an $\mathcal{L}$-program $\mathfrak{q}(\mathfrak{p})$ such that for all $\sigma \in \mathrm{STATES}_{\mathcal{A}}$, $\mathcal{S}^{pr}_{\mathcal{A}'}(\mathfrak{p})(\sigma)$ is defined if and only if $\mathcal{S}^{pr}_{\mathcal{A}}(\mathfrak{q}(\mathfrak{p}))(\sigma)$ is defined and if both* *are defined, then these states agree on all the variables that occur in* $\mathfrak{p}$.

Proof. The statement can be obtained from Theorem 6.4.15 by choosing $\mathfrak{q}(\mathfrak{p}) = \mathfrak{q}(\mathfrak{p}, \mathrm{PVAR}(\mathfrak{p}))$. $\square$

6.5 Hoare Triples

Having established the syntax and semantics of a programming language (in our case $\mathtt{WHILE}_{\mathcal{L}}$), we now want to discuss formally program correctness. Before a program can be proven to be correct, we need to specify what the program is to accomplish. Intuitively, a specification is a statement called a *postcondition* of what should be true after the program halts. In many cases, we don't expect the postcondition to hold in every circumstance but just in case a certain statement called a *precondition* holds before the program is run. In general, we will use formulas of first-order logic to express pre- and postconditions because these formulas are well-understood yet are sufficiently expressive.

We consider two notions of correctness for programs. *Total correctness* in the context of a specification means that if the program is started in a state in which the precondition is true, then the program halts in a state in which the postcondition is true. *Partial correctness* means that if the program is started in a state in which the precondition is true and if the program halts, then the state in which it halts satisfies the postcondition. Total correctness corresponds to the intuitive meaning of correctness. Because a program that always goes into an infinite loop is partially correct with respect to any specification, it may seem that partial correctness is not a very helpful notion. However, a program is totally correct for a specification if and only if it is partially correct for the specification and it halts when started in any state that satisfies the precondition of the specification. Thus, total correctness can be shown by proving partial correctness and termination separately, and this approach has proven to be productive. We will see a natural formal system for showing partial correctness. Termination is usually shown using ad-hoc methods. Attempts to strengthen formal systems to prove total correctness yield less than satisfactory results.

To formalize the notion of specification, we need the following definition:

Definition 6.5.1. A formula φ is an *assertion* if $\mathtt{BV}(\varphi) \subseteq \text{SVAR}$. An $\mathcal{L}$*-assertion* is an assertion which is also an $\mathcal{L}$-formula.

The set of assertions is denoted by ASSERT, while the set of $\mathcal{L}$-assertions is denoted by $\text{ASSERT}_{\mathcal{L}}$. □

Starting from an $\mathcal{L}$-formula φ, we define an $\mathcal{L}$-assertion $\varphi_\star$ such that $\varphi_\star$ is a variant of φ. The definition of $\varphi_\star$ is done inductively on the formula φ as follows.

- If φ is an atomic formula, then $\varphi_\star = \varphi$.
- If $\varphi = (\neg\alpha)$, then $\varphi_\star = (\neg\alpha_\star)$.
- If $\varphi = (\alpha C \beta)$, then $\varphi_\star = (\alpha_\star C \beta_\star)$, where C is a binary connective symbol.
- When $\varphi = (Qx)\alpha$, then $\varphi_\star = (Qy)(\alpha_\star)_{x:=y}$, where y is the first specification variable that does not occur in $(Qx)\alpha_\star$.

The reader can easily verify that $\varphi_\star$ is a variant of φ and an $\mathcal{L}$-assertion.

There are two types of syntactic objects used to give specifications for programs. They are introduced next.

Definition 6.5.2. Let $\mathcal{L}$ be a first-order language. An $\mathcal{L}$*-partial correctness Hoare triple* is a string of symbols of the form $\mathcal{H} = \{\varphi\}\mathfrak{p}\{\psi\}$ and an $\mathcal{L}$*-total correctness Hoare triple* is a string of symbols of the form $\mathcal{H}' = [\varphi]\mathfrak{p}[\psi]$, where φ, ψ are $\mathcal{L}$-assertions and $\mathfrak{p}$ is a $\mathcal{L}$-program. We will also refer to $\mathcal{L}$-partial correctness Hoare triples simply as $\mathcal{L}$-Hoare triples or just Hoare triples when $\mathcal{L}$ is understood from the context.

The set of all $\mathcal{L}$-partial correctness Hoare triples is denoted by $\mathsf{HPT}_{\mathcal{L}}$; the set of all $\mathcal{L}$-total correctness Hoare triples is denoted by $\mathsf{HTT}_{\mathcal{L}}$. The special subset of $\mathsf{HPT}_{\mathcal{L}}$ that consists of all $\mathcal{L}$-partial correctness triples of the form $\{\mathsf{true}^{\mathcal{L}}_\star\}\mathfrak{p}\{\mathsf{false}^{\mathcal{L}}_\star\}$ will be denoted by $\mathsf{HPT}^{\uparrow}_{\mathcal{L}}$. Recall that $\mathsf{true}^{\mathcal{L}}$ and $\mathsf{false}^{\mathcal{L}}$ were defined prior to Theorem 4.3.15.

Symbols in $\mathcal{L}$ that do not occur in $\mathfrak{p}$ are called *specification symbols* for the triples $\mathcal{H}, \mathcal{H}'$.

A string of symbols that is an $\mathcal{L}$-partial ($\mathcal{L}$-total) correctness Hoare triple for some first-order language $\mathcal{L}$ is called a *partial correctness Hoare triple*, respectively a *total correctness Hoare triple.*

The set of all partial correctness Hoare triples will be denoted by HPT and the set of total correctness Hoare triples will be denoted by HTT. $\square$

The next definition reflects our intuitive descriptions of partial and total correctness.

Definition 6.5.3. Let $\mathcal{L}$ be a first-order language, $\mathcal{A}$ be an $\mathcal{L}$-structure and σ be an $\mathcal{A}$-state. The pair $(\mathcal{A}, \sigma)$ *satisfies the triple* $\{\varphi\}\mathfrak{p}\{\psi\} \in \mathit{HPT}_{\mathcal{L}}$ if one of the following cases occurs:

(1) $(\mathcal{A}, \sigma) \not\models \varphi$;
(2) $\mathcal{S}^{pr}_{\mathcal{A}}(\mathfrak{p})(\sigma)$ is undefined;
(3) $\sigma' = \mathcal{S}^{pr}_{\mathcal{A}}(\mathfrak{p})(\sigma)$ is defined and $(\mathcal{A}, \sigma') \models \psi$.

If this is the case, we write $(\mathcal{A}, \sigma) \models \{\varphi\}\mathfrak{p}\{\psi\}$.

The pair $(\mathcal{A}, \sigma)$ *satisfies the triple* $[\varphi]\mathfrak{p}[\psi] \in \mathit{HTT}_{\mathcal{L}}$ if one of the following cases occurs:

(1) $(\mathcal{A}, \sigma) \not\models \varphi$;
(2) $\sigma' = \mathcal{S}^{pr}_{\mathcal{A}}(\mathfrak{p})(\sigma)$ is defined and $(\mathcal{A}, \sigma') \models \psi$.

If this is the case, we write $(\mathcal{A}, \sigma) \models [\varphi]\mathfrak{p}[\psi]$.

The $\mathcal{L}$-structure $\mathcal{A}$ is a *model* of the $\mathcal{L}$-partial correctness Hoare triple $\mathcal{H} = \{\varphi\}\mathfrak{p}\{\psi\}$ if $(\mathcal{A}, \sigma) \models \mathcal{H}$ for all $\mathcal{A}$-states σ; we also say in this case that $\mathcal{H}$ is *valid* in $\mathcal{A}$. $\mathcal{H}$ is *valid* if it is valid in every $\mathcal{L}$-structure.

$\mathcal{A}$ is a *model* of the $\mathcal{L}$-total correctness Hoare triple $\mathcal{H}' = [\varphi]\mathfrak{p}[\psi]$ if $(\mathcal{A}, \sigma) \models \mathcal{H}'$ for all $\mathcal{A}$-states σ; we also say in this case that $\mathcal{H}'$ is *valid* in $\mathcal{A}$. As above, $\mathcal{H}'$ is *valid* if it is valid in every $\mathcal{L}$-structure.

If $\mathcal{A}$ is a model of $\mathcal{H}$, where $\mathcal{H}$ is a total or partial correctness triple, then we write $\mathcal{A} \models \mathcal{H}$. If $\mathcal{H}$ is valid, we write $\models \mathcal{H}$. $\square$

The previous discussion about the interplay between partial and total correctness is formalized in the next theorem.

Theorem 6.5.4. *Let $\mathcal{L}$ be a first-order language, $\mathcal{A}$ be an $\mathcal{L}$-structure, φ, ψ be $\mathcal{L}$-assertions, and $\mathfrak{p}$ be an $\mathcal{L}$-program. Then, $\mathcal{A} \models [\varphi]\mathfrak{p}[\psi]$ if and only if $\mathcal{A} \models \{\varphi\}\mathfrak{p}\{\psi\}$ and $\mathcal{S}^{pr}_{\mathcal{A}}(\mathfrak{p})(\sigma)$ is defined for all states σ such that $(\mathcal{A}, \sigma) \models \varphi$.*

Proof. This follows immediately from Definition 6.5.3. □

Theorem 6.5.5. *Let $\mathcal{L}$ be a first-order language, $\mathcal{A}$ be an $\mathcal{L}$-structure, $\varphi, \varphi', \psi, \psi'$ be $\mathcal{L}$-assertions, and $\mathfrak{p}$ be an $\mathcal{L}$-program such that $\mathcal{A} \models (\varphi' \to \varphi)$ and $\mathcal{A} \models (\psi \to \psi')$. If $\mathcal{A} \models \{\varphi\}\mathfrak{p}\{\psi\}$ ($\mathcal{A} \models [\varphi]\mathfrak{p}[\psi]$), then $\mathcal{A} \models \{\varphi'\}\mathfrak{p}\{\psi'\}$ ($\mathcal{A} \models [\varphi']\mathfrak{p}[\psi']$).*

Proof. This statement follows directly from Definition 6.5.3. □

Example 6.5.6. The program $\mathfrak{p}$ introduced in Example 6.2.11 is intended to swap the values of its variables x and y when run in the $\mathcal{L}$-structure $\mathcal{A}$ considered in Example 6.3.12. To formulate the specification that the program is intended to meet, we need to refer in the postcondition to the initial values assumed by the variables x and y. Thus, we will consider two specification variables $x_\star$ and $y_\star$ which would allow us to state the precondition φ and the postcondition ψ as the $\mathcal{L}$-formulas

$$\varphi = ((x = x_\star) \wedge (y = y_\star))$$
$$\psi = ((x = y_\star) \wedge (y = x_\star)).$$

By Corollary 6.3.6, if $\sigma' = \mathcal{S}^{pr}_{\mathcal{A}}(\mathfrak{p})(\sigma)$ and $(\mathcal{A}, \sigma) \models \varphi$, then $\sigma'(x_\star) = \sigma(x_\star) = \sigma(x)$ and, similarly, $\sigma'(y_\star) = \sigma(y_\star) = \sigma(y)$, so in the postcondition, $x_\star$ and $y_\star$ denote the initial values of x and y.

Thus, the total correctness Hoare triple $\mathcal{H} = [\varphi]\mathfrak{p}[\psi]$ expresses our intentions for the program $\mathfrak{p}$.

Example 6.3.12 establishes that for every state $\sigma \in \mathrm{STATES}_{\mathcal{A}}$ such that $(\mathcal{A}, \sigma) \models \varphi$, $\sigma' = \mathcal{S}^{pr}_{\mathcal{A}}(\mathfrak{p})(\sigma)$ is defined and that $\sigma'(x) = \sigma(y) = \sigma'(y_\star)$ and $\sigma'(y) = \sigma(x) = \sigma'(x_\star)$, so $\mathcal{A} \models \mathcal{H}$. ▯

Example 6.5.7. Let $\mathcal{L}'$ be the first-order language defined by $\mathcal{L}' = \mathcal{L} \cup \{\mathrm{gcd}\}$, where $\mathcal{L}$ is the language introduced in Example 6.2.12 and gcd is a binary function symbol. We denote the formula $((t > u) \vee (t = u))$ by $t \geq u$, for all $t, u \in \mathrm{TERM}_{\mathcal{L}'}$. Consider the $\mathcal{L}'$-structure $\mathcal{A}'$ obtained by extending the structure $\mathcal{A}$ of Example 6.3.13 with the definition

$$\mathrm{gcd}^{\mathcal{A}'}(m, n) = \begin{cases} \text{the greatest common divisor} & \text{if } m \neq 0 \text{ or } n \neq 0 \\ \quad \text{of } m \text{ and } n & \\ 0 & \text{otherwise.} \end{cases}$$

The specification symbol gcd was added to the language $\mathcal{L}$ in order to allow us to specify the intended purpose of the program $\mathfrak{p}$ introduced in Example 6.2.12, namely, to compute the greatest common divisor of two natural numbers. Of course, in the wider language $\mathcal{L}'$, we could write $z \leftarrow \text{gcd}(x, y)$ as an $\mathcal{L}'$-program $\mathfrak{p}'$ which will compute the function gcd. However, such a program would not tell us anything about an actual algorithm for computing the greatest common divisor of two numbers, and thus it would be uninformative.

The total correctness Hoare triple

$$\mathcal{H} = [((x \geq 0) \wedge (y \geq 0) \wedge (x = x_\star) \wedge (y = y_\star))]\mathfrak{p}[z = \text{gcd}(x_\star, y_\star)]$$

expresses our specification of the behavior of the program: when $\mathfrak{p}$ is started with nonnegative values in the variables x, y, then it halts and produces in the variable z the greatest common divisor of its two initial arguments.

To show that $\mathcal{A}' \models \mathcal{H}$, by Theorem 6.5.4, it suffices to show that $\mathcal{A}' \models \mathcal{H}_0$, where

$$\mathcal{H}_0 = \{((x \geq 0) \wedge (y \geq 0) \wedge (x = x_\star) \wedge (y = y_\star))\}\mathfrak{p}\{z = \text{gcd}(x_\star, y_\star)\}$$

and that $\mathcal{S}^{pr}_{\mathcal{A}}(\mathfrak{p})(\sigma)$ is defined for all states σ such that $\sigma(x) \geq 0, \sigma(y) \geq 0, \sigma(x) = \sigma(x_\star)$ and $\sigma(y) = \sigma(y_\star)$.

Towards showing $\mathcal{A}' \models \mathcal{H}_0$, we note that for any two integers a, b, $\text{gcd}^{\mathcal{A}'}(a, b) = \text{gcd}^{\mathcal{A}'}(a - b, b) = \text{gcd}^{\mathcal{A}'}(a, b - a)$, and that if $a, b \geq 0$, $\text{gcd}^{\mathcal{A}'}(a, 0) = a$ and $\text{gcd}^{\mathcal{A}'}(0, b) = b$.

Recall that in Example 6.3.13 we wrote the program $\mathfrak{p}$ as $\mathfrak{p}_0; \mathfrak{p}_1$. Let $\mathfrak{q}$ be the $\mathcal{L}$-program defined by

$$\mathfrak{p}_0 = \textbf{ while } (x \neq 0 \wedge y \neq 0) \textbf{ do } \mathfrak{q} \textbf{ endwhile}.$$

In other words, $\mathfrak{q} = \textbf{ if } x > y \textbf{ then } x \leftarrow x - y \textbf{ else } y \leftarrow y - x \textbf{ endif}$. Note that for any state τ, $\tau' = \mathcal{S}^{pr}_{\mathcal{A}'}(\mathfrak{q})(\tau)$ is defined and by the previously mentioned properties of the greatest common divisor, $\text{gcd}^{\mathcal{A}'}(\tau(x), \tau(y)) = \text{gcd}^{\mathcal{A}'}(\tau'(x), \tau'(y))$. Furthermore, if $\tau(x), \tau(y) \geq 0$, then $\tau'(x), \tau'(y) \geq 0$.

Consider a state σ such that $(\mathcal{A}', \sigma) \models (x \geq 0 \wedge y \geq 0 \wedge x = x_\star \wedge y = y_\star)$. For $j \in \mathbf{N}$, let $\sigma_j = \left(\mathcal{S}^{pr}_{\mathcal{A}'}(\mathfrak{q})\right)^{(j)}(\sigma)$. An argument by induction on j (left to the reader) shows that for all $j \in \mathbf{N}$, $\text{gcd}^{\mathcal{A}'}(\sigma_j(x), \sigma_j(y)) = \text{gcd}^{\mathcal{A}'}(\sigma(x_\star), \sigma(y_\star))$ and that $\sigma_j(x), \sigma_j(y) \geq 0$.

To show that $(\mathcal{A}', \sigma) \models \mathcal{H}_0$, suppose that the state $\rho = \mathcal{S}^{pr}_{\mathcal{A}'}\textbf{(}\mathfrak{p}\textbf{)}(\sigma)$ is defined. Then, ρ can be written as $\mathcal{S}^{pr}_{\mathcal{A}'}\textbf{(}\mathfrak{p}_1\textbf{)}(\sigma')$, where $\sigma' = \mathcal{S}^{pr}_{\mathcal{A}'}\textbf{(}\mathfrak{p}_0\textbf{)}(\sigma)$. Since σ' is defined, $\sigma' = \sigma_k$, where k is the least number j such that

$$(\mathcal{A}', \sigma_j) \not\models (x \neq 0 \land y \neq 0).$$

This is equivalent to saying that at least one of $\sigma'(x), \sigma'(y)$ is 0. Since they are both nonnegative numbers, $\gcd^{\mathcal{A}'}(\sigma'(x), \sigma'(y))$ is the larger of these numbers, that is, $\rho(z) = \gcd^{\mathcal{A}'}(\sigma'(x), \sigma'(y)) = \gcd^{\mathcal{A}'}(\sigma(x_\star), \sigma(y_\star))$. Note also that $\rho(x_\star) = \sigma(x_\star)$ and $\rho(y_\star) = \sigma(y_\star)$ by Corollary 6.3.6, so $(\mathcal{A}', \rho) \models z = \gcd(x_\star, y_\star)$, which shows that $\mathcal{A}' \models \mathcal{H}_0$.

To show termination, consider a state σ such that $(\mathcal{A}', \sigma) \models (x \geq 0 \land y \geq 0 \land x = x_\star \land y = y_\star)$ and let σ_j be defined as above, $\sigma_j = \left(\mathcal{S}^{pr}_{\mathcal{A}'}\textbf{(}\mathfrak{q}\textbf{)}\right)^{(j)}(\sigma)$ for $j \in \mathbf{N}$. We already know that $\sigma_j(x) + \sigma_j(y) \geq 0$ for all j. Further, if $(\mathcal{A}', \sigma_j) \models (x \neq 0 \land y \neq 0)$, then $\sigma_{j+1}(x) + \sigma_{j+1}(y) < \sigma_j(x) + \sigma_j(y)$, as the reader can easily verify. Suppose that $\mathcal{S}^{pr}_{\mathcal{A}'}\textbf{(}\mathfrak{p}_0\textbf{)}(\sigma)$ is undefined. This means that for all $j \in \mathbf{N}$, $(\mathcal{A}', \sigma_j) \models (x \neq 0 \land y \neq 0)$, and thus we have the infinite descending chain of numbers $\sigma_0(x) + \sigma_0(y) > \sigma_1(x) + \sigma_1(y) > \cdots$ each of which is nonnegative. Since this is impossible, $\sigma' = \mathcal{S}^{pr}_{\mathcal{A}'}\textbf{(}\mathfrak{p}_0\textbf{)}(\sigma)$ is defined. It follows that $\mathfrak{p}$ halts when started in state σ. $\square$

Example 6.5.8. Let $\mathfrak{p} = z \leftarrow 1; \mathfrak{p}_1$ be the $\mathcal{L}$-program, where $\mathcal{L}$ and $\mathfrak{p}$ were introduced in Example 6.2.13 and discussed further in Example 6.3.14. Define $\mathcal{L}' = \mathcal{L} \cup \{\exp\}$, where exp is a binary function symbol used as a specification symbol for the program $\mathfrak{p}$. We denote the term $\exp(u, v)$ by u^v, where u, v are two $\mathcal{L}'$-terms. Consider the extension $\mathcal{A}'$ of the structure $\mathcal{A}$ defined in Example 6.3.14 where $\exp^{\mathcal{A}'}$ is the exponentiation function on $\mathbf{N}$. (We assume here that $\exp^{\mathcal{A}'}(m, 0) = 1$ for $m \in \mathbf{N}$.)

The specification for the desired behavior of the program is the total correctness triple

$$\mathcal{H} = [(x = x_\star \land y = y_\star)]\mathfrak{p}[z = x_\star^{y_\star}].$$

We begin the argument that $\mathcal{A}' \models \mathcal{H}$ by showing that $\mathcal{A}' \models \mathcal{H}_0$, where

$$\mathcal{H}_0 = \{(x = x_\star \land y = y_\star)\}\mathfrak{p}\{z = x_\star^{y_\star}\}.$$

Let σ be an $\mathcal{A}'$-state such that $(\mathcal{A}', \sigma) \models (x = x_\star \wedge y = y_\star)$, that is, $\sigma(x) = \sigma(x_\star)$ and $\sigma(y) = \sigma(y_\star)$. Let $\mathfrak{p}'$ be, as in Example 6.3.14, the body of the **while** loop and let $\sigma_0 = [z \to 1]\sigma$ and $\sigma_k = (\mathcal{S}^{pr}_{\mathcal{A}'}(\mathfrak{p}'))^{(k)}(\sigma_0)$ for $k \geq 1$.

Let $\theta = (\exists \ell)(x = x_\star^{2^\ell} \wedge y = y_\star \operatorname{div} 2^\ell \wedge z = x_\star^{y_\star \bmod 2^\ell})$. We claim that if $(\mathcal{A}', \sigma) \models \theta$, then $\sigma' = \mathcal{S}^{pr}_{\mathcal{A}'}(\mathfrak{p}')(\sigma)$ is defined and $(\mathcal{A}', \sigma') \models \theta$. Since $(\mathcal{A}', \sigma) \models \theta$, there is an $r \in \mathbf{N}$, $\sigma(x) = m^{2^r}$, $\sigma(y) = \lfloor n/2^r \rfloor$ and $\sigma(z) = m^{(n - \lfloor n/2^r \rfloor \cdot 2^r)}$, where $\sigma(x_\star) = m$ and $\sigma(y_\star) = n$. We have

$$\sigma'(x) = (\sigma(x))^2$$

$$\sigma'(y) = \lfloor \sigma(y)/2 \rfloor$$

$$\sigma'(z) = \begin{cases} \sigma(z) \cdot \sigma(x) & \text{if } \sigma(y) \text{ is odd} \\ \sigma(z) & \text{if } \sigma(y) \text{ is even.} \end{cases}$$

This implies immediately $\sigma'(x) = m^{2^{r+1}}$. For $\sigma'(y)$ we have

$$\sigma'(y) = \lfloor \lfloor n/2^r \rfloor / 2 \rfloor = \lfloor n/2^{r+1} \rfloor$$

because of the elementary identity $\lfloor \lfloor p \rfloor / 2 \rfloor = \lfloor p/2 \rfloor$ which is left to the reader for verification. Finally, for $\sigma'(z)$ we can write

$$\sigma'(z) = \begin{cases} m^{n - \lfloor n/2^r \rfloor \cdot 2^r + 2^r} & \text{if } \lfloor n/2^r \rfloor \text{ is odd} \\ m^{n - \lfloor n/2^r \rfloor \cdot 2^r} & \text{if } \lfloor n/2^r \rfloor \text{ is even.} \end{cases}$$

When $\lfloor n/2^r \rfloor$ is odd, we have $\lfloor n/2^{r+1} \rfloor = \lfloor \lfloor n/2^r \rfloor / 2 \rfloor = \lfloor n/2^r \rfloor / 2 - 1/2$, so $n - \lfloor n/2^r \rfloor \cdot 2^r + 2^r = n - \lfloor n/2^{r+1} \rfloor \cdot 2^{r+1}$. In the alternative case, when $\lfloor n/2^r \rfloor$ is even, we have $\lfloor n/2^{r+1} \rfloor = \lfloor \lfloor n/2^r \rfloor / 2 \rfloor = \lfloor n/2^r \rfloor / 2$, which allows us to conclude that $n - \lfloor n/2^r \rfloor \cdot 2^r = n - \lfloor n/2^{r+1} \rfloor \cdot 2^{r+1}$. So, in either case $\sigma'(z) = m^{n - \lfloor n/2^{r+1} \rfloor \cdot 2^{r+1}}$. This establishes that $(\mathcal{A}', \sigma') \models \theta$.

An easy induction argument on k shows σ_k is defined and $(\mathcal{A}', \sigma_k) \models \theta$. The basis follows from the fact that z is distinct from x and y and that $(\mathcal{A}', \sigma) \models (x = x_\star \wedge y = y_\star)$. The inductive step follows from the previous claim.

Suppose that $\rho = \mathcal{S}^{pr}_{\mathcal{A}'}(\mathfrak{p})(\sigma)$ is defined. Then $\rho = \sigma_k$, where k is the least number such that $\sigma_k(y) = 0$. Since $(\mathcal{A}', \sigma_k) \models \theta$, $\sigma_k(y_\star) < 2^{\sigma_k(\ell)}$, so $\sigma_k(y_\star) \operatorname{mod}^{\mathcal{A}'} 2^{\sigma_k(\ell)} = \sigma_k(y_\star)$ which implies $(\mathcal{A}', \sigma_k) \models z = x_\star^{y_\star}$. This concludes the argument for partial correctness.

To prove termination, note that if $\sigma(y) > 0$, then $\mathcal{S}^{pr}_{\mathcal{A}'}(\mathfrak{p}')(\sigma)(y) < \sigma(y)$. Suppose $\mathfrak{p}$ does not terminate when started in a state $\sigma \in \mathrm{STATES}_{\mathcal{A}'}$. Using the previous notation σ_k, this implies $\sigma_k(y) \neq 0$, which is equivalent to $\sigma_k(y) > 0$, for $k \in \mathbf{N}$ since $|\mathcal{A}'| = \mathbf{N}$. By the previous observation, we thus have the infinite descending sequence of natural numbers $\sigma_0(y) > \sigma_1(y) > \cdots$ which is a contradiction. ∎

Theorem 6.5.9. *Let $\mathcal{L}$ be a first-order language, v be program variable, t be an $(\mathcal{L}, PVAR)$-term and φ be an $\mathcal{L}$-assertion. Then, $\models \{(\varphi)_{v:=t}\}v{\leftarrow}t\{\varphi\}$.*

Proof. Suppose that for an $\mathcal{L}$-structure $\mathcal{A}$ and $\sigma \in \mathrm{STATES}_{\mathcal{A}}$, we have $(\mathcal{A}, \sigma) \models (\varphi)_{v:=t}$. Since the program $v{\leftarrow}t$ always halts, we need to show that $(\mathcal{A}, \mathcal{S}^{pr}_{\mathcal{A}}(v{\leftarrow}t)(\sigma)) \models \varphi$. By Definition 6.3.2, this amounts to $(\mathcal{A}, [v \to \sigma^{\mathcal{A}}(t)]\sigma) \models \varphi$, which holds by Corollary 4.6.6 and the fact that t is substitutable for v in φ because all bound variables in φ are specification variables which do not appear in t. □

Theorem 6.5.10. *Let $\mathcal{L}$ be a first-order language, $\mathcal{A}$ be an $\mathcal{L}$-structure, $\mathfrak{p}_0, \mathfrak{p}_1$ be $\mathcal{L}$-programs, φ, ψ be $\mathcal{L}$-assertions and let β be a quantifier-free $(\mathcal{L}, PVAR)$-formula. Then, $\mathcal{A} \models \{\varphi\}$ **if** β **then** $\mathfrak{p}_0$ **else** $\mathfrak{p}_1$ **endif** $\{\psi\}$ if and only if $\mathcal{A} \models \{(\varphi \wedge \beta)\}\mathfrak{p}_0\{\psi\}$ and $\mathcal{A} \models \{(\varphi \wedge (\neg\beta))\}\mathfrak{p}_1\{\psi\}$.*

Proof. The proof is a direct application of Definitions 6.3.2 and 6.5.3 and is left to the reader. □

Theorem 6.5.11. *Let $\mathcal{L}$ be a first-order language, $\mathcal{A}$ be an $\mathcal{L}$-structure, $\mathfrak{p}$ be an $\mathcal{L}$-program, φ, ψ be $\mathcal{L}$-assertions, and let β be a quantifier-free $(\mathcal{L}, PVAR)$-formula. Then, $\mathcal{A} \models \{\varphi\}$ **while** β **do** $\mathfrak{p}$ **endwhile**$\{\psi\}$ if and only if the following two conditions are satisfied:*

(1) $\mathcal{A} \models ((\varphi \wedge (\neg\beta)) \to \psi)$;
(2) $\mathcal{A} \models \{(\varphi \wedge \beta)\}\mathfrak{p}$; ***while*** β ***do*** $\mathfrak{p}$ ***endwhile***$\{\psi\}$.

Proof. Let $\mathfrak{q}$ be the program **while** β **do** $\mathfrak{p}$ **endwhile** and suppose that $\mathcal{A} \models \{\varphi\}\mathfrak{q}\{\psi\}$. To prove the first statement, suppose that $(\mathcal{A}, \sigma) \models (\varphi \wedge (\neg\beta))$. Since $(\mathcal{A}, \sigma) \not\models \beta$, it follows that $\mathcal{S}^{pr}_{\mathcal{A}}(\mathfrak{q})(\sigma) = \sigma$ and therefore $(\mathcal{A}, \sigma) \models \psi$, because $\mathcal{A} \models \{\varphi\}\mathfrak{q}\{\psi\}$. This allows us to conclude that $\mathcal{A} \models ((\varphi \wedge (\neg\beta)) \to \psi)$.

To show the second part, we have to prove that if $(\mathcal{A}, \sigma) \models (\varphi \wedge \beta)$ and $\sigma' = \mathcal{S}^{pr}_{\mathcal{A}}(\mathfrak{p};\mathfrak{q})(\sigma)$ is defined, then $(\mathcal{A}, \sigma') \models \psi$. Let $\tau = \mathcal{S}^{pr}_{\mathcal{A}}(\mathfrak{p})(\sigma)$. By Theorem 6.3.3, there is a number $k \in \mathbf{N}$ such that $\sigma' = (\mathcal{S}^{pr}_{\mathcal{A}}(\mathfrak{p}))^{(k)}(\tau)$. Further, $(\mathcal{A}, (\mathcal{S}^{pr}_{\mathcal{A}}(\mathfrak{p}))^{(k)}(\tau)) \not\models \beta$ and $(\mathcal{A}, (\mathcal{S}^{pr}_{\mathcal{A}}(\mathfrak{p}))^{(j)}(\tau)) \models \beta$ for $j < k$. The definition of τ implies that $(\mathcal{A}, (\mathcal{S}^{pr}_{\mathcal{A}}(\mathfrak{p}))^{(k+1)}(\sigma)) \not\models \beta$ and $(\mathcal{A}, (\mathcal{S}^{pr}_{\mathcal{A}}(\mathfrak{p}))^{(j)}(\sigma)) \models \beta$ for $1 \leq j \leq k$. In addition, since $(\mathcal{A}, \sigma) \models \beta$, we have $(\mathcal{A}, (\mathcal{S}^{pr}_{\mathcal{A}}(\mathfrak{p}))^{(j)}(\sigma) \models \beta$ for $0 \leq j \leq k$. Thus, $\mathcal{S}^{pr}_{\mathcal{A}}(\mathfrak{q})(\sigma) = \sigma'$ and we may conclude that $(\mathcal{A}, \sigma') \models \psi$ because $(\mathcal{A}, \sigma) \models \varphi$ and $\mathcal{A} \models \{\varphi\}\mathfrak{q}\{\psi\}$.

The argument for the reverse implication is left to the reader. □

Definition 6.5.12. Let Γ be a set of $\mathcal{L}$-formulas and $\mathcal{H}$ be an $\mathcal{L}$-partial correctness Hoare triple or an $\mathcal{L}$-total correctness Hoare triple. We write $\Gamma \approx\!\!\!\models \mathcal{H}$ and say that Γ *entails* $\mathcal{H}$ if every model of Γ is also a model of $\mathcal{H}$. □

Total correctness depends not only of the theory of a structure, but also on the actual structure. In this sense, total correctness is not a first-order property of structures, as defined in Definition 4.13.14. We prove this formally in the next theorem.

Theorem 6.5.13. *There is a first-order language $\mathcal{L}$ and a triple $\mathcal{H} = [\varphi]\mathfrak{p}[\psi] \in HPT_{\mathcal{L}}$ such that the class of all $\mathcal{L}$-structures $\mathcal{A}$ that model $\mathcal{H}$ is not a first-order property.*

Proof. Let $\mathcal{H} = [\mathsf{true}^{\mathcal{L}_{ar}}_{\star}]\mathfrak{p}[\mathsf{true}^{\mathcal{L}_{ar}}_{\star}]$ be the $\mathcal{L}_{ar}$-total correctness Hoare triple, where $\mathfrak{p}$ is the $\mathcal{L}_{ar}$-program:

```
$x \leftarrow 0$;
while $(x \neq y)$
    do $x \leftarrow s(x)$
endwhile
```

By Exercise 172 of Chapter 4, it suffices to show that we have two $\mathcal{L}_{ar}$-structures $\mathcal{A}$ and $\mathcal{B}$ such that $\mathrm{Th}^{\mathcal{L}_{ar}}(\mathcal{A}) = \mathrm{Th}^{\mathcal{L}_{ar}}(\mathcal{B})$, $\mathcal{A} \models \mathcal{H}$ and $\mathcal{B} \not\models \mathcal{H}$. We claim that we can choose $\mathcal{A} = \mathcal{A}_{ar}$, the standard model of arithmetic, and $\mathcal{B}$ be a nonstandard model of arithmetic as shown to exist in Theorem 4.13.11. The fact that $\mathcal{A}_{ar} \models \mathcal{H}$ follows from the fact that a sufficient number of increments by 1 of 0 allows us to reach any natural number y. On the other hand, if the initial value of y is a nonstandard element of $\mathcal{B}$ (see Definition 4.13.13), then

y is unreachable from 0 which means that the program will never terminate. Thus $\mathcal{B} \not\models \mathcal{H}$. □

In contrast to the previous theorem, we will prove that partial correctness is a first-order property. To show this, we will need the concept of "weakest liberal precondition".

Definition 6.5.14. Let $\mathcal{L}$ be a first-order language, $\mathcal{A}$ be an $\mathcal{L}$-structure, $\mathfrak{p}$ be an $\mathcal{L}$-program and ψ be an $\mathcal{L}$-assertion. The *weakest liberal precondition set* is the set $\mathsf{WLP}_{\mathcal{A}}(\mathfrak{p}, \psi)$ that consists of the states σ in $\mathrm{STATES}_{\mathcal{A}}$ such that one of the following two conditions is satisfied:

(1) $\mathcal{S}^{pr}_{\mathcal{A}}(\mathfrak{p})(\sigma)$ is undefined;
(2) $\mathcal{S}^{pr}_{\mathcal{A}}(\mathfrak{p})(\sigma)$ is defined and $(\mathcal{A}, \mathcal{S}^{pr}_{\mathcal{A}}(\mathfrak{p})(\sigma)) \models \psi$.

Let n be a positive natural number. The *n-limited weakest liberal precondition* is the set $\mathsf{WLP}^{n}_{\mathcal{A}}(\mathfrak{p}, \psi)$ that consists of the states $\sigma \in \mathrm{STATES}_{\mathcal{A}}$ such that one of the following three conditions is satisfied:

(1) $\mathsf{time}^{pr}_{\mathcal{A}}(\mathfrak{p})(\sigma)$ is undefined (and therefore, $\mathcal{S}^{pr}_{\mathcal{A}}(\mathfrak{p})(\sigma)$ is undefined);
(2) $\mathsf{time}^{pr}_{\mathcal{A}}(\mathfrak{p})(\sigma) > n$;
(3) $\mathsf{time}^{pr}_{\mathcal{A}}(\mathfrak{p})(\sigma) \leq n$ and $(\mathcal{A}, \mathcal{S}^{pr}_{\mathcal{A}}(\mathfrak{p})(\sigma)) \models \psi$.

▯

Theorem 6.5.15. *Let $\mathcal{L}$ be a first-order language, $\mathcal{A}$ be an $\mathcal{L}$-structure, $\mathfrak{p}$ be an $\mathcal{L}$-program and ψ be an $\mathcal{L}$-assertion. We have*

$$\mathit{WLP}_{\mathcal{A}}(\mathfrak{p}, \psi) = \bigcap_{n \geq 1} \mathit{WLP}^{n}_{\mathcal{A}}(\mathfrak{p}, \psi).$$

Proof. We leave this argument to the reader. □

Definition 6.5.16. Let $\mathcal{L}$ be a first-order language and let $\mathcal{A}$ be an $\mathcal{L}$-structure. A set of $\mathcal{L}$-formulas Γ *expresses* a set S of $\mathcal{A}$-states if $S = \{\sigma \in \mathrm{STATES}_{\mathcal{A}} \mid (\mathcal{A}, \sigma) \models \varphi \text{ for every } \varphi \in \Gamma\}$.

An $\mathcal{L}$-formula φ expresses a set of $\mathcal{A}$-states S if $\{\varphi\}$ expresses S. ▯

Lemma 6.5.17. *Suppose that $\mathfrak{q}$ is an $\mathcal{L}$-program and for all $\mathcal{L}$-assertions φ and positive numbers $n \in \mathbf{N}$, we have a formula $\omega^{n}(\mathfrak{q}, \varphi)$ that expresses $\mathit{WLP}^{n}_{\mathcal{A}}(\mathfrak{q}, \varphi)$ in every $\mathcal{L}$-structure $\mathcal{A}$.*

For $l \geq 2$ and $n_0, \ldots, n_{l-1}$ positive numbers, define $\omega(\mathbf{q}, \varphi)n_0, \ldots,$ n_{l-1} recursively as

$$\omega^{n_0,\ldots,n_{l-2}}(\mathfrak{q}, \omega^{n_{l-1}}(\mathfrak{q}, \varphi)).$$

Then, for every $\mathcal{L}$-structure $\mathcal{A}$, the formula $\omega^{n_0,\ldots,n_{l-1}}(\mathfrak{q}, \varphi)$ expresses the set of $\mathcal{A}$-states σ such that if $(\mathcal{S}^{pr}_{\mathcal{A}}(\mathfrak{q}))^{(l)}(\sigma)$ is defined and

$$\mathit{time}^{pr}_{\mathcal{A}}(\mathfrak{q})((\mathcal{S}^{pr}_{\mathcal{A}}(\mathfrak{q}))^{(i)}(\sigma)) \leq n_i, \quad \textit{for all } i, 0 \leq i \leq l-1,$$

then $(\mathcal{A}, (\mathcal{S}^{pr}_{\mathcal{A}}(\mathfrak{q}))^{(l)}(\sigma)) \models \varphi$.

Proof. The proof is by induction on the length of the sequence $n_0, \ldots, n_{l-1}$. The basis step, $l = 1$, follows immediately from the assumption of the lemma.

Suppose now that the statement is true for sequences of length l and consider the sequence $n_0, \ldots, n_l$. If $\psi = \omega^{n_l}(\mathfrak{q}, \varphi)$, then

$$\omega^{n_0,\ldots,n_l}(\mathfrak{q}, \varphi) = \omega^{n_0,\ldots,n_{l-1}}(\mathfrak{q}, \psi).$$

Thus, for an $\mathcal{L}$-structure $\mathcal{A}$ and a state $\sigma \in \mathrm{STATES}_{\mathcal{A}}$, we have the following three equivalent statements:

(1) $(\mathcal{A}, \sigma) \models \omega^{n_0,\ldots,n_l}(\mathfrak{q}, \varphi)$;
(2) $(\mathcal{A}, \sigma) \models \omega^{n_0,\ldots,n_{l-1}}(\mathfrak{q}, \psi)$;
(3) if $(\mathcal{S}^{pr}_{\mathcal{A}}(\mathfrak{q}))^{(l)}(\sigma)$ is defined and $\mathsf{time}^{pr}_{\mathcal{A}}(\mathfrak{q})((\mathcal{S}^{pr}_{\mathcal{A}}(\mathfrak{q}))^{(i)}(\sigma)) \leq n_i$, for all i, $0 \leq i \leq l-1$, then $(\mathcal{A}, (\mathcal{S}^{pr}_{\mathcal{A}}(\mathfrak{q}))^{(l)}(\sigma)) \models \psi$.

The equivalence between the second and the third statements follows from the inductive hypothesis.

By the assumption of the lemma, the third statement above is equivalent to the statement:

> if $(\mathcal{S}^{pr}_{\mathcal{A}}(\mathfrak{q}))^{(l)}(\sigma)$ is defined and $\mathsf{time}^{pr}_{\mathcal{A}}(\mathfrak{q})((\mathcal{S}^{pr}_{\mathcal{A}}(\mathfrak{q}))^{(i)}(\sigma)) \leq n_i$, for all i, $0 \leq i \leq l-1$, then $(\mathcal{S}^{pr}_{\mathcal{A}}(\mathfrak{q}))^{(l)}(\sigma) \in \mathsf{WLP}^{n_l}_{\mathcal{A}}(\mathfrak{q}, \varphi)$.

This in turn is equivalent to

> if $(\mathcal{S}^{pr}_{\mathcal{A}}(\mathfrak{q}))^{(l)}(\sigma)$ is defined and $\mathsf{time}^{pr}_{\mathcal{A}}(\mathfrak{q})((\mathcal{S}^{pr}_{\mathcal{A}}(\mathfrak{q}))^{(i)}(\sigma)) \leq n_i$, for all i, $0 \leq i \leq l-1$, then, if $\mathsf{time}^{pr}_{\mathcal{A}}(\mathfrak{q})((\mathcal{S}^{pr}_{\mathcal{A}}(\mathfrak{q}))^{(l)}(\sigma)) \leq n_l$, then $(\mathcal{A}, (\mathcal{S}^{pr}_{\mathcal{A}}(\mathfrak{q}))^{(l+1)}(\sigma)) \models \varphi$,

which is equivalent to the statement needed for the inductive step. □

Theorem 6.5.18. *There is an algorithm that, for each program $\mathfrak{p}$, assertion ψ, and positive number $n \in \mathbf{N}$, produces a formula $\omega^n(\mathfrak{p}, \psi)$ such that for every first-order language $\mathcal{L}$ and $\mathcal{L}$-structure $\mathcal{A}$, if $\mathfrak{p}$ is an $\mathcal{L}$-program and ψ is an $\mathcal{L}$-assertion, then $\omega^n(\mathfrak{p}, \psi)$ is an $\mathcal{L}$-formula that expresses $\mathsf{WLP}^n_{\mathcal{A}}(\mathfrak{p}, \psi)$.*

Proof. As usual, we define two functions $\omega^{n,at}(\mathfrak{p}, \psi)$ and $\omega^n(\mathfrak{r}, \psi)$; the first is applicable to atomic programs, while the second is applicable to all programs. The definition is by recursion on programs and it is applicable to all assertions ψ and positive natural numbers n.

If $\mathfrak{p}$ is the atomic program $v{\leftarrow}t$, we define $\omega^{n,at}(\mathfrak{p}, \psi) = (\psi)_{v:=t}$ for every $n \geq 1$.

If $\mathfrak{p}$ is the atomic program **if** β **then** $\mathfrak{q}$ **else** $\mathfrak{r}$ **endif**, then

$$\omega^{n,at}(\mathfrak{p}, \psi) = \begin{cases} ((\beta \to \omega^{n-1}(\mathfrak{q}, \psi)) \wedge ((\neg\beta) \to \omega^{n-1}(\mathfrak{r}, \psi)) & \text{if } n > 1, \\ \mathsf{true}_{\star}^{\mathcal{L}_\psi} & \text{if } n = 1. \end{cases}$$

For the last atomic case, suppose that $\mathfrak{p}$ is **while** β **do** $\mathfrak{q}$ **endwhile**. For $n \geq 1$, define the set of nonnull sequences of positive natural numbers D_n as

$$D_n = \left\{ (n_0, \ldots, n_{l-1}) \middle| n \geq l + 1 + \sum_{i=0}^{l-1} n_i \right\}.$$

Note that $D_1 = D_2 = \emptyset$, $D_3 = \{(1)\}$, $D_4 = D_3 \cup \{(2)\}$, $D_5 = D_4 \cup \{(3), (1,1)\}$.

For $(n_0, \ldots, n_{l-1}) \in D_n$, define the formula $\varepsilon^{n_0,\ldots,n_{l-1}}(\mathfrak{q}, \beta, \psi)$ as

$$\begin{aligned} &((\beta \wedge \omega^{n_0}(\mathfrak{q}, \beta) \wedge \omega^{n_0,n_1}(\mathfrak{q}, \beta) \wedge \\ &\cdots \wedge \omega^{n_0,\ldots,n_{l-2}}(\mathfrak{q}, \beta) \wedge \omega^{n_0,\ldots,n_{l-1}}(\mathfrak{q}, (\neg\beta))) \\ &\to \omega^{n_0,\ldots,n_{l-1}}(\mathfrak{q}, \psi)), \end{aligned}$$

where $\omega^{n_0,\ldots,n_{k-1}}(\mathfrak{q}, \varphi)$ is as introduced in Lemma 6.5.17.

The formula $\omega^{n,at}(\mathfrak{p},\psi)$ for $n \geq 3$ is

$$(((\neg\beta) \to \psi) \wedge \bigwedge\{\varepsilon^{n_0,\dots,n_{l-1}}(\mathfrak{q},\beta,\psi) \mid (n_0,\dots,n_{l-1}) \in D_n\}).$$

For $n \in \{1,2\}$, $\omega^{n,at}(\mathfrak{p},\psi)$ is $((\neg\beta) \longrightarrow \psi)$. For any atomic program $\mathfrak{p}$, $\omega^{n}(\mathfrak{p},\psi) = \omega^{n,at}(\mathfrak{p},\psi)$. If $\mathfrak{r} = \mathfrak{p}';\mathfrak{p}''$, where $\mathfrak{p}'$ is atomic, then

$$\omega^n(\mathfrak{r},\psi) = \begin{cases} \bigwedge\{\omega^{k'}(\mathfrak{p}',\omega^{k''}(\mathfrak{p}'',\psi)) \mid k',k'' \geq 1 \text{ and } k'+k'' \leq n\} & \text{if } n > 1 \\ \mathsf{true}_\star^{\mathcal{L}_\psi} & \text{if } n = 1. \end{cases}$$

Let $\mathcal{L}$ be a first-order language. A straightforward inductive argument shows that $\omega^n(\mathfrak{p},\psi)$ is an $\mathcal{L}$-formula for all $\mathcal{L}$-programs $\mathfrak{p}$ and $\mathcal{L}$-assertions ψ.

We now prove by simultaneous induction on the definition of atomic $\mathcal{L}$-programs $\mathfrak{p}$ and $\mathcal{L}$-programs $\mathfrak{r}$ that the formulas $\omega^{n,at}(\mathfrak{p},\psi)$ and $\omega^n(\mathfrak{r},\psi)$ express the sets of states $\mathsf{WLP}^n_{\mathcal{A}}(\mathfrak{p},\psi)$ and $\mathsf{WLP}^n_{\mathcal{A}}(\mathfrak{r},\psi)$, respectively, for all $\mathcal{L}$-structures $\mathcal{A}$, $\mathcal{L}$-assertions ψ and $n \geq 1$.

If $\mathfrak{p}$ is the atomic program $v \leftarrow t$, the running time of $\mathfrak{p}$ is 1 for any state. Thus we have the following four equivalent statements:

(1) $\sigma \in \mathsf{WLP}^n_{\mathcal{A}}(\mathfrak{p},\psi)$;
(2) $(\mathcal{A}, \mathcal{S}^{pr}_{\mathcal{A}}\mathbf{(}v \leftarrow t\mathbf{)}(\sigma)) \models \psi$;
(3) $(\mathcal{A}, [v \to \sigma^{\mathcal{A}}(t)]\sigma) \models \psi$;
(4) $(\mathcal{A},\sigma) \models (\psi)_{v:=t}$.

The equivalence of (1) and (2) follows from the definition of the weakest liberal precondition. The second statement is equivalent to (3) because of the definition of $\mathcal{S}^{pr}_{\mathcal{A}}$. Finally, the equivalence of (3) and (4) follows from the substitutability of t for v in the $\mathcal{L}$-assertion ψ and from Corollary 4.6.6.

Let now $\mathfrak{p}$ be the atomic program **if** β **then** $\mathfrak{q}$ **else** $\mathfrak{r}$ **endif** and assume that the statement holds for $\mathfrak{q}$ and $\mathfrak{r}$ for every n and ψ. We need to prove that $(\mathcal{A},\sigma) \models \omega^n(\mathfrak{p},\psi)$ if and only if $\mathsf{time}^{at\text{-}pr}_{\mathcal{A}}\mathbf{(}\mathfrak{p}\mathbf{)}(\sigma) \leq n$ implies $(\mathcal{A}, \mathcal{S}^{at\text{-}pr}_{\mathcal{A}}\mathbf{(}\mathfrak{p}\mathbf{)}(\sigma)) \models \psi$ for every $\mathcal{L}$-structure $\mathcal{A}$, $\mathcal{A}$-state σ, $\mathcal{L}$-assertion ψ and number $n \geq 1$.

Suppose that $(\mathcal{A},\sigma) \models \omega^n(\mathfrak{p},\psi)$ and $\mathsf{time}^{at\text{-}pr}_{\mathcal{A}}\mathbf{(}\mathfrak{p}\mathbf{)}(\sigma) \leq n$. Observe that we must have $n > 1$ because the running time of $\mathfrak{p}$ cannot be 1. We first consider the case when $(\mathcal{A},\sigma) \models \beta$ and, therefore, $(\mathcal{A},\sigma) \models \omega^{n-1}(\mathfrak{q},\psi)$. In addition, because $\mathsf{time}^{at\text{-}pr}_{\mathcal{A}}\mathbf{(}\mathfrak{p}\mathbf{)}(\sigma) \leq n$, we have $\mathsf{time}^{at\text{-}pr}_{\mathcal{A}}\mathbf{(}\mathfrak{q}\mathbf{)}(\sigma) \leq n-1$. By the inductive hypothesis, we have

$(\mathcal{A}, \mathcal{S}^{pr}_{\mathcal{A}}(\mathfrak{q})(\sigma)) \models \psi$ and, in this case, since $\mathcal{S}^{pr}_{\mathcal{A}}(\mathfrak{q})(\sigma) = \mathcal{S}^{at\text{-}pr}_{\mathcal{A}}(\mathfrak{p})(\sigma)$, we obtain the desired conclusion. The case when $(\mathcal{A}, \sigma) \models (\neg\beta)$ is similar.

Conversely, suppose that $\mathsf{time}^{at\text{-}pr}_{\mathcal{A}}(\mathfrak{p})(\sigma) \leq n$ implies $(\mathcal{A}, \mathcal{S}^{at\text{-}pr}_{\mathcal{A}}(\mathfrak{p})(\sigma)) \models \psi$. We have to show that $(\mathcal{A}, \sigma) \models \omega^n(\mathfrak{p}, \psi)$. The case when $n = 1$ is immediate. Therefore, suppose that $n > 1$. Suppose that $(\mathcal{A}, \sigma) \models \beta$. The conclusion follows if we can show $(\mathcal{A}, \sigma) \models \omega^{n-1}(\mathfrak{q}, \psi)$. By the inductive hypothesis, this amounts to showing that $\mathsf{time}^{pr}_{\mathcal{A}}(\mathfrak{q})(\sigma) \leq n-1$ implies $(\mathcal{A}, \mathcal{S}^{pr}_{\mathcal{A}}(\mathfrak{q})(\sigma)) \models \psi$. Assume, therefore, that $\mathsf{time}^{pr}_{\mathcal{A}}(\mathfrak{q})(\sigma) \leq n-1$, which means that $\mathsf{time}^{at\text{-}pr}_{\mathcal{A}}(\mathfrak{p})(\sigma) \leq n$. By the initial supposition of this paragraph, $(\mathcal{A}, \mathcal{S}^{at\text{-}pr}_{\mathcal{A}}(\mathfrak{p})(\sigma)) \models \psi$, which gives the desired conclusion because $\mathcal{S}^{at\text{-}pr}_{\mathcal{A}}(\mathfrak{p})(\sigma) = \mathcal{S}^{pr}_{\mathcal{A}}(\mathfrak{q})(\sigma)$. The alternative, $(\mathcal{A}, \sigma) \models (\neg\beta)$ is similar.

Suppose now that $\mathfrak{p}$ is the atomic program **while** β **do** $\mathfrak{q}$ **endwhile** and that the statement holds for $\mathfrak{q}$ and every n and ψ. We prove first that if $(\mathcal{A}, \sigma) \models \omega^{n,at}(\mathfrak{p}, \psi)$ and $\mathsf{time}^{at\text{-}pr}_{\mathcal{A}}(\mathfrak{p})(\sigma) \leq n$, then $(\mathcal{A}, \mathcal{S}^{at\text{-}pr}_{\mathcal{A}}(\mathfrak{p})(\sigma)) \models \psi$, for all $\mathcal{L}$-structures $\mathcal{A}$, $\mathcal{A}$-states σ, $n \geq 1$ and $\mathcal{L}$-assertions ψ.

Let l be the least number such that $(\mathcal{A}, (\mathcal{S}^{at\text{-}pr}_{\mathcal{A}}(\mathfrak{q}))^{(l)}(\sigma)) \models (\neg\beta)$.

If $l = 0$, this means that $(\mathcal{A}, \sigma) \models (\neg\beta)$ so $(\mathcal{A}, \sigma) \models \psi$ because of the definition of the formula $\omega^{n,at}(\mathfrak{p}, \psi)$. Since, in this case, $\sigma = \mathcal{S}^{at\text{-}pr}_{\mathcal{A}}(\mathfrak{p})(\sigma)$, we have the desired conclusion.

Let now $l > 0$ and for i such that $0 \leq i \leq l-1$, define

$$n_i = \mathsf{time}^{at\text{-}pr}_{\mathcal{A}}(\mathfrak{q})((\mathcal{S}^{at\text{-}pr}_{\mathcal{A}}(\mathfrak{q}))^{(i)}(\sigma)).$$

The execution time is therefore $l + 1 + \sum_{i=0}^{l-1} n_i = \mathsf{time}^{at\text{-}pr}_{\mathcal{A}}(\mathfrak{p})(\sigma) \leq n$, which implies that $(n_0, \ldots, n_{l-1}) \in D_n$. Consequently, by the definition of $\omega^{n,at}(\mathcal{A}, \psi)$, $(\mathcal{A}, \sigma) \models \varepsilon^{n_0,\ldots,n_{l-1}}(\mathfrak{q}, \beta, \psi)$. By the minimality of l, we have $(\mathcal{A}, (\mathcal{S}^{at\text{-}pr}_{\mathcal{A}}(\mathfrak{q}))^{(i)}(\sigma)) \models \beta$, for $0 \leq i \leq l-1$, so, by the inductive hypothesis and Lemma 6.5.17, the hypotheses of the implication $\varepsilon^{n_0,\ldots,n_{l-1}}(\mathfrak{q}, \beta, \psi)$ are satisfied by $(\mathcal{A}, \sigma)$, which means that $(\mathcal{A}, \sigma) \models \omega^{n_0,\ldots,n_{l-1}}(\mathfrak{q}, \psi)$. Again, by the inductive hypothesis and Lemma 6.5.17, we have $(\mathcal{A}, (\mathcal{S}^{pr}_{\mathcal{A}}(\mathfrak{q}))^{(l)}(\sigma)) \models \psi$. Since, in this case, $(\mathcal{S}^{pr}_{\mathcal{A}}(\mathfrak{q}))^{(l)}(\sigma) = \mathcal{S}^{at\text{-}pr}_{\mathcal{A}}(\mathfrak{p})(\sigma)$, we have the desired conclusion.

In the reverse direction, assume that

$$\mathsf{time}^{at\text{-}pr}_{\mathcal{A}}(\mathfrak{p})(\sigma) \leq n \text{ implies } (\mathcal{A}, \mathcal{S}^{at\text{-}pr}_{\mathcal{A}}(\mathfrak{p})(\sigma)) \models \psi$$

for an $\mathcal{L}$-structure $\mathcal{A}$, an $\mathcal{A}$-state σ, $n > 0$ and $\mathcal{L}$-assertion ψ. We need to prove that $(\mathcal{A}, \sigma) \models \omega^{n,at}(\mathfrak{p}, \psi)$.

We show first that $(\mathcal{A}, \sigma) \models ((\neg\beta) \to \psi)$. If $(\mathcal{A}, \sigma) \models (\neg\beta)$, then $\mathsf{time}_{\mathcal{A}}^{at\text{-}pr}\textbf{(}\mathfrak{p}\textbf{)}(\sigma) = 1 \leq n$ and the conclusion follows from the assumption and the fact that $\sigma = \mathcal{S}_{\mathcal{A}}^{at\text{-}pr}\textbf{(}\mathfrak{p}\textbf{)}(\sigma)$. This shows that $(\mathcal{A}, \sigma) \models \omega^{n,at}(\mathfrak{p}, \psi)$ if $n \in \{1, 2\}$.

For $n \geq 3$, let $(n_0, \ldots, n_{l-1}) \in D_n$. We need to show that

$$(\mathcal{A}, \sigma) \models \varepsilon^{n_0,\ldots,n_{l-1}}(\mathfrak{q}, \beta, \psi).$$

Taking into account the definition of $\varepsilon^{n_0,\ldots,n_{l-1}}(\mathfrak{q}, \beta, \psi)$, suppose that

$$\begin{aligned} &(\mathcal{A}, \sigma) \models \beta \qquad (6.3)\\ &(\mathcal{A}, \sigma) \models \omega^{n_0}(\mathfrak{q}, \beta)\\ &\qquad\vdots\\ &(\mathcal{A}, \sigma) \models \omega^{n_0,\ldots,n_{l-2}}(\mathfrak{q}, \beta)\\ &(\mathcal{A}, \sigma) \models \omega^{n_0,\ldots,n_{l-1}}(\mathfrak{q}, (\neg\beta)). \end{aligned}$$

We will be done if we show that $(\mathcal{A}, \sigma) \models \omega^{n_0,\ldots,n_{l-1}}(\mathfrak{q}, \psi)$. By the inductive hypothesis and Lemma 6.5.17, this amounts to proving that the set of inequalities

$$\mathsf{time}_{\mathcal{A}}^{at\text{-}pr}\textbf{(}\mathfrak{q}\textbf{)}((\mathcal{S}_{\mathcal{A}}^{at\text{-}pr}\textbf{(}\mathfrak{q}\textbf{)})^{(i)}(\sigma)) \leq n_i \qquad (6.4)$$

for $0 \leq i \leq l-1$ implies $(\mathcal{A}, (\mathcal{S}_{\mathcal{A}}^{at\text{-}pr}\textbf{(}\mathfrak{q}\textbf{)})^{(l)}(\sigma)) \models \psi$. By repeatedly using the inductive hypothesis, Lemma 6.5.17 and (6.3) and (6.4), it follows that for $0 \leq i \leq l-1$, we have $(\mathcal{A}, (\mathcal{S}_{\mathcal{A}}^{pr}\textbf{(}\mathfrak{q}\textbf{)})^{(i)}(\sigma)) \models \beta$ and $(\mathcal{A}, (\mathcal{S}_{\mathcal{A}}^{pr}\textbf{(}\mathfrak{q}\textbf{)})^{(l)}(\sigma)) \models (\neg\beta)$. This means that the running time $\mathsf{time}_{\mathcal{A}}^{at\text{-}pr}\textbf{(}\mathfrak{p}\textbf{)}(\sigma) \leq n$, so, by the hypothesis, we obtain $(\mathcal{A}, \mathcal{S}_{\mathcal{A}}^{at\text{-}pr}\textbf{(}\mathfrak{p}\textbf{)}(\sigma)) \models \psi$ which implies that

$$(\mathcal{A}, (\mathcal{S}_{\mathcal{A}}^{pr}\textbf{(}\mathfrak{q}\textbf{)})^{(l)}(\sigma)) \models \psi.$$

Let now $\mathfrak{r} = \mathfrak{p}'; \mathfrak{p}''$, where $\mathfrak{p}'$ is atomic and the statement holds for $\mathfrak{p}'$ and $\mathfrak{p}''$. Suppose that $(\mathcal{A}, \sigma) \models \omega^n(\mathfrak{r}, \psi)$ and $\mathsf{time}_{\mathcal{A}}^{pr}\textbf{(}\mathfrak{r}\textbf{)}(\sigma) \leq n$. We want to prove that $(\mathcal{A}, \mathcal{S}_{\mathcal{A}}^{pr}\textbf{(}\mathfrak{r}\textbf{)}(\sigma)) \models \psi$. Note that $n > 1$ because $\mathfrak{r}$ cannot run in fewer than two steps. There are k', k'' such that $\mathsf{time}_{\mathcal{A}}^{at\text{-}pr}\textbf{(}\mathfrak{p}'\textbf{)}(\sigma) = k'$, $\mathsf{time}_{\mathcal{A}}^{pr}\textbf{(}\mathfrak{p}''\textbf{)}(\mathcal{S}_{\mathcal{A}}^{at\text{-}pr}\textbf{(}\mathfrak{p}'\textbf{)}(\sigma)) = k''$, and $k' + k'' \leq n$.

By the definition of $\omega^n(\mathfrak{r},\psi)$, we have $(\mathcal{A},\sigma) \models \omega^{k'}(\mathfrak{p}',\omega^{k''}(\mathfrak{p}'',\psi))$. By the inductive hypothesis, applied to $\mathfrak{p}'$, we have $(\mathcal{A},\mathcal{S}^{at\text{-}pr}_{\mathcal{A}}(\mathfrak{p}')(\sigma)) \models \omega^{k''}(\mathfrak{p}'',\psi)$. A new application of the inductive hypothesis to $\mathfrak{p}''$ yields $(\mathcal{A},\mathcal{S}^{pr}_{\mathcal{A}}(\mathfrak{p}'')(\mathcal{S}^{at\text{-}pr}_{\mathcal{A}}(\mathfrak{p}')(\sigma))) \models \psi$. This gives the desired conclusion because $\mathcal{S}^{pr}_{\mathcal{A}}(\mathfrak{p}'')(\mathcal{S}^{at\text{-}pr}_{\mathcal{A}}(\mathfrak{p}')(\sigma)) = \mathcal{S}^{pr}_{\mathcal{A}}(\mathfrak{r})(\sigma)$.

Conversely, assume that $\mathsf{time}^{pr}_{\mathcal{A}}(\mathfrak{r})(\sigma) \leq n$ implies $(\mathcal{A},\mathcal{S}^{pr}_{\mathcal{A}}(\mathfrak{r})(\sigma)) \models \psi$. We must show that $(\mathcal{A},\sigma) \models \omega^n(\mathfrak{r},\beta)$.

The case when $n = 1$ is immediate, by the definition of $\omega^1(\mathfrak{r},\beta)$. Suppose therefore that $n > 1$ and fix $k', k'' \geq 1$ such that $k' + k'' \leq n$. We must show that $(\mathcal{A},\sigma) \models \omega^{k'}(\mathfrak{p}',\omega^{k''}(\mathfrak{p}'',\psi))$. By the inductive hypothesis, it suffices to assume that $\mathsf{time}^{at\text{-}pr}_{\mathcal{A}}(\mathfrak{p}')(\sigma) \leq k'$ and to show that $(\mathcal{A},\mathcal{S}^{at\text{-}pr}_{\mathcal{A}}(\mathfrak{p}')(\sigma)) \models \omega^{k''}(\mathfrak{p}'',\psi)$. In turn, by another application of the inductive hypothesis, it suffices to assume that $\mathsf{time}^{pr}_{\mathcal{A}}(\mathfrak{p}'')(\mathcal{S}^{at\text{-}pr}_{\mathcal{A}}(\mathfrak{p}')(\sigma)) \leq k''$ and to show that $(\mathcal{A},\mathcal{S}^{pr}_{\mathcal{A}}(\mathfrak{p}'')(\mathcal{S}^{at\text{-}pr}_{\mathcal{A}}(\mathfrak{p}')(\sigma))) \models \psi$. By the two previous temporal assumptions just made on $\mathfrak{p}'$ and $\mathfrak{p}''$, we have $\mathsf{time}^{pr}_{\mathcal{A}}(\mathfrak{r})(\sigma) \leq k'+k'' \leq n$, and so, by the initial assumption, we obtain $(\mathcal{A},\mathcal{S}^{pr}_{\mathcal{A}}(\mathfrak{r})(\sigma)) \models \psi$. □

Corollary 6.5.19. *Let $\mathcal{L}$ be a first-order language, $\mathfrak{p}$ be an $\mathcal{L}$-program and ψ be an $\mathcal{L}$-assertion. There is a set of $\mathcal{L}$-formulas $\Omega(\mathfrak{p},\psi)$ such that for all $\mathcal{L}$-structures $\mathcal{A}$ the set of formulas expresses $\mathit{WLP}_{\mathcal{A}}(\mathfrak{p},\psi)$.*

Proof. Define $\Omega(\mathfrak{p},\psi) = \{\omega^n(\mathfrak{p},\psi) \mid n \geq 1\}$. The statement follows from Theorems 6.5.15 and 6.5.18. □

Theorem 6.5.20. *There is an algorithm that, for each partial correctness Hoare triple $\mathcal{H}$, produces a set of sentences $\Sigma_{\mathcal{H}}$ such that for every first-order language $\mathcal{L}$ and $\mathcal{L}$-structure $\mathcal{A}$, if $\mathcal{H}$ is an $\mathcal{L}$-partial correctness Hoare triple, then $\Sigma_{\mathcal{H}}$ is a set of $\mathcal{L}$-sentences and we have $\mathcal{A} \models \mathcal{H}$ if and only if $\mathcal{A} \models \Sigma_{\mathcal{H}}$.*

Proof. Let $\mathcal{H} = \{\varphi\}\mathfrak{p}\{\psi\}$ be a partial correctness triple and let

$$\Sigma_{\mathcal{H}} = \{(\varphi \to \omega^n(\mathfrak{p},\psi))^{\forall} \mid n \geq 1\},$$

where $\omega^n(\mathfrak{p},\psi)$ is the formula defined in Theorem 6.5.18.

Suppose $\mathcal{L}$ is a first-order language, $\mathcal{H}$ is an $\mathcal{L}$-partial correctness triple, and $\mathcal{A}$ is an $\mathcal{L}$-structure. By Theorem 6.5.18, each of the formulas $\omega^n(\mathfrak{p},\psi)$ is an $\mathcal{L}$-formula, so $\Sigma_{\mathcal{H}}$ is a set of $\mathcal{L}$-sentences.

Assume that $\mathcal{A} \models \mathcal{H}$. To show that $\mathcal{A} \models \Sigma_{\mathcal{H}}$, it suffices to show, by Theorem 4.5.58, that $\mathcal{A} \models (\varphi \rightarrow \omega^n(\mathfrak{p}, \psi))$ for all $n \geq 1$. Suppose $(\mathcal{A}, \sigma) \models \varphi$. If $\mathsf{time}^{pr}_{\mathcal{A}}(\mathfrak{p})(\sigma) \leq n$, $\mathcal{S}^{pr}_{\mathcal{A}}(\mathfrak{p})(\sigma)$ is defined and, because $\mathcal{A} \models \mathcal{H}$, we have $(\mathcal{A}, \mathcal{S}^{pr}_{\mathcal{A}}(\mathfrak{p})(\sigma)) \models \psi$, which allows us to conclude $\mathcal{A} \models (\varphi \rightarrow \omega^n(\mathfrak{p}, \psi))$.

Now, suppose that $\mathcal{A} \models \Sigma_{\mathcal{H}}$. To show that $\mathcal{A} \models \mathcal{H}$, assume that $(\mathcal{A}, \sigma) \models \varphi$ and $\mathcal{S}^{pr}_{\mathcal{A}}(\mathfrak{p})(\sigma)$ is defined. Then, $\mathsf{time}^{pr}_{\mathcal{A}}(\mathfrak{p})(\sigma) = n$ for some $n \geq 1$. Since $\mathcal{A} \models (\varphi \rightarrow \omega^n(\mathfrak{p}, \psi))^{\forall}$, by Theorem 4.5.58, it follows that $\mathcal{A} \models (\varphi \rightarrow \omega^n(\mathfrak{p}, \psi))$, so $(\mathcal{A}, \sigma) \models \omega^n(\mathfrak{p}, \psi)$. Thus, $(\mathcal{A}, \mathcal{S}^{pr}_{\mathcal{A}}(\mathfrak{p})(\sigma)) \models \psi$ because the program $\mathfrak{p}$ runs in time n when started in σ. This yields the desired conclusion. $\square$

Theorem 6.5.20 shows that for every $\mathcal{L}$-partial correctness Hoare triple $\mathcal{H}$, the collection of $\mathcal{L}$-structures $\{\mathcal{A} \mid \mathcal{A} \models \mathcal{H}\}$ is a first-order property. Informally, we say that partial correctness is a first-order property.

The effectiveness of the construction of the formulas $\omega^n(\mathfrak{p}, \psi)$ proven in Theorem 6.5.18 shows that given a partial correctness Hoare triple $\mathcal{H}$, we can list effectively the set $\Sigma_{\mathcal{H}}$ whose existence was established in Theorem 6.5.20.

Corollary 6.5.21. *Let $\mathcal{A}$ be a $\mathcal{L}$-structure and let $\mathcal{H}$ be an $\mathcal{L}$-partial correctness Hoare triple. We have $\mathcal{A} \models \mathcal{H}$ if and only if $Th^{\mathcal{L}}(\mathcal{A}) \approx\!\!\!\models \mathcal{H}$.*

Proof. Suppose $\mathcal{A} \models \mathcal{H}$. Then, by Theorem 6.5.20, we have $\Sigma_{\mathcal{H}} \subseteq \mathrm{Th}^{\mathcal{L}}(\mathcal{A})$. Let $\mathcal{B}$ be a model of $\mathrm{Th}^{\mathcal{L}}(\mathcal{A})$. Then $\mathcal{B} \models \Sigma_{\mathcal{H}}$, so $\mathcal{B} \models \mathcal{H}$. Thus, $\mathrm{Th}^{\mathcal{L}}(\mathcal{A}) \approx\!\!\!\models \mathcal{H}$.

The converse implication follows immediately from the fact that $\mathcal{A}$ is a model of $\mathrm{Th}^{\mathcal{L}}(\mathcal{A})$. $\square$

We now embark on proving a series of results about the weakest liberal precondition some of which will be useful in obtaining a limited completeness result for a formal system that deals with partial correctness Hoare triples.

Theorem 6.5.22. *Let v be a program variable, t be an $(\mathcal{L}, PVAR)$-term and ψ be an $\mathcal{L}$-assertion. Then, the formula $(\psi)_{v:=t}$ expresses $\mathsf{WLP}_{\mathcal{A}}(v \leftarrow t, \psi)$ for every $\mathcal{L}$-structure $\mathcal{A}$.*

Proof. We have shown in the proof of Theorem 6.5.18 that the formula $(\psi)_{v:=t}$ expresses $\mathsf{WLP}^n_{\mathcal{A}}(v \leftarrow t, \psi)$ for every $n \geq 1$.

Since $\mathsf{WLP}^n_{\mathcal{A}}(v{\leftarrow}t, \psi) = \mathsf{WLP}_{\mathcal{A}}(v{\leftarrow}t, \psi)$ for every $n \geq 1$, the statement of the theorem follows immediately. $\square$

Theorem 6.5.23. *Let β be a quantifier-free $(\mathcal{L}, PVAR)$-formula, $\mathcal{A}$ be an $\mathcal{L}$-structure, ψ be an $\mathcal{L}$-assertion and $\mathfrak{p}_0, \mathfrak{p}_1$ be two $\mathcal{L}$-programs.*

If γ expresses $\mathsf{WLP}_{\mathcal{A}}(\mathfrak{p}_0, \psi)$ and δ expresses $\mathsf{WLP}_{\mathcal{A}}(\mathfrak{p}_1, \psi)$, then $\theta = ((\beta \rightarrow \gamma) \wedge ((\neg\beta) \rightarrow \delta))$ expresses $\mathsf{WLP}_{\mathcal{A}}(\mathfrak{p}, \psi)$, where

$$\mathfrak{p} = \textbf{ if } \beta \textbf{ then } \mathfrak{p}_0 \textbf{ else } \mathfrak{p}_1 \textbf{ endif}.$$

Proof. We need to prove that $(\mathcal{A}, \sigma) \models \theta$ if and only if either $\mathcal{S}^{pr}_{\mathcal{A}}(\mathfrak{p})(\sigma)$ is undefined, or $\sigma' = \mathcal{S}^{pr}_{\mathcal{A}}(\mathfrak{p})(\sigma)$ is defined and $(\mathcal{A}, \sigma') \models \psi$.

Suppose that $(\mathcal{A}, \sigma) \models \theta$ and that $\sigma' = \mathcal{S}^{pr}_{\mathcal{A}}(\mathfrak{p})(\sigma)$ is defined. We have both that $(\mathcal{A}, \sigma) \models (\beta \rightarrow \gamma)$ and $(\mathcal{A}, \sigma) \models ((\neg\beta) \rightarrow \delta)$. If $(\mathcal{A}, \sigma) \models \beta$, it follows that $(\mathcal{A}, \sigma) \models \gamma$ and $\mathcal{S}^{pr}_{\mathcal{A}}(\mathfrak{p}_0)(\sigma)$ is defined and equals σ'. This implies that $(\mathcal{A}, \sigma') \models \psi$ because γ expresses $\mathsf{WLP}_{\mathcal{A}}(\mathfrak{p}_0, \psi)$. If $(\mathcal{A}, \sigma) \not\models \beta$, then $(\mathcal{A}, \sigma) \models (\neg\beta)$ and a similar argument applies.

To prove the converse implication, suppose that either $\mathcal{S}^{pr}_{\mathcal{A}}(\mathfrak{p})(\sigma)$ is undefined, or $\sigma' = \mathcal{S}^{pr}_{\mathcal{A}}(\mathfrak{p})(\sigma)$ is defined and $(\mathcal{A}, \sigma') \models \psi$.

If $(\mathcal{A}, \sigma) \models \beta$, we have either $\mathcal{S}^{pr}_{\mathcal{A}}(\mathfrak{p}_0)(\sigma)$ is undefined, or $\sigma' = \mathcal{S}^{pr}_{\mathcal{A}}(\mathfrak{p}_0)(\sigma)$ is defined and $(\mathcal{A}, \sigma') \models \psi$. Then, $\sigma \in \mathsf{WLP}_{\mathcal{A}}(\mathfrak{p}_0, \psi)$, so $(\mathcal{A}, \sigma) \models \gamma$. Thus, $(\mathcal{A}, \sigma) \models (\beta \rightarrow \gamma)$. Further, $(\mathcal{A}, \sigma) \models ((\neg\beta) \rightarrow \delta)$, because $(\mathcal{A}, \sigma) \models \beta$. Thus, $(\mathcal{A}, \sigma) \models \theta$.

The case when $(\mathcal{A}, \sigma) \models (\neg\beta)$ is similar and is left to the reader. $\square$

Theorem 6.5.24. *Let $\mathcal{L}$ be a first-order language, $\mathcal{A}$ be an $\mathcal{L}$-structure, $\alpha, \gamma, \varphi, \psi$ be $\mathcal{L}$-assertions and let $\mathfrak{p}_0, \mathfrak{p}_1$ be $\mathcal{L}$-programs. Suppose that γ expresses $\mathsf{WLP}_{\mathcal{A}}(\mathfrak{p}_1, \psi)$. Then, we have:*

(1) *$\mathcal{A} \models \{\varphi\}\mathfrak{p}_0; \mathfrak{p}_1\{\psi\}$ if and only if $\mathcal{A} \models \{\varphi\}\mathfrak{p}_0\{\gamma\}$.*
(2) *If α expresses $\mathsf{WLP}_{\mathcal{A}}(\mathfrak{p}_0, \gamma)$, then α expresses $\mathsf{WLP}_{\mathcal{A}}(\mathfrak{p}_0; \mathfrak{p}_1, \psi)$.*

Proof. To prove the first part, we first suppose that

$$\mathcal{A} \models \{\varphi\}\mathfrak{p}_0; \mathfrak{p}_1\{\psi\} \tag{6.5}$$

and show that $\mathcal{A} \models \{\varphi\}\mathfrak{p}_0\{\gamma\}$. To this end, suppose that $(\mathcal{A}, \sigma) \models \varphi$ and the state $\sigma' = \mathcal{S}^{pr}_{\mathcal{A}}(\mathfrak{p}_0)(\sigma)$ is defined. We consider two cases. If $\mathcal{S}^{pr}_{\mathcal{A}}(\mathfrak{p}_1)(\sigma')$ is not defined, then $\sigma' \in \mathsf{WLP}_{\mathcal{A}}(\mathfrak{p}_1, \psi)$, so $(\mathcal{A}, \sigma') \models \gamma$. Suppose now that $\sigma'' = \mathcal{S}^{pr}_{\mathcal{A}}(\mathfrak{p}_1)(\sigma') = \mathcal{S}^{pr}_{\mathcal{A}}(\mathfrak{p}_0; \mathfrak{p}_1)(\sigma)$

is defined. This implies $(\mathcal{A}, \sigma'') \models \psi$, because of (6.5), which means that $\sigma' \in \mathsf{WLP}_{\mathcal{A}}(\mathfrak{p}_0, \psi)$, which, as above, means that $(\mathcal{A}, \sigma') \models \gamma$.

Conversely, suppose that

$$\mathcal{A} \models \{\varphi\}\mathfrak{p}_0\{\gamma\}, \tag{6.6}$$

$(\mathcal{A}, \sigma) \models \varphi$ and $\sigma'' = \mathcal{S}^{pr}_{\mathcal{A}}(\mathfrak{p}_0; \mathfrak{p}_1)(\sigma)$ is defined. This implies that the state $\sigma' = \mathcal{S}^{pr}_{\mathcal{A}}(\mathfrak{p}_0)(\sigma)$ is defined and therefore $(\mathcal{A}, \sigma') \models \gamma$, because of (6.6). Thus, $\sigma' \in \mathsf{WLP}_{\mathcal{A}}(\mathfrak{p}_1, \psi)$. Since $\mathcal{S}^{pr}_{\mathcal{A}}(\mathfrak{p}_1)(\sigma')$ is defined (and equal to σ''), it follows that $(\mathcal{A}, \sigma'') \models \psi$.

To prove the second part, assuming that γ expresses $\mathsf{WLP}_{\mathcal{A}}(\mathfrak{p}_1, \psi)$, suppose that the formula α expresses $\mathsf{WLP}_{\mathcal{A}}(\mathfrak{p}_0, \gamma)$. To show that α expresses $\mathsf{WLP}_{\mathcal{A}}(\mathfrak{p}_0; \mathfrak{p}_1, \psi)$, we need to demonstrate that $(\mathcal{A}, \sigma) \models \alpha$ if and only if $\mathcal{S}^{pr}_{\mathcal{A}}(\mathfrak{p}_0; \mathfrak{p}_1)(\sigma)$ is undefined or $(\mathcal{A}, \mathcal{S}^{pr}_{\mathcal{A}}(\mathfrak{p}_0; \mathfrak{p}_1)(\sigma)) \models \psi$.

Suppose that $(\mathcal{A}, \sigma) \models \alpha$. By the hypothesis on α, $\mathcal{S}^{pr}_{\mathcal{A}}(\mathfrak{p}_0)(\sigma)$ is undefined or $(\mathcal{A}, \mathcal{S}^{pr}_{\mathcal{A}}(\mathfrak{p}_0)(\sigma)) \models \gamma$. In the first case, $\mathcal{S}^{pr}_{\mathcal{A}}(\mathfrak{p}_0; \mathfrak{p}_1)(\sigma)$ is undefined, which gives the needed conclusion. The second case has two subcases. In the first subcase, $\mathcal{S}^{pr}_{\mathcal{A}}(\mathfrak{p}_1)(\mathcal{S}^{pr}_{\mathcal{A}}(\mathfrak{p}_0)(\sigma))$ is undefined, which gives the desired conclusion. In the second subcase, $(\mathcal{A}, \mathcal{S}^{pr}_{\mathcal{A}}(\mathfrak{p}_1)(\mathcal{S}^{pr}_{\mathcal{A}}(\mathfrak{p}_0)(\sigma))) \models \psi$, which concludes this implication.

To prove the reverse implication, suppose that $\mathcal{S}^{pr}_{\mathcal{A}}(\mathfrak{p}_0; \mathfrak{p}_1)(\sigma)$ is undefined, or $(\mathcal{A}, \mathcal{S}^{pr}_{\mathcal{A}}(\mathfrak{p}_0; \mathfrak{p}_1)(\sigma)) \models \psi$. This supposition breaks up into three cases.

If $\mathcal{S}^{pr}_{\mathcal{A}}(\mathfrak{p}_0)(\sigma)$ is undefined, then $\sigma \in \mathsf{WLP}_{\mathcal{A}}(\mathfrak{p}_0, \gamma)$, which implies $(\mathcal{A}, \sigma) \models \alpha$.

In the second case, $\mathcal{S}^{pr}_{\mathcal{A}}(\mathfrak{p}_0)(\sigma)$ is defined, but $\mathcal{S}^{pr}_{\mathcal{A}}(\mathfrak{p}_1)(\mathcal{S}^{pr}_{\mathcal{A}}(\mathfrak{p}_0)(\sigma))$ is undefined. This implies that $\mathcal{S}^{pr}_{\mathcal{A}}(\mathfrak{p}_0)(\sigma) \in \mathsf{WLP}_{\mathcal{A}}(\mathfrak{p}_1, \psi)$ which in turn implies $(\mathcal{A}, \mathcal{S}^{pr}_{\mathcal{A}}(\mathfrak{p}_0)(\sigma)) \models \gamma$. This allows us to conclude that $(\mathcal{A}, \sigma) \models \alpha$.

Finally, suppose that $(\mathcal{A}, \mathcal{S}^{pr}_{\mathcal{A}}(\mathfrak{p}_0; \mathfrak{p}_1)(\sigma)) \models \psi$, that is,

$$(\mathcal{A}, \mathcal{S}^{pr}_{\mathcal{A}}(\mathfrak{p}_0)(\mathcal{S}^{pr}_{\mathcal{A}}(\mathfrak{p}_1)(\sigma))) \models \psi.$$

This implies that $\mathcal{S}^{pr}_{\mathcal{A}}(\mathfrak{p}_0)(\sigma) \in \mathsf{WLP}_{\mathcal{A}}(\mathfrak{p}_1, \psi)$ hence, $(\mathcal{A}, \mathcal{S}^{pr}_{\mathcal{A}}(\mathfrak{p}_0)(\sigma)) \models \gamma$. Consequently, $\mathcal{S}^{pr}_{\mathcal{A}}(\mathfrak{p}_0)(\sigma) \in \mathsf{WLP}_{\mathcal{A}}(\mathfrak{p}_0, \gamma)$ which again implies $(\mathcal{A}, \sigma) \models \alpha$. □

Using Theorems 6.5.22 through 6.5.24, it is possible to give an algorithm which, starting from a program $\mathfrak{p}$ that does not contain any **while**, and a postcondition ψ, produces a formula φ that expresses

$\mathsf{WLP}_{\mathcal{A}}(\mathfrak{p}, \psi)$ in all $\mathcal{L}$-structures $\mathcal{A}$. We will produce such an algorithm in the context of annotated programs which we introduce in Section 6.7.

6.6 Hoare Theories

In this section, we investigate the decidability of the set of partial Hoare triples which are valid in a structure.

Definition 6.6.1. The set of $\mathcal{L}$-partial correctness Hoare triples $\{\mathcal{H} \mid \mathcal{A} \models \mathcal{H}\}$, that is the *Hoare partial correctness theory of* $\mathcal{A}$, will be denoted by $\mathsf{HPT}_{\mathcal{L}}(\mathcal{A})$ and the corresponding set of $\mathcal{L}$-total correctness triples, that is the *Hoare total correctness theory of* $\mathcal{A}$, will be denoted by $\mathsf{HTT}_{\mathcal{L}}(\mathcal{A})$.

We will denote the fragment $\mathsf{HPT}_{\mathcal{L}}(\mathcal{A}) \cap \mathsf{HPT}^{\uparrow}_{\mathcal{L}}$ by $\mathsf{HPT}^{\uparrow}_{\mathcal{L}}(\mathcal{A})$. ▯

Theorem 6.6.2. *Let $\mathcal{A}$ be an $\mathcal{L}$-structure, where $\mathcal{L}$ is a first-order language. For any $\mathcal{L}$-sentence φ, we have $\varphi \in Th^{\mathcal{L}}(\mathcal{A})$ if and only if*

$$\{\mathsf{true}_\star^{\mathcal{L}_\varphi}\}x_0{\leftarrow}x_0\{\varphi_\star\} \in \mathsf{HPT}_{\mathcal{L}}(\mathcal{A}).$$

Proof. By Theorem 4.6.23, $\varphi \equiv \varphi_\star$ and $\mathsf{true}^{\mathcal{L}_\varphi} \equiv \mathsf{true}_\star^{\mathcal{L}_\varphi}$, because $\psi_\star$ is a variant of ψ. Suppose first that $\varphi \in \mathrm{Th}^{\mathcal{L}}(\mathcal{A})$. Then, $\mathcal{A} \models \{\theta\}\mathfrak{p}\{\varphi_\star\}$ for any $\theta, \mathfrak{p}$. which in particular shows that $\varphi \in \mathrm{Th}^{\mathcal{L}}(\mathcal{A})$ implies

$$\{\mathsf{true}_\star^{\mathcal{L}_\varphi}\}x_0{\leftarrow}x_0\{\varphi_\star\} \in \mathsf{HPT}_{\mathcal{L}}(\mathcal{A}).$$

Conversely, suppose that $\{\mathsf{true}_\star^{\mathcal{L}_\varphi}\}x_0{\leftarrow}x_0\{\varphi_\star\} \in \mathsf{HPT}_{\mathcal{L}}(\mathcal{A})$. Let σ be an $\mathcal{A}$-state assumed to be the state before the program $x_0{\leftarrow}x_0$ begins. Since $(\mathcal{A}, \sigma) \models \mathsf{true}_\star^{\mathcal{L}_\varphi}$ and the state at the completion of the program $x_0{\leftarrow}x_0$ is still σ, we have $(\mathcal{A}, \sigma) \models \varphi_\star$, which means that $(\mathcal{A}, \sigma) \models \varphi$, that is, $\varphi \in \mathrm{Th}^{\mathcal{L}}(\mathcal{A})$ because φ is a sentence. □

Corollary 6.6.3. *Let $\mathcal{L}$ be a first-order language and $\mathcal{A}$ be an $\mathcal{L}$-structure. Then, $Th^{\mathcal{L}}(\mathcal{A}) \leq_m \mathsf{HPT}_{\mathcal{L}}(\mathcal{A})$.*

Proof. In the sense of Definition 1.4.8, we take the universal set of $\mathrm{Th}^{\mathcal{L}}(\mathcal{A})$ to be SENT and the universal set for $\mathsf{HPT}_{\mathcal{L}}(\mathcal{A})$ to be HPT.

The many-to-one reduction function $f : \mathrm{SENT} \longrightarrow \mathsf{HPT}$ is given by

$$f(\varphi) = \{\mathsf{true}_\star^{\mathcal{L}_\varphi}\}x_0{\leftarrow}x_0\{\varphi_\star\}.$$

If $\varphi \in \mathrm{SENT} - \mathrm{SENT}_\mathcal{L}$, then $f(\varphi) \notin \mathsf{HPT}_\mathcal{L}$, while if $\varphi \in \mathrm{SENT}_\mathcal{L}$, then, by Theorem 6.6.2, $\varphi \in \mathrm{Th}^\mathcal{L}(\mathcal{A})$ if and only if $f(\varphi) \in \mathsf{HPT}_\mathcal{L}(\mathcal{A})$. □

Corollary 6.6.4. *If $\mathcal{L}$ is a decidable first-order language and $Th^\mathcal{L}(\mathcal{A})$ is undecidable, then $\mathsf{HPT}_\mathcal{L}(\mathcal{A})$ is not semidecidable (and thus is undecidable).*

Proof. This statement follows directly from Corollaries 4.13.22 and 6.6.3, and Theorem 1.4.9. □

Corollary 6.6.5. *The set $\mathsf{HPT}_{\mathcal{L}_{ar}}(\mathcal{A}_{ar})$ is not semidecidable (and thus is undecidable).*

Proof. This statement follows from Corollaries 4.14.16 and 6.6.4. □

There are structures for which a reverse reduction is possible: namely, it is possible to show that $\mathsf{HPT}_\mathcal{L}(\mathcal{A}) \leq_m \mathrm{Th}^\mathcal{L}(\mathcal{A})$. A sufficient condition under which the reverse reduction is possible is formalized in the next definition.

Definition 6.6.6. Let $\mathcal{L}$ be a first-order language. An $\mathcal{L}$-structure $\mathcal{A}$ is *expressive* if for all $\mathcal{L}$-programs $\mathfrak{p}$ and all $\mathcal{L}$-assertions ψ, the set $\mathsf{WLP}_\mathcal{A}(\mathfrak{p}, \psi)$ is expressible by a single formula $\omega^\mathcal{A}(\mathfrak{p}, \psi)$.

$\mathcal{A}$ is *effectively expressive* if there is an effective way of constructing the formula $\omega^\mathcal{A}(\mathfrak{p}, \psi)$ given $\mathfrak{p}$ and ψ. ▯

More precisely, $\mathcal{A}$ is effectively expressive if there is an effectively computable function $\omega^\mathcal{A} : \mathtt{WHILE} \times \mathrm{ASSERT} \longrightarrow \mathrm{FORM}$ such that when $\mathfrak{p} \in \mathtt{WHILE}_\mathcal{L}$ and $\psi \in \mathrm{ASSERT}_\mathcal{L}$, the formula $\omega^\mathcal{A}(\mathfrak{p}, \psi)$ expresses $\mathsf{WLP}_\mathcal{A}(\mathfrak{p}, \psi)$.

In the previous definition, one could assume that $\gamma = \omega^\mathcal{A}(\mathfrak{p}, \psi)$ is an assertion by replacing it with the assertion $\gamma_\star$.

The next theorem shows that in effectively expressive structures, the reverse reduction mentioned above is possible.

Theorem 6.6.7. *Let $\mathcal{L}$ be a first-order language, $\mathcal{A}$ be an $\mathcal{L}$-structure and $\mathcal{H} = \{\varphi\}\mathfrak{p}\{\psi\}$ be a Hoare partial correctness triple. Assume that* *the $\mathcal{L}$-formula $\omega^{\mathcal{A}}(\mathfrak{p}, \psi)$ expresses $\mathsf{WLP}_{\mathcal{A}}(\mathfrak{p}, \psi)$. Then, the following conditions are equivalent*:

(1) $\mathcal{H} \in \mathsf{HPT}_{\mathcal{L}}(\mathcal{A})$;
(2) $\mathcal{A} \models (\varphi \to \omega^{\mathcal{A}}(\mathfrak{p}, \psi))$;
(3) $(\varphi \to \omega^{\mathcal{A}}(\mathfrak{p}, \psi))^{\forall} \in Th^{\mathcal{L}}(\mathcal{A})$.

Proof. We leave to the reader the proof of the equivalence between (1) and (2), which is a mere application of the relevant definitions; the equivalence between (2) and (3) is a consequence of the second part of Theorem 4.5.58 and the definition of $\mathrm{Th}^{\mathcal{L}}(\mathcal{A})$. □

Corollary 6.6.8. *Let $\mathcal{L}$ be a decidable first-order language and $\mathcal{A}$ be an effectively expressive $\mathcal{L}$-structure. Then, $\mathsf{HPT}_{\mathcal{L}}(\mathcal{A}) \leq_m Th^{\mathcal{L}}(\mathcal{A})$.*

Proof. The reduction function $f : \mathsf{HPT} \longrightarrow \mathrm{SENT}$ is given by

$$f(\{\varphi\}\mathfrak{p}\{\psi\}) = \begin{cases} (\varphi \to \omega^{\mathcal{A}}(\mathfrak{p}, \psi))^{\forall} & \text{if } \{\varphi\}\mathfrak{p}\{\psi\} \in \mathsf{HPT}_{\mathcal{L}}, \\ \alpha_0 & \text{otherwise}, \end{cases}$$

where $\omega^{\mathcal{A}}(\mathfrak{p}, \psi)$ is as in the definition of effectively expressive $\mathcal{L}$-structure and α_0 is a fixed element of $\mathrm{SENT} - \mathrm{SENT}_{\mathcal{L}}$. The decidability of $\mathcal{L}$ implies that $\mathsf{HPT}_{\mathcal{L}}$ is decidable and hence f is computable. The statement follows from Theorem 6.6.7. □

Corollary 6.6.9. *Let $\mathcal{L}$ be a decidable first-order language, $\mathcal{A}$ be an effectively expressive $\mathcal{L}$-structure. If $Th^{\mathcal{L}}(\mathcal{A})$ is decidable, then $\mathsf{HPT}_{\mathcal{L}}(\mathcal{A})$ is decidable.*

Proof. This follows from Corollary 6.6.8 and Theorem 1.4.9. □

We now explore the expressiveness of specific structures. We begin by proving the existence of an upper bound on the running time of programs in finite structures.

Theorem 6.6.10. *There is an effectively computable function*

$$\mathsf{B}^{pr} : \mathtt{WHILE} \times \mathbf{P} \longrightarrow \mathbf{N}$$

(where $\mathbf{P}$ is the set of positive natural numbers) such that for all first-order languages $\mathcal{L}$ and finite $\mathcal{L}$-structures $\mathcal{A}$, if $\mathfrak{p}$ is an $\mathcal{L}$-program, $|\mathcal{A}|$ contains n elements, and $\sigma \in \mathrm{STATES}_{\mathcal{A}}$ is such that $\mathsf{time}^{pr}_{\mathcal{A}}(\mathfrak{p})(\sigma)$ is defined, we have $\mathsf{time}^{pr}_{\mathcal{A}}(\mathfrak{p})(\sigma) \leq \mathsf{B}^{pr}(\mathfrak{p}, n)$.

Proof. We will define B^{pr} recursively on the definition of programs. This kind of definition requires that we construct simultaneously two functions B^{pr} and $\mathsf{B}^{at\text{-}pr}$, where the first function will be defined for programs in general and the second will be defined for atomic programs.

If $\mathfrak{p}$ is $v{\leftarrow}t$, we have $\mathsf{B}^{at\text{-}pr}(\mathfrak{p}, n) = 1$ for all n.

If $\mathfrak{p}$ is the atomic program **if** β **then** $\mathfrak{q}$ **else** $\mathfrak{r}$ **endif**, then

$$\mathsf{B}^{at\text{-}pr}(\mathfrak{p}, n) = 1 + \max\{\mathsf{B}^{pr}(\mathfrak{q}, n), \mathsf{B}^{pr}(\mathfrak{r}, n)\}.$$

Suppose now that $\mathfrak{p}$ is the atomic program **while** β **do** $\mathfrak{q}$ **endwhile**. Let m be the number of variables that occur in $\mathfrak{p}$ and let $s = n^m$. Define $\mathsf{B}^{at\text{-}pr}(\mathfrak{p}, n) = s\mathsf{B}^{pr}(\mathfrak{q}, n) + s + 1$.

For an atomic program $\mathfrak{p}$, define $\mathsf{B}^{pr}(\mathfrak{p}, n) = \mathsf{B}^{at\text{-}pr}(\mathfrak{p}, n)$ for all n.

Finally, if $\mathfrak{p} = \mathfrak{p}'; \mathfrak{p}''$, where $\mathfrak{p}'$ is an atomic program and $\mathfrak{p}''$ is a program, define $\mathsf{B}^{pr}(\mathfrak{p}, n) = \mathsf{B}^{at\text{-}pr}(\mathfrak{p}', n) + \mathsf{B}^{pr}(\mathfrak{p}'', n)$.

Let $\mathcal{A}$ be a finite $\mathcal{L}$-structure such that $|\mathcal{A}|$ contains n elements. The proof of the inequality $\mathsf{time}^{pr}_{\mathcal{A}}(\mathfrak{p})(\sigma) \leq \mathsf{B}^{pr}(\mathfrak{p}, n)$ when $\mathsf{time}^{pr}_{\mathcal{A}}(\mathfrak{p})(\sigma)$ is defined is by induction on programs. Actually, we need to show that $\mathsf{time}^{at\text{-}pr}_{\mathcal{A}}(\mathfrak{p})(\sigma) \leq \mathsf{B}^{at\text{-}pr}(\mathfrak{p}, n)$ for all atomic programs $\mathfrak{p}$ when $\mathsf{time}^{at\text{-}pr}_{\mathcal{A}}(\mathfrak{p})(\sigma)$ is defined, and $\mathsf{time}^{pr}_{\mathcal{A}}(\mathfrak{p})(\sigma) \leq \mathsf{B}^{pr}(\mathfrak{p}, n)$ for all programs $\mathfrak{p}$ when $\mathsf{time}^{pr}_{\mathcal{A}}(\mathfrak{p})(\sigma)$ is defined. We discuss only one inductive step, namely when $\mathfrak{p}$ is the atomic program **while** β **do** $\mathfrak{q}$ **endwhile**.

Assume that $\mathsf{time}^{at\text{-}pr}_{\mathcal{A}}(\mathfrak{p})(\sigma)$ is defined for a state σ, but $\mathsf{time}^{at\text{-}pr}_{\mathcal{A}}(\mathfrak{p})(\sigma) > \mathsf{B}^{at\text{-}pr}(\mathfrak{p}, n) = s\mathsf{B}^{pr}(\mathfrak{q}, n) + s + 1$. Let $\sigma_i = \left(\mathcal{S}^{at\text{-}pr}_{\mathcal{A}}(\mathfrak{q})\right)^{(i)}(\sigma)$ and let k be the least integer such that $(\mathcal{A}, \sigma_k) \models (\neg\beta)$. We claim that under this assumption, $k > s$. Indeed, if k were less than or equal to s we would have

$$\begin{aligned}\mathsf{time}^{at\text{-}pr}_{\mathcal{A}}(\mathfrak{p})(\sigma) &= k + 1 + \sum_{i=0}^{k-1} \mathsf{time}^{pr}_{\mathcal{A}}(\mathfrak{q})(\sigma_i)\\ &\leq k + 1 + k\mathsf{B}^{pr}(\mathfrak{q}, n)\\ &\leq s + 1 + s\mathsf{B}^{pr}(\mathfrak{q}, n),\end{aligned}$$

which contradicts our initial assumption. Note that s is the number of distinct ways of assigning values in $|\mathcal{A}|$ to the variables that occur in the program. Since $k > s$, there are i, j, $0 \leq i < j < k$, such that $\sigma_i(v) = \sigma_j(v)$ for every variable v that occurs in the program $\mathfrak{p}$. Actually, $\sigma_i = \sigma_j$, by Theorem 6.3.5, which means that the program $\mathfrak{p}$ will cycle indefinitely between these states. Specifically, there is an l such that $i \leq l < j$ and $\sigma_l = \sigma_k$, and, because $(\mathcal{A}, \sigma_l) \models \beta$, this contradicts the definition of k. $\square$

Corollary 6.6.11. *For any finite $\mathcal{L}$-structure $\mathcal{A}$ with $|\mathcal{A}|$ containing n elements, $\mathcal{L}$-assertion ψ and $\mathcal{L}$-program $\mathfrak{p}$, we have*

$$\mathsf{WLP}_{\mathcal{A}}(\mathfrak{p}, \psi) = \mathsf{WLP}_{\mathcal{A}}^{\mathsf{B}^{pr}(\mathfrak{p},n)}(\mathfrak{p}, \psi).$$

Proof. The statement follows from Theorem 6.6.10 and Definition 6.5.14. $\square$

Theorem 6.6.12. *Every finite $\mathcal{L}$-structure $\mathcal{A}$ is effectively expressive.*

Proof. The effectiveness of the construction of B^{pr} established in Theorem 6.6.10 combined with the effectiveness of the construction of the formulas $\omega^n(\mathfrak{p}, \psi)$ shown in Theorem 6.5.18 shows that we can construct effectively the formula $\omega^{\mathsf{B}^{pr}(\mathfrak{p},n)}(\mathfrak{p}, \psi)$, which expresses $\mathsf{WLP}_{\mathcal{A}}^{\mathsf{B}^{pr}(\mathfrak{p},n)}(\mathfrak{p}, \psi)$. By Corollary 6.6.11 this formula also expresses the set $\mathsf{WLP}_{\mathcal{A}}(\mathfrak{p}, \psi)$, which concludes the argument. $\square$

Corollary 6.6.13. *Let $\mathcal{L}$ be a finite language and $\mathcal{A}$ be a finite $\mathcal{L}$-structure. Then the Hoare partial correctness theory of $\mathcal{A}$ is decidable.*

Proof. This statement is a consequence of Corollary 6.6.9, Theorem 6.6.12 and Theorem 4.14.3. $\square$

We now turn to proving the effective expressiveness of arithmetic. This will require the introduction of some notations and preliminary results. Recall that in Section 4.5.2 we introduced the notation $[a \to b]$ for $[a \to b]\emptyset$. Thus, when $a_0, \ldots, a_{n-1}$ are distinct, $[a_0 \to b_0] \cdots [a_{n-1} \to b_{n-1}]$ denotes the function f defined on the set $\{a_0, \ldots, a_{n-1}\}$ with $f(a_i) = b_i$ for $0 \leq i \leq n-1$. In a further simplification, the same function will be denoted by $[a_0, \ldots, a_{n-1} \to b_0, \ldots, b_{n-1}]$.

Definition 6.6.14. Let $\mathcal{A} = (A, \mathcal{I})$ be a $\mathcal{L}$-structure. An $(\mathcal{A}, V)$-*partial state* is a partial assignment $\sigma : V \rightsquigarrow |\mathcal{A}|$ with domain V. We will denote the set of $(\mathcal{A}, V)$-states by $\mathrm{PSTATES}_{\mathcal{A}}(V)$. $\square$

The State Agreement Theorem allows us to define the semantics of programs in terms of transformations of partial states.

Definition 6.6.15. Let $\mathcal{L}$ be a first-order language, $\mathcal{A}$ be an $\mathcal{L}$-structure, and let $\mathfrak{p}$ be an $\mathcal{L}$-program. The *restricted semantics function*

$$\mathcal{RS}^{pr}_{\mathcal{A}}(\mathfrak{p}) : \mathrm{PSTATES}_{\mathcal{A}}(\mathrm{PVAR}(\mathfrak{p})) \rightsquigarrow \mathrm{PSTATES}_{\mathcal{A}}(\mathrm{PVAR}(\mathfrak{p}))$$

is the partial function defined by $\mathcal{RS}^{pr}_{\mathcal{A}}(\mathfrak{p})(\sigma) = \mathcal{S}^{pr}_{\mathcal{A}}(\mathfrak{p})(\sigma') \restriction \mathrm{PVAR}(\mathfrak{p})$, where σ' is some $\mathcal{A}$-state such that $\sigma' \restriction \mathrm{PVAR}(\mathfrak{p}) = \sigma$, whenever $\mathcal{S}^{pr}_{\mathcal{A}}(\mathfrak{p})(\sigma')$ is defined. Otherwise, $\mathcal{RS}^{pr}_{\mathcal{A}}(\mathfrak{p})$ is not defined for σ. □

Observe that by the State Agreement Theorem, if σ', σ'' are two $\mathcal{A}$-states that extend the partial $\mathcal{A}$-state σ, then $\mathcal{S}^{pr}_{\mathcal{A}}(\mathfrak{p})(\sigma')$ is defined if and only if $\mathcal{S}^{pr}_{\mathcal{A}}(\mathfrak{p})(\sigma'')$ is defined and if these states are defined, their restrictions to $\mathrm{PVAR}(\mathfrak{p})$ coincide. This shows that the function $\mathcal{RS}^{pr}_{\mathcal{A}}(\mathfrak{p})$ is well-defined.

Lemma 6.6.16. *Let $\mathcal{L}$ be a first-order language with equality, $\mathcal{A}$ be an $\mathcal{L}$-structure, and let $\mathfrak{p}$ be an $\mathcal{L}$-program, where $PVAR(\mathfrak{p}) = \{v_0, \ldots, v_{n-1}\}$.*

Let $y_0, \ldots, y_{n-1}, z_0, \ldots, z_{n-1}, z'_0, \ldots, z'_{n-1}$ be distinct program variables that do not occur in $\mathfrak{p}$. Suppose that the formula α expresses the weakest liberal precondition $\mathsf{WLP}_{\mathcal{A}}(\mathfrak{p}, \psi)$, where ψ is

$$(\neg(v_0 = z_0 \wedge \cdots \wedge v_{n-1} = z_{n-1} \wedge y_0 = z'_0 \wedge \cdots \wedge y_{n-1} = z'_{n-1})).$$

Define the formulas β, θ as

$$\begin{aligned} \beta &= (\exists v_0) \cdots (\exists v_{n-1})(\exists y_0) \cdots (\exists y_{n-1})(v_0 = y_0 \wedge \cdots \\ &\quad \wedge v_{n-1} = y_{n-1} \wedge (\neg\alpha)) \\ \theta &= \langle \beta \rangle_{z_0, \ldots, z_{n-1}, z'_0, \ldots, z'_{n-1} := v_0, \ldots, v_{n-1}, y_0, \ldots, y_{n-1}}, \end{aligned}$$

Then, for all $\mathcal{A}$-states σ, $(\mathcal{A}, \sigma) \models \theta$ if and only if

$$\mathcal{RS}^{pr}_{\mathcal{A}}(\mathfrak{p})([v_0, \ldots, v_{n-1} \to \sigma(y_0), \ldots, \sigma(y_{n-1})]) = \sigma \restriction PVAR(\mathfrak{p}).$$

Proof. We will show first the equivalence of the following statements:

(1) $(\mathcal{A},\sigma) \models \theta$.
(2) $(\mathcal{A},\sigma') \models \beta$, where σ' is

$$[z_0,\dots,z_{n-1},z'_0,\dots,z'_{n-1} \\ \to \sigma(v_0),\dots,\sigma(v_{n-1}),\sigma(y_0),\dots,\sigma(y_{n-1})]\sigma.$$

(3) There are $a_0,\dots,a_{n-1}$ in $|\mathcal{A}|$ such that for σ'' given by

$$[v_0,\dots,v_{n-1},y_0,\dots,y_{n-1} \to a_0,\dots,a_{n-1},a_0,\dots,a_{n-1}]\sigma'$$

and σ' given by

$$[z_0,\dots,z_{n-1},z'_0,\dots,z'_{n-1} \\ \to \sigma(v_0),\dots,\sigma(v_{n-1}),\sigma(y_0),\dots,\sigma(y_{n-1})]\sigma.$$

we have $(\mathcal{A},\sigma'') \models (\neg\alpha)$.
(4) There are $a_0,\dots,a_{n-1}$ in $|\mathcal{A}|$ such that $\sigma''' = \mathcal{S}^{pr}_{\mathcal{A}}(\mathfrak{p})(\sigma'')$ is defined and we have

$$(\mathcal{A},\sigma''') \models (v_0 = z_0 \wedge\cdots\wedge v_{n-1} = z_{n-1} \wedge y_0 = z'_0 \wedge\cdots\wedge y_{n-1} = z'_{n-1}),$$

where σ'' is

$$[v_0,\dots,v_{n-1},y_0,\dots,y_{n-1} \to a_0,\dots,a_{n-1},a_0,\dots,a_{n-1}]\sigma'$$

and σ' is

$$[z_0,\dots,z_{n-1},z'_0,\dots,z'_{n-1} \\ \to \sigma(v_0),\dots,\sigma(v_{n-1}),\sigma(y_0),\dots,\sigma(y_{n-1})]\sigma.$$

The equivalence of (1) and (2) follows from Supplement 90 of Chapter 4. The equivalence of (2) and (3) follows from Tarski's definition of truth applied to the formula β. Finally, the equivalence between (3) and (4) follows from the fact that α expresses the weakest liberal precondition corresponding to the formula ψ and $\mathfrak{p}$.

Thus, it suffices to show that (4) is equivalent to

$$\mathcal{RS}^{pr}_{\mathcal{A}}(\mathfrak{p})([v_0,\dots,v_{n-1} \to \sigma(y_0),\dots,\sigma(y_{n-1})]) = \sigma\restriction \mathrm{PVAR}(\mathfrak{p}).$$

Suppose that (4) holds. Observe that $\sigma'(z_i) = \sigma(v_i)$, $\sigma'(z'_i) = \sigma(y_i)$, $\sigma''(v_i) = \sigma''(y_i) = a_i$, $\sigma''(z_i) = \sigma'(z_i)$, $\sigma''(z'_i) = \sigma'(z'_i)$, $\sigma'''(v_i) = \sigma'''(z_i)$, and $\sigma'''(y_i) = \sigma'''(z'_i)$. Moreover, since y_i, z_i, z'_i do not occur

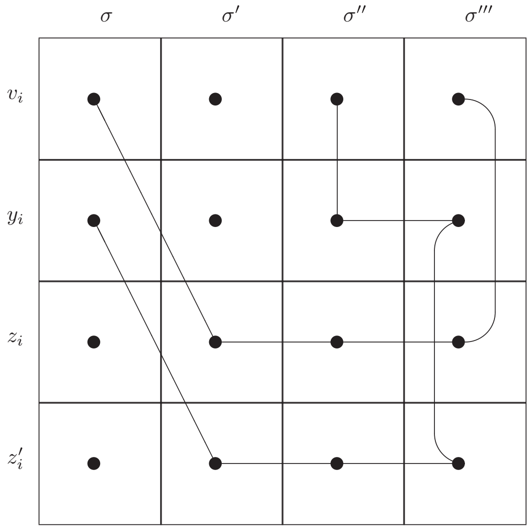


Fig. 6.1. Diagram representing equalities.

in $\mathfrak{p}$, we also have $\sigma'''(y_i) = \sigma''(y_i)$, $\sigma'''(z_i) = \sigma''(z_i)$ and $\sigma'''(z'_i) = \sigma''(z'_i)$. The equalities inferred above are summarized in Figure 6.1. An inspection of this figure shows immediately that $\sigma''(v_i) = \sigma(y_i)$ and $\sigma'''(v_i) = \sigma(v_i)$, for $0 \leq i \leq n-1$. Note that $\sigma''\restriction\mathrm{PVAR}(\mathfrak{p}) = [v_0, \ldots, v_{n-1} \to \sigma(y_0), \ldots, \sigma(y_{n-1})]$ and $\sigma'''\restriction\mathrm{PVAR}(\mathfrak{p}) = \sigma\restriction\mathrm{PVAR}(\mathfrak{p})$. Since

$$\mathcal{RS}^{pr}_{\mathcal{A}}\boldsymbol{(}\mathfrak{p}\boldsymbol{)}(\sigma''\restriction\mathrm{PVAR}(\mathfrak{p})) = \sigma'''\restriction\mathrm{PVAR}(\mathfrak{p}),$$

we obtain the desired result.

Conversely, suppose that

$$\mathcal{RS}^{pr}_{\mathcal{A}}\boldsymbol{(}[v_0, \ldots, v_{n-1} \to \sigma(y_0), \ldots, \sigma(y_{n-1})]\boldsymbol{)} = \sigma\restriction\mathrm{PVAR}(\mathfrak{p}).$$

Define $a_i = \sigma(y_i)$ for $0 \leq i \leq n-1$,

$$\sigma' = [z_0, \ldots, z_{n-1}, z'_0, \ldots, z'_{n-1} \to \sigma(v_0), \ldots, \sigma(v_{n-1}),$$
$$\sigma(y_0), \ldots, \sigma(y_{n-1})]\sigma.$$

and

$$\sigma'' = [v_0, \ldots, v_{n-1}, y_0, \ldots, y_{n-1} \to a_0, \ldots, a_{n-1}, a_0, \ldots, a_{n-1}]\sigma'.$$

Observe that $\sigma''\restriction\mathrm{PVAR}(\mathfrak{p}) = [v_0, \ldots, v_{n-1} \to \sigma(y_0), \ldots, \sigma(y_{n-1})]$ and, therefore, by hypothesis, $\sigma''' = \mathcal{S}^{pr}_{\mathcal{A}}\boldsymbol{(}\mathfrak{p}\boldsymbol{)}(\sigma'')$ is defined and

$\sigma'''\lceil \mathrm{PVAR}(\mathfrak{p}) = \sigma\lceil \mathrm{PVAR}(\mathfrak{p})$. Note now that we have the following two chains of equalities:

$$\begin{aligned}&\sigma'''(v_i) = \sigma(v_i) = \sigma'(z_i) = \sigma''(z_i) = \sigma'''(z_i),\\ &\sigma'''(y_i) = \sigma''(y_i) = a_i = \sigma(y_i) = \sigma'(z'_i) = \sigma''(z'_i) = \sigma'''(z'_i),\end{aligned}$$

which shows that

$$(\mathcal{A}, \sigma''') \models (v_0 = z_0 \wedge \cdots \wedge v_{n-1} = z_{n-1} \wedge y_0 = z'_0 \wedge \cdots \wedge y_{n-1} = z'_{n-1}).$$

□

Let $\mathcal{L}_{ar,f_{\mathsf{b}}}$ be the language obtained from $\mathcal{L}_{ar}$ by adding a ternary function symbol f_{b} and let $\mathcal{A}_{ar,f_{\mathsf{b}}}$ be the expansion of $\mathcal{A}_{ar}$ obtained by defining $f_{\mathsf{b}}^{\mathcal{A}_{ar,f_{\mathsf{b}}}} = \mathsf{b}$, where b is the Gödel function defined on page 583. We denote the formula

$$\begin{aligned}&(w_0 = f_{\mathsf{b}}(v_c, v_d, v_k \cdot s^n(0)) \wedge w_1 = f_{\mathsf{b}}(v_c, v_d, v_k \cdot s^n(0) + s(0)) \wedge \cdots\\ &\wedge w_{n-1} = f_{\mathsf{b}}(v_c, v_d, v_k \cdot s^n(0) + s^{n-1}(0)))\end{aligned}$$

by $\vec{w} = \mathsf{seq}_{v_c,v_d,v_k,n}$, where $v_c, v_d, v_k, w_0, \ldots, w_{n-1}$ are variables. Then, we have $(\mathcal{A}_{ar,\mathsf{b}}, \sigma) \models \vec{w} = \mathsf{seq}_{v_c,v_d,v_k,n}$ if and only if for every i, $0 \le i \le n-1$, we have $\sigma(w_i) = \mathsf{b}(\sigma(v_c), \sigma(v_d), n\sigma(v_k) + i)$. This corresponds to dividing the sequence coded by $(\sigma(v_c), \sigma(v_d))$ into blocks of length n and asserting that the values in the $(k+1)$st block are in the variables $w_0, \ldots, w_{n-1}$, where $k = \sigma(v_k)$.

Similarly, we denote

$$\begin{aligned}&(v_0 = f_{\mathsf{b}}(v_c, v_d, 0) \wedge v_1 = f_{\mathsf{b}}(v_c, v_d, s(0)) \wedge \cdots\\ &\wedge v_{n-1} = f_{\mathsf{b}}(v_c, v_d, s^{n-1}(0)))\end{aligned}$$

by $\vec{v} = \mathsf{initseq}_{v_c,v_d,n}$, where $v_c, v_d, v_0, \ldots, v_{n-1}$ are distinct variables.

Lemma 6.6.17. *Let $\mathcal{L}_{ar}$ be the language of arithmetic, and let $\mathfrak{p}$ be an $\mathcal{L}_{ar}$-program, where $PVAR(\mathfrak{p}) = \{v_0, \ldots, v_{n-1}\}$. Further, suppose that the formula α expresses the weakest liberal precondition $\mathit{WLP}_{\mathcal{A}_{ar}}(\mathfrak{p}, \psi')$, where ψ' is:*

$$(\neg(v_0 = z_0 \wedge \cdots \wedge v_{n-1} = z_{n-1} \wedge y_0 = z'_0 \wedge \cdots \wedge y_{n-1} = z'_{n-1})),$$

and $y_0, \ldots, y_{n-1}, z_0, \ldots, z_{n-1}, z'_0, \ldots, z'_{n-1}$ are distinct program variables that do not occur in $\mathfrak{p}$.

Then, there is an effective construction which, given an $\mathcal{L}_{ar}$-assertion ψ, produces an $\mathcal{L}_{ar}$-assertion $\omega^{\mathcal{A}_{ar}}(\mathfrak{q}, \psi)$ that expresses the weakest liberal precondition of the program

$$\mathfrak{q} = \textbf{\textit{while}}\ \beta\ \textbf{\textit{do}}\ \mathfrak{p}\ \textbf{\textit{endwhile}}$$

and the assertion ψ, where β is a quantifier-free $(\mathcal{L}_{ar}, PVAR)$-formula.

Proof. We will first construct an $\mathcal{L}_{ar,f_{\mathsf{b}}}$-formula $\mathsf{precond}$ starting from the following five subformulas:

- The formula init_{v_c,v_d} is given by $\vec{v} = \mathsf{initseq}_{v_c,v_d,n}$. Clearly, we have $(\mathcal{A}_{ar,f_{\mathsf{b}}}, \sigma) \models \mathsf{init}_{v_c,v_d}$ if and only if $\sigma(v_0), \ldots, \sigma(v_{n-1})$ are the first n components of the sequence encoded by the pair $(\sigma(v_c), \sigma(v_d))$.
- The formula $\mathsf{step}_{v_c,v_d,v_m}$ is given by

$$(\forall w_0) \cdots (\forall w_{n-1})(\forall w'_0) \cdots (\forall w'_{n-1})(\forall v_k)((v_k < v_m \wedge \\ \vec{w} = \mathsf{seq}_{v_c,v_d,v_k} \wedge \vec{w'} = \mathsf{seq}_{v_c,v_d,v_k+s(0)}) \rightarrow \langle\theta\rangle_{\vec{v},\vec{y}:=\vec{w'},\vec{w}}),$$

  where $w_0, \ldots, w_{n-1}, w'_0, \ldots, w'_{n-1}, v_k$ are pairwise distinct variables that are also distinct from v_c, v_d, v_m and θ is the formula introduced in Lemma 6.6.16.
  Let σ be an $\mathcal{A}_{ar}$-state. Now, we have $(\mathcal{A}_{ar,f_{\mathsf{b}}}, \sigma) \models \vec{w} = \mathsf{seq}_{v_c,v_d,v_k}$ if and only if $\sigma(w_0), \ldots, \sigma(w_{n-1})$ is the $(k+1)$st block of length n of the sequence encoded by $(\sigma(v_c), \sigma(v_d))$, where $k = \sigma(v_k)$. Likewise, $(\mathcal{A}_{ar,f_{\mathsf{b}}}, \sigma) \models \vec{w'} = \mathsf{seq}_{v_c,v_d,v_k+s(0)}$ if and only if $\sigma(w'_0), \ldots, \sigma(w'_{n-1})$ is the $(k+2)$nd block of length n of the same sequence. Finally, we have $(\mathcal{A}_{ar,f_{\mathsf{b}}}, \sigma) \models \langle\theta\rangle_{\vec{v},\vec{y}:=\vec{w'},\vec{w}}$ if and only if

$$(\mathcal{A}_{ar,f_{\mathsf{b}}}, [\vec{v}, \vec{y} \rightarrow \sigma(\vec{w'}), \sigma(\vec{w})]\sigma) \models \theta.$$

  This is equivalent to

$$\mathcal{RS}^{pr}_{\mathcal{A}_{ar,f_{\mathsf{b}}}}(\mathfrak{p})([v_0, \ldots, v_{n-1} \rightarrow \sigma(w_0), \ldots, \sigma(w_{n-1})]) \\ = [v_0, \ldots, v_{n-1} \rightarrow \sigma(w'_0), \ldots, \sigma(w'_{n-1})],$$

  by Lemma 6.6.16.
  This allows us to conclude that $(\mathcal{A}_{ar,f_{\mathsf{b}}}, \sigma) \models \mathsf{step}_{v_c,v_d,v_m}$ if and only if for the first $\sigma(v_m) + 1$ blocks of length n of the sequence

encoded by $(\sigma(v_c), \sigma(v_d))$, the $(k+1)$st block represents the values of the program variables of $\mathfrak{p}$ when $\mathfrak{p}$ is started with the values in the kth block. This latter condition is equivalent to saying that for any state τ with $\tau(v_j) = \mathsf{b}(\sigma(v_c), \sigma(v_d), j)$ for $0 \leq j \leq n-1$, the state $\tau_i = (\mathcal{S}^{pr}_{\mathcal{A}_{ar}}(\mathfrak{p}))^{(i)}(\tau)$ is defined for $0 \leq i < \sigma(v_m)$ and $\tau_i(v_j)$ is the jth value in the $(i+1)$st block of length n of the sequence encoded by the pair $(\sigma(v_c), \sigma(v_d))$.

- The formula $\mathsf{continuation}_{v_c,v_d,v_m}$ is

$$(\forall w_0)\cdots(\forall w_{n-1})(\forall v_k)((v_k < v_m \wedge \vec{w} = \mathsf{seq}_{v_c,v_d,v_k}) \to (\beta)_{\vec{v}:=\vec{w}}).$$

  We have $(\mathcal{A}_{ar,f_{\mathsf{b}}}, \sigma) \models \mathsf{continuation}_{v_c,v_d,v_m}$ if and only if for the first $\sigma(v_m)$ blocks of length n of the sequence encoded by $(\sigma(v_c), \sigma(v_d))$, we have $(\mathcal{A}_{ar,f_{\mathsf{b}}}, [v_0, \ldots, v_{n-1} \to a_0, \ldots, a_{n-1}]\sigma) \models \beta$, where $a_0, \ldots, a_{n-1}$ are the entries of the block.

- The formula $\mathsf{termination}_{v_c,v_d,v_m}$ is

$$(\forall w_0)\cdots(\forall w_{n-1})(\vec{w} = \mathsf{seq}_{v_c,v_d,v_m} \to \langle(\neg\beta)\rangle_{\vec{v}:=\vec{w}}).$$

  Now, we have $(\mathcal{A}_{ar,f_{\mathsf{b}}}, \sigma) \models \mathsf{termination}_{v_c,v_d,v_m}$ if and only if

$$(\mathcal{A}_{ar,f_{\mathsf{b}}}, [v_0, \ldots, v_{n-1} \to a_0, \ldots, a_{n-1}]\sigma) \models (\neg\beta),$$

  where $a_0, \ldots, a_{n-1}$ are the entries of the $(\sigma(v_m)+1)$st block of the sequence encoded by the pair $(\sigma(v_c), \sigma(v_d))$.

- The formula $\mathsf{verification}_{v_c,v_d,v_m}$ is

$$(\forall w_0)\cdots(\forall w_{n-1})(\vec{w} = \mathsf{seq}_{v_c,v_d,v_m} \to \langle\psi\rangle_{\vec{v}:=\vec{w}}).$$

  We have $(\mathcal{A}_{ar,f_{\mathsf{b}}}, \sigma) \models \mathsf{verification}_{v_c,v_d,v_m}$ if and only if

$$(\mathcal{A}_{ar,f_{\mathsf{b}}}, [v_0, \ldots, v_{n-1} \to a_0, \ldots, a_{n-1}]\sigma) \models \psi,$$

  where the entries of the $(\sigma(v_m)+1)$st block of the sequence encoded by the pair $(\sigma(v_c), \sigma(v_d))$ are $a_0, \ldots, a_{n-1}$.

Thus, the formula $\mathsf{precond}$ we are seeking is

$$(\forall v_c)(\forall v_d)(\forall v_m)((\mathsf{init}_{v_c,v_d} \wedge \mathsf{step}_{v_c,v_d,v_m} \wedge \mathsf{continuation}_{v_c,v_d,v_m} \wedge \\ \mathsf{termination}_{v_c,v_d,v_m}) \to \mathsf{verification}_{v_c,v_d,v_m}),$$

where v_c, v_d, v_m are variables distinct from the variables $v_0, \ldots, v_{n-1}$ and from the bound variables $w_0, \ldots, w_{n-1}, w'_0, \ldots, w'_{n-1}$ and do not appear in β or in ψ.

We prove that $(\mathcal{A}_{ar,f_{\mathsf{b}}}, \sigma) \models \mathsf{precond}$ if and only if $\sigma \in \mathsf{WLP}_{\mathcal{A}_{ar,f_{\mathsf{b}}}}(\mathfrak{q}, \psi)$. To this end, we denote the formula

$$(\mathsf{init}_{v_c,v_d} \wedge \mathsf{step}_{v_c,v_d,v_m} \wedge \mathsf{continuation}_{v_c,v_d,v_m} \wedge \mathsf{termination}_{v_c,v_d,v_m})$$

by $\mathsf{hyp}_{v_c,v_d,v_m}$. We begin by showing that $(\mathcal{A}_{ar,f_{\mathsf{b}}}, [v_c, v_d, v_m \to c, d, m]\sigma) \models \mathsf{hyp}_{v_c,v_d,v_m}$ if and only if the following statements hold:

(1) For all $i \leq m$, the state $\sigma_i = (\mathcal{S}^{pr}_{\mathcal{A}_{ar}}(\mathfrak{p}))^{(i)}(\sigma)$ is defined.
(2) For all $i \leq m$, the $(i+1)$st block of the sequence encoded by (c, d) contains $\sigma_i(v_0), \ldots, \sigma_i(v_{n-1})$.
(3) For $i < m$, $(\mathcal{A}_{ar,f_{\mathsf{b}}}, \sigma_i) \models \beta$.
(4) $(\mathcal{A}_{ar,f_{\mathsf{b}}}, \sigma_m) \models (\neg\beta)$.

First, suppose that $(\mathcal{A}_{ar,f_{\mathsf{b}}}, [v_c, v_d, v_m \to c, d, m]\sigma) \models \mathsf{hyp}_{v_c,v_d,v_m}$. Since $\sigma_0 = \sigma$, the first block of the sequence encoded by (c, d) contains the elements $\sigma_0(v_0), \ldots, \sigma_0(v_{n-1})$ because $(\mathcal{A}_{ar,f_{\mathsf{b}}}, [v_c, v_d, v_m \to c, d, m]\sigma) \models \mathsf{init}_{v_c,v_d}$. This shows that the first two statements hold for $i = 0$. It now follows that these statements hold for all $i \leq m$ since $(\mathcal{A}_{ar,f_{\mathsf{b}}}, [v_c, v_d, v_m \to c, d, m]\sigma) \models \mathsf{step}_{v_c,v_d,v_m}$.

Since $(\mathcal{A}_{ar,f_{\mathsf{b}}}, [v_c, v_d, v_m \to c, d, m]\sigma) \models \mathsf{continuation}_{v_c,v_d,v_m}$, it follows that for $i < m$ if $a^i_0, \ldots, a^i_{n-1}$ are the entries of the $(i+1)$st block of length n of the sequence encoded by (c, d), we have

$$(\mathcal{A}_{ar,f_{\mathsf{b}}}, [v_0, \ldots, v_{n-1} \to a^i_0, \ldots, a^i_{n-1}][v_c, v_d, v_m \to c, d, m]\sigma) \models \beta.$$

Note that by the second statement, $a^i_j = \sigma_i(v_j)$ for $0 \leq j \leq n-1$. Let $\hat{\sigma}_i = [v_0, \ldots, v_{n-1} \to a^i_0, \ldots, a^i_{n-1}][v_c, v_d, v_m \to c, d, m]\sigma$. We claim that $\hat{\sigma}_i(z) = \sigma_i(z)$ for every variable z that occurs in β. This is clear if $z \in \{v_0, \ldots, v_{n-1}\}$. Otherwise, $\hat{\sigma}_i(z) = \sigma(z)$, since v_c, v_d, v_m do not occur in β and by Theorem 6.3.5, $\sigma_i(z) = \sigma(z)$ since z does not occur in $\mathfrak{p}$. This allows us to conclude that for $i < m$, $(\mathcal{A}_{ar,f_{\mathsf{b}}}, \sigma_i) \models \beta$.

A similar argument, using the fact that $(\mathcal{A}_{ar,f_{\mathsf{b}}}, \sigma) \models \mathsf{termination}_{v_c,v_d,v_m}$ shows that $(\mathcal{A}_{ar}, \sigma_m) \models (\neg\beta)$.

We leave to the reader to prove that conditions (1) to (4) imply

$$(\mathcal{A}_{ar,f_{\mathsf{b}}}, [v_c, v_d, v_m \to c, d, m]\sigma) \models \mathsf{hyp}_{v_c,v_d,v_m}.$$

Suppose that $(\mathcal{A}_{ar,f_{\mathsf{b}}}, \sigma) \models \mathsf{precond}$. To show that $\sigma \in \mathsf{WLP}_{\mathcal{A}_{ar}}(\mathfrak{q}, \psi)$, we assume that $\sigma' = \mathcal{S}^{pr}_{\mathcal{A}_{ar}}(\mathfrak{q})(\sigma)$ is defined. We prove that $(\mathcal{A}_{ar}, \sigma') \models \psi$. Define $\sigma_i = (\mathcal{S}^{pr}_{\mathcal{A}_{ar}}(\mathfrak{p}))^{(i)}(\sigma)$ and let m be least number such that $(\mathcal{A}_{ar}, \sigma_m) \models (\neg\beta)$. Clearly, $\sigma_0 = \sigma$, $\sigma_m = \sigma'$, and $(\mathcal{A}_{ar}, \sigma_i) \models \beta$ for $0 \leq i < m$. By Theorem 4.7.7, there is a pair (c, d) that encodes the sequence

$$(\sigma_0(v_0), \ldots, \sigma_0(v_{n-1}), \sigma_1(v_0), \ldots, \sigma_1(v_{n-1}), \ldots, \sigma_m(v_0), \ldots, \sigma_m(v_{n-1})).$$

The properties (1) to (4) mentioned above hold and therefore,

$$(\mathcal{A}_{ar,f_{\mathsf{b}}}, [v_c, v_d, v_m \to c, d, m]\sigma) \models \mathsf{hyp}_{v_c,v_d,v_m}.$$

Since $(\mathcal{A}_{ar,f_{\mathsf{b}}}, \sigma) \models \mathsf{precond}$, it follows that

$$(\mathcal{A}_{ar,f_{\mathsf{b}}}, [v_c, v_d, v_m \to c, d, m]\sigma) \models \mathsf{verification}_{v_c,v_d,v_m}.$$

Thus,

$$(\mathcal{A}_{ar}, [v_0, \ldots, v_{n-1} \to \sigma_m(v_0), \ldots, \sigma_m(v_{n-1})][v_c, v_d, v_m \to c, d, m]\sigma) \models \psi.$$

By an argument similar to one given earlier in the proof, it follows that $(\mathcal{A}_{ar}, \sigma') \models \psi$.

Conversely, suppose that $\sigma \in \mathsf{WLP}_{\mathcal{A}_{ar}}(\mathfrak{q}, \psi)$. We need to prove that $(\mathcal{A}_{ar,f_{\mathsf{b}}}, \sigma) \models \mathsf{precond}$. In other words, we need to show that for all $c, d, m \in \mathbf{N}$, if

$$(\mathcal{A}_{ar,f_{\mathsf{b}}}, [v_c, v_d, v_m \to c, d, m]\sigma) \models \mathsf{hyp}_{v_c,v_d,v_m},$$

then

$$(\mathcal{A}_{ar,f_{\mathsf{b}}}, [v_c, v_d, v_m \to c, d, m]\sigma) \models \mathsf{verification}_{v_c,v_d,v_m}.$$

By the assumption made above, conditions (1) to (4) are satisfied and so the state σ_m is $\mathcal{S}^{pr}_{\mathcal{A}_{ar}}(\mathfrak{q})(\sigma)$. Since $\sigma \in \mathsf{WLP}_{\mathcal{A}_{ar}}(\mathfrak{q}, \psi)$, $(\mathcal{A}_{ar}, \sigma_m) \models \psi$, so $(\mathcal{A}_{ar,f_{\mathsf{b}}}, [v_c, v_d, v_m \to c, d, m]\sigma) \models \mathsf{verification}_{v_c,v_d,v_m}$.

We have shown that the formula $\mathsf{precond}$ expresses the set of $\mathcal{A}_{ar}$-states $\mathsf{WLP}_{\mathcal{A}_{ar}}(\mathfrak{q}, \psi)$.

Next, we obtain an $\mathcal{L}_{ar}$ formula $\mathsf{precond}'$ from the $\mathcal{L}_{ar,f_{\mathsf{b}}}$-formula $\mathsf{precond}$ by applying the effective method of Supplement 101 of Chapter 4 and using the fact that the Gödel function b is definable in $\mathcal{A}_{ar}$ as shown in Example 4.7.8.

Consequently, the desired assertion $\omega^{\mathcal{A}_{ar}}(\mathfrak{q}, \psi)$ is $\mathsf{precond}'_\star$ because this latest formula is a variant of $\mathsf{precond}'$ and, therefore, it expresses the same set of $\mathcal{A}$-states.

Observe that the argument of the proof provides an effective construction of the assertion $\omega^{\mathcal{A}_{ar}}(\mathfrak{q}, \psi)$. □

Theorem 6.6.18. *The structure $\mathcal{A}_{ar}$ is effectively expressive.*

Proof. We will prove that there is an effective construction that for every $\mathcal{L}_{ar}$-program $\mathfrak{p}$ and for every $\mathcal{L}_{ar}$-assertion ψ, generates an $\mathcal{L}_{ar}$-assertion $\omega^{\mathcal{A}_{ar}}(\mathfrak{p}, \psi)$ that expresses $\mathsf{WLP}_{\mathcal{A}_{ar}}(\mathfrak{p}, \psi)$. This suffices to show that $\mathcal{A}_{ar}$ is effectively expressive (see Exercise 49). The function $\omega^{\mathcal{A}_{ar}}$ is defined recursively. Since this requires unique readability, we need to formulate this definition separately for atomic programs and programs in general.

For the initial step, suppose that $\mathfrak{p}$ is $v \leftarrow t$ and let ψ be an assertion. In this case, by Theorem 6.5.22, we can define $\omega^{at,\mathcal{A}_{ar}}(\mathfrak{p}, \psi)$ as $(\psi)_{v:=t}$, because $(\psi)_{v:=t}$ is an assertion (since ψ is one).

For the first recursive step, let $\mathfrak{p}$ be the program

$$\textbf{if } \beta \textbf{ then } \mathfrak{p}_0 \textbf{ else } \mathfrak{p}_1 \textbf{ endif}$$

and define $\omega^{at,\mathcal{A}_{ar}}(\mathfrak{p}, \psi)$ as the assertion

$$((\beta \to \omega^{\mathcal{A}_{ar}}(\mathfrak{p}_0, \psi)) \wedge ((\neg\beta) \to \omega^{\mathcal{A}_{ar}}(\mathfrak{p}_1, \psi))).$$

For the next recursive step, suppose that $\mathfrak{p}$ is **while** β **do** $\mathfrak{q}$ **endwhile** and ψ is an assertion. Let $\mathrm{PVAR}(\mathfrak{q}) = \{v_0, \ldots, v_{n-1}\}$ and let ψ' be

$$(\neg(v_0 = z_0 \wedge \cdots \wedge v_{n-1} = z_{n-1} \wedge y_0 = z'_0 \wedge \cdots \wedge y_{n-1} = z'_{n-1})),$$

where $y_0, \ldots, y_{n-1}, z_0, \ldots, z_{n-1}, z'_0, \ldots, z'_{n-1}$ are distinct program variables that do not occur in $\mathfrak{q}$. Let $\alpha = \omega^{\mathcal{A}_{ar}}(\mathfrak{q}, \psi')$. By Lemma 6.6.17, we obtain the assertion $\omega^{at,\mathcal{A}_{ar}}(\mathfrak{p}, \psi)$.

For an atomic program $\mathfrak{p}$, we define $\omega^{\mathcal{A}_{ar}}(\mathfrak{p}, \psi) = \omega^{at,\mathcal{A}_{ar}}(\mathfrak{p}, \psi)$.

For the last recursive step, let $\mathfrak{p} = \mathfrak{p}_0; \mathfrak{p}_1$, where $\mathfrak{p}_0$ is an atomic program and, again, let ψ be an assertion. Define $\omega^{\mathcal{A}_{ar}}(\mathfrak{p}, \psi)$ as

$$\omega^{at,\mathcal{A}_{ar}}(\mathfrak{p}_0, \omega^{\mathcal{A}_{ar}}(\mathfrak{p}_1, \psi)),$$

which is the needed assertion by Theorem 6.5.24.

Since the constructions in the preliminary steps described by the theorems and the lemma quoted in this proof are effective, it follows that the construction of the assertion $\omega^{\mathcal{A}_{ar}}(\mathfrak{p}, \psi)$ is effective. $\square$

Our goal now is to show the undecidability of the Hoare partial correctness theory of Presburger arithmetic. In fact, we will show the stronger result that this set is not even semidecidable. As we have shown in Exercise 185 of Chapter 4, the first-order theory of Presburger arithmetic is decidable, which makes this case especially interesting in that it provides an example where the membership problem of the Hoare partial correctness theory is strictly more difficult that the membership problem of the first-order theory of the corresponding structure. The underlying reason for the greater difficulty of the membership problem of the Hoare partial correctness theory is that certain operations can be defined by programs in a structure but not by first-order formulas. In particular, multiplication can be defined through $\mathcal{L}_{pra}$-programs, as we will show, but it cannot be defined by first-order formulas in $\mathcal{A}_{pra}$.

We will prove that a special fragment of the Hoare partial correctness theory of Presburger arithmetic, namely, $\mathsf{HPT}^{\uparrow}_{\mathcal{L}_{pra}}(\mathcal{A}_{pra})$, is undecidable. This is achieved by reducing the Post Correspondence Problem to this fragment. In order to construct this reduction, we will introduce a numerical encoding for words over finite alphabets.

Definition 6.6.19. Let $V = \{a_0, \ldots, a_{n-1}\}$ be an alphabet. The *coding function* is the mapping $k_V : V^* \longrightarrow \mathbf{N}$ defined by

$$k_V(a_{i_{l-1}} \cdots a_{i_0}) = (i_{l-1} + 1)n^{l-1} + \cdots + (i_1 + 1)n + (i_0 + 1).$$

$\square$

Note that $k_V(\lambda) = 0$. The value of $k_V(w)$ depends on the order chosen for the symbols of the alphabet V.

Example 6.6.20. Let V be the three-symbol alphabet $V = \{a_0, a_1, a_2\}$. We have $k_V(a_2a_2a_0a_1) = 3 \cdot 3^3 + 3 \cdot 3^2 + 1 \cdot 3 + 2 = 113$. $\square$

Note that for $w, w' \in V^*$, we have

$$k_V(ww') = k_V(w)n^{|w'|} + k_V(w'). \tag{6.7}$$

Theorem 6.6.21. *The mapping k_V introduced in Definition* 6.6.19 *is a bijection.*

Proof. We show by strong induction on m that there is at most one word $w \in V^*$ such that $m = k_V(w)$. Let $\text{trim}(w) = w'$ and $\text{last}(w) = a$ if $w = w'a$ for some $a \in V$ and $\text{trim}(\lambda) = \text{last}(\lambda) = \lambda$.

Note that $k_V(w) = nk_V(\text{trim}(w)) + k_V(\text{last}(w))$ and if $w \neq \lambda$, then $k_V(\text{trim}(w)) < k_V(w)$.

For the basis step, $m = 0$, the statement is immediate because $w \neq \lambda$ implies $k_V(w) \geq 1$. Suppose the statement is true for numbers less than $m > 0$ and that $m = k_V(w_0) = k_V(w_1)$. Since $m > 0$, we have $w_0, w_1 \neq \lambda$. By a previous observation, we have

$$nk_V(\text{trim}(w_0)) + k_V(\text{last}(w_0)) = nk_V(\text{trim}(w_1)) + k_V(\text{last}(w_1)),$$

which implies

$$|k_V(\text{last}(w_0)) - k_V(\text{last}(w_1))| = n|k_V(\text{trim}(w_1)) - k_V(\text{trim}(w_0))|.$$

Therefore, $k_V(\text{last}(w_0)) = k_V(\text{last}(w_1))$ since otherwise we would have a number between 1 and $n-1$ in the left side of the last equality which cannot be a multiple of n. Thus, w_0 and w_1 have the same last symbols and we have $k_V(\text{trim}(w_0)) = k_V(\text{trim}(w_1)) < m$. By the inductive hypothesis, $\text{trim}(w_0) = \text{trim}(w_1)$ and this allows us to conclude that $w_0 = w_1$.

To prove the surjectivity of k_V, we show by strong induction on m, that there is a word $w \in V^*$ such that $k_V(w) = m$. The basis step, $m = 0$ is immediate. Suppose the statement holds for numbers less than m. There are two numbers $q, k \in \mathbf{N}$ such that $0 \leq q < m$ and $1 \leq k \leq n$ such that $m = nq + k$. By the inductive hypothesis, there is a word w' such that $k_V(w') = q$. Therefore, $k_V(w'a_{k-1}) = m$. $\square$

Example 6.6.22. The function $\text{len}_V : \mathbf{N} \longrightarrow \mathbf{N}$ given by $\text{len}_V(p) = q$ if $p = k_V(w)$ and $q = |w|$ for some $w \in V^*$ is computable in the structure $\mathcal{A}_{ar}$ by the $\mathcal{L}_{ar}$-program $\mathfrak{p}_{len}$ and the sequence of variables (x, y), where $\mathfrak{p}_{len}$ is:

```
$y \leftarrow 0$;
$s \leftarrow s(0)$;
$t \leftarrow s(0)$;
while ($s \leq x$)
      do  $t \leftarrow t \cdot s^n(0)$;
          $s \leftarrow s + t$;
          $y \leftarrow s(y)$
endwhile
```

The program is based on the idea that the length of the word whose code is x is the least number k such that $1+n+\cdots+n^k > x$. $\square$

Example 6.6.23. The function $\mathrm{conc}_V : \mathbf{N}^2 \longrightarrow \mathbf{N}$, where $\mathrm{conc}_V(p,p') = q$ if $p = k_V(w)$, $p' = k_V(w')$, and $k_V(ww') = q$, is computable in $\mathcal{A}_{arxl_V}$ by the program $\mathfrak{p}_{conc}$ and the variables x, x', y, where $\mathfrak{p}_{conc}$ is the $\mathcal{L}_{arxl_V}$-program:

$$y \leftarrow \exp(s^n(0), \mathsf{len}_V(x')) \cdot x + x'$$

Here $\mathcal{L}_{arxl_V}$ is the language $\mathcal{L}_{ar} \cup \{\exp, \mathsf{len}_V\}$ and $\mathcal{A}_{arxl_V}$ is the expansion of $\mathcal{A}_{ar}$ to $\mathcal{L}_{arxl_V}$, where the function symbols exp and len_V have the interpretations of exponentiation and len_V, respectively. $\square$

In the sequel, we will use the following sequence of six first-order languages, whose definitions are included here for convenience:

$$\begin{aligned}
\mathcal{L}_{pra} &= \{=, <, 0, s, +\} \\
\mathcal{L}_{ar} &= \mathcal{L}_{pra} \cup \{\cdot\} \\
\mathcal{L}_{arx} &= \mathcal{L}_{ar} \cup \{\exp\} \\
\mathcal{L}_{arxl_V} &= \mathcal{L}_{arx} \cup \{\mathsf{len}_V\} \\
\mathcal{L}_{arxl_V c_V} &= \mathcal{L}_{arxl_V} \cup \{\mathsf{conc}_V\} \\
\mathcal{L}_{arxl_V c_V \mathsf{b}} &= \mathcal{L}_{arxl_V c_V} \cup \{f_{\mathsf{b}}\}.
\end{aligned}$$

Also, we will use the structures $\mathcal{A}_{pra}, \mathcal{A}_{ar}, \mathcal{A}_{arx}, \mathcal{A}_{arxl_V}, \mathcal{A}_{arxl_V c_V}$, and $\mathcal{A}_{arxl_V c_V \mathsf{b}}$. Here, the last five structures are extensions of $\mathcal{A}_{pra}$ to the first-order languages with the same index, respectively. For $\mathcal{A}_{arxl_V c_V}$,

$$\mathsf{conc}_V^{\mathcal{A}_{arxl_V c_V}}(p,p') = \mathrm{conc}_V(p,p').$$

Also, in the last structure,

$$\mathsf{f}_{\mathsf{b}}^{\mathcal{A}_{arxl_V c_V \mathsf{b}}} = \mathsf{b},$$

where b is the Gödel function.

We will use

```
for $k = t$ to $u$ do
    $\mathfrak{p}$
endfor
```

as an abbreviation for

```
$k = t$
while $k \leq u$ do
    $\mathfrak{p}$;
    $k \leftarrow s(k)$
endwhile
```

Theorem 6.6.24. *Let V be the alphabet $\{a, b\}$. For every V-instance $\Im$ of the PCP, an $\mathcal{L}_{arxl_V c_V \mathsf{b}}$-program $\mathfrak{p}_\Im$ can be constructed effectively from $\Im$ such that $\Im$ has a solution if and only if*

$$\mathcal{A}_{arxl_V c_V \mathsf{b}} \not\models \{\mathit{true}_\star^{\mathcal{L}_{arxl_V c_V b}}\}\mathfrak{p}_\Im\{\mathit{false}_\star^{\mathcal{L}_{arxl_V c_V b}}\}.$$

Proof. Let $\Im = (V, (q_0, \ldots, q_{n-1}), (r_0, \ldots, r_{n-1}))$ be a V-instance of the PCP. The idea of the program $\mathfrak{p}_\Im$, shown in Figure 6.2, is to check the subsequence consisting of those entries of the finite sequence of indices $(\mathsf{b}(c, d, 0), \ldots, \mathsf{b}(c, d, k-1))$ which are less than n, to see if it is a solution to $\Im$. Note that the Gödel function b is being used to encode finite sequences. This is made possible by Theorem 4.7.7 which says that every finite sequence can be obtained in this way for appropriate values of c, d, k.

The variable *top* gives the code $k_V(q_{h_0} \cdots q_{h_{p-1}})$, while *bottom* contains the code $k_V(r_{h_0} \cdots r_{h_{p-1}})$, where the pair (c, d) defines the sequence of subscripts $(i_0, \ldots, i_{k-1})$ via the Gödel function b; in other words, $\mathsf{b}(c, d, j) = i_j$ for $0 \leq j < k$ and $(h_0, \ldots, h_{p-1})$ is the subsequence of the sequence $(i_0, \ldots, i_{k-1})$ which contains those entries less than n.

Suppose that $\Im$ has a solution, $(i_0, \ldots, i_{k-1})$. By Theorem 4.7.7, there are $c, d \in \mathbf{N}$ such that for every $j, 0 \leq j < k$, $\mathsf{b}(c, d, j) = i_j$. Note that when $\mathfrak{p}_\Im$ is run in an input state where c, d and k are initialized to these values, it halts, while if $\Im$ has no solution, then no matter what the input state is, $\mathfrak{p}_\Im$ does not halt when started in that state. □

```
$top \leftarrow 0$;
$bottom \leftarrow 0$;
for i = 0 to k  do
    if $(f_{\mathsf{b}}(c,d,i) = 0)$
        then $top \leftarrow \mathsf{conc}_V(top, s^{k_V(q_0)}(0))$;
             $bottom \leftarrow \mathsf{conc}_V(bottom, s^{k_V(r_0)}(0))$
        else if$(f_{\mathsf{b}}(c,d,i) = 1)$
             then $top \leftarrow \mathsf{conc}_V(top, s^{k_V(q_1)}(0))$;
                  $bottom \leftarrow \mathsf{conc}_V(bottom, s^{k_V(r_1)}(0))$
             else ...
           ⋮
               else if$(f_{\mathsf{b}}(c,d,i) = n-1)$
                    then $top \leftarrow \mathsf{conc}_V(top, s^{k_V(q_{n-1})}(0))$;
                         $bottom \leftarrow \mathsf{conc}_V(bottom, s^{k_V(r_{n-1})}(0))$
                    else $top \leftarrow top$;
                         $bottom \leftarrow bottom$
                 endif
               ⋮
           endif
    endif
endfor;
if $((top \neq bottom) \vee (top = 0))$
    then while $\mathsf{true}_{\star}^{\mathcal{L}_{pra}}$ do
              $top \leftarrow top$
           endwhile
    else $top \leftarrow top$
 endif
```

Fig. 6.2. The program $\mathfrak{p}_{\mathfrak{S}}$.

Theorem 6.6.25. *The set of triples $\mathit{HPT}^{\uparrow}_{\mathcal{L}_{arxl_V c_V \mathsf{b}}}(\mathcal{A}_{arxl_V c_V \mathsf{b}})$, which is a subset of the partial-correctness Hoare theory of the structure $\mathcal{A}_{arxl_V c_V \mathsf{b}}$ is undecidable. Further, the full partial-correctness Hoare theory of the same structure is undecidable.*

Proof. By Theorem 6.6.24, the decidability of the subset mentioned in the theorem would entail the decidability of the PCP for the alphabet $\{a, b\}$, which contradicts Theorem 4.14.10. The second part of the statement follows immediately. □

Theorem 6.6.26. *Let $\mathcal{L}$ be a decidable first-order language, $\mathcal{A}$ be an $\mathcal{L}$-structure and let $g : |\mathcal{A}|^n \longrightarrow |\mathcal{A}|$ be computable by an $\mathcal{L}$-program $\mathfrak{r}_g$ with a sequence $(y_0, \ldots, y_n)$. Let f_g be an n-ary function symbol* *not in $\mathcal{L}$ and define $\mathcal{L}' = \mathcal{L} \cup \{f_g\}$. Define the expansion $\mathcal{A}'$ of $\mathcal{A}$ to $\mathcal{L}'$ by $f_g^{\mathcal{A}'} = g$.*

The fragment $\mathit{HPT}^{\uparrow}_{\mathcal{L}'}(\mathcal{A}')$ is m-reducible to $\mathit{HPT}^{\uparrow}_{\mathcal{L}}(\mathcal{A})$, i.e.,

$$\mathit{HPT}^{\uparrow}_{\mathcal{L}'}(\mathcal{A}') \leq_m \mathit{HPT}^{\uparrow}_{\mathcal{L}}(\mathcal{A}).$$

Proof. The computability of the function $\mathsf{q} : \mathtt{WHILE}_{\mathcal{L}'} \longrightarrow \mathtt{WHILE}_{\mathcal{L}}$ shown in Theorem 6.4.16 implies that the function $F : \mathsf{HPT}_{\mathcal{L}'} \longrightarrow \mathsf{HPT}_{\mathcal{L}}$ given by $F(\{\mathsf{true}_\star^{\mathcal{L}'}\}\mathfrak{p}\{\mathsf{false}_\star^{\mathcal{L}'}\}) = \{\mathsf{true}_\star^{\mathcal{L}}\}\mathsf{q}(\mathfrak{p})\{\mathsf{false}_\star^{\mathcal{L}}\}$ is computable. This function is the needed m-reduction. □

Theorem 6.6.27. *The fragment $\mathit{HPT}^{\uparrow}_{\mathcal{L}_{pra}}(\mathcal{A}_{pra})$ of the partial-correctness Hoare theory of Presburger arithmetic is undecidable.*

Proof. This result follows from a sequence of reductions which we outline below. In Example 6.4.5, we have shown that the function b is computable in the structure $\mathcal{A}_{ar}$ and therefore is computable in $\mathcal{A}_{arxl_V c_V}$. By Theorem 6.6.26, we have

$$\mathsf{HPT}^{\uparrow}_{\mathcal{L}_{arxl_V c_V \mathsf{b}}}(\mathcal{A}_{arxl_V c_V \mathsf{b}}) \leq_m \mathsf{HPT}^{\uparrow}_{\mathcal{L}_{arxl_V c_V}}(\mathcal{A}_{arxl_V c_V}).$$

Next, the function conc_V is computable in the structure $\mathcal{A}_{arxl_V}$ as we have shown in Example 6.6.23. This implies the reduction

$$\mathsf{HPT}^{\uparrow}_{\mathcal{L}_{arxl_V c_V}}(\mathcal{A}_{arxl_V c_V}) \leq_m \mathsf{HPT}^{\uparrow}_{\mathcal{L}_{arxl_V}}(\mathcal{A}_{arxl_V}).$$

To obtain the next reduction, recall that the function len_V is computable in the structure $\mathcal{A}_{ar}$ as we established in Example 6.6.22 and therefore in the structure $\mathcal{A}_{arx}$. This gives us the reduction

$$\mathsf{HPT}^{\uparrow}_{\mathcal{L}_{arxl_V}}(\mathcal{A}_{arxl_V}) \leq_m \mathsf{HPT}^{\uparrow}_{\mathcal{L}_{arx}}(\mathcal{A}_{arx}).$$

The exponentiation function is computable in $\mathcal{A}_{ar}$ as shown in Example 6.4.4, so $\mathsf{HPT}^{\uparrow}_{\mathcal{A}_{arx}}(\mathcal{A}_{arx}) \leq_m \mathsf{HPT}^{\uparrow}_{\mathcal{L}_{ar}}(\mathcal{A}_{ar})$. Finally, the computability of multiplication in the structure $\mathcal{A}_{pra}$ shown in Example 6.4.3 gives the last reduction $\mathsf{HPT}^{\uparrow}_{\mathcal{L}_{ar}}(\mathcal{A}_{ar}) \leq_m \mathsf{HPT}^{\uparrow}_{\mathcal{L}_{pra}}(\mathcal{A}_{pra})$. Putting together these reductions yields $\mathsf{HPT}^{\uparrow}_{\mathcal{L}_{arxl_V c_V \mathsf{b}}}(\mathcal{A}_{arxl_V c_V \mathsf{b}}) \leq_m \mathsf{HPT}^{\uparrow}_{\mathcal{L}_{pra}}(\mathcal{A}_{pra})$. The undecidability of $\mathsf{HPT}^{\uparrow}_{\mathcal{L}_{arxl_V c_V \mathsf{b}}}(\mathcal{A}_{arxl_V c_V \mathsf{b}})$ shown in Theorem 6.6.25 together with Theorem 1.4.9, gives the desired conclusion. □

Corollary 6.6.28. *The partial-correctness theory of Presburger arithmetic $\mathsf{HPT}_{\mathcal{L}_{pra}}(\mathcal{A}_{pra})$ is undecidable.*

Proof. This follows from Theorem 6.6.27. □

Theorem 6.6.29. *Let $\mathcal{L}$ be a first-order language and $\mathcal{A}$ be an $\mathcal{L}$-structure, If $\mathrm{Th}^{\mathcal{L}}(\mathcal{A})$ is decidable, then the complement of $\mathsf{HPT}_{\mathcal{L}}(\mathcal{A})$ is semidecidable.*

Proof. By Theorem 6.5.20, we have $\mathcal{A} \not\models \mathcal{H}$ if and only if $\mathcal{A} \not\models \Sigma_{\mathcal{H}}$. For a given $\mathcal{H}$, let $\theta_0, \ldots, \theta_n, \ldots$ be a listing of the formulas of $\Sigma_{\mathcal{H}}$. As noted after Theorem 6.5.20, such a listing can be effectively obtained given $\mathcal{H}$. To determine if $\mathcal{H} \not\in \mathsf{HPT}_{\mathcal{L}}(\mathcal{A})$, we seek to find an index i such that $\theta_i \not\in \mathrm{Th}^{\mathcal{L}}(\mathcal{A})$. The membership of θ_i in $\mathrm{Th}^{\mathcal{L}}(\mathcal{A})$ can be tested effectively under the hypotheses of the theorem. If such an i can be found, we may conclude that $\mathcal{H} \not\in \mathsf{HPT}_{\mathcal{L}}(\mathcal{A})$, which shows that the complement of $\mathsf{HPT}_{\mathcal{L}}(\mathcal{A})$ is semidecidable. □

Corollary 6.6.30. *The $\mathcal{L}_{pra}$-partial correctness theory of Presburger arithmetic, $\mathsf{HPT}_{\mathcal{L}_{pra}}(\mathcal{A}_{pra})$ is not semidecidable.*

Proof. We know that $\mathsf{HPT}_{\mathcal{L}_{pra}}(\mathcal{A}_{pra})$ is undecidable (Corollary 6.6.28). By Exercise 185 of Chapter 4, $\mathrm{Th}^{\mathcal{L}_{pra}}(\mathcal{A}_{pra})$ is decidable and therefore, Theorem 6.6.29 implies that the complement of $\mathsf{HPT}_{\mathcal{L}_{pra}}(\mathcal{A}_{pra})$ is semidecidable. By Theorem 1.4.6, $\mathsf{HPT}_{\mathcal{L}_{pra}}(\mathcal{A}_{pra})$ is not semidecidable. □

6.7 A Formal System for Hoare Triples

It would be desirable to have a sound, complete, effectively specified formal system for Hoare triples. Namely, we would like to have, for each first-order language $\mathcal{L}$, a formal system $\mathcal{F}_{\mathcal{L}}$ such that for every set of $\mathcal{L}$-formulas Γ and $\mathcal{L}$-partial correctness Hoare triple $\mathcal{H}$, $\Gamma \models\!\approx \mathcal{H}$ if and only if $\Gamma \vdash_{\mathcal{F}_{\mathcal{L}}} \mathcal{H}$. We would even settle for the existence of such a formal system for the case when Γ would be a theory of the form $\mathrm{Th}^{\mathcal{L}}(\mathcal{A})$ for an $\mathcal{L}$-structure $\mathcal{A}$. Unfortunately, as we show, there are structures for which this is unachievable by any formal system.

Theorem 6.7.1. *There is no effectively specified formal system $\mathcal{F}$ whose set of objects consists of first-order $\mathcal{L}_{pra}$-formulas and*

$\mathcal{L}_{pra}$-partial correctness Hoare triples such that for all $\mathcal{L}_{pra}$-partial correctness triples $\mathcal{H}$, we have $Th^{\mathcal{L}_{pra}}(\mathcal{A}_{pra}) \approx\!\!\models \mathcal{H}$ if and only if $Th^{\mathcal{L}_{pra}}(\mathcal{A}_{pra}) \vdash_{\mathcal{F}} \mathcal{H}$.

Proof. Suppose that a formal system $\mathcal{F}$ with the properties specified in the theorem exists. Then, by Exercise 185 of Chapter 4 and Corollary 1.8.30, the set of objects inferrable in $\mathcal{F}$ from $\mathrm{Th}^{\mathcal{L}_{pra}}(\mathcal{A}_{pra})$ is semidecidable. Consequently, by Exercise 38 of Chapter 1, the set $\{\mathcal{H} \mid \mathrm{Th}^{\mathcal{L}_{pra}}(\mathcal{A}_{pra}) \vdash_{\mathcal{F}} \mathcal{H}\}$ is semidecidable. By our assumption, this would imply that the set $\{\mathcal{H} \mid \mathrm{Th}^{\mathcal{L}_{pra}}(\mathcal{A}_{pra}) \approx\!\!\models \mathcal{H}\}$ is semidecidable. Since this is $\mathsf{HPT}_{\mathcal{L}_{pra}}(\mathcal{A}_{pra})$ by Corollary 6.5.21, we contradict Corollary 6.6.30. □

In view of the previous negative result, the introduction of a formal system that proves Hoare triples may seem futile. However, we will exhibit a sound system that is effectively specified and complete in the sense of Theorem 6.7.1 for certain structures.

Definition 6.7.2. Let $\mathcal{L}$ be a first-order language. The *Hilbert/Hoare formal system $\mathcal{HH}^{\mathcal{L}}$* consists of:

- $\mathrm{FORM}_{\mathcal{L}} \cup \mathsf{HPT}_{\mathcal{L}}$ as set of objects.
- The rule $R^{\mathcal{L}}_{mp}$ inherited from the Hilbert/Frege formal system, supplemented by four rules $R^{\mathcal{L}}_{if}, R^{\mathcal{L}}_{while}, R^{\mathcal{L}}_{comp}$ and $R^{\mathcal{L}}_{imp}$, which are given by:
  - the *if rule*:

$$\frac{\{(\beta \wedge \varphi)\}\mathfrak{p}\{\psi\}, \{((\neg\beta) \wedge \varphi)\}\mathfrak{p}'\{\psi\}}{\{\varphi\}\ \textbf{if}\ \beta\ \textbf{then}\ \mathfrak{p}\ \textbf{else}\ \mathfrak{p}'\ \textbf{endif}\{\psi\}}\, R^{\mathcal{L}}_{if}$$

  - the *while rule*:

$$\frac{\{(\varphi \wedge \beta)\}\mathfrak{p}\{\varphi\}}{\{\varphi\}\ \textbf{while}\ \beta\ \textbf{do}\ \mathfrak{p}\ \textbf{endwhile}\{(\varphi \wedge (\neg\beta))\}}\, R^{\mathcal{L}}_{while}$$

  - the *composition rule*:

$$\frac{\{\varphi\}\mathfrak{p}\{\theta\}, \{\theta\}\mathfrak{p}'\{\psi\}}{\{\varphi\}\mathfrak{p};\mathfrak{p}'\{\psi\}}\, \mathbf{R}^{\mathcal{L}}_{comp}$$

  - the *implication rule*:

$$\frac{(\varphi \to \varphi'), \{\varphi'\}\mathfrak{p}\{\psi'\}, (\psi' \to \psi)}{\{\varphi\}\mathfrak{p}\{\psi\}}\, R^{\mathcal{L}}_{imp}$$

for $\varphi, \psi, \beta, \theta, \varphi', \psi' \in \text{ASSERT}_{\mathcal{L}}$ and $\mathfrak{p}, \mathfrak{p}' \in \texttt{WHILE}_{\mathcal{L}}$.

- The axiom set consists of all axioms of the Hilbert/Frege system $\mathcal{HF}^{\mathcal{L}}$ supplemented by the *assignment axiom set* consisting of the $\mathcal{L}$-partial correctness Hoare triples of the form $\{(\varphi)_{v:=t}\}v{\leftarrow}t\{\varphi\}$ for $\varphi \in \text{ASSERT}_{\mathcal{L}}$, $v \in \text{PVAR}$ and $t \in \text{TERM}_{\mathcal{L}}(\text{PVAR})$.

$\blacksquare$

Lemma 6.7.3. *Let $\mathcal{L}$ be a first-order language. If the $\mathcal{L}$-structure $\mathcal{A}$ is a model of the hypotheses of an instance of the rule $R^{\mathcal{L}}_{if}$, then $\mathcal{A}$ is a model of the conclusion of this instance.*

Proof. Let

$$\frac{\{(\beta \wedge \varphi)\}\mathfrak{p}\{\psi\}, \{((\neg\beta) \wedge \varphi)\}\mathfrak{p}'\{\psi\}}{\{\varphi\}\ \textbf{if}\ \beta\ \textbf{then}\ \mathfrak{p}\ \textbf{else}\ \mathfrak{p}'\ \textbf{endif}\ \{\psi\}}$$

be an instance of the rule $R^{\mathcal{L}}_{if}$. Assume that $(\mathcal{A}, \sigma) \models \varphi$, and that

$$\mathcal{S}^{pr}_{\mathcal{A}}\textbf{(if}\ \beta\ \textbf{then}\ \mathfrak{p}\ \textbf{else}\ \mathfrak{p}'\ \textbf{endif)}(\sigma)$$

is defined. We have either $(\mathcal{A}, \sigma) \models \beta$ or $(\mathcal{A}, \sigma) \models (\neg\beta)$.

In the first case, we have $(\mathcal{A}, \sigma) \models (\beta \wedge \varphi)$ and

$$\mathcal{S}^{pr}_{\mathcal{A}}\textbf{(if}\ \beta\ \textbf{then}\ \mathfrak{p}\ \textbf{else}\ \mathfrak{p}'\ \textbf{endif)}(\sigma) = \mathcal{S}^{pr}_{\mathcal{A}}\textbf{(}\mathfrak{p}\textbf{)}(\sigma),$$

so $\mathcal{S}^{pr}_{\mathcal{A}}\textbf{(}\mathfrak{p}\textbf{)}(\sigma)$ is defined and $(\mathcal{A}, \mathcal{S}^{pr}_{\mathcal{A}}\textbf{(}\mathfrak{p}\textbf{)}(\sigma)) \models \psi$ because $\mathcal{A} \models \{(\beta \wedge \varphi)\}\mathfrak{p}\{\psi\}$. Thus, we have shown that $(\mathcal{A}, \sigma)$ satisfies the conclusion of the rule.

The second case is similar and is left to the reader. Since the above statements hold for every σ, we may conclude that $\mathcal{A}$ is a model of the conclusion of this instance. $\square$

Lemma 6.7.4. *Let $\mathcal{L}$ be a first-order language. If the $\mathcal{L}$-structure $\mathcal{A}$ is a model of the hypotheses of an instance of the rule $R^{\mathcal{L}}_{while}$, then $\mathcal{A}$ is a model of the conclusion of this instance.*

Proof. Consider an instance of the rule $R^{\mathcal{L}}_{while}$:

$$\frac{\{(\varphi \wedge \beta)\}\mathfrak{p}\{\varphi\}}{\{\varphi\}\ \textbf{while}\ \beta\ \textbf{do}\ \mathfrak{p}\ \textbf{endwhile}\{(\varphi \wedge (\neg\beta))\}}$$

Suppose that $(\mathcal{A}, \sigma) \models \varphi$ and $\sigma' = \mathcal{S}^{pr}_{\mathcal{A}}\textbf{(}\mathfrak{q}\textbf{)}(\sigma)$ is defined, where $\mathfrak{q} =$ **while** β **do** $\mathfrak{p}$ **endwhile**. Let σ_j denote the state $(\mathcal{S}^{pr}_{\mathcal{A}}\textbf{(}\mathfrak{p}\textbf{)})^{(j)}(\sigma)$, whenever this state is defined and let k be the least number such that $(\mathcal{A}, \sigma_j) \models \beta$ for $0 \leq j < k$ and $(\mathcal{A}, \sigma_k) \models (\neg\beta)$. Note that $\sigma' = \sigma_k$.

We show by induction on j that $(\mathcal{A}, \sigma_j) \models \varphi$ for $0 \leq j \leq k$. The basis step is immediate by the hypothesis made above. Suppose the statement holds for $i < k$. Then, $(\mathcal{A}, \sigma_i) \models (\varphi \wedge \beta)$ and therefore $(\mathcal{A}, \sigma_{i+1}) \models \varphi$ because $\sigma_{i+1} = \mathcal{S}^{pr}_{\mathcal{A}}(\mathfrak{p})(\sigma_i)$ and the fact that $\mathcal{A}$ is a model of the hypothesis of the rule. Thus, we can conclude that $(\mathcal{A}, \sigma') \models (\varphi \wedge (\neg\beta))$ because, as we stated above, $\sigma' = \sigma_k$. $\square$

Lemma 6.7.5. *Let $\mathcal{L}$ be a first-order language. If the $\mathcal{L}$-structure $\mathcal{A}$ is a model of the hypotheses of an instance of the rule $R^{\mathcal{L}}_{comp}$, then $\mathcal{A}$ is a model of the conclusion of this instance.*

Proof. Let

$$\frac{\{\varphi\}\mathfrak{p}\{\theta\}, \{\theta\}\mathfrak{p}'\{\psi\}}{\{\varphi\}\mathfrak{p};\mathfrak{p}'\{\psi\}}$$

be an instance of the $\mathbf{R}^{\mathcal{L}}_{comp}$ rule.

Let $\mathcal{A}$ be an $\mathcal{L}$-structure such that $\mathcal{A} \models \{\varphi\}\mathfrak{p}\{\theta\}$ and $\mathcal{A} \models \{\theta\}\mathfrak{p}'\{\psi\}$. Suppose that $(\mathcal{A}, \sigma) \models \varphi$ and that $\sigma' = \mathcal{S}^{pr}_{\mathcal{A}}(\mathfrak{p};\mathfrak{p}')(\sigma)$ is defined. This means that $\sigma_1 = \mathcal{S}^{pr}_{\mathcal{A}}(\mathfrak{p})(\sigma)$ is defined and $\sigma' = \mathcal{S}^{pr}_{\mathcal{A}}(\mathfrak{p}')(\sigma_1)$. The first part of the hypothesis of the lemma implies that $(\mathcal{A}, \sigma_1) \models \theta$ and therefore, by the second part of the hypothesis, $(\mathcal{A}, \sigma') \models \psi$. $\square$

Lemma 6.7.6. *Let $\mathcal{L}$ be a first-order language. If the $\mathcal{L}$-structure $\mathcal{A}$ is a model of the hypotheses of an instance of the rule $R^{\mathcal{L}}_{imp}$, then $\mathcal{A}$ is a model of the conclusion of this instance.*

Proof. Consider an instance

$$\frac{(\varphi \rightarrow \varphi'), \{\varphi'\}\mathfrak{p}\{\psi'\}, (\psi' \rightarrow \psi)}{\{\varphi\}\mathfrak{p}\{\psi\}}$$

of the $R^{\mathcal{L}}_{imp}$ rule.

Let $\mathcal{A}$ be an $\mathcal{L}$-structure such that $\mathcal{A} \models (\varphi \rightarrow \varphi')$, $\mathcal{A} \models \{\varphi'\}\mathfrak{p}\{\psi'\}$ and $\mathcal{A} \models (\psi' \rightarrow \psi)$. Suppose that $(\mathcal{A}, \sigma) \models \varphi$ and $\sigma' = \mathcal{S}^{pr}_{\mathcal{A}}(\mathfrak{p})(\sigma)$ is defined. Clearly, we have $(\mathcal{A}, \sigma) \models \varphi'$ and, therefore, $(\mathcal{A}, \sigma') \models \psi'$, which allows us to conclude that $(\mathcal{A}, \sigma') \models \psi$. $\square$

Theorem 6.7.7 (Soundness of $\mathcal{HH}^{\mathcal{L}}$). *Let $\mathcal{L}$ be a first-order language and let Γ be a set of $\mathcal{L}$-formulas. If $\Gamma \vdash_{\mathcal{HH}^{\mathcal{L}}} \theta$, then $\Gamma \approx\!\!\!\models \theta$.*

Proof. The argument is by induction on the theorems of $\mathcal{HH}^{\mathcal{L}}_{\Gamma}$. If θ is a formula of Γ, the conclusion is immediate. If θ is an axiom of $\mathcal{HF}^{\mathcal{L}}$, the conclusion follows from the logical validity of θ established in the proof of Theorem 5.2.2. If θ is an assignment axiom, then the result follows from Theorem 6.5.9.

To prove the inductive steps, assume that $\Gamma \approx\!\!\!\models \theta_i$ for $0 \leq i \leq n-1$, where

$$\frac{\theta_0, \ldots, \theta_{n-1}}{\theta}\, R$$

is an instance of a rule R of $\mathcal{HH}^{\mathcal{L}}_{\Gamma}$. Then, if R is not $R^{\mathcal{L}}_{mp}$, by Lemmas 6.7.3 to 6.7.6, $\Gamma \approx\!\!\!\models \theta$. We leave to the reader the case when R is $R^{\mathcal{L}}_{mp}$. □

Theorem 6.7.8. *Let $\mathcal{L}$ be a first-order language. The rule given by:*

$$\frac{\{\varphi_0\}\mathfrak{p}_0\{\psi\}, \{\varphi_1\}\mathfrak{p}_1\{\psi\}}{\{((\beta \rightarrow \varphi_0) \wedge ((\neg\beta) \rightarrow \varphi_1))\} \textbf{ if } \beta \textbf{ then } \mathfrak{p}_0 \textbf{ else } \mathfrak{p}_1 \textbf{ endif}\{\psi\}}\, R^{call}_{altif}$$

is a derived rule of the Hilbert/Hoare formal system $\mathcal{HH}^{\mathcal{L}}$, for $\varphi_0, \varphi_1, \psi \in ASSERT_{\mathcal{L}}$ and $\mathfrak{p}_0, \mathfrak{p}_1 \in \mathtt{WHILE}_{\mathcal{L}}$.

Proof. Let $\varphi' = ((\beta \rightarrow \varphi_0) \wedge ((\neg\beta) \rightarrow \varphi_1))$. Since $((\varphi' \wedge \beta) \rightarrow \varphi_0)$ and $((\varphi' \wedge (\neg\beta)) \rightarrow \varphi_1)$ are tautologies, by the completeness of the Hilbert/Frege formal system, they are theorems of the system, and, therefore, they are theorems of the Hilbert/Hoare system. By two applications of the implication rule, we obtain the Hoare triples $\{(\varphi' \wedge \beta)\}\mathfrak{p}_0\{\psi\}$ and $\{(\varphi' \wedge (\neg\beta))\}\mathfrak{p}_1\{\psi\}$. Finally, by an application of the if rule, we obtain the triple $\{\varphi'\}$ **if** β **then** $\mathfrak{p}_0$ **else** $\mathfrak{p}_1$ **endif**$\{\psi\}$, which concludes the argument. □

Theorem 6.7.9. *Let $\mathcal{A}$ be an $\mathcal{L}$-structure and let θ be an object of the formal system $\mathcal{HH}^{\mathcal{L}}$. The following three statements are equivalent:*

(1) $\mathcal{A} \models \theta$;
(2) $Th^{\mathcal{L}}(\mathcal{A}) \models \theta$;
(3) $Th^{\mathcal{L}}(\mathcal{A}) \approx\!\!\!\models \theta$.

Proof. We proved the equivalence of the statements for $\theta \in \mathrm{FORM}_{\mathcal{L}}$ in Theorem 4.13.28. Therefore, we can assume that θ is a $\mathcal{L}$-partial correctness triple $\mathcal{H}$. We shall prove the implications (1) $\Rightarrow$ (3) $\Rightarrow$ (2) $\Rightarrow$ (1).

(1) $\Rightarrow$ (3): Suppose that $\mathcal{A} \models \mathcal{H}$. By Theorem 6.5.20, $\mathcal{A} \models \Sigma_{\mathcal{H}}$, which implies that $\Sigma_{\mathcal{H}} \subseteq \mathrm{Th}^{\mathcal{L}}(\mathcal{A})$. Let $\mathcal{B}$ be a model of $\mathrm{Th}^{\mathcal{L}}(\mathcal{A})$. We have $\mathcal{B} \models \Sigma_{\mathcal{H}}$, which implies $\mathcal{B} \models \mathcal{H}$, again by Theorem 6.5.20. Thus, (3) holds.

(3) $\Rightarrow$ (2): Suppose that $\mathrm{Th}^{\mathcal{L}}(\mathcal{A}) \approx\!\!\!\!\models \mathcal{H}$. If $(\mathcal{B},\sigma) \models \mathrm{Th}^{\mathcal{L}}(\mathcal{A})$, then $\mathcal{B} \models \mathrm{Th}^{\mathcal{L}}(\mathcal{A})$ because $\mathrm{Th}^{\mathcal{L}}(\mathcal{A})$ consists of sentences. By (3), $\mathcal{B} \models \mathcal{H}$, so $(\mathcal{B},\sigma) \models \mathcal{H}$. Thus, (2) holds.

(2) $\Rightarrow$ (1): Suppose that $\mathrm{Th}^{\mathcal{L}}(\mathcal{A}) \models \mathcal{H}$. For every $\mathcal{A}$-state σ, we have $(\mathcal{A},\sigma) \models \mathrm{Th}^{\mathcal{L}}(\mathcal{A})$, which implies $(\mathcal{A},\sigma) \models \mathcal{H}$. Therefore, we obtain statement (1), that is $\mathcal{A} \models \mathcal{H}$. □

Theorem 6.7.10 (Relative Completeness of $\mathcal{HH}^{\mathcal{L}}$). *Let $\mathcal{L}$ be a first-order language. If $\mathcal{A}$ is an expressive $\mathcal{L}$-structure and $Th^{\mathcal{L}}(\mathcal{A}) \approx\!\!\!\!\models \theta$, then $Th^{\mathcal{L}}(\mathcal{A}) \vdash_{\mathcal{HH}^{\mathcal{L}}} \theta$, for every object θ of $\mathcal{HH}^{\mathcal{L}}$.*

Proof. Suppose $\theta \in \mathrm{FORM}_{\mathcal{L}}$. Then, by Theorem 4.13.28, $\mathrm{Th}^{\mathcal{L}}(\mathcal{A}) \models \theta$. By the completeness of the Hilbert/Frege system, we have $\mathrm{Th}^{\mathcal{L}}(\mathcal{A}) \vdash_{\mathcal{HF}^{\mathcal{L}}} \theta$. Since the axioms and inference rules of $\mathcal{HF}^{\mathcal{L}}$ exist also in the Hilbert/Hoare system, it follows that $\mathrm{Th}^{\mathcal{L}}(\mathcal{A}) \vdash_{\mathcal{HH}^{\mathcal{L}}} \theta$.

Now suppose that $\theta = \{\varphi\}\mathfrak{p}\{\psi\}$. The argument is by induction on the $\mathcal{L}$-program $\mathfrak{p}$.

For the basis step, suppose that $\mathfrak{p} = v{\leftarrow}t$. By Corollary 6.5.21, we have $\mathcal{A} \models \{\varphi\}\mathfrak{p}\{\psi\}$. By Theorem 6.5.22, $(\psi)_{v:=t}$ expresses $\mathsf{WLP}_{\mathcal{A}}(v{\leftarrow}t,\psi)$ and thus, by Theorem 6.6.7, $\mathcal{A} \models (\varphi \to (\psi)_{v:=t})$. Another application of Theorem 4.13.28 yields $\mathrm{Th}^{\mathcal{L}}(\mathcal{A}) \models (\varphi \to (\psi)_{v:=t})$. Consequently, $\mathrm{Th}^{\mathcal{L}}(\mathcal{A}) \vdash_{\mathcal{HH}^{\mathcal{L}}} (\varphi \to (\psi)_{v:=t})$. Since $\{(\psi)_{v:=t}\}v{\leftarrow}t\{\psi\}$ is an assignment axiom of $\mathcal{HH}^{\mathcal{L}}$, an application of the implication rule $R^{\mathcal{L}}_{imp}$ shows that $\mathrm{Th}^{\mathcal{L}}(\mathcal{A}) \vdash_{\mathcal{HH}^{\mathcal{L}}} \{\varphi\}v{\leftarrow}t\{\psi\}$.

The first inductive step involves programs of the form

$$\mathfrak{p} = \textbf{ if } \beta \textbf{ then } \mathfrak{p}_0 \textbf{ else } \mathfrak{p}_1 \textbf{ endif},$$

where the inductive hypothesis holds for $\mathfrak{p}_0$ and $\mathfrak{p}_1$. Note that $\mathcal{A} \models \theta$ by Corollary 6.5.21. Then, by Theorem 6.5.10, we have $\mathcal{A} \models \{(\beta \wedge \varphi)\}\mathfrak{p}_0\{\psi\}$ and $\mathcal{A} \models \{((\neg\beta) \wedge \varphi)\}\mathfrak{p}_1\{\psi\}$. As a result, we obtain $\mathrm{Th}^{\mathcal{L}}(\mathcal{A}) \approx \{(\beta \wedge \varphi)\}\mathfrak{p}_0\{\psi\}$ and $\mathrm{Th}^{\mathcal{L}}(\mathcal{A}) \approx \{((\neg\beta) \wedge \varphi)\}\mathfrak{p}_1\{\psi\}$ by Corollary 6.5.21. By the inductive hypothesis, we have

$$\begin{aligned}&\mathrm{Th}^{\mathcal{L}}(\mathcal{A}) \vdash_{\mathcal{HH}^{\mathcal{L}}} \{(\beta \wedge \varphi)\}\mathfrak{p}_0\{\psi\}\\ &\mathrm{Th}^{\mathcal{L}}(\mathcal{A}) \vdash_{\mathcal{HH}^{\mathcal{L}}} \{((\neg\beta) \wedge \varphi)\}\mathfrak{p}_1\{\psi\}.\end{aligned}$$

Finally, an application of rule $R^{\mathcal{L}}_{if}$ yields the desired conclusion.

For the second inductive step, let $\mathfrak{p} =$ **while** β **do** $\mathfrak{q}$ **endwhile**, where the inductive hypothesis holds for the program $\mathfrak{q}$. As above, by Corollary 6.5.21, we have $\mathcal{A} \models \theta$. Since $\mathcal{A}$ is expressive, there is an assertion γ that expresses $\mathsf{WLP}_{\mathcal{A}}(\mathfrak{p}, \psi)$. By Theorem 6.6.7, we have

$$\mathcal{A} \models \{\gamma\} \textbf{ while } \beta \textbf{ do } \mathfrak{q} \textbf{ endwhile}\{\psi\}.$$

By the second part of Theorem 6.5.11, we obtain

$$\mathcal{A} \models \{(\gamma \wedge \beta)\}\mathfrak{q};\ \textbf{while } \beta \textbf{ do } \mathfrak{q} \textbf{ endwhile}\{\psi\}.$$

Then, by the first part of Theorem 6.5.24, we have $\mathcal{A} \models \{(\gamma \wedge \beta)\}\mathfrak{q}\{\gamma\}$, which implies $\mathrm{Th}^{\mathcal{L}}(\mathcal{A}) \approx \{(\gamma \wedge \beta)\}\mathfrak{q}\{\gamma\}$ by Corollary 6.5.21. By inductive hypothesis, we obtain $\mathrm{Th}^{\mathcal{L}}(\mathcal{A}) \vdash_{\mathcal{HH}^{\mathcal{L}}} \{(\gamma \wedge \beta)\}\mathfrak{q}\{\gamma\}$. An application of the rule $R^{\mathcal{L}}_{while}$ gives $\mathrm{Th}^{\mathcal{L}}(\mathcal{A}) \vdash_{\mathcal{HH}^{\mathcal{L}}} \{\gamma\}$ **while** β **do** $\mathfrak{q}$ **endwhile**$\{(\gamma \wedge (\neg\beta))\}$. By Theorem 6.6.7, we have $\mathcal{A} \models (\varphi \to \gamma)$, which yields $\mathrm{Th}^{\mathcal{L}}(\mathcal{A}) \vdash_{\mathcal{HH}^{\mathcal{L}}} (\varphi \to \gamma)$, by an argument already made above using Theorem 4.13.28. By the first part of Theorem 6.5.11, we have $\mathcal{A} \models ((\gamma \wedge (\neg\beta)) \to \psi)$ which implies $\mathrm{Th}^{\mathcal{L}}(\mathcal{A}) \vdash_{\mathcal{HH}^{\mathcal{L}}} ((\gamma \wedge (\neg\beta)) \to \psi)$. An application of the rule $R^{\mathcal{L}}_{imp}$ gives the needed result.

Finally, for the third inductive step, let $\mathfrak{p}$ be a program of the form $\mathfrak{p}_0; \mathfrak{p}_1$, where the statement holds for $\mathfrak{p}_0, \mathfrak{p}_1$. As above, $\mathcal{A} \models \theta$ and the expressiveness of $\mathcal{A}$ shows that there is a formula γ that expresses $\mathsf{WLP}_{\mathcal{A}}(\mathfrak{p}_1, \psi)$. By Part (1) of Theorem 6.5.24, we have $\mathcal{A} \models \{\varphi\}\mathfrak{p}_0\{\gamma\}$. Also, by Theorem 6.6.7, $\mathcal{A} \models \{\gamma\}\mathfrak{p}_1\{\psi\}$, which, in turn, show that

$$\begin{aligned}&\mathrm{Th}^{\mathcal{L}}(\mathcal{A}) \approx \{\varphi\}\mathfrak{p}_0\{\gamma\}\\ &\mathrm{Th}^{\mathcal{L}}(\mathcal{A}) \approx \{\gamma\}\mathfrak{p}_1\{\psi\}.\end{aligned}$$

The inductive hypothesis implies that

$$\begin{aligned}&\mathrm{Th}^{\mathcal{L}}(\mathcal{A}) \vdash_{\mathcal{HH}^{\mathcal{L}}} \{\varphi\}\mathfrak{p}_0\{\gamma\}\\ &\mathrm{Th}^{\mathcal{L}}(\mathcal{A}) \vdash_{\mathcal{HH}^{\mathcal{L}}} \{\gamma\}\mathfrak{p}_1\{\psi\},\end{aligned}$$

which implies $\mathrm{Th}^{\mathcal{L}}(\mathcal{A}) \vdash_{\mathcal{HH}^{\mathcal{L}}} \theta$ through an application of the rule $R^{\mathcal{L}}_{comp}$. $\square$

To facilitate obtaining proofs in the Hilbert/Hoare formal system, we introduce below a special notation for these proofs called annotated program.

Definition 6.7.11. A *$\mathcal{L}$-bracketed assertion* is a string of the form $\{\varphi\}$, where φ is an $\mathcal{L}$-assertion. When the language $\mathcal{L}$ is clear from the context, we will refer to $\mathcal{L}$-bracketed assertions simply as bracketed assertions. $\square$

Theorem 6.7.12. *Let q be a string of basic symbols, $\{, \}$, and symbols from a first-order language $\mathcal{L}$. If $(\{\varphi\}, i)$and $(\{\psi\}, j)$ are two distinct occurrences of bracketed assertions in q, then these occurrences do not overlap.*

Proof. This statement follows immediately from the observation that no formula may contain an occurrence of $\{$ or $\}$. $\square$

Note that by Theorem 6.7.12, a string q that begins with a bracketed assertion determines uniquely this assertion, and the same is true for the last bracketed assertion.

Definition 6.7.13. Let $\mathcal{L}$ be a first-order language. We define inductively the sets $\mathtt{AP}^{at}_{\mathcal{L}}$ and $\mathtt{AP}_{\mathcal{L}}$ called the *set of atomic annotated $\mathcal{L}$-programs* and the *set of annotated $\mathcal{L}$-programs*, respectively. These sets consist of sequences of basic symbols, specification variables, symbols of $\mathcal{L}$, and the symbols $\{$ and $\}$.

(1) $\{(\varphi)_{v:=t}\}$
$v \leftarrow t$
$\{\varphi\}$
is an atomic annotated $\mathcal{L}$-program, where φ is an $\mathcal{L}$-assertion, v is a program variable, and t is an $\mathcal{L}$-term containing only program variables.

(2) If $\mathfrak{a}$ is an annotated $\mathcal{L}$-program that begins with $\{(\beta \wedge \varphi)\}$ and ends with $\{\psi\}$ and $\mathfrak{b}$ is an annotated $\mathcal{L}$-program that begins with $\{((\neg\beta) \wedge \varphi)\}$ and ends with $\{\psi\}$, then

```
{$\varphi$}
 **if** $\beta$
    **then**
     $\mathfrak{a}$
    **else**
     $\mathfrak{b}$
    **endif**
{$\psi$}
```

is an atomic annotated $\mathcal{L}$-program. Here β is a quantifier-free $\mathcal{L}$-formula.

(3) If $\mathfrak{a}$ is an annotated $\mathcal{L}$-program that begins with $\{(\beta \wedge \varphi)\}$ and ends with $\{\varphi\}$, then

```
{$\varphi$}
 **while** $\beta$ **do**
   $\mathfrak{a}$
 **endwhile**
{$((\neg\beta) \wedge \varphi)$}
```

is an atomic annotated $\mathcal{L}$-program, where β is a quantifier-free $\mathcal{L}$-formula.

(4) If $\mathfrak{a}$ is an annotated $\mathcal{L}$-program and φ', ψ' are $\mathcal{L}$-assertions, then

```
{$\varphi'$}
$\mathfrak{a}$
{$\psi'$}
```

is an atomic annotated $\mathcal{L}$-program.

(5) Every atomic annotated $\mathcal{L}$-program is an annotated $\mathcal{L}$-program.

(6) If $\mathfrak{a}_0$ is an atomic annotated $\mathcal{L}$-program and $\mathfrak{a}_1$ is an annotated $\mathcal{L}$-program, then $\mathfrak{a}_0; \mathfrak{a}_1$ is an annotated $\mathcal{L}$-program. □

To prove the unique readability of the definition of annotated program, we need to introduce the notions of opening and closing assertion and to prove several related technical results.

Definition 6.7.14. An occurrence of a bracketed assertion in an annotated program $\mathfrak{a}$ is *closing* if either the suffix of $\mathfrak{a}$ following the occurrence consists only of bracketed assertions or the first symbol following the occurrence that is not part of a bracketed assertion is among the symbols

endwhile **else** **endif** ;

Any occurrence which is not closing is *opening*. □

Example 6.7.15. By Part 1 of Definition 6.7.13, the following sequences

$\mathfrak{a}_0$: $\{(((x+y)-((x+y)-y))=y_\star \wedge ((x+y)-y)=x_\star)\}$
$\quad x{\leftarrow}x+y$
$\{((x-(x-y))=y_\star \wedge (x-y)=x_\star)\}$

$\mathfrak{a}_1$: $\{((x-(x-y))=y_\star \wedge (x-y)=x_\star)\}$
$\quad y{\leftarrow}x-y$
$\{((x-y)=y_\star \wedge y=x_\star)\}$

$\mathfrak{a}_2$: $\{((x-y)=y_\star \wedge y=x_\star)\}$
$\quad x{\leftarrow}x-y$
$\{(x=y_\star \wedge y=x_\star\}$

are atomic annotated programs. Then, by Part 5 of the definition, $\mathfrak{a}_2$ is an annotated program and, therefore, by Part 6, $\mathfrak{a}_1;\mathfrak{a}_2$ is an annotated program.

Next, by Part 6, $\mathfrak{a}_0;\mathfrak{a}_1;\mathfrak{a}_2$ is an annotated program Finally, by Part 4 followed by Part 5, the sequence

$$\{(x=x_\star \wedge y=y_\star)\}\mathfrak{a}_0;\mathfrak{a}_1;\mathfrak{a}_2\{(x=y_\star \wedge y=x_\star)\},$$

that is,

0) $\{(x=x_\star \wedge y=y_\star)\}$
1) $\{(((x+y)-((x+y)-y))=y_\star \wedge ((x+y)-y)=x_\star)\}$
2) $\quad x{\leftarrow}x+y$
3) $\{((x-(x-y))=y_\star \wedge (x-y)=x_\star)\}$;
4) $\{((x-(x-y))=y_\star \wedge (x-y)=x_\star)\}$
5) $\quad y{\leftarrow}x-y$
6) $\{((x-y)=y_\star \wedge y=x_\star)\}$;
7) $\{((x-y)=y_\star \wedge y=x_\star)\}$
8) $\quad x{\leftarrow}x-y$
9) $\{(x=y_\star \wedge y=x_\star)\}$
10) $\{(x=y_\star \wedge y=x_\star)\}$

is an annotated program. (We numbered the lines for easy reference; the line numbers are not part of the annotated program.) The bracketed assertions on lines 3, 6, 9, and 10 are closing occurrences. The remaining occurrences are opening. □

Theorem 6.7.16. *Let $\mathcal{L}$ be a first-order language and let $\mathfrak{a}_i$ be annotated $\mathcal{L}$-programs for $0 \leq i \leq 3$ and $\mathfrak{a}$ be an atomic annotated $\mathcal{L}$-program, where*

- $\mathfrak{a}_1$ *begins with* $\{(\beta \wedge \varphi)\}$ *and ends with* $\{\varphi\}$;
- $\mathfrak{a}_2$ *begins with* $\{(\beta \wedge \varphi)\}$ *and ends with* $\{\psi\}$;
- $\mathfrak{a}_3$ *begins with* $\{((\neg\beta) \wedge \varphi)\}$ *and ends with* $\{\psi\}$.

Consider the annotated programs:

$\mathfrak{b}_0$: $\{\varphi'\}\mathfrak{a}_0\{\psi'\}$
$\mathfrak{b}_1$: $\{\varphi\}$ **while** β **do** $\mathfrak{a}_1$ **endwhile**$\{((\neg\beta) \wedge \varphi)\}$
$\mathfrak{b}_2$: $\{\varphi\}$ **if** β **then** $\mathfrak{a}_2$ **else** $\mathfrak{a}_3$ **endif**$\{\beta\}$
$\mathfrak{b}_4$: $\mathfrak{a}; \mathfrak{a}_0$

(1) *An occurrence of a bracketed assertion in* $\mathfrak{a}_0$ *is closing if and only if the corresponding occurrence in* $\mathfrak{b}_0$ *is closing.*
(2) *An occurrence of a bracketed assertion in* $\mathfrak{a}_1$ *is closing if and only if the corresponding occurrence in* $\mathfrak{b}_1$ *is closing.*
(3) *An occurrence of a bracketed assertion in* $\mathfrak{a}_2$ *or* $\mathfrak{a}_3$ *is closing if and only if the corresponding occurrence in* $\mathfrak{b}_2$ *is closing.*
(4) *An occurrence of a bracketed assertion in* $\mathfrak{a}$ *or* $\mathfrak{a}_0$ *is closing if and only if the corresponding occurrence in* $\mathfrak{b}_3$ *is closing.*

Proof. We prove only the second part of the theorem, since the other parts have similar arguments. Let o be an occurrence of a bracketed assertion in $\mathfrak{a}_1$ and o' be the corresponding occurrence in $\mathfrak{b}_1$. Suppose that o is closing in $\mathfrak{a}_1$. If the suffix of $\mathfrak{a}_1$ following o consists of bracketed assertions, then the first symbol in $\mathfrak{b}_1$ following o' that is not contained in a bracketed assertion is **endwhile**, so o' is closing in $\mathfrak{b}_1$. On the other hand, if the first symbol in $\mathfrak{a}_1$ following o that is not part of a bracketed assertion is among the symbols **endwhile**, **else**, **endif**, ;, then the same is true for the first symbol which follows o' in $\mathfrak{b}_1$ and is not a part of a braketed assertion. Again, we conclude that o' is closing in $\mathfrak{b}_1$.

Conversely, suppose that o' is closing in $\mathfrak{b}_1$. In this case, the first symbol in $\mathfrak{b}_1$ that occurs outside of a bracketed assertion is one of the symbols **endwhile**, **else**, **endif**, ;. If this symbol occurs inside $\mathfrak{a}_1$, then o is clearly closing in $\mathfrak{a}_1$. Otherwise, this occurrence is the **endwhile** that follows $\mathfrak{a}_1$, which implies that o is followed in $\mathfrak{a}_1$ only by bracketed assertions and so is closing. □

Theorem 6.7.17. *Every annotated program begins with an opening occurrence of a bracketed assertion and ends with a closing occurrence of a bracketed assertion.*

Proof. The argument for the opening occurrence is by induction on annotated programs. If the annotated program is

$\{(\varphi)_{v:=t}\}$
$v{\leftarrow}t$
$\{\varphi\}$

then the occurrence $(\{(\varphi)_{v:=t}\}, 0)$ is clearly opening.

Similarly, the initial occurrence $(\{\varphi\}, 0)$ in the annotated programs introduced by Parts 2 and 3 of Definition 6.7.13 are opening.

Suppose now that $\mathfrak{b}$ is the annotated program

$\{\varphi'\}$
$\mathfrak{a}$
$\{\psi'\}$

where the statement holds for $\mathfrak{a}$. The beginning occurrence of a bracketed assertion in $\mathfrak{a}$ is opening and therefore, the first symbol that follows this occurrence in $\mathfrak{a}$ and is not included in a bracketed assertion is not one of the special symbols identified above. This implies that the same thing is true for the initial occurrence $(\{\varphi'\}, 0)$ of $\mathfrak{b}$, so this occurrence is opening. We leave to the reader the treatment of the last case of Definition 6.7.13.

It is easy to show by induction that every annotated program ends with an occurrence of a bracketed assertion and this is necessarily closing. □

Theorem 6.7.18. *Every annotated program has equal numbers of opening and closing occurrences of bracketed assertions.*

Proof. The argument is by induction on annotated programs using Theorems 6.7.16 and 6.7.17 and is left to the reader. □

Definition 6.7.19. Let $\mathfrak{a}$ be an annotated program and let (s, i) be an occurrence of a symbol s in $\mathfrak{a}$. The *level*, $\mathrm{lev}_{\mathfrak{a}}(s, i)$, is the number of complete occurrences of opening assertions, **while** s, and **if** s located to the left of the occurrence minus the number of complete occurrences of closing assertions, **endwhile**s, and **endif**s located to the left of the occurrence. ▯

For annotated programs, we have a result similar to Lemma 6.2.7:

Lemma 6.7.20. *Let $\mathcal{L}$ be a first-order language. For every annotated program $\mathfrak{a} \in \mathtt{AP}_{\mathcal{L}}$, we have $|\mathfrak{a}|_{\mathbf{while}} = |\mathfrak{a}|_{\mathbf{endwhile}}$ and $|\mathfrak{a}|_{\mathbf{if}} = |\mathfrak{a}|_{\mathbf{endif}}$.*

Proof. The argument is by induction based on Definition 6.7.13 and is left to the reader. □

Further, we have a lemma that is the analog of Lemmas 6.2.8 and 6.2.9:

Lemma 6.7.21. *Let $\mathcal{L}$ be a first-order language and $\mathfrak{a}$ be an annotated $\mathcal{L}$-program. For any occurrence of a semicolon in $\mathfrak{a}$, the level of the occurrence is nonnegative. Further, if $\mathfrak{a}$ is an atomic annotated program, the level of any occurrence of a semicolon in $\mathfrak{a}$ is positive. Also, the level of any occurrence of* **else** *in $\mathfrak{a}$ is positive.*

Proof. The proof is by induction on annotated programs and parallels the arguments of Lemma 6.2.8 and 6.2.9. □

Theorem 6.7.22. *Definition* 6.7.13 *meets the unique readability condition.*

Proof. We need to prove:

- Every atomic annotated program is put in the set $\mathtt{AP}^{at}_{\mathcal{L}}$ by only one of Rules (1)–(4) of the definition.
- If an atomic annotated program enters $\mathtt{AP}^{at}_{\mathcal{L}}$ by Rule (2), (3) or (4), then $\mathfrak{a}$ (and $\mathfrak{b}$ for Rule (2)), are uniquely determined.
- Every annotated program is put in the set $\mathtt{AP}_{\mathcal{L}}$ by only one of Rules (5) and (6) of the definition.
- If an annotated program enters $\mathtt{AP}_{\mathcal{L}}$ by Rule (6), then the atomic annotated program $\mathfrak{a}_0$ and the annotated program $\mathfrak{a}_1$ are uniquely determined.

Note that in principle a uniqueness condition has to be proved for Rule (5). However, this condition is trivially satisfied.

If $\mathfrak{c}$ is an atomic annotated program, we need to examine the first symbol following the initial bracketed assertion. If this symbol is a variable, then $\mathfrak{c}$ entered $\mathtt{AP}^{at}_{\mathcal{L}}$ due to Rule (1). Otherwise, if this symbol is **if**, **while** or {, then $\mathfrak{c}$ entered $\mathtt{AP}^{at}_{\mathcal{L}}$ due to Rules (2), (3), or (4), respectively.

If $\mathfrak{c} = \{\varphi\}$ **if** β **then** $\mathfrak{a}$ **else** $\mathfrak{b}$ **endif**$\{\psi\}$, then $\mathfrak{c}$ contains exactly one occurrence of **else** at level 2. Indeed, note that any occurrence of **else** in $\mathfrak{a}$ or $\mathfrak{b}$ is at least at level 1 within these annotated programs and each such occurrence is preceded by an opening occurrence of a bracketed assertion and by an **if** in $\mathfrak{c}$, which guarantees that such occurrences would be at least level 3. Therefore, $\mathfrak{a}$ is the infix of $\mathfrak{c}$ located between the first occurrence of **then** and the occurrence of **else** at level 2, and $\mathfrak{b}$ is the infix located between this occurrence of **else** and the last occurrence of **endif**.

If $\mathfrak{c} = \{\varphi\}$ **while** β **do** $\mathfrak{a}$ **endwhile**$\{((\neg\beta) \wedge \varphi)\}$, then $\mathfrak{a}$ is the infix of $\mathfrak{c}$ located between the initial **do** and final **endwhile**.

When $\mathfrak{c} = \{\varphi'\}\mathfrak{a}\{\psi'\}$, then $\mathfrak{a}$ is the infix between the first bracketed assertion and the last bracketed assertion.

An annotated program enters $\mathtt{AP}_{\mathcal{L}}$ by Rule (6) if and only if it contains an occurrence of a semicolon at level 0, since every semicolon that occurs in an atomic annotated program is at positive level by Lemma 6.7.21. If the program was put in by Rule (6), then the atomic annotated program $\mathfrak{a}_0$ is the prefix of $\mathfrak{c}$ that ends just before the first level 0 semicolon, and $\mathfrak{a}_1$ is the suffix of $\mathfrak{c}$ that follows this semicolon occurrence. $\square$

For each annotated $\mathcal{L}$-program $\mathfrak{a}$, we will define an $\mathcal{L}$-program $\mathsf{prog}(\mathfrak{a})$ and a set formulas $\mathsf{VC}(\mathfrak{a})$ called the set of *verification conditions* for $\mathfrak{a}$ in Definitions 6.7.23 and 6.7.24. The idea is that $\mathfrak{a}$ is an attempted argument to show that $\mathcal{H} = \{\varphi\}\mathsf{prog}(\mathfrak{a})\{\psi\}$ holds in an $\mathcal{L}$-structure $\mathcal{A}$, where $\{\varphi\}$ is the initial bracketed assertion in $\mathfrak{a}$ and $\{\psi\}$ is the final bracketed assertion in $\mathfrak{a}$. The set of formulas $\mathsf{VC}(\mathfrak{a})$ will be defined such that if $\mathcal{A} \models \mathsf{VC}(\mathfrak{a})$, then $\mathrm{Th}^{\mathcal{L}}(\mathcal{A}) \vdash_{\mathcal{HH}^{\mathcal{L}}} \mathcal{H}$ and thus, by the soundness of $\mathcal{HH}^{\mathcal{L}}$, $\mathcal{A} \models \mathcal{H}$. We will refer to the $\mathcal{L}$-partial correctness triple $\mathcal{H}$ as the *target* of $\mathfrak{a}$.

Since Definition 6.7.13 satisfies the unique readability condition, the definitions of prog and VC can be given recursively.

Definition 6.7.23. Let $\mathcal{L}$ be a first-order language. We define two functions $\mathsf{prog}^{at}_{\mathcal{L}} : \mathtt{AP}^{at}_{\mathcal{L}} \longrightarrow \mathtt{WHILE}_{\mathcal{L}}$ and $\mathsf{prog}_{\mathcal{L}} : \mathtt{AP}_{\mathcal{L}} \longrightarrow \mathtt{WHILE}_{\mathcal{L}}$ by:

(1) If $\mathfrak{c}$ is the atomic annotated $\mathcal{L}$-program

$\{(\varphi)_{v:=t}\}$
$v{\leftarrow}t$
$\{\varphi\}$

then $\mathsf{prog}^{at}_{\mathcal{L}}(\mathfrak{c}) = v{\leftarrow}t$.

(2) Let $\mathfrak{a}$ be an annotated $\mathcal{L}$-program that begins with $\{(\beta \wedge \varphi)\}$ and ends with $\{\psi\}$. Also, let $\mathfrak{b}$ be an annotated $\mathcal{L}$-program that begins with $\{((\neg\beta)\wedge\varphi)\}$ and ends with $\{\psi\}$. If $\mathfrak{c}$ is the annotated $\mathcal{L}$-program

```
{φ}
 if β
    then
     𝔞
    else
     𝔟
    endif
{ψ}
```

then

$$\mathsf{prog}^{at}_{\mathcal{L}}(\mathfrak{c}) = \textbf{ if } \beta \textbf{ then } \mathsf{prog}_{\mathcal{L}}(\mathfrak{a}) \textbf{ else } \mathsf{prog}_{\mathcal{L}}(\mathfrak{b}) \textbf{ endif}.$$

(3) Let $\mathfrak{a}$ be an annotated $\mathcal{L}$-program that begins with $\{(\beta\wedge\varphi)\}$ and ends with $\{\varphi\}$. If $\mathfrak{c}$ is the atomic annotated $\mathcal{L}$-program:

```
{φ}
 while β do
   𝔞
 endwhile
{((¬β) ∧ φ)}
```

then

$$\mathsf{prog}^{at}_{\mathcal{L}}(\mathfrak{c}) = \textbf{ while } \beta \textbf{ do } \mathsf{prog}_{\mathcal{L}}(\mathfrak{a}) \textbf{ endwhile}.$$

(4) Let $\mathfrak{a}$ be an annotated $\mathcal{L}$-program and let $\mathfrak{c}$ be the atomic annotated $\mathcal{L}$-program

```
{φ′}
𝔞
{ψ′}
```

Then, $\mathsf{prog}^{at}_{\mathcal{L}}(\mathfrak{c}) = \mathsf{prog}_{\mathcal{L}}(\mathfrak{a})$.

(5) If $\mathfrak{c}$ is an atomic annotated $\mathcal{L}$-program, then $\mathsf{prog}_{\mathcal{L}}(\mathfrak{c}) = \mathsf{prog}^{at}_{\mathcal{L}}(\mathfrak{c})$.

(6) Let $\mathfrak{a}_0$ be an atomic annotated $\mathcal{L}$-program and $\mathfrak{a}_1$ be an annotated $\mathcal{L}$-program. For the annotated $\mathcal{L}$-program $\mathfrak{c} = \mathfrak{a}_0; \mathfrak{a}_1$ we have $\mathsf{prog}_{\mathcal{L}}(\mathfrak{c}) = \mathsf{prog}^{at}_{\mathcal{L}}(\mathfrak{a}_0); \mathsf{prog}_{\mathcal{L}}(\mathfrak{a}_1)$.

The $\mathcal{L}$-partial correctness Hoare triple $\{\varphi\}\mathsf{prog}_{\mathcal{L}}(\mathfrak{c})\{\psi\}$, where $\{\varphi\}$ is the initial bracketed assertion of $\mathfrak{c}$ and $\{\psi\}$ is the final bracketed assertion of $\mathfrak{c}$ is called the *target of* $\mathfrak{c}$ and is denoted by $\mathsf{target}_{\mathcal{L}}(\mathfrak{c})$. Also, we say that $\mathfrak{c}$ is an *annotated program for the triple* $\mathsf{target}_{\mathcal{L}}(\mathfrak{c})$. □

Definition 6.7.24. Let $\mathcal{L}$ be a first-order language. We define two functions $\mathsf{VC}^{at}_{\mathcal{L}} : \mathtt{AP}^{at}_{\mathcal{L}} \longrightarrow \mathcal{P}(\mathrm{FORM}_{\mathcal{L}})$ and $\mathsf{VC}_{\mathcal{L}} : \mathtt{AP}_{\mathcal{L}} \longrightarrow \mathcal{P}(\mathrm{FORM}_{\mathcal{L}})$ by:

(1) If $\mathfrak{c}$ is the atomic annotated $\mathcal{L}$-program

```
{$(\varphi)_{v:=t}$}
$v \leftarrow t$
{$\varphi$}
```

then $\mathsf{VC}^{at}_{\mathcal{L}}(\mathfrak{c}) = \emptyset$.

(2) If $\mathfrak{a}$ is an annotated $\mathcal{L}$-program that begins with $\{(\beta \wedge \varphi)\}$ and ends with $\{\psi\}$ and $\mathfrak{b}$ is an annotated $\mathcal{L}$-program that begins with $\{((\neg\beta) \wedge \varphi)\}$ and ends with $\{\psi\}$, and $\mathfrak{c}$ is the atomic annotated $\mathcal{L}$-program

```
{$\varphi$}
 if $\beta$
    then
     $\mathfrak{a}$
    else
     $\mathfrak{b}$
    endif
{$\psi$}
```

then $\mathsf{VC}^{at}_{\mathcal{L}}(\mathfrak{c}) = \mathsf{VC}_{\mathcal{L}}(\mathfrak{a}) \cup \mathsf{VC}_{\mathcal{L}}(\mathfrak{b})$.

(3) If $\mathfrak{a}$ is an annotated $\mathcal{L}$-program that begins with $\{(\beta \wedge \varphi)\}$ and ends with $\{\varphi\}$ and $\mathfrak{c}$ is the atomic annotated $\mathcal{L}$-program

```
{$\varphi$}
 while $\beta$ do
   $\mathfrak{a}$
 endwhile
{$((\neg\beta) \wedge \varphi)$}
```

then $\mathsf{VC}^{at}_{\mathcal{L}}(\mathfrak{c}) = \mathsf{VC}_{\mathcal{L}}(\mathfrak{a})$.

(4) If $\mathfrak{a}$ is an annotated $\mathcal{L}$-program and $\mathfrak{c}$ is the atomic annotated $\mathcal{L}$-program

```
{$\varphi'$}
$\mathfrak{a}$
{$\psi'$}
```

then $\mathsf{VC}^{at}_{\mathcal{L}}(\mathfrak{c}) = \mathsf{VC}_{\mathcal{L}}(\mathfrak{a}) \cup \{(\varphi' \to \varphi), (\psi \to \psi')\}$, where $\{\varphi\}$ is the initial bracketed assertion of $\mathfrak{a}$ and $\{\psi\}$ is the final bracketed assertion of $\mathfrak{a}$.

(5) If $\mathfrak{c}$ is an atomic annotated $\mathcal{L}$-program, then $\mathsf{VC}_{\mathcal{L}}(\mathfrak{c}) = \mathsf{VC}^{at}_{\mathcal{L}}(\mathfrak{c})$.

(6) If $\mathfrak{a}_0$ is an atomic annotated $\mathcal{L}$-program that ends with the bracketed assertion $\{\psi_0\}$, $\mathfrak{a}_1$ is an annotated $\mathcal{L}$-program that begins with the bracketed assertion $\{\varphi_1\}$, and $\mathfrak{c}$ is the annotated program $\mathfrak{a}_0; \mathfrak{a}_1$, then $\mathsf{VC}_{\mathcal{L}}(\mathfrak{c}) = \mathsf{VC}^{at}_{\mathcal{L}}(\mathfrak{a}_0) \cup \mathsf{VC}_{\mathcal{L}}(\mathfrak{a}_1) \cup \{(\psi_0 \to \varphi_1)\}$.

When there is no risk of confusion, we will drop the subscript $\mathcal{L}$ and write simply $\mathsf{VC}^{at}(\mathfrak{c})$ and $\mathsf{VC}(\mathfrak{c})$. We refer to the set $\mathsf{VC}(\mathfrak{c})$ as the set of *verification conditions* for $\mathfrak{c}$. □

Theorem 6.7.25. *Let $\mathcal{L}$ be a first-order language, $\mathcal{A}$ be an $\mathcal{L}$-structure, and let $\mathfrak{c}$ be an annotated $\mathcal{L}$-program. If $\mathcal{A} \models \mathsf{VC}_{\mathcal{L}}(\mathfrak{c})$, then $Th^{\mathcal{L}}(\mathcal{A}) \vdash_{\mathcal{HH}^{\mathcal{L}}} \mathsf{target}_{\mathcal{L}}(\mathfrak{c})$.*

Proof. The argument is by induction on annotated programs. The basis step, when $\mathfrak{c}$ is the annotated program

```
$\{(\varphi)_{v:=t}\}$
$v \leftarrow t$
$\{\varphi\}$
```

is immediate because $\mathsf{target}(\mathfrak{c}) = \mathfrak{c}$ is an assignment axiom of $\mathcal{HH}^{\mathcal{L}}$.

We need to consider several inductive steps. Suppose that $\mathfrak{c}$ is the annotated program

```
$\{\varphi\}$
  while $\beta$ do
    $\mathfrak{a}$
  endwhile
$\{((\neg\beta) \wedge \varphi)\}$
```

and the statement holds for the annotated program $\mathfrak{a}$ that begins with $\{(\beta \wedge \varphi)\}$ and ends with $\{\varphi\}$.

Since $\mathcal{A} \models \mathsf{VC}(\mathfrak{c}) = \mathsf{VC}(\mathfrak{a})$, by the inductive hypothesis, we have

$$\mathrm{Th}^{\mathcal{L}}(\mathcal{A}) \vdash_{\mathcal{HH}^{\mathcal{L}}} \mathsf{target}(\mathfrak{a}) = \{(\beta \wedge \varphi)\}\mathsf{prog}(\mathfrak{a})\{\varphi\}.$$

By applying the rule $R^{\mathcal{L}}_{while}$, we obtain

$$\begin{aligned}\mathrm{Th}^{\mathcal{L}}(\mathcal{A}) \vdash_{\mathcal{HH}^{\mathcal{L}}} & \{\varphi\}\ \mathbf{while}\ \beta\ \mathbf{do}\ \mathsf{prog}(\mathfrak{a})\ \mathbf{endwhile}\{((\neg\beta) \wedge \varphi)\} \\ &= \mathsf{target}(\mathfrak{c}).\end{aligned}$$

The case when $\mathfrak{c}$ is of the form $\{\varphi\}$ **if** $\beta \cdots \{\psi\}$ is similar and is left to the reader.

Suppose now that $\mathfrak{c}$ is the atomic annotated program

$\{\varphi'\}$
$\mathfrak{a}$
$\{\psi'\}$

where the statement holds for the annotated program $\mathfrak{a}$ that begins with $\{\varphi\}$ and ends with $\{\psi\}$. Since $\mathcal{A} \models \mathsf{VC}(\mathfrak{c}) = \mathsf{VC}(\mathfrak{a}) \cup \{(\varphi' \to \varphi), (\psi \to \psi')\}$, it follows by the inductive hypothesis, that $\mathrm{Th}^{\mathcal{L}}(\mathcal{A}) \vdash_{\mathcal{HH}^{\mathcal{L}}} \mathsf{target}(\mathfrak{a}) = \{\varphi\}\mathsf{prog}(\mathfrak{a})\{\psi\} = \{\varphi\}\mathsf{prog}(\mathfrak{c})\{\psi\}$. By Theorem 4.13.28, we have $\mathrm{Th}^{\mathcal{L}}(\mathcal{A}) \models (\varphi' \to \varphi)$ and $\mathrm{Th}^{\mathcal{L}}(\mathcal{A}) \models (\psi \to \psi')$. By the completeness of $\mathcal{HF}^{\mathcal{L}}$, we have $\mathrm{Th}^{\mathcal{L}}(\mathcal{A}) \vdash_{\mathcal{HF}^{\mathcal{L}}} (\varphi' \to \varphi)$ and $\mathrm{Th}^{\mathcal{L}}(\mathcal{A}) \vdash_{\mathcal{HF}^{\mathcal{L}}} (\psi \to \psi')$ and the same inferences can be made in $\mathcal{HH}^{\mathcal{L}}$. Therefore, by applying the implication rule, we obtain $\mathrm{Th}^{\mathcal{L}}(\mathcal{A}) \vdash_{\mathcal{HH}^{\mathcal{L}}} \{\varphi'\}\mathsf{prog}(\mathfrak{c})\{\psi'\} = \mathsf{target}(\mathfrak{c})$.

Let $\mathfrak{c}$ be the annotated program $\mathfrak{a}_0; \mathfrak{a}_1$, where $\mathfrak{a}_0$ is an atomic annotated program ending with $\{\psi_0\}$, $\mathfrak{a}_1$ is an annotated program beginning with $\{\varphi_1\}$, and the statement holds for $\mathfrak{a}_0$ and $\mathfrak{a}_1$. Since $\mathcal{A} \models \mathsf{VC}(\mathfrak{c}) = \mathsf{VC}(\mathfrak{a}_0) \cup \mathsf{VC}(\mathfrak{a}_1) \cup \{(\psi_0 \to \varphi_1)\}$, it follows from the inductive hypothesis that $\mathrm{Th}^{\mathcal{L}}(\mathcal{A}) \vdash_{\mathcal{HH}^{\mathcal{L}}} \mathsf{target}(\mathfrak{a}_0) = \{\varphi_0\}\mathsf{prog}(\mathfrak{a}_0)\{\psi_0\}$ and $\mathrm{Th}^{\mathcal{L}}(\mathcal{A}) \vdash_{\mathcal{HH}^{\mathcal{L}}} \mathsf{target}(\mathfrak{a}_1) = \{\varphi_1\}\mathsf{prog}(\mathfrak{a}_1)\{\psi_1\}$, where $\{\varphi_0\}$ is the initial bracketed assertion of $\mathfrak{a}_0$ and $\{\psi_1\}$ is the final bracketed assertion of $\mathfrak{a}_1$. Since $\mathcal{A} \models (\psi_0 \to \varphi_1)$, by an argument similar to the one in the previous paragraph, we have $\mathrm{Th}^{\mathcal{L}}(\mathcal{A}) \vdash_{\mathcal{HH}^{\mathcal{L}}} (\psi_0 \to \varphi_1)$, and clearly, $\mathrm{Th}^{\mathcal{L}}(\mathcal{A}) \vdash_{\mathcal{HH}^{\mathcal{L}}} (\varphi_0 \to \varphi_0)$. By applying $R^{\mathcal{L}}_{imp}$, we obtain $\mathrm{Th}^{\mathcal{L}}(\mathcal{A}) \vdash_{\mathcal{HH}^{\mathcal{L}}} \{\varphi_0\}\mathsf{prog}(\mathfrak{a}_0)\{\varphi_1\}$. Finally, an application of the composition rule yields

$$\begin{aligned}\mathrm{Th}^{\mathcal{L}}(\mathcal{A}) \vdash_{\mathcal{HH}^{\mathcal{L}}} &\{\varphi_0\}\mathsf{prog}(\mathfrak{a}_0); \mathsf{prog}(\mathfrak{a}_1)\{\psi_1\}\\ &= \{\varphi_0\}\mathsf{prog}(\mathfrak{c})\{\psi_1\} = \mathsf{target}(\mathfrak{c}).\end{aligned}$$

□

Corollary 6.7.26. *Let $\mathcal{L}$ be a first-order language, $\mathcal{A}$ be an $\mathcal{L}$-structure, and let $\mathfrak{c}$ be an annotated $\mathcal{L}$-program. If $\mathcal{A} \models \mathsf{VC}_{\mathcal{L}}(\mathfrak{c})$, then $\mathcal{A} \models \mathsf{target}_{\mathcal{L}}(\mathfrak{c})$.*

Proof. Suppose that $\mathcal{A} \models \mathsf{VC}_{\mathcal{L}}(\mathfrak{c})$. By the previous theorem and by the Soundness of $\mathcal{HH}^{\mathcal{L}}$, we have $\mathrm{Th}^{\mathcal{L}}(\mathcal{A}) \models\!\approx \mathsf{target}_{\mathcal{L}}(\mathfrak{c})$. Therefore, by Theorem 6.7.9, we have $\mathcal{A} \models \mathsf{target}_{\mathcal{L}}(\mathfrak{c})$. □

Example 6.7.27. For the atomic annotated programs $\mathfrak{a}_0, \mathfrak{a}_1, \mathfrak{a}_2$ of Example 6.7.15, we have

$$\begin{aligned}\mathsf{prog}(\mathfrak{a}_0) &= x{\leftarrow}x+y\\ \mathsf{prog}(\mathfrak{a}_1) &= y{\leftarrow}x-y\\ \mathsf{prog}(\mathfrak{a}_2) &= x{\leftarrow}x-y.\end{aligned}$$

Therefore, $\mathsf{prog}(\mathfrak{a}_1;\mathfrak{a}_2) = y{\leftarrow}x-y;\ x{\leftarrow}x-y$ and thus, $\mathsf{prog}(\mathfrak{a}_0;\mathfrak{a}_1;\mathfrak{a}_2) = x{\leftarrow}x+y; y{\leftarrow}x-y; x{\leftarrow}x-y$. Consequently, if

$$\mathfrak{c} = \{(x = x_\star \wedge y = y_\star)\}\mathfrak{a}_0;\mathfrak{a}_1;\mathfrak{a}_2\{(x = y_\star \wedge y = x_\star)\},$$

then $\mathsf{prog}(\mathfrak{c}) = x{\leftarrow}x+y; y{\leftarrow}x-y; x{\leftarrow}x-y$, so the target of $\mathfrak{c}$ is the Hoare triple $\mathcal{H}_1$ given by:

$$\{(x = x_\star \wedge y = y_\star)\}x{\leftarrow}x+y; y{\leftarrow}x-y; x{\leftarrow}x-y\{(x = y_\star \wedge y = x_\star)\},$$

which is the partial correctness version of the total correctness Hoare triple $\mathcal{H}$ of Example 6.5.6.

For the initial atomic annotated programs $\mathfrak{a}_0, \mathfrak{a}_1, \mathfrak{a}_2$, we have $\mathsf{VC}(\mathfrak{a}_i) = \emptyset$, for $0 \leq i \leq 2$. Then, $\mathsf{VC}(\mathfrak{a}_1;\mathfrak{a}_2) = \{\theta_{12}\}$, where $\theta_{12} = (((x-y) = y_\star \wedge y = x_\star) \to ((x-y) = y_\star \wedge y = x_\star))$. Further, $\mathsf{VC}(\mathfrak{a}_0;\mathfrak{a}_1;\mathfrak{a}_2) = \{\theta_{12}, \theta_{0,12}\}$, where $\theta_{0,12} = (((x-(x-y)) = y_\star \wedge (x-y) = x_\star) \to ((x-(x-y)) = y_\star \wedge (x-y) = x_\star))$. Finally, $\mathsf{VC}(\mathfrak{c}) = \{\theta_{12}, \theta_{0,12}, \theta, \theta'\}$, where $\theta = ((x = x_\star \wedge y = y_\star) \to (((x+y)-((x+y)-y)) = y_\star \wedge ((x+y)-y) = x_\star))$ and $\theta' = ((x = y_\star \wedge y = x_\star) \to (x = y_\star \wedge y = x_\star))$.

If $\mathcal{A}$ is the structure introduced in Example 6.3.12, then it is clear that $\mathcal{A} \models \{\theta_{12}, \theta_{0,12}, \theta'\}$ because all these formulas are tautologies, and it is easy to see that $\mathcal{A} \models \theta$. Therefore, by Corollary 6.7.26, $\mathcal{A} \models \mathsf{target}_{\mathcal{L}}(\mathfrak{c})$. Indeed, we have shown the stronger total correctness of $\mathcal{H}$ in $\mathcal{A}$ in Example 6.5.6. The purpose of this example was to prove the partial correctness using the Hilbert/Hoare formal system $\mathcal{HH}^{\mathcal{L}}$.
$\square$

Definition 6.7.28. Let $\mathfrak{a}_0, \mathfrak{a}_1$ be two annotated programs such that the initial bracketed assertion of $\mathfrak{a}_i$ is $\{\varphi_i\}$ for $i = 0, 1$ and the closing bracketed assertion of both $\mathfrak{a}_0$ and $\mathfrak{a}_1$ is $\{\psi\}$. Also, let φ' be the formula $((\beta \to \varphi_0) \wedge ((\neg\beta) \to \varphi_1)$. Then, the sequence

```
{$\varphi'$}
 if $\beta$
    then
     $\mathfrak{a}_0$
    else
     $\mathfrak{a}_1$
    endif
{$\psi$}
```

is an abbreviation of the following annotated program:

```
{$\varphi'$}
 if $\beta$
    then
     {$(\beta \wedge \varphi')$}
       $\mathfrak{a}_0$
     {$\psi$}
    else
     {$((\neg\beta) \wedge \varphi')$}
     $\mathfrak{a}_1$
     {$\psi$}
    endif
{$\psi$}
```

∎

Theorem 6.7.29. *Let* $\mathfrak{a}_0, \mathfrak{a}_1$ *be two annotated programs such that the initial bracketed assertion of* $\mathfrak{a}_i$ *is* $\{\varphi_i\}$ *for* $i = 0, 1$ *and the closing bracketed assertion of both* $\mathfrak{a}_0$ *and* $\mathfrak{a}_1$ *is* $\{\psi\}$*. Also, let* φ' *be the formula* $((\beta \to \varphi_0) \wedge ((\neg\beta) \to \varphi_1)$*. For the annotated program* $\mathfrak{c}$ *given by*:

```
{$\varphi'$}
 if $\beta$
    then
     $\mathfrak{a}_0$
    else
     $\mathfrak{a}_1$
    endif
{$\psi$}
```

we have:

$$\mathit{prog}(\mathfrak{c}) = \textbf{ if } \beta \textbf{ then } \mathit{prog}(\mathfrak{a}_0) \textbf{ else } \mathit{prog}(\mathfrak{a}_1) \textbf{ endif}$$

$$\begin{aligned} \mathit{target}(\mathfrak{c}) &= \{\varphi'\}\mathit{prog}(\mathfrak{c})\{\psi\} \\ \mathit{VC}(\mathfrak{c}) &= \mathit{VC}(\mathfrak{a}_0) \cup \mathit{VC}(\mathfrak{a}_1) \cup \\ &\qquad \{((\beta \wedge \varphi') \rightarrow \varphi_0), (\psi \rightarrow \psi), (((\neg\beta) \wedge \varphi') \rightarrow \varphi_1)\} \end{aligned}$$

Proof. This follows directly from the application of the definitions. □

Note that all verification conditions of the program $\mathfrak{c}$ of the previous theorem that do not belong to $\mathsf{VC}(\mathfrak{a}_0) \cup \mathsf{VC}(\mathfrak{a}_1)$ are tautologies.

Example 6.7.30. We construct an annotated program $\mathfrak{a}$ for the triple

$$\mathcal{H} = \{(x \geq 0 \wedge y \geq 0 \wedge x = x_\star \wedge y = y_\star)\}\mathfrak{p}\{z = \gcd(x_\star, y_\star)\},$$

where $\mathfrak{p}$ is the program

```
while ($x \neq 0 \wedge y \neq 0$)
    do  if  $x > y$
                then $x \leftarrow x - y$
                else $y \leftarrow y - x$
            endif
endwhile;
if $x = 0$
    then $z \leftarrow y$
    else $z \leftarrow x$
endif
```

of Example 6.2.12, which can be written as $\mathfrak{p}_0; \mathfrak{p}_1$, where $\mathfrak{p}_0$ is

```
while ($x \neq 0 \wedge y \neq 0$)
    do  if  $x > y$
                then $x \leftarrow x - y$
                else $y \leftarrow y - x$
            endif
endwhile
```

and $\mathfrak{p}_1$ is the rest of $\mathfrak{p}$.

The process of constructing $\mathfrak{a}$ involves "pulling up" the postcondition ψ which is the assertion $z = \gcd(x_\star, y_\star)$.

We begin by pulling up the postcondition through the assignment statements $z \leftarrow y$ and $z \leftarrow x$ of $\mathfrak{p}_1$ to obtain the annotated programs:

```
$\{y = \mathrm{gcd}(x_\star, y_\star)\}$
  $z \leftarrow y$
$\{z = \mathrm{gcd}(x_\star, y_\star)\}$
```

and

```
$\{x = \mathrm{gcd}(x_\star, y_\star)\}$
  $z \leftarrow x$
$\{z = \mathrm{gcd}(x_\star, y_\star)\}$
```

Starting from these two annotated programs, we obtain the annotated program $\mathfrak{d}_1$

```
$\{((x = 0 \to y = \mathrm{gcd}(x_\star, y_\star)) \land ((\neg(x = 0)) \to x = \mathrm{gcd}(x_\star, y_\star)))\}$
  **if** $x = 0$
    **then**
   $\{y = \mathrm{gcd}(x_\star, y_\star)\}$
     $z \leftarrow y$
   $\{z = \mathrm{gcd}(x_\star, y_\star)\}$
  **else**
   $\{x = \mathrm{gcd}(x_\star, y_\star)\}$
     $z \leftarrow x$
   $\{z = \mathrm{gcd}(x_\star, y_\star)\}$
  **endif**
$\{z = \mathrm{gcd}(x_\star, y_\star)\}$
```

The target of $\mathfrak{d}_1$ is the triple

```
$\{((x = 0 \to y = \mathrm{gcd}(x_\star, y_\star)) \land ((\neg(x = 0)) \to x = \mathrm{gcd}(x_\star, y_\star)))\}$
$\mathfrak{p}_1$
$\{z = \mathrm{gcd}(x_\star, y_\star)\}$
```

and the non-tautologous part of $\mathsf{VC}(\mathfrak{d}_1)$ is empty. Now we have reached a **while** loop **while** $\beta \cdots$ and we have to produce a new postcondition of the form $((\neg\beta) \wedge \theta)$, where θ is a loop invariant such that

$$\mathcal{A}' \models (((\neg\beta) \wedge \theta) \to \varphi$$

where $\varphi = ((x = 0 \to y = \gcd(x_\star, y_\star)) \wedge ((\neg(x = 0)) \to x = \gcd(x_\star, y_\star)))$ is the precondition of $\mathfrak{d}_1$ and $\mathcal{A}'$ is the structure introduced in Example 6.5.7. We choose $\theta = (x \geq 0 \wedge y \geq 0 \wedge \gcd(x, y) = \gcd(x_\star, y_\star))$. In building the annotated program $\mathfrak{d}_0$, we will also use the formulas:

$$(\theta)_{y:=y-x} = (x \geq 0 \wedge y - x \geq 0 \wedge \gcd(x, y - x) = \gcd(x_\star, y_\star))$$
$$(\theta)_{x:=x-y} = (x - y \geq 0 \wedge y \geq 0 \wedge \gcd(x - y, y) = \gcd(x_\star, y_\star)).$$

Pulling θ through the assignment statements $x \leftarrow x - y$ and $y \leftarrow y - x$ gives the annotated programs:

```
{$(\theta)_{x:=x-y}$}
  $x \leftarrow x - y$
{$\theta$}
```

and

```
{$(\theta)_{y:=y-x}$}
  $y \leftarrow y - x$
{$\theta$}
```

which give us the annotated program

```
{$((x > y \to (\theta)_{x:=x-y}) \wedge ((\neg(x > y)) \to (\theta)_{y:=y-x}))$}
 **if** $x > y$
   **then**
    {$(\theta)_{x:=x-y}$}
      $x \leftarrow x - y$
    {$\theta$}
   **else**
    {$(\theta)_{y:=y-x}$}
      $y \leftarrow y - x$
    {$\theta$}
   **endif**
{$\theta$}
```

whose set of non-tautologous verification conditions is empty. Next, since we need the conjunction of the while condition and the loop invariant at the top of the annotated program inside the **while** loop, we construct the annotated program $\mathfrak{e}$:

```
$\{(x \neq 0 \wedge y \neq 0 \wedge \theta)\}$
$\{((x > y \to (\theta)_{x:=x-y}) \wedge ((\neg(x > y)) \to (\theta)_{y:=y-x}))\}$
 **if** $x > y$
   **then**
    $\{(\theta)_{x:=x-y}\}$
      $x \leftarrow x - y$
    $\{\theta\}$
   **else**
    $\{(\theta)_{y:=y-x}\}$
      $y \leftarrow y - x$
    $\{\theta\}$
   **endif**
$\{\theta\}$
$\{\theta\}$
```

For this annotated program, we have

$$\mathsf{VC}(\mathfrak{e}) = \{((x \neq 0 \wedge y \neq 0 \wedge \theta) \to ((x > y \to (\theta)_{x:=x-y}) \wedge ((\neg(x > y)) \to (\theta)_{y:=y-x}))), (\theta \to \theta)\}.$$

The final step in obtaining $\mathfrak{d}_0$ is shown in Figure 6.3, where $\mathsf{VC}(\mathfrak{d}_0) = \mathsf{VC}(\mathfrak{e})$.

The annotated programs $\mathfrak{d}_0, \mathfrak{d}_1$ are now concatenated and $\mathsf{VC}(\mathfrak{d}_0; \mathfrak{d}_1) = \mathsf{VC}(\mathfrak{d}_0) \cup \mathsf{VC}(\mathfrak{d}_1) \cup \{(((\neg(x \neq 0 \wedge y \neq 0)) \wedge \theta) \to ((x = 0 \to y = \gcd(x_\star, y_\star)) \wedge ((\neg(x = 0)) \to x = \gcd(x_\star, y_\star))))\}$.

Finally, $\mathfrak{a}$ is the annotated program

```
$\{(x \geq 0 \wedge y \geq 0 \wedge x = x_\star \wedge y = y_\star)\}$
$\mathfrak{d}_0$;
$\mathfrak{d}_1$
$\{z = \gcd(x_\star, y_\star)\}$
```

```
{$\theta$}
 **while** $(x \neq 0 \wedge y \neq 0)$ **do**
  {$(x \neq 0 \wedge y \neq 0 \wedge \theta)$}
  {$((x > y \rightarrow (\theta)_{x:=x-y}) \wedge ((\neg(x > y)) \rightarrow (\theta)_{y:=y-x}))$}
   **if** $x > y$
     **then**
      {$(\theta)_{x:=x-y}$}
        $x \leftarrow x - y$
      {$\theta$}
     **else**
      {$(\theta)_{y:=y-x}$}
        $y \leftarrow y - x$
      {$\theta$}
     **endif**
  {$\theta$}
  {$\theta$}
 **endwhile**
{$((\neg(x \neq 0 \wedge y \neq 0)) \wedge \theta)$}
```

Fig. 6.3. Annotated program $\mathfrak{d}_0$.

which is shown in full in Figure 6.4, and $\mathsf{VC}(\mathfrak{a})$ consists of the following formulas:

$$\begin{aligned}
&((x \neq 0 \wedge y \neq 0 \wedge \theta) \rightarrow ((x > y \rightarrow (\theta)_{x:=x-y}) \wedge \\
&\qquad\qquad\qquad\qquad ((\neg(x > y)) \rightarrow (\theta)_{y:=y-x}))), \\
&(\theta \rightarrow \theta), \\
&(((\neg(x \neq 0 \wedge y \neq 0)) \wedge \theta) \rightarrow ((x = 0 \rightarrow y = \gcd(x_\star, y_\star)) \wedge \\
&\qquad\qquad\qquad\qquad ((\neg(x = 0)) \rightarrow x = \gcd(x_\star, y_\star)))), \\
&((x \geq 0 \wedge y \geq 0 \wedge x = x_\star \wedge y = y_\star) \rightarrow \theta), \\
&(z = \gcd(x_\star, y_\star) \rightarrow z = \gcd(x_\star, y_\star)).
\end{aligned}$$

The nontautological verification conditions from the list above are:

$$\begin{aligned}
&((x \neq 0 \wedge y \neq 0 \wedge (x \geq 0 \wedge y \geq 0 \wedge \gcd(x, y) = \gcd(x_\star, y_\star))) \rightarrow \\
&\quad ((x > y \rightarrow (x - y \geq 0 \wedge y \geq 0 \wedge \gcd(x - y, y) = \gcd(x_\star, y_\star))) \wedge \\
&\quad ((\neg(x > y)) \rightarrow (x \geq 0 \wedge y - x \geq 0 \wedge \gcd(x, y - x) = \gcd(x_\star, y_\star))))), \\
&(((\neg(x \neq 0 \wedge y \neq 0)) \wedge (x \geq 0 \wedge y \geq 0 \wedge \gcd(x, y) = \gcd(x_\star, y_\star))) \rightarrow \\
&\quad ((x = 0 \rightarrow y = \gcd(x_\star, y_\star)) \wedge \\
&\quad ((\neg(x = 0)) \rightarrow x = \gcd(x_\star, y_\star)))), \\
&((x \geq 0 \wedge y \geq 0 \wedge x = x_\star \wedge y = y_\star) \rightarrow \\
&\quad (x \geq 0 \wedge y \geq 0 \wedge \gcd(x, y) = \gcd(x_\star, y_\star))).
\end{aligned}$$

```
{$(x \geq 0 \wedge y \geq 0 \wedge x = x_\star \wedge y = y_\star)$}
{$\theta$}
 while $(x \neq 0 \wedge y \neq 0)$ do
  {$(x \neq 0 \wedge y \neq 0 \wedge \theta)$}
  {$((x > y \to (\theta)_{x:=x-y}) \wedge ((\neg(x > y)) \to \quad (\theta)_{y:=y-x}))$}
   if $x > y$
     then
      {$(\theta)_{x:=x-y}$}
        $x \leftarrow x - y$
      {$\theta$}
     else
      {$(\theta)_{y:=y-x}$}
        $y \leftarrow y - x$
      {$\theta$}
     endif
  {$\theta$}
  {$\theta$}
 endwhile
{$(\neg(x \neq 0 \wedge y \neq 0) \wedge \theta)$};
{$((x = 0 \to y = \gcd(x_\star, y_\star)) \wedge (\neg(x = 0) \to x = \gcd(x_\star, y_\star)))$}
 if $x = 0$
   then
  {$y = \gcd(x_\star, y_\star)$}
    $z \leftarrow y$
  {$z = \gcd(x_\star, y_\star)$}
 else
  {$x = \gcd(x_\star, y_\star)$}
    $z \leftarrow x$
  {$z = \gcd(x_\star, y_\star)$}
 endif
{$z = \gcd(x_\star, y_\star)$}
{$z = \gcd(x_\star, y_\star)$}
```

Fig. 6.4. Annotated program $\mathfrak{a}$.

The argument that these formulas are valid in $\mathcal{A}'$ has been essentially made in Example 6.5.7 and therefore, by Theorem 6.7.25, we have another argument that $\mathrm{Th}^{\mathcal{L}'}(\mathcal{A}') \vdash_{\mathcal{HH}^{\mathcal{L}'}} \mathcal{H}$, and therefore, by soundness of $\mathcal{HH}^{\mathcal{L}'}$, $\mathcal{A}' \models \mathcal{H}$.

To increase readability of annotated programs, we will replace systematically subsequences of the form $\{\alpha\}\{\alpha\}$ or $\{\alpha\};\{\alpha\}$ with $\{\alpha\}$ and $;\{\alpha\}$, respectively. Since such subsequences yield only tautological verification conditions of the form $(\alpha \to \alpha)$, we will ignore these associated verification conditions. With this notational convention, the previous annotated program is given in Figure 6.5.

$\square$

```
{$(x \geq 0 \wedge y \geq 0 \wedge x = x_\star \wedge y = y_\star)$}
{$\theta$}
 while $(x \neq 0 \wedge y \neq 0)$ do
  {$(x \neq 0 \wedge y \neq 0 \wedge \theta)$}
  {$((x > y \rightarrow (\theta)_{x:=x-y}) \wedge ((\neg(x > y)) \rightarrow (\theta)_{y:=y-x}))$}
   if $x > y$
     then
      {$(\theta)_{x:=x-y}$}
        $x \leftarrow x - y$
      {$\theta$}
     else
      {$(\theta)_{y:=y-x}$}
        $y \leftarrow y - x$
      {$\theta$}
     endif
  {$\theta$}
 endwhile
{$(\neg(x \neq 0 \wedge y \neq 0) \wedge \theta)$};
{$((x = 0 \rightarrow y = \gcd(x_\star, y_\star)) \wedge (\neg(x = 0) \rightarrow x = \gcd(x_\star, y_\star)))$}
 if $x = 0$
   then
  {$y = \gcd(x_\star, y_\star)$}
    $z \leftarrow y$
  {$z = \gcd(x_\star, y_\star)$}
 else
  {$x = \gcd(x_\star, y_\star)$}
    $z \leftarrow x$
  {$z = \gcd(x_\star, y_\star)$}
 endif
{$z = \gcd(x_\star, y_\star)$}
```

Fig. 6.5. Simplified annotated program $\mathfrak{a}$.

Example 6.7.31. We wish to build an annotated program whose target is the triple

$$\mathcal{H}_0 = \{(x = x_\star \wedge y = y_\star)\}\mathfrak{p}\{z = x_\star^{y_\star}\},$$

where $\mathfrak{p}$ is the program discussed in Examples 6.2.13, 6.3.14, and 6.5.8. We begin by choosing the formula

$$\theta = (\exists \ell)(x = x_\star^{2^\ell} \wedge y = y_\star \operatorname{div} 2^\ell \wedge z = x_\star^{y_\star \bmod 2^\ell})$$

as an invariant for the **while** loop. Pulling up this invariant through the body of the **while** loop gives the annotated program of Figure 6.6.

Next, we preface this annotated program with $\{(y \neq 0 \wedge \theta)\}$ and place the newly obtained annotated program inside the **while** loop to obtain the annotated program shown in Figure 6.7.

```
{$(((d \neq 0 \to (\theta)_{x:=x*x,z:=z*x})\wedge$
$((\neg(d \neq 0)) \to (\theta)_{x:=x*x,z:=z})))_{y:=y \text{ div } 2,d:=y \text{ mod } 2}$}
$d \leftarrow y$ mod 2;
{$(((d \neq 0 \to (\theta)_{x:=x*x,z:=z*x})\wedge$
 $((\neg(d \neq 0)) \to (\theta)_{x:=x*x,z:=z})))_{y:=y \text{ div } 2}$}
$y \leftarrow y$ div 2;
{$((d \neq 0 \to (\theta)_{x:=x*x,z:=z*x}) \wedge ((\neg(d \neq 0)) \to (\theta)_{x:=x*x,z:=z}))$}
 **if** $d \neq 0$
   **then**
    {$(\theta)_{x:=x*x,z:=z*x}$}
      $z \leftarrow z * x$
    {$(\theta)_{x:=x*x}$}
   **else**
    {$(\theta)_{x:=x*x,z:=z}$}
      $z \leftarrow z$
    {$(\theta)_{x:=x*x}$}
 **endif**;
{$(\theta)_{x:=x*x}$}
  $x \leftarrow x * x$
{$\theta$}
```

Fig. 6.6. Initial annotated program for $\mathcal{H}_0$.

```
{$\theta$}
 **while** $y \neq 0$ **do**
  {$(y \neq 0 \wedge \theta)$}
  {$(((d \neq 0 \to (\theta)_{x:=x*x,z:=z*x})\wedge$
   $((\neg(d \neq 0)) \to (\theta)_{x:=x*x,z:=z})))_{y:=y \text{ div } 2,d:=y \text{ mod } 2}$}
     $d \leftarrow y$ mod 2;
  {$(((d \neq 0 \to (\theta)_{x:=x*x,z:=z*x})\wedge$
   $((\neg(d \neq 0)) \to (\theta)_{x:=x*x,z:=z})))_{y:=y \text{ div } 2}$}
     $y \leftarrow y$ div 2;
  {$((d \neq 0 \to (\theta)_{x:=x*x,z:=z*x}) \wedge ((\neg(d \neq 0)) \to (\theta)_{x:=x*x,z:=z}))$}
   **if** $d \neq 0$
     **then**
      {$(\theta)_{x:=x*x,z:=z*x}$}
        $z \leftarrow z * x$
      {$(\theta)_{x:=x*x}$}
       **else**
      {$(\theta)_{x:=x*x,z:=z}$}
        $z \leftarrow z$
      {$(\theta)_{x:=x*x}$}
   **endif**;
  {$(\theta)_{x:=x*x}$}
    $x \leftarrow x * x$
  {$\theta$}
 **endwhile**
{$((\neg(y \neq 0)) \wedge \theta)$}
```

Fig. 6.7. Intermediate annotated program for $\mathcal{H}_0$.

```
$\{(x = x_\star \wedge y = y_\star)\}$
$\{(\theta)_{z:=1}\}$
  $z \leftarrow 1;$
$\{\theta\}$
 while $y \neq 0$ do
  $\{(y \neq 0 \wedge \theta)\}$
  $\{(((d \neq 0 \rightarrow (\theta)_{x:=x*x,z:=z*x}) \wedge$
    $((\neg(d \neq 0)) \rightarrow (\theta)_{x:=x*x,z:=z})))_{y:=y \text{ div } 2, d:=y \bmod 2}\}$
     $d \leftarrow y \bmod 2;$
  $\{(((d \neq 0 \rightarrow (\theta)_{x:=x*x,z:=z*x}) \wedge$
    $((\neg(d \neq 0)) \rightarrow (\theta)_{x:=x*x,z:=z})))_{y:=y \text{ div } 2}\}$
     $y \leftarrow y \text{ div } 2;$
  $\{((d \neq 0 \rightarrow (\theta)_{x:=x*x,z:=z*x}) \wedge ((\neg(d \neq 0)) \rightarrow (\theta)_{x:=x*x,z:=z}))\}$
   if $d \neq 0$
     then
      $\{(\theta)_{x:=x*x,z:=z*x}\}$
        $z \leftarrow z * x$
      $\{(\theta)_{x:=x*x}\}$
       else
      $\{(\theta)_{x:=x*x,z:=z}\}$
        $z \leftarrow z$
      $\{(\theta)_{x:=x*x}\}$
   endif;
  $\{(\theta)_{x:=x*x}\}$
    $x \leftarrow x * x$
  $\{\theta\}$
 endwhile
$\{((\neg(y \neq 0)) \wedge \theta)\}$
$\{z = x_\star^{y_\star}\}$
```

Fig. 6.8. Final annotated program for $\mathcal{H}_0$.

Finally, we pull θ through $z \leftarrow 1$ and we add the pre- and postconditions to get the annotated program of Figure 6.8. The nontrivial verification conditions for this annotated program are:

$$
\begin{aligned}
&((x = x_\star \wedge y = y_\star) \rightarrow (\theta)_{z:=1}) \\
&((y \neq 0 \wedge \theta) \rightarrow (((d \neq 0 \rightarrow (\theta)_{x:=x*x,z:=z*x}) \wedge \\
&\qquad\qquad ((\neg(d \neq 0)) \rightarrow (\theta)_{x:=x*x,z:=z})))_{y:=y \text{ div } 2, d:=y \bmod 2}) \\
&(((\neg(y \neq 0)) \wedge \theta) \rightarrow z = x_\star^{y_\star}).
\end{aligned}
$$

Applying now the necessary substitutions, these verification conditions become:

$$((x = x_\star \wedge y = y_\star) \to (\exists \ell)(x = x_\star^{2^\ell} \wedge y = y_\star \operatorname{div} 2^\ell \wedge 1 = x_\star^{y_\star \bmod 2^\ell})$$

$$\begin{aligned}
&((y \neq 0 \wedge (\exists \ell)(x = x_\star^{2^\ell} \wedge y = y_\star \operatorname{div} 2^\ell \wedge z = x_\star^{y_\star \bmod 2^\ell})) \to\\
&\quad ((y \bmod 2 \neq 0 \to\\
&\qquad (\exists \ell)(x * x = x_\star^{2^\ell} \wedge y \operatorname{div} 2 = y_\star \operatorname{div} 2^\ell \wedge z * x = x_\star^{y_\star \bmod 2^\ell}))\wedge\\
&\quad ((\neg(y \bmod 2 \neq 0)) \to\\
&\qquad (\exists \ell)(x * x = x_\star^{2^\ell} \wedge y \operatorname{div} 2 = y_\star \operatorname{div} 2^\ell \wedge z = x_\star^{y_\star \bmod 2^\ell}))
\end{aligned}$$

$$\begin{aligned}
&(((\neg(y \neq 0)) \wedge (\exists \ell)(x = x_\star^{2^\ell} \wedge y = y_\star \operatorname{div} 2^\ell \wedge z = x_\star^{y_\star \bmod 2^\ell})) \to\\
&\qquad z = x_\star^{y_\star}).
\end{aligned}$$

The validity of these formulas in the structure $\mathcal{A}'$ introduced in Example 6.5.8 has been discussed in that example. Thus, we have another proof that $\mathcal{A}' \models \mathcal{H}_0$. □

The "pulling up" process we applied in Examples 6.7.27 to 6.7.31 involves at times "creative" steps, namely, when we have to deal with **while** loop fragments. In such cases, we devised loop invariants and used these invariants to produce nontrivial verification conditions.

In Figure 6.9, we review succinctly the pulling up technique. In Figure 6.9(a), the process has reached the end of the **while** loop. At stage (b), we invent the loop invariant φ and introduce the bracketed assertion $\{((\neg\beta) \wedge \varphi)\}$, which yields the nontrivial verification condition $(((\neg\beta)\wedge\varphi) \to \theta)$. At stage (c), the invariant is placed below the body $\mathfrak{p}$ of the **while** loop. Next, at stage (d), the invariant has been pulled through $\mathfrak{p}$ to yield the bracketed assertion $\{\alpha\}$ and we introduce the bracketed assertion $\{(\beta \wedge \varphi)\}$. This creates another nontrivial verification condition $((\beta \wedge \varphi) \to \alpha)$. Finally, we obtain the annotated program shown in Figure 6.9(e).

The previous discussion points out two ways of obtaining nontrivial verification conditions linked to a **while** loop. The third way of generating such a verification condition of an annotated program for a triple $\{\varphi\}\mathfrak{p}\{\psi\}$ is to pull the postcondition ψ through the

```
while $\beta$ do
$\mathfrak{p}$
endwhile;
$\{\theta\}$
$\mathfrak{q}$
$\{\psi\}$
```

(a)

```
while $\beta$ do
$\mathfrak{p}$
endwhile
$\{((\neg\beta) \wedge \varphi)\}$;
$\{\theta\}$
$\mathfrak{q}$
$\{\psi\}$
```

(b)

```
while $\beta$ do
$\mathfrak{p}$
$\{\varphi\}$
endwhile
$\{((\neg\beta) \wedge \varphi)\}$;
$\{\theta\}$
$\mathfrak{q}$
$\{\psi\}$
```

(c)

```
while $\beta$ do
$\{(\beta \wedge \varphi)\}$
$\{\alpha\}$
$\mathfrak{p}$
$\{\varphi\}$
endwhile
$\{((\neg\beta) \wedge \varphi)\}$;
$\{\theta\}$
$\mathfrak{q}$
$\{\psi\}$
```

(d)

```
$\{\varphi\}$
while $\beta$ do
$\{(\beta \wedge \varphi)\}$
$\{\alpha\}$
$\mathfrak{p}$
$\{\varphi\}$
endwhile
$\{((\neg\beta) \wedge \varphi)\}$;
$\{\theta\}$
$\mathfrak{q}$
$\{\psi\}$
```

(e)

Fig. 6.9. Step-by-step construction of an annotated program.

program $\mathfrak{p}$ to obtain the annotated program

$$\begin{array}{c}\{\psi'\}\\ \vdots \\ \{\psi\}\end{array}$$

and prepending it with the bracketed assertion $\{\varphi\}$. This creates the verification condition $(\varphi \to \psi')$.

6.8 Exercises and Supplements

The $\texttt{WHILE}_{\mathcal{L}}$ Programming Language–Syntax

(1) If q is a prefix of an $\mathcal{L}$-program, prove that

$$|q|_{\textbf{if}} \geq |q|_{\textbf{then}} \geq |q|_{\textbf{else}} \geq |q|_{\textbf{endif}}$$
$$|q|_{\textbf{while}} \geq |q|_{\textbf{endwhile}}.$$

Conclude, using Lemma 6.2.7 that if r is a suffix of an $\mathcal{L}$-program, then

$$|r|_{\mathbf{if}} \leq |r|_{\mathbf{then}} \leq |r|_{\mathbf{else}} \leq |r|_{\mathbf{endif}}$$
$$|r|_{\mathbf{while}} \leq |r|_{\mathbf{endwhile}}.$$

(2) Prove that if (s, i) is an occurrence of a symbol s in an $\mathcal{L}$-program $\mathfrak{p}$, then $\mathrm{lev}_{\mathfrak{p}}(s, i) \geq 0$.

(3) Prove that for every occurrence ($\mathbf{if}$,i) in an $\mathcal{L}$-program $\mathfrak{p}$, we have $\mathrm{lev}_{\mathfrak{p}}(\mathbf{if}, i) \geq \mathit{0}$; similarly, for every occurrence $(\mathbf{while}, i)$, we have $\mathrm{lev}_{\mathfrak{p}}(\mathbf{while}, i) \geq 0$. Further, show that if $\mathfrak{p}$ is an atomic program, the level of an occurrence of either **if** or **while** is greater than 0 unless the occurrence is the first symbol of the program, in which case the level is equal to 0.

(4) Prove that for every occurrence $(\mathbf{endif}, i)$ in an $\mathcal{L}$-program $\mathfrak{p}$, we have $\mathrm{lev}_{\mathfrak{p}}(\mathbf{endif}, i) \geq 1$; similarly, for every occurrence $(\mathbf{endwhile}, i)$, we have $\mathrm{lev}_{\mathfrak{p}}(\mathbf{endwhile}, i) \geq 1$. Further, show that if $\mathfrak{p}$ is an atomic program, the level of an occurrence of either **endif** or **endwhile** is greater than 1 unless the occurrence is the last symbol of the program, in which case the level is equal to 1.

(5) Prove that no proper prefix (proper suffix) of an atomic $\mathcal{L}$-program $\mathfrak{q}$ that is not a single variable is a suffix (prefix) of an $\mathcal{L}$-program $\mathfrak{p}$.
Infer that no proper prefix (proper suffix) of an atomic $\mathcal{L}$-program is an $\mathcal{L}$-program.
Solution: We provide the solution for proper prefixes, that is, we prove that no proper prefix of an atomic $\mathcal{L}$-program that is distinct from a single variable can be a suffix of an $\mathcal{L}$-program. The argument is by induction on the program $\mathfrak{p}$.
For the basis step, we have $\mathfrak{p} = v{\leftarrow}t$, where v is a variable and t is an $(\mathcal{L}, \mathrm{PVAR})$-term. Let q be a suffix of $\mathfrak{p}$ that is not a single variable and suppose that q is a proper prefix of an atomic $\mathcal{L}$-program $\mathfrak{q}$. Since q does not contain **if** or **while**, $\mathfrak{q}$ must have the form $v'{\leftarrow}t'$. Because q is not a single variable, we have $q = v'{\leftarrow}q'$, where $q' = \lambda$ or q' is a proper prefix of t'. Since q is a suffix of $v{\leftarrow}t$, and t does not contain $\leftarrow$, we must $v' = v$ and $q' = t$. But then, either $t = \lambda$ which is impossible, or $t = q'$, where q' is proper prefix of t'. This violates Lemma 1.5.11.

There are several inductive steps depending on the nature of $\mathfrak{p}$. Suppose initially that

$$\mathfrak{p} = \textbf{ while } \beta \textbf{ do } \mathfrak{p}_0 \textbf{ endwhile}$$

where the statement holds for $\mathfrak{p}_0$. Let q be a suffix of $\mathfrak{p}$ that is a proper prefix of an atomic program $\mathfrak{q}$. There are several subcases to consider:

- If $q = \lambda$, then q is not a proper prefix of $\mathfrak{q}$.
- If $q =$ **endwhile**, then $\mathfrak{q}$ starts with **endwhile**, which is impossible.
- If $q = q'$ **endwhile** where q' is a nonnull suffix of $\mathfrak{p}_0$, then q' is a proper prefix of $\mathfrak{q}$, which violates the inductive hypothesis, unless q' is a single variable. In the latter case, the second symbol of $\mathfrak{q}$ would be **endwhile**, which is impossible.
- If $q =$ **do** $\mathfrak{p}_0$ **endwhile**, then $\mathfrak{q}$ starts with **do**, which is impossible.
- Suppose $q = q'$ **do** $\mathfrak{p}_0$ **endwhile**, where q' is a nonnull suffix of β. Then, since β does not contain either **if** or **while**, $\mathfrak{q}$ must be an assignment statement, but also contains **endwhile**, which is impossible.
- Finally, suppose that $q = \mathfrak{p}$. Then, the level of the final **endwhile** of $\mathfrak{p}$ is the same in both $\mathfrak{p}$ and $\mathfrak{q}$. This violates the conclusion of Exercise 4 because the level of this **endwhile** in $\mathfrak{p}$ is 1 and the level of the same is greater than 1 in $\mathfrak{q}$.

The case when $\mathfrak{p}$ is an **if**-statement is similar to the previous case.

Consider now the case when $\mathfrak{p} = \mathfrak{p}_0; \mathfrak{p}_1$, where the assumption holds for $\mathfrak{p}_0$ and $\mathfrak{p}_1$. Let q be a suffix of $\mathfrak{p}$ that is not a single variable and is a proper prefix of an atomic program $\mathfrak{q}$. There are again several subcases to consider:

- If q is a suffix of $\mathfrak{p}_1$, this would violate the inductive hypothesis.
- If q is $;\mathfrak{p}_1$, we obtain a contradiction because no atomic program begins with a semicolon.
- Suppose that $q = q'; \mathfrak{p}_1$, where q' is a nonnull suffix of $\mathfrak{p}_0$. If q' is not a single variable, then this violates the inductive hypothesis because q' is a proper prefix of $\mathfrak{q}$ and a suffix of $\mathfrak{p}_0$. If q' is a single variable, then the second symbol of $\mathfrak{q}$ is ";", which is impossible.

We leave the similar result involving proper suffixes of an atomic program to the reader.

(6) Prove that if $(;, i)$ is an occurrence at level 0 in an $\mathcal{L}$-program $\mathfrak{p}$, then $\mathfrak{p}$ can be written as $\mathfrak{p} = \mathfrak{p}_0; \mathfrak{p}_1$, where $|\mathfrak{p}_0| = i$ and $\mathfrak{p}_0, \mathfrak{p}_1$ are $\mathcal{L}$-programs.

(7) Prove that if an $\mathcal{L}$-program $\mathfrak{r}$ is a proper suffix of an $\mathcal{L}$-program $\mathfrak{p}$, then there is an $\mathcal{L}$-program $\mathfrak{q}$ such that $\mathfrak{p} = \mathfrak{q}; \mathfrak{r}$.
Also, prove that if an $\mathcal{L}$-program $\mathfrak{r}$ is a proper prefix of an $\mathcal{L}$-program $\mathfrak{p}$, then there is an $\mathcal{L}$-program $\mathfrak{q}$ such that $\mathfrak{p} = \mathfrak{r}; \mathfrak{q}$.
Solution: For the first part, we use induction on $\mathfrak{p}$. If $\mathfrak{p}$ is atomic, by Supplement 5, no proper suffix of $\mathfrak{p}$ is a program, so the result is vacuously true.
Suppose that $\mathfrak{p} = \mathfrak{p}'; \mathfrak{p}''$, where $\mathfrak{p}', \mathfrak{p}''$ are $\mathcal{L}$-programs for which the inductive hypothesis holds. If $\mathfrak{r}$ is a proper suffix of $\mathfrak{p}''$, then by inductive hypothesis, there is an $\mathcal{L}$-program $\mathfrak{q}'$ with $\mathfrak{p}'' = \mathfrak{q}'; \mathfrak{r}$. Defining $\mathfrak{q} = \mathfrak{p}'; \mathfrak{q}'$, it follows that $\mathfrak{q}$ is an $\mathcal{L}$-program and $\mathfrak{q}; \mathfrak{r} = \mathfrak{p}'; \mathfrak{q}'; \mathfrak{r} = \mathfrak{p}'; \mathfrak{p}'' = \mathfrak{p}$.
If $\mathfrak{r} = \mathfrak{p}''$, then let $\mathfrak{q} = \mathfrak{p}'$. In this case, we have $\mathfrak{q}; \mathfrak{r} = \mathfrak{p}'; \mathfrak{p}'' = \mathfrak{p}$.
If $\mathfrak{p}''$ is a proper suffix of $\mathfrak{r}$, say $\mathfrak{r} = \mathfrak{r}'; \mathfrak{p}''$, we will show that the level in $\mathfrak{r}$ of the semicolon following $\mathfrak{r}'$ is 0. Observe that by Lemma 6.2.7

$$|\mathfrak{p}''|_{\mathbf{if}} = |\mathfrak{p}''|_{\mathbf{endif}}, |\mathfrak{p}''|_{\mathbf{while}} = |\mathfrak{p}''|_{\mathbf{endwhile}},$$
$$|\mathfrak{r}|_{\mathbf{if}} = |\mathfrak{r}|_{\mathbf{endif}}, |\mathfrak{r}|_{\mathbf{while}} = |\mathfrak{r}|_{\mathbf{endwhile}},$$

so we have

$$|\mathfrak{r}'|_{\mathbf{if}} = |\mathfrak{r}'|_{\mathbf{endif}}, |\mathfrak{r}'|_{\mathbf{while}} = |\mathfrak{r}'|_{\mathbf{endwhile}}.$$

Thus, the level in $\mathfrak{r}$ of the semicolon following $\mathfrak{r}'$ is 0. By Exercise 6, $\mathfrak{r}'$ is an $\mathcal{L}$-program. Since $\mathfrak{r}'$ is a proper suffix of $\mathfrak{p}'$, by inductive hypothesis, there is an $\mathcal{L}$-program $\mathfrak{q}$ with $\mathfrak{p}' = \mathfrak{q}; \mathfrak{r}'$. We have $\mathfrak{q}; \mathfrak{r} = \mathfrak{q}; \mathfrak{r}'; \mathfrak{p}'' = \mathfrak{p}'; \mathfrak{p}'' = \mathfrak{p}$, as desired.
The argument for the second part is left to the reader.

(8) Prove that if we replace **endwhile** and **endif** in Definition 6.2.3 by **end**, then the new definition of $\mathtt{WHILE}^{at}_{\mathcal{L}}$ and $\mathtt{WHILE}_{\mathcal{L}}$ still has the unique readability property.

(9) Prove the following analog for programs of the Occurrence Theorem for first-order formulas:
Let $\mathfrak{p}$, $\mathfrak{q}$, $\mathfrak{r}$ be three programs in $\mathtt{WHILE}_{\mathcal{L}}$ and let β be an $\mathcal{L}$-formula.

If $\mathfrak{p}$ is an assignment statement and $\mathfrak{r} \neq \mathfrak{p}$, then $\mathfrak{r}$ does not occur in $\mathfrak{p}$.
If $\mathfrak{r}$ is distinct from $\mathfrak{p}_0 = $ **if** β **then** $\mathfrak{p}$ **else** $\mathfrak{q}$ **endif**, then every occurrence of $\mathfrak{r}$ in $\mathfrak{p}_0$ is part of either $\mathfrak{p}$ or $\mathfrak{q}$. (More exactly, each occurrence of $\mathfrak{r}$ in $\mathfrak{p}_0$ is either part of the occurrence $(\mathfrak{p}, 2 + |\beta|)$ or part of the occurrence $(\mathfrak{q}, 3 + |\beta| + |\mathfrak{p}|)$.)
If $\mathfrak{r}$ is distinct from **while** β **do** $\mathfrak{p}$ **endwhile**, then every occurrence of $\mathfrak{r}$ in **while** β **do** $\mathfrak{p}$ **endwhile** is part of $\mathfrak{p}$. (More exactly, each occurrence of $\mathfrak{r}$ in **while** β **do** $\mathfrak{p}$ **endwhile** is part of the occurrence $(\mathfrak{p}, 2 + |\beta|)$.
If $\mathfrak{r} \neq \mathfrak{p}; \mathfrak{q}$, then every occurrence $(\mathfrak{r}, i)$ in $\mathfrak{p}; \mathfrak{q}$ falls under one of the following two cases:

(a) $(\mathfrak{r}, i)$ is part of either $\mathfrak{p}$ or $\mathfrak{q}$. (More exactly, each occurrence of $\mathfrak{r}$ in $\mathfrak{p}; \mathfrak{q}$ is either part of the occurrence $(\mathfrak{p}, 0)$ or part of the occurrence $(\mathfrak{q}, 1 + |\mathfrak{p}|)$.)
(b) There are $\mathcal{L}$-programs $\mathfrak{s}, \mathfrak{s}'$ such that $\mathfrak{p}; \mathfrak{q} = \mathfrak{s}; \mathfrak{r}; \mathfrak{s}'$ and $i = |\mathfrak{s}| + 1$, or $\mathfrak{p}; \mathfrak{q} = \mathfrak{s}; \mathfrak{r}$ and $i = |\mathfrak{s}| + 1$, or $\mathfrak{p}; \mathfrak{q} = \mathfrak{r}; \mathfrak{s}'$ and $i = 0$.

Solution: If $\mathfrak{p}$ is the assignment statement $v \leftarrow t$ and $\mathfrak{r}$ occurs in $\mathfrak{p}$, then $\mathfrak{r}$ must be an assignment statement $v' \leftarrow t'$, since $\mathfrak{p}$ does not contain **if**, **while** or ;. Thus the two occurrences of $\leftarrow$ have to match and this implies that $v' = v$ and t' is a prefix of t. Since no proper prefix of a term is a term, it follows that $t' = t$ and $\mathfrak{r} = \mathfrak{p}$.
If $\mathfrak{r}$ occurs in $\mathfrak{p}_0 = $ **if** β **then** $\mathfrak{p}$ **else** $\mathfrak{q}$ **endif**, we consider several cases:

- If $\mathfrak{r}$ begins with the initial **if**, then since $\mathfrak{r} \neq \mathfrak{p}_0$, a proper prefix of an atomic program is a program and this violates Supplement 5.
- If $\mathfrak{r}$ begins within β, it must extend beyond β because every program contains an occurrence of $\leftarrow$. Therefore, $\mathfrak{r}$ contains an occurrence of **then** without a preceding occurrence of **if**, which is impossible.
- It is impossible for $\mathfrak{r}$ to start with **then**, **else**, or **endif**
- If $\mathfrak{r}$ starts in $\mathfrak{p}$, then $\mathfrak{r}$ must end in $\mathfrak{p}$ because otherwise $\mathfrak{r}$ has a prefix that consists of a suffix of $\mathfrak{p}$ and **else**. This is impossible because by the second part Exercise 1, the number of occurrences of **else** in the prefix is more than the number of occurrences of **if**, which violates the first part of Exercise 1.

- If $\mathfrak{r}$ starts in $\mathfrak{q}$, then it ends in $\mathfrak{q}$ by an argument similar to the one used in the previous part.

When $\mathfrak{r}$ occurs in **while** β **do** $\mathfrak{p}$ **endwhile**, the argument is similar to the previous one.
Suppose $\mathfrak{r}$ occurs in $\mathfrak{p};\mathfrak{q}$. If $\mathfrak{r}$ occurs in either $\mathfrak{p}$ or $\mathfrak{q}$, then we are done, so suppose that $\mathfrak{r}$ starts in $\mathfrak{p}$ and ends in $\mathfrak{q}$, say, $\mathfrak{r} = \mathfrak{r}_0;\mathfrak{r}_1$, where $\mathfrak{r}_0$ is a suffix of $\mathfrak{p}$ and $\mathfrak{r}_1$ is a prefix of $\mathfrak{q}$. Since $\mathfrak{r}_0$ is a prefix of $\mathfrak{r}$, by Exercise 1 $|\mathfrak{r}_0|_{\textbf{if}} \geq |\mathfrak{r}_0|_{\textbf{endif}}$ and $|\mathfrak{r}_0|_{\textbf{while}} \geq |\mathfrak{r}_0|_{\textbf{endwhile}}$. Since $\mathfrak{r}_0$ is also a suffix of $\mathfrak{p}$, by the same exercise, $|\mathfrak{r}_0|_{\textbf{if}} \leq |\mathfrak{r}_0|_{\textbf{endif}}$ and $|\mathfrak{r}_0|_{\textbf{while}} \leq |\mathfrak{r}_0|_{\textbf{endwhile}}$, which allows us to conclude that $|\mathfrak{r}_0|_{\textbf{if}} = |\mathfrak{r}_0|_{\textbf{endif}}$ and $|\mathfrak{r}_0|_{\textbf{while}} = |\mathfrak{r}_0|_{\textbf{endwhile}}$ and $\text{lev}_{\mathfrak{r}}(;, |\mathfrak{r}_0|) = 0$. By Exercise 6, $\mathfrak{r} = \mathfrak{r}_0';\mathfrak{r}_1'$, where $\mathfrak{r}_0'$ and $\mathfrak{r}_1'$ are $\mathcal{L}$-programs and $|\mathfrak{r}_0'| = |\mathfrak{r}_0|$, which implies that $\mathfrak{r}_0$ and $\mathfrak{r}_1$ are $\mathcal{L}$-programs.
If $\mathfrak{r}_0 = \mathfrak{p}$ and $\mathfrak{r}_1 = \mathfrak{q}$, then $\mathfrak{r} = \mathfrak{r}_0;\mathfrak{r}_1 = \mathfrak{p};\mathfrak{q}$. If $\mathfrak{r}_0 = \mathfrak{p}$ and $\mathfrak{r}_1$ is a proper prefix of $\mathfrak{q}$, then, by Supplement 7, there is an $\mathcal{L}$-program $\mathfrak{s}'$ with $\mathfrak{q} = \mathfrak{r}_1;\mathfrak{s}'$, so $\mathfrak{p};\mathfrak{q} = \mathfrak{r}_0;\mathfrak{r}_1;\mathfrak{s}' = \mathfrak{r};\mathfrak{s}'$.
If $\mathfrak{r}_0$ is a proper suffix of $\mathfrak{p}$ and $\mathfrak{r}_1 = \mathfrak{q}$, then by the same Supplement 7, there is an $\mathcal{L}$-program $\mathfrak{s}$ with $\mathfrak{p} = \mathfrak{s};\mathfrak{r}_0$, so $\mathfrak{p};\mathfrak{q} = \mathfrak{s};\mathfrak{r}_0;\mathfrak{r}_1 = \mathfrak{s};\mathfrak{r}$.
Finally, if $\mathfrak{r}_0$ is a proper suffix of $\mathfrak{p}$ and $\mathfrak{r}_1$ is a proper prefix of $\mathfrak{q}$, then, as above, $\mathfrak{p} = \mathfrak{s};\mathfrak{r}_0$, $\mathfrak{q} = \mathfrak{r}_1;\mathfrak{s}'$ and $\mathfrak{p};\mathfrak{q} = \mathfrak{s};\mathfrak{r};\mathfrak{s}'$.

(10) Let $\mathfrak{p}, \mathfrak{q}, \mathfrak{r}$ be $\mathcal{L}$-programs. Prove that if $(\mathfrak{q}, i)$ is an occurrence of $\mathfrak{q}$ in $\mathfrak{p}$, then `replace` $(\mathfrak{p}, (\mathfrak{q}, i), \mathfrak{r})$ is an $\mathcal{L}$-program.

(11) Prove that every $\mathcal{L}$-program $\mathfrak{p}$ can be written uniquely as

$$\mathfrak{p} = \mathfrak{r}_0; \cdots ; \mathfrak{r}_{m-1}$$

where $m \geq 1$ and $\mathfrak{r}_0, \ldots, \mathfrak{r}_{m-1}$ are atomic $\mathcal{L}$-programs.
Solution: The existence of a decomposition $\mathfrak{p} = \mathfrak{r}_0; \cdots ; \mathfrak{r}_{m-1}$ is easily shown by induction on programs. We prove next the uniqueness of this decomposition by induction on m. For the basis step, $m = 1$, suppose that we also have $\mathfrak{p} = \mathfrak{r}_0'; \cdots ; \mathfrak{r}_{\ell-1}'$, where $\mathfrak{r}_j'$ is an atomic $\mathcal{L}$-program for $0 \leq j \leq \ell - 1$. If $\ell > 1$, then $\mathfrak{r}_0'$ is a proper prefix of the atomic program $\mathfrak{r}_0$ which is impossible by Supplement 5. Thus, $\ell = 1$ and $\mathfrak{r}_0' = \mathfrak{p} = \mathfrak{r}_0$.
For the inductive step, assume that $m > 1$, the result is true for $m-1$, and let $\mathfrak{p} = \mathfrak{r}_0; \cdots ; \mathfrak{r}_{m-1}$. Suppose that we also have $\mathfrak{p} = \mathfrak{r}_0'; \cdots ; \mathfrak{r}_{\ell-1}'$, where $\mathfrak{r}_0', \ldots, \mathfrak{r}_{\ell-1}'$ are atomic $\mathcal{L}$-programs. Since no atomic program can be a proper prefix of another

atomic program, by Supplement 5, we must have $\mathfrak{r}_0 = \mathfrak{r}'_0$ and hence $\mathfrak{r}_1; \cdots ; \mathfrak{r}_{m-1} = \mathfrak{r}'_1; \cdots ; \mathfrak{r}'_{\ell-1}$. By inductive hypothesis, we have $\ell = m$ and $\mathfrak{r}_h = \mathfrak{r}'_h$ for $1 \leq h \leq m-1$.

(12) Prove that if $(\mathfrak{r}, i)$ is an occurrence of an $\mathcal{L}$-program $\mathfrak{r}$ in an $\mathcal{L}$-program $\mathfrak{p} = \mathfrak{p}_0; \cdots ; \mathfrak{p}_{m-1}$, where $m \geq 1$ and $\mathfrak{p}_i$ is an atomic $\mathcal{L}$-program for $0 \leq i \leq m-1$, then either the occurrence of $\mathfrak{r}$ is part of one the $\mathfrak{p}_i$ or there are $0 \leq j < k \leq m-1$ such that $\mathfrak{r} = \mathfrak{p}_j; \cdots ; \mathfrak{p}_k$.

Solution: If the occurrence of $\mathfrak{r}$ is part of one of the atomic programs $\mathfrak{p}_i$, we are done. Otherwise, since a program can neither start nor end with ";", there are j, k with $0 \leq j < k \leq m-1$ such that the occurrence of $\mathfrak{r}$ starts in $\mathfrak{p}_j$ and ends in $\mathfrak{p}_k$.

If the occurrence of $\mathfrak{r}$ does not start with the first symbol of $\mathfrak{p}_j$, then a proper suffix of $\mathfrak{p}_j$ is a prefix of $\mathfrak{r}$. If the proper suffix is not a single variable, this contradicts Supplement 5 and if the suffix is a single variable, then the second symbol of $\mathfrak{r}$ would be a semicolon, which is impossible. Thus, the occurrence of $\mathfrak{r}$ starts with the first symbol of $\mathfrak{p}_j$.

By a similar argument, the end of the occurrence of $\mathfrak{r}$ is the last symbol of $\mathfrak{p}_k$. Thus, $\mathfrak{r} = \mathfrak{p}_j; \cdots ; \mathfrak{p}_k$.

(13) Let $\mathcal{L}$ be a first-order language and **repeat**, **until**, and **endrepeat** be three new program symbols. Define inductively the set $\texttt{R-WHILE}_{\mathcal{L}}$ by replacing $\texttt{WHILE}_{\mathcal{L}}$ with $\texttt{R-WHILE}_{\mathcal{L}}$ in the rules of Definition 6.2.1 and by adding the following rule:

> If $\mathfrak{p}$ belongs to $\texttt{R-WHILE}_{\mathcal{L}}$, and β is an $(\mathcal{L}, \text{PVAR})$-formula that is quantifier-free, then
>
> $$\textbf{repeat}\mathfrak{p}\ \textbf{until}\ \beta\ \textbf{endrepeat}$$
>
> belongs to $\texttt{R-WHILE}_{\mathcal{L}}$.

Prove that any sequence in $\texttt{R-WHILE}_{\mathcal{L}}$ has the same number of occurrences of the symbols **repeat**, **until** and **endrepeat**.

(14) Let $\mathcal{L}$ be a first-order language. Define the sets of sequences $\texttt{R-WHILE}^{at}_{\mathcal{L}}$ and $\texttt{R-WHILE}'_{\mathcal{L}}$ by replacing $\texttt{WHILE}^{at}_{\mathcal{L}}$ by $\texttt{R-WHILE}^{at}_{\mathcal{L}}$ and $\texttt{WHILE}'_{\mathcal{L}}$ by $\texttt{R-WHILE}'_{\mathcal{L}}$ in Definition 6.2.3 and by adding the rule:

> If $\mathfrak{p}$ belongs to $\texttt{R-WHILE}'_{\mathcal{L}}$ and β is an $(\mathcal{L}, \text{PVAR})$-formula that is quantifier-free, then
>
> $$\textbf{repeat}\mathfrak{p}\ \textbf{until}\ \beta\ \textbf{endrepeat}$$
>
> belongs to $\texttt{R-WHILE}^{at}_{\mathcal{L}}$.

(a) Show that if $\mathfrak{p}, \mathfrak{p}' \in \texttt{R-WHILE}'_{\mathcal{L}}$, then $\mathfrak{p}; \mathfrak{p}' \in \texttt{R-WHILE}'_{\mathcal{L}}$.
(b) Prove that $\texttt{R-WHILE}'_{\mathcal{L}} = \texttt{R-WHILE}_{\mathcal{L}}$.

(15) Extend the definition of the level of an occurrence (s, i) of a symbol s in a program $\mathfrak{p} \in \texttt{R-WHILE}_{\mathcal{L}}$ by

$$\operatorname{lev}_{\mathfrak{p}}(s,i) = |q|_{\mathbf{while}} - |q|_{\mathbf{endwhile}} + |q|_{\mathbf{if}} - |q|_{\mathbf{endif}} + |q|_{\mathbf{repeat}} - |q|_{\mathbf{endrepeat}},$$

where $q = \mathrm{PREF}(\mathfrak{p}, i)$. Prove that

(a) the level of any occurrence of a semicolon in $\mathfrak{p}$ is nonnegative;
(b) if $\mathfrak{p}$ is atomic, the level of any occurrence of a semicolon in $\mathfrak{p}$ is positive;
(c) the level of any occurrence of **else** in $\mathfrak{p}$ is positive.

(16) Prove that the definition of the sets $\texttt{R-WHILE}^{at}_{\mathcal{L}}$ and $\texttt{R-WHILE}'_{\mathcal{L}} = \texttt{R-WHILE}_{\mathcal{L}}$ given in Exercise 14 meets the unique readability condition.

(17) Show that if (**repeat**$, i$) is an occurrence in a program $\mathfrak{p} \in \texttt{R-WHILE}_{\mathcal{L}}$, then there is a program $\mathfrak{q} \in \texttt{R-WHILE}_{\mathcal{L}}$ which is a **repeat**-statement such that $(\mathfrak{q}, i)$ is an occurrence in $\mathfrak{p}$. Formulate and prove similar statements involving the symbols **if** and **while**.

(18) Reformulate Exercises 1 to 12 for $\texttt{R-WHILE}_{\mathcal{L}}$-programs.

The $\texttt{WHILE}_{\mathcal{L}}$ Programming Language–Semantics

Two programs $\mathfrak{p}, \mathfrak{q} \in \texttt{WHILE}_{\mathcal{L}}$ are $\mathcal{A}$-*equivalent* if $\mathcal{S}^{pr}_{\mathcal{A}}(\mathfrak{p}) = \mathcal{S}^{pr}_{\mathcal{A}}(\mathfrak{q})$. The programs $\mathfrak{p}, \mathfrak{q}$ are *equivalent* if they are $\mathcal{A}$-equivalent for every $\mathcal{L}$-structure $\mathcal{A}$.

(19) Give an example of two programs $\mathfrak{p}, \mathfrak{q}$ and a structure $\mathcal{A}$ such that the programs are $\mathcal{A}$-equivalent but not equivalent.

(20) Replacing an occurrence of a program $\mathfrak{p}$ in another program $\mathfrak{q}$ with an equivalent program $\mathfrak{p}'$ does not change the meaning of $\mathfrak{q}$. This analog of the Replacement Theorem for formulas is formally stated next.
Let $\mathfrak{q}$ be an $\mathcal{L}$-program. Prove that if $\mathfrak{p}, \mathfrak{p}'$ are $\mathcal{A}$-equivalent programs and $(\mathfrak{p}, i)$ is an occurrence of $\mathfrak{p}$ in $\mathfrak{q}$, then $\texttt{replace}\,(\mathfrak{q}, (\mathfrak{p}, i), \mathfrak{p}')$ is $\mathcal{A}$-equivalent to $\mathfrak{q}$.

Conclude that if $\mathfrak{p}, \mathfrak{p}'$ are equivalent programs and $(\mathfrak{p}, i)$ is an occurrence of $\mathfrak{p}$ in $\mathfrak{q}$, then `replace` $(\mathfrak{q}, (\mathfrak{p}, i), \mathfrak{p}')$ is equivalent to $\mathfrak{q}$.

(21) Prove for R-WHILE$_{\mathcal{L}}$-programs an analog of the Replacement Theorem for WHILE$_{\mathcal{L}}$-programs given in Exercise 20.

(22) Let β be a quantifier-free $(\mathcal{L}, \mathrm{PVAR})$-formula and let $\mathfrak{p}$ be an $\mathcal{L}$-program. Define the programs

$$\begin{aligned} \mathfrak{q}_0 = {} & \textbf{while } \beta \textbf{ do } \mathfrak{p} \textbf{ endwhile} \\ \mathfrak{q}_1 = {} & \textbf{if } \beta \textbf{ then } \mathfrak{p};\ \textbf{while } \beta \textbf{ do } \mathfrak{p} \textbf{ endwhile else} \\ & \quad v \leftarrow v \textbf{ endif} \end{aligned}$$

where v is a fixed program variable. Prove that for all $\mathcal{L}$-structures $\mathcal{A}$, $\mathcal{S}^{pr}_{\mathcal{A}}(\mathfrak{q}_0) = \mathcal{S}^{pr}_{\mathcal{A}}(\mathfrak{q}_1)$.

(23) Let $\mathfrak{p}_{pred}$ be the WHILE$_{\mathcal{L}_{pra}}$-program:

```
$z \leftarrow 0$;
 **while** $(s(z) < x)$
    **do** $z \leftarrow s(z)$
 **endwhile**
```

(a) Verify that if $\sigma \in \mathrm{STATES}_{\mathcal{A}_{pra}}$ is a state such that $\sigma(x) = 5$, then $\sigma' = \mathcal{S}^{pr}_{\mathcal{A}_{pra}}(\mathfrak{p}_{pred})(\sigma)$ is defined and $\sigma'(z) = 4$.

(b) Justify that $\mathfrak{p}_{pred}$ always halts in a state σ' when started in an $\mathcal{A}_{pra}$-state σ and that the value of $\sigma'(z)$ is $\sigma(x) - 1$, if $\sigma(x) > 0$ and $\sigma'(z) = 0$ if $\sigma(x) = 0$.

(c) Evaluate $\mathsf{time}^{pr}_{\mathcal{A}_{pra}}(\mathfrak{p}_{pred})(\sigma)$ in terms of $\sigma(x)$.

(24) Let $\mathfrak{p}_{div}$ be the WHILE$_{\mathcal{L}_{ar}}$-program

```
**if** $y = 0$
     **then** $z \leftarrow 0$
     **else** $z \leftarrow 0$;
            **while** $y \cdot s(z) \leq x$
               **do** $z \leftarrow s(z)$
            **endwhile**
**endif**
```

(a) Verify that if $\sigma \in \mathrm{STATES}_{\mathcal{A}_{ar}}$ is a state such that $\sigma(x) = 7$ and $\sigma(y) = 2$, then $\sigma' = \mathcal{S}^{pr}_{\mathcal{A}_{ar}}(\mathfrak{p}_{div})(\sigma)$ is defined and $\sigma'(z) = 3$.

(b) Justify that $\mathfrak{p}_{div}$ always halts in a state σ' when started in an $\mathcal{A}_{ar}$-state σ and that the value of $\sigma'(z)$ is $\lfloor \sigma(x)/\sigma(y) \rfloor$ if $\sigma(y) > 0$, and $\sigma'(z) = 0$ if $\sigma(y) = 0$.

(c) Evaluate $\mathsf{time}^{pr}_{\mathcal{A}_{pra}}(\mathfrak{p}_{div})(\sigma)$ in terms of $\sigma(x)$ and $\sigma(y)$.

(25) Let $\mathfrak{p}_{sqrt}$ be the $\texttt{WHILE}_{\mathcal{L}_{pra}}$-program

```
$z \leftarrow 0$;
$w \leftarrow s(0)$;
$u \leftarrow s(0)$;
 **while** $u \leq x$
     **do** $w \leftarrow s(s(w))$;
        $z \leftarrow s(z)$;
        $u \leftarrow u + w$
 **endwhile**
```

(a) Verify that if $\sigma \in \mathrm{STATES}_{\mathcal{A}_{pra}}$ is a state such that $\sigma(x) = 10$, then $\sigma' = \mathcal{S}^{pr}_{\mathcal{A}_{pra}}(\mathfrak{p}_{sqrt})(\sigma)$ is defined and $\sigma'(z) = 3$.

(b) Justify that $\mathfrak{p}_{sqrt}$ always halts in a state σ' when started in an $\mathcal{A}_{pra}$-state σ and that the value of $\sigma'(z)$ is $\lfloor \sqrt{\sigma(x)} \rfloor$.

(c) Evaluate $\mathsf{time}^{pr}_{\mathcal{A}_{pra}}(\mathfrak{p}_{sqrt})(\sigma)$ in terms of $\sigma(x)$.

(26) Let $\mathcal{L}$ be a first-order language and let $\mathcal{A}$ be a $\mathcal{L}$-structure. We extend the semantics of $\texttt{WHILE}_{\mathcal{L}}$-programs to accommodate the **repeat** construct introduced in Exercise 13 by adding the following item to Definition 6.3.2:

> If $\mathfrak{p}_0$ is the program **repeat**$\mathfrak{p}$ **until** β **endrepeat**, where β is a quantifier-free formula, $\mathfrak{p}$ is in $\texttt{R-WHILE}_{\mathcal{L}}$, and $\sigma \in \mathrm{STATES}_{\mathcal{A}}$, let $k \in \mathbf{N}$ be the least number at least equal to 1 such that there is a state $\sigma_k = \left(\mathcal{S}^{pr}_{\mathcal{A}}(\mathfrak{p})\right)^{(k)}(\sigma)$ and $\mathcal{S}^{for}_{\mathcal{A}}(\beta)(\sigma_k) = \mathbf{T}$, if such a number exists. When the number k exists, $\mathcal{S}^{at\text{-}pr}_{\mathcal{A}}(\mathfrak{p}_0)(\sigma) = \sigma_k$; otherwise, $\mathcal{S}^{at\text{-}pr}_{\mathcal{A}}(\mathfrak{p}_0)(\sigma)$ is undefined.

This definition captures the intuitive semantics of the **repeat** construct: the program $\mathfrak{p}$ is executed at least once and its execution is repeated until β is true.

Let $\mathfrak{p}_1 \in \texttt{R-WHILE}_{\mathcal{L}}$ be $\mathfrak{p}$; **while** $(\neg\beta)$ **do** $\mathfrak{p}$ **endwhile**. Prove that $\mathcal{S}^{pr}_{\mathcal{A}}(\mathfrak{p}_0) = \mathcal{S}^{pr}_{\mathcal{A}}(\mathfrak{p}_1)$.

(27) Prove that Theorems 6.3.3 through 6.3.8 remain valid when instead of $\texttt{WHILE}_{\mathcal{L}}$-programs we consider $\texttt{R-WHILE}_{\mathcal{L}}$-programs.

(28) Show that $\texttt{R-WHILE}_{\mathcal{L}}$-programs can be effectively transformed into equivalent $\texttt{WHILE}_{\mathcal{L}}$-programs.
Hint. Proceed by induction on the number of occurrences of the symbol **repeat**, using each time the rightmost occurrence of this symbol. (See also Exercises 17, 26, and 21.)

(29) Definition 6.3.9 is extended to $\texttt{R-WHILE}_{\mathcal{L}}$-programs by adding the following item:

Let $\mathfrak{p}_0 =$ **repeat**$\mathfrak{p}$ **until** β **endrepeat**, where β is a quantifier-free formula, $\mathfrak{p} \in \texttt{R-WHILE}_{\mathcal{L}}$, and $\sigma \in \text{STATES}_{\mathcal{A}}$. Let $k \in \mathbf{N}$ be the least positive number such that there is a state $\sigma_k = \left(\mathcal{S}^{pr}_{\mathcal{A}}(\mathfrak{p})\right)^{(k)}(\sigma)$ and $\mathcal{S}^{for}_{\mathcal{A}}(\beta)(\sigma_k) = \mathbf{T}$, if such a number exists. When the number k exists,

$$\mathsf{time}^{at\text{-}pr}_{\mathcal{A}}(\mathfrak{p}_0)(\sigma) = k + \sum_{i=0}^{k-1} \mathsf{time}^{pr}_{\mathcal{A}}(\mathfrak{p})(\sigma_i);$$

otherwise, $\mathsf{time}^{at\text{-}pr}_{\mathcal{A}}(\mathfrak{p}_0)(\sigma)$ is undefined.

Prove that Theorems 6.3.10 and 6.3.11 remain valid when $\texttt{WHILE}_{\mathcal{L}}$-programs are replaced by $\texttt{R-WHILE}_{\mathcal{L}}$-programs.

Functions Computable by Programs

(30) Let $\mathsf{pred} : \mathbf{N} \longrightarrow \mathbf{N}$, be the function defined by

$$\mathsf{pred}(n) = \begin{cases} 0 & \text{if } n = 0 \\ n - 1 & \text{otherwise.} \end{cases}$$

Prove that the function pred is computed by the program $\mathfrak{p}_{pred}$ given in Exercise 23 with the sequence of variables (x, z).

(31) Let $\mathsf{div} : \mathbf{N}^2 \longrightarrow \mathbf{N}$, be the function defined by

$$\mathsf{div}(n,p) = \begin{cases} 0 & \text{if } p = 0 \\ \lfloor n/p \rfloor & \text{otherwise.} \end{cases}$$

Prove that the function div is computed by the program $\mathfrak{p}_{div}$ given in Exercise 24 with the sequence of variables (x, y, z).

(32) Let $\mathsf{sqrt} : \mathbf{N} \longrightarrow \mathbf{N}$, be the function defined by

$$\mathsf{sqrt}(n) = \lfloor \sqrt{n} \rfloor.$$

Prove that the function sqrt is computed by the program $\mathfrak{p}_{sqrt}$ given in Exercise 25 with the sequence of variables (x, z).

(33) Let $\mathcal{L}$ be a first-order language, $\mathcal{A}$ be an $\mathcal{L}$-structure, and let $n, i \in \mathbf{N}$ be such that $0 \leq i \leq n-1$. The ith *projection* is the function $p^{\mathcal{A}}_{n,i}$ given by $p^{\mathcal{A}}_{n,i}(a_0, \ldots, a_{n-1}) = a_i$ for all $a_0, \ldots, a_{n-1} \in |\mathcal{A}|$.
Prove that every projection is computable in $\mathcal{A}$ by an $\mathcal{L}$-program with a suitable sequence of variables.

Let $\mathcal{L}$ be a first-order language, $\mathcal{A}$ be an $\mathcal{L}$-structure, and let $h : |\mathcal{A}|^n \rightsquigarrow |\mathcal{A}|$, $g_i : |\mathcal{A}|^m \rightsquigarrow |\mathcal{A}|$ be partial functions on $|\mathcal{A}|$ for $0 \leq i \leq n-1$, where $n, m \in \mathbf{N}$. Define the function $\ell : |\mathcal{A}|^m \rightsquigarrow |\mathcal{A}|$ by

$$\ell(a_0, \ldots, a_{m-1}) = h(g_0(a_0, \ldots, a_{m-1}), \ldots, g_{n-1}(a_0, \ldots, a_{m-1}))$$

for $a_0, \ldots, a_{m-1} \in |\mathcal{A}|$. Of course, for $\ell(a_0, \ldots, a_{m-1})$ to be defined, all values in the right side of the above equality must be defined. We refer to the function ℓ as the *composition* of $h, g_0, \ldots, g_{n-1}$ and we denote ℓ by $h(g_0, \ldots, g_{n-1})$.

(34) Prove that if $h, g_0, \ldots, g_{n-1}$ are partial functions computable in an $\mathcal{L}$-structure $\mathcal{A}$ with suitable sequences of variables, then the partial function $\ell = h(g_0, \ldots, g_{n-1})$ is computable in $\mathcal{A}$ with a suitable sequence of variables.
Solution: Let $y_0, \ldots, y_{m-1}, z_n$ be distinct program variables. If $n = 0$, by Corollary 6.4.10, there is an $\mathcal{L}$-program $\mathfrak{q}$ that computes h with (z_0). Then, $\mathfrak{q}$ computes ℓ with $(y_0, \ldots, y_{m-1}, z_0)$. Suppose now that $n > 0$. Let z_{n-1} be a program variable distinct from $y_0, \ldots, y_{m-1}, z_n$. By Corollary 6.4.11, we may assume

that the function g_{n-1} is computed by $\mathfrak{p}_{n-1}$ with the sequence of variables $(y_0, \ldots, y_{m-1}, z_{n-1})$ preserving inputs. Again, using the same corollary, we may assume that g_{n-2} is computed by $\mathfrak{p}_{n-2}$ with the sequence of variables $(y_0, \ldots, y_{m-1}, z_{n-2})$ preserving inputs, where z_{n-2} is a program variable that does not occur in $\mathfrak{p}_{n-1}$ and is distinct from $y_0, \ldots, y_{m-1}, z_{n-1}, z_n$.
In general, we may assume that g_{n-i} is computed by $\mathfrak{p}_{n-i}$ with the sequence of variables $(y_0, \ldots, y_{m-1}, z_{n-i})$ preserving inputs, where z_{n-i} does not occur in the programs $\mathfrak{p}_{n-i+1}, \ldots, \mathfrak{p}_{n-1}$ and is distinct from $y_0, \ldots, y_{m-1}, z_{n-i+1}, \ldots, z_n$. In this manner, we construct a sequence $(z_0, \ldots, z_{n-1})$ of distinct program variables and we may assume now that the function h is computed by $\mathfrak{q}$ with the sequence $(z_0, \ldots, z_{n-1}, z_n)$. Then, the following program computes the function ℓ with the sequence of variables $(y_0, \ldots, y_{m-1}, z_n)$:

$$\begin{array}{l} \mathfrak{p}_0; \\ \vdots \\ \mathfrak{p}_{n-1}; \\ \mathfrak{q} \end{array}$$

Let m be a natural number and suppose that $g : \mathbf{N}^m \rightsquigarrow \mathbf{N}$ and $h : \mathbf{N}^{m+2} \rightsquigarrow \mathbf{N}$. Then, there is a unique function $\ell : \mathbf{N}^{m+1} \rightsquigarrow \mathbf{N}$ that satisfies

$$\ell(a_0, \ldots, a_{m-1}, 0) = g(a_0, \ldots, a_{m-1}) \text{ for all } a_0, \ldots, a_{m-1} \text{ in } \mathbf{N},$$
$$\ell(a_0, \ldots, a_{m-1}, n+1) = h(a_0, \ldots, a_{m-1}, n, \ell(a_0, \ldots, a_{m-1}, n))$$
$$\text{for all } a_0, \ldots, a_{m-1}, n \text{ in } \mathbf{N},$$

where the recursion equations are taken to mean that

(1) for all $a_0, \ldots, a_{m-1}$ in $\mathbf{N}$, $\ell(a_0, \ldots, a_{m-1}, 0)$ is defined if and only if $g(a_0, \ldots, a_{m-1})$ is defined, and if $\ell(a_0, \ldots, a_{m-1}, 0)$ is defined, then the given equality holds, and
(2) for all $a_0, \ldots, a_{m-1}, n$ in $\mathbf{N}$, $\ell(a_0, \ldots, a_{m-1}, n+1)$ is defined if and only if $\ell(a_0, \ldots, a_{m-1}, n)$ and $h(a_0, \ldots, a_{m-1}, n, \ell(a_0, \ldots, a_{m-1}, n))$ are both defined, and if $\ell(a_0, \ldots, a_{m-1}, n+1)$ is defined, then the given equality holds.

We say that ℓ is defined by primitive recursion from g and h.

(35) Recall that we introduced the $\mathcal{L}_s$-structure $\mathcal{A}_s$ as the reduct of the $\mathcal{A}_{ar}$ to the language $\mathcal{L}_s = \{=, 0, s\}$. Prove that the set of functions computable in $\mathcal{A}_s$ is closed under primitive recursion.
Solution: Let $x_0, \dots, x_{m-1}, w, z, v$ be distinct program variables. By Corollary 6.4.11 we can assume that the program $\mathfrak{p}_g$ computes g with $(x_0, \dots, x_{m-1}, z)$ preserving inputs and that $\mathfrak{p}_h$ computes h with $(x_0, \dots, x_{m-1}, w, z, v)$ preserving inputs. Let y be a program variable distinct from $x_0, \dots, x_{m-1}, w, z$ that does not occur in $\mathfrak{p}_g$ or $\mathfrak{p}_h$. Then, the following program computes ℓ with $(x_0, \dots, x_{m-1}, y, z)$:

```
$\mathfrak{p}_g$;
$w \leftarrow 0$;
 while $w \neq y$
    do $\mathfrak{p}_h$;
       $z \leftarrow v$;
       $w \leftarrow s(w)$
 endwhile
```

Let $n \in \mathbf{N}$ and $g : \mathbf{N}^{n+1} \rightsquigarrow \mathbf{N}$. Then, recall that the function obtained from g by *minimization* is the function $\mu g : \mathbf{N}^n \rightsquigarrow \mathbf{N}$ defined by setting $\mu g(x_0, \dots, x_{n-1})$ to be the least natural number y such that $(x_0, \dots, x_{n-1}, y')$ is in the domain of g for all y' with $0 \leq y' \leq y$ and $g(x_0, \dots, x_{n-1}, y) = 0$, if such a y exists, and to be undefined otherwise.

To indicate that $\ell = \mu g$, we sometimes write

$$\ell(x_0, \dots, x_{n-1}) = \mu y[g(x_0, \dots, x_{n-1}, y) = 0]$$

and read μy as "the least y."

(36) Prove that the set of functions computable in $\mathcal{A}_s$, where $\mathcal{A}_s$ is the structure mentioned in Supplement 35, is closed under minimization.
Solution: Let $x_0, \dots, x_{n-1}, y, z$ be distinct program variables. By Corollary 6.4.11, we may assume that the program $\mathfrak{p}_g$ computes g with $(x_0, \dots, x_{n-1}, y, z)$ in $\mathcal{A}_s$ preserving inputs. Then,

the following program computes ℓ with the sequence of variables $(x_0, \ldots, x_{n-1}, y)$:

```
$y \leftarrow 0$;
$\mathfrak{p}_g$;
 **while** $z \neq 0$
    **do** $y \leftarrow s(y)$;
       $\mathfrak{p}_g$
 **endwhile**
```

The *initial functions* are the following functions:

(1) the *zero function* $Z : \mathbf{N}^0 \to \mathbf{N}$ given by $Z() = 0$;
(2) the projection functions $p_{n,i} : \mathbf{N}^n \to \mathbf{N}$;
(3) the successor function $s(x) = x + 1$.

The *partial recursive functions* are defined as follows:

(1) every initial function is partial recursive;
(2) if h and $g_0, \ldots, g_{n-1}$ are all partial recursive, and ℓ is obtained from these functions by composition, then ℓ is partial recursive;
(3) if g and h are partial recursive and ℓ is obtained from g and h by primitive recursion, then ℓ is partial recursive;
(4) If g is partial recursive and ℓ is obtained from g by minimization, then ℓ is partial recursive.

(37) Prove that every partial recursive function is computable by a program in the structure $\mathcal{A}_s$.
Hint. The statement follows immediately from Supplements 33 to 36.

For a program $\mathfrak{p} \in$ `WHILE`, denote by $\hat{\mathcal{L}}_{\mathfrak{p}}$ the set of function and relation symbols contained in $\mathfrak{p}$.

(38) Prove the following extension of Theorem 6.4.16 which uses the same notations as the theorem. There is an effectively computable function $\mathfrak{q} :$ `WHILE` $\longrightarrow$ `WHILE` such that

- for all $\mathfrak{p} \in$ `WHILE`

$$\hat{\mathcal{L}}_{\mathfrak{p}} - \{f_g\} \subseteq \hat{\mathcal{L}}_{\mathfrak{q}(\mathfrak{p})} \subseteq (\hat{\mathcal{L}}_{\mathfrak{p}} - \{f_g\}) \cup \mathcal{L};$$

- for all $\mathfrak{p} \in \mathtt{WHILE}_{\mathcal{L}'}$ and $\sigma \in \text{STATES}_{\mathcal{A}}$, $\mathcal{S}^{pr}_{\mathcal{A}'}(\mathfrak{p})(\sigma)$ is defined if and only if $\mathcal{S}^{pr}_{\mathcal{A}}(\mathfrak{q}(\mathfrak{p}))(\sigma)$ is defined and if both are defined, then these states agree on all the variables that occur in $\mathfrak{p}$.

Hint. Begin by proving suitably modified versions of Theorems 6.4.12 through 6.4.15.

Hoare Triples

(39) Let $\mathcal{L}$ be the extension of $\mathcal{L}_{ar}$ obtained by adding the binary function symbol div and the unary function symbols pred and sqrt and let $\mathcal{A}$ be the expansion of $\mathcal{A}_{ar}$ obtained by defining $\text{pred}^{\mathcal{A}} = \mathsf{pred}$, $\text{div}^{\mathcal{A}} = \mathsf{div}$, and $\text{sqrt}^{\mathcal{A}} = \mathsf{sqrt}$, where the functions $\mathsf{pred}, \mathsf{div}, \mathsf{sqrt}$ were introduced in Exercises 30 to 32.
Prove that

$$\mathcal{A} \models [x = x_\star]\mathfrak{p}_{pred}[z = \text{pred}(x_\star)],$$
$$\mathcal{A} \models [(x = x_\star \wedge y = y_\star)]\mathfrak{p}_{div}[z = \text{div}(x_\star, y_\star)],$$
$$\mathcal{A} \models [x = x_\star]\mathfrak{p}_{sqrt}[z = \text{sqrt}(x_\star)],$$

where $\mathfrak{p}_{pred}, \mathfrak{p}_{div}, \mathfrak{p}_{sqrt}$ are the programs introduced in Exercises 23 to 25.

(40) Let $\mathcal{L}$ and $\mathcal{A}$ be as in Exercise 39. Prove that

$$\mathcal{A} \models \{(x = x_\star \wedge y = y_\star)\}\mathfrak{p}\{z = \text{div}(x_\star, y_\star)\},$$

where $\mathfrak{p}$ is the program

```
$z \leftarrow 0$;
 while $y \cdot s(z) \leq x$
    do $z \leftarrow s(z)$
 endwhile
```

Also, show that $\mathcal{A} \not\models [(x = x_\star \wedge y = y_\star)]\mathfrak{p}[z = \text{div}(x_\star, y_\star)]$.

(41) Let $\mathcal{L}$ be a first-order language, $\varphi, \psi \in \text{ASSERT}_{\mathcal{L}}$, and let $\mathfrak{p} \in \mathtt{WHILE}_{\mathcal{L}}$. Prove that if $\mathcal{A}$ is an $\mathcal{L}$-structure and $\sigma \in \text{STATES}_{\mathcal{A}}$, then we have:

(a) $(\mathcal{A}, \sigma) \models [\varphi]\mathfrak{p}[\psi]$ if and only if one of the following holds:

- $(\mathcal{A}, \sigma) \not\models \{\mathsf{true}^{\mathcal{L}}_\star\}x_0 \leftarrow x_0\{\varphi\}$, or
- $(\mathcal{A}, \sigma) \models \{\varphi\}\mathfrak{p}\{\psi\}$ and $(\mathcal{A}, \sigma) \not\models \{\mathsf{true}^{\mathcal{L}}_\star\}\mathfrak{p}\{\mathsf{false}^{\mathcal{L}}_\star\}$.

(b) $(\mathcal{A}, \sigma) \models \{\varphi\}\mathfrak{p}\{\psi\}$ if and only if one of the following holds:

- $(\mathcal{A}, \sigma) \models [\varphi]\mathfrak{p}[\psi]$, or
- $(\mathcal{A}, \sigma) \not\models [\mathsf{true}_\star^{\mathcal{L}}]\mathfrak{p}[\mathsf{true}_\star^{\mathcal{L}}]$.

(42) Prove the following stronger variant of Theorem 6.5.10: Let $\mathcal{L}$ be a first-order language, $\mathcal{A}$ be an $\mathcal{L}$-structure, and $\sigma \in \mathrm{STATES}_{\mathcal{A}}$. Then, $(\mathcal{A}, \sigma) \models \{\varphi\}$ **if** β **then** $\mathfrak{p}_0$ **else** $\mathfrak{p}_1$ **endif**$\{\psi\}$ if and only if $(\mathcal{A}, \sigma) \models \{(\varphi \wedge \beta)\}\mathfrak{p}_0\{\psi\}$ and $(\mathcal{A}, \sigma) \models \{(\varphi \wedge (\neg\beta))\}\mathfrak{p}_1\{\psi\}$.

(43) Show that Theorem 6.5.10 holds if we replace $\mathtt{WHILE}_{\mathcal{L}}$-programs with $\mathtt{R\text{-}WHILE}_{\mathcal{L}}$-programs.

(44) Show that Theorem 6.5.11 holds if we replace the $\mathtt{WHILE}_{\mathcal{L}}$-program $\mathfrak{p}$ with an $\mathtt{R\text{-}WHILE}_{\mathcal{L}}$-program.

(45) Show that Thereof 6.5.18 holds if programs are replaced with repeat programs.

(46) Show that Theorem 6.5.20 and Corollary 6.5.21 hold if the partial correctness triple $\mathcal{H}$ involves $\mathtt{R\text{-}WHILE}_{\mathcal{L}}$-programs.

(47) Show that Theorem 6.5.24 holds if we replace $\mathtt{WHILE}_{\mathcal{L}}$-programs with $\mathtt{R\text{-}WHILE}_{\mathcal{L}}$-programs.

Hoare Theories

(48) Let $\mathcal{L}$ be a decidable first-order language. Prove that $\mathsf{HPT}_{\mathcal{L}}$ and $\mathsf{HTT}_{\mathcal{L}}$ are decidable subsets of HPT and HTT, respectively.

(49) Let $\mathcal{L}$ be a decidable first-order language and $\mathcal{A}$ be an $\mathcal{L}$-structure. Prove that $\mathcal{A}$ is effectively expressive if and only if there is an effectively computable function $\omega^{\mathcal{A}} : \mathtt{WHILE}_{\mathcal{L}} \times \mathrm{ASSERT}_{\mathcal{L}} \longrightarrow \mathrm{FORM}_{\mathcal{L}}$ such that for $\mathfrak{p} \in \mathtt{WHILE}_{\mathcal{L}}$ and $\psi \in \mathrm{ASSERT}_{\mathcal{L}}$, the formula $\omega^{\mathcal{A}}(\mathfrak{p}, \psi)$ expresses $\mathsf{WLP}_{\mathcal{A}}(\mathfrak{p}, \psi)$.

(50) Let $\mathcal{A}$ be an $\mathcal{L}$-structure, where $\mathcal{L}$ is a first-order language. Prove that for any $\mathcal{L}$-sentence φ, we have $\varphi \in \mathrm{Th}^{\mathcal{L}}(\mathcal{A})$ if and only if $[\mathsf{true}_\star^{\mathcal{L}_\varphi}]x_0 \leftarrow x_0[\varphi_\star] \in \mathsf{HTT}_{\mathcal{L}}(\mathcal{A})$.
Conclude that $\mathrm{Th}^{\mathcal{L}}(\mathcal{A}) \leq_m \mathsf{HTT}_{\mathcal{L}}(\mathcal{A})$.

(51) Use Exercise 50 to show that $\mathsf{HTT}_{\mathcal{L}_{ar}}(\mathcal{A}_{ar})$ is undecidable.

(52) Let $\mathcal{L}$ be a first-order language with no 0-ary relation symbols and let $\mathcal{A}$ be an effectively expressive $\mathcal{L}$-structure. Prove that $\mathsf{HPT}_{\mathcal{L}}(\mathcal{A}) \leq_m \mathrm{Th}^{\mathcal{L}}(\mathcal{A})$.
Solution: Let $\omega^{\mathcal{A}} : \mathtt{WHILE} \times \mathrm{ASSERT} \longrightarrow \mathrm{FORM}$ be as given following Definition 6.6.6. Define $\hat{\omega}^{\mathcal{A}} : \mathtt{WHILE} \times \mathrm{ASSERT} \longrightarrow$

FORM by

$$\hat{\omega}^{\mathcal{A}}(\mathfrak{p},\psi) = \begin{cases} \omega^{\mathcal{A}}(\mathfrak{p},\psi) \wedge \theta^{\mathfrak{p},\psi} & \text{if } \mathfrak{p} \text{ or } \psi \text{ contains a relation symbol} \\ & \text{of positive arity,} \\ \alpha_0 & \text{otherwise,} \end{cases}$$

where $\theta^{\mathfrak{p},\psi}$ is a logically valid formula containing all the function and relation symbols in $\mathfrak{p}$ and ψ and no others and α_0 is a fixed formula in $\mathrm{SENT} - \mathrm{SENT}_{\mathcal{L}}$. Note that $\theta^{\mathfrak{p},\psi}$ can be chosen such that $\hat{\omega}^{\mathcal{A}}$ is effectively computable.
Define the effectively computable function $f : \mathsf{HPT} \longrightarrow \mathrm{SENT}$ by

$$f(\{\varphi\}\mathfrak{p}\{\psi\}) = (\varphi \to \hat{\omega}^{\mathcal{A}}(\mathfrak{p},\psi))^{\forall}.$$

If $\{\varphi\}\mathfrak{p}\{\psi\} \in \mathsf{HPT}_{\mathcal{L}}$, then ψ contains a relation symbol of positive arity (since $\mathcal{L}$ has no 0-ary relation symbols), so $\hat{\omega}^{\mathcal{A}}(\mathfrak{p},\psi) = \omega^{\mathcal{A}}(\mathfrak{p},\psi) \wedge \theta^{\mathfrak{p},\psi} \equiv \omega^{\mathcal{A}}(\mathfrak{p},\psi)$ and $f(\{\varphi\}\mathfrak{p}\{\psi\}) \equiv (\varphi \to \omega^{\mathcal{A}}(\mathfrak{p},\psi))^{\forall}$. By Theorem 6.6.7, $\{\varphi\}\mathfrak{p}\{\psi\} \in \mathsf{HPT}_{\mathcal{L}}(\mathcal{A})$ if and only if $f(\{\varphi\}\mathfrak{p}\{\psi\}) \in \mathrm{Th}^{\mathcal{L}}(\mathcal{A})$. If $\{\varphi\}\mathfrak{p}\{\psi\} \notin \mathsf{HPT}_{\mathcal{L}}(\mathcal{A})$, then $f(\{\varphi\}\mathfrak{p}\{\psi\}) \notin \mathrm{SENT}_{\mathcal{L}}$.

The notion of weakest liberal precondition introduced in Definition 6.5.14 can be extended to $\mathtt{R\text{-}WHILE}_{\mathcal{L}}$-programs by replacing the term “$\mathtt{WHILE}_{\mathcal{L}}$-program” with “$\mathtt{R\text{-}WHILE}_{\mathcal{L}}$-program”.

The notions of *r-expressive structure* and *effectively r-expressive structure* are introduced by replacing “$\mathtt{WHILE}_{\mathcal{L}}$-program” in Definition 6.6.6 with “$\mathtt{R\text{-}WHILE}_{\mathcal{L}}$-program”.

(53) Show that Theorem 6.6.7 remains true if if the partial correctness triple $\mathcal{H}$ involves $\mathtt{R\text{-}WHILE}_{\mathcal{L}}$-programs.

(54) Prove that an $\mathcal{L}$-structure $\mathcal{A}$ is (effectively) r-expressive if and only if it is (effectively) expressive.
Hint. Use Exercise 28.

Let $\mathcal{L}$ be a first-order language, $\mathcal{A}$ be an $\mathcal{L}$-structure, φ be an $\mathcal{L}$-assertion, and $\mathfrak{p} \in \mathtt{WHILE}_{\mathcal{L}}$. The *strongest postcondition of* φ *and* $\mathfrak{p}$ is the set

$$\mathsf{STP}_{\mathcal{A}}(\varphi,\mathfrak{p}) = \{\sigma' \in \mathrm{STATES}_{\mathcal{A}} \mid \text{there is } \sigma \in \mathrm{STATES}_{\mathcal{A}} \text{ such that } (\mathcal{A},\sigma) \models \varphi \text{ and } \mathcal{S}^{pr}_{\mathcal{A}}(\mathfrak{p})(\sigma) = \sigma'\}.$$

(55) Let θ be an $\mathcal{L}$-formula that expresses the set of states $\mathsf{STP}_{\mathcal{A}}(\varphi, \mathfrak{p})$, where $\mathcal{A}$ is an $\mathcal{L}$-structure, φ is an $\mathcal{L}$-assertion, and $\mathfrak{p} \in \texttt{WHILE}_{\mathcal{L}}$. Prove that the following statements are equivalent for any $\mathcal{L}$-assertion ψ:

(a) $\mathcal{A} \models \{\varphi\}\mathfrak{p}\{\psi\}$;
(b) $\mathcal{A} \models (\theta \to \psi)$;
(c) $(\theta \to \psi)^{\forall} \in \mathrm{Th}^{\mathcal{L}}(\mathcal{A})$.

(56) Let $\mathcal{L}$ be a first-order language with equality, $\mathfrak{p} \in \texttt{WHILE}_{\mathcal{L}}$, and $\varphi \in \mathrm{FORM}_{\mathcal{L}}$ and let $v_0, \ldots, v_{n-1}$ be the variables that occur in $\mathfrak{p}$ or φ. Assume that $y_0, \ldots, y_{n-1}$ are variables that occur neither in $\mathfrak{p}$ nor in φ and let $\vec{v} = (v_0, \ldots, v_{n-1})$ and $\vec{y} = (y_0, \ldots, y_{n-1})$. Prove that if the formula θ expresses $\mathsf{WLP}_{\mathcal{A}}(\mathfrak{p}, (\neg(v_0 = y_0 \wedge \cdots \wedge v_{n-1} = y_{n-1})))$, then the formula

$$\gamma = \langle (\exists v_0) \cdots (\exists v_{n-1})(\varphi \wedge (\neg\theta)) \rangle_{\vec{y}:=\vec{v}}$$

expresses $\mathsf{STP}_{\mathcal{A}}(\varphi, \mathfrak{p})$.

Solution: We begin by observing that the following four statements are equivalent:

(i) $(\mathcal{A}, \sigma') \models \gamma$;
(ii) $(\mathcal{A}, [\vec{y} \to \sigma'(\vec{v})]\sigma') \models (\exists v_0) \cdots (\exists v_{n-1})(\varphi \wedge (\neg\theta))$, where $\sigma'(\vec{v}) = (\sigma'(v_0), \ldots, \sigma'(v_{n-1}))$;
(iii) there is $\vec{a} = (a_0, \ldots, a_{n-1}) \in |\mathcal{A}|^n$ such that $(\mathcal{A}, [\vec{v} \to \vec{a}][\vec{y} \to \sigma'(\vec{v})]\sigma') \models (\varphi \wedge (\neg\theta))$;
(iv) there is $\vec{a} = (a_0, \ldots, a_{n-1}) \in |\mathcal{A}|^n$ such that $(\mathcal{A}, [\vec{v} \to \vec{a}][\vec{y} \to \sigma'(\vec{v})]\sigma') \models \varphi$, the state

$$\sigma_1 = \mathcal{S}^{pr}_{\mathcal{A}}(\mathfrak{p})([\vec{v} \to \vec{a}][\vec{y} \to \sigma'(\vec{v})]\sigma')$$

exists and $\sigma_1(v_0) = \sigma_1(y_0), \ldots, \sigma_1(v_{n-1}) = \sigma_1(y_{n-1})$.

Suppose that $(\mathcal{A}, \sigma') \models \gamma$, so the fourth statement in the list holds. Define $\sigma = [v_0, \ldots, v_{n-1} \to a_0, \ldots, a_{n-1}]\sigma'$. Since σ coincides with the state $[\vec{v} \to \vec{a}][\vec{y} \to \sigma'(\vec{v})]\sigma'$ on the variables which appear in φ and $\mathfrak{p}$, it follows that $(\mathcal{A}, \sigma) \models \varphi$ and the state $\hat{\sigma} = \mathcal{S}^{pr}_{\mathcal{A}}(\mathfrak{p})(\sigma)$ is defined. Further, $\hat{\sigma}(v_i) = \sigma_1(v_i) = \sigma_1(y_i) = \sigma'(v_i)$ and if $z \notin \{v_0, \ldots, v_{n-1}\}$, then $\hat{\sigma}(z) = \sigma(z) = \sigma'(z)$. Thus, $\hat{\sigma} = \sigma'$ and we have shown that $\mathcal{S}^{pr}_{\mathcal{A}}(\mathfrak{p})(\sigma) = \sigma'$. Consequently, $\sigma' \in \mathsf{STP}_{\mathcal{A}}(\varphi, \mathfrak{p})$.

Conversely, suppose that $\sigma' \in \mathsf{STP}_{\mathcal{A}}(\varphi, \mathfrak{p})$, that is for some state σ such that $(\mathcal{A}, \sigma) \models \varphi$, we have $\mathcal{S}^{pr}_{\mathcal{A}}(\mathfrak{p})(\sigma) = \sigma'$. Let $a_i = \sigma(v_i)$ for $0 \leq i \leq n-1$. Let $\tilde{\sigma}$ be the state $[\vec{v} \to \vec{a}][\vec{y} \to \sigma'(\vec{v})]\sigma'$. The state $\tilde{\sigma}$ coincides with σ on the variables that occur in the program $\mathfrak{p}$ and the formula φ. Thus, $(\mathcal{A}, \tilde{\sigma}) \models \varphi$, $\sigma_1 = \mathcal{S}^{pr}_{\mathcal{A}}(\mathfrak{p})(\tilde{\sigma})$ exists and $\sigma_1(v_i) = \sigma'(v_i) = \tilde{\sigma}(y_i) = \sigma_1(y_i)$ for $0 \leq i \leq n-1$, which shows that the fourth condition holds and hence $(\mathcal{A}, \sigma) \models \gamma$.

(57) Let $\mathcal{L}$ be a first-order language with equality, $\mathfrak{p} \in \mathtt{WHILE}_{\mathcal{L}}$, and $\psi \in \mathrm{FORM}_{\mathcal{L}}$ and let $v_0, \ldots, v_{n-1}$ be the variables that occur in $\mathfrak{p}$ or φ. Assume that $y_0, \ldots, y_{n-1}$ are variables that occur neither in $\mathfrak{p}$ nor in ψ and let $\vec{v} = (v_0, \ldots, v_{n-1})$ and $\vec{y} = (y_0, \ldots, y_{n-1})$. Prove that if the formula θ expresses $\mathsf{STP}_{\mathcal{A}}(\mathfrak{p}, (v_0 = y_0 \wedge \cdots \wedge v_{n-1} = y_{n-1}))$, then the formula

$$\gamma = \langle (\forall v_0) \cdots (\forall v_{n-1})(\theta \to \psi) \rangle_{\vec{y} := \vec{v}}$$

expresses $\mathsf{WLP}_{\mathcal{A}}(\psi, \mathfrak{p})$.

Solution: Note that the following four statements are equivalent:

(i) $(\mathcal{A}, \sigma) \models \gamma$;

(ii) $(\mathcal{A}, [\vec{y} \to \sigma(\vec{v})]\sigma) \models (\forall v_0) \cdots (\forall v_{n-1})(\theta \to \psi)$;

(iii) for all $\vec{a} \in |\mathcal{A}|^n$, if $(\mathcal{A}, [\vec{v} \to \vec{a}][\vec{y} \to \sigma(\vec{v})]\sigma) \models \theta$, then $(\mathcal{A}, [\vec{v} \to \vec{a}][\vec{y} \to \sigma(\vec{v})]\sigma) \models \psi$;

(iv) for all $\vec{a} \in |\mathcal{A}|^n$, if there is $\sigma' \in \mathrm{STATES}_{\mathcal{A}}$ such that $\sigma'(v_i) = \sigma'(y_i)$ for $0 \leq i \leq n-1$ and $\mathcal{S}^{pr}_{\mathcal{A}}(\mathfrak{p})(\sigma') = [\vec{v} \to \vec{a}][\vec{y} \to \sigma(\vec{v})]\sigma$, then $(\mathcal{A}, [\vec{v} \to \vec{a}][\vec{y} \to \sigma(\vec{v})]\sigma) \models \psi$.

Suppose that $(\mathcal{A}, \sigma) \models \gamma$, so the fourth condition above holds. We must show that $\sigma \in \mathsf{WLP}_{\mathcal{A}}(\psi, \mathfrak{p})$. Assume that $\mathcal{S}^{pr}_{\mathcal{A}}(\mathfrak{p})(\sigma) = \sigma_1$ exists. We need to prove that $(\mathcal{A}, \sigma_1) \models \psi$. Let $\sigma' = [\vec{y} \to \sigma(\vec{v})]\sigma$. Observe that σ' and σ agree on all variables of $\mathfrak{p}$, that is $\sigma'(v_i) = \sigma(v_i)$, for $0 \leq i \leq n-1$. Thus, the state $\mathcal{S}^{pr}_{\mathcal{A}}(\mathfrak{p})(\sigma') = \hat{\sigma}$ exists and, furthermore, $\hat{\sigma}(z) = \sigma_1(z)$ for every variable $z \notin \{y_0, \ldots, y_{n-1}\}$. Let $\vec{a} = \hat{\sigma}(\vec{v})$. Note that $\sigma'(v_i) = \sigma(v_i) = \sigma'(y_i)$. Let $\tilde{\sigma}$ be the state $[\vec{v} \to \vec{a}][\vec{y} \to \sigma(\vec{v})]\sigma$.

We claim that $\hat{\sigma} = \tilde{\sigma}$. Observe first that $\hat{\sigma}(v_i) = a_i = \tilde{\sigma}(v_i)$. Next, we have $\hat{\sigma}(y_i) = \sigma'(y_i) = \sigma(v_i) = \tilde{\sigma}(y_i)$. Finally, if z is distinct from $v_0, \ldots, v_{n-1}, y_0, \ldots, y_{n-1}$, then $\hat{\sigma}(z) = \sigma'(z) = \sigma(z) = \tilde{\sigma}(z)$.

By the fourth condition, we have $(\mathcal{A}, \tilde{\sigma}) \models \psi$. Note that $\tilde{\sigma}(v_i) = a_i = \hat{\sigma}(v_i) = \mathcal{S}^{pr}_{\mathcal{A}}(\mathfrak{p})(\sigma')(v_i) = \mathcal{S}^{pr}_{\mathcal{A}}(\mathfrak{p})(\sigma)(v_i) = \sigma_1(v_i)$, for $0 \leq i \leq n-1$. Therefore, $(\mathcal{A}, \sigma_1) \models \psi$.
Conversely, suppose that $\sigma \in \mathsf{WLP}_{\mathcal{A}}(\mathfrak{p}, \psi)$. We will show that the fourth condition holds, and, therefore, $(\mathcal{A}, \sigma) \models \gamma$. Let $\vec{a} \in |\mathcal{A}|^n$ and suppose that there is a state σ' such that $\sigma'(v_i) = \sigma'(y_i)$, for $0 \leq i \leq n-1$ and $\mathcal{S}^{pr}_{\mathcal{A}}(\mathfrak{p})(\sigma') = [\vec{v} \to \vec{a}][\vec{y} \to \sigma(\vec{v})]\sigma$. Denote $[\vec{v} \to \vec{a}][\vec{y} \to \sigma(\vec{v})]\sigma$ as $\tilde{\sigma}$. To prove the fourth condition, we need to show that $(\mathcal{A}, \tilde{\sigma}) \models \psi$.
Observe that $\sigma'(v_i) = \sigma'(y_i) = \tilde{\sigma}(y_i) = \sigma(v_i)$, so σ and σ' agree on all variables of $\mathfrak{p}$. Therefore, $\mathcal{S}^{pr}_{\mathcal{A}}(\mathfrak{p})(\sigma) = \sigma_1$ exists and $\sigma_1(v_i) = \tilde{\sigma}(v_i)$ for $0 \leq i \leq n-1$. Since $\sigma \in \mathsf{WLP}_{\mathcal{A}}(\mathfrak{p}, \psi)$ it follows that $(\mathcal{A}, \sigma_1) \models \psi$, so $(\mathcal{A}, \tilde{\sigma}) \models \psi$.

We call an $\mathcal{L}$-structure $\mathcal{A}$ *dually expressive* if for all $\texttt{WHILE}_{\mathcal{L}}$-programs $\mathfrak{p}$ and $\mathcal{L}$-assertions φ, the set of states $\mathsf{STP}_{\mathcal{A}}(\varphi, \mathfrak{p})$ is expressible by a single formula. If this formula can be effectively constructed given $\mathfrak{p}$ and φ, we say that $\mathcal{A}$ is *effectively dually expressive.*

(58) Prove that if $\mathcal{L}$ is a language with equality, then an $\mathcal{L}$-structure is (effectively) dually expressive if and only if it is (effectively) expressive.
Hint. Use Supplements 56 and 57.

Let $\mathcal{L}$ be a first-order language, $\mathcal{A}$ be an $\mathcal{L}$-structure, $\mathfrak{p}$ be an $\mathcal{L}$-program and ψ be an $\mathcal{L}$-assertion. The *weakest precondition set* is the set $\mathsf{WP}_{\mathcal{A}}(\mathfrak{p}, \psi)$ that consists of all states $\sigma \in \mathrm{STATES}_{\mathcal{A}}$ such that $\mathcal{S}^{pr}_{\mathcal{A}}(\mathfrak{p})(\sigma)$ is defined and $(\mathcal{A}, \mathcal{S}^{pr}_{\mathcal{A}}(\mathfrak{p})(\sigma)) \models \psi$.

(59) Let $\mathcal{L}$ be a first-order language, $\mathcal{A}$ be an $\mathcal{L}$-structure and $\mathcal{H} = [\varphi]\mathfrak{p}[\psi]$ be a Hoare total correctness triple. Assume that the $\mathcal{L}$-formula $\hat{\omega}$ expresses $\mathsf{WP}_{\mathcal{A}}(\mathfrak{p}, \psi)$. Prove that the following conditions are equivalent:

(a) $\mathcal{H} \in \mathsf{HTT}_{\mathcal{L}}(\mathcal{A})$;
(b) $\mathcal{A} \models (\varphi \to \hat{\omega})$;
(c) $(\varphi \to \hat{\omega})^{\forall} \in \mathrm{Th}^{\mathcal{L}}(\mathcal{A})$.

(60) Let $\mathcal{L}$ be a first-order language, $\mathcal{A}$ be an $\mathcal{L}$-structure, $\mathfrak{p}$ be an $\mathcal{L}$-program, ψ be an $\mathcal{L}$-assertion, and let $\sigma \in \mathrm{STATES}_{\mathcal{A}}$. Prove that:

(a) $\sigma \in \mathsf{WP}_{\mathcal{A}}(\mathfrak{p}, \psi)$ if and only if

$$\sigma \in \mathsf{WLP}_{\mathcal{A}}(\mathfrak{p}, \psi) \text{ and } \sigma \notin \mathsf{WLP}_{\mathcal{A}}(\mathfrak{p}, \mathsf{false}_{\star}^{\mathcal{L}}).$$

(b) $\sigma \in \mathsf{WLP}_{\mathcal{A}}(\mathfrak{p}, \psi)$ if and only if

$$\sigma \in \mathsf{WP}_{\mathcal{A}}(\mathfrak{p}, \psi) \text{ or } \sigma \notin \mathsf{WP}_{\mathcal{A}}(\mathfrak{p}, \mathsf{true}_{\star}^{\mathcal{L}}).$$

Let $\mathcal{L}$ be a first-order language. An $\mathcal{L}$-structure $\mathcal{A}$ is *expressive for total correctness* if for all $\mathcal{L}$-programs $\mathfrak{p}$ and $\mathcal{L}$-assertions ψ, $\mathsf{WP}_{\mathcal{A}}(\mathfrak{p}, \psi)$ is expressible by a single $\mathcal{L}$-formula $\hat{\omega}^{\mathcal{A}}(\mathfrak{p}, \psi)$.

$\mathcal{A}$ is *effectively expressive for total correctness* if $\hat{\omega}^{\mathcal{A}}(\mathfrak{p}, \psi)$ can be found effectively given $\mathfrak{p}$ and ψ.

(61) Prove that an $\mathcal{L}$-structure $\mathcal{A}$ is (effectively) expressive if and only if $\mathcal{A}$ is (effectively) expressive for total correctness.

(62) Let $\mathcal{L}$ be a decidable first-order language. Prove that if an $\mathcal{L}$-structure $\mathcal{A}$ is effectively expressive for total correctness, then $\mathsf{HTT}_{\mathcal{L}}(\mathcal{A}) \leq_m \mathrm{Th}^{\mathcal{L}}(\mathcal{A})$.

(63) Let $\mathcal{L}$ be a decidable first-order language and let $\mathcal{A}$ be an effectively expressive $\mathcal{L}$-structure. Prove that

$$\mathrm{Th}^{\mathcal{L}}(\mathcal{A}) \equiv_m \mathsf{HPT}_{\mathcal{L}}(\mathcal{A}) \equiv_m \mathsf{HTT}_{\mathcal{L}}(\mathcal{A}).$$

Hint. Use Corollaries 6.6.3 and 6.6.8, and Exercises 50, 61 and 62.

(64) Let V be the alphabet $\{a, b\}$. Prove that for every V-instance $\Im$ of the PCP, there is an $\mathcal{L}_{arxl_V c_V \mathsf{b}}$-program $\mathfrak{p}_{\Im}$ constructed effectively from $\Im$ such that $\Im$ has a solution if and only if $\mathcal{A}_{arxl_V c_V \mathsf{b}} \models [\mathsf{true}_{\star}^{\mathcal{L}_{arxl_V c_V \mathsf{b}}}]\mathfrak{p}_{\Im}[\mathsf{true}_{\star}^{\mathcal{L}_{arxl_V c_V \mathsf{b}}}]$.
Conclude that $\mathsf{HTT}_{\mathcal{L}_{arxl_V c_V \mathsf{b}}}(\mathcal{A}_{arxl_V c_V \mathsf{b}})$ is undecidable.
Hint. Construct a program $\mathfrak{p}_{\Im}$ which ignores its input and searches systematically for a solution to $\Im$ such that if a solution is found, $\mathfrak{p}_{\Im}$ halts and if none is found the program cycles indefinitely. Use techniques similar to the ones applied in the proof of Theorem 6.6.24.

(65) Let $\mathcal{L}$ be a decidable first-order language, $\mathcal{A}$ be an $\mathcal{L}$-structure and let $g : |\mathcal{A}|^n \longrightarrow |\mathcal{A}|$ be computable by an $\mathcal{L}$-program $\mathfrak{r}_g$ with a sequence $(y_0, \ldots, y_n)$. Let f_g be an n-ary function symbol not in $\mathcal{L}$ and define $\mathcal{L}' = \mathcal{L} \cup \{f_g\}$. Define the expansion $\mathcal{A}'$ of $\mathcal{A}$ to $\mathcal{L}'$ by $f_g^{\mathcal{A}'} = g$.

Prove that $\mathsf{HTT}_{\mathcal{L}'}(\mathcal{A}')$ is m-reducible to $\mathsf{HTT}_{\mathcal{L}}(\mathcal{A})$, i.e.,

$$\mathsf{HTT}_{\mathcal{L}'}(\mathcal{A}') \leq_m \mathsf{HTT}_{\mathcal{L}}(\mathcal{A}).$$

Hint. Use techniques similar to the ones applied in the proof of Theorem 6.6.26.

(66) Prove that $\mathsf{HTT}_{\mathcal{L}_{pra}}(\mathcal{A}_{pra})$ is undecidable.
Hint. Use Exercises 64 and 65.

(67) Show that Theorem 6.6.26 holds for any first-order language $\mathcal{L}$.
Solution: We regard $\mathsf{HPT}^{\uparrow}_{\mathcal{L}'}(\mathcal{A}')$ and $\mathsf{HPT}^{\uparrow}_{\mathcal{L}}(\mathcal{A})$ as subsets of HPT and define $F : \mathsf{HPT} \longrightarrow \mathsf{HPT}$ as

$$F(\{\varphi\}\mathfrak{p}\{\psi\}) = \begin{cases} \mathcal{H}_0 & \text{if } \varphi \neq \mathsf{true}^{\mathcal{L}'} \text{ or } \psi \neq \mathsf{false}^{\mathcal{L}'}, \\ \{\mathsf{true}^{\mathcal{L}}\}\mathfrak{q}(\mathfrak{p})\{\mathsf{false}^{\mathcal{L}}\} & \text{otherwise,} \end{cases}$$

where $\mathcal{H}_0$ is a fixed arbitrary element of $\mathsf{HPT} - \mathsf{HPT}^{\uparrow}_{\mathcal{L}}(\mathcal{A})$ and $\mathfrak{q}$ is as in Exercise 38.

A Formal System for Hoare Triples

(68) Let $\mathcal{L}$ be a first-order language, $\mathcal{A}$ be an $\mathcal{L}$-structure, and let $\sigma \in \mathrm{STATES}_{\mathcal{A}}$. Prove that if $(\mathcal{A}, \sigma)$ satisfies the hypotheses of an instance of the rule $R^{\mathcal{L}}_{if}$, then $(\mathcal{A}, \sigma)$ satisfies the conclusion of this instance.

(69) Let $\mathcal{L}$ be the first-order language and let $\mathcal{A}$ be the $\mathcal{L}$-structure introduced in Exercise 39. Construct an annotated program for the triple $\{x = x_\star\}\mathfrak{p}_{pred}\{z = \mathrm{pred}(x_\star)\}$, where $\mathfrak{p}_{pred}$ is the program introduced in Exercise 23 and prove that $\mathcal{A}$ is a model of the verification conditions generated by this annotated program, thereby showing that the triple is valid in $\mathcal{A}$.
Hint. Use the formula $((((x = 0) \wedge (z = 0)) \vee (s(z) \leq x)) \wedge (x = x_\star))$ as a loop invariant.

(70) As in Exercise 69, let $\mathcal{L}$ be the first-order language and let $\mathcal{A}$ be the $\mathcal{L}$-structure introduced in Exercise 39. Construct an annotated program for the triple $\{(x = x_\star \wedge y = y_\star)\}\mathfrak{p}_{div}\{z = \mathrm{div}(x_\star, y_\star)\}$, where $\mathfrak{p}_{div}$ is the program introduced in Exercise 24 and prove that $\mathcal{A}$ is a model of the verification conditions generated by this annotated program, thereby showing that the triple is valid in $\mathcal{A}$.
Hint. Use the formula $((y \cdot z \leq x) \wedge (x = x_\star) \wedge (y = y_\star))$ as a loop invariant.

(71) As in Exercise 69, let $\mathcal{L}$ be the first-order language and let $\mathcal{A}$ be the $\mathcal{L}$-structure introduced in Exercise 39. Construct an annotated program for the triple $\{x = x_\star\}\mathfrak{p}_{sqrt}\{z = \text{sqrt}(x_\star)\}$, where $\mathfrak{p}_{sqrt}$ is the program introduced in Exercise 25 and prove that $\mathcal{A}$ is a model of the verification conditions generated by this annotated program, thereby showing that the triple is valid in $\mathcal{A}$.
Hint. Use the formula $((w = s(z+z)) \wedge (u = s(z)\cdot s(z)) \wedge (z\cdot z \leq x) \wedge (x = x_\star))$ as a loop invariant.

The notions of $\mathcal{L}$-partial correctness Hoare triple, satisfaction of a Hoare triple by a pair $(\mathcal{A}, \sigma)$, model of a Hoare triple, and entailment introduced in Definitions 6.5.2, 6.5.3, and 6.5.12 can be extended naturally to involve $\texttt{R-WHILE}_{\mathcal{L}}$-programs rather than $\texttt{WHILE}_{\mathcal{L}}$-programs. We shall use the term *$\mathcal{L}$-partial correctness Hoare r-triple* to refer to a triple of the form $\{\varphi\}\mathfrak{p}\{\psi\}$, when $\mathfrak{p}$ is an $\texttt{R-WHILE}_{\mathcal{L}}$-program. The set of all $\mathcal{L}$-partial correctness Hoare r-triples is denoted by $\mathsf{R\text{-}HPT}_{\mathcal{L}}$.

We define a new formal system $\mathcal{R} - \mathcal{HH}^{\mathcal{L}}$ that has as its set of objects $\text{FORM}_{\mathcal{L}} \cup \mathsf{R\text{-}HPT}_{\mathcal{L}}$. The rules of the new system are obtained by extending the applicability of the rules of $\mathcal{HH}^{\mathcal{L}}$ to the new set of objects and by adding the new rule

$$\frac{\{\varphi\}\mathfrak{p}\{\psi\}, \{(\psi \wedge (\neg\beta))\}\mathfrak{p}\{\psi\}}{\{\varphi\}\ \textbf{repeat}\mathfrak{p}\ \textbf{until}\ \beta\ \textbf{endrepeat}\{(\psi \wedge \beta)\}}\, R^{\mathcal{L}}_{repeat}$$

for $\varphi, \psi, \beta, \in \text{FORM}_{\mathcal{L}}$ and $\mathfrak{p} \in \texttt{WHILE}_{\mathcal{L}}$. We refer to this rule as the *repeat rule.*

(72) Prove the soundness of the formal system $\mathcal{R} - \mathcal{HH}^{\mathcal{L}}$, that is, prove that if Γ is a set of $\mathcal{L}$-formulas, $\theta \in \text{FORM}_{\mathcal{L}} \cup \mathsf{R\text{-}HPT}_{\mathcal{L}}$ and $\Gamma \vdash_{\mathcal{R}-\mathcal{HH}^{\mathcal{L}}} \theta$, then $\Gamma \models\!\!\!\approx \theta$.

(73) Prove the relative completeness of the formal system $\mathcal{R} - \mathcal{HH}^{\mathcal{L}}$, that is, prove that if $\mathcal{A}$ is an expressive $\mathcal{L}$-structure, $\theta \in \text{FORM}_{\mathcal{L}} \cup \mathsf{R\text{-}HPT}_{\mathcal{L}}$ and $\text{Th}^{\mathcal{L}}(\mathcal{A}) \models\!\!\!\approx \theta$, then $\text{Th}^{\mathcal{L}}(\mathcal{A}) \vdash_{\mathcal{R}-\mathcal{HH}^{\mathcal{L}}} \theta$.
Solution: The proof is similar to the proof of the relative completeness of the formal system $\mathcal{HH}^{\mathcal{L}}$ (using Exercises 43–53), with the addition of the following inductive step.
Assume that $\theta = \{\varphi\}$ **repeat**$\mathfrak{q}$ **until** β **endrepeat**$\{\psi\}$ and the inductive hypothesis holds for $\mathfrak{q}$. By Exercise 46, $\mathcal{A} \models \theta$. By Exercise 26, we also have $\mathcal{A} \models \{\varphi\}\mathfrak{q};$

while $(\neg\beta)$ **do** $\mathfrak{q}$ **endwhile**$\{\psi\}$. By Exercise 54, since $\mathcal{A}$ is expressive, there is an assertion γ that expresses $\mathsf{WLP}_{\mathcal{A}}($ **while** $(\neg\beta)$ **do** $\mathfrak{q}$ **endwhile**$, \psi)$. Further, by Exercise 47, we have $\mathcal{A} \models \{\varphi\}\mathfrak{q}\{\gamma\}$. By inductive hypothesis, it follows that $\mathrm{Th}^{\mathcal{L}}(\mathcal{A}) \vdash_{\mathcal{R}-\mathcal{HH}^{\mathcal{L}}} \{\varphi\}\mathfrak{q}\{\gamma\}$.
By Exercise 53, we have $\mathcal{A} \models \{\gamma\}$ **while** $(\neg\beta)$ **do** $\mathfrak{q}$ **endwhile**$\{\psi\}$. By Exercise 44, we have

$$\mathcal{A} \models \{(\gamma \wedge (\neg\beta))\}\mathfrak{q};\ \textbf{while}\ (\neg\beta)\ \textbf{do}\ \mathfrak{q}\ \textbf{endwhile}\{\psi\}$$

and also $\mathcal{A} \models ((\gamma \wedge (\neg(\neg\beta))) \rightarrow \psi)$, which is equivalent to $\mathcal{A} \models ((\gamma \wedge \beta) \rightarrow \psi)$. Applying again Exercise 47, we obtain $\mathcal{A} \models \{(\gamma \wedge (\neg\beta))\}\mathfrak{q}\{\gamma\}$. By the inductive hypothesis, $\mathrm{Th}^{\mathcal{L}}(\mathcal{A}) \vdash_{\mathcal{R}-\mathcal{HH}^{\mathcal{L}}} \{(\gamma \wedge (\neg\beta))\}\mathfrak{q}\{\gamma\}$. Thus, by applying the rule $R^{\mathcal{L}}_{repeat}$, we obtain

$$\mathrm{Th}^{\mathcal{L}}(\mathcal{A}) \vdash_{\mathcal{R}-\mathcal{HH}^{\mathcal{L}}} \{\varphi\}\ \textbf{repeat}\mathfrak{q}\ \textbf{until}\ \beta\ \textbf{endrepeat}\{(\gamma \wedge \beta)\}.$$

A final application of the rule $R^{\mathcal{L}}_{imp}$ yields $\mathrm{Th}^{\mathcal{L}}(\mathcal{A}) \models \theta$.

6.9 Bibliographical Comments

Basic references for this chapter are the books by Loeckx and Sieber [22] and Dijkstra [12]. The chapter by Cousot [11] in Volume B of *Handbook of Theoretical Computer Science* should also be consulted.

The field was initiated by R. W. Floyd and C. A. R. Hoare in the fundamental papers [15] and [20]. The notions of expressive structure and relative completeness are due to S. A. Cook [10].

The series of papers by Lipton [21], Clarke [8], and Clarke, German, Halpern [9] investigates the existence of sound and relatively complete Hoare logics for increasingly complex programming languages.

The paper [3] by K. R. Apt is an excellent survey of the first ten years of Hoare logic.

Bibliography

[1] (1967). From frege to gödel: A source book in mathematical logic 1879–1931, (Harvard University Press, Cambridge).

[2] (1969). Paramodulation and theorem-proving in first-order theories with equality, in D. Michie and R. Melzer (eds.), *Machine Intelligence*, Vol. 4 (Edinburg University Press), pp. 135–150.

[3] Apt, K. R. (1981). Ten years of Hoare's logic: a survey, *ACM Transactions on Programming Languages and Systems* **3**, pp. 431–483.

[4] Beth, E. W. (1955). Semantic entailment and formal derivability, *Mededelingen van de Koninklijke Nederlandse Akademie van Wetenschappen, Afdeling Letterkunde* **18**, pp. 309–342.

[5] Boolos, G. (1984). Don't eliminate cut, *Journal of Philosophical Logic* **13**, pp. 373–378.

[6] Brand, D. (1975). Proving theorems with the modification method, *SIAM Journal on Computing* **4**, pp. 412–430.

[7] Church, A. (1936). A note on the entscheidungsproblem, *Journal of Symbolic Logic* **1**, pp. 40–41.

[8] Clarke, E. M. (1979). Programming language constructs for which it is impossible to obtain good Hoare-like axioms, *Journal of ACM* **26**, pp. 129–147.

[9] Clarke, E. M., German, S. M. and Halpern, J. Y. (1983). On effective axiomatizations of Hoare logics, *Journal of the ACM* **30**, pp. 612–636.

[10] Cook, S. A. (1978). Soundness and completeness of an axiom system for program verification, *SIAM Journal on Computing* **7**, pp. 70–90.

[11] Cousot, P. (1990). Methods and logics for proving programs, in J. van Leeuwen (ed.), *Handbook of Theoretical Computer Science*, Vol. B: Formal Models and Semantics (Elsevier), pp. 841–994.

[12] Dijskstra, D. W. (1976). *A Discipline of Programming* (Prentice Hall, Englewood Cliffs, NJ).

[13] Enderton, H. B. (1972). *A Mathematical Introduction to Logic* (Academic Press, New York).

[14] Fejer, P. A. and Simovici, D. A. (1991). *Mathematical Foundations of Computer Science. Volume I: Sets, Relations, and Induction* (Springer, New York).

[15] Floyd, R. W. (1967). Assigning meanings to programs, in *Mathematical Aspects of Computer Science*, Vol. 19, pp. 19–32.

[16] Gentzen, G. (1932). Über die existenz unabhängiger axiomensysteme zu unedlichen satzsystemen, *Mathematische Annalen* **107**, pp. 329–350.

[17] Gentzen, G. (1935). Untersuchungen über das logische schliessen, *Mathematische Zeitschrift* **39**, pp. 176–210, 405–431.

[18] Gilmore, P. C. (1960). A proof method for quantification theory: Its justification and realization, *IBM journal of research and development* **4**, pp. 28–35.

[19] Herbrand, J. (1930). Recherches sur la theorie de la demonstration, *Travaux de la Societé des Sciences at des Lettres de Varsovie, Classe III, Science Mathématique et Physique* **33**.

[20] Hoare, C. A. R. (1969). An axiomatic basis for computer programming, *Communications of ACM* **12**, pp. 576–583.

[21] Lipton, R. J. (1977). A necessary and sufficient condition for the existence of Hoare logics, in *Proceedings of the 18th IEEE Symposium on Foundations of Computer Science*, pp. 1–6.

[22] Loeckx, J. and Sieber, K. (1987). *The Foundations of Program Verification*, 2nd edn., Wiley-Teubner Series in Computer Science (John Wiley and B. G. Teubner, Chichester, West Anglia and Stuttgart).

[23] Löwenheim, L. (1915). Uber möglichkeiten im relativkalkül, *Mathematische Annalen* **76**, pp. 447–470.

[24] Peterson, G. E. (1983). A technique for establishing completeness results in theorem proving with equality, *SIAM J. on Computing* **12**, pp. 82–100.

[25] Presburger, M. (1929). Über die Vollständigkeit eines gewissen Systems der Arithmetik ganzer Zahlen, in welchem die Addition als einzige Operation hervortritt, *Comptes Rendus du I congrès de Mathématiciens des Pays Slaves*, pp. 92–101.

[26] Robinson, J. A. (1965a). Automatic deduction with hyper-resolution, *International Journal of Computational Mathematics* **1**, pp. 227–234.

[27] Robinson, J. A. (1965b). A machine-oriented logic based on the resolution principle, *Journal of the ACM* **12**, pp. 23–41.

[28] Sági, G. (2010). A short proof for the completeness of paramodulation, *Bulletin of the Section of Logic* **39**, pp. 147–152.

[29] Skolem, T. (1920). Logisch-kombinatorische untersuchungen über die erfülbarkeit oder beweisbarkeit mathematischer sätze nebst einem theoreme über dichte menge, *Videnskapsselskapets skrifter, I. Matematisk-naturvidenskabelig klasse* **Klasse 6**, pp. 1–36.

[30] Skolem, T. (1934). Über die Nicht-charakterisierbarkeit der Zahlenreihe mittels endlich oder abzählbar unendlich vieler Aussagen mit ausschliesslich Zahlenvariablen, *Fundamenta Mathematicae* **23**, pp. 150–161.

[31] Smullyan, R. M. (1995). *First-Order Logic* (Dover Publications, New York).

[32] Tarski, A. (1936). Der Wahrheitsbegriff in den Formalisierten Sprachen, *Studia Philosophica* **1**, pp. 261–405.

[33] Tarski, A. (1951a). *A Decision Method for Elementary Algebra and Geometry*, 2nd edn. (University of California Press, Berkeley).

[34] Tarski, A. (1951b). *A Decision Method for Elementary Algebra and Geometry: Prepared for Publication with the Assistance of J.C.C. McKinsey* (RAND Corporation, Santa Monica, CA).

[35] Tarski, A. (1956). *Logic, Semantics, Metamathematics* (Oxford at the Clarendon Press, Oxford).

[36] Treinen, R. (1992). A new method for undecidability proofs of first order theories, *J. Symbolic Computation* **14**, pp. 437–457.

List of Notations

$F_{\mathcal{L}}$ the set of function symbols of a first-order language $\mathcal{L}$, 535
$\mathcal{L}_{ar}$ the language of arithmetic, 536
$\mathcal{L}_{pra}$ the language of Presburger arithmetic, 536
code the encoding substitution of symbols of first-order logic, 537
$\mathrm{TERM}_{\mathcal{L}}(V)$ the set of $(\mathcal{L}, V)$-terms, 538
$s^n(t)$ term obtained by iterating n times the application of the successor, 538
$\mathrm{GTERM}_{\mathcal{L}}$ the set of ground terms of the first-order language$\mathcal{L}$, 539
GTERM the set of ground terms of first-order logic, 539
$\mathrm{AFORM}_{\mathcal{L}}$ set of atomic formulas of $\mathcal{L}$, 540
$\mathrm{FORM}_{\mathcal{L}}$ the set of formulas of a first-order language $\mathcal{L}$, 540
$\mathrm{AFORM}_{\mathcal{L}}(V)$ the set of atomic formulas of $\mathcal{L}$ such that all variables that occur in the formula belong to V, 541
$\mathrm{LIT}_{\mathcal{L}}$ the set of literals of the first-order language $\mathcal{L}$, 541
$\mathrm{LIT}_{\mathcal{L}}(V)$ the set of literals of $\mathcal{L}$ such that all variables that occur in the formula belong to V, 541
LIT set of all literals, 541
$\mathrm{LIT}(V)$ set of literals using the variables of V, 541
$\mathrm{GAFORMNE}_{\mathcal{L}}$, 541
$\mathrm{GAFORM}_{\mathcal{L}}$ the set of ground atomic formulas of $\mathcal{L}$, 541
$\mathrm{GEQ}_{\mathcal{L}}$, 541
$\overline{\varphi}$ complement of a first-order formula φ, 542
$\mathsf{false}^{\mathcal{L}}$ abbreviation for the $\mathcal{L}$-formula $(\varphi_0 \wedge (\neg\varphi_0))$, 544
$\mathtt{size}(\varphi)$ the size of a first-order formula φ, 544
$\mathsf{true}^{\mathcal{L}}$ abbreviation for the $\mathcal{L}$-formula $(\varphi_0 \vee (\neg\varphi_0))$, 544

$\mathsf{SCTCUT}(\mathcal{L}, V)$ the set of strongly closed $(\mathtt{T}(\lambda), \mathcal{L}, V)$-tableau with cut $\mathtt{T}$, 916
$\mathsf{s}_t^a(\mathtt{T})$ the tableau obtained from the tableau $\mathtt{T}$ by substituting the constant symbol a by t in every formula that occurs in a node of the tableau, 917
$(\Delta)_{x:=t}$ the set of signed formulas obtained from Δ by replacing the variable x by the term t, 933
$(\Gamma)_{x:=t}$ the set of formulas obtained from Γ by replacing the variable x by the term t, 933
$\mathsf{SQT}_{\mathcal{L},V}$ set of all $(\mathcal{L}, V)$-sequents, 936
$\mathsf{SQT}_{\mathcal{L}}$, 936
$\mathsf{SQT}^{fin}{}_{\mathcal{L},V}$ set of all finite $(\mathcal{L}, V)$-sequents, 936
$\kappa_0 \cup \kappa_1$ the union of the $(\mathcal{L}, V)$-sequents κ_0, κ_1, 937
$\mathtt{sf}_{\mathcal{L},V}(\kappa)$ the set of signed $(\mathcal{L}, V)$-formulas that corresponds to an $(\mathcal{L}, V)$-sequent κ, 937
$\mathtt{sqt}_{\mathcal{L},V}(\Delta)$ the sequent that corresponds to a set of signed $(\mathcal{L}, V)$-formulas Δ, 937
$\mathcal{F}^{\text{seq,cut}}_{\mathcal{L},V}$ the formal system obtained from $\mathcal{F}^{\text{seq}}_{\mathcal{L},V}$ by adding the cut rule, 950
$\mathcal{F}^{\mathcal{L}}_{\text{fond}}$ primary system for first-order natural deduction, 954
$\text{FONDT}^{\mathcal{L}}$ the set of natural deduction trees of the first-order language $\mathcal{L}$, 956
$\Gamma \overset{\bullet}{\vdash}_{\text{fond}^{\mathcal{L}}} \varphi$ φ is $fond$-derivable from Γ, 959
$\mathsf{CLAUSES}_{\mathcal{L}}$ set of all $\mathcal{L}$-clauses, 988
$\text{LIT}(\mathcal{C})$ set of literals which appear in some clause of C, 989
$\mathtt{V}(\mathcal{C})$ set of statement variables in clauses of C, 989
$\mathtt{V}(C)$ set of variables in clause C, 989
φ_C disjunctive normal form formula that represents a clause C, 992
$\Gamma_{\mathcal{C}}$ set of formulas in disjunctive normal form that represents a set of clauses $\mathcal{C}$, 992
$\text{Res}_{\mathcal{L}}(\mathcal{C})$ $\mathcal{C}$ augmented by the set of $\mathcal{L}$-resolvents of $\mathcal{C}$, 1004
$\mathcal{R}^*_{\mathcal{L}}(\mathcal{C})$ first-order $\mathcal{R}$-resolution closure of $\mathcal{C}$, 1005
$\text{Res}^{mgu}_{\mathcal{L}}(\mathcal{C})$ $\mathcal{C}$ augmented by the set of most general resolvents of $\mathcal{C}$, 1005
$\mathcal{FRES}^{\mathcal{L}}$ $\mathcal{L}$-resolution formal system, 1009
$\mathcal{FFRES}^{\mathcal{L}}$ full $\mathcal{L}$-resolution formal system, 1010
$\mathcal{FRES}^{\mathcal{L}}_{mgu}$ most general $\mathcal{L}$-resolution formal system, 1010
$\mathcal{FRES}^{2\mathcal{L}}$ binary $\mathcal{L}$-resolution formal system, 1020

$\mathsf{HPT}^{\uparrow}_{\mathcal{L}}(\mathcal{A})$ the fragment of $\mathsf{HPT}_{\mathcal{L}}(\mathcal{A})$ that consists of triple of the form $\{\mathsf{true}^{\mathcal{L}}_{\star}\}\mathfrak{p}\{\mathsf{false}\mathcal{L}_{\star}\}$, 1204
$\mathsf{HPT}_{\mathcal{L}}(\mathcal{A})$ the Hoare $\mathcal{L}$-partial correctness theory of $\mathcal{A}$, 1204
$\mathsf{HPT}_{\mathcal{L}}(\mathcal{A})$ the Hoare $\mathcal{L}$-total correctness theory of $\mathcal{A}$, 1204
$\mathrm{PSTATES}_{\mathcal{A}}(V)$ set of partial $\mathcal{A}$-states with domain V, 1208
$\mathcal{HH}^{\mathcal{L}}$ Hilbert/Hoare-style system for partial correctness Hoare triples, 1225
$\mathsf{target}_{\mathcal{L}}(\mathfrak{c})$ target of the annotated program $\mathfrak{c}$, 1239
$\mathsf{VC}(\mathfrak{c})$ set of verification conditions for the annotated program $\mathfrak{c}$, 1240
$p^{\mathcal{A}}_{n,i}$ projection in the structure $\mathcal{A}$, 1265
$\mathsf{R\text{-}HPT}_{\mathcal{L}}$ set of all $\mathcal{L}$-partial correctness Hoare r-triples, 1277

List of Results

Index