THE LOGIC BOOK

Sixth Edition



Merrie Bergmann | James Moor | Jack Nelson

THE LOGIC BOOK

Sixth Edition

MERRIE BERGMANN *Smith College, Emerita*
JAMES MOOR *Dartmouth College*
JACK NELSON



The McGraw-Hill Companies

Connect Learn Succeed™

THE LOGIC BOOK, SIXTH EDITION

Published by McGraw-Hill, a business unit of The McGraw-Hill Companies, Inc., 1221 Avenue of the Americas, New York, NY, 10020. Copyright © 2014, 2009, 2004, 1998, 1990, and 1980 by Merrie Bergmann, James Moor, and Jack Nelson. All rights reserved. No part of this publication may be reproduced or distributed in any form or by any means, or stored in a database or retrieval system, without the prior written consent of The McGraw-Hill Companies, Inc., including, but not limited to, in any network or other electronic storage or transmission, or broadcast for distance learning.

Some ancillaries, including electronic and print components, may not be available to customers outside the United States.

This book is printed on acid-free paper.

1 2 3 4 5 6 7 8 9 0 DOC/DOC 1 0 9 8 7 6 5 4 3

ISBN 978-0-07-803841-9
MHID 0-07-803841-3

Senior Vice President, Products & Markets: *Kurt L. Strand*
Vice President, General Manager, Products & Markets: *Michael Ryan*
Vice President, Content Production & Technology Services: *Kimberly Meriwether David*
Executive Director of Development: *Lisa Pinto*
Managing Director: *William Glass*
Brand Manager: *Laura Wilk*
Marketing Specialist: *Alexandra Schultz*
Managing Editor: *Sara Jaeger*
Director, Content Production: *Terri Schiesl*
Content Project Manager: *Mary Jane Lampe*
Buyer: *Nichole Birkenholz*
Media Project Manager: *Sridevi Palani*
Cover Designer: *Studio Montage, St. Louis, MO*
Cover Image: Jacopo de'Barbari, (c.1460/70-c.1516). *Portrait of the mathematician Luca Pacioli, the "father of accounting," and an unknown young man.* Museo Nazionale di Capodimonte, Naples, Italy. © Scala / Art Resource, NY
Typeface: *10/12 ITC New Baskerville Roman*
Compositor: *Aptara®, Inc.*
Printer: *R. R. Donnelley*

Library of Congress Cataloging-in-Publication Data
(CIP has been applied for)

The Internet addresses listed in the text were accurate at the time of publication. The inclusion of a website does not indicate an endorsement by the authors or McGraw-Hill, and McGraw-Hill does not guarantee the accuracy of the information presented at these sites.

www.mhhe.com

ABOUT THE AUTHORS

MERRIE BERGMANN received her Ph.D. in philosophy from the University of Toronto. She is an emerita professor of computer science at Smith College. She is the author of *An Introduction to Many-Valued and Fuzzy Logic: Semantics, Algebras, and Derivation Systems* and has published articles in formal semantics, logic, philosophy of logic, philosophy of language, philosophy of humor, and computational linguistics. She is an avid blue-water sailor and is currently a digital nomad, living the cruising life.

JAMES MOOR received his Ph.D. in history and philosophy of science from Indiana University. He is currently Daniel P. Stone Professor of Intellectual and Moral Philosophy at Dartmouth College. He has numerous publications in philosophy of science, philosophy of mind, logic, philosophy of artificial intelligence, and computer ethics. Moor is a former editor of the journal *Minds and Machines* and a recipient of the American Philosophical Association Barwise Prize.

JACK NELSON received his Ph.D. from the University of Chicago. He was a member of the Philosophy Department of Temple University for over 25 years, 10 of them as Dean of the Graduate School. Subsequently he served as Vice Chancellor for Academic Affairs at the University of Missouri–St. Louis and later held the same position at the University of Washington, Tacoma. After retiring from the University of Washington he served for three years as Interim Chair of Philosophy at Arizona State University. He has published articles in epistemology, identity, and the philosophy of science and is co-author, with Lynn Hankinson Nelson, of *On Quine*.

CONTENTS

Preface ix

CHAPTER 1 INTRODUCTION TO DEDUCTIVE LOGIC 1
1.1 Introduction 1
1.2 Core Concepts of Deductive Logic 3
1.3 Special Cases of Logical Concepts 12

CHAPTER 2 SYNTAX AND SYMBOLIZATION 15
2.1 The Syntax of *SL* 15
2.2 Introduction to Symbolization 24
2.3 More Complex Symbolizations 37
2.4 Non-Truth-Functional Uses of Connectives 58

CHAPTER 3 SENTENTIAL LOGIC: SEMANTICS 69
3.1 Truth-Value Assignments and Truth-Tables for Sentences 69
3.2 Truth-Functional Truth, Falsity, and Indeterminacy 77
3.3 Truth-Functional Equivalence 87
3.4 Truth-Functional Consistency 92

3.5 Truth-Functional Entailment and Truth-Functional Validity 95
3.6 Truth-Functional Properties and Truth-Functional Consistency 106

CHAPTER 4 SENTENTIAL LOGIC: TRUTH-TREES 110
4.1 The Truth-Tree Method 110
4.2 Truth-Tree Rules 111
4.3 Using Truth-Trees To Test for Other Truth-Functional Properties 129

CHAPTER 5 SENTENTIAL LOGIC: DERIVATIONS 146
5.1 The Derivation System *SD* 146
5.2 Basic Concepts of *SD* 175
5.3 Strategies for Constructing Derivations in *SD* 179
5.4 The Derivation System *SD*+ 214

CHAPTER 6 SENTENTIAL LOGIC: METATHEORY 226
6.1 Mathematical Induction 226
6.2 Truth-Functional Completeness 234
6.3 The Soundness of *SD* and *SD*+ 244
6.4 The Completeness of *SD* and *SD*+ 252

CHAPTER 7 PREDICATE LOGIC: SYNTAX AND SYMBOLIZATION 262
7.1 Predicates, Singular Terms, and Quantity Expressions of English 262
7.2 The Formal Syntax of *PL* 268
7.3 Introduction To Symbolization 276
7.4 Symbolization Fine-Tuned 296
7.5 The Language *PLE* (Predicate Logic Extended) 310

CHAPTER 8 PREDICATE LOGIC: SEMANTICS 329
8.1 Interpretations 329
8.2 Quantificational Truth, Falsehood, and Indeterminacy 351
8.3 Quantificational Equivalence and Consistency 358
8.4 Quantificational Entailment and Validity 363
8.5 Truth-Functional Expansions 369
8.6 Semantics for Predicate Logic with Identity and Functors 381

CHAPTER 9 PREDICATE LOGIC: TRUTH-TREES 402
9.1 Truth-Tree Rules for *PL* 402
9.2 Truth-Trees and Quantificational Consistency 410
9.3 Truth-Trees and Other Semantic Properties 416
9.4 Fine-Tuning the Tree Method for *PL* 425
9.5 Truth-Trees for *PLE* 441
9.6 Fine-Tuning the Tree Method for *PLE* 457

CHAPTER 10 PREDICATE LOGIC: DERIVATIONS 474
10.1 The Derivation System *PD* 474
10.2 Using Derivations to Establish Syntactic Properties of *PD* 492
10.3 The Derivation System *PD*+ 521
10.4 The Derivation System *PDE* 526

CHAPTER 11 PREDICATE LOGIC: METATHEORY 545
11.1 Semantic Preliminaries for *PL* 545
11.2 Semantic Preliminaries for *PLE* 558
11.3 The Soundness of *PD*, *PD*+, and *PDE* 561
11.4 The Completeness of *PD*, *PD*+, and *PDE* 566
11.5 The Soundness of the Tree Method 584
11.6 The Completeness of the Tree Method 596

Appendix 1 A-1
Selected Bibliography B-1
Index I-1
Index of Symbols I-7

PREFACE

Our overall goal in the sixth edition of *The Logic Book* remains the same as in earlier versions: presenting deductive symbolic logic in an accessible yet formally rigorous way. To this end, we have extensively reorganized and rewritten several chapters. We have also condensed presentations throughout the book.

Chapter 1 now focuses almost exclusively on deductive logic. Chapter 2 presents and discusses the formal syntax for the language *SL* before turning to symbolizations. Chapter 4 presents all of the truth-tree rules in the first section, and Chapter 5 does the same for the derivation rules of *SD*. The discussion of the completeness proof in Chapter 6 has been rewritten to make the flow of the proof more apparent. Like Chapter 2, Chapter 7 now presents the formal syntax of *PL* before discussing symbolization, and the Aristotelian square of opposition figures less prominently than it did in previous editions. Chapter 8 begins with a presentation of the formal semantics for predicate logic, discussing the formal semantics at greater length and with more examples. (However, those who want to skip most of the formal semantics can do so—we indicate this in the middle of Section 8.1, and we continue to display interpretations in the style of symbolization keys in most of the remainder of the chapter.) All interpretations presented in Chapter 8, except for some exercises for the first section, now use the set of positive integers as the UD. Chapter 9 recovers only extensions of predicates, rather than English readings of those predicates, from completed open branches of truth-trees. Finally, we have added an appendix with some facts about the positive integers; this can serve as a refresher for students as they work through symbolization in Chapter 7 and the construction of interpretations in Chapter 8.

The Logic Book presupposes no previous training in logic, and because it covers sentential logic through the metatheory of first-order predicate logic, it is suitable for both introductory and intermediate courses in symbolic logic.

The instructor who does not want to emphasize metatheory can simply omit Chapters 6 and 11. The chapters on truth-trees and the chapters on derivations are independent, so it is possible to cover truth-trees but not derivations and vice versa. However, the chapters on truth-trees do depend on the chapters presenting semantics; that is, Chapter 4 depends on Chapter 3 and Chapter 9 depends on Chapter 8. In contrast, the derivation chapters can be covered without first covering semantics.

The Logic Book includes large exercise sets for all chapters. Answers to unstarred exercises appear in the *Student Solutions Manual*, available at www.mhhe.com/bergmann6e, while answers to starred exercises appear in the *Instructor's Manual,* which can be obtained by following the instructions on the same web page.

ACKNOWLEDGMENTS

We are grateful to Bernard Kobes and his students at Arizona State University, Mark Gardiner, Johannes Hafner, Robert Robinson and his students at CUNY City College, Trish Savage, Scott Schaerer, and Scott Stapleford for valuable comments on the previous edition and suggestions for the present edition. We are also grateful to the reviewers of this edition, who include

Jamin Asay, *University of North Carolina at Chapel Hill*
John Rawling, *The Florida State University*
Charles Cross, *University of Georgia*
Colin McLarty, *Case Western Reserve University*
Leemon McHenry, *California State University Northridge*
Meggan Payne, *Bellevue College*
Elaine Landry, *UC, Davis*
Arnold Smith, *Kent State University*
Craig Fox, *California University of Pennsylvania*

M. B.
J. M.
J. N.

Chapter 1

INTRODUCTION TO DEDUCTIVE LOGIC

1.1 INTRODUCTION

This is a text in deductive logic—more specifically, in formal or symbolic deductive logic. Chapters 1–5 are devoted to sentential logic, the branch of symbolic deductive logic that takes sentences as the fundamental units of logical analysis. Chapters 7–10 are devoted to predicate logic, the branch of symbolic deductive logic that takes predicates and individual terms as the fundamental units of logical analysis. Chapter 6 is devoted to the metatheory of sentential logic, while Chapter 11 is devoted to the metatheory of predicate logic.

The hallmark of deductive logic is **truth-preservation.** Reasoning that is acceptable by the standards of deductive logic is always truth-preserving; that is, it never takes one from truths to a falsehood. The following syllogism provides an example of such reasoning:

All mammals are vertebrates.

Some sea creatures are mammals.

Some sea creatures are vertebrates.

If the first and second sentences (the **premises**) are true, then the third sentence (the **conclusion**) must also be true. In deductive logic, reasoning that is truth-preserving is said to be '**valid'.**[1]

Over the centuries, a variety of systems of deductive logic have been developed. One of the oldest is Euclid's axiomatization of plane geometry, developed around 300 BCE in classical Greece. All of the truths or theorems of plane geometry can be derived from the five fundamental assumptions or axioms of Euclid's system. Many have attempted to axiomatize other areas of knowledge, including many of the sciences and many areas of mathematics. Giuseppe Peano successfully axiomatized arithmetic in 1889. Aristotle (350 BCE), a near contemporary of Euclid, developed a system of deductive logic that is known as "categorical" or "syllogistic" logic. Our earlier example of valid deductive reasoning was an Aristotelian syllogism. Aristotle's system is built around the logic of terms that identify categories of things, fish, human beings, animals, and so on. Aristotelian logic is still taught today, and the Law School Admissions Test (LSAT) usually contains questions about Aristotelian logic. However, Aristotle's system is limited in some important ways. For example, every syllogism must have exactly two premises, and the premises and conclusion of a syllogism must be structured according to very restrictive rules. Aristotelian logic cannot accommodate such obviously valid reasoning as

> Either the maid or the butler killed Watson.
>
> If it was the maid, Watson was poisoned.
>
> Watson wasn't poisoned.
>
> ---
>
> The butler killed Watson.

because there are three premises, not two, and the first and second premises have more complex forms than can be accommodated in Aristotelian logic.

The systems of deductive logic that we present in this text have their foundations in the work of Gottlob Frege, David Hilbert, Bertrand Russell, and other logicians in the late nineteenth and early twentieth centuries. Unlike axiomatic systems, which are based on a (usually) small number of axioms, the deductive systems in this text are based on a small number of reasonably intuitive rules that govern how sentences can be derived from other sentences.

There are a variety of reasons for studying deductive logic. It is a well-developed discipline that many find interesting in its own right, a discipline that has a rich history and important current research programs and practical applications. Certainly, those who plan to major or do graduate work in areas such as philosophy, mathematics, and computer science should have a solid grounding in skills that are needed for presenting and evaluating arguments in

[1]Deductive logic's requirement that good reasoning be truth-preserving sets a very high standard for acceptable reasoning. This stands in contrast to inductive logic, which sets a more modest standard for good reasoning, namely that if the claims with which one starts are true, then the claims one reaches by using inductive principles are likely to be true. A great deal of the reasoning used in the sciences and in ordinary life is judged by inductive rather than deductive standards.

any discipline. Another reason for studying symbolic logic is that, in learning to symbolize natural language sentences (in our case, English sentences) in a formal language, one becomes more aware and more appreciative of the importance of the structure and complexities of natural languages. The specific words that we use have a direct bearing on whether a piece of reasoning is valid or invalid. For example, it is essential to distinguish between 'Roberta will pass if she completes all the homework' and 'Roberta will pass only if she completes all the homework' if we want to reason well about Roberta's prospects for passing. Finally, the concepts that we explore in this text are abstract concepts. Learning to think about abstract concepts and the relations between them is an important skill that is useful in a wide range of theoretical and applied disciplines.

1.2 CORE CONCEPTS OF DEDUCTIVE LOGIC

Many—but not all—sentences of English are either true or false (this is true of any natural language). We will say that true sentences have the **truth-value T** and that false sentences have the **truth-value F.** Sentences that are true or false include

Canada is located in South America.

Beethoven composed nine symphonies.

The Boston Red Sox will win the next World Series.

On December 29, 1012, it rained in what is now Manhattan.

The first of these sentences is false and therefore has the truth-value **F**. The second sentence is true and has the truth-value **T**. We do now know whether the third and fourth sentences are true or false, but we do know that each is one or the other. Time will tell whether the third sentence has the truth-value **T** or the truth-value **F,** but we will probably never know the truth-value of the fourth sentence. However, regardless of the state of our knowledge, the fourth sentence does have either the truth-value **T** or the truth-value **F**. It is important not to confuse our inability to know which truth-value a sentence has with the sentence's lack of a truth-value. There are obviously many sentences whose truth-values we will never know but that do nevertheless have truth-values.

Examples of sentences that lack truth-values include

Do I really have to do the homework to do well in this course?

Lock the door when you leave.

Hurrah!

The first of these sentences is a question. The second is a request or command and the third is an exclamation. To have a truth-value, a sentence must assert something. These three sentences do not assert or claim anything and hence do not have truth-values. In this text, we will be concerned only with sentences

that do have truth-values; when we refer to a sentence or sentences we are referring to sentences that do have truth-values.

When we are talking about expressions of English we will often use the **variables p, q, r**, and **s** to do so. We use these variables in the same way that mathematicians use **x** and **y** as variables when they are talking about positive integers, that is, the numbers 1, 2, 3, . . . For example, the claim 'If **x** is an even positive integer and **y** is an odd positive integer then **x** plus **y** is an odd positive integer' is a true claim of arithmetic. So too, where **p** and **q** are variables that take expressions of English as their values, the following is true:

> If **p** is a sentence of English and **q** is a sentence of English then
>
> Either **p** or **q**
>
> is also a sentence of English.

The use of variables provides a convenient way for us to make claims about all expressions of English of a certain sort.

We define an **argument** as follows:

> An *argument* is a set of two or more sentences, one of which is designated as the conclusion and the others as the premises.[2]

Note that this definition uses the concept of a set of sentences. Sets are abstract objects that have members (zero or more). We can specify a finite set by listing its members, separated by commas, within a set of curly brackets. Here, for example, is a set of three English sentences:

> {Helen is not very well educated if she believes there is intelligent life on Mars, Helen is very well educated, Helen does not believe there is intelligent life on Mars}

If we designate the first two members that we have listed as the premises and the third sentence as the conclusion, then we have the argument

> Helen is not very well educated if she believes there is intelligent life on Mars.
>
> Helen is very well educated.
>
> ---
>
> Helen does not believe there is intelligent life on Mars.

We adopt the convention of displaying arguments by listing the premises with a horizontal line under the last premise, followed by the conclusion. We will say that arguments displayed in this way are in **standard form.**

[2]This definition allows arguments to have *any* number of premises, including an infinite number. However, all the arguments that we use as examples in this text have only a finite number of premises.

We now have all the terminology we need to introduce the core concepts of deductive logic. The first concept is **logical validity,** a concept that applies to arguments:

> *Logically valid argument:* An argument is *logically valid* if and only if it is not possible for all the premises to be true and the conclusion false. An argument is *logically invalid* if and only if it is not logically valid.

A logically valid argument is truth-preserving. If the premises are true, then the conclusion must also be true. The previous argument about Helen is logically valid because it is impossible for the premises to be true and the conclusion false. That is, if the premises are all true, then the conclusion must be true as well. Note that to determine validity, we do not need to know whether the premises or conclusion are in fact true. All that we need to know is the logical relation between the premises and the conclusion.

The following argument is not logically valid:

> If Sara receives an A in her chemistry class, she will graduate with a 3.5 grade point average.
>
> If Sara graduates with a 3.5 grade point average, she will get into medical school.
>
> Sara will get into medical school
>
> ---
>
> Sara will receive an A in her chemistry class.

This argument is invalid because it is possible for all three premises to be true and the conclusion false. Perhaps Sara will only receive a B in her chemistry class but will nevertheless graduate with a 3.5 grade point average because her other grades are so high, or perhaps she'll get into medical school with less than a 3.5 average because her medical school admissions interview was exceptional.

An argument that is logically valid and that has true premises is said to be **logically sound:**

> *Logically sound argument:* An argument is *logically sound* if and only if it is logically valid and all of its premises are true. An argument is *logically unsound* if and only if it is not logically sound.

Obviously, all logically sound arguments are logically valid, but not all logically valid arguments are logically sound because not all logically valid arguments have premises that are all true. The following argument is logically valid but is not logically sound:

> Italy is a country that is located in North America.
>
> Every country that is located in North America uses the United States dollar as its currency.
>
> ---
>
> Italy uses the United States dollar as its currency.

This argument is logically valid because if the premises were both true, the conclusion would have to be true as well. Obviously, however, the premises are not both true; in fact, they are both false (as is the conclusion), and so the argument is not logically sound, On the other hand, the following logically valid argument is also logically sound, because both of its premises are true:

> The United States is a country that is located in North America.
>
> No country that is located in North America uses the euro as its currency.
>
> ---
>
> The United States does not use the euro as its currency.

Note that if an argument is logically sound, its conclusion will also be true. This is because if the premises of a logically valid argument are true, then, because it is impossible for the argument's premises to be true and its conclusion false, the conclusion must also be true.

Identifying passages of English that contain arguments, extracting those arguments, and presenting them in standard form are important skills that must be mastered before the techniques presented in this text can be used to evaluate English arguments. The following expressions often signal that the sentence that follows is the conclusion of an argument:

> therefore
> thus
> it follows that
> so
> hence
> consequently
> as a result

We will call these 'conclusion indicator expressions'. Similarly, expressions such as

> since
> for
> because
> on account of
> inasmuch as
> for the reason that

often indicate that the sentences following these expressions are the premises of an argument, and we will call these 'premise indicator expressions'.

Sometimes premise and conclusion indicators signal that what is grammatically a single sentence can reasonably be recast as an argument. For example, in the sentence

> If Ron went to the store, he'd be home by now, but he isn't home, and so we may conclude that he didn't go to the store.

the words 'so we may conclude' indicate that the sentence should be understood as an argument. We can present the argument in standard form as follows:

> If Ron went to the store, Ron would be home by now.
>
> Ron isn't home yet.
>
> ---
>
> Ron didn't go to the store.

This argument is, by the way, logically valid.

Note that we cannot assume that the premises of an argument always occur before the conclusion in natural language discourse. The following sentence can also be recast as an argument, and the argument's conclusion occurs at the beginning, rather than the end, of the sentence:

> Michael will not get the job, for the person that gets the job must have strong references, and Michael's references are not strong.

The extracted argument, which is logically valid, is

> The person that gets the job must have strong references.
>
> Michael's references are not strong.
>
> ---
>
> Michael will not get the job.

The conclusion of an argument can also occur between its premises, and an entire argument may be buried in a larger piece of prose or discourse. Here is an example:

> I've got more relatives than I know what to do with. I've got relatives in Idaho and in New Jersey, in Ireland, and in Israel. Among them are a couple of cousins, Tom and Fred Culverson. Both Tom and Fred are hardworking, and Tom is as tenacious as a bulldog. So Tom is sure to be a success, for if there is one thing I have learned in life, it is that everyone who is both hardworking and tenacious succeeds. But I'm sure success won't change Tom. He'll work just as hard after he makes his first million as he does now. He is, after all, a Culverson. And no one is as predictable as a Culverson, unless it's a Hutchings. There are lots of Hutchings on my mother's side, but I haven't had much to do with them . . .

The following explicit argument can be extracted from this passage:

> Tom and Fred are hardworking.
>
> Tom is tenacious.
>
> Everyone who is both hardworking and tenacious succeeds.
>
> ---
>
> Tom will succeed.

As frequently happens, there is a lot of information in the passage that is not relevant to the specific argument we have extracted.

We note that our formal definition of 'argument' allows for arguments in which the premise or premises provide no support whatsoever for the conclusion. Here is an example:

> Two is the smallest prime number.
>
> ---
>
> In 1967 the Green Bay Packers won the first Super Bowl.

The one premise of this argument is true, as is the conclusion. But it is certainly possible for the premise to be true and the conclusion false, so the argument is invalid. (The Kansas City Chiefs lost to the Green Bay Packers in the first super bowl game, but they might have won.) Some might object that we should not count this as an argument at all, for the premises of an argument usually have something to do with the conclusion. But it is not at all clear what kind of connections there must be between sentences before we are willing to count one of them as the conclusion and the rest as the premises of an argument. Some benighted and superstitious soul might think that the fact that a bluebird landed on his window sill in the morning is relevant to his winning the lottery and 'argue' as follows:

> This morning a bluebird landed on my bedroom window as I was getting out of bed.
> Therefore, this will be my lucky day and I will win today's lottery.

Although where and when bluebirds land has nothing to do with who wins lotteries and when this happens, it is prudent to count this as an argument, because we can then use the tools of deductive logic to show that it is a bad–and clearly invalid—argument.

Every sentence that has a truth-value is either logically true, logically false, or logically indeterminate:

> A sentence is *logically true* if and only if it is not possible for the sentence to be false.
>
> A sentence is *logically false* if and only if it is not possible for the sentence to be true.

A sentence is *logically indeterminate* if and only if it is neither logically true nor logically false.

The sentence 'Either June will pass Chemistry 101 or June will not pass Chemistry 101' is logically true, for it is impossible for this sentence to be false. The sentence is true by virtue of its structure alone. That is, all sentences of the form **p** or not **p**, where **p** is any sentence that has a truth-value and **not p** is a denial of **p**, is logically true. So, for example, 'Either it is raining or it is not raining' is logically true. Of course, there are other forms of logically true sentences. For example, 'If all dogs have tails, then there are no dogs that do not have tails' is logically true. Logically true sentences are uninformative; they don't tell us anything about the 'real world'. The sentence about June tells us nothing about her chances of passing Chemistry 101, and the sentence about dogs tells us nothing about whether dogs do or do not have tails. An example of a logically false sentence is

June will pass Chemistry 101 and she will not pass Chemistry 101.

Logically false sentences are false by virtue of their structure. This sentence is logically false because it is impossible for June to both pass and not pass Chemistry 101. Like logically true sentences, logically false sentences give us no information about the world. The sentence 'It rained on July 6, 1309, in what is now San Francisco' is logically indeterminate. Whether this sentence is true or false is not a matter of logic but rather depends on what in fact happened on that day in that place. Most of the sentences that we encounter in ordinary conversation, writing, and reading are logically indeterminate. Whether they are true or false depends upon how the world in fact is.

Logical equivalence is a relation between sentences:

Logically equivalent sentences: Sentences **p** and **q** are *logically equivalent* if and only if it is not possible for one of these sentences to be true while the other sentence is false.

The sentences 'Jake loves Henry' and 'Henry is loved by Jake' are logically equivalent. One cannot be true without the other being true, and if either is false the other is false. Given a pair of logically equivalent sentences, both of which are logically indeterminate, we know that either both members of the pair are true or both are false, but we don't always know which. We know the sentences 'Mary is taller than Henry' and 'Henry is shorter than Mary' are logically equivalent. Mary cannot be taller than Henry without Henry being shorter than she is, and Henry cannot be shorter than Mary without Mary being taller than he is. But the fact that the sentences are logically equivalent doesn't tell us whether they are both true or both false. Since we may replace the variables **p** and **q** with the same sentence, it follows from our definition of logical equivalence that every sentence is equivalent to itself.

Logical consistency is defined as follows:

> A set of sentences is *logically consistent* if and only if it is possible for all the members of that set to be true. A set of sentences is *logically inconsistent* if and only if it is not logically consistent.

The set

{Tom is left-handed, Carol is left-handed, Mona is left-handed}

is logically consistent. Whoever Tom, Carol, and Mona are, it is logically possible that they are all left-handed. But

{Everyone in the room is left-handed, Mona is in the room, Mona is not left-handed}

is logically inconsistent. The three sentences in the set cannot all be true. If everyone in the room is left-handed and Mona is in the room, then Mona must be left-handed, so it is false that she is not left-handed.

It may seem that the notion of what is and is not logically possible, a concept we have used in defining all of the core concepts of deductive logic, is itself in need of clarification. One of the motivations for developing formal or symbolic systems of deductive logic is in fact to refine the concept of logical possibility so that there are clear criteria for what is logically possible and what is not. These criteria form the bases of our formal deductive systems in the rest of this book. In Chapter 2 we will present the symbolic language *SL* (for 'Sentential Logic'), and in Chapter 3 we will define formal sentential logic versions of the core concepts of deductive logic. In Chapter 7 we will present the far more powerful language *PL* (for 'Predicate Logic'), and in Chapter 8 we will define formal predicate logic versions of the core concepts.

The last core concept of deductive logic is that of **entailment.** This concept is closely related to, but not identical, with that of validity:

A set of sentences *logically entails* a sentence if and only if it is impossible for the members of the set to be true and that sentence false.

The set

{Henry and Joan will both receive their law degrees in June}

logically entails the sentence 'Joan will receive her law degree in June', for it is impossible for the sentence 'Joan will receive her law degree in June' to be false if the single sentence in the set is true. On the other hand, the set

{Andrew plays soccer, Siri plays tennis}

does not logically entail the sentence 'Andrew does not play tennis and Siri does not play soccer', for it may be the case that in addition to playing soccer, Andrew plays tennis and in addition to playing tennis, Siri plays soccer.

If an argument is logically valid, then the set consisting of the premises of the argument logically entails the argument's conclusion. However, logical entailment is a more general concept than is logical validity, for there are sentences that are entailed by the empty set, while arguments must have at least one premise. The former may seem odd, but there are good reasons for introducing the more general concept—one reason being that the concept of logical entailment facilitates reasoning about logical systems, as we will do in Chapters 6 and 11.

What sentences are entailed by the empty set? One example is the sentence 'Either June will pass Chemistry 101 or she will not pass Chemistry 101'. This sentence is logically entailed by the empty set of sentences because it is impossible for all of the members of the empty set (there are none) to be true and 'Either June will pass Chemistry 101 or she will not pass Chemistry 101' to be false, precisely because this sentence cannot be false. One way that we might intuitively understand this sentence's and all logical truths being entailed by the empty set is by noting that this sentence requires no support. It is true regardless of what the world is like. This means that reasoning that starts with no assumptions (the empty set) and reaches this sentence is truth-preserving.

1.2**E** EXERCISES

Note: Solutions to unstarred exercises can be accessed from the following web page: http://highered.mcgraw-hill.com/sites/007353563x/information_center_view0/

Select 'Student Edition' on the left-hand side of the page, then select 'Chapter 1' and the solutions will appear as a pdf file. If you do not have *Adobe Reader* you can download it for free from Adobe's website.

1. For each of the following, indicate whether it is the kind of sentence that falls within the scope of this text—that is, whether it is a sentence that has a truth-value. If it is not this kind of sentence, explain why not.

a. George Washington was the second president of the United States.
*b. The next president of the United States will be a Republican.
c. Turn in your homework on time or not at all.
*d. Would that 9/11 had never happened.
e. Two is the smallest prime number.
*f. One is the smallest prime number.
g. George Bush, Senior, immediately preceded George W. Bush as president.
*h. At 3:00 pm EST on January 15, 1134, there was a snowstorm in what is now Manhattan.

2. For each of the following passages, determine whether it advances an argument. If an argument is probably being expressed, restate the argument in

standard form. If the intent is probably not to express an argument, explain why not.

a. When Mike, Sharon, Sandy, and Vicky are all away from the Tacoma office, no important decisions get made. Mike is off skiing, Sharon is in Spokane, Vicky is in Olympia, and Sandy is in Seattle. So no decisions will be made today.

*b. Our press releases are always crisp and upbeat. Mike is the press officer, and if he has his way, all press releases are crisp and upbeat. Mike always has his way.

c. Shelby and Noreen are wonderful at dealing with irate students and faculty members. Stephanie is very good at managing the chancellor's very demanding schedule, and Tina keeps everything moving and cheers everyone up.

*d. Tom and Ray are our office assistants, and one of them is incompetent and the other one is lazy. So either Tom is incompetent and Ray is lazy, or Tom is lazy and Ray is incompetent.

e. We won't be able to repair the deck because to do so we need stainless steel screws. All the screws we have are in the nail and screw cabinet. The first drawer contains only galvanized nails, the second contains only ordinary nails, and the third contains only drywall screws. And the fourth and bottom drawers contain only brass screws.

*f. If the budget is to be balanced we will have to raise taxes or cut spending. If we cut spending, then entitlement programs, including Medicare and Social Security, will have to be significantly cut and the Democrats will have a fit. If we raise taxes, Republicans will go ballistic. So if the budget is balanced, either the Democrats will have a fit or the Republicans will go ballistic.

g. The weather is perfect, the view is wonderful, and we're on vacation. So why are you unhappy?

*h. The new kitchen will be finished on time only if our carpenter works over the weekend. He will work over the weekend only if he doesn't go duck hunting, but he will go duck hunting. So the new kitchen will not be finished on time.

i. If Sarah did the wiring, it was done right, and if Marcie did the plumbing, it was done right. As neither the wiring nor the plumbing was done right, Sarah didn't do the wiring and Marcie didn't do the plumbing.

*j. Sarah, John, Rita, and Bob have all worked hard, and one of them will be promoted. Their company is having a cash flow problem and is offering its employees who are over 55 a $50,000 bonus if they retire at the end of this year. Sarah, John, and Bob are all over 55 and will take early retirement. So Rita will be promoted.

k. Having cancer is a good, for whatever is required by something that is a good is itself a good. Being cured of cancer is a good, and being cured of cancer requires having cancer.

*l. Humpty Dumpty sat on a wall. Humpty Dumpty had a great fall. All the king's horses and all the king's men couldn't put Humpty together again. So they made him into an omelet and had a great lunch.[3]

1.3 SPECIAL CASES OF LOGICAL CONCEPTS

In this section we point out and explain some special cases of our logical concepts. These may seem counterintuitive, but we shall explain why they follow from our definitions.

[3]With apologies to Lewis Carroll.

The first special case is that of an argument whose conclusion is logically true. Such an argument is logically valid no matter what premises it has. Here is such an argument:

> The Philadelphia Phillies is the best team in the National League.
>
> ---
>
> Either the next president of the United States will be a woman or the next president will not be a woman.

The conclusion of this argument is logically true. No matter who wins the next presidential election, that person either will or will not be a woman. Because the conclusion is logically true, it is impossible for the argument's premise to be true and the conclusion false, because it is impossible for the conclusion to be false. So, the argument is logically valid; it will never lead us from truths to a falsehood. The second, and related, special case is that of every logically true sentence being entailed by every set of sentences, including the empty set, because it is impossible for a logically true sentence to be false and hence impossible for the members of a set, any set, all to be true and that logically true sentence false.

The third special case concerns arguments whose premises form logically inconsistent sets. Here is an example:

> Albert is brighter than all his sisters.
>
> Sally is Albert's sister.
>
> Sally is brighter than all her brothers.
>
> ---
>
> *Tyrannosaurus rex* was the fiercest of all dinosaurs.

The premises form a logically inconsistent set because they cannot all be true. If the first and second premises are both true, for example, the third premise cannot be true. And if the premises cannot all be true, it is impossible for the premises to be true and the conclusion false. The argument is therefore logically valid. Although every argument whose premises form an inconsistent set is logically valid, of course no such argument can be logically sound. Note that every argument that has at least one logically false premise is an instance of this special case, because any set of sentences that contains a logically false sentence is logically inconsistent.

Our fourth and fifth special cases concern logical equivalence, which we have defined as follows:

> Sentences **p** and **q** are *logically equivalent* if and only if it is not possible for one of the sentences to be true while the other sentence is false.

By this definition, all logically true sentences **p** and **q** are logically equivalent. Because it is not possible for any logically true sentence to be false, it is impossible for logically true sentences **p** and **q** to be such that one is true while the other is false. It follows that all logically true sentences are logically equivalent. Similar reasoning shows that all logically false sentences are logically equivalent. Of course, not all logically indeterminate sentences are logically equivalent.

1.3**E** EXERCISES

1. Which of the following are true, and which are false? Explain your answers, giving examples where appropriate.

a. Every argument that has 'Whatever will be, will be' as a conclusion is logically valid.

*b. Any argument that includes among its premises 'Everyone is a scoundrel' and 'I'm no scoundrel' is logically valid.

c. Every argument that has 'Everyone is a scoundrel and I'm no scoundrel' as a conclusion is logically invalid.

*d. The premises of a logically valid argument always provide support for the conclusion of the argument.

2. Answer each of the following.

a. Suppose that an argument has a premise that is logically true. Must the argument be logically valid? Explain.

*b. Suppose that an argument has a premise that is logically equivalent to a logically false sentence. Must the argument be logically valid? Explain.

c. Suppose that an argument has a logically true sentence as its conclusion. Explain why the argument must be valid no matter what its premises are. Explain why some such arguments are sound and some are not.

*d. Suppose that the premises of an argument form a logically inconsistent set. Explain why the argument must be logically valid but unsound.

GLOSSARY

ARGUMENT: An argument is a set of two or more sentences, one of which is designated as the conclusion and the others as the premises.

LOGICAL VALIDITY: An argument is *logically valid* if and only if it is not possible for the premises to be true and the conclusion false. An argument is *logically invalid* if and only if it is not logically valid.

LOGICAL SOUNDNESS: An argument is *logically sound* if and only if it is logically valid and all its premises are true. An argument is *logically unsound* if and only if it is not logically sound.

LOGICAL TRUTH: A sentence is *logically true* if and only if it is not possible for the sentence to be false.

LOGICAL FALSITY: A sentence is *logically false* if and only if it is not possible for the sentence to be true.

LOGICAL INDETERMINACY: A sentence is *logically indeterminate* if and only if it is neither logically true nor logically false.

LOGICAL EQUIVALENCE: Sentences **p** and **q** are *logically equivalent* if and only if it is not possible for one of these sentences to be true while the other sentence is false.

LOGICAL CONSISTENCY: A set of sentences is *logically consistent* if and only if it is possible for all the members of that set to be true. A set of sentences is *logically inconsistent* if and only if it is not logically consistent.

LOGICAL ENTAILMENT: A set of sentences *logically entails* a sentence if and only if it is impossible for all the members of the set to be true and that sentence false.

Chapter 2

SYNTAX AND SYMBOLIZATION

Section 2.1 of this chapter presents the formal language *SL* ('*SL*' is short for 'Sentential Logic'). Section 2.2 introduces the symbolization process, that is, how English sentences are symbolized in *SL*. Section 2.3 is devoted to developing proficiency in the symbolization process. Section 2.4 explores some of the complexities involved in symbolizing English sentences in *SL*.

2.1 THE SYNTAX OF *SL*

The **syntax** of a language specifies the basic expressions of a language and the rules that determine which combinations of those expressions count as sentences of the language. The syntax of a language does not specify how the sentences of the language are to be interpreted; that is a matter for **semantics**, which we will address in Chapter 3. The syntax of English, and every other natural language, is enormously complex. Fortunately, the syntax of *SL* is simple, straightforward, and easily learned.

But before we lay out the syntax of *SL* we need to introduce some terminology.

METALANGUAGE/OBJECT LANGUAGE

Throughout the rest of this text we will be using English to talk about two formal languages, first *SL* and later *PL*. When we use a language to talk about

a language, we are using that language as a **metalanguage,** and the language that we are talking about is the **object language.** So when we are talking about *SL* and *PL* they are the object languages.[1]

USE AND MENTION

We regularly use language to talk about or of a host of different things,

> . . . of shoes—and ships—and sealing-wax—Of cabbages—and kings—

as Lewis Carroll wrote. But we also have occasion to use language to talk about language. In this text we will frequently talk of expressions, sentences, and arguments of *SL* (and later of *PL*), as well as words, sentences, and arguments of English. When we talk about these or other linguistic constructions, large or small, we are **mentioning** rather than **using** those constructions and it is important that we have ways of indicating that we are doing so. Failure to indicate that we are mentioning rather than using a piece of language can lead to confusion. Consider the sentence:

> Minnesota derives from a Native American word.

We can probably all figure out what a person who asserts this sentence means, namely, that the name of the state located between the Dakotas and Wisconsin derives from a Native American word. But what the sentence literally says is that Minnesota, the state, a political entity, derives from a Native American word, and this is clearly false.

In this text, we use two conventions to indicate that we are mentioning or talking about language. The first is to place the linguistic expression we are mentioning within single quotation marks. So we can make the intended claim about the origin of the name of Minnesota by saying that 'Minnesota' derives from a Native American word. The second convention we will use is that of displaying the language we wish to talk about or mention on an indented line or lines. Thus we can truly say that the following sentence is about the origin of the name of Minnesota:

> 'Minnesota' derives from a Native American word.

We have just said something about a sentence, and we indicated that we were doing so by displaying that sentence on a line by itself. Within the displayed sentence, we used the convention of placing an expression that we are talking about within single quotes. We have used both of these conventions earlier in this text, and we will use them throughout the rest of this text.

---

[1]In a German class the instructor uses English to talk about German, and in this instance English is the metalanguage and German is the object language. And when a grammar instructor uses English to talk about the rules of English grammar English is both the metalanguage and the object language.

METAVARIABLES

Most of us are familiar with mathematicians' use of the letters 'x' and 'y' to make arithmetic claims such as

> For any positive integers x and y, if x is even and y is odd then x + y is odd.

Of course 'x' and 'y' are not integers. They are letters of the English alphabet. But they are used by mathematicians to make general claims about, in this case, integers. The letters 'x' and 'y' when so used are said to be **variables** and to **range over** or **take as values** the positive integers, that is, the numbers 1, 2, 3, 4 . . .

Analogously to the way mathematicians use 'x' and 'y' as variables ranging over numbers we will use the boldface capital letters '**P**', '**Q**', '**R**', and '**S**', with or without subscripts, as in

$$\mathbf{P}\ \mathbf{P}_1\quad \mathbf{Q}_3$$

as **metavariables** ranging over expressions of the object languages *SL* and *PL*. These variables are termed 'metavariables' because they are parts of the metalanguage we are using, English, not parts of the object languages *SL* and *PL*. We will similarly use the boldfaced lowercase letters '**p**', '**q**', '**r**', and '**s**', with or without following primes, as metavariables ranging over expressions of English.

We can now lay out the syntax of *SL*. We begin by specifying the expressions or **vocabulary** of *SL*. These are

> **Sentence Letters**: the capital Roman letters 'A' through 'Z', with or without positive integer subscripts[2]:
>
> A, B, C, . . . , A_1, B_1, C_1, . . . , A_2, B_2, C_2, . . .
>
> **Sentential Connectives**:
>
> ~ (called the 'tilde')
>
> & (called the 'ampersand')
>
> $\vee$ (called the 'wedge')
>
> $\supset$ (called the 'horseshoe')
>
> $\equiv$ (called the 'triple bar')

[2]The inclusion of capital Roman letters with positive integer subscripts among the sentence letters of *SL* means that there are infinitely many sentence letters of *SL*. This is appropriate as there are infinitely many claims that can be made about the universe and its contents and we never know how many of these claims someone may want to symbolize by using sentence letters of *SL*.

The connective tilde is a **unary connective**; the remaining connectives are **binary connectives.** Binary connectives connect, as the name suggests, two sentences to form a new sentence. The unary connective tilde attaches to a single sentence to form a new sentence.[3]

Punctuation marks: '(' and ')'

Recursive Definition of 'Sentence of *SL*'

We define '**sentence of *SL***' as follows:

1. Every sentence letter of *SL* is a sentence of *SL*.
2. If **P** is a sentence of *SL*, then ~ **P** is a sentence of *SL*.
3. If **P** and **Q** are sentences of *SL*, then (**P** & **Q**) is a sentence of *SL*.
4. If **P** and **Q** are sentences of *SL*, then (**P** ∨ **Q**) is a sentence of *SL*.
5. If **P** and **Q** are sentences of *SL*, then (**P** ⊃ **Q**) is a sentence of *SL*.
6. If **P** and **Q** are sentences of *SL*, then (**P** ≡ **Q**) is a sentence of *SL*.
7. Nothing is a sentence of *SL* unless it can be formed by repeated application of clauses 1–6.[4]

Our specification of the syntax of *SL* is now complete. Our recursive definition of 'sentence of *SL*' provides a complete specification of what expressions counts as a sentence of *SL*.

All of the following expressions are sentences of *SL*, as we shall explain:

(B & D)
((B ≡ D) ∨ ~ C)
~ ~ D
((A & B) & ~ (C ≡ ~ D))

'(B & D)' contains two sentence letters, 'B' and 'D'. By clause 1, they are both sentences of *SL*. Since they are sentences of *SL*, '(B & D)' is also, by clause

[3]Expressions that attach to a single sentence to form a new sentence, as does the tilde, are traditionally, though somewhat misleadingly, termed 'connectives' though they do not connect two sentences.

[4]Readers are unlikely to have difficulty understanding clauses 2–6 of our recursive definition of 'sentence of *SL*. For example, the import of clause 3

If **P** and **Q** are sentences of *SL*, then (**P** & **Q**) is a sentence of *SL*,

is simply that the result of placing '&' between any two sentences of *SL* and enclosing the result in parentheses is a sentence of *SL*. But there is a complexity here that we note for the sake of completeness. We stipulate that in expressions that contain both metavariables and expressions of *SL* the metavariables are being used and the expressions of *SL* are being mentioned. We need this stipulation because in claims such as clauses 2–6 of our recursive definition we are *talking about* what expressions constitute sentences of *SL*, not using sentences of *SL*. We adopt a parallel convention for hybrid expressions in which we use the metalinguistic variables '**p**', '**q**', '**r**', and/or '**s**' to refer to expressions of English.

3, a sentence of *SL*. The second listed sentence contains the sentence letters 'B', 'C', and 'D'. These are all sentences of *SL* by clause 1. Therefore '(B ≡ D)' is a sentence of *SL*, by clause 6, and '~ C' is a sentence of *SL*, by clause 2. Since '(B & D)' and '~ C' are both sentences of *SL*, '((B ≡ D) ∨ ~ C)' is also a sentence of *SL*, by clause 4.

Turning to the third sentence, 'D' is a sentence of *SL*, by clause 1. Therefore, by clause 2, '~ D' is also a sentence of *SL*, and since '~ D' is a sentence of *SL*, so is '~ ~ D', again by clause 2. The fourth listed sentence contains four sentence letters, 'A', 'B', 'C', and 'D' and these are all sentences of *SL* by clause 1. Since 'A' and 'B' are sentences of *SL* by clause 1 so is '(A & B)', by clause 3. And since 'D' is a sentence of *SL*, so is '~ D', by clause 2. Because 'C' and '~ D' are both sentences of *SL*, '(C ≡ ~ D)' is a sentence of *SL*, by clause 6. It follows that '~ (C ≡ ~ D)' is a sentence of *SL* by clause 2. Finally, it follows by clause 3 that '((A & B) & ~ (C ≡ D)' is a sentence of *SL*.

The following expressions are *not* sentences of *SL*:

B & D
∨ A
(BC ⊃ D)
(B ⊂ (C ∨ D))
(**P** ≡ **Q**)

- 'B & D' is not a sentence of *SL* because the only clause of our definition that introduces an ampersand is clause 3 and it requires that when sentences are joined by an ampersand the result be placed within parentheses. 'B & D' contains no parentheses, so it is not a sentence of *SL*. (However, we shall adopt an *informal* convention of allowing the deletion of outer parentheses, so that 'B & D' will count informally as a sentence of *SL*. All parentheses other than outer parentheses are necessary to make it clear what sentences binary connectives are connecting.)
- '∨ A' contains a wedge, and the only clause that introduces a wedge is clause 4, which requires a sentence in front of the wedge as well as a sentence after the wedge. So '∨ A' is not a sentence of *SL*.
- '(BC ⊃ D)' contains a horseshoe, and clause 5 is the only clause that introduces a horseshoe. But for clause 5 to be applicable, 'BC' would have to be a sentence of *SL*. It is not, because there is no clause in our definition that allows two sentences of *SL* to be concatenated without placing a connective between them. So '(BC ⊃ D)' is not a sentence of *SL*.

- '(B ⊂ (C ∨ D))' is not a sentence of *SL* because '⊂' is not a symbol of *SL*.
- '(**P** ≡ **Q**)' is not a sentence because neither '**P**' nor '**Q**' is a sentence of *SL*. The sentence letters of *SL* do not include boldface letters.

We complete this section by laying out terminology associated with the syntax of *SL*. First, our definition of 'sentence of *SL*' is a **recursive definition**. Recursive definitions start by directly identifying some items that the concept being defined applies to. In our case clause 1 of our definition specifies that the concept 'sentence of *SL*' applies to the sentence letters of *SL*. Subsequent clauses specify that if one or more items are such that the concept in question applies to them, then that concept applies to some additional item. In our definition, clauses 2 through 6 do this. These clauses say that if some expression or expressions are sentences of *SL* then so is another expression. Our recursive definition of 'sentence of *SL*' ends with a closure clause, which says that there is nothing else the concept being defined applies to.

Sentences of *SL* are of two sorts: **atomic sentences** and **compound sentences**. The sentence letters of *SL* constitute the atomic sentences of *SL*. (They are called 'atomic sentences' because they are not formed or compounded from other sentences.) All non-atomic sentences are **compound** sentences, so called because they are formed or compounded from other sentences of *SL*. All compound sentences contain at least one sentential connective. There are five types of compound sentences and each type has a **main connective** and an **immediate component** or **components**:

> **Negations:** sentences of the form ~ **P**. The *main connective* of ~ **P** is '~' and **P** is the immediate component.
>
> **Conjunctions:** sentences of the form (**P** & **Q**). The *main connective* of (**P** & **Q**) is '&' and **P** and **Q** are the immediate components.
>
> **Disjunctions:** sentences of the form (**P** ∨ **Q**). The *main connective* of (**P** ∨ **Q**) is '∨' and **P** and **Q** are the immediate components.
>
> **Material Conditionals:** sentences of the form (**P** ⊃ **Q**). The *main connective* of (**P** ⊃ **Q**) is '⊃' and **P** and **Q** are the immediate components.
>
> **Material Biconditionals:** sentences of the form (**P** ≡ **Q**). The *main connective* of (**P** ≡ **Q**) is '≡' and **P** and **Q** are the immediate components.

The immediate components of a conjunction are the **conjuncts** of that conjunction and the immediate components of a disjunction are the **disjuncts** of that disjunction. The immediate components of a material conditional are the **antecedent**—which precedes the main connective—and the **consequent**—which follows the main connective.

Below we list several examples of each kind of truth-functional compound sentence of *SL*. The arrows point to the main connectives of these sentences.

All of the following are *negations* of *SL*:

$$\downarrow$$
$$\sim A$$
$$\downarrow$$
$$\sim (B \,\&\, C)$$
$$\downarrow$$
$$\sim (A \vee (D \,\&\, B))$$
$$\downarrow$$
$$\sim \sim (A \equiv D)$$

All of the following are *conjunctions* of *SL*:

$$\downarrow$$
$$(A \,\&\, B)$$
$$\downarrow$$
$$(A \,\&\, (B \vee C))$$
$$\downarrow$$
$$((C \,\&\, \sim D) \,\&\, (\sim D \vee B))$$
$$\downarrow$$
$$(D \,\&\, ((C \equiv A) \vee \sim (A \supset B)))$$

All of the following are *disjunctions* of *SL*:

$$\downarrow$$
$$(D \vee \sim A)$$
$$\downarrow$$
$$(B \vee (A \supset \sim D))$$
$$\downarrow$$
$$((B \equiv \sim D) \vee (A \,\&\, \sim C))$$
$$\downarrow$$
$$(\sim \sim (A \,\&\, \sim D) \vee (B \supset (A \equiv \sim D)))$$

All of the following are *material conditionals* of *SL*:

$$\downarrow$$
$$(A \supset B)$$
$$\downarrow$$
$$((B \,\&\, \sim C) \supset \sim D)$$
$$\downarrow$$
$$(\sim D \supset (B \,\&\, (C \vee \sim A))$$
$$\downarrow$$
$$((A \equiv \sim B) \supset (B \supset \sim A))$$

All of the following are *material biconditionals* of *SL*:

$$\begin{array}{l} \quad\downarrow \\ (\sim D \equiv \sim A) \\ \quad\downarrow \\ (B \equiv (\sim A \mathbin{\&} \sim C)) \\ \qquad\downarrow \\ ((\sim A \mathbin{\&} \sim B) \equiv (C \vee \sim D)) \\ \qquad\downarrow \\ ((C \equiv \sim D) \equiv \sim A) \end{array}$$

We next introduce the notion of a **component** of a sentence of *SL*.

> The *components* of a sentence **P** of *SL* are
>
> - **P** itself,
> - The immediate components (if any) of **P**,
> - The components of **P**'s immediate components.

What this comes to is that every sentence that can be found within **P**, as well as **P** itself, counts as a component of **P**. This is how it works. Consider the compound sentence

$$\sim (A \supset (B \mathbin{\&} \sim D))$$

By definition, this sentence is a component of itself. The immediate component of this negation,

$$(A \supset (B \mathbin{\&} \sim D))$$

is also by definition a component of the negation. The two immediate components of this sentence:

$$A$$
$$(B \mathbin{\&} \sim D)$$

are therefore components of '$(A \supset (B \mathbin{\&} \sim D))$' and therefore of '$\sim (A \supset (B \mathbin{\&} \sim D))$', and so on. By this definition, the components of '$\sim (A \supset (B \mathbin{\&} \sim D))$' are

$$\sim (A \supset (B \mathbin{\&} \sim D))$$
$$(A \supset (B \mathbin{\&} \sim D))$$
$$A$$
$$(B \mathbin{\&} \sim D)$$
$$B$$
$$\sim D$$
$$D$$

Finally, we establish two conventions that make it easier to work with sentences of *SL*. First, we informally allow the deletion of **outermost parentheses** as deleting them does not introduce any ambiguity as to what sentences binary connectives are connecting. A sentence of *SL* that begins with a left parenthesis and ends with a right parenthesis has outermost parentheses, and they can, by our convention, be omitted. So we can write 'A ⊃ (B & ~ D)' rather than '(A ⊃ (B & ~ D))'. Note that this convention about outermost parentheses does not apply to negations of compound sentences. (Negations do not have outermost parentheses; they start with a '~', not a left parentheses.) For example, '~ ((A ∨ B) ⊃ ~ (C ≡ D))' cannot be rewritten as '~ (A ∨ B) ⊃ ~ (C ≡ D)'. The former is a negation and the latter is a material conditional. We also allow the use of square brackets ('[' and ']') in place of parentheses to make complicated sentences easier to read. For example, if we write '~ ((A ∨ B) ⊃ ((A ⊃ ~ C) ≡ D))' as '~ [(A ∨ B) ⊃ [(A ⊃ ~ C) ≡ D]]', it becomes easier to discern the structure of the sentence.

2.1**E** EXERCISES

1. Which of the following are sentences of *SL* and which are not? For those that are not, explain why they are not.

a. & H
*b. B & Z
c. ~ O
*d. M ~ N
e. J ⊃ (K ⊃ (A ∨ N))
*f. **P** ∨ **Q**
g. (I ∨ [T & E])
*h. (U & C & ~ L)
i. [(G ∨ E) ⊃ (~ H & (K ⊃ B)]
*j. (F ≡ K) ⊃ [M ∨ K]

2. For each of the following sentences, indicate whether the sentence is a negation, a conjunction, a disjunction, a material conditional, or a material biconditional.

a. A ⊃ B
*b. ~ A ∨ B
c. ~ A ≡ ~ B
*d. ~ ~ (A ⊃ B)
e. ~ A ⊃ (B & ~ D)
*f. (D ≡ ~ A) ≡ B
g. ~ (A ≡ B) & (~ C ⊃ D)
*h. ~ ~ ~ B
i. [A & ~ (B ∨ C)] ⊃ [(A & ~ B) & (A & ~ C)]
*j. (A ⊃ B) & (B ⊃ A)
k. ~ (~ A ⊃ ~ B)
*l. ~ A ⊃ B
m. ~ ~ (A ⊃ B) ∨ (C ⊃ D)
*n. (A ∨ ~ B) ⊃ ~ (C & ~ D)

3. For each of the following sentences, circle the main connective and underline the immediate sentential component(s). Then list all the sentential components, including the atomic components.

a. ~ A & H
*b. ~ (A & H)
c. ~ (S & G) ∨ B
*d. K ⊃ (~ K ⊃ K)
e. (C ≡ K) ⊃ [~ H ⊃ (M & N)]
*f. M ⊃ [~ N ⊃ ((B & C) ≡ ~ [(L ⊃ J) ∨ X])]

4. Which of the following characters can occur immediately to the left of '~' in a sentence of *SL*? When one can so occur, give a sentence of *SL* in which it does; when it cannot so occur, explain why. Which of these characters can occur immediately to the right of 'A' in a sentence of *SL*? When one can so occur, give a sentence of *SL* in which it does; when it cannot so occur, explain why.

a. H
*b. &
c. (
*d.)
e. [
*f. ~

2.2 INTRODUCTION TO SYMBOLIZATION

As we have seen, the sentence letters or atomic sentences of *SL* can be combined using connectives and parentheses to form compound sentences of considerable complexity. But what is the relation between sentences of *SL* and English sentences? The sentence letters of *SL* can be used to symbolize English sentences. In theory, any sentence letter of *SL* can symbolize any English sentence. Recall the simple argument we used as an example in Chapter 1:

Either the maid or the butler killed Watson.
If it was the maid, Watson was poisoned.
Watson wasn't poisoned.

The butler killed Watson.

For convenience, we will refer to this English language argument as our "whodunit". We could use 'A' to symbolize the first premise, 'B' to symbolize the second, 'C' to symbolize the third, and 'D' to symbolize the conclusion. Our symbolic argument would then be

A
B
C
—
D

Our whodunit is clearly valid. But the premises of our *symbolic* argument provide no apparent support for the conclusion, and that argument will turn out to be an invalid argument of *SL*. The problem is that this symbolic argument does not reflect the structure of the English language argument. That is, there are important relationships among the premises and conclusion that are not reflected in our symbolization of our whodunit. To capture those relationships, we need to use compound sentences of *SL* to symbolize the premises and conclusion of our whodunit. To this end, we need to know how the sentential connectives of *SL* are to be interpreted. The following list pairs connectives of *SL* with expressions of English to which they roughly correspond.

~	It is not the case that . . .
&	. . . and . . .
∨	. . . or . . .
⊃	if . . . then . . .
≡	. . . if and only if . . .

Given this information about the connectives, a far better symbolization of our whodunit is

M ∨ B
M ⊃ W
~ W
———
B

We can specify the English sentences that we are symbolizing with the sentence letters 'M', 'B', and 'W' as follows:

M: The maid killed Watson.
B: The Butler killed Watson.
W: Watson was poisoned.

We call such specifications '**symbolization keys**', and we will use them throughout this chapter. A symbolization key for a group of atomic sentences of *SL* allows us to construct English readings for sentences of *SL* that contain those sentence letters. For example, an appropriate English reading of '~ (M & B)' given our current symbolization key is

It is not the case that both the maid and the butler killed Watson,

or, more colloquially,

The maid and the butler didn't both kill Watson.

Note that the first premise of our whodunit, 'Either the maid or the butler killed Watson' is not literally a compound sentence, that is, it does not consist of two sentences connected by 'or'. But it is clearly equivalent to such a compound sentence, namely

> The maid killed Watson or the butler killed Watson.

Similarly, 'The butler and the maid both hated Watson' is not a compound sentence consisting of two English sentences connected by 'and'; but it is equivalent to the compound sentence 'The butler hated Watson and the maid hated Watson'. If we expand our current symbolization key to include:

> H: The butler hated Watson
> A: The maid hated Watson

we can symbolize this additional information about the maid and the butler as 'H & A'.

When we symbolize English sentences, we usually choose sentence letters of *SL* that may help us to remember which sentences they are symbolizing. For example, we earlier used 'M' to symbolize the sentence 'The maid killed Watson' in the expectation that using 'M' will help us remember that 'M' is symbolizing a sentence about the maid. In symbolizing our whodunit we selected 'B' and 'W' for analogous reasons. There is no formal requirement that sentence letters be correlated in this manner with the sentences that they symbolize, and when we are using one symbolization key to symbolize a significant number of sentences, it often becomes impossible to use this mnemonic device. It is, however, a requirement that each sentence letter symbolize only one sentence in a given symbolization key. So we cannot expand the previous whodunit symbolization key to include 'M' as a symbolization of 'The maid hated Watson' because in that symbolization key 'M' is already used to symbolize 'The maid killed Watson'. Once we selected 'H' to symbolize 'The butler hated Watson' no obvious mnemonic letter is available to symbolize 'The maid hated Watson'. Hence we arbitrarily chose the letter 'A'.

It is time to make the process of symbolizing English sentences somewhat more systematic. We have already seen that when the sentences to be symbolized are compound sentences whose main connective is 'or' or 'and', or are equivalent to such sentences, they can be symbolized as compound sentences of *SL*. But the question of when an English sentence is, or is equivalent to, a compound sentence that can be symbolized as a compound sentence of *SL* is often more complicated than the rather simple examples we have used would suggest. We have provided a table that gives rough English interpretations of the connectives of *SL*, but we need to be more precise. We begin with the concept of the **truth-functional use** of a sentential connective:

> A sentential connective of a formal or natural language is *used truth-functionally* if and only if it is used to generate a compound sentence

from one or more sentences in such a way that the truth-value of the generated compound is wholly determined by the truth-values of those one or more sentences from which the compound is generated, no matter what those truth-values may be.

English contains a number of sentential connectives that are always or nearly always used truth-functionally, some that are frequently used truth-functionally but also frequently used non-truth-functionally, and many that have no truth-functional uses. The connectives of *SL*, on the other hand, have only truth-functional uses. Because an understanding of how the connectives of *SL* work is required to appropriately symbolize English sentences in *SL* we here present the semantics or interpretation of the connectives of *SL*. The full semantics of *SL* is given in Chapter 3.

The following 'characteristic truth-tables' for the connectives of *SL fully define* the connectives *of SL*. It follows from these definitions that the connectives of *SL* have only truth-functional uses.

Negation

P	~ P
T	F
F	T

Conjunction

P	Q	(P & Q)
T	T	T
T	F	F
F	T	F
F	F	F

Disjunction

P	Q	(P ∨ Q)
T	T	T
T	F	T
F	T	T
F	F	F

Material Conditional

P	Q	(P ⊃ Q)
T	T	T
T	F	F
F	T	T
F	F	T

Material Biconditional

P	Q	(P ≡ Q)
T	T	T
T	F	F
F	T	F
F	F	T

The truth-value of a compound sentence of *SL* is fully determined by the truth-value(s) of its immediate component(s). Characteristic truth-tables display, in the columns to the left of the vertical line, all the combinations of truth-values the immediate components of compounds generated by the connective being defined can have. The truth-value the compound sentence has for each of those combinations of truth-values is displayed to the right of the vertical line.

NEGATIONS

The tilde is the connective of *SL* that roughly corresponds to the English connective 'It is not the case that'. The tilde turns sentences of *SL* that have the truth-value **T** into sentences that have the truth-value **F**—this is indicated by the first row of the table for negations—and it turns sentences of *SL* that have the truth-value **F** into sentences that have the truth-value **T**—this is indicated by the second row of the table. This means that a negation of *SL* is true if and only if the negated sentence is false.

CONJUNCTIONS

The ampersand is the connective of *SL* that roughly corresponds to the English connective 'and'. The characteristic truth-table for conjunctions has a '**T**' beneath the ampersand in the first row and only in the first row, and this is the only row in which there is a '**T**' beneath both '**P**' and '**Q**'. This means that a conjunction of *SL* is true if and only if both conjuncts are true and is false if and only if at least one conjunct is false.

DISJUNCTIONS

The wedge is the connective of *SL* that roughly corresponds to the English connective 'or'. The characteristic truth-table for disjunctions has a '**T**' under the wedge in every row in which there is a '**T**' beneath '**P**' *or* beneath '**Q**'. This means that a disjunction of *SL* is true if and only if at least one of its disjuncts is true and is false if and only if both of its disjuncts are false.

MATERIAL CONDITIONALS

The horseshoe is the connective of *SL* that roughly corresponds to the two-part English connective 'if . . . then'. The characteristic truth-table for material conditionals has a '**T**' under the horseshoe in every row in which there is a '**T**' under the consequent and in every row in which there is an '**F**' under the antecedent. This means that a material conditional of *SL* is true if and only if either the antecedent is false or the consequent is true. It is false if and only if the antecedent is true and the consequent is false.

MATERIAL BICONDITIONALS

The triple bar is the connective of *SL* that roughly corresponds to the English connective 'if and only if'. The characteristic truth-table for material biconditionals has a '**T**' under the triple bar in the row in which there is a '**T**' under

both '**P**' and '**Q**' and in the row in which there is an '**F**' under both '**P**' and '**Q**'. This means that a material biconditional of *SL* is true if and only if its immediate components have the same truth-value and is false if and only if its immediate components have different truth-values.

We can now lay out the two-step process we will use in symbolizing English sentences in *SL*. The first step is to construct a truth-functional paraphrase of the sentence or sentences to be symbolized. Some examples will be useful. We have seen that sentences of English that can reasonably be symbolized as truth-functional compounds of *SL* are not themselves always compound sentences—for example, while 'The butler and the maid both hated Watson' is not a conjunction of two English sentences, it is equivalent to a conjunction of two English sentences. So the first step in symbolizing this sentence is to paraphrase it as

> The butler hated Watson and the maid hated Watson.

Our paraphrase is an explicit conjunction of two sentences. We have underlined the connective, in this case 'and', to indicate that it is being used purely truth-functionally. Here is another example:

> The pitcher for the home team will be either Betty or Margaret.

We paraphrase this sentence as an explicit disjunction of two sentences:

> Betty will be the pitcher for the home team or Margaret will be the pitcher for the home team,

again underlining the connective to indicate it is being used purely truth-functionally in the paraphrase.

We can paraphrase 'Margaret will be the pitcher for the home team if her shoulder has healed' as

> If Margaret's shoulder has healed then Margaret will be the pitcher for the home team.

In this case we made two changes to the original sentence. First, we reversed the order in which the component sentences occur, placing the sentence following 'if' first. Second, we replaced 'she' with 'Margaret' so that each sentence written alone is completely interpreted. That is, we have explicitly indicated to whom 'her' refers.

The purpose of paraphrasing an English sentence is to produce a sentence that can easily be symbolized in *SL*. When an English sentence is or can be paraphrased as a truth-functional compound, the paraphrase will obviously be a truth-functionally compound sentence and its structure will mirror the structure of the sentence of *SL* we will use to symbolize it. If an English sentence cannot be paraphrased as a truth-functionally compound sentence, then we will let that sentence serve as its own paraphrase and will symbolize

it as an atomic sentence of *SL*.[5] Because paraphrases that are truth-functional compounds mirror the structure of the sentences that will symbolize them in *SL*, we will speak of them, as well as of sentences of *SL*, as being negations, conjunctions, disjunctions, material conditionals, and material biconditionals and as having main connectives and immediate components.

Symbolizing truth-functional paraphrases in *SL* is straightforward. If a paraphrase is not a truth-functional compound, we will symbolize the paraphrase as an atomic sentence of *SL*. If it is a truth-functional compound, then we symbolize it as a truth-functional compound of *SL* with the same structure.

Here is a group of sentences that can be paraphrased as negations and symbolized as sentences of the form ~ **P**. Note that only one of these English sentences contains the word 'not'.

The United States isn't a confederation of states.
Chlorine is a nonmetal.
Aristotle was unmarried.
Not everyone likes hip hop music.
Someone isn't telling the truth.
No one always tells the truth.

Paraphrasing the first four examples is straightforward:

It is not the case that the United States is a confederation of states.
It is not the case that chlorine is a metal.
It is not the case that Aristotle was married.
It is not the case that everyone likes hip hop music.

The fifth and sixth examples are less straightforward. It would be a mistake to paraphrase 'Someone isn't telling the truth' as

It is not the case that someone is telling the truth,

because this purported paraphrase is equivalent to 'No one is telling the truth', a far stronger claim than 'Someone isn't telling the truth'. A correct paraphrase for the fifth sentence is

It is not the case that everyone is telling the truth.

The sixth sentence, 'No one always tells the truth', is a denial of the claim made by the sentence 'There is someone who always tells the truth' and is therefore correctly paraphrased as

It is not the case that there is someone who always tells the truth.

[5]We refer to the sentences that result from the paraphrase process, including those that serve as their own paraphrases, as 'truth-functional paraphrases'. Note that those that serve as their own paraphrases are not truth-functionally compound sentences.

Using the symbolization key

U: The United States is a confederation of states.
C: Chlorine is a metal.
A: Aristotle was married.
E: Everyone likes hip hop music.
T: Everyone is telling the truth.
S: There is someone who always tells the truth.

we can symbolize the paraphrases as '~ U', '~ C', '~ A', '~ E', '~ T', and '~ S', respectively.

The following English sentences can be paraphrased as conjunctions:

Handel and Mozart both composed operas.
Beethoven composed symphonies and piano sonatas.
Beethoven composed nine symphonies, as did Mahler.
Mahler's *Kindertoten Lieder* are beautiful but also sad.

Here are our paraphrases:

Handel composed operas and Mozart composed operas.
Beethoven composed symphonies and Beethoven composed piano sonatas.
Beethoven composed nine symphonies and Mahler composed nine symphonies.
Mahler's *Kindertoten Lieder* are beautiful and Mahler's *Kindertoten Lieder* are sad.

The first three paraphrases clearly capture the full meaning of the sentences being paraphrased. But it is arguable that the fourth paraphrase does not capture the full meaning of the original, which uses the word 'but' rather than 'and' as a sentential connective. The word 'but' suggests a contrast or tension between a composition's being beautiful and its being sad, such that it is surprising to hear that a beautiful composition is at the same time sad. This suggestion is not present in the truth-functional paraphrase. Nonetheless, the paraphrase does capture what is asserted, rather than just suggested, by the original sentence. So what is asserted by the original sentence is true if both 'Mahler's *Kindertoten Lieder* are beautiful' and 'Mahler's *Kindertoten Lieder* are sad' are true.

Just how much of the content of the original sentence a truth-functional paraphrase must capture to be a reasonable paraphrase may depend on the context, but usually the loss of a suggestion will not matter to the logical analysis of a sentence or passage that has been truth-functionally paraphrased. Other English words that may be rendered as a truth-functional 'and' in paraphrases, some of which may also suggest an element of surprise or unexpectedness, include 'nevertheless', 'moreover', 'while', 'although', and 'albeit'.

Using the symbolization key

H: Handel composed operas.
M: Mozart composed operas.
S: Beethoven composed symphonies.
P: Beethoven composed piano concertos.
N: Beethoven composed nine symphonies.
A: Mahler composed nine symphonies.
B: Mahler's *Kindertoten Lieder* are beautiful.
K: Mahler's *Kindertoten Lieder* are sad.

our paraphrases can be symbolized as 'H & M', 'S & P', 'N & A', and 'B & K', respectively.

All of the following sentences can be paraphrased as disjunctions:

Maggie or Ronald will win the race.
Jim likes either jazz or hip hop.
Karen likes classical music, unless her tastes in music have changed.
At least one of the two finalists, Betty and Larry, will be very happy.

Appropriate paraphrases are

Maggie will win the race or Ronald will win the race.
Jim likes jazz or Jim likes hip hop.
Karen likes classical music or Karen's tastes in music have changed.
Betty will be very happy or Larry will be very happy.

The first two paraphrases are straightforward. The third sentence that we have paraphrased as a disjunction contains the sentential connective 'unless' rather than 'or', but it is clear that 'unless' in this sentence is correctly rendered as the truth-functional connective 'or'. If someone asks us what type of music Karen likes, and we know that Karen liked classical music the last time we talked with her, which was a year ago, we might say "Karen likes classical music, unless her tastes in music have changed". Here we mean that either she likes classical music (as she did a year ago) or her tastes have changed (in which case she might no longer like classical music). The fourth paraphrase may not jump out as an obvious truth-functional paraphrase if we only look at the original sentence, but it clearly captures what the original is saying.

We will use the following symbolization key to symbolize these paraphrases in *SL*:

M: Maggie will win the race
R: Ronald will win the race.
J: Jim likes jazz.
H: Jim likes hip hop.
K: Karen likes classical music.

C: Karen's tastes in music have changed.
B: Betty will be very happy.
L: Larry will be very happy.

Our symbolizations are 'M ∨ R', 'J ∨ H', 'K ∨ C', and 'B ∨ L', respectively.

Recall that disjunctions of *SL*, sentences of *SL* of the form **P** ∨ **Q**, are true if either **P** is true, or **Q** is true, or both are true. It is sometimes claimed that there are two uses of 'or' in English, one in which the 'or' means 'either this or that or both' and the other in which it means 'either this or that and not both'. The former is described as the inclusive use of 'or', the latter the exclusive use. It may appear that 'or' is being used in the exclusive sense in sentences such as

Louise Penny or Miles Blunt will win this year's Silver Dagger Award,

for surely two authors can't win the mystery writers' award in question. But we have made a misstep here. First, for all we know, the awards committee does sometimes award two or more authors the Silver Dagger Award in a single year. More importantly, if the rules governing the awarding of the Silver Dagger allow for only one winner per year, then *it is those rules*, not the meaning of 'or', that keep both Louise Penny and Miles Blunt from winning this year's award.

Those who think 'or' does have an exclusive use in English often cite the use of 'or' in choices we are offered. When a spokesperson for a state lottery announces that the grand prize winner can choose to receive $12 million in a lump sum or $1 million per year for the next fifteen years, we all know the winner won't be able to get both $12 million in a lump sum and $1 million a year for the next fifteen years. It is clearly one or the other and not both. And when studying a menu that contains the sentence 'With the Chef's Special you may have either an egg roll or hot and sour soup', almost everyone will know that they cannot get both an egg roll and hot and sour soup with the Chef's Special, at least not without paying extra. Whether we know these things because we recognize that 'or' is being used, in these cases of proffered choices, in the exclusive sense, or because we know about the customs and conventions prevalent when we are given a choice, is another matter.

If there is an exclusive sense of 'or' in English we can capture that sense in *SL*. For example, if we want to say that that Sally is in either New York or Chicago but not both, we can do so with the following paraphrase and symbolization:

(Sally is in New York or Sally is in Chicago) and it is not the case that (Sally is in New York and Sally is in Chicago)

(N ∨ C) & ~ (N & C)

Here we have used 'N' to symbolize 'Sally is in New York' and 'C' to symbolize 'Sally is in Chicago'.

The following four sentences can be paraphrased as material conditionals:

> If Sheila's hard work is recognized, she will get a raise.
> Pamela will get a raise if everyone does.
> Cynthia will get a raise provided she finishes her current assignment on time.
> Harry will get a raise only if his boss is a damn fool.

Appropriate paraphrases of our four examples are

> If Sheila's hard work is recognized then Sheila will get a raise.
> If everyone will get a raise then Pamela will get a raise.
> If Cynthia finishes her current assignment on time then Cynthia will get a raise.
> If Harry will get a raise then Harry's boss is a damn fool.

We will use the following symbolization key:

> S: Sheila's hard work is recognized.
> R: Sheila will get a raise.
> E: Someone will get a raise.
> P: Pamela will get a raise.
> F: Cynthia finishes her current assignment on time.
> C: Cynthia will get a raise.
> H: Harry will get a raise.
> B: Harry's boss is a damn fool.

The first paraphrase is straightforward and can be symbolized as 'S ⊃ R'. Our paraphrase for the second sentence reverses the order of 'Pamela will get a raise' and 'Everyone will [get a raise]' in order to place the 'if' part of the sentence at the beginning of the paraphrase. This sentence is symbolized as 'E ⊃ P'. The third example illustrates that not every English sentence that can be paraphrased as a material conditional contains the word 'if'. In this example, 'provided that' plays the role of 'if'. This paraphrase is symbolized as 'F ⊃ C'. The sentences 'Cynthia will get a raise, assuming she finishes her current assignment on time' and 'Should Cynthia finish her current on time, she will get a raise' can both be paraphrased as 'If Cynthia finishes her current assignment on time then Cynthia will get a raise'.

The fourth example is intended to illustrate the difference between 'if' and 'only if'. Note that the sentence being paraphrased is 'Harry will get a raise *only if* his boss is a damn fool', *not* 'Harry will get a raise *if* Harry's boss is a damn fool'. The latter tells us that if Harry's boss is a damn fool, then Harry will get a raise. But the former tells us that if Harry does get a raise, then his boss is a damn fool. These are not equivalent claims. Our symbolization is 'H ⊃ B'.

To put the point more generally, when we are told that **p** only if **q** we are being told that if **p** is true **q** is true as well, and this is not the same as saying that if **q** is true **p** is also true. A university may require that all students complete two semesters of a foreign language before graduating. If so, then Kurt will graduate only if he has two semesters of a foreign language. But the university undoubtedly has other graduation requirements, for example, completing 120 semester hours of academic credit. If Kurt doesn't meet these additional requirements, he won't graduate—*even if* he does have two semesters of a foreign language.

English sentences of the form

p only if **q**

should therefore be paraphrased as sentences of the form

if **p** then **q**,

that is, the sentential component *following* 'if' becomes the *consequent*, not the antecedent, of the paraphrase.

On the other hand, sentences of the forms

if **p** (then) **q**
q if **p**
q provided that **p**
assuming **p**, **q**
q, assuming **p**

should all be paraphrased as sentences of the form

if **p** then **q**.

The following sentence can be straightforwardly paraphrased as a material biconditional:

> The global financial crisis will be resolved if but only if the world's major economic powers cut long-term spending.

Here is our paraphrase:

> The global financial crisis will be resolved if and only if the world's major economic powers cut long-term spending.

Note that in constructing this truth-functional paraphrase we replaced 'if but only if' with 'if and only if', and we did so for the reason that we earlier replaced the simple connective 'but' with 'and'. That is, the use of 'but' suggests, but does not assert, that it is not clear or obvious that the world's major economic

powers will cut long-term spending. Our paraphrase can be symbolized as 'G ≡ W', using the following symbolization key:

> G: The global financial crisis will be resolved.
> W: The world's major economic powers cut long-term spending.

Although sentences of English that are appropriately paraphrased as material biconditionals often contain the expression 'if and only if' or the variant 'if but only if', sentences containing the expression 'just in case' can also sometimes be paraphrased as material biconditionals. Consider the following sentences:

> Fighter pilots carry parachutes just in case they have to eject from their planes.
> The House will pass the tax reform bill just in case there is great public pressure for tax reform.

The first sentence clearly should *not* be paraphrased as 'Fighter pilots carry parachutes if and only if fighter pilots have to eject from their planes', for the paraphrase says that fighter pilots carry parachutes when they have to eject and only at such times. Clearly the English sentence allows for fighter pilots carrying parachutes at all times, whether or not these are times when they have to eject. The English sentence should therefore not be interpreted as a claim about when pilots carry parachutes, but rather as an explanation of why they carry parachutes, namely, to be prepared for emergencies. But the second sentence can correctly be paraphrased as a material biconditional:

> The House will pass the tax reform bill if and only if there is great public pressure for tax reform.

This can be symbolized in *SL* as 'H ≡ G', using the following symbolization key:

> H: The House will pass the tax reform bill
> G: There is great public pressure for reform.

2.2**E** EXERCISES

1. Paraphrase and then symbolize each of the following sentences, indicating which sentences the sentence letters you use symbolize.

a. Bob isn't a marathon runner.
*b. Albert and Bob are joggers.
c. If Carol is a jogger she is also a marathon runner.
*d. Some joggers are marathon runners.
e. Carol will run in the Boston marathon if and only if Albert does.
*f. Not all joggers are marathon runners.
g. Either Carol or Albert will run in the Boston marathon.
*h. If Carol will run in the Boston marathon so will Albert.

2. Paraphrase and then symbolize each of the following sentences, indicating which sentences the sentence letters you use symbolize.

a. If Felice vacations in Bermuda so will Clarence.
*b. Veronica will vacation in Bermuda only if Clarence does.
c. Veronica will vacation in Bermuda if Felice does.
*d. Either Clarence or Robert will vacation in Bermuda.
e. Veronica will vacation in Bermuda provided that Clarence will.
*f. Robert won't vacation in Bermuda.

2.3 MORE COMPLEX SYMBOLIZATIONS

In this section we will paraphrase and symbolize more complex sentences and sets of sentences in *SL*. Along the way, we will also illustrate and discuss some of the finer nuances of the symbolization process. We shall continue to symbolize English sentences in two stages, first constructing truth-functional paraphrases of the sentence or sentences to be symbolized and then symbolizing the paraphrases in *SL*. We begin by laying out guidelines for the construction of truth-functional paraphrases:

1. Determine whether the sentence to be symbolized can reasonably be paraphrased as a truth-functionally compound sentence.
   a. If it cannot, use the sentence as its own paraphrase.
   b. If it can, determine whether its immediate component(s) and their components can also reasonably be paraphrased as truth-functional compounds.
2. Use one or more of the connectives '<u>it is not the case that</u> . . .', '. . . <u>and</u> . . .', '. . . <u>or</u> . . .', '<u>if</u> . . . <u>then</u> . . .', and '. . . <u>if and only if</u> . . .' to construct truth-functional paraphrases of each sentence that can reasonably be paraphrased as a truth-functionally compound sentence.
3. Where applicable, use parentheses and square brackets to indicate which sentences are the immediate components of truth-functional compounds.
4. When paraphrasing an argument, present the paraphrased premises and conclusion in standard form. That is, list the paraphrased premises, draw a line beneath the last premise, and then list the paraphrased conclusion.
5. Reword the sentences being paraphrased so that all immediate components of the paraphrase are complete sentences with no cross-references between components, and if there are two or more wordings of the same claim, use just one wording in the paraphrase.

Some explanatory comments are in order. Many English sentences are *not* compound sentences. Among them are 'Canada is a member of the Commonwealth of Nations' and 'George W. Bush was the 43rd President of the United States'.

Guideline 1 specifies that such sentences should serve as their own paraphrases, as these are not compound sentences. Each such sentence will of course be symbolized by a single sentence letter of *SL*. There are also English sentences that contain sentences as proper constituents that should be used as their own paraphrases. One example is 'Archie believes that playing the lottery is the best way to get rich'. This sentence is formed by placing 'Archie believes that' in front of 'Playing the lottery is the best way to get rich'. But 'Archie believes that playing the lottery is the best way to get rich' cannot be paraphrased as a truth-functionally compound sentence in which 'Playing the lottery is the best way to get rich' is a component, because the truth-value of the former is not determined by the truth-value of the latter. Given only that a sentence is true, it does not follow that Archie believes it, and it does not follow that he does not believe it. Similarly, given only that a sentence is false it follows neither that Archie believes it nor that he does not believe it. Hence, 'Archie believes that playing the lottery is the best way to get rich' and all other sentences that cannot reasonably be paraphrased as truth-functional compounds should be used as their own paraphrases and symbolized as atomic sentences of *SL*. (Non-truth-functionally compound sentences will be further discussed in Section 2.4.)

Guideline 2 simply lists the connectives that are available for constructing truth-functionally compound paraphrases of English sentences. We shall prove in Chapter 6 that the structure of *every* truth-functionally compound sentence of English, no matter how complex, can be captured in a paraphrase that uses only the five truth-functional connectives listed in Guideline 2.

Guideline 3 calls for using parentheses and/or square brackets in paraphrases to indicate which sentences are being connected by which binary connectives. Doing so serves to eliminate ambiguities and also to mirror the syntax of *SL*, where parentheses are necessary to indicate grouping. Some English sentences that contain multiple sentential connectives are ambiguous, from a syntactic point of view. Consider

> Paul is taking saxophone lessons and Ellen is taking saxophone lessons or Karen is taking saxophone lessons.

Someone who asserts this sentence might intend to say that Paul is taking saxophone lessons and either Ellen or Karen is also taking saxophone lessons, making 'and' the main connective of the sentence. Alternatively, the intent might be to say that either Paul and Ellen are both taking saxophone lessons or Karen is taking saxophone lessons, making 'or' the main connective. That is, an English sentence of the form

> **p** and **q** or **r**

is syntactically ambiguous. From the syntax alone we don't know whether the intended meaning is

> **p** and (**q** or **r**)

or

(**p** and **q**) or **r**

In actual English discourse the context or the tone of voice or the emphasis a speaker places on one connective or the other often removes ambiguities like this. But *SL* does not contain ambiguities, and we shall reflect this in our truth-functional paraphrases by using parentheses to remove such ambiguities.

Some English sentences containing multiple sentential connectives are not ambiguous. These include sentences that can be recast as what we will call 'extended conjunctions' and 'extended disjunctions'. An example of the first sort is

Paul, Ellen, and Karen are all taking saxophone lessons.

This sentence can clearly be recast as:

Paul is taking saxophone lessons and Ellen is taking saxophone lessons and Karen is taking saxophone lessons.

A sentence that can be recast as an extended disjunction is

Either Paul or Ellen or Karen is taking saxophone lessons.

This sentence can be recast as

Paul is taking saxophone lessons or Ellen is taking saxophone lessons or Karen is taking saxophone lessons.

In neither case do we need to "figure out" which connective of the recast sentence is the main connective, as neither the original sentences nor their recastings are ambiguous. The first is true if and only if all three of the named individuals are taking saxophone lessons, the second if and only if at least one of them is taking saxophone lessons. However, because we will have occasion to symbolize such sentences in *SL*, where parentheses are required in sentences containing multiple ampersands or multiple wedges, we will use parentheses and/or square brackets to identify a main connective in paraphrasing such sentences. Of course there are multiple ways of doing this—that is, we can use parentheses to make the first occurrence of 'and' (or 'or'), or the second or the third, the main connective. All of these paraphrases are equally appropriate. For example, here are two equally acceptable truth-functional paraphrases of our original conjunction:

Paul is taking saxophone lessons and (Ellen is taking saxophone lessons and Karen is taking saxophone lessons),

and

> (Paul is taking saxophone lessons and Ellen is taking saxophone lessons) and Karen is taking saxophone lessons.[6]

Guideline 4 is straightforward. But guideline 5 needs some explanation. One way to eliminate cross-reference is to replace pronouns with the terms for which they are going proxy. In paraphrasing

> If John is late he will have a good excuse,

it is obviously appropriate to replace 'he' with 'John'. More complex rewordings are often necessary. Suppose we are asked to paraphrase the following very simple argument:

> If Sally is late for class she will miss the discussion of Darwin's study of pigeons. Sally will be late for class, so she will miss the discussion.

A paraphrase of this argument that does *not* follow guideline 5 is

> If Sally is late for class then Sally will miss the discussion of Darwin's study of pigeons.
>
> Sally will be late for class.
>
> ---
>
> Sally will miss the discussion.

This paraphrase contains the following four distinct component sentences: 'Sally is late for class', 'Sally will miss the discussion of Darwin's study of pigeons', 'Sally will be late for class', and 'Sally will miss the discussion'. Using the four sentence letters 'S', 'M', 'W', and 'D', respectively, to symbolize these component sentences generates the following argument of *SL*:

$$\begin{array}{l} S \supset M \\ W \\ \hline D \end{array}$$

This is not a valid argument of *SL*. Yet the original English argument is valid. The problem is that our paraphrase does not reflect the fact that in the original argument 'Sally is late for class' and 'Sally will be late for class' express the same claim. It is also implicit in the original argument that the discussion Sally

[6]Since the grouping we use in extended conjunctions and extended disjunctions is arbitrary, it may appear that we should allow extended conjunctions and disjunctions in *SL* without the use of parentheses to indicate which connective is the main connective. We do not and cannot allow this because to apply the rules of the logical systems that we develop for *SL*, it will be necessary to know for every use of a binary connective which sentences are connected by that connective.

will miss is the discussion of Darwin's study of pigeons, not some other discussion. Hence the following rewording provides an appropriate paraphrase:

> If Sally will be late for class then Sally will miss the discussion of Darwin's study of pigeons.
>
> Sally will be late for class.
>
> ---
>
> Sally will miss the discussion of Darwin's study of pigeons.

Using 'S' to symbolize 'Sally will be late for class' and 'M' to symbolize 'Sally will miss the discussion of Darwin's study of pigeons' we can symbolize this paraphrased argument as follows:

$$\begin{array}{l} S \supset M \\ S \\ \hline M \end{array}$$

This argument *is* valid in *SL*.

The following argument calls for even more extensive rewording in the paraphrase:

> Either Jim will not pass the test or Jim spent last night studying logic. Jim's night was not spent poring over his logic text. Hence, Jim will fail the test.

Here is an *inappropriate* paraphrase of this argument, a paraphrase that ignores our fifth guideline:

> It is not the case that Jim will pass today's logic test or Jim spent last night studying logic.
>
> It is not the case that Jim's night was spent poring over his logic text.
>
> ---
>
> Jim will fail the test.

Symbolizing this paraphrase requires the use of four different sentence letters of *SL*:

$$\begin{array}{l} \sim J \vee S \\ \sim P \\ \hline F \end{array}$$

This is not a valid argument of *SL*. Yet, the English language argument with which we started is valid. Our paraphrase fails to reflect the fact that in this argument, 'Jim will not pass the test' and 'Jim will fail the test' are intended to

be equivalent—these two sentences are making the same claim.[7] So too, 'Jim spent last night studying logic' is making the same claim as is 'Jim's night was spent poring over his logic text'.

A better paraphrase of our argument is

It is not the case that Jim will pass the logic test or Jim spent last night studying logic.

It is not the case that Jim spent last night studying logic.

It is not the case that Jim will pass the logic test.

This argument can be symbolized in *SL* using just two sentence letters, with 'J' symbolizing 'Jim will pass the logic test' and 'S' symbolizing 'Jim spent last night studying logic':

$$\begin{array}{l} \sim J \vee S \\ \sim S \\ \hline \sim J \end{array}$$

And this argument will turn out to be a valid argument of *SL*.

Using our guidelines for constructing paraphrases we will now paraphrase and symbolize additional sentences, passages, and arguments in *SL*. Our first group of sentences concerns contemporary mystery writers. We will use the following symbolization key:

F: Ted has read *A Fine Red Rain*.
B: Ted has read *Bury Your Dead*.
D: Ted has read *The Old Fox Deceived*.
R: Ted has read *Rough Country*.

The first sentence we will paraphrase and symbolize is

Ted has read all of the books *A Fine Red Rain*, *Bury Your Dead*, *The Old Fox Deceived*, and *Rough Country*.

The paraphrase is straightforward:

(Ted has read *A Fine Red Rain* and Ted has read *Bury Your Dead*) and (Ted has read *The Old Fox Deceived* and Ted has read *Rough Country*),

[7]Failing and not passing are not always the same. If Sally is not enrolled in Jim's logic class, then she does not pass the test in that class, because she does not take it, but it is not true that she fails that test. More generally, that two claims can, in a given context, have a common paraphrase does not show that those two claims can be paraphrased as the same claim in every context.

as is the symbolization:

(F & B) & (D & R)

Because all of the connectives are ampersands, we could have grouped the conjuncts of both our paraphrase and our symbolization in several other ways, including:

Ted has read *A Fine Red Rain* and [Ted has read *Bury Your Dead* and (Ted has read *The Old Fox Deceived* and Ted has read *Rough Country*)]

F & [B & (D & R)]

Our next sentence can be paraphrased as an extended disjunction:

Ted has read at least one of the books *A Fine Red Rain*, *Bury Your Dead*, *The Old Fox Deceived,* and *Rough Country*.

An appropriate paraphrase is

(Ted has read *A Fine Red Rain* or Ted has read *Bury Your Dead*) or (Ted has read *The Old Fox Deceived* or Ted has read *Rough Country*).

(The grouping in this paraphrase is also arbitrary.) The symbolization is

(F ∨ B) ∨ (D ∨ R)

The sentence

Ted hasn't read any of the books *A Fine Red Rain*, *Bury Your Dead*, *The Old Fox Deceived*, or *Rough Country*

can be paraphrased as

(It is not the case that Ted has read *A Fine Red Rain* and it is not the case that Ted has read *Bury Your Dead*) and (it is not the case that Ted has read *The Old Fox Deceived* and it is not the case that Ted has read *Rough Country*)

and symbolized as

(~ F & ~ B) & (~ D & ~ R)

This sentence can also (and equivalently) be paraphrased as

It is not the case that [(Ted has read *A Fine Red Rain* or Ted has read *Bury Your Dead*) or (Ted has read *The Old Fox Deceived* or Ted has read *Rough Country*)]

and symbolized as

$$\sim [(F \vee B) \vee (D \vee R)]$$

The sentence

> Ted has read one but not both of the books *A Fine Red Rain* and *Bury Your Dead*

can be paraphrased as

> (Ted has read *A Fine Red Rain* or Ted has read *Bury Your Dead*) and it is not the case that (Ted has read *A Fine Red Rain* and Ted has read *Bury Your Dead*)

and symbolized as

$$(F \vee B) \ \& \sim (F \ \& \ B)$$

Note that 'F ∨ B' alone is not an acceptable symbolization of the sentence as the wedge of *SL* is inclusive, that is, the sentence 'F ∨ B' is true if Ted has read either or *both* of the books in question. Our current example can also be paraphrased and symbolized as follows:

> (Ted has read *A Fine Red Rain* and it is not the case that Ted has read *Bury Your Dead*) or (Ted has read *Bury Your Dead* and it is not the case that Ted has read *A Fine Red Rain*).

$$(F \ \& \sim B) \vee (B \ \& \sim F)$$

The sentence

> Ted has read exactly two of the books *A Fine Red Rain*, *Bury Your Dead*, and *The Old Fox Deceived*.

lists three books and says that Ted has read exactly two of them, but it doesn't say which two. We can capture this claim by spelling out the three possibilities in our paraphrase:

> [(Ted has read *A Fine Red Rain* and Ted has read *Bury Your Dead*) and it is not the case that Ted has read *The Old Fox Deceived*] or ([(Ted has read *A Fine Red Rain* and Ted has read *The Old Fox Deceived*) and it is not the case that Ted has read *Bury Your Dead*] or [(Ted has read *Bury Your Dead* and Ted has read *The Old Fox Deceived*) and it is not the case that Ted has read *A Fine Red Rain*]).

This is symbolized as

$$[(F \,\&\, B) \,\&\, \sim D] \vee ([(F \,\&\, D) \,\&\, \sim B] \vee [(B \,\&\, D) \,\&\, \sim F])$$

Again, the grouping of the disjuncts is arbitrary, as is the grouping of the conjuncts within each disjunction. The sentence can also be paraphrased and symbolized as saying that Ted has read at least two of the books, but not all three:

> ([(Ted has read *A Fine Red Rain* and Ted has read *Bury Your Dead*) or (Ted has read *A Fine Red Rain* and Ted has read *The Old Fox Deceived*)] or (Ted has read *Bury Your Dead* and Ted has read *The Old Fox Deceived*)) and it is not the case that [(Ted has read *A Fine Red Rain* and Ted has read *Bury Your Dead*) and Ted has read *The Old Fox Deceived*].
>
> $([(F \,\&\, B) \vee (F \,\&\, D)] \vee (B \,\&\, D)) \,\&\, \sim [(F \,\&\, B) \,\&\, D]$

We next paraphrase a series of sentences concerning various genres of music. We follow each paraphrase with a symbolization key and a symbolization of the paraphrase.

- Jazz is invigorating and classical music is uplifting, but neither is broadly popular.

This sentence can be paraphrased as a conjunction whose left conjunct is itself a conjunction and whose right conjunct is the negation of a disjunction:

> (Jazz is invigorating and classical music is uplifting) and it is not that case that (jazz is broadly popular or classical music is broadly popular).
>
> J: Jazz is invigorating.
> C: Classical music is uplifting.
> B: Jazz is broadly popular.
> P: Classical music is broadly popular.
>
> $(J \,\&\, C) \,\&\, \sim (B \vee P)$

- Opera enthusiasts are small in number and very devoted to opera, but not always tolerant of other forms of music.

This sentence can also be paraphrased as a conjunction:

> Opera lovers are small in number and (opera lovers are very devoted to opera and it is not the case that opera lovers are always tolerant of other music).

O: Opera lovers are small in number.
D: Opera lovers are very devoted to opera.
T: Opera lovers are always tolerant of other forms of music.

O & (D & ~ T)

- Country and western music is wildly popular and is both funky and funny.

Our paraphrase is

Country and western music is wildly popular and (country and western music is funny and country and western music is funky).

C: Country and western music is wildly popular.
N: Country and western music is funny.
K: Country and western music is funky.

C & (N & K)

- Folk music was the rage in the 60s but has only a small following today, and it will make a comeback if and only if country and western music proves to be a fad, but it won't prove to be a fad.

A little reflection will show that our paraphrase should contain three conjunctions, one formed by the first 'but', a second by the 'and' occurring before 'it will make a comeback', and the third formed by the 'but' occurring before 'it won't prove to be a fad'. The paraphrase will also contain a material biconditional. We can treat the 'and' or either of the occurrences of 'but' as the main connective. We choose to treat the occurrence of 'and' as the main connective:

(Folk music was the rage in the 60s and folk music has only a small following today) and [(folk music will make a comeback if and only if country and western music proves to be a fad) and it is not the case that folk music will prove to be a fad].

R: Folk music was the rage in the 60s.
F: Folk music has only a small following today.
C: Folk music will make a comeback.
P: Country and western music proves to be a fad.

(R & F) & [(C ≡ P) & ~ P]

- Either hip hop is more popular than it deserves to be or there is more to it than there seems to be, but there isn't.

This sentence can be paraphrased as a conjunction whose left conjunct is a disjunction and whose right conjunct is a negation:

> (Hip hop is more popular than it deserves to be or there is more to hip hop than there seems to be) and it is not the case that there is more to hip hop than there seems to be.

> H: Hip hop is more popular than it deserves to be.
> M: There is more to hip hop than there seems to be.

> $(H \vee M) \ \& \sim M$

There are several points to note about the paraphrases and symbolizations we have just given. First, in paraphrasing our first and second sentences, we treated 'but' as surrogate for 'and'. Conversely, we point out that the word 'and' in the phrase 'country and western music' is not being used as a truth-functional connective. Our paraphrases and symbolizations also demonstrate that the mnemonic device of selecting a sentence letter based on an important word in the sentence to be symbolized is often of limited use. The paraphrase of our third sentence yielded three component sentences, all about country and western music:

> C: Country and western music is wildly popular.
> N: Country and western music is funny.
> K: Country and western music is funky.

We chose to use 'C' to symbolize the first of these, and 'C' may well remind us that it is symbolizing a sentence about country and western music, but it cannot remind us of which of the three component sentences about country and western music it symbolizes. We used 'N' and 'K' to symbolize the other two component sentences, and 'N' may serve to remind us that it symbolizes a sentence containing 'funny' (though we could equally well have used it to symbolize the third component sentence, which contains the word 'funky'). Similarly, our paraphrase of our fourth sentence yielded four component sentences, three of them about folk music. Again, the conclusion to be drawn from these examples is that the mnemonic device of using sentence letters that remind us of an important word in the sentence being symbolized is often of limited use.

We next paraphrase and symbolize several arguments. The first is

> Tim will go to the Blue Olive if and only if it is featuring a jazz trio but Susan will go if and only if the Blue Olive is featuring piano jazz. Ralph will go to the Blue Olive if Susan goes and Tim doesn't. Bill will go to the Blue Olive if they have a country and western band, but they don't. The Blue Olive is featuring piano jazz, not a jazz trio. So neither Tim nor Bill will go to the Blue Olive, but Susan and Ralph will.

Our paraphrase of this argument follows:

> (Tim will go to the Blue Olive if and only if the Blue Olive is featuring a jazz trio) and (Susan will go to the Blue Olive if and only if the Blue Olive is featuring piano jazz).
>
> If (Susan will go to the Blue Olive and it is not the case that Tim will go to the Blue Olive) then Ralph will go to the Blue Olive.
>
> (If the Blue Olive has a country and western band then Bill will go to the Blue Olive) and it is not the case that the Blue Olive has a country and western band.
>
> The Blue Olive is featuring piano jazz and it is not the case that the Blue Olive is featuring a jazz trio.
>
> ---
>
> It is not the case that (Tim will go to the Blue Olive or Bill will go to the Blue Olive) and (Susan will go to the Blue Olive and Ralph will go to the Blue Olive).

Using the symbolization key,

> T: Tim will go to the Blue Olive.
> J: The Blue Olive is featuring a jazz trio.
> S: Susan will go to the Blue Olive.
> P: The Blue Olive is featuring piano jazz.
> R: Ralph will go to the Blue Olive.
> C: The Blue Olive has a country and western band.
> B: Bill will go to the Blue Olive.

we can symbolize the paraphrased argument as follows:

$$(T \equiv J) \ \& \ (S \equiv P)$$
$$(S \ \& \sim T) \supset R$$
$$(C \supset B) \ \& \sim C$$
$$P \ \& \sim J$$
$$\therefore \sim (T \vee B) \ \& \ (S \ \& \ R)$$

In subsequent chapters we will be able to show that this argument of *SL* is valid.

Our second argument is

> If the Outback Coral gets a liquor license before the end of the week it will feature a country and western band this weekend. Monica loves country and western music, and she will go to the Outback Coral this weekend if it does feature a country and western band. Eric hates country and western music but he is infatuated with Monica, and if Monica goes to the Outback Coral this weekend Eric

will also go, even though he will hate every minute of his time there. The Outback Coral will get a liquor license by the end of the week. Hence, Monica and Eric will both go to the Outback Coral this weekend and Eric will hate every minute of his time there.

We paraphrase the passage as an argument in standard form:

If the Outback Coral gets a liquor license before the end of the week then the Outback Coral will feature a country and western band this weekend.

Monica loves country and western music and (if the Outback Coral will feature a country and western band this weekend then Monica will go to the Outback Coral this weekend).

(Eric hates country and western music and Eric is infatuated with Monica) and (if Monica goes to the Outback Coral this weekend then (Eric will go to the Outback Coral this weekend and Eric will hate every minute of his time at the Outback Coral)).

The Outback Coral will get a liquor license by the end of the week.

Monica will go to the Outback Coral this weekend and (Eric will go to the Outback Coral this weekend and Eric will hate every minute of his time at the Outback Coral).

Our paraphrase of the argument yields eight sentences that will be symbolized as atomic sentences of *SL*. Our symbolization key follows. Note that we were not, in every case, able to use a sentence letter that bears a strong mnemonic connection to an important word in the sentence it symbolizes.

O: The Outback Coral will get a liquor license before the end of the week.
C: The Outback Coral will feature a country and western band this weekend.
L: Monica loves country and western music.
M: Monica will go to the Outback Coral this weekend.
H: Eric hates country and western music.
I: Eric is infatuated with Monica.
E: Eric will go to the Outback Coral this weekend.
T: Eric will hate every minute of his time at the Outback Coral.

$O \supset C$

$L \ \& \ (C \supset M)$

$(H \ \& \ I) \ \& \ [M \supset (E \ \& \ T)]$

O

$M \ \& \ (E \ \& \ T)$

In subsequent chapters we will also show that this argument of *SL* is valid.

Our third argument concerns contemporary mystery writers:

> At least one of the authors Louise Penny, Giles Blunt, Donna Leon, and Charles Todd will be nominated for this year's Gold Dagger Award. Everyone who is nominated will publish a new mystery this year. Neither Todd nor Blunt will publish a new mystery this year. Louise Penny will publish a new mystery this year if and only if Donna Leon does. Therefore, both Donna Leon and Louise Penny will publish new mysteries this year and at least one of them will be nominated for Gold Dagger Award.

Here is our paraphrase of this argument:

> (Louise Penny will be nominated for this year's Gold Dagger Award or Giles Blunt will be nominated for this year's Gold Dagger Award) or (Donna Leon will be nominated for this year's Gold Dagger Award or Charles Todd will be nominated for this year's Gold Dagger Award).
>
> [(If Louise Penny will be nominated for this year's Gold Dagger Award then Louise Penny will publish a new mystery this year) and (if Giles Blunt will be nominated for this year's Gold Dagger Award then Giles Blunt will publish a new mystery this year)] and [(if Donna Leon will be nominated for this year's Gold Dagger Award then Donna Leon will publish a new mystery this year) and (if Charles Todd will be nominated for this year's Gold Dagger Award then Charles Todd will publish a new mystery this year)].
>
> It is not the case that Charles Todd will publish a new mystery this year and it is not the case that Giles Blunt will publish a new mystery this year.
>
> Louise Penny will publish a new mystery this year if and only if Donna Leon will publish a new mystery this year.
>
> ---
>
> (Donna Leon will publish a new mystery this year and Louise Penny will publish a new mystery this year) and (Donna Leon will be nominated for this year's Gold Dagger Award or Louise Penny will be nominated for this year's Gold Dagger Award).

Note that we have paraphrased the first premise as an extended disjunction. The occurrence of 'and' in the first premise *does not* signal that the premise should be paraphrased as a conjunction. Rather it is used to specify the members of the group, *one of whom* will be nominated. The second premise, 'Everyone who is nominated will publish a new mystery this year' is about all potential nominees, not just the four authors named in the first premise. But the second premise is relevant to the validity of the argument only as it

applies to the listed authors. Hence our paraphrase. We will use the following symbolization key:

P: Louise Penny will be nominated for this year's Gold Dagger Award.
B: Giles Blunt will be nominated for this year's Gold Dagger Award.
L: Donna Leon will be nominated for this year's Gold Dagger Award.
T: Charles Todd will be nominated for this year's Gold Dagger Award.
E: Louise Penny will publish a new mystery this year.
G: Giles Blunt will publish a new mystery this year.
D: Donna Leon will publish a new mystery this year.
C: Charles Todd will publish a new mystery this year.

$$(P \vee B) \vee (L \vee T)$$
$$[(P \supset E) \ \& \ (B \supset G)] \ \& \ [(L \supset D) \ \& \ (T \supset C)]$$
$$\sim C \ \& \sim G$$
$$E \equiv D$$
$$\therefore (D \ \& \ E) \ \& \ (L \vee P)$$

There are, of course, multiple ways in which the first and second paraphrases and symbolizations can be grouped. On the other hand, the second premise *cannot* correctly be regrouped and symbolized as '$[(P \ \& \ B) \ \& \ (L \ \& \ T)] \supset [(E \ \& \ G) \ \& \ (D \ \& \ C)]$'. This sentence would serve as a symbolization of the claim that if they *all* win, then they will *all* publish a new mystery this year. It will turn out that the argument with the correct symbolization given above is a valid argument of *SL*.

Here's another argument:

> If Henry is after pure suspense he will read a Jeffrey Deaver mystery, and if he wants wonderfully rich characters and doesn't care about subtle plots, he will read a Martha Grimes mystery. But if he wants richly developed characters and a subtle plot he will read a Louise Penny mystery. Although Henry doesn't care about character development or subtle plots, he does want pure suspense, so Henry will read a Jeffrey Deaver mystery.

Here is our paraphrase:

> (If Henry wants pure suspense then Henry will read a Jeffrey Deaver mystery) and [if (Henry wants well-developed characters and it is not the case that Henry cares about subtle plots) then Henry will read a Martha Grimes mystery].
>
> If (Henry wants well-developed characters and Henry cares about subtle plots) then Henry will read a Louise Penny mystery.

It is not the case that (Henry wants well-developed characters or Henry cares about subtle plots) and Henry wants pure suspense.

Henry will read a Jeffrey Deaver mystery.

In paraphrasing the original argument we have taken 'Henry is after pure suspense' and 'Henry does want pure suspense' to express the same claim, and we have used the latter in our paraphrases. Henry's view about character development is also variously expressed in the original argument. The first premise mentions 'wonderfully rich characters', the second mentions 'richly developed characters', and the third simply mentions character development. We have paraphrased all three as 'Henry wants well-developed characters'. Here are our symbolization key and our symbolization of the argument:

S: Henry wants pure suspense.
D: Henry will read a Jeffrey Deaver mystery.
W: Henry wants well-developed characters.
P: Henry cares about subtle plots.
M: Henry will read a Martha Grimes novel.
L: Henry will read a Louise Penny mystery.

$(S \supset D)\ \&\ [(W\ \&\ \sim P) \supset M]$

$(W\ \&\ P) \supset L$

$\sim (W \vee P)\ \&\ S$

D

We will show in the next few chapters that this is also a valid argument of *SL*.

Our final argument is

Christine will read a mystery if and only if there are no new science fiction novels in our library and there are no new science fiction movies available on Netflix. She will only read a mystery if it is set in the United States. Our library doesn't have any new science fiction novels and there are no new science fictions on Netflix. Donna Leon's mysteries are set in Venice, Louise Penny's mysteries and Giles Blunt's mysteries are set in Canada, and Charles Todd's mysteries are set in England. John Sandford's mysteries are set in the United States. Christine will therefore read a John Sandford mystery.

We paraphrase this as

Christine will read a mystery if and only if (it is not the case that there are new science fiction novels in our library and it is not the case that new science fiction movies are available on Netflix).

Christine will only read a mystery if it is set in the United States.

It is not the case that there are new science fiction novels in our library and it is not the case that new science fiction movies are available on Netflix.

(Donna Leon's mysteries are set in Venice and Louise Penny's mysteries are set in Canada) and (Giles Blunt's mysteries are set in Canada and Charles Todd's mysteries are set in England).

John Sandford's mysteries are set in the United States

Christine will read a John Sandford mystery.

We have used the second sentence of the original passage as its own paraphrase. This may seem strange, as the second sentence is an 'only if' claim and it may seem obvious that it should be paraphrased as

If Christine reads a mystery then it is set in the United States.

The problem is that in a truth-functional conditional what follows the 'then' must be an independent sentence that has a truth-value. 'It is set in the United States' is not such a sentence. Nor can it be turned into one by replacing 'it' with the name of a particular mystery, because 'it' does not refer to a particular mystery. If there were only a small number of mysteries, say two—*A Fine Red Rain* and *Rough Country*—then we could paraphrase the second premise as

(If Christine reads *A Fine Red Rain* then *A Fine Red Rain* is set in the United States) and (if Christine reads *Routh Country* then *Rough Country* is set in the United States).

But there are in fact an enormous number of mysteries, so it is not practical to paraphrase the second premise as a very long conjunction of material conditionals, each of which deals with one mystery. Rather, we will need the more powerful language *PL*, which is presented in Chapter 7, to adequately capture the structure of the second premise.

Here are our symbolization key and symbolizations:

C: Christine will read a mystery.
S: Our library has new science fiction novels.
N: New science fiction movies are available on Netflix.
U: Christine will only read a mystery if it is set in the United States.
D: Donna Leon's mysteries are set in Venice.
L: Louise Penny's mysteries are set in Canada.
G: Giles Blunt's mysteries are set in Canada.
T: Charles Todd's mysteries are set in England.

J: John Sandford's mysteries are set in the United States
R: Christine will read a John Sandford mystery.

$$\begin{array}{l} C \equiv (\sim S \ \& \sim N) \\ U \\ (D \ \& \ L) \ \& \ (G \ \& \ T) \\ J \\ \hline R \end{array}$$

This is not a valid argument of *SL*. Our paraphrase and symbolization do not bring out what is implicit in the original, that a mystery set in Venice is not set in the United States, that a mystery set in Canada is not set in the United States, and that a mystery set in England is not set in the United States. And even if these bits of geographic information were explicitly included in the argument and symbolization, the result would still not be a valid argument, because our symbolization of the second premise in *SL* does not show the relation between Christine's reading a mystery and the setting of that mystery, and also because John Sandford's mysteries are not the only mysteries set in the United States.

SUMMARY OF SOME COMMON CONNECTIVES

English Connective	*Paraphrase*	*Symbolization in SL*
not **p**	it is not the case that **p**	~ **P**
p and **q** **p** but **q** **p** however **q** **p** although **q** **p** nevertheless **q** **p** nonetheless **q** **p** moreover **q**	**p** and **q**	**P** & **Q**
p or **q** **p** unless **q**	**p** or **q**	**P** ∨ **Q**
p or **q** (exclusive sense)	**p** or **q** and it is not the case that (**p** and **q**)	(**P** ∨ **Q**) & ~ (**P** & **Q**)
if **p** then **q** **p** only if **q** **q** if **p** **q** provided that **p** **q** given **p**	if **p** then **q**	**P** ⊃ **Q**
p if and only if **q** **p** if but only if **q** **p** just in case **q**	**p** if and only if **q**	**P** ≡ **Q**

2.3**E** EXERCISES

1. Construct truth-functional paraphrases of the following sentences and symbolize those paraphrases in *SL*, using the following symbolization key:

P: The Red Sox improve their pitching.
G: The Red Sox have a good chance of winning the American League pennant.
Y: The Yankees will win the pennant.
F: The Red Sox falter.
T: The Twins win tonight.
M: The Mariners win tonight.
A: The Angels win tonight.
I: The Indians win tonight.
S: The Indians' starting pitcher can go the full nine innings.
N: The Angels move into first place.
H: The rain stops within an hour.
G: The game will be postponed.
R: The Royals are in the running for the pennant.

a. If the Red Sox improve their pitching they have a good chance of winning the American League pennant.
*b. The Yankees will win the pennant if the Red Sox falter and the Twins lose tonight.
c. If the Twins and the Mariners both lose tonight the Angels will move into first place.
*d. Assuming the rain stops within an hour the game will not be postponed.
e. The Indians will win tonight provided their starting pitcher can go the full nine innings.
*f. The Angels will move into first place only if the Twins and the Indians both lose tonight.
g. Assuming either the Twins or the Mariners win tonight, the Royals will be out of the running for the pennant.
*h. The Red Sox have a good chance of winning the pennant if and only if the Mariners and the Angels and the Twins all lose tonight.
i. The Royals are out of the race for the pennant and the Yankees will win the pennant if and only if the Twins win tonight and the Mariners and the Angels both lose tonight.
*j. The Red Sox have a good chance of winning the American League pennant but the Yankees will win the pennant if either the Red Sox falter or the Twins, the Angels, and the Mariners all win tonight.

2. Construct truth-functional paraphrases for the following, then provide a symbolization key and use it to symbolize your paraphrases in *SL*.
a. Either George or Emily will graduate with honors.
*b. Both George and Emily will graduate with honors or neither will.
c. At least one of George, Emily, Donna, and Fred will graduate with honors.
*d. If Donna graduates with honors so will Fred, and if Bob graduates with honors so will Emily.

e. Either they (Fred, George, Emily, and Donna) will all graduate with honors or none of them will.
*f. Either Fred won't graduate with honors or Emily and Donna both will.
g. Fred and George will graduate with honors if and only if Donna and Emily graduate with honors.
*h. Either George or Emily will graduate with honors but they won't both graduate with honors.
i. George won't graduate with honors but Fred will, and Donna will graduate with honors if and only if Emily does.
*j. If Emily and Donna don't both graduate with honors then neither George or Fred will graduate with honors.

3. Construct a truth-functional paraphrase of each of the following sentences, then provide a symbolization key and use it to symbolize your paraphrases.
a. If Felice vacations in Bermuda so will Clarence.
*b. Veronica will vacation in Bermuda only if both Clarence and Robert will also do so.
c. If either Felice or Veronica vacation in Bermuda they both will.
*d. Clarence will vacation in Bermuda only if Robert does and neither Felice nor Veronica do.
e. If Veronica vacations in Bermuda then Clarence will but Felice won't.
*f. Robert will vacation in Bermuda if and only if Clarence does, and Veronica will vacation in Bermuda if and only if Felice does.
g. Veronica will vacation in Bermuda if and only if Clarence doesn't, and Felice will vacation in Bermuda if and only if Robert does.
*h. Felice will vacation in Bermuda if and only if Veronica does and Robert doesn't, and Veronica will vacation in Bermuda if and only if Robert does and Clarence doesn't.

4. For each of the following, provide a truth-functional paraphrase and then symbolize your paraphrases in *SL*, indicating what sentence each of the sentence letters you use symbolizes.
a. *Casablanca*, *The Lion in Winter*, *Witness for the Prosecution*, *The Third Man*, and *Charade* will all be shown at this year's classical film festival.
*b. If Phil sees *Casablanca* he will enjoy Bogart's and Bergman's performances but he won't hear Bogart say "Play it again, Sam".
c. Phil will see *The Lion in Winter* only if Marion will and both of them will see *Charade*.
*d. Eric will see *The Lion in Winter* if and only if Betty does and if they see it they will love it.
e. If *Witness for the Prosecution* and *The Lion in Winter* are both screened at 8:00 pm, Marion and Phil will see *Witness for the Prosecution* and Eric and Betty will see *The Lion in Winter*.
*f. Phil will see *Charade* if and only if Audrey Hepburn and Cary Grant are both in it, and they are.
g. If it's the case that if Eric likes Katherine Hepburn then he'll see *The Lion in Winter*, then if Marion likes Eric she will see *The Lion in Winter*.
*h. If Claude Raines, Sydney Greenstreet, and Peter Lorre were in the movie Betty saw last night then she saw *Casablanca*.
i. Neither Betty nor Eric like James Coburn but they do both like Audrey Hepburn and if Audrey Hepburn is in *Charade* they will both see it (and she is).

5. Construct a truth-functional paraphrase of each of the following arguments, then provide a symbolization key and use it to symbolize your paraphrase of the argument in *SL*.

a. If Betty sees *Casablanca* and *The Third Man* then she won't see either *Witness for the Prosecution* or *The Lion in Winter*. She will see *Witness for the Prosecution* but she won't see *The Lion in Winter*. So either she won't see *Casablanca* or she won't see *The Third Man*.

*b. If Phil likes either Joseph Cotton or Orson Wells he will like *The Third Man*, if he sees it. If he likes either Peter O'Toole or Katharine Hepburn he'll like *The Lion in Winter*, if he sees it. He doesn't like either Joseph Cotton or Orson Wells, but he does like Katharine Hepburn. So if he sees *The Lion in Winter* he will like it.

c. Phil will see *The Third Man* if and only if he likes both Joseph Cotton and Orson Wells, and he will see *Witness for the Prosecution* if and only if he likes both Marlene Dietrich and Charles Laughton. He doesn't like either Joseph Cotton or Orson Wells, but he does like Marlene Dietrich and Charles Laughton. So Phil will see *Witness for the Prosecution*.

*d. Betty will see either *The Lion in Winter* or *Witness for the Prosecution*. Marion will see *Casablanca* and *Charade*. If Betty sees *Witness for the Prosecution* Eric won't, but he will see *Casablanca* if Marion does. Betty won't see *Witness for the Prosecution*, and she will see *The Lion in Winter* if and only if Phil does. So Phil will see *The Lion in Winter*.

6. Construct truth-functional paraphrases of each of the following passages. If a passage is an argument, present your paraphrase of the argument in standard form. Provide symbolization keys for your paraphrases of these passages and symbolize your paraphrases in *SL*.

a. Fred will go to New York only if he can get a first class air ticket and get tickets to a Yankees game. Fred will go to Chicago only if he can travel by train and get tickets to a White Sox game. He can't get a first class air ticket and he can't get tickets to a White Sox game, so he won't go to either New York or Chicago.

*b. If Lisa goes on vacation it will be to either Toronto, Montreal, Quebec, or Vancouver. If she goes to Toronto she will visit the University of Toronto; if she goes to Montreal she will eat great French food; if she goes to Quebec she will visit the Plains of Abraham; and if she goes to Vancouver she will go whale watching. She won't visit the University of Toronto; she won't eat great French food; and she won't go whale watching. So if she goes on vacation she will visit the Plains of Abraham.

c. Alice will go to Vienna if but only if Burt is willing to go with her and Burt speaks German. If Alice does go to Vienna she will take the Orient Express to Istanbul, unless Burt refuses to travel by train. Burt is willing to go with Alice to Vienna and he does speak German, but he won't travel by train. Hence if Alice goes to Vienna she will not take the Orient Express to Istanbul.

*d. Ben will go either to Duluth or to Kansas City. If it is the case that when Ben travels he travels by train, then if he travels to Duluth there is a train to Duluth and if he travels to Kansas City there is a train to Kansas City. Ben travels only by train and there is no train to Duluth. So Ben will travel to Kansas City and there is a train to Kansas City.

e. Charles Todd's mysteries are good mysteries. A good mystery has memorable characters, a plot that keeps the reader in suspense, and contains enough factual information to allow the reader to actually learn some interesting things; and Charles Todd's mysteries have all of these features.

2.4 NON-TRUTH-FUNCTIONAL USES OF CONNECTIVES

In Section 2.2 we introduced the notion of the truth-functional use of sentential connectives. As we explained there,

> A sentential connective, of a formal or a natural language, is used *truth-functionally* if and only if it is used to generate a compound sentence from one or more sentences in such a way that the truth-value of the generated compound is wholly determined by the truth-values of those one or more sentences from which the compound is generated, no matter what those truth-values may be.

The sentential connectives of *SL* are fully defined by their characteristic truth-tables and therefore have only truth-functional uses. As we have seen, we can often paraphrase English compounds as truth-functional compounds without weakening or distorting the content of the sentences being paraphrased. Paraphrases that are negations, conjunctions, and disjunctions often capture all or almost all of the content of the sentences being paraphrased. A sentence such as 'Aristotle and Alexander were both Greek' can be paraphrased as

> Aristotle was Greek and Alexander was Greek

with no loss of content. The English sentence says neither more nor less than the truth-functional paraphrase. In contrast, English conditionals frequently express links or connections between their antecedents and consequents that are lost when we paraphrase them as material conditionals. For example, consider the sentence

> Assuming the rain stops within an hour, the game will not be postponed.

This sentence is appropriately paraphrased as

> If the rain stops within an hour then it is not the case that the game will be postponed

and the paraphrase can be symbolized as 'S ⊃ ~ P'. But it can be argued that our paraphrase does not capture all of the content of the sentence it paraphrases. In the original there is at least the suggestion that the rain's stopping, if it does, will be the reason the game will not be postponed and that the rain's not stopping, if it doesn't, will be the reason for the game's being postponed. Often this kind of loss of content will not matter for the purposes at hand, for example, determining the validity of an argument. But when an English conditional is based on a scientific law, paraphrasing that conditional as a material conditional can be problematic. An example is

> If this rod is made of metal, then it will expand when heated.

A simple law of physics lies behind this claim: all metals expand when heated, and the conditional is in effect claiming that if the rod in question is made of metal then heating it will *cause* it to expand. A paraphrase of this causal claim as a material conditional does not capture this causal connection. The failure to capture such causal connections may or may not be acceptable, depending on the context and on what questions we are asking about the sentence or set of sentences being paraphrased.

When we use the tools that we develop in subsequent chapters to analyze sentences and sets of sentences of *SL,* the results that we obtain will apply directly to the sentences and sets of sentences of *SL* we are analyzing and to the *truth-functional paraphrases* those sentences symbolize. The results will also apply to the English sentences from which the paraphrases are obtained to the extent, and only to the extent, that the paraphrases capture the content of the original sentences. Here's an example of how things can go wrong if we ignore this caveat. Suppose some benighted person incorrectly believes that metals contract when they are heated. Such a person might make the following claim about a rod whose composition is unknown: 'If this rod is made of metal, then this rod will contract when heated'. Taken as a causal claim, this is clearly false. Metals expand when heated; they don't contract. Here is a truth-functional paraphrase of the claim, and a symbolization of that paraphrase in *SL*:

If this rod is made of metal then this rod will contract when heated.

$M \supset C$

We have used 'M' to symbolize 'This rod is made of metal' and 'C' to symbolize 'This rod will contract when heated'. Now suppose that the rod is in fact plastic, not metal. Then the antecedent of '$M \supset C$' and of the paraphrase it symbolizes are both false, making the material conditional of *SL* and our paraphrase *both true*, even though the English sentence we paraphrased and symbolized is clearly false. In this case, where the alleged causal connection between antecedent and consequent is crucial to the claim being made, it is wise to treat the original sentence as a non-truth-functional compound and symbolize it as an atomic sentence of *SL*.

Of course, there are many English conditionals that can be appropriately paraphrased as material conditionals with no loss of content. We are all familiar with, and probably have made, claims of the sort

If such-and-such then I'm a such-and-such,

where the first 'such-and-such' is replaced by some very improbable claim and the second with a known falsehood. For example, if someone tells us that Jones, whom we know to be barely literate, is going to write the great American novel, one of us might comment 'If Jones can write the great American novel I can leap tall buildings in a single bound'. We all know the consequent of this conditional is false. By asserting the conditional, knowing the consequent

is false, the speaker is implicitly asserting that the antecedent is also false, and the antecedent's being false makes the conditional true.

There are other English conditionals called 'subjective conditionals' that cannot adequately be paraphrased as material conditionals. Here are two examples:

> If Harry were to win the lottery, he would give all the proceeds to charity.

and

> If Hitler had not invaded Russia, he would have defeated Great Britain and won the Second World War.

We might be tempted to paraphrase the claim about Harry as a material conditional, that is, as

> If Harry wins the lottery then Harry will give all the proceeds to charity.

A material conditional is true when its antecedent is false. Now suppose that the antecedent of our paraphrase is false; Harry does not win the lottery (as will almost certainly be the case). Our paraphrase is then true. But Harry's failure to win hardly makes the original subjunctive claim true. Suppose we know that Harry is by nature not a generous person, and we know that he has never given a dime to charity in his life. Moreover he has frequently railed against those who do give to charity. If we know all of this, then we will reject the subjunctive conditional. Harry is just not the sort of person who gives money to charity. So we will conclude that it is not the case that if Harry were to win the lottery he would give all the proceeds to charity. That he did not win the lottery is irrelevant to this reasoning.

Our second example of a subjunctive conditional also cannot be paraphrased as a material conditional. All historians know that Hitler *did* invade Russia and *did not* win the Second World War. But they do not take those facts to determine the truth-value of the above subjunctive conditional concerning Hitler. In fact, historians continue to disagree about the truth-value of that subjunctive conditional.

English has a large number of non-truth-functional connectives. 'I believe that . . .' is one. Attach 'I believe that' to any sentence of English that has a truth-value and the result is a sentence of English that has a truth-value. But the result is not a truth-functional compound. Given any sentence of the form 'I believe that **p**', the truth-value of that sentence is not determined by the truth-value of **p**. No matter how obviously false **p** may be, I might still believe it, and no matter how obviously true it may be, I may still not believe it. 'It is alleged that' is not a truth-functional connective for similar reasons. Attaching 'It is alleged that' to any sentence with a truth-value yields a sentence with a truth-value, but the truth-value of the sentence to which 'It is alleged that' is attached does not determine the truth-value of the resulting sentence. Suppose

that it is false that Senator Bigmouth took a bribe when he was mayor of Littletown. 'It is alleged that Senator Bigmouth took a bribe when he was mayor of Littletown' may nonetheless be true. All sorts of false things are alleged. And all sorts of true things are not alleged. Similar to the case of 'I believe that', the truth-value of 'Senator Bigmouth took a bribe when he was mayor of Littletown' and the truth-value of 'It is alleged that Senator Bigmouth took a bribe when he was mayor of Littletown' are logically independent. They can both be true, they can both be false, the first can be true while the second is false, and the second can be true while the first is false. Other non-truth-functional unary connectives of English include

> It is probable that
> Necessarily
> It would not be surprising if
> We are convinced that
> We hope that . . .
> I know that . . .

The truth-value of a sentence formed by attaching any one of these expressions other than 'I know that . . .' to a sentence **p** that has a truth-value is logically independent of the truth-value of **p**. The generated compound and **p** may both be true, they may both be false, **p** may be true and the compound false, and **p** may be false and the compound true. 'I know that' is different in that if this connective is attached to a false sentence then the compound that is generated is also false. But the compound may be either true or false when the sentence to which 'I know that' is attached is true.

Though connectives of the sort we have been discussing are all non-truth-functional, some of the compound sentences they generate can be paraphrased as truth-functional compounds. An example is

> Commentators believe the Republicans will retain control of the House and the Democrats will retain control of the Senate.

This claim can reasonably be paraphrased as the truth-functionally compound sentence

> Commentators believe the Republicans will retain control of the House and commentators believe the Democrats will retain control of the Senate.

This paraphrase is a truth-functional compound, a conjunction, each of whose conjuncts is a non-truth-functional compound. But we must be careful here.

> Commentators believe the Republicans will gain control of the Senate or of the House

is not reasonably paraphrased as

> Commentators believe the Republicans will gain control of the Senate or commentators believe the Republicans will gain control of the House.

Commentators may believe the Republicans will gain control of at least one chamber, the House or the Senate, but have no opinion about which chamber it will be. As another example, consider the flipping of a fair coin. We all believe that the coin will either come up heads or come up tails. But this is not equivalent to

> We all believe the coin will come up heads or we all believe the coin will come up tails.

There are also binary connectives of English that are never used truth-functionally. One is 'because'. Consider the sentence

> Henry will not read *Drawing Conclusions* because it is set in Venice.

The truth-value of this compound is *not* wholly determined by the truth-values of its immediate components. While the falsity of either 'Henry will not read *Drawing Conclusions*' or '*Drawing Conclusions* is set in Venice' is sufficient for the falsity of the compound, the truth of these components does not determine the truth-value of the compound. It is true that *Drawing Conclusions* is set in Venice and it may be true that Harry will not read it, but the reason he will not read it may have nothing to do with its being set in Venice. Perhaps the reason is that Henry's library doesn't have a copy and Henry is too cheap to buy a copy.

The connective 'before' is also a non-truth-functional connective. Consider the sentences

> Jimmy Carter was elected president before Ronald Reagan was elected president

and

> Ronald Reagan was elected president before Jimmy Carter was elected president.

The component sentences 'Ronald Reagan was elected president' and 'Jimmy Carter was elected president' are both true, but the first compound sentence is true while the second is false. Hence the truth-values of the components do not, in every case, determine the truth-value of the compound sentences and 'before' is, therefore, not a truth-functional connective. The same is true of 'after'. More generally, given that either **p** is false or **q** is false, we may conclude that both

> **p** before **q**

and

p after **q**

are false, but we cannot conclude anything about the truth or falsity of either of these claims *given only that* both **p** and **q** are true.

The safest policy for paraphrasing non-truth-functionally compound sentences is to let them be their own paraphrases and symbolize them as atomic sentences of *SL*. However, there are cases in which we can construct truth-functionally compound sentences of English that capture some of the content of non-truth-functionally compound English sentences, and it is sometimes useful to do so. First, a definition. We have noted several times that a paraphrase or proposed paraphrase fails to capture all of the content of the sentence being paraphrased. In such cases the paraphrase is "weaker" than the original. What it means to say that one sentence is weaker than, or stronger than, another is not entirely clear in ordinary English, and we therefore provide a **stipulative** definition of these terms that states what logicians mean when we use the words 'weaker' and 'stronger' to describe relationships between sentences:

> A sentence **p** of a natural or formal language is *stronger* than a sentence **q** of a natural or formal language (and **q** is *weaker* than **p**) if and only if **q** follows from **p** but **p** does not follow from **q**.[8]

For example, 'Aristotle was Greek and Alexander was Greek' is stronger than 'Aristotle was Greek', because the latter follows from the former, but not vice versa. For the same reason, 'Aristotle was Greek' is weaker than 'Aristotle was Greek and Alexander was Greek'.

Here is an argument all of whose premises are conditionals:

> If the rain continues the game will be postponed. If the game is postponed Bronson won't have to pitch today. If Bronson won't have to pitch today he will be ready to pitch tomorrow. If Bronson is ready to pitch tomorrow his team will win tomorrow. The rain will continue. So Bronson's team will win tomorrow.

This argument, which is valid, connects a series of envisioned events, the first being its continuing to rain today and the last being Bronson's team's winning tomorrow. Arguably, the envisioned events are presented as being connected in more than a truth-functional way. For example, it is at least implicit that if the rain continues, its doing so will cause the game to be postponed, and that if the game is postponed, the postponement will be responsible for Bronson's not having to pitch today, and that if Bronson's not having to pitch today will ensure that he'll be ready to pitch tomorrow, and that his being ready to pitch

[8]This is a stipulative definition because it does not fully accord with all the ways 'stronger than' and 'weaker than' are used in ordinary English. For example, most of us would take 'There is a cougar in the yard' to be a stronger claim than is 'I think there is a cougar in the yard'. But 'There is a cougar in the yard' does not follow from 'I think there is a cougar in the yard'. Such uses of 'stronger' in ordinary English perhaps convey that one claim conveys more reliable, or more important information than does another.

tomorrow will lead to and be responsible for his team's victory. These implicit causal relationships are lost in the following truth-functional paraphrase of the argument:

> If the rain will continue then the game will be postponed.
>
> If the game will be postponed then it is not the case that Bronson will have to pitch today.
>
> If it is not the case that Bronson will have to pitch today then Bronson will be ready to pitch tomorrow.
>
> If Bronson will be ready to pitch tomorrow then Bronson's team will win tomorrow.
>
> The rain will continue.
>
> ---
>
> Bronson's team will win tomorrow.

But the conclusion of the paraphrase is identical to the conclusion of the original, and it does follow from the paraphrased premises. And since each of the paraphrased premises is weaker than (and therefore follows from) the premise it paraphrases, the conclusion also follows from the original premises, and so we may conclude that the original argument is valid.

In paraphrasing the original argument we weakened each premise, by replacing a causal conditional with a material conditional. Causal conditionals are stronger than material conditionals. For example, it follows from the causal conditional

> If this rod is made of metal it will expand when heated,

that either the rod is not made of metal or it will expand when heated, which is all that the material conditional

> If this rod is made of metal then it will expand when heated

comes to, but the causal conditional does not follow from the material conditional.

We have seen that if we weaken the premises of an argument in the paraphrase process but do not weaken the conclusion, and the paraphrase and its symbolization turn out to be valid, we may safely conclude that the original argument is also valid. *But* if the paraphrased argument and its symbolization turn out to be *invalid* we *cannot* conclude that the original argument is invalid. That a conclusion does not follow from one set of premises (our paraphrases of the original premises) does not show that it does not follow from a stronger set of premises (the premises of the original argument). Hence while we can sometimes use a paraphrase whose premises are weaker than the premises of the original argument to show that the original argument is valid, we can never use such paraphrases to show the argument being paraphrased is invalid.

Here is a simple example that illustrates this point.

> Aristotle and Plato were both Greek. If Aristotle was Greek he wasn't Roman, and if Plato was Greek he wasn't Roman. So neither Aristotle nor Plato was Roman.

This is obviously a valid argument, as are its truth-functional paraphrase and symbolization:

> Aristotle was Greek and Plato was Greek.
>
> (If Aristotle was Greek then it is not the case that Aristotle was Roman) and (if Plato was Greek then it is not the case that Plato was Roman).
>
> ---
>
> It is not the case that Aristotle was Roman and it is not the case that Plato was Roman.

Using 'A' to symbolize 'Aristotle was Greek', 'P' to symbolize 'Plato was Greek', 'R' to symbolize 'Aristotle was Roman', and 'L' to symbolize 'Plato was Roman', we can symbolize our paraphrased argument as

$$\begin{array}{l} A \,\&\, P \\ \underline{(A \supset \sim R) \,\&\, (P \supset \sim L)} \\ \sim R \,\&\, \sim L \end{array}$$

This symbolic argument is valid. Now suppose we weaken our paraphrase of the first premise by replacing it with 'Aristotle was Greek', a sentence that is clearly weaker than (because if follows from) the first premise of the original argument. The symbolization of our revised paraphrase will be

$$\begin{array}{l} A \\ \underline{(A \supset \sim R) \,\&\, (P \supset \sim O)} \\ \sim R \,\&\, \sim O \end{array}$$

This symbolic argument is invalid, as is the truth-functional paraphrase it symbolizes. But the original argument is valid. Again, showing that a paraphrased argument is invalid where the premises of the paraphrase are weaker than the premises of the original argument *does not show* that the original argument is invalid.

Here is a more interesting case in which weakening the premises of an argument in paraphrasing it is both appropriate and useful. Suppose that a detective reasons as follows:

> If Williams is the murderer he had to be in Philadelphia on the 5th. Because we know that Williams was in Rome on the 5th, we know that he was not in Philadelphia on the 5th. So Williams isn't the murderer.

This seems to be a valid piece of reasoning, and if we use 'Williams was in Rome on the 5th and it is not the case that Williams was in Philadelphia on the night of the 5th, for the second premise, it seems that we can capture the structure of that reasoning:

> If Williams is the murderer then Williams was in Philadelphia on the 5th.
>
> Williams was in Rome on the 5th and it is not the case that Williams was in Philadelphia on the 5th.
>
> ---
>
> It is not the case that Williams is the murderer.

Our paraphrase of the first premise is weaker than the premise it paraphrases: we have replaced 'had to be in Philadelphia' with 'was in Philadelphia'. Our paraphrase of the second premise is weaker than the original and follows from it. Both **p** and **q** follow from causal claims of the sort

> Because **p**, **q**

and **p** follows from

> We know that **p**,

though not vice versa. So

> We know that Williams was in Rome on the 5th and we know that Williams was not in Philadelphia on the 5th

follows from the original second premise. And our paraphrase follows from this conjunction of two knowledge claims. Using obvious choices of sentence letters, we can symbolize the paraphrase as

$$\begin{array}{l} W \supset P \\ R \,\&\, \sim P \\ \hline \sim W \end{array}$$

This is a valid argument of *SL*.

Similarly, although English subjunctive conditionals are not truth-functional compounds, it is sometimes possible and appropriate to use material conditionals as paraphrases of subjunctive conditionals. Suppose that a doctor who is testifying at an inquest argues as follows:

> Had the deceased died of strychnine poisoning, there would have been traces of that poison in the body. The autopsy would have found those traces had they been there. The autopsy did not reveal any traces of strychnine. Hence the deceased did not die of strychnine poisoning.

Replacing the subjunctive conditionals with material conditionals we obtain the following paraphrase of this argument:

> If the deceased died of strychnine poisoning then there were traces of strychnine in the body.
>
> If there were traces of strychnine in the body then the autopsy found traces of strychnine in the body.
>
> It is not the case that the autopsy found traces of strychnine in the body.
>
> ---
>
> It is not the case that the deceased died of strychnine poisoning.

Using obvious choices of sentence letters, we can symbolize this argument in *SL* as

$$\begin{array}{l} S \supset T \\ T \supset F \\ \sim F \\ \hline \sim S \end{array}$$

This is a valid argument of *SL*, as we will be able to show in subsequent chapters.

We have discussed when it is appropriate to weaken the premises of an argument when paraphrasing them. Recall that entailment is a notion that nearly parallels that of validity (the difference being that some sentences are entailed by the empty set but there are no arguments with no premises). Accordingly, it is sometimes appropriate to weaken the members of a set when trying to determine whether that set entails a given sentence.

Care must also be taken when weakening or strengthening a sentence in the paraphrase process when we are concerned with determining the consistency of a set of sentences, the equivalence of sentences, or the logical status of a sentence (logically true, logically false, or logically indeterminate). For example, if we are interested in whether a set of sentences of English is consistent and in the paraphrase process we weaken one of the members of the set, then showing that the set consisting of the paraphrased sentences is consistent will not establish that the original set of sentences is consistent. And if in the paraphrase process we strengthen one of the members of the set, then showing that the set consisting of the paraphrased sentences is inconsistent will not show that the original set is inconsistent, though if the set of paraphrased sentences turns out to be consistent, so is the original set. Similarly, if we are interested in whether two sentences are equivalent and weaken or strengthen either or both of the sentences in the paraphrase process, then showing that the paraphrases are, or are not, equivalent will not, in general, constitute showing that the original sentences are or are not equivalent.

What can be said, and we have so said before, is that the results we obtain by using the techniques developed in subsequent chapters to test for validity, entailment, consistency, and the other core semantical concepts apply directly only to the paraphrases and symbolizations we have constructed.

2.4**E** EXERCISES

1. Paraphrase and symbolize each of the following sentences that can reasonably be paraphrased as a truth-functional compound. If a sentence cannot be so paraphrased, explain why this is so. Provide a symbolization key when it is not obvious what sentence your sentence letters are symbolizing.

a. It's likely that either the Boston Red Sox or the New York Yankees will win the World Series this year.

*b. Either Rocky or George knows what time the concert starts.

c. Marcie thinks that either Helen or Stephanie will be elected.

*d. Tamara won't be visiting tonight because she is working late.

e. Although Tamara won't stop by, she has promised to phone early in the evening.

*f. If the victim had been strangled there would have been marks on his throat, and there weren't.

g. John believes that our manuscript has been either lost or stolen.

*h. John believes that our manuscript has been stolen, and Howard believes that it has been lost.

i. The defendant confessed only after much of her testimony was discredited.

*j. It is possible that the Twins will win tonight and possible that the Red Sox will win tonight, but it is not likely that they will both win tonight.

2. Construct truth-functional paraphrases of the premises and conclusions of the following arguments, provide symbolization keys, and symbolize your paraphrases in *SL*.

a. The murder was committed by the maid only if she believed her life was in danger. Had the butler done it, it would have been done silently and the body would not have been mutilated. As a matter of fact it was done silently; however, the maid's life was not in danger. The butler did it if and only if the maid failed to do it. Hence the maid did it.

*b. If this piece of metal is gold, then it has atomic number 79. Nordvik believes this piece of metal is gold. Therefore Nordvik believes this piece of metal has atomic number 79.

c. If Charles Babbage had had the theory of the modern computer and had had modern electronic parts, then the modern computer would have been developed before the beginning of the twentieth century. In fact, although he lived in the early nineteenth century, Babbage had the theory of the modern computer. But he did not have access to modern electronic parts, and he was forced to construct his computers out of mechanical gears and levers. Therefore, if Charles Babbage had had modern electronic parts available to him, the modern computer would have been developed before the beginning of the twentieth century.

GLOSSARY

TRUTH-FUNCTIONAL USE OF A CONNECTIVE: A sentential connective, of a formal or a natural language, is used *truth-functionally* if and only if it is used to generate a compound sentence from one or more sentences in such a way that the truth-value of the generated compound is wholly determined by the truth-values of those one or more sentences from which the compound is generated, no matter what those truth-values may be.

Chapter 3

SENTENTIAL LOGIC: SEMANTICS

Section 3.1 introduces the foundations of the truth-functional semantics for *SL*: truth-value assignments and the truth-tables that record them. Sections 3.2 through 3.5 present the truth-functional versions of the core logical concepts: truth-functional truth, falsehood, and indeterminacy; truth-functional equivalence; truth-functional consistency; and truth-functional entailment and validity. Section 3.6 explicates all of the truth-functional concepts in terms of truth-functional consistency to provide a framework for truth-trees, which are presented in Chapter 4.

3.1 TRUTH-VALUE ASSIGNMENTS AND TRUTH-TABLES FOR SENTENCES

In Chapter 1, we introduced logical concepts such as logical truth and logical validity. In this chapter we shall develop formal tests for truth-functional versions of the core logical concepts introduced in Chapter 1. Specifically, we will develop tests for truth-functional truth, falsity, and indeterminacy; truth-functional equivalence; truth-functional consistency; and truth-functional entailment and validity. All these concepts fall within the realm of *semantics:* They concern the truth-values and truth-conditions of sentences and sets of sentences of *SL*. Before defining these truth-functional concepts, our first task

is to specify how truth-values and truth-conditions for sentences of *SL* are determined.

Every sentence of *SL* can be built up from its atomic components in accordance with the definition of sentences. Similarly the truth-value of a sentence of *SL* is completely determined by the truth-values of its atomic components in accordance with the characteristic truth-tables for the connectives. We repeat the characteristic truth-tables here:

P	~ P
T	F
F	T

P	Q	P & Q
T	T	T
T	F	F
F	T	F
F	F	F

P	Q	P ∨ Q
T	T	T
T	F	T
F	T	T
F	F	F

P	Q	P ⊃ Q
T	T	T
T	F	F
F	T	T
F	F	T

P	Q	P ≡ Q
T	T	T
T	F	F
F	T	F
F	F	T

These tables tell us how to determine the truth-value of a truth-functionally compound sentence given the truth-values of its immediate sentential components.

The truth-values of atomic sentences are fixed by **truth-value assignments:**

> A truth-value assignment is an assignment of truth-values (**T**s and **F**s) to the atomic sentences of *SL*.

The concept of a truth-value assignment is the basic semantic concept of *SL*. Intuitively, each truth-value assignment gives us a description of a way the world *might* be, for in each we consider a combination of truth-values that atomic sentences might have. We assume that the atomic sentences of *SL* are truth-functionally independent—that is, that the truth-value assigned to one does not affect the truth-value assigned to any other. For generality we stipulate that a truth-value assignment must assign a truth-value to every atomic sentence, so that a truth-value assignment gives a *complete* description of a way the world might be. The truth-values of truth-functionally compound sentences of *SL* are uniquely and completely determined by the truth-values of their atomic components, so it follows that every truth-functionally compound sentence also has a truth-value, either **T** or **F**, on each truth-value assignment.

A truth-table for a sentence of *SL* is used to record its truth-value on each truth-value assignment. Because a truth-value assignment assigns truth-values to an infinite number of atomic sentences (*SL* has infinitely many atomic sentences), we cannot list an entire truth-value assignment in a truth-table.

Instead, we list all the possible combinations of truth-values that the sentence's atomic components may have on a truth-value assignment. As an example, here is the beginning of a truth-table for '~ B ⊃ C':

B	C	~ B ⊃ C
T	**T**	
T	**F**	
F	**T**	
F	**F**	

The atomic components of the sentence are 'B' and 'C', and the four rows of the table display the four combinations of truth-values that these components might have. Each row represents an infinite number of truth-value assignments, namely, all the truth-value assignments that assign to 'B' and 'C' the values indicated in that row. Since the truth-value of '~ B ⊃ C' on a truth-value assignment depends only on the truth-values that its atomic components have on that assignment, the four combinations that we have displayed will allow us to determine the truth-value of '~ B ⊃ C' on any truth-value assignment.

The first step in constructing a truth-table for a sentence **P** of *SL* is to determine the number of different combinations of truth-values that its atomic components can have. There is a simple way to do this. Consider first the case in which **P** has one atomic component. There are two different truth-values that the single atomic component may have: **T** and **F**. Now suppose that **P** is a sentence with two atomic components. In this case there are four combinations of truth-values that the atomic components of **P** might have, as we have seen in the case of '~ B ⊃ C' above.

If **P** has three atomic components, there are eight combinations of truth-values that its atomic components might have. To see this, suppose we want to expand this truth-table to record truth-values for a modified sentence that has three atomic components:

A	B	C	(~ B ⊃ C) & (A ≡ B)
	T	**T**	
	T	**F**	
	F	**T**	
	F	**F**	

What truth-values do we enter in the first row under 'A'? The combination of truth-values that would be displayed by entering **T** there is different from the combination that would be displayed by entering **F**. And the same holds for each row. So we need to list each of the four combinations of truth-values that 'B' and 'C' may have *twice* in order to represent all combinations of truth-values for the three atomic components.

A	B	C	(~ B ⊃ C) & (A ≡ B)
T	**T**	**T**	
T	**T**	**F**	
T	**F**	**T**	
T	**F**	**F**	
F	**T**	**T**	
F	**T**	**F**	
F	**F**	**T**	
F	**F**	**F**	

Extending this reasoning, we find that every time we add a new atomic sentence to the list the number of rows in the truth-table doubles. If **P** has **n** distinct atomic components, there are $2^{\mathbf{n}}$ different combinations of truth-values for its atomic components.[1]

In constructing a truth-table, we adopt a systematic method of listing the combinations of truth-values that the atomic components of a sentence **P** might have. We first list the atomic components of **P** to the left of the vertical line at the top of the truth-table, in alphabetical order.[2]

Under the first sentence letter listed, we write a column of $2^{\mathbf{n}}$ entries, the first half of which are **T**s and the second half of which are **F**s. In the second column the number of **T**s and **F**s being alternated is half the number alternated in the first column. In the column under the third sentence letter listed, the number of **T**s and **F**s being alternated will again be half the number in the second column. We repeat this process until a column has been entered under each sentence letter to the left of the vertical line. The column under the last sentence letter in this list will then consist of single **T**s alternating with single **F**s. For a truth-table with **n** distinct sentence letters, the first column consists of $2^{\mathbf{n}-1}$ **T**s alternating with $2^{\mathbf{n}-1}$ **F**s, the second of $2^{\mathbf{n}-2}$ **T**s alternating with $2^{\mathbf{n}-2}$ **F**s, and in general the ***i***th column consists of $2^{\mathbf{n}-i}$ **T**s alternating with $2^{\mathbf{n}-i}$ **F**s.

Now we can complete the rest of the truth-table for '(~ B ⊃ C) & (A ≡ B)'. We first repeat under 'A', 'B', and 'C', wherever these occur, the columns we have already entered under these letters to the left of the vertical line:

A	B	C	(~ B	⊃	C)	&	(A	≡	B)
T	**T**	**T**	**T**		**T**		**T**		**T**
T	**T**	**F**	**T**		**F**		**T**		**T**
T	**F**	**T**	**F**		**T**		**T**		**F**
T	**F**	**F**	**F**		**F**		**T**		**F**
F	**T**	**T**	**T**		**T**		**F**		**T**
F	**T**	**F**	**T**		**F**		**F**		**T**
F	**F**	**T**	**F**		**T**		**F**		**F**
F	**F**	**F**	**F**		**F**		**F**		**F**

[1] $2^{\mathbf{n}}$ is 2 if **n** = 1, 2 × 2 if **n** = 2, 2 × 2 × 2 if **n** = 3, and so on. 2^0 is 1.

[2] This is an extended sense of 'alphabetical order' since some sentence letters have subscripts. In this order all the nonsubscripted letters appear first, then all letters subscripted with '1', then all letters subscripted with '2', and so on.

Next we may enter the column for the component '~ B' under its main connective, the tilde. In each row in which 'B' has the truth-value **T**, '~ B' has the truth-value **F**, and in each row in which 'B' has the truth-value **F**, '~ B' has the truth-value **T**:

A	B	C	(~ B	⊃	C)	&	(A	≡	B)
T	**T**	**T**	**F**	**T**	**T**		**T**		**T**
T	**T**	**F**	**F**	**T**	**F**		**T**		**T**
T	**F**	**T**	**T**	**F**	**T**		**T**		**F**
T	**F**	**F**	**T**	**F**	**F**		**T**		**F**
F	**T**	**T**	**F**	**T**	**T**		**F**		**T**
F	**T**	**F**	**F**	**T**	**F**		**F**		**T**
F	**F**	**T**	**T**	**F**	**T**		**F**		**F**
F	**F**	**F**	**T**	**F**	**F**		**F**		**F**

The column for '~ B ⊃ C' is entered under the horseshoe. A material biconditional has the truth-value **F** when its antecedent has the truth-value **T** and its consequent has the truth-value **F**, and it has the truth-value **T** in all other cases:

A	B	C	(~ B	⊃	C)	&	(A	≡	B)
T	**T**	**T**	**F**	**T**	**T**		**T**		**T**
T	**T**	**F**	**F**	**T**	**F**		**T**		**T**
T	**F**	**T**	**T**	**F**	**T**		**T**		**F**
T	**F**	**F**	**T**	**F**	**F**		**T**		**F**
F	**T**	**T**	**F**	**T**	**T**		**F**		**T**
F	**T**	**F**	**F**	**T**	**F**		**F**		**T**
F	**F**	**T**	**T**	**F**	**T**		**F**		**F**
F	**F**	**F**	**T**	**F**	**F**		**F**		**F**

We now enter the column for 'A ≡ B' in accordance with the characteristic truth-table for '≡':

A	B	C	(~ B	⊃	C)	&	(A	≡	B)
T	**T**	**T**	**F**	**T**	**T**		**T**	**T**	**T**
T	**T**	**F**	**F**	**T**	**F**		**T**	**T**	**T**
T	**F**	**T**	**T**	**F**	**T**		**T**	**F**	**F**
T	**F**	**F**	**T**	**F**	**F**		**T**	**F**	**F**
F	**T**	**T**	**F**	**T**	**T**		**F**	**F**	**T**
F	**T**	**F**	**F**	**T**	**F**		**F**	**F**	**T**
F	**F**	**T**	**T**	**F**	**T**		**F**	**T**	**F**
F	**F**	**F**	**T**	**F**	**F**		**F**	**T**	**F**

Remember that a material biconditional has the truth-value **T** on all truth-value assignments on which its immediate components have the same truth-value,

and the truth-value **F** on all other truth-value assignments. Finally we enter the column for '(~ B ⊃ C) & (A ≡ B)' under its main connective, the ampersand:

						↓			
A	B	C	(~	B	⊃	C)	&	(A	≡	B)
T	**T**	**T**	**F**	**T**	**T**	**T**	**T**	**T**	**T**	**T**
T	**T**	**F**	**F**	**T**	**T**	**F**	**T**	**T**	**T**	**T**
T	**F**	**T**	**T**	**F**	**T**	**T**	**F**	**T**	**F**	**F**
T	**F**	**F**	**T**	**F**	**F**	**F**	**F**	**T**	**F**	**F**
F	**T**	**T**	**F**	**T**	**T**	**T**	**F**	**F**	**F**	**T**
F	**T**	**F**	**F**	**T**	**T**	**F**	**F**	**F**	**F**	**T**
F	**F**	**T**	**T**	**F**	**T**	**T**	**T**	**F**	**T**	**F**
F	**F**	**F**	**T**	**F**	**F**	**F**	**F**	**F**	**T**	**F**

We use arrows to indicate the main connective of the sentence. Each row of the truth-table displays, underneath the arrow, the truth-value that the sentence has on every truth-value assignment that assigns the truth-values displayed to the left of the vertical line to its atomic components.

Here is the truth-table for the sentence '[A ≡ (B ≡ A)] ∨ ~ C':

								↓		
A	B	C	[A	≡	(B	≡	A)]	∨	~	C
T	**T**	**T**	**T**	**T**	**T**	**T**	**T**	**T**	**F**	**T**
T	**T**	**F**	**T**	**T**	**T**	**T**	**T**	**T**	**T**	**F**
T	**F**	**T**	**T**	**F**	**F**	**F**	**T**	**F**	**F**	**T**
T	**F**	**F**	**T**	**F**	**F**	**F**	**T**	**T**	**T**	**F**
F	**T**	**T**	**F**	**T**	**T**	**F**	**F**	**T**	**F**	**T**
F	**T**	**F**	**F**	**T**	**T**	**F**	**F**	**T**	**T**	**F**
F	**F**	**T**	**F**	**F**	**F**	**T**	**F**	**F**	**F**	**T**
F	**F**	**F**	**F**	**F**	**F**	**T**	**F**	**T**	**T**	**F**

The column for '~ C' is constructed in accordance with the characteristic truth-table for the tilde. '~ C' has the truth-value **T** on all and only those truth-value assignments on which 'C' has the truth-value **F**. The column for '~ C' appears directly underneath the tilde. '(B ≡ A)' has the truth-value **T** for the combinations of truth-values displayed in the first two and last two rows of the truth-table, because 'B' and 'A' have the same truth-value in those rows, and the truth-value **F** for the other combinations.

Similarly '[A ≡ (B ≡ A)]' has the truth-value **T** on exactly those truth-value assignments on which 'A' and '(B ≡ A)' have the same truth-value. The column for '[A ≡ (B ≡ A)]' appears directly underneath its main connective, which is the first occurrence of the triple bar. '[A ≡ (B ≡ A)] ∨ ~ C' has the truth-value **T** on every truth-value assignment on which either '[A ≡ (B ≡ A)]' or '~ C' has the truth-value **T** and the truth-value **F** when both of its immediate

components do. The truth-value of the entire sentence for each combination of truth-values assigned to its atomic components is written in the column directly underneath the wedge, the sentence's main connective.

Here is the truth-table for the sentence '~ [(U ∨ (W ⊃ ~ U)) ≡ W]':

			↓								
U	W	~	[(U	∨	(W	⊃	~	U))	≡	W]
T	**T**	**F**	**T**	**T**	**T**	**F**	**F**	**T**	**T**	**T**
T	**F**	**T**	**T**	**T**	**F**	**T**	**F**	**T**	**F**	**F**
F	**T**	**F**	**F**	**T**	**T**	**T**	**T**	**F**	**T**	**T**
F	**F**	**T**	**F**	**T**	**F**	**T**	**T**	**F**	**F**	**F**

The column under the first occurrence of the tilde displays the truth-value of the entire sentence '~ [(U ∨ (W ⊃ ~ U)) ≡ W]' for each combination of truth-values that its atomic components might have. The truth-table tells us that '~ [(U ∨ (W ⊃ ~ U)) ≡ W]' has the truth-value **T** on those truth-value assignments on which either 'U' is assigned the truth-value **T** and 'W' is assigned the truth-value **F** or both 'U' and 'W' are assigned the truth-value **F**; the sentence is false on every other truth-value assignment.

Sometimes we are not interested in determining the truth-value of a sentence **P** on every truth-value assignment but are interested only in the truth-value of **P** on a particular truth-value assignment. In this case we may construct a shortened truth-table for **P** that records only the truth-values that its atomic components have on that truth-value assignment. For example, suppose we want to know the truth-value of '(A & B) ⊃ B' on a truth-value assignment that assigns **F** to 'A' and **T** to 'B' and all the other atomic sentences of *SL*. We head the shortened truth-table as before. We list only the combination of truth-values that 'A' and 'B' have on the assignment we are interested in:

					↓		
A	B	(A	&	B)	⊃	B
F	**T**	**F**	**F**	**T**	**T**	**T**

Our table shows that '(A & B)' has the truth-value **F** on this truth-value assignment, for 'A' has the truth-value **F**. Since the antecedent of '(A & B) ⊃ B' has the truth-value **F** and the consequent the truth-value **T**, '(A & B) ⊃ B' has the truth-value **T**.

We emphasize that, when we want to determine the truth-value of a sentence on a particular truth-value assignment we display only the truth-values that the assignment assigns to the atomic components of the sentence for which we are constructing a truth-table.

To review: The truth-value of a sentence **P** on a truth-value assignment is determined by starting with the truth-values of the atomic components of **P**

on the truth-value assignment and then using the characteristic truth-tables for the connectives of *SL* to compute the truth-values of larger and larger sentential components of **P** on the truth-value assignment. Ultimately we determine the truth-value of the largest sentential component of **P**, namely, **P** itself.

We also define the notions of being **true on a truth-value assignment** and **false on a truth-value assignment:**

> A sentence is *true on a truth-value assignment* if and only if it has the truth-value **T** on that truth-value assignment.
>
> A sentence is *false on a truth-value assignment* if and only if it has the truth-value **F** on that truth-value assignment.

3.1**E** EXERCISES

1. How many rows will be in the truth-table for each of the following sentences?

a. $A \equiv (\sim A \equiv A)$
*b. $[\sim D \ \& \ (B \vee G)] \supset [\sim (H \ \& \ A) \vee \sim D]$
c. $(B \ \& \ C) \supset [B \vee (C \ \& \sim C)]$

2. Construct truth-tables for the following sentences.

a. $\sim \sim (E \ \& \sim E)$
*b. $(A \ \& \ B) \equiv \sim B$
c. $A \equiv [J \equiv (A \equiv J)]$
*d. $[A \supset (B \supset C)] \ \& \ [(A \supset B) \supset C]$
e. $[\sim A \vee (H \supset J)] \supset (A \vee J)$
*f. $(\sim \sim A \ \& \sim B) \supset (\sim A \equiv B)$
g. $\sim (A \vee B) \supset (\sim A \vee \sim B)$
*h. $\sim D \ \& \ [\sim H \vee (D \ \& \ E)]$
i. $\sim (E \ \& \ [H \supset (B \ \& \ E)])$
*j. $\sim (D \equiv (\sim A \ \& \ B)) \vee (\sim D \vee \sim B)$
k. $\sim [D \ \& \ (E \vee F)] \equiv [\sim D \ \& \ (E \ \& \ F)]$
*l. $(J \ \& \ [(E \vee F) \ \& \ (\sim E \ \& \sim F)]) \supset \sim J$
m. $(A \vee (\sim A \ \& \ (H \supset J))) \supset (J \supset H)$

3. Construct shortened truth-tables to determine the truth-value of each of the following sentences on the truth-value assignment that assigns **T** to 'B' and 'C', and **F** to 'A' and to every other atomic sentence of *SL*.

a. $\sim [\sim A \vee (\sim C \vee \sim B)]$
*b. $\sim [A \vee (\sim C \ \& \sim B)]$
c. $(A \supset B) \vee (B \supset C)$
*d. $(A \supset B) \supset (B \supset C)$
e. $(A \equiv B) \vee (B \equiv C)$
*f. $\sim A \supset (B \equiv C)$
g. $\sim [B \supset (A \vee C)] \ \& \sim \sim B$
*h. $\sim [\sim A \equiv \sim (B \equiv \sim [A \equiv (B \ \& \ C)])]$
i. $\sim [\sim (A \equiv \sim B) \equiv \sim A] \equiv (B \vee C)$
*j. $\sim (B \supset \sim A) \ \& \ [C \equiv (A \ \& \ B)]$

3.2 TRUTH-FUNCTIONAL TRUTH, FALSITY, AND INDETERMINACY

In Chapter 1 we introduced the concepts of logical truth, logical falsity, and logical indeterminacy. Recall that a logically true sentence of English is one that cannot possibly be false. A sentence that is logically true (or logically false) may be so on purely truth-functional grounds. For example, we may symbolize 'Either Cynthia will get a job or Cynthia will not get a job' as 'C ∨ ~ C', and the truth-table for this sentence shows that it is true on every truth-value assignment:

		↓		
C	C	∨	~	C
T	**T**	**T**	**F**	**T**
F	**F**	**T**	**T**	**F**

A sentence that is logically true on truth-functional grounds is a **truth-functionally true** sentence.

> A sentence **P** of *SL* is *truth-functionally true* if and only if **P** is true on every truth-value assignment.[3]

Alternatively, a sentence **P** is truth-functionally true if and only if there is no truth-value assignment on which **P** is false.

Once the truth-table for a sentence has been constructed, it is a simple matter to determine whether that sentence is truth-functionally true: the sentence is truth-functionally true if and only if the column of truth-values under its main connective consists solely of **T**s. Since the rows of the truth-table represent *all* combinations of truth-values that may be assigned to the sentence's atomic components by any truth-value assignment, the absence of **F**s under the main connective shows that there is no truth-value assignment on which the sentence is false.

Here is the truth-table for another truth-functionally true sentence:

			↓			
X	Z	Z	⊃	(X	∨	Z)
T	**T**	**T**	**T**	**T**	**T**	**T**
T	**F**	**F**	**T**	**T**	**T**	**F**
F	**T**	**T**	**T**	**F**	**T**	**T**
F	**F**	**F**	**T**	**F**	**F**	**F**

[3]Truth-functionally true sentences are sometimes called *tautologies* or *truth-functionally valid* sentences. Truth-functionally false sentences (introduced shortly) are sometimes called *contradictions,* or *self-contradictory* sentences. Truth-functionally indeterminate sentences (also to be introduced) are sometimes called *contingent* sentences.

The column under the main connective of 'Z ⊃ (X ∨ Z)' contains only **T**s. Note that the immediate sentential components of a truth-functionally true sentence need not themselves be truth-functionally true.

Truth-functional falsity is also defined in terms of truth-value assignments.

> A sentence **P** of *SL* is *truth-functionally false* if and only if **P** is false on every truth-value assignment.

It follows that if **P** is truth-functionally false then there is no truth-value assignment on which **P** is true. We can show that a sentence of *SL* is truth-functionally false by constructing a truth-table for the sentence; if the column of truth-values under the sentence's main connective contains only **F**s, then the sentence is truth-functionally false. Here are truth-tables for two truth-functionally false sentences:

		↓	
A	A	&	~ A
T	**T**	**F**	**F T**
F	**F**	**F**	**T F**

											↓	
H	K	[(H	∨	K)	⊃	~	(H	∨	K)]	&	H
T	**T**	**T**	**T**	**T**	**F**	**F**	**T**	**T**	**T**	**F**	**T**
T	**F**	**T**	**T**	**F**	**F**	**F**	**T**	**T**	**F**	**F**	**T**
F	**T**	**F**	**T**	**T**	**F**	**F**	**F**	**T**	**T**	**F**	**F**
F	**F**	**F**	**F**	**F**	**T**	**T**	**F**	**F**	**F**	**F**	**F**

Note that the immediate sentential components of a truth-functionally false sentence need not themselves be truth-functionally false. Whenever we negate a truth-functionally true sentence, the result is a truth-functionally false sentence, as the following example shows:

	↓			
A	~	(A	∨	~ A)
T	**F**	**T**	**T**	**F T**
F	**F**	**F**	**T**	**T F**

If we add another tilde to obtain '~ ~ (A ∨ ~ A)', we will once again have a truth-functionally true sentence.

Although the two sentences 'A ⊃ (B ⊃ A)' and '(A ⊃ B) ⊃ A' look very much alike, one is truth-functionally true and the other is not:

				↓			
A	B	A	⊃	(B	⊃	A)
T	**T**	**T**	**T**	**T**	**T**	**T**
T	**F**	**T**	**T**	**F**	**T**	**T**
F	**T**	**F**	**T**	**T**	**F**	**F**
F	**F**	**F**	**T**	**F**	**T**	**F**

					↓	
A	B	(A	⊃	B)	⊃	A
T	**T**	**T**	**T**	**T**	**T**	**T**
T	**F**	**T**	**F**	**F**	**T**	**T**
F	**T**	**F**	**T**	**T**	**F**	**F**
F	**F**	**F**	**T**	**F**	**F**	**F**

'A ⊃ (B ⊃ A)' is true on every truth-value assignment, whereas '(A ⊃ B) ⊃ A' is not. The latter sentence is **truth-functionally indeterminate.**

> A sentence **P** of *SL* is *truth-functionally indeterminate* if and only if **P** is neither truth-functionally true nor truth-functionally false.

A truth-functionally indeterminate sentence is true on at least one truth-value assignment and false on at least one truth-value assignment. Every atomic sentence of *SL* is truth-functionally indeterminate. For example, the truth-table for 'H' is

	↓
H	H
T	**T**
F	**F**

'H' is true on every truth-value assignment on which it is assigned the truth-value **T**, and false on every other truth-value assignment. Truth-tables for several truth-functionally indeterminate sentences appeared in Section 3.1. Every sentence of *SL* is either truth-functionally true, truth-functionally false, or truth-functionally indeterminate.

Sometimes we can show that a sentence is not truth-functionally true or is not truth-functionally false by constructing a shortened truth-table. Consider the sentence '(A & ~ A) ∨ ~ A'. If this sentence is truth-functionally true, then there is no truth-value assignment on which it is false. So, if we can show that the sentence is false on at least one truth-value assignment, then we can conclude that it is not truth-functionally true. The following shortened truth-table shows this:

					↓		
A	(A	&	~ A)	∨	~ A
T	**T**	**F**	**F T**	**F**	**F T**

This shortened truth-table shows that the sentence '(A & ~ A) ∨ ~ A' is false on every truth-value assignment that assigns the truth-value **T** to 'A'. Note that the shortened table shows *only* that '(A & ~ A) ∨ ~ A' is not truth-functionally true. The table does not show whether the sentence is truth-functionally false or truth-functionally indeterminate. However, it is easy to show that it is the latter by constructing a shortened truth-table in which the value under the main connective is **T**.

Similarly we may construct a shortened truth-table in order to show that 'J & (~ K ∨ ~ J)' is not truth-functionally false:

				↓				
J	K	J	&	(~ K	∨	~ J)		
T	**F**	**T**	**T**	**T**	**F**	**T**	**F**	**T**

This truth-table shows that the sentence is true on every truth-value assignment that assigns **T** to 'J' and **F** to 'K'. We thus know that the sentence is either truth-functionally indeterminate or truth-functionally true.

There is a systematic way to develop a shortened truth-table that shows that a sentence is true on at least one truth-value assignment or false on at least one truth-value assignment. Let's first consider the previous example, in which we wanted to show that 'J & (~ K ∨ ~ J)' is true on at least one truth-value assignment. We start by placing a **T** under the main connective:

			↓			
J	K	J	&	(~ K	∨	~ J)
			T			

Because the main connective is an ampersand, we know that each conjunct must be true as well:

			↓			
J	K	J	&	(~ K	∨	~ J)
		T	**T**			**T**

Whenever we place a **T** or **F** under a sentence letter, we repeat it under all occurrences of that sentence letter:

			↓				
J	K	J	&	(~ K	∨	~ J)	
T		**T**	**T**			**T**	**T**

Once we have placed a **T** under 'J', we know that we must fill in an **F** under the tilde preceding 'J', since a negation is false if the negated sentence is true:

				↓				
J	K	J	&	(~ K	∨	~ J)		
T		**T**	**T**			**T**	**FT**	

Now we have a true disjunction with one false disjunct, so we know that the other disjunct must be true (otherwise the disjunction could not be true):

				↓				
J	K	J	&	(~ K	∨	~ J)		
T		**T**	**T**	**T**		**T**	**FT**	

And if '~ K' is true, then 'K' must be false:

					↓				
J	K	J	&	(~ K	∨	~ J)			
T	**F**	**T**	**T**	**T**	**F**	**T**	**FT**		

Note that we also placed an **F** under the occurrence of 'K' to the left of the vertical bar. This completes our shortened truth-table, and we have shown that the sentence is not truth-functionally false.

Now consider the earlier example, in which we wanted to show that '(A & ~ A) ∨ ~ A' is false on at least one truth-value assignment (and therefore not truth-functionally true). We begin by placing an **F** under the sentence's main connective:

					↓	
A	(A	&	~ A)		∨	~ A
					F	

If a disjunction is false, both of its disjuncts must be false:

					↓	
A	(A	&	~ A)		∨	~ A
		F			**F**	**F**

We have just recorded an **F** for '~ A', and since '~ A' occurs elsewhere in the sentence, we repeat the **F** there:

					↓	
A	(A	&	~ A)		∨	~ A
	F	**F**			**F**	**F**

Note that we have now assigned the value **F** to one of the conjuncts of '(A & ~ A)', thus ensuring that the conjunction is false, so it won't matter if we end up assigning the value **T** to the other conjunct. Next we note that if '~ A' is false then 'A' must be true:

					↓		
A	(A	&	~ A)	∨		~ A	
T	**T**	**F**	**F T**	**F**		**F T**	

And this completes the shortened truth-table.

In these two examples, every addition to the table was dictated by some previous truth-value that had been entered: If a conjunction is true, both conjuncts must be true; if a disjunction is false, both disjuncts must be false; a negation is true if and only if the negated sentence is false; and a component of a sentence must have the same truth-value for each of its occurrences. But sometimes choices have to be made. For example, suppose we want to show that the sentence '(A ⊃ B) ≡ (B ⊃ A)' is not truth-functionally true. We can begin constructing a shortened truth-table by placing an **F** under the sentence's main connective:

						↓			
A	B	(A	⊃	B)	≡	(B	⊃	A)
					F			

At this point we have to make a choice, because there are two ways that a biconditional can be false. Either the first immediate component is true and the second false, or the first immediate component is false and the second true. There is no simple rule of thumb to follow in this case. So we'll try one of the possibilities and see where it leads:

						↓			
A	B	(A	⊃	B)	≡	(B	⊃	A)
			T		**F**		**F**		

Since '(B ⊃ A)' is false, we know that 'B' must be true and 'A' false. We'll add these values:

						↓			
A	B	(A	⊃	B)	≡	(B	⊃	A)
F	**T**		**T**		**F**	**T**	**F**	**F**	

We also need to add the values under the other occurrences of 'A' and 'B'—but in doing so we must make sure that these values are consistent with the assignment

of **T** to the conditional '(A ⊃ B)':

					↓			
A	B	(A	⊃	B)	≡	(B	⊃	A)
F	**T**	**F**	**T**	**T**	**F**	**T**	**F**	**F**

Fortunately they are: A conditional with a false antecedent and a true consequent is itself true. So we have successfully completed the shortened table.

It turns out that we could have assigned **F** to the first immediate component of the biconditional and **T** to the second and produced another shortened truth-table representing a different set of truth-value assignments on which the biconditional is false. But sometimes, when we have a choice, one possible way of assigning truth-values won't work while another one will. Suppose, for example, that we want to show that the sentence '(A ⊃ B) ⊃ (B ⊃ ~ A)' is not truth-functionally false—that is, that there is at least one truth-value assignment on which it is true. We start with

					↓			
A	B	(A	⊃	B)	⊃	(B	⊃	~ A)
					T			

There are three ways in which a conditional can be true: Both the antecedent and consequent are true, or the antecedent is false and the consequent is true, or the antecedent is false and the consequent is false. We might try the first case first:

					↓			
A	B	(A	⊃	B)	⊃	(B	⊃	~ A)
			T		**T**		**T**	

We now have two true conditionals whose immediate components do not have truth-values. We'll work with the first one, and again, let's make its antecedent true and its consequent true:

					↓			
A	B	(A	⊃	B)	⊃	(B	⊃	~ A)
		T	**T**	**T**	**T**		**T**	

Filling in **T** under 'A' and 'B' wherever they occur—because 'A' and 'B' have each been assigned the truth-value **T**—we get

					↓			
A	B	(A	⊃	B)	⊃	(B	⊃	~ A)
T	**T**	**T**	**T**	**T**	**T**	**T**	**T**	**T**

Now we must put **F** under the tilde:

					↓				
A	B	(A	⊃	B)	⊃	(B	⊃	~ A)	
T	**T**	**T**	**T**	**T**	**T**	**T**	**T**	**F**	**T**

FAILURE!

The problem is that the conditional '(B ⊃ ~ A)' cannot be true if 'B' is true and '~ A' is false.

But we must not conclude that the sentence *cannot* be true. All we conclude is that we haven't come up with a way of assigning truth-values that will make it true. We can go back to

					↓				
A	B	(A	⊃	B)	⊃	(B	⊃	~ A)	
			T		**T**		**T**		

and try another way to make the conditional '(A ⊃ B)' true—say, by making 'A' false and 'B' true. This yields

					↓				
A	B	(A	⊃	B)	⊃	(B	⊃	~ A)	
F	**T**	**F**	**T**	**T**	**T**	**T**	**T**	**F**	

and we can fill in a **T** under the tilde:

					↓				
A	B	(A	⊃	B)	⊃	(B	⊃	~ A)	
F	**T**	**F**	**T**	**T**	**T**	**T**	**T**	**T**	**F**

Note that this time the conditional '(B ⊃ ~ A)' will be true since both of its immediate components are, so we have produced a shortened truth-table that shows the sentence is not truth-functionally false. But even if this hadn't worked, there are still other possibilities, including trying to make the entire sentence true by a different assignment of truth-values to its immediate components.[4]

Of course, we may fail even when we try all the possibilities—which means that, although we thought a sentence might be true (or false) on some truth-value assignment, we were incorrect. Here's a simple example: We'll try to

[4]Sometimes we have to try every possibility before coming up with a correct shortened truth-table (or concluding that there is no such table). The problem in constructing a shortened truth-table to show that a sentence can be true or that it can be false is one of a class of problems known to theoreticians as 'NP-complete problems'. These are problems for which the only known solutions guaranteed to produce a correct result are solutions that require us, in the worst case, to try every possibility.

produce a shortened truth-table with an assignment of truth-values that makes the sentence 'A ⊃ A' false:

A	A	⊃	A
		↓	
		F	

If the conditional is false, the antecedent must be true and the consequent false:

A	A	⊃	A	
		↓		
	T	**F**	**F**	**FAILURE!**

We failed because 'A' cannot have two different truth-values on the same truth-value assignment. Here we have, in fact, tried all the possibilities for making the conditional false (the antecedent must be true and the conclusion must be false)—unsuccessfully. That's as it should be, since the sentence is truth-functionally true.

3.2**E** EXERCISES

1. Construct a full truth-table for each of the following sentences of *SL*, and state whether the sentence is truth-functionally true, truth-functionally false, or truth-functionally indeterminate.

a. ~ A ⊃ A
*b. J ⊃ (K ⊃ J)
c. (A ≡ ~ A) ⊃ ~ (A ≡ ~ A)
*d. (E ≡ H) ⊃ (~ E ⊃ ~ H)
e. (~ B & ~ D) ∨ ~ (B ∨ D)
*f. ([(C ⊃ D) & (D ⊃ E)] & C) & ~ E
g. [(A ∨ B) & (A ∨ C)] ⊃ ~ (B & C)
*h. ~ [[(A ∨ B) & (B ∨ B)] & (~ A & ~ B)]
i. (J ∨ ~ K) ≡ ~ ~ (K ⊃ J)
*j. ~ B ⊃ [(B ∨ D) ⊃ D]
k. [(A ∨ ~ D) & ~ (A & D)] ⊃ ~ D
*l. (M ≡ ~ N) & (M ≡ N)

2. For each of the following sentences, either show that the sentence is truth-functionally true by constructing a full truth-table or show that the sentence is not truth-functionally true by constructing an appropriate shortened truth-table.

a. (F ∨ H) ∨ (~ F ≡ H)
*b. (F ∨ H) ∨ ~ (~ F ⊃ H)
c. ~ A ⊃ [(B & A) ⊃ C]
*d. A ≡ (B ≡ A)
e. [(C ∨ ~ C) ⊃ C] ⊃ C
*f. [C ⊃ (C ∨ ~ D)] ⊃ (C ∨ D)

3. Construct truth-tables to show that the following sentences of *SL* are truth-functionally true.

a. $A \supset (A \vee B)$
*b. $A \supset (B \supset A)$
c. $A \supset [B \supset (A \,\&\, B)]$
*d. $(A \,\&\, B) \supset [(A \vee C) \,\&\, (B \vee C)]$
e. $(A \equiv B) \supset (A \supset B)$
*f. $(A \,\&\, \sim A) \supset (B \,\&\, \sim B)$
g. $(A \supset B) \supset [(C \supset A) \supset (C \supset B)]$
*h. $A \vee \sim A$
i. $[(A \supset B) \,\&\, \sim B] \supset \sim A$
*j. $(A \,\&\, A) \equiv A$
k. $A \supset [B \supset (A \supset B)]$
*l. $\sim A \supset [(B \,\&\, A) \supset C]$
m. $(A \supset B) \supset [\sim B \supset \sim (A \,\&\, D)]$
*n. $[(A \supset B) \supset A] \supset A$
o. $\sim (A \equiv B) \equiv (\sim A \equiv B)$
*p. $(\sim A \equiv B) \equiv (A \equiv \sim B)$

4. For each of the following sentences of *SL*, either show that the sentence is truth-functionally false by constructing a full truth-table or show that the sentence is not truth-functionally false by constructing an appropriate shortened truth-table.

a. $(B \equiv D) \,\&\, (B \equiv \sim D)$
*b. $(B \supset H) \,\&\, (B \supset \sim H)$
c. $A \equiv (B \equiv A)$
*d. $[(F \,\&\, G) \supset (C \,\&\, \sim C)] \,\&\, F$
e. $[(C \vee D) \equiv C] \supset \sim C$
*f. $[\sim (A \,\&\, F) \supset (B \vee A)] \,\&\, \sim [\sim B \supset \sim (F \vee A)]$

5. Which of the following claims about sentences of *SL* are true? Explain.

a. A conjunction with one truth-functionally true conjunct must itself be truth-functionally true.
*b. A disjunction with one truth-functionally true disjunct must itself be truth-functionally true.
c. A material conditional with a truth-functionally true consequent must itself be truth-functionally true.
*d. A conjunction with one truth-functionally false conjunct must itself be truth-functionally false.
e. A disjunction with one truth-functionally false disjunct must itself be truth-functionally false.
*f. A material conditional with a truth-functionally false consequent must itself be truth-functionally false.
g. A sentence is truth-functionally true if and only if its negation is truth-functionally false.
*h. A sentence is truth-functionally indeterminate if and only if its negation is truth-functionally indeterminate.
i. A material conditional with a truth-functionally true antecedent must itself be truth-functionally true.
*j. A material conditional with a truth-functionally false antecedent must itself be truth-functionally false.

6. Where **P** and **Q** are sentences of *SL*, answer the following questions; explain your answers.

a. Suppose that **P** is a truth-functionally true sentence and **Q** is a truth-functionally false sentence. On the basis of this information, can you determine whether **P** ≡ **Q** is truth-functionally true, false, or indeterminate? If so, which is it?

*b. Suppose that **P** and **Q** are truth-functionally indeterminate sentences. Does it follow that **P** & **Q** is truth-functionally indeterminate?

c. Suppose that **P** and **Q** are truth-functionally indeterminate. Does it follow that **P** ∨ **Q** is truth-functionally indeterminate?

*d. Suppose that **P** is a truth-functionally true sentence and that **Q** is truth-functionally indeterminate. On the basis of this information, can you determine whether **P** ⊃ **Q** is truth-functionally true, false, or indeterminate? If so, which is it?

3.3 TRUTH-FUNCTIONAL EQUIVALENCE

We now introduce the concept of **truth-functional equivalence.**

> Sentences **P** and **Q** of *SL* are *truth-functionally equivalent* if and only if there is no truth-value assignment on which **P** and **Q** have different truth-values.

To show that **P** and **Q** are truth-functionally equivalent, we construct a single truth-table for both **P** and **Q** and show that in each row the two sentences have the same truth-value. The columns under the *main* connectives must be identical.

The sentences 'A & A' and 'A ∨ A' are truth-functionally equivalent, as shown by the following truth-table:

		↓			↓	
A	A	&	A	A	∨	A
T	**T**	**T**	**T**	**T**	**T**	**T**
F	**F**	**F**	**F**	**F**	**F**	**F**

On any truth-value assignment that assigns **T** to 'A', both sentences are true. On any truth-value assignment that assigns **F** to 'A', both sentences are false. The sentences '(W & Y) ⊃ H' and 'W ⊃ (Y ⊃ H)' are also truth-functionally equivalent:

						↓			↓			
H	W	Y	(W	&	Y)	⊃	H	W	⊃	(Y	⊃	H)
T	**T**	**T**	**T**	**T**	**T**	**T**	**T**	**T**	**T**	**T**	**T**	**T**
T	**T**	**F**	**T**	**F**	**F**	**T**	**T**	**T**	**T**	**F**	**T**	**T**
T	**F**	**T**	**F**	**F**	**T**	**T**	**T**	**F**	**T**	**T**	**T**	**T**
T	**F**	**F**	**F**	**F**	**F**	**T**	**T**	**F**	**T**	**F**	**T**	**T**
F	**T**	**T**	**T**	**T**	**T**	**F**	**F**	**T**	**F**	**T**	**F**	**F**
F	**T**	**F**	**T**	**F**	**F**	**T**	**F**	**T**	**T**	**F**	**T**	**F**
F	**F**	**T**	**F**	**F**	**T**	**T**	**F**	**F**	**T**	**T**	**F**	**F**
F	**F**	**F**	**F**	**F**	**F**	**T**	**F**	**F**	**T**	**F**	**T**	**F**

The columns under the main connectives of '(W & Y) ⊃ H' and 'W ⊃ (Y ⊃ H)' are identical, which shows that the two sentences have the same truth-value on every truth-value assignment.

Now consider the following truth-table:

E	H	J	E	∨	H	(H	∨	J)	∨	E
T	**T**	**T**	**T**	**T**	**T**	**T**	**T**	**T**	**T**	**T**
T	**T**	**F**	**T**	**T**	**T**	**T**	**T**	**F**	**T**	**T**
T	**F**	**T**	**T**	**T**	**F**	**F**	**T**	**T**	**T**	**T**
T	**F**	**F**	**T**	**T**	**F**	**F**	**F**	**F**	**T**	**T**
F	**T**	**T**	**F**	**T**	**T**	**T**	**T**	**T**	**T**	**F**
F	**T**	**F**	**F**	**T**	**T**	**T**	**T**	**F**	**T**	**F**
F	**F**	**T**	**F**	**F**	**F**	**F**	**T**	**T**	**T**	**F**
F	**F**	**F**	**F**	**F**	**F**	**F**	**F**	**F**	**F**	**F**

The table shows that the sentences 'E ∨ H' and '(H ∨ J) ∨ E' are not truth-functionally equivalent, for they have different truth-values on any truth-value assignment that assigns **F** to 'E' and 'H' and **T** to 'J'. When a truth-table shows that two sentences are not truth-functionally equivalent, we will draw a box around a row of the truth-table in which the sentences do not have the same truth-value.

All truth-functionally true sentences are truth-functionally equivalent. This is because every truth-functionally true sentence has the truth-value **T** on every truth-value assignment. For example, '~ (C & ~ C)' and 'A ⊃ (B ⊃ A)' are truth-functionally equivalent:

A	B	C	~	(C	&	~ C)	A	⊃	(B	⊃	A)
T	**T**	**T**	**T**	**T**	**F**	**F T**	**T**	**T**	**T**	**T**	**T**
T	**T**	**F**	**T**	**F**	**F**	**T F**	**T**	**T**	**T**	**T**	**T**
T	**F**	**T**	**T**	**T**	**F**	**F T**	**T**	**T**	**F**	**T**	**T**
T	**F**	**F**	**T**	**F**	**F**	**T F**	**T**	**T**	**F**	**T**	**T**
F	**T**	**T**	**T**	**T**	**F**	**F T**	**F**	**T**	**T**	**F**	**F**
F	**T**	**F**	**T**	**F**	**F**	**T F**	**F**	**T**	**T**	**F**	**F**
F	**F**	**T**	**T**	**T**	**F**	**F T**	**F**	**T**	**F**	**T**	**F**
F	**F**	**F**	**T**	**F**	**F**	**T F**	**F**	**T**	**F**	**T**	**F**

The columns under the main connectives are identical. Likewise, all truth-functionally false sentences are truth-functionally equivalent.

But not all truth-functionally indeterminate sentences are truth-functionally equivalent—for example,

			↓					↓		
B	D	B	&	D		~	B	&	D	
T	**T**	**T**	**T**	**T**		**F**	**T**	**F**	**T**	
T	**F**	**T**	**F**	**F**		**F**	**T**	**F**	**F**	
F	**T**	**F**	**F**	**T**		**T**	**F**	**T**	**T**	
F	**F**	**F**	**F**	**F**		**T**	**F**	**F**	**F**	

On any truth-value assignment on which 'B' and 'D' are both true, or 'B' is false and 'D' is true, the sentences 'B & D' and '~ B & D' have different truth-values. Hence they are not truth-functionally equivalent.

If **P** and **Q** are not truth-functionally equivalent, we can construct a shortened truth-table to show this. The shortened truth-table will display a combination of truth-values for which one sentence is true and the other false. For example, the following shortened truth-table shows that 'A' and 'A ∨ B' are not truth-functionally equivalent:

		↓		↓	
A	B	A	A	∨	B
F	**T**	**F**	**F**	**T**	**T**

The shortened truth-table shows that, on any truth-value assignment that assigns **F** to 'A' and **T** to 'B', 'A' is false and 'A ∨ B' is true. Note that, if we construct a shortened truth-table that includes a row in which both sentences have the same truth-value, this is not sufficient to show that they are truth-functionally equivalent. This is because they are truth-functionally equivalent if and only if they have the same truth-value on *every* truth-value assignment. To show this, we must consider every combination of truth-values that their atomic components might have.

We can construct shortened truth-tables for two (or more) sentences in a systematic way, just as we did for single sentences in Section 3.2. For example, we could begin constructing the previous table by assigning the sentence 'A' the truth-value **F** and 'A ∨ B' the truth-value **T**:

		↓		↓	
A	B	A	A	∨	B
		F		**T**	

(We might first have tried to make 'A' true and 'A ∨ B' false, but this would not lead to a correct truth-table since we would have a false disjunction with a true disjunct.) Filling in **F** under all the other occurrences of 'A' yields

		↓		↓	
A	B	A	A	∨	B
F		**F**	**F**	**T**	

Now we can make 'B' true, which will secure the truth of the disjunction:

		↓			↓	
A	B	A		A	∨	B
F	**T**	**F**		**F**	**T**	**T**

In Chapter 2 we noted that compound sentences whose main connective is 'unless' can be paraphrased either as disjunctions or as material conditionals. That is, English sentences of the form

p unless **q**

can be paraphrased and symbolized in all of the following ways:

Either **p** or **q**	**P** ∨ **Q**
If it is not the case that **p** then **q**	~ **P** ⊃ **Q**
If it is not the case that **q** then **p**	~ **Q** ⊃ **P**

These paraphrases and symbolizations are all correct because, as we can now show, for any sentences **P** and **Q** of *SL*, the sentences **P** ∨ **Q**, ~ **P** ⊃ **Q**, and ~ **Q** ⊃ **P** are truth-functionally equivalent:

			↓				↓				↓		
P	**Q**	**P**	∨	**Q**	~	**P**	⊃	**Q**	~	**Q**	⊃	**P**
T	**T**	**T**	**T**	**T**	**F**	**T**	**T**	**T**	**F**	**T**	**T**	**T**
T	**F**	**T**	**T**	**F**	**F**	**T**	**T**	**F**	**T**	**F**	**T**	**T**
F	**T**	**F**	**T**	**T**	**T**	**F**	**T**	**T**	**F**	**T**	**T**	**F**
F	**F**	**F**	**F**	**F**	**T**	**F**	**F**	**F**	**T**	**F**	**F**	**F**

Note that the above table is not a truth-table for specific sentences of *SL*, because '**P**' and '**Q**' are not sentences of *SL* but metavariables ranging over sentences of *SL*.

Similarly, for any sentences **P** and **Q** of *SL*, ~ (**P** & **Q**) and ~ **P** ∨ ~ **Q** are also truth-functionally equivalent:

		↓					↓		
P	**Q**	~	(**P**	**&**	**Q**)	~	**P**	∨	~	**Q**
T	**T**	**F**	**T**	**T**	**T**	**F**	**T**	**F**	**F**	**T**
T	**F**	**T**	**T**	**F**	**F**	**F**	**T**	**T**	**T**	**F**
F	**T**	**T**	**F**	**F**	**T**	**T**	**F**	**T**	**F**	**T**
F	**F**	**T**	**F**	**F**	**F**	**T**	**F**	**T**	**T**	**F**

as are ~ (**P** ∨ **Q**) and ~ **P** & ~ **Q**:

			↓						↓		
P	**Q**		~	(**P**	∨	**Q**)	~	**P**	**&**	~	**Q**
T	**T**		**F**	**T**	**T**	**T**	**F**	**T**	**F**	**F**	**T**
T	**F**		**F**	**T**	**T**	**F**	**F**	**T**	**F**	**T**	**F**
F	**T**		**F**	**F**	**T**	**T**	**T**	**F**	**F**	**F**	**T**
F	**F**		**T**	**F**	**F**	**F**	**T**	**F**	**T**	**T**	**F**

3.3**E** EXERCISES

1. Determine, by constructing full truth-tables, which of the following pairs of sentences of *SL* are truth-functionally equivalent.

a.	~ (A & B)	~ (A ∨ B)
*b.	A ⊃ (B ⊃ A)	(C & ~ C) ∨ (A ⊃ A)
c.	K ≡ H	~ K ≡ ~ H
*d.	C & (B ∨ A)	(C & B) ∨ A
e.	(G ⊃ F) ⊃ (F ⊃ G)	(G ≡ F) ∨ (~ F ∨ G)
*f.	~ C ⊃ ~ B	B ⊃ C
g.	~ (H & J) ≡ (J ≡ ~ K)	(H & J) ⊃ ~ K
*h.	~ (D ∨ B) ⊃ (C ⊃ B)	C ⊃ (D & B)
i.	[A ∨ ~ (D & C)] ⊃ ~ D	[D ∨ ~ (A & C)] ⊃ ~ A
*j.	A ⊃ [B ⊃ (A ⊃ B)]	B ⊃ [A ⊃ (B ⊃ A)]
k.	F ∨ ~ (G ∨ ~ H)	(H ≡ ~ F) ∨ G

2. For each of the following pairs of sentences of *SL*, either show that the sentences are truth-functionally equivalent by constructing a full truth-table or show that they are not truth-functionally equivalent by constructing an appropriate shortened truth-table.

a.	G ∨ H	~ G ⊃ H
*b.	~ (B & ~ A)	A ∨ B
c.	(D ≡ A) & D	D & A
*d.	F & (J ∨ H)	(F & J) ∨ H
e.	A ≡ (~ A ≡ A)	~ (A ⊃ ~ A)
*f.	~ (~ B ∨ (~ C ∨ ~ D))	(D ∨ C) & ~ B

3. Symbolize each of the following pairs of sentences and determine which of the pairs of sentences are truth-functionally equivalent by constructing truth-tables.

a. Unless the sky clouds over, the night will be clear and the moon will shine brightly.
   The moon will shine brightly if and only if the night is clear and the sky doesn't cloud over.

*b. Although the new play at the Roxy is a flop, critics won't ignore it unless it is canceled.
   The new play at the Roxy is a flop, and if it is canceled critics will ignore it.

c. If the *Daily Herald* reports on our antics, then the antics are effective.
   If our antics aren't effective, then the *Daily Herald* won't report on them.

*d. The year 1972 wasn't a good vintage year, 1973 was, and neither 1974 nor 1975 was.
Neither 1974 nor 1972 was a good vintage year, and not both 1973 and 1975 were.

e. If Mary met Tom and she liked him, then Mary didn't ask George to the movies.
If Mary met Tom and she didn't like him, then Mary asked George to the movies.

*f. Either the blue team or the red team will win the tournament, and they won't both win.
The red team will win the tournament if and only if the blue team won't win the tournament.

4. Suppose that sentences **P** and **Q** are truth-functionally equivalent.

a. Are ~ **P** and ~ **Q** truth-functionally equivalent? Explain.

*b. Show that **P** and **P** & **Q** are also truth-functionally equivalent.

c. Show that ~ **P** ∨ **Q** is truth-functionally true.

5. Suppose we construct two truth-tables, one for 'A ∨ B' and another for 'B ∨ C', and that the columns of truth-values under the sentences' main connectives are identical. Does it follow that 'A ∨ B' and 'B ∨ C' are truth-functionally equivalent? Explain.

6. Show that for any sentences **P** and **Q** of *SL*, the following pairs of sentences are truth-functionally equivalent.

a.	**P** ≡ **Q**	(**P** ⊃ **Q**) & (**Q** ⊃ **P**)
*b.	**P** ⊃ **Q**	~ **Q** ⊃ ~ **P**
c.	**P** & (**Q** ∨ **R**)	(**P** & **Q**) ∨ (**P** & **R**)
*d.	**P** ∨ (**Q** & **R**)	(**P** ∨ **Q**) & (**P** ∨ **R**)
e.	~ (**P** ≡ **Q**)	~ **P** ≡ **Q**
*f.	**P** & (**Q** & **R**)	(**P** & **Q**) & **R**

3.4 TRUTH-FUNCTIONAL CONSISTENCY

To define truth-functional consistency, we need the notion of a *set* of sentences, informally introduced in Chapter 1. A set of sentences of *SL* is a collection, possibly empty, of zero or more sentences of *SL*, the members of the set. We can specify a finite set of sentences by listing the names of the sentences, separated by commas, within a pair of curly brackets. Thus {A, B ⊃ H, C ∨ A} is the set of sentences consisting of 'A', 'B ⊃ H', and 'C ∨ A'. We adopt the convention that *SL* sentences occurring between the curly brackets are being mentioned, so that we do not need to enclose them within quotation marks.

All sets of sentences of *SL* that have at least one member are nonempty sets of sentences. The empty set, denoted by '∅', has no members. In what follows we shall use the variable 'Γ' (gamma), with or without a subscript, to range over sets of sentences of *SL*.

We can now introduce **truth-functional consistency:**

A set of sentences of *SL* is *truth-functionally consistent* if and only if there is at least one truth-value assignment on which all the members of the set are true. A set of sentences of *SL* is *truth-functionally inconsistent* if and only if it is not truth-functionally consistent.

The set {A, B ⊃ H, B} is truth-functionally consistent, as is shown by the following truth-table:

			↓			↓		↓
A	B	H	A		B	⊃	H	B
T	**T**	**T**	**T**		**T**	**T**	**T**	**T**
T	**T**	**F**	**T**		**T**	**F**	**F**	**T**
T	**F**	**T**	**T**		**F**	**T**	**T**	**F**
T	**F**	**F**	**T**		**F**	**T**	**F**	**F**
F	**T**	**T**	**F**		**T**	**T**	**T**	**T**
F	**T**	**F**	**F**		**T**	**F**	**F**	**T**
F	**F**	**T**	**F**		**F**	**T**	**T**	**F**
F	**F**	**F**	**F**		**F**	**T**	**F**	**F**

The truth-table shows that, on any truth-value assignment on which 'A', 'B', and 'H' are all true, all three set members are true. So the set is truth-functionally consistent. We have drawn a box around the row of the truth-table that shows this (in this case, there is only one such row).

The set of sentences {L, L ⊃ J, ~ J} is truth-functionally inconsistent:

		↓			↓			↓
J	L	L		L	⊃	J		~ J
T	**T**	**T**		**T**	**T**	**T**		**F T**
T	**F**	**F**		**F**	**T**	**T**		**F T**
F	**T**	**T**		**T**	**F**	**F**		**T F**
F	**F**	**F**		**F**	**T**	**F**		**T F**

In each row at least one of the three sentences has the truth-value **F** under its main connective. Hence there is no truth-value assignment on which all three set members are true. The following set of sentences is also truth-functionally inconsistent: {C ∨ ~ C, ~ C & D, ~ D}.

			↓				↓			↓		
C	D	C	∨	~ C		~ C	&	D		~ D		
T	**T**	**T**	**T**	**F T**		**F T**	**F**	**T**		**F T**		
T	**F**	**T**	**T**	**F T**		**F T**	**F**	**F**		**T F**		
F	**T**	**F**	**T**	**T F**		**T F**	**T**	**T**		**F T**		
F	**F**	**F**	**T**	**T F**		**T F**	**F**	**F**		**T F**		

In this case it does not matter that one of the sentences, 'C ∨ ~ C', is true on every truth-value assignment. All that matters for establishing truth-functional inconsistency is that there is no truth-value assignment on which all three members are true.

We can show that a finite set of sentences of *SL* is truth-functionally consistent by constructing a shortened truth-table that displays one row in which all the set members are true. For instance, the following shortened truth-table shows that the set {(E ≡ H) ≡ E, H & ~ E} is truth-functionally consistent:

					↓			↓		
E	H	(E	≡	H)	≡	E	H	&	~	E
F	**T**	**F**	**F**	**T**	**T**	**F**	**T**	**T**	**T**	**F**

Note that if we construct a shortened table that displays a row in which not all the members of the set are true, this is not sufficient to show that the set is truth-functionally inconsistent. This is because a set of sentences is truth-functionally inconsistent if and only if there is *no* truth-value assignment on which every member of the set is true. To show this, we have to consider every combination of truth-values that the atomic components of the set members might have.

3.4**E** EXERCISES

1. Construct full truth-tables for each of the following sets of sentences and indicate whether they are truth-functionally consistent or truth-functionally inconsistent.

a. {A ⊃ B, B ⊃ C, A ⊃ C}
*b. {B ≡ (J & K), ~ J, ~ B ⊃ B}
c. {~ [J ∨ (H ⊃ L)], L ≡ (~ J ∨ ~ H), H ≡ (J ∨ L)}
*d. {(A & B) & C, C ∨ (B ∨ A), A ≡ (B ⊃ C)}
e. {(J ⊃ J) ⊃ H, ~ J, ~ H}
*f. {U ∨ (W & H), W ≡ (U ∨ H), H ∨ ~ H}
g. {A, B, C}
*h. {~ (A & B), ~ (B & C), ~ (A & C), A ∨ (B & C)}
i. {(A & B) ∨ (C ⊃ B), ~ A, ~ B}
*j. {A ⊃ (B ⊃ (C ⊃ A)), B ⊃ ~ A}

2. For each of the following sets of sentences, either show that the set is truth-functionally consistent by constructing an appropriate shortened truth-table or show that the set is truth-functionally inconsistent by constructing a full truth-table.

a. {B ⊃ (D ⊃ E), ~ D & B}
*b. {H ≡ (~ H ⊃ H)}
c. {F ⊃ (J ∨ K), F ≡ ~ J}
*d. {~ (~ C ∨ ~ B) & A, A ≡ ~ C}
e. {(A ⊃ B) ≡ (~ B ∨ B), A}
*f. {H ⊃ J, J ⊃ K, K ⊃ ~ H}

3. Symbolize each of the following passages in *SL* and determine whether the resulting set of sentences is truth-functionally consistent. If the set is truth-functionally consistent, construct a shortened truth-table that shows this. If it is truth-functionally inconsistent, construct a full truth-table.

a. If space is infinitely divisible, then Zeno's paradoxes are compelling. Zeno's paradoxes are neither convincing nor compelling. Space is infinitely divisible.

*b. Newtonian mechanics can't be right if Einsteinian mechanics is. But Einsteinian mechanics is right if and only if space is non-Euclidean. Space is non-Euclidean, or Newtonian mechanics is correct.

c. Eugene O'Neil was an alcoholic. His plays show it. But *The Iceman Cometh* must have been written by a teetotaler. O'Neill was an alcoholic unless he was a fake.

*d. Neither sugar nor saccharin is desirable if and only if both are lethal. Sugar is lethal if and only if saccharin is desirable. Sugar is undesirable if and only if saccharin isn't lethal.

e. If the Red Sox win next Sunday, then if Joan bet $5 against them she'll buy Ed a hamburger. The Red Sox won't win, and Joan won't buy Ed a hamburger.

*f. Either Johnson or Hartshorne pleaded guilty, or neither did. If Johnson pleaded guilty, then the newspaper story is incorrect. The newspaper story is correct, and Hartshorne pleaded guilty.

4. Where **P** and **Q** are sentences of *SL*,

a. Prove that {**P**} is truth-functionally inconsistent if and only if ~ **P** is truth-functionally true.

*b. If {**P**} is truth-functionally consistent, must {~ **P**} be truth-functionally consistent as well? Show that you are right.

c. If **P** and **Q** are truth-functionally indeterminate, does it follow that {**P**, **Q**} is truth-functionally consistent? Explain your answer.

*d. Prove that if **P** ≡ **Q** is truth-functionally true then {**P**, ~ **Q**} is truth-functionally inconsistent.

3.5 TRUTH-FUNCTIONAL ENTAILMENT AND TRUTH-FUNCTIONAL VALIDITY

Truth-functional entailment is a relation that may hold between a sentence of *SL* and a set of sentences of *SL*.

> A set Γ of sentences of *SL truth-functionally entails* a sentence **P** of *SL* if and only if there is no truth-value assignment on which every member of Γ is true and **P** is false.

In other words, Γ truth-functionally entails **P** just in case **P** is true on every truth-value assignment on which every member of Γ is true. We have a special symbol for truth-functional entailment: the double turnstile '⊨'. The expression

$$\Gamma \models \mathbf{P}$$

is read

Γ truth-functionally entails **P**.

To indicate that Γ does not truth-functionally entail **P**, we write

$$\Gamma \nvDash \mathbf{P}$$

Thus

$$\{A, B \,\&\, C\} \vDash B$$

and

$$\{A, B \vee C\} \nvDash B$$

mean, respectively,

{A, B & C} truth-functionally entails 'B'

and

{A, B ∨ C} does not truth-functionally entail 'B'.

Here we have adopted the convention that, when using the turnstile notation, we drop the single quotation marks around the sentence following the turnstile. We also have a special abbreviation to indicate that a sentence is truth-functionally entailed by the empty set of sentences:

$$\vDash \mathbf{P}$$

The expression '⊨ **P**' is an abbreviation for '∅ ⊨ **P**'. All and only truth-functionally true sentences are truth-functionally entailed by the empty set of sentences; the proof of this is left as an exercise in Section 3.6.

If Γ is a finite set, we can determine whether Γ truth-functionally entails a sentence **P** by constructing a truth-table for the members of Γ and for **P**. If there is a row in the truth-table in which all the members of Γ have the truth-value **T** and **P** has the truth-value **F**, then Γ does not truth-functionally entail **P**. If there is no such row, then Γ does truth-functionally entail **P**. We can establish that {A, B & C} ⊨ B by constructing the following truth-table:

			↓				↓
A	B	C	A	B	&	C	B
T	**T**	**T**	**T**	**T**	**T**	**T**	**T**
T	**T**	**F**	**T**	**T**	**F**	**F**	**T**
T	**F**	**T**	**T**	**F**	**F**	**T**	**F**
T	**F**	**F**	**T**	**F**	**F**	**F**	**F**
F	**T**	**T**	**F**	**T**	**T**	**T**	**T**
F	**T**	**F**	**F**	**T**	**F**	**F**	**T**
F	**F**	**T**	**F**	**F**	**F**	**T**	**F**
F	**F**	**F**	**F**	**F**	**F**	**F**	**F**

There is only one row in which both members of {A, B & C} are true, namely, the row in which 'A', 'B', and 'C' all have the truth-value **T**. But since 'B' is true in this row, it follows that there is no truth-value assignment on which 'A' and 'B & C' are true and 'B' is false. Hence {A, B & C} ⊨ B.

The following truth-table shows that {W ∨ J, (W ⊃ Z) ∨ (J ⊃ Z), ~ Z} ⊨ ~ (W & J):

					↓					↓				↓		↓				
J	W	Z	W	∨	J	(W	⊃	Z)	∨	(J	⊃	Z)	~	Z	~	(W	&	J)
T	**T**	**T**	**T**	**T**	**T**	**T**	**T**	**T**	**T**	**T**	**T**	**T**	**F**	**T**	**F**	**T**	**T**	**T**
T	**T**	**F**	**T**	**T**	**T**	**T**	**F**	**F**	**F**	**T**	**F**	**F**	**T**	**F**	**F**	**T**	**T**	**T**
T	**F**	**T**	**F**	**T**	**T**	**F**	**T**	**T**	**T**	**T**	**T**	**T**	**F**	**T**	**T**	**F**	**F**	**T**
T	**F**	**F**	**F**	**T**	**T**	**F**	**T**	**F**	**T**	**T**	**F**	**F**	**T**	**F**	**T**	**F**	**F**	**T**
F	**T**	**T**	**T**	**T**	**F**	**T**	**T**	**T**	**T**	**F**	**T**	**T**	**F**	**T**	**T**	**T**	**F**	**F**
F	**T**	**F**	**T**	**T**	**F**	**T**	**F**	**F**	**T**	**F**	**T**	**F**	**T**	**F**	**T**	**T**	**F**	**F**
F	**F**	**T**	**F**	**F**	**F**	**F**	**T**	**T**	**T**	**F**	**T**	**T**	**F**	**T**	**T**	**F**	**F**	**F**
F	**F**	**F**	**F**	**F**	**F**	**F**	**T**	**F**	**T**	**F**	**T**	**F**	**T**	**F**	**T**	**F**	**F**	**F**

The fourth and sixth rows are the only ones in which all the set members are true, and '~ (W & J)' is true in these rows as well. The following truth-table shows that {K ∨ J, ~ (K ∨ J)} ⊨ K:

					↓		↓				↓
J	K	K	∨	J	~	(K	∨	J)	K
T	**T**	**T**	**T**	**T**	**F**	**T**	**T**	**T**	**T**
T	**F**	**F**	**T**	**T**	**F**	**F**	**T**	**T**	**F**
F	**T**	**T**	**T**	**F**	**F**	**T**	**T**	**F**	**T**
F	**F**	**F**	**F**	**F**	**T**	**F**	**F**	**F**	**F**

There is no row in which 'K ∨ J' and '~ (K ∨ J)' are both true and hence no truth-value assignment on which the set members are both true. Consequently there is no truth-value assignment on which the members of the set are both true and 'K' is false; so the set truth-functionally entails 'K'.

On the other hand, {A, B ∨ C} does *not* truth-functionally entail 'B'. The following shortened truth-table shows this:

					↓		↓			↓
A	B	C	A		B	∨	C		B
T	**F**	**T**	**T**		**F**	**T**	**T**		**F**

This shortened truth-table shows that 'A' and 'B ∨ C' are both true and 'B' is false on any truth-value assignment that assigns **T** to 'A' and 'C' and **F** to 'B'.

An **argument** of *SL* is a set of two or more sentences of *SL*, one of which is designated as the conclusion and the others as the premises.

> An argument of *SL* is *truth-functionally valid* if and only if there is no truth-value assignment on which all the premises are true and the conclusion is false. An argument of *SL* is *truth-functionally invalid* if and only if it is not truth-functionally valid.

Put another way, an argument of *SL* is truth-functionally valid just in case the conclusion is true on every truth-value assignment on which all of the premises are true. This means that an argument is truth-functionally valid if and only if the set consisting of the premises of the argument truth-functionally entails the conclusion.

We can use full truth-tables to determine whether arguments with a finite number of premises are truth-functionally valid, and we can use shortened truth-tables to show truth-functionally invalid arguments with a finite number of premises are truth-functionally invalid. The argument

$$\begin{array}{l} \mathrm{F} \equiv \mathrm{G} \\ \mathrm{F} \vee \mathrm{G} \\ \hline \mathrm{F} \,\&\, \mathrm{G} \end{array}$$

is truth-functionally valid, as the following truth-table shows:

			↓			↓				↓	
F	G	F	≡	G	F	∨	G	F	&	G	
T	**T**	**T**	**T**	**T**	**T**	**T**	**T**	**T**	**T**	**T**	
T	**F**	**T**	**F**	**F**	**T**	**T**	**F**	**T**	**F**	**F**	
F	**T**	**F**	**F**	**T**	**F**	**T**	**T**	**F**	**F**	**T**	
F	**F**	**F**	**T**	**F**	**F**	**F**	**F**	**F**	**F**	**F**	

The first row displays the only combination of truth-values for the atomic components of these sentences for which the premises, 'F ≡ G' and 'F ∨ G', are both true, and the conclusion, 'F & G', is true in this row as well. Similarly, the argument

$$\begin{array}{l} (\mathrm{A} \,\&\, \mathrm{G}) \vee (\mathrm{B} \supset \mathrm{G}) \\ \sim \mathrm{G} \vee \mathrm{B} \\ \hline \sim \mathrm{B} \vee \mathrm{G} \end{array}$$

is truth-functionally valid, as the following truth-table establishes.

A	B	G	(A & G)	∨	(B ⊃ G)	~ G	∨	B	~ B	∨	G
				↓			↓			↓	
T	**T**	**T**	**T T T**	**T**	**T T T**	**F T**	**T**	**T**	**F T**	**T**	**T**
T	**T**	**F**	**T F F**	**F**	**T F F**	**T F**	**T**	**T**	**F T**	**F**	**F**
T	**F**	**T**	**T T T**	**T**	**F T T**	**F T**	**F**	**F**	**T F**	**T**	**T**
T	**F**	**F**	**T F F**	**T**	**F T F**	**T F**	**T**	**F**	**T F**	**T**	**F**
F	**T**	**T**	**F F T**	**T**	**T T T**	**F T**	**T**	**T**	**F T**	**T**	**T**
F	**T**	**F**	**F F F**	**F**	**T F F**	**T F**	**T**	**T**	**F T**	**F**	**F**
F	**F**	**T**	**F F T**	**T**	**F T T**	**F T**	**F**	**F**	**T F**	**T**	**T**
F	**F**	**F**	**F F F**	**T**	**F T F**	**T F**	**T**	**F**	**T F**	**T**	**F**

The conclusion, '~ B ∨ G', is true on every truth-value assignment on which the premises are true.

The following argument is truth-functionally invalid:

D ≡ (~ W ∨ G)
G ≡ ~ D

~ D

This is shown by the following truth-table:

D	G	W	D	≡	(~ W	∨	G)	G	≡	~ D	~ D
				↓					↓		↓
T	**T**	**T**	**T**	**T**	**F T**	**T**	**T**	**T**	**F**	**F T**	**F T**
T	**T**	**F**	**T**	**T**	**T F**	**T**	**T**	**T**	**F**	**F T**	**F T**
T	**F**	**T**	**T**	**F**	**F T**	**F**	**F**	**F**	**T**	**F T**	**F T**
T	**F**	**F**	**T**	**T**	**T F**	**T**	**F**	**F**	**T**	**F T**	**F T**
F	**T**	**T**	**F**	**F**	**F T**	**T**	**T**	**T**	**T**	**T F**	**T F**
F	**T**	**F**	**F**	**F**	**T F**	**T**	**T**	**T**	**T**	**T F**	**T F**
F	**F**	**T**	**F**	**T**	**F T**	**F**	**F**	**F**	**F**	**T F**	**T F**
F	**F**	**F**	**F**	**F**	**T F**	**T**	**F**	**F**	**F**	**T F**	**T F**

The premises, 'D ≡ (~ W ∨ G)' and 'G ≡ ~ D', are both true on every truth-value assignment that assigns **T** to 'D' and **F** to 'G' and 'W', and the conclusion, '~ D', is false on these truth-value assignments.

If an argument is truth-functionally invalid, we can show this by constructing a shortened truth-table that displays a row in which the premises are true and the conclusion false. The argument

~ (B ∨ D)
~ H

B

is truth-functionally invalid, as the following shortened truth-table shows:

				↓					↓		↓
B	D	H		~	(	B	∨	D)	~	H	B
F	**F**	**F**		**T**		**F**	**F**	**F**	**T**	**F**	**F**

There is an obvious relationship between validity and entailment: an argument of *SL* that has a finite number of premises is truth-functionally valid if and only if the set consisting of the premises of the argument truth-functionally entails the conclusion of the argument. There is also a relation between an argument of *SL* and a sentence called its *corresponding material conditional*, namely, the argument is truth-functionally valid if and only if its corresponding material conditional is truth-functionally true. To form an argument's corresponding material conditional, we first need the concept of an *iterated conjunction*. The iterated conjunction of a sentence **P** is just **P**, while the interated conjunction of sentences $\mathbf{P}_1$, $\mathbf{P}_2$, . . . , $\mathbf{P}_n$ is (. . . ($\mathbf{P}_1$ & $\mathbf{P}_2$) & . . . & $\mathbf{P}_n$). (We form a conjunction of the first sentence and the second sentence, then a conjunction of that conjunction and the third sentence, if any, and so on.) The *corresponding material conditional* for an argument of *SL* with a finite number of premises is the material conditional whose antecedent is the iterated conjunction of the argument's premises and whose consequent is the conclusion of the argument.[5] So the corresponding material conditional for the argument

$\mathbf{P}_1$
.
.
.
$\mathbf{P_n}$

$\mathbf{Q}$

is

(. . . ($\mathbf{P}_1$ & $\mathbf{P}_2$)& . . . & $\mathbf{P_n}$) ⊃ $\mathbf{Q}$

We will shortly prove that the argument is truth-functionally valid if and only if the corresponding material conditional is truth-functionally true, but first we will consider two examples.

[5]Strictly speaking, an argument with more than one premise will have more than one corresponding material conditional. This is because the premises of an argument can be conjoined in more than one order. But all the corresponding material conditionals for any one argument are truth-functionally equivalent, and so we speak loosely of *the* corresponding material conditional for a given argument.

We can show that the argument

$$\frac{\begin{array}{l}A\\A \supset B\end{array}}{B}$$

is truth-functionally valid by showing that the corresponding material conditional '[A & (A ⊃ B)] ⊃ B' is truth-functionally true:

							↓		
A	B	[A	&	(A	⊃	B)]	⊃	B
T	**T**	**T**	**T**	**T**	**T**	**T**	**T**	**T**
T	**F**	**T**	**F**	**T**	**F**	**F**	**T**	**F**
F	**T**	**F**	**F**	**F**	**T**	**T**	**T**	**T**
F	**F**	**F**	**F**	**F**	**T**	**F**	**T**	**F**

There is no truth-value assignment on which 'A & (A ⊃ B)' is true and 'B' is false, which means that there is no truth-value assignment on which 'A' and 'A ⊃ B' are both true and 'B' is false. And we can show that the argument

$$\frac{\begin{array}{l}\sim A \equiv \sim B\\B \vee A\end{array}}{\sim A}$$

is truth-functionally invalid by showing that the corresponding material conditional is not truth-functionally true. The following shortened truth-table shows this:

													↓			
A	B	((~	A	≡	~	B)	&	(B	∨	A))	⊃	~	A
T	**T**	**F**	**T**	**T**	**F**	**T**	**T**	**T**	**T**	**T**	**F**	**F**	**T**

The single row of this table represents truth-value assignments on which the antecedent is true and the consequent false. On these truth-value assignments the premises of the argument, '~ A ≡ ~ B' and 'B ∨ A', are both true and the conclusion, '~ A', is false. Hence the argument is truth-functionally invalid.

We now prove that an argument of *SL* with a finite number of premises is truth-functionally valid if and only if its corresponding material conditional is truth-functionally true.

Suppose that

$$\frac{\begin{array}{l}\mathbf{P}_1\\.\\.\\.\\\mathbf{P_n}\end{array}}{\mathbf{Q}}$$

is a truth-functionally valid argument of *SL*. Then there is no truth-value assignment on which $\mathbf{P}_1, \ldots, \mathbf{P_n}$ are all true and **Q** is false. Because the iterated conjunction $(\ldots(\mathbf{P}_1 \,\&\, \mathbf{P}_2) \,\&\, \ldots \mathbf{P_n})$ has the truth-value **T** on a truth-value assignment if and only if all of $\mathbf{P}_1, \ldots, \mathbf{P_n}$ have the truth-value **T** on that assignment, it follows that there is no truth-value assignment on which the antecedent of the corresponding material conditional, $(\ldots(\mathbf{P}_1 \,\&\, \mathbf{P}_2) \,\&\, \ldots \,\&\, \mathbf{P_n}) \supset \mathbf{Q}$, is true while the consequent is false. Thus, the material conditional is true on every truth-value assignment and is therefore truth-functionally true.

Now assume that $(\ldots(\mathbf{P}_1 \,\&\, \mathbf{P}_2) \,\&\, \ldots \,\&\, \mathbf{P_n}) \supset Q$ is truth-functionally true. Then there is no truth-value assignment on which the antecedent is true and the consequent false. But the iterated conjunction is true on a truth-value assignment if and only if the sentences $\mathbf{P}_1, \ldots, \mathbf{P_n}$ are all true. So there is no truth-value assignment on which $\mathbf{P}_1, \ldots, \mathbf{P_n}$ are all true and **Q** is false; hence the argument is truth-functionally valid.

3.5**E** EXERCISES

1. Construct truth-tables and state whether the following arguments are truth-functionally valid.

a. $A \supset (H \,\&\, J)$
$J \equiv H$
$\sim J$

$\sim A$

*b. $B \vee (A \,\&\, \sim C)$
$(C \supset A) \equiv B$
$\sim B \vee A$

$\sim (A \vee C)$

c. $(D \equiv \sim G) \,\&\, G$
$(G \vee [(A \supset D) \,\&\, A]) \supset \sim D$

$G \supset \sim D$

*d. $\sim (Y \equiv A)$
$\sim Y$
$\sim A$

$W \,\&\, \sim W$

e. $(C \supset D) \supset (D \supset E)$
D

$C \supset E$

*f. $B \vee B$

$[\sim B \supset (\sim D \vee \sim C)] \ \& \ [(\sim D \vee C) \vee (\sim B \vee C)]$

C

g. $(G \equiv H) \vee (\sim G \equiv H)$

$(\sim G \equiv \sim H) \vee \sim (G \equiv H)$

*h. $[(J \ \& \ T) \ \& \ Y] \vee (\sim J \supset \sim Y)$

$J \supset T$

$T \supset Y$

$Y \equiv T$

i. $\sim \sim F \supset \sim \sim G$

$\sim G \supset \sim F$

$G \supset F$

*j. $[A \ \& \ (B \vee C)] \equiv (A \vee B)$

$B \supset \sim B$

$C \vee A$

2. For each of the following arguments, either show that the argument is truth-functionally invalid by constructing an appropriate shortened truth-table or show that the argument is truth-functionally valid by constructing a full truth-table.

a. $(J \vee M) \supset \sim (J \ \& \ M)$

$M \equiv (M \supset J)$

$M \supset J$

*b. $B \ \& \ F$

$\sim (B \ \& \ G)$

G

c. $A \supset \sim A$

$(B \supset A) \supset B$

$A \equiv \sim B$

*d. $J \vee [M \supset (T \equiv J)]$

$(M \supset J) \ \& \ (T \supset M)$

$T \ \& \sim M$

e. $A \ \& \sim [(B \ \& \ C) \equiv (C \supset A)]$

$B \supset \sim B$

$\sim C \supset C$

3. Construct the corresponding material conditional for each of the following arguments. For each of the arguments, either show that the argument is truth-functionally invalid by constructing an appropriate shortened truth-table for the corresponding material conditional or show that the argument is truth-functionally valid by constructing a full truth-table for the corresponding material conditional.

a. $B \ \& \ C$

$\underline{\qquad\qquad}$

$B \vee C$

*b. $K \equiv L$

$L \supset J$

$\sim J$

$\underline{\qquad\qquad}$

$\sim K \vee L$

c. $(J \supset T) \supset J$

$\underline{(T \supset J) \supset T}$

$\sim J \vee \sim T$

*d. $\underline{(A \vee C) \ \& \sim H}$

$\sim C$

e. $B \ \& \ C$

$\underline{B \vee D}$

D

*f. $\sim [A \vee \sim (B \vee \sim C)]$

$\underline{B \supset (A \supset C)}$

$\sim A \equiv \sim B$

4. Symbolize each of the following arguments and use truth-tables to test for truth-functional validity. Use full truth-tables to establish truth-functional validity and shortened truth-tables to establish truth-functional invalidity.

a. 'Stern' means the same as 'star' if 'Nacht' means the same as 'day'. But 'Nacht' doesn't mean the same as 'day'; therefore 'Stern' means something different from 'star'.

*b. Many people believe that war is inevitable. But war is inevitable if and only if our planet's natural resources are nonrenewable. So many people believe that our natural resources are nonrenewable.

c. If Sophie is in her right mind she doesn't believe in trolls, and she is in her right mind. If Jason is in his right mind he doesn't believe in trolls, but he isn't in his right mind. So Sophie doesn't believe in trolls but Jason does.

d. Sophie doesn't believe in trolls, but she does believe in Bigfoot. Jason believes in both trolls and Bigfoot. If Sophie or Jason both believe in trolls, then neither is a critical thinker. Therefore, Sophie is a critical thinker but Jason isn't.

e. Computers can think if and only if they can have emotions. If computers can have emotions, then they can have desires as well. But computers can't think if they have desires. Therefore computers can't think.

*f. If the butler murdered Devon, then the maid is lying, and if the gardener murdered Devon, then the weapon was a slingshot. The maid is lying if and only if the weapon wasn't a slingshot, and if the weapon wasn't a slingshot, then the butler murdered Devon. Therefore the butler murdered Devon.

5. Where **P**, **Q**, and **R** are sentences of *SL*, prove each of the following.

*a. Show that {**P**} ⊨ **Q** and {**Q**} ⊨ **P** if and only if **P** and **Q** are truth-functionally equivalent.

b. Suppose that {**P**} ⊨ **Q** ∨ **R**. Does it follow that either {**P**} ⊨ **Q** or {**P**} ⊨ **R**? Show that you are right.

*c. Show that if {**P**} ⊨ **Q** and {**Q**} ⊨ **R**, then {**P**} ⊨ **R**.

3.6 TRUTH-FUNCTIONAL PROPERTIES AND TRUTH-FUNCTIONAL CONSISTENCY

In this section we show that the truth-functional concepts of truth-functional truth, truth-functional falsehood, truth-functional indeterminacy, truth-functional equivalence, truth-functional entailment, and truth-functional validity can all be explicated in terms of truth-functional consistency. This is important because in Chapter 4 we shall introduce an alternative test for truth-functional consistency, and the possibility of explicating the other concepts in terms of truth-functional consistency means that we shall be able to use the test we develop in Chapter 4 to determine whether other truth-functional properties of sentences and sets of sentences hold.

We will now state how each truth-functional concept other than consistency can be stated in terms of consistency, and prove each statement.

> A sentence **P** of *SL* is *truth-functionally false* if and only if {**P**} is truth-functionally inconsistent.

Proof: Assume that **P** is truth-functionally false. Then, by definition, there is no truth-value assignment on which **P** is true. Consequently, as **P** is the only member of the unit set {**P**}, there is no truth-value assignment on which every member of that set is true. So {**P**} is truth-functionally inconsistent. Now assume that {**P**} is truth-functionally inconsistent. Then, by definition, there is no truth-value assignment on which every member of {**P**} is true. Since **P** is the only member of its unit set, there is no truth-value assignment on which **P** is true. Hence **P** is truth-functionally false.

> A sentence **P** of *SL* is *truth-functionally true* if and only if {~ **P**} is truth-functionally inconsistent.

Proof: Assume that **P** is truth-functionally true. Then, by definition, **P** is true on every truth-value assignment. We know that a sentence is true on

a truth-value assignment if and only if the negation of the sentence is false on that truth-value assignment. So it follows from our assumption that ~ **P** is false on every truth-value assignment; that is, there is no truth-value assignment on which ~ **P** is true. But then there is no truth-value assignment on which every member of {~ **P**} is true, which means that {~ **P**} is truth-functionally inconsistent. The proof of the converse, that if {~ **P**} is truth-functionally inconsistent then **P** is truth-functionally true, is left as an exercise.

> A sentence **P** of *SL* is *truth-functionally indeterminate* if and only if both {~ **P**} and {**P**} are truth-functionally consistent.

Proof: A sentence **P** is truth-functionally indeterminate if and only if **P** is neither truth-functionally true nor truth-functionally false, and hence, by the previous results, if and only if both {~ **P**} and {**P**} are truth-functionally consistent.

> Sentences **P** and **Q** of *SL* are *truth-functionally equivalent* if and only if {~ (**P** ≡ **Q**)} is truth-functionally inconsistent.

Proof: Where **P** and **Q** are sentences of *SL*, **P** ≡ **Q** is their *corresponding material biconditional*. It is straightforward to show that **P** and **Q** are truth-functionally equivalent if and only if their corresponding material biconditional is truth-functionally true. Assume that **P** and **Q** are truth-functionally equivalent. Then, by definition, **P** and **Q** have the same truth-value on every truth-value assignment. We know that a material biconditional has the truth-value **T** on every truth-value assignment on which its immediate sentential components have the same truth-value. It follows that **P** ≡ **Q** is true on every truth-value assignment and hence is truth-functionally true and therefore, by the second result above, {~ (**P** ≡ **Q**)} is truth-functionally inconsistent. The proof of the converse, that if {~ (**P** ≡ **Q**)} is truth-functionally inconsistent, **P** and **Q** are truth-functionally equivalent, is left as an exercise.

To make these results more concrete, we shall consider an example. The set {~ [(A ∨ B) ≡ (~ A ⊃ B)]} is truth-functionally inconsistent, as shown by the following truth-table:

			↓								
A	B		~	[(A	∨	B)	≡	(~ A	⊃	B)]
T	**T**		**F**	**T**	**T**	**T**	**T**	**F**	**T**	**T**	**T**
T	**F**		**F**	**T**	**T**	**F**	**T**	**F**	**T**	**T**	**F**
F	**T**		**F**	**F**	**T**	**T**	**T**	**T**	**F**	**T**	**T**
F	**F**		**F**	**F**	**F**	**F**	**T**	**T**	**F**	**F**	**F**

The set is truth-functionally inconsistent because there is no truth-value assignment on which every member of the set (in this case there is just one member) is true. From this we know the following:

1. '~ [(A ∨ B) ≡ (~ A ⊃ B)]' is truth-functionally false. (**P** is truth-functionally false if and only if {**P**} is truth-functionally inconsistent. Here {~ [(A ∨ B) ≡ (~ A ⊃ B)]} is truth-functionally inconsistent. Hence there is no truth-value assignment on which the only member of that set, '~ [(A ∨ B) ≡ (~ A ⊃ B)]', is true. That one member is thus truth-functionally false.)

2. '(A ∨ B) ≡ (~ A ⊃ B)' is truth-functionally true. (**P** is truth-functionally true if and only if {~ **P**} is truth-functionally inconsistent. We have just reasoned that '~ [(A ∨ B) ≡ (~ A ⊃ B)]' is truth-functionally false. Hence the sentence of which it is the negation, '(A ∨ B) ≡ (~ A ⊃ B)', is true on every truth-value assignment—it is a truth-functionally true sentence.)
3. 'A ∨ B' and '~ A ⊃ B' are truth-functionally equivalent. (**P** and **Q** are truth-functionally equivalent if and only if {~ (**P** ≡ **Q**)} is truth-functionally inconsistent. Since '(A ∨ B) ≡ (~ A ⊃ B)' is truth-functionally true, 'A ∨ B' and '~ A ⊃ B' have the same truth-value on every truth-value assignment—they are truth-functionally equivalent.)

Of course, each of these claims can be directly verified by examining the truth-table, but our general proofs show that this is not necessary.

Next we relate the concepts of truth-functional entailment and truth-functional consistency. Where Γ is a set of sentences of *SL* and **P** is any sentence of *SL*, we may form a set that contains **P** and all the members of Γ. This set is represented as

$$\Gamma \cup \{\mathbf{P}\}$$

which is read as

> the union of gamma and the unit set of **P**

Thus, if Γ is {A, A ⊃ B} and **P** is 'J', then Γ ∪ {**P**}—that is, {A, A ⊃ B} ∪ {J}—is {A, A ⊃ B, J}. Of course, if **P** is a member of Γ, then Γ ∪ {**P**} is identical with Γ. So {A, A ⊃ B} ∪ {A ⊃ B} is simply {A, A ⊃ B}. In the case where Γ is ∅ (the empty set), Γ ∪ {**P**} is simply {**P**}. This follows because ∅ contains no members.

We can now explicate truth-functional entailment in terms of truth-functional inconsistency:

> A set Γ of sentences of *SL* truth-functionally entails a sentence **P** of *SL* if and only if Γ ∪ {~ **P**} is truth-functionally inconsistent.

Proof: Assume that Γ truth-functionally entails **P**. Then, by the definition of truth-functional entailment, there is no truth-value assignment on which all the members of Γ are true and **P** is false. We know that **P** is false on a truth-value assignment if and only if ~ **P** is true on that assignment, so it follows that there is no truth-value assignment on which all the members of Γ are true and ~ **P** is also true. Therefore, Γ ∪ {~ **P**} is truth-functionally inconsistent. The proof of the converse, that if Γ ∪ {~ **P**} is truth-functionally inconsistent then Γ ⊨ **P**, is left as an exercise.

An argument of *SL* is truth-functionally valid if and only if the set consisting of the premises of the argument and the negation of the conclusion of the argument is truth-functionally inconsistent.

Proof: This follows immediately from the previous result.

So the argument

$$\begin{array}{l} (A \supset D)\ \&\ H \\ F \vee H \\ \hline D \end{array}$$

is truth-functionally valid if and only if $\{(A \supset D)\ \&\ H, F \vee H, \sim D\}$ is truth-functionally inconsistent.

3.6**E** EXERCISES

1. Where **P** and **Q** are sentences of *SL* and Γ is a set of sentences of *SL*, prove each of the following:

a. If $\{\sim \mathbf{P}\}$ is truth-functionally inconsistent, then **P** is truth-functionally true.

*b. If $\mathbf{P} \equiv \mathbf{Q}$ is truth-functionally true, then **P** and **Q** are truth-functionally equivalent.

c. If $\Gamma \cup \{\sim \mathbf{P}\}$ is truth-functionally inconsistent, then $\Gamma \vDash \mathbf{P}$.

2. Where Γ is a set of sentences of *SL* and **P** and **Q** are sentences of *SL*, prove each of the following:

a. A sentence **P** is truth-functionally true if and only if $\varnothing \vDash \mathbf{P}$.

*b. $\Gamma \vDash \mathbf{P} \supset \mathbf{Q}$ if and only if $\Gamma \cup \{\mathbf{P}\} \vDash \mathbf{Q}$.

c. If Γ is truth-functionally inconsistent, then Γ truth-functionally entails every sentence of *SL*.

*d. For any set Γ of sentences of *SL* and any truth-functionally false sentence **P** of *SL*, $\Gamma \cup \{\mathbf{P}\}$ is truth-functionally inconsistent.

3. Where Γ is a set of sentences of *SL* and **P** and **Q** are sentences of *SL*, prove each of the following:

a. If Γ is truth-functionally consistent and **P** is truth-functionally true, then $\Gamma \cup \{\mathbf{P}\}$ is truth-functionally consistent.

*b. If $\Gamma \vDash \mathbf{P}$ and $\Gamma \vDash \sim \mathbf{P}$, then Γ is truth-functionally inconsistent.

4. Where Γ and Γ' are sets of sentences of *SL* and **P**, **Q** , and **R** are sentences of *SL*, prove each of the following:

a. If $\{\mathbf{P}\} \vDash \mathbf{Q}$ and $\{\sim \mathbf{P}\} \vDash \mathbf{R}$, then $\mathbf{Q} \vee \mathbf{R}$ is truth-functionally true.

*b. If **P** and **Q** are truth-functionally equivalent, then $\{\mathbf{P}\} \vDash \mathbf{R}$ if and only if $\{\mathbf{Q}\} \vDash \mathbf{R}$.

c. If $\Gamma \vDash \mathbf{P}$ and $\Gamma' \vDash \mathbf{Q}$, then $\Gamma \cup \Gamma' \vDash \mathbf{P}\ \&\ \mathbf{Q}$, where $\Gamma \cup \Gamma'$ is the set that contains all the sentences in Γ and all the sentences in Γ'.

GLOSSARY

TRUTH-FUNCTIONAL TRUTH: A sentence **P** of *SL* is *truth-functionally true* if and only if **P** is true on every truth-value assignment.

TRUTH-FUNCTIONAL FALSITY: A sentence **P** of *SL* is *truth-functionally false* if and only if **P** is false on every truth-value assignment.

TRUTH-FUNCTIONAL INDETERMINACY: A sentence **P** of *SL* is *truth-functionally indeterminate* if and only if **P** is neither truth-functionally true nor truth-functionally false.

TRUTH-FUNCTIONAL EQUIVALENCE: Sentences **P** and **Q** of *SL* are *truth-functionally equivalent* if and only if there is no truth-value assignment on which **P** and **Q** have different truth-values.

TRUTH-FUNCTIONAL CONSISTENCY: A set of sentences of *SL* is *truth-functionally consistent* if and only if there is at least one truth-value assignment on which all the members of the set are true. A set of sentences of *SL* is *truth-functionally inconsistent* if and only if the set is not truth-functionally consistent.

TRUTH-FUNCTIONAL ENTAILMENT: A set Γ of sentences of *SL truth-functionally entails* a sentence **P** of *SL* if and only if there is no truth-value assignment on which every member of Γ is true and **P** is false.

TRUTH-FUNCTIONAL VALIDITY: An argument of *SL* is *truth-functionally valid* if and only if there is no truth-value assignment on which all the premises are true and the conclusion is false. An argument of *SL* is *truth-functionally invalid* if and only if it is not truth-functionally valid.

Chapter 4

SENTENTIAL LOGIC: TRUTH-TREES

In Section 4.1 we introduce truth-trees, show how they are constructed, and show that they can be used to test a set of sentences of *SL* for truth-functional consistency. In Section 4.2 we lay out how truth-trees can also be used to test for truth-functional truth, falsity, and indeterminacy as well as truth-functional equivalence, entailment, and validity.

4.1 THE TRUTH-TREE METHOD

In Chapter 3 we used the notion of a truth-value assignment to give formal accounts of the important semantic concepts of truth-functional logic. At the end of Chapter 3, we saw that, once truth-functional consistency has been defined based on the concept of a truth-value assignment, the remaining semantic concepts of sentential logic can be explicated in terms of truth-functional consistency. In this chapter we make use of this fact to provide an additional method, the **truth-tree** method, of determining whether truth-functional properties hold for sentences and sets of sentences of *SL*. Truth-trees provide a systematic method of searching for truth-value assignments that are of special interest—for example, a truth-value assignment on which a given sentence of *SL*

is false, or a truth-value assignment on which the premises of a given argument of *SL* are true and the conclusion false. The truth-tree method also reveals when no such truth-value assignments exist.

The truth-table method is mechanical. And the truth-tree method we develop in this chapter can easily be made so. The advantage of truth-tables is that they graphically display how the truth-values of truth-functionally compound sentences are generated from the truth-values of their components. The disadvantage of truth-tables is that they become unwieldy when the number of distinct atomic components of the sentence or sentences being tested is much greater than 3. Truth-trees, it must be admitted, can also become unwieldy. However, the size and complexity of truth-trees are not as direct a function of the number of distinct atomic components of the sentences being tested as are the size and complexity of truth-tables. Sets of sentences with a large number of distinct atomic components frequently have reasonably concise truth-trees. What is of theoretical importance here, as with truth-tables, is that the truth-tree system can be used, for any finite set of sentences of *SL,* to yield, in a finite number of steps, an answer to the question 'Is this set truth-functionally consistent?' We establish this claim in Chapter 11.

The rules we will use in constructing truth-trees are derived directly from the characteristic truth-tables for the five truth-functional connectives. For this reason, and because truth-value assignments on which all the members of the set being tested are true can readily be recovered from truth-trees for consistent sets, we take truth-trees to constitute a second semantic method of determining whether the truth-functional properties defined in Chapter 3 hold for sentences and finite sets of sentences of *SL*.

It is obvious that if a set of sentences of *SL* is truth-functionally consistent there must be a truth-value assignment on which all the atomic sentences in the set are assigned the truth-value **T** and all the atomic sentences whose negations are members of the set are assigned the truth-value **F**. We will call atomic sentences and their negations '**literals**'**.** In this section we present the rules we will use to decompose nonliteral sentences of *SL*. By using these rules we will be able to determine which atomic components of these nonliterals must be assigned the truth-value **T** and which must be assigned the truth-value **F** if those nonliterals are to be true on a truth-value assignment.

We divide the nonliteral sentences of *SL* into nine groups, one group for each kind of binary compound (conjunctions, disjunctions, material conditionals, and material biconditionals) and one group for each kind of negation (negated negations, negated conjunctions, negated disjunctions, negated material conditionals, and negated material biconditionals). Here are truth-table templates for these nine kinds of sentences. Note that the templates for the sentence types on the left have a **T** under the sentences's main connective in *exactly one row,* while the templates on the right have *more than one* **T** under the main connective.

Group 1

Conjunction

P	Q	P & Q
T	T	Ⓣ
T	F	F
F	T	F
F	F	F

Negated Disjunction

P	Q	~(P ∨ Q)	
T	T	F	T
T	F	F	T
F	T	F	T
F	F	Ⓣ	F

Negated Material Conditional

P	Q	~ (P ⊃ Q)	
T	T	F	T
T	F	Ⓣ	F
F	T	F	T
F	F	F	T

Negated Negation

P	~ ~ P	
T	Ⓣ	F
F	F	T

Group 2

Negated Conjunction

P	Q	~ (P & Q)	
T	T	F	T
T	F	Ⓣ	F
F	T	Ⓣ	F
F	F	Ⓣ	F

Disjunction

P	Q	(P ∨ Q)
T	T	Ⓣ
T	F	Ⓣ
F	T	Ⓣ
F	F	F

Material Conditional

P	Q	(P ⊃ Q)
T	T	Ⓣ
T	F	F
F	T	Ⓣ
F	F	Ⓣ

Material Biconditional

P	Q	(P ⊃ Q)
T	T	Ⓣ
T	F	F
F	T	F
F	F	Ⓣ

Negated Material Biconditional

P	Q	~ (P ≡ Q)	
T	T	F	T
T	F	Ⓣ	F
F	T	Ⓣ	F
F	F	F	T

Group 1 sentences are such that they are true on only one combination of truth-values displayed to the left of the vertical line in their truth-table templates. That is, if we know that a sentence of the form (**P** & **Q**) or ~ (**P** ∨ **Q**) or ~ (**P** ⊃ **Q**) is true then we also know the truth-value of the components **P** and **Q**. And if we know that a sentence of the form ~ ~ **P** is true we also know the

truth-value of **P**. In contrast, if a Group 2 sentence is true, there are multiple combinations of truth-values that **P** and **Q** can have. Note that the schema for **P** & **Q**, **P** ∨ **Q**, **P** ⊃ **Q** and their respective negations are paired in the sense that one member of each pair is in Group 1 and the other in Group 2. However both **P** ≡ **Q** and ~ (**P** ≡ **Q**) are in Group 2.

The decomposition rules for Group 1 sentences require that these sentences be **decomposed** to sentences whose truth is required for the truth of the sentences being decomposed, as specified by the foregoing truth-table templates. Here are the rules for decomposing Group 1 sentences:

Negated Negation Decomposition (~ ~ D)

~ ~ P
P

Conjunction Decomposition (& D)

P & Q
P
Q

Negated Disjunction Decomposition (~ ∨D)

~ (P ∨ Q)
~ P
~ Q

Negated Conditional Decomposition (~ ⊃D)

~ (P ⊃ Q)
P
~ Q

A sentence of the form ~ ~ **P** is true on a truth-value assignment if and only if **P** is also true on that assignment, so the answer to the question 'What sentence(s) have to be true for ~ ~ **P** to be true?' is **P**, and hence we decompose ~ ~ **P** to **P**. Similarly, a sentence of the form **P** & **Q** is true on a truth-value assignment if and only if **P** and **Q** are both true on that assignment, so we decompose **P** & **Q** to **P** and to **Q**. And a sentence of the form ~ (**P** ∨ **Q**) is true on a truth-value assignment if and only if both ~ **P** and ~ **Q** are true on that assignment. So we decompose ~ (**P** ∨ **Q**) to ~ **P** and to ~ **Q**. (Recall that a sentence of the form ~ (**P** ∨ **Q**) is equivalent to the corresponding sentence of the form ~ **P** & ~ **Q**.) Finally, a sentence of the form ~ (**P** ⊃ **Q**) is true on a truth-value assignment if and only if its antecedent, **P**, is true on that assignment and its consequent, **Q**, is false on that assignment, that is, if and only if **P** and ~ **Q** are both true on that assignment. So we decompose ~ (**P** ⊃ **Q**) to **P** and to ~ **Q**.

Before presenting the rules for Group 2 sentences, we will use the Group 1 rules to construct two **truth-trees**[1]. Suppose we want to determine whether the set { ~ ~ B, C, ~ A, ~ (B ⊃ C)} is truth-functionally consistent. We begin by listing the members of the set one below the other in the middle of our work area. We annotate the tree by placing line numbers on

[1]Truth-trees, unlike real trees, grow from the top down, not from the bottom up.

the left. On the right we enter the notation 'SM' (for 'Set Member') in a justification column:

1	~ ~ B	SM
2	C	SM
3	~ A	SM
4	~ (B ⊃ C)	SM

The justification column indicates the reason that a sentence has been added to the tree. The sentences on lines 2 and 3 are literals and do not need to be decomposed. The sentence on line 1 is a negated negation, and the rule for decomposing a negated negation ~ ~ **P** instructs us to add **P** to our tree and to 'check off' the sentence ~ ~ **P**, indicating that it has been decomposed. So we continue the tree as follows:

1	~ ~ B ✓	SM
2	C	SM
3	~ A	SM
4	~ (B ⊃ C)	SM
5	B	1 ~ ~ D

Our justification for entering 'B' at line 5 is that it is the result of decomposing '~ ~ B' on line 1 using Negated Negation Decomposition. To complete our tree, we need to decompose the negated material conditional on line 4. The rule Negated Material Conditional Decomposition calls for entering the antecedent of the material conditional and the negation of its consequent on our tree:

1	~ ~ B ✓	SM
2	C	SM
3	~ A	SM
4	~ (B ⊃ C) ✓	SM
5	B	1 ~ ~D
6	B	4 ~ ⊃D
7	~ C	4 ~ ⊃D
	×	

Notice that lines 6 and 7 are both justified by Negated Material Conditional Decomposition, as that rule requires entering two sentences on our tree. The tree now contains only literals and check-off nonliterals, so we have decomposed every sentence that can be decomposed. This is indicated by the fact that every nonliteral on the tree has been checked off.

Our tree shows that the set we are testing is truth-functionally inconsistent. Each decomposition rule specifies the sentence(s) that must be true if the sentence being decomposed is true. That is, if the set members are all true, so is every other sentence on the tree. In particular, all of the literals on the tree must be true. So if the set { ~ ~ B, C, ~ A, ~ (B ⊃ C)} is truth-functionally

consistent, then there must be a truth-value assignment on which 'C', '~ A', 'B', and '~ C' are all true. But of course there can be no such assignment, because 'C' and '~ C' cannot both be true on the same truth-value assignment.

Where **P** is any atomic sentence of *SL*, we call the literals **P** and ~ **P** (such as 'C' and '~ C') '**contradictory literals**'. Contradictory literals cannot both be true on a truth-value assignment, so we used their presence to conclude that the set we were testing is truth-functionally inconsistent. We have placed an '×' at the bottom of our tree to indicate that a truth-value assignment *cannot be recovered* from this tree.

We will now test the set {(D & ~ A), ~ (B ∨ A), ~ ~ D, ~ (~ A ⊃ B)} for truth-functional consistency. We begin, as always, by listing the members of our set, with line numbers on the left and the annotation 'SM' on the right:

1	D & ~ A	SM
2	~ (B ∨ A)	SM
3	~ ~ D	SM
4	~ (~ A ⊃ B)	SM

Note that our set contains no literals. Every member of the set will have to be decomposed. We can use Ampersand Decomposition to decompose the sentence on line 1, which results in adding two sentences to our tree:

1	D & ~ A ✓	SM
2	~ (B ∨ A)	SM
3	~ ~ D	SM
4	~ (~ A ⊃ B)	SM
5	D	1 &D
6	~ A	1 &D

Our tree now contains three sentences that have not yet been decomposed and are not literals. The sentence on line 2 is a negated disjunction. Negated Disjunction Decomposition specifies that we add the negations of the two disjuncts to the tree:

1	D & ~ A ✓	SM
2	~ (B ∨ A) ✓	SM
3	~ ~ D	SM
4	~ (~ A ⊃ B)	SM
5	D	1 &D
6	~ A	1 &D
7	~ B	2 ~ ∨D
8	~ A	2 ~ ∨D

Our tree now contains three distinct literals, 'D', '~ A', and '~ B'. So far so good: we can make all of these literals true, by assigning the truth-value **T** to 'D' and the truth-value **F** to 'A' and to 'B'. But we still have two sentences

to decompose. The sentence on line 3 is a negated negation. Decomposing it will result in adding 'D' to our tree. The sentence on line 4 is a negated material conditional. Decomposing it will result in adding both '~ A' and '~ B' to our tree:

1	D & ~ A ✓	SM
2	~ (B ∨ A) ✓	SM
3	~ ~ D ✓	SM
4	~ (~ A ⊃ B) ✓	SM
5	D	1 &D
6	~ A	1 &D
7	~ B	2 ~ ∨D
8	~ A	2 ~ ∨D
9	D	3 ~ ~D
10	~ A	4 ~ ⊃D
11	~ B	4 ~ ⊃D
	o	

Every sentence on our tree is now either a literal or a checked-off nonliteral, so the tree is complete. Note that several times (on lines 8, 9, 10, and 11) we added a literal to our tree even though it already occurred on the tree. While these duplicate literals add no new information to the tree, the decomposition rules require adding the results of each decomposition, *even if* those results already occur on the tree.

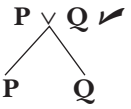
We have placed a lower case 'o' at the bottom of our tree to indicate that the tree is complete and is **open**, that is, it *does not* contain contradictory literals. This tree shows that that the set we are testing is truth-functionally consistent. Moreover, we can use the literals on the tree to **recover** a set of truth-value assignments on which all the sentences in the set we are testing are true. The literals on the tree and consequently every member of our set will be true on every truth-value assignment that makes the following assignments to 'A', 'B', and 'D':

A	B	D
F	**F**	**T**

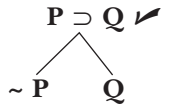
So far we have laid out the rules for decomposing *SL* sentences that are not literals and that, when decomposed, yield one or two sentences that are entered one below the other on our tree (because all of these sentences have to be true for the decomposed sentences to be true). But if we are to be able to construct trees for all finite *SL* sets we will need more rules than we have so far.

The rules for Group 1 sentences are called 'non-branching rules' because they do not introduce new branches to a truth-tree. The decomposition rules for Group 2 sentences do add branches and are accordingly called 'branching rules'. Here are the rules for decomposing disjunctions and material conditionals:

Disjunction Decomposition (∨D)



Conditional Decomposition (⊃ D)

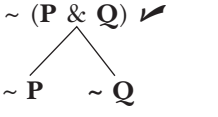


A disjunction is true if either disjunct is true. So our rule for decomposing disjunctions branches to represent these possibilities. It might appear that the rule for decomposing disjunctions should be more complicated, as a disjunction is true when *both* disjuncts are true as well as when one disjunct, but not the other, is true. However, the rule is correct as given. Neither the branch with **P** nor the branch with **Q** requires that the other disjunct be false. That is, the rule does not require us to add ~ **Q** to the left branch or ~ **P** to the right branch.

Our rule for decomposing material conditionals reflects the fact that if a material conditional is true, then either its antecedent is false or its consequent is true; and similar to the rule for disjunction, it allows for both possibilities. Keeping in mind that sentences of the form **P** ⊃ **Q** are equivalent to sentences of the form **P** ∨ **Q** may make it easier to remember this rule.

The negation of a conjunction is true if and only if at least one of the conjunction's conjuncts is false. Hence the rule

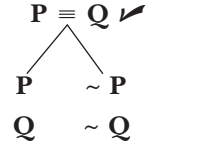
Negated Conjunction Decomposition (~ & D)



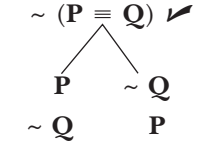
Keeping in mind that sentences of the form ~ (**P** & **Q**) are equivalent to sentences of the form ~ **P** ∨ ~ **Q** may make it easier to remember this rule.

A material biconditional is true if and only if both of its immediate components are true or both are false, and a negated material biconditional is true if and only if its immediate components have different truth-values. So the relevant decomposition rules are

Biconditional Decomposition (~ ≡ D)

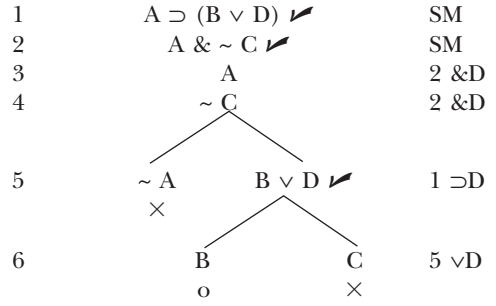


Negated Biconditional Decomposition (≡D)



We now have all the rules needed to construct truth-trees for any finite set of sentences of *SL*.

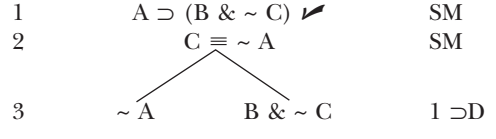
The following tree for the set {A ⊃ (B ∨ C), A & ~ C} uses two branching rules, Conditional Decomposition and Disjunction Decomposition:



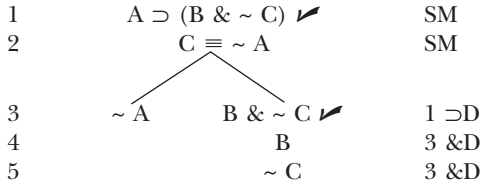
We start by decomposing the sentence on line 2, 'A & ~ C', using Conjunction Decomposition and entering the results, 'A' and '~ C,' on lines 3 and 4, respectively. We next use Conditional Decomposition to decompose 'A ⊃ (B ∨ C)'. The result is a branch on the left ending in '~ A' and one on the right containing 'B ∨ C'. The left branch closes immediately because contradictory literals, 'A' and '~ A' occur on that branch. The right branch remains open as of line 5. We next decompose 'B ∨ C', using the branching rule Disjunction Decomposition and entering 'B' on the left and 'C' on the right. The branch ending in 'C' closes, as it contains the contradictory literals '~ C' (at line 4) and 'C' (at line 6). The branch ending in 'B' remains open. As all non-literals on this branch have been decomposed we enter an 'o' below 'B' to indicate that we have a completed open branch. From that branch we can recover a set of truth-value assignments on which every member of the set {A ⊃ (B ∨ C), A & ~ C} is true, namely the set of truth-value assignments that assign **T** to 'A' and to 'B' and **F** to 'C'.

We note that the sentences on lines 1-4, which constitute the trunk of the tree, occur on all three branches of the tree. 'B ∨ C', which occurs on line 5, occurs on both the middle branch and the right-hand branch. 'B' and 'C', which occur on line 6, each occur on only one branch, 'B' on the middle branch and 'C' on the right-hand branch.

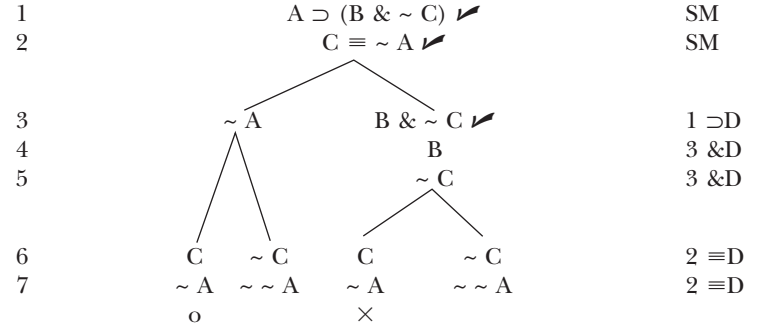
We will next construct a truth-tree for the set {A ⊃ (B & ~ C), C ≡ ~ A}. This tree will also have multiple branches. After listing the members of the set, we decompose 'A ⊃ (B & ~ C)', a material conditional, at line 3. The rule for decomposing material conditionals is a branching rule, so we enter '~ A' to the left and 'B & ~ C' to the right.



'B & ~ C' occurs on the right branch, but not on the left branch. Accordingly, we enter the results of decomposing it *only on* the right branch.



'C ≡ ~ A' occurs on both branches (both branches pass through it), so the results of decomposing 'C ≡ ~ A' must be entered on both branches. Because the rule for decomposing material biconditionals is a branching rule, the resulting tree has *four* branches:

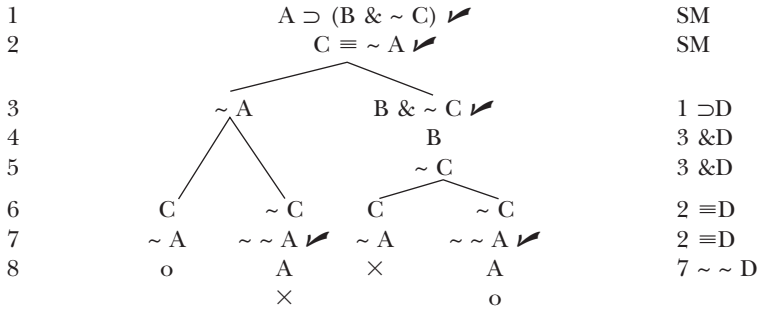


One of the resulting four branches, the third from the left, contains a literal and its negation ('C' and '~ C'), so we put an '×' below that branch. The only sentences on the leftmost branch are the literals '~ A' and 'C' and nonliteral sentences that have been checked off. Therefore, all the sentences on this branch that can be decomposed have been decomposed. This branch is open, that is, it does not contain contradictory literals, and we have indicated this by placing a lower case 'o' below the branch. The fact that this branch is open demonstrates that our set is consistent. From this branch we can recover two sets of truth-value assignments on which every member of the set we are testing is true. This is because while both '~ A' and 'C' occur on the branch, neither 'B' nor '~ B' does. The significance of this is that so long as 'A' is assigned the truth-value **F** (because '~ A' occurs on the branch) and 'C' is assigned the truth-value **T** (because 'C' occurs on the branch), every member of the set we are testing will be true, no matter what truth-value is assigned to 'B'. The two sets of truth-value assignments we can recover are

A	B	C
F	**T**	**T**
F	**F**	**T**

We have shown that the set we are testing is consistent, even though our truth-tree is not complete. '~ ~ A' occurs on the second branch from the left and on the rightmost branch, and this sentence has not been decomposed. If all we want to know is whether the set we are testing is truth-functionally consistent we can stop at this point, for we have shown that it is. Completing the tree will give

us additional information, which we may or may not be interested in, namely, completing the tree will show whether there are additional sets of truth-value assignments on which every member of the set is true. We complete the tree by decomposing '~ ~ A' to 'A', entering 'A' on each of the open branches.



The second branch from the left now contains contradictory literals ('A' and '~ A'), so we have placed an '×' below that branch. The rightmost branch contains only literals and checked-off nonliteral sentences, and it does not contain contradictory literals, so we place an 'o' below the branch. This branch reveals that every member of our set will be true on every truth-value assignment that assigns the following values to 'A', 'B', and 'C':

A	B	C
T	**T**	**F**

We have recovered an additional set of truth-value assignments on which all the members of the set we are testing are true. Note that we decomposed both occurrences of '~ ~ A' in one step, entering the results of both decompositions on line 8. We were able to do this because the two instances of '~ ~ A' occur on the same line and because we use the same rule to decompose them.

It is time to formally define the terms associated with truth-trees:

Closed branch: A branch on which contradictory literals occur.
Open branch: A branch that is not closed.
Completed open branch: An open branch on which every sentence is either a literal or has been decomposed (checked off).
Closed truth-tree: A truth-tree each of whose branches is closed.
Open truth-tree: A truth-tree that is not closed.
Completed truth-tree: A truth-tree each of whose branches is either closed or a completed open branch.

A finite set Γ of sentences of *SL* is *truth-functionally consistent* if and only if Γ has a truth-tree with at least one completed open branch.

A finite set Γ of sentences of *SL* of *SL* is *truth-functionally inconsistent* if and only if Γ has a closed truth-tree.

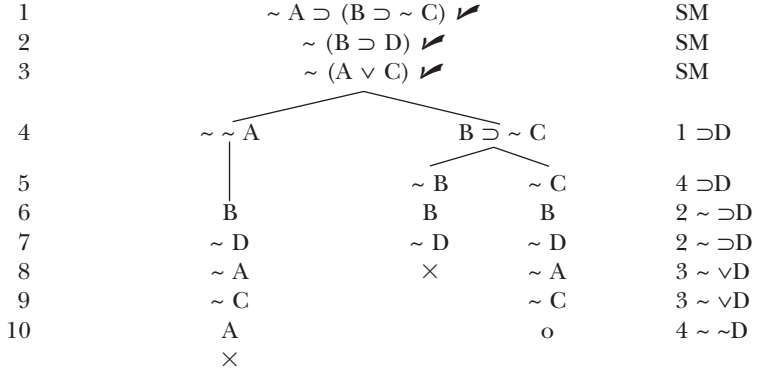
The construction of truth-trees is not, as we have developed it, a mechanical process, though it could be made mechanical by mandating the order in which nonliteral sentences are to be decomposed. Because we have not mandated such an order a single set of sentences of *SL* can have many distinct truth-trees, each constructed in accordance with the rules we have given. However, all completed truth-trees for a given set will yield the same result: either all the trees will be closed (showing that the set is truth-functionally inconsistent) or all the trees will have at least one completed open branch. In the latter case, all of the trees will yield the same sets of truth-value assignments on which all the members of the set are true.

For practical purposes, we would like the truth-trees we construct to be as concise and clear as possible. Adhering to the following guidelines will help produce such trees.

1. Stop when a tree yields the answer to the question being asked.
2. Give priority to decomposing sentences whose decomposition does not require branching.
3. Give priority to decomposing sentences whose decompositions result in the closing of one or more branches.

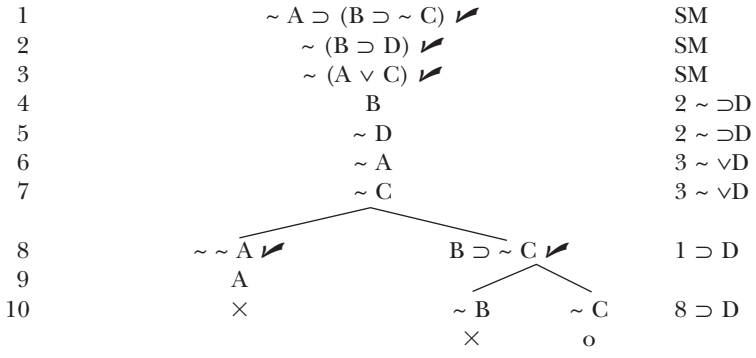
For example, the last tree we constructed had a completed open branch as of line 7. At that point we knew the set we were testing is truth-functionally consistent. So we could have stopped there if that was what we wanted to know. Of course, if the question we wanted to answer had been 'What are all of the truth-value assignments on which the m embers of this set are true?' we would have had to complete the tree to recover all of the truth-value assignments.

The following truth-tree is not constructed in accordance with our second guideline:



Decomposing the sentence on line 1, a material conditional, requires the use of a branching rule and the result is a tree with two open branches as of line 4. Decomposing 'B ⊃ ~ C' on line 4 adds an additional branch to the tree. A more concise tree can be constructed by decomposing the

sentences on lines 2 and 3 first, for their decomposition does not add new branches to the tree:



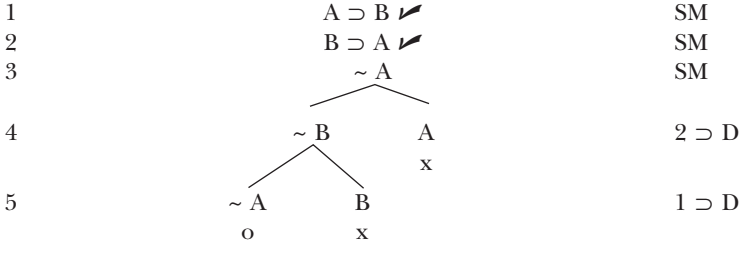
The same set of truth-value assignments can be recovered from both trees, namely, the set of truth-value assignments that assign the following truth-values to the four sentence letters:

A	B	C	D
F	**T**	**F**	**F**

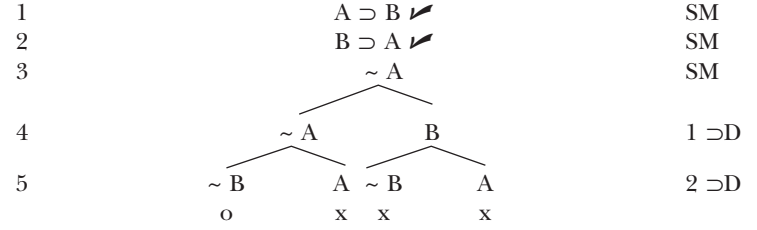
To illustrate the value of the third guideline, suppose that we want to show that the set {A ⊃ B, B ⊃ A, ~ A} is truth-functionally consistent. We first list the members of the set:

1	A ⊃ B	SM
2	B ⊃ A	SM
3	~ A	SM

We now have to decide which nonliteral to decompose first, the one on line 1 or the one on line 2. Both sentences are material conditionals, so both decompositions will produce a tree with two branches. A little thought will show that it is better to decompose the sentence on line 2 first, for doing so will produce a tree with '~ B' on the left branch and 'A' on the right branch, and the right branch will close, leaving only one open branch. Decomposing 'A ⊃ B' first will produce a tree with '~ A' on the left branch and 'B' on the right branch, and both will remain open. Here are the two resulting trees:



This tree follows our third guideline. The following tree does not do so and is somewhat more complex than is our first tree.



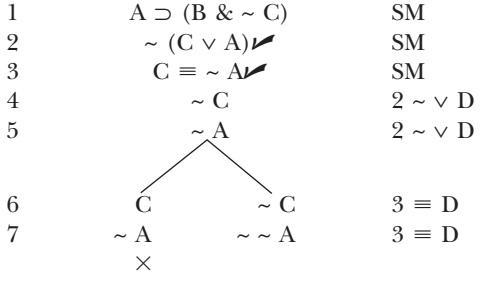
Both trees have a completed open branch, so the set we are testing is truth-functionally consistent. And the trees yield the same set of truth-value assignments:

A	B
F	**F**

Next consider the set {A ⊃ (B & ~ C), ~ (C ∨ A), C ≡ ~ A}. In constructing a truth-tree for this set, we start by listing the members of the set, one below the other:

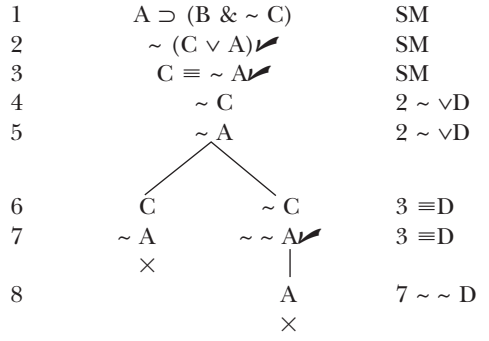
1	A ⊃ (B & ~ C)	SM
2	~ (C ∨ A)	SM
3	C ≡ ~ A	SM

No member of the set is a literal; hence all are candidates for decomposition. The sentence on line 1, a material conditional, will branch when decomposed. So will the sentence on line 3, a material biconditional. But the sentence on line 2, a negated disjunction, will not branch, so we decompose it first. This leaves two undecomposed sentences on the tree that are not literals, 'A ⊃ (B & ~ C)' and 'C ≡ ~ A', both of which will branch when decomposed. Decomposing the material conditional will yield two open branches, one ending in '~ A' and the other in 'B & ~ C'. Neither of these branches will close immediately. However, decomposing the material biconditional yields an immediate branch closure:



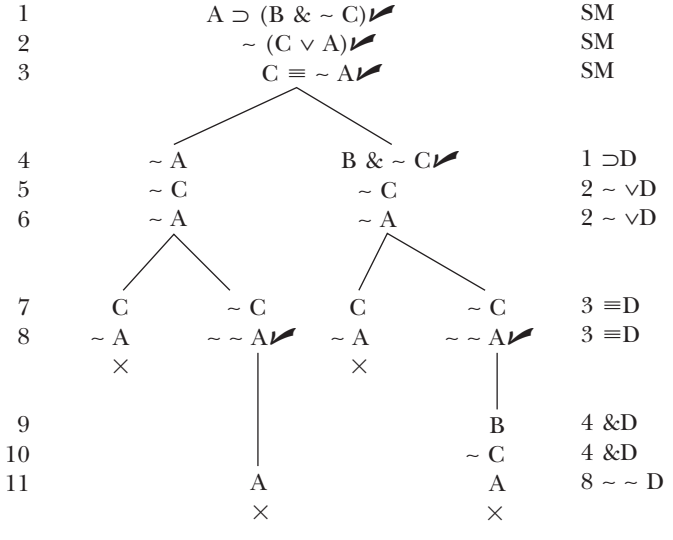
We are now left with just one open branch. There are two undecomposed nonliterals on that branch, 'A ⊃ (B & ~ C)' and '~ ~ A'. We decompose '~ ~ A' first, since negated negations do not branch when decomposed. Moreover, when we

decompose '~ ~ A', we add 'A' to the one open branch of our tree and thus close that branch and the tree:



Several aspects of this tree are of interest. First, the tree is closed, and the set we are testing, {A ⊃ (B & ~ C), ~ (C ∨ A), C ≡ ~ A}, is therefore truth-functionally inconsistent. Every attempt to find a set of truth-value assignments on which every member of that set is true ended in failure. Second, we have shown that the set is inconsistent without decomposing every nonliteral on the tree. The sentence on line 1 was never decomposed, since decomposing the other nonliterals on the tree generated a closed tree. What this shows is that the set we are testing would be inconsistent even without its first member, 'A ⊃ (B & ~ C)'. Whenever a branch closes we are through with that branch, even if it contains one or more undecomposed nonliterals.

This fairly concise tree was generated by following our strategies of giving priority to sentences whose decomposition does not require branching and to sentences whose decomposition generates one or more closed branches. Had we ignored these strategies and simply worked straight down the tree, always decomposing every nonliteral on a given level before moving to a lower level, the result would have been the following more complex tree:



Here we have four branches, whereas in the earlier tree we had only two. Moreover this tree takes eleven lines to construct; the earlier one took only eight. But the difference between the trees is only one of complexity. Each tree shows equally well that the set of sentences we are testing is truth-functionally inconsistent, for each tree is closed.

The last of the preceding trees can be used to illustrate two important aspects of tree construction. Note that when we decompose '~ (C ∨ A)' at lines 5 and 6, the tree already has two open branches. Hence the results of decomposing this sentence must be entered on each of these open branches. The results of decomposing a sentence must always be entered on *every open branch that runs through the sentence being decomposed.*

Consider the tree after line 8 is completed: Two branches are closed, but two are open. We next decompose 'B & ~ C' on the right-hand branch only, at lines 9 and 10 (not on the left-hand branch because, although it is open, it does not go through 'B & ~ C'). We then decompose each occurrence of '~ ~ A' on line 8 by writing 'A' on line 11, at the end of each branch (since each branch does go through '~ ~ A'). Why didn't we put 'A' on the left-hand branch at line 9 at the same time that we put 'B' on the right-hand branch at line 9? It is because the policy we follow is this: *Trees are to be so constructed that every line of the tree is fully justified either by writing 'SM' in the justification column or by entering the number of one and only one earlier line and one and only one rule abbreviation in the justification column.* All the entries made on line 7 come from line 3, and they are all obtained by one rule, Material Biconditional Decomposition. Had we tried to write 'A' at line 9 on the second branch from the left, we would have had two entries on that line coming from two different lines, by the use of two different rules, and thus would have been forced to enter both '8 ~ ~ D' and '4 &D' in the justification column for line 9, in clear violation of our policy.

We have so far specified three guidelines for keeping truth-trees concise. We repeat them here and add a fourth:

Guidelines for Constructing Truth-Trees

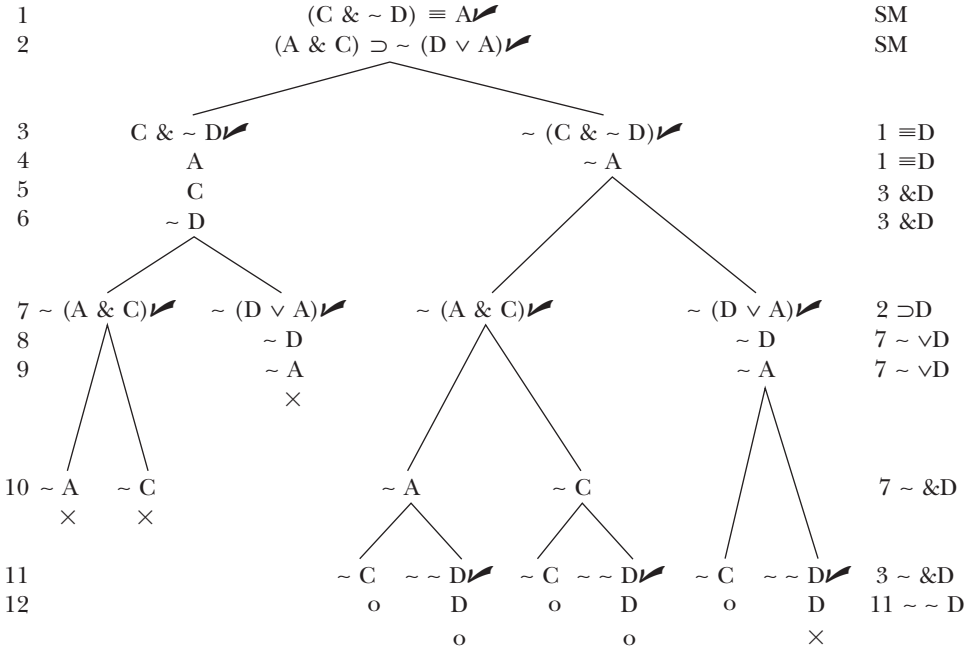
1. Stop when a tree yields the answer to the question being asked.
2. Give priority to decomposing sentences whose decomposition does not require branching.
3. Give priority to decomposing sentences whose decompositions result in the closing of one or more branches.
4. When guidelines 1–3 are not applicable, decompose the more complex sentences first.

The rationale for the first three strategies should be clear by now. The fourth strategy is designed to save tedious work, for a complex sentence takes more work to decompose than does a less complex one. Moreover, if a complex

sentence is decomposed early in a tree, chances are there will be only a few open branches on which the results must be entered. If complex sentences are left until the end, it is likely that the results of decomposing them will have to be entered on many open branches. Roughly speaking, a sentence **P** is more complex than a sentence **Q** if decomposing **P** requires entering more sentences or longer sentences on a tree than does decomposing **Q**. In this sense longer sentences are generally more complex than shorter ones, and material biconditionals and negations of material biconditionals are more complex than other sentences of approximately the same length.

The guidelines we have presented are just that, guidelines, not rules, for constructing truth-trees. Disregarding one or more of them may produce a more complex tree than is necessary but will never yield a completed open branch where following them would yield a closed tree, or vice versa.

As a final example we construct a truth-tree for {(C & ~ D) ≡ A, (A & C) ⊃ ~ (D ∨ A)}:



This tree has five completed open branches. The literals occurring on the left-most completed open branch are '~ C' and '~ A'. Hence this branch tells us that, to make all the sentences in the set we are testing true, it is sufficient to make '~ C' and '~ A' both true, and, to do this, we must assign the truth-value **F** to both 'A' and 'C'. But note that 'D' is also an atomic component of both members of that set. No assignment has yet been made to 'D' because neither 'D' nor '~ D' occurs on the open branch just examined. The significance of

the nonoccurrence of both 'D' and '~ D' is this: It does not matter which truth-value we assign to 'D'; the sentences we are testing will both be true as long as we assign the truth-value **F** to 'A' and to 'C', no matter what we assign to 'D'. Thus we can recover *two* sets of truth-value assignments from the left-hand open branch, those that assign the values in the first row and those that assign the values in the second row:

A	C	D
F	**F**	**T**
F	**F**	**F**

The next open branch we come to contains the literals 'D' and '~ A'. Neither 'C' nor '~ C' occurs on this second open branch. Hence we can expect to recover two sets of truth-value assignments from this branch as well, those that assign the values indicated in the first row below and those that assign the values indicated in the second row:

A	C	D
F	**T**	**T**
F	**F**	**T**

In fact, only the first of these rows specifies a new set of truth-value assignments; the second set we also obtained from the first open branch. The third open branch contains the same literals as the first, so here there are no new truth-value assignments to be recovered. The fourth open branch contains the literals 'D', '~ C', and '~ A'; from this branch we can recover the set of truth-value assignments that assign the following values to 'A', 'C', and 'D':

A	C	D
F	**F**	**T**

This set is also yielded by the other three open branches we have examined. The last open branch contains the literals '~ C', '~ A', and '~ D' and so yields the set of truth-value assignments that assign the following values to 'A', 'C', and 'D':

A	C	D
F	**F**	**F**

This is also not a new set of truth-value assignments; we can recover this set from the first and third open branches as well. In sum, we have five completed open branches on our tree, and these branches yield three distinct

sets of truth-value assignments. The number of open branches on a completed truth-tree is, again, no guide to the number of distinct sets of truth-value assignments that can be recovered from that tree. Of course, to show that a set of sentences is truth-functionally consistent, we need only show that there is at least one truth-value assignment on which all the members of the set are true. And, to show that there *is* such an assignment, it suffices to recover one set of truth-value assignments.

Sets of truth-value assignments can be recovered only from completed open branches. Closed branches represent unsuccessful attempts to find such assignments. Thus the branches of a truth-tree should not be thought of as corresponding to the rows of a truth-table. They do not. However, constructing a truth-tree for a set of sentences does tell us a lot about what the truth-table for the same set of sentences would be like. If the tree is closed, we know that there is no row in the corresponding truth-table in which every member of the set in question has a **T** under its main connective. If the tree has a completed open branch, we know that there is at least one row in that table in which every member of the set in question has a **T** under its main connective. And, if we count the number of distinct sets of truth-value assignments we can recover, we know that there will be exactly that many rows in the corresponding truth-table such that every member of the set in question has a **T** under its main connective.

4.1**E** EXERCISES

1. Construct truth-trees to test each of the following sets of sentences for truth-functional consistency. If a set is consistent, recover one set of truth-value assignments from your tree that shows this.

a. {H ∨ G, ~ G & ~ H}
*b. {K ∨ (M & ~ M), J & ~ C}
c. {H ≡ J, ~ H ∨ B}
*d. {~ (M & ~ N), ~ (K ∨ M) & ~ ~ M}
e. {~ (A ⊃ B), ~ (B ⊃ A)}
*f. {H & (~ K ⊃ M), ~ K, ~ (H ⊃ M)}
g. {B ⊃ (D ⊃ E), D & B}
*h. {(A & B) ∨ (A & C), ~ (A & B)}
i. {H ≡ G, ~ (H ⊃ G)}
*j. {~ [~ A ⊃ (B ⊃ C)], A ⊃ C}
k. {L ≡ (J & K), ~ J, ~ L ⊃ L}
*l. {H ≡ ~ G, H ⊃ G}
m. {~ [(A ≡ B) ≡ A]}
*n. {A ⊃ ~ (A ≡ B), ~ (A ⊃ B)}
o. {~ [(A ⊃ ~ B) ⊃ (B ⊃ A)], ~ (~ A ⊃ ~ B)}
*p. {A ⊃ [B ⊃ (C ⊃ A)], ~ (B ⊃ ~ A)}
q. {(A & ~ C) ∨ (A & ~ B), A ⊃ B, C}
*r. {~ (A & ~ B) ⊃ ~ A, ~ (~ A & B) ⊃ ~ B, B & ~ A}

2. Which of the following claims are true? Explain your reasoning.

a. If a completed truth-tree contains at least one open branch, then at least one set of truth-value assignments on which all the members of the set being tested are true can be recovered from that open branch.

*b. A completed open branch of a truth-tree yields at most one set of truth-value assignments on which every member of the set being tested is true.

c. If a set of sentences has a truth-tree with a completed open branch, then that set is truth-functionally consistent.

*d. If a truth-tree is closed, there are no open branches on the tree.

e. If a truth-tree is closed, the set of sentences being tested is truth-functionally inconsistent.

*f. If a truth-tree is closed, every sentence on the tree either has been decomposed or is a literal.

g. If there are eight distinct atomic components of the members of a set Γ of sentences of *SL,* then a completed tree for Γ will have eight branches.

*h. A completed truth-tree with at least one open branch and at least one closed branch is an open tree.

i. If a tree has a closed branch, then there is a truth-value assignment on which all the members of the set being tested are false.

*j. If a set Γ of sentences of *SL* has a tree with a completed open branch, then every nonempty subset of Γ also has a tree with a completed open branch.

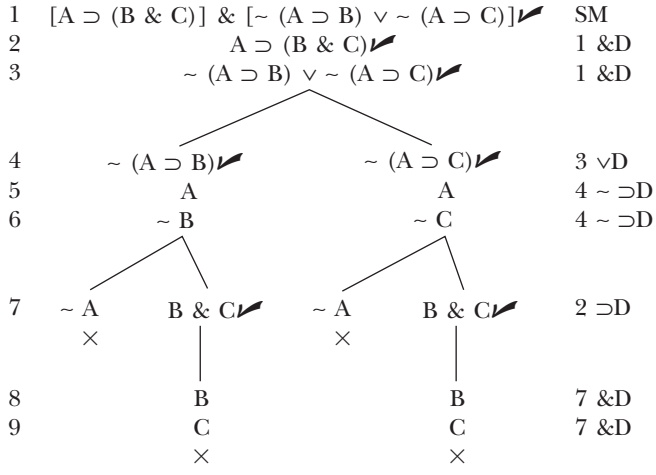
k. If no member of a set Γ of sentences of *SL* contains a tilde, then no tree for Γ will have a closed branch.

4.2 USING TRUTH-TREES TO TEST FOR OTHER TRUTH-FUNCTIONAL PROPERTIES

Because the core concepts of sentential logic other than truth-functional consistency can be explicated in terms of truth-functional consistency, truth-trees can be used to determine when these concepts apply to sentences and sets of sentences of *SL*. We know that each sentence of *SL* is either truth-functionally true, truth-functionally false, or truth-functionally indeterminate. Truth-trees can be used to determine into which of these categories a particular sentence of *SL* falls. Suppose that we want to know whether a sentence **P** is truth-functionally false. Remember that, if **P** is not truth-functionally false, there is some truth-value assignment on which it is true; hence the unit set {**P**} will be truth-functionally consistent. However, if **P** is truth-functionally false, there is no truth-value assignment on which it is true; hence there is no assignment on which every member of {**P**} is true, and so {**P**} is truth-functionally inconsistent.

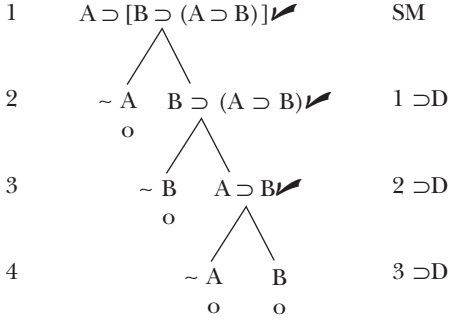
> A sentence **P** of *SL* is *truth-functionally false* if and only if {**P**} has a closed truth-tree.

Here is a tree for the set {[A ⊃ (B & C] & [~ (A ⊃ B) ∨ ~ (A ⊃ C)]}:



All the branches of this tree do close, so there is no truth-value assignment on which the one member of the set we are testing is true. Hence the set is truth-functionally inconsistent, and its single member is truth-functionally false. In constructing this tree we were able to save work at lines 5 and 6 by decomposing two sentences, '~ (A ⊃ B)' and '~ (A ⊃ C)', in one step. We could do so because these sentences occur on the same line, line 4, and are decomposed by the same rule, Negated Conditional Decomposition. Of course, we also could have done them separately.

Next we use the tree method to determine whether 'A ⊃ [B ⊃ (A ⊃ B)]' is truth-functionally false:



This tree obviously has a completed open branch (in fact it has four), so the unit set we are testing is truth-functionally consistent. Hence there is at least one truth-value assignment on which the one member of that set is true, and that sentence is thus not truth-functionally false. (Note that we could have stopped at line 2, where the first completed open branch ends.)

Although we know that 'A ⊃ [B ⊃ (A ⊃ B)]' is not truth-functionally false, we do not yet know whether this sentence is truth-functionally indeterminate

or truth-functionally true. We can find out by constructing another tree. Suppose that the sentence we are concerned with, '$A \supset [B \supset (A \supset B)]$', is truth-functionally true. Then its negation, '$\sim (A \supset [B \supset (A \supset B)])$', must be truth-functionally false. So we can determine whether the sentence is truth-functionally true by testing whether its negation is truth-functionally false, that is, by seeing whether the unit set of its negation has a closed tree. Here is a tree for that set:

1	$\sim (A \supset [B \supset (A \supset B)])$✓	SM
2	A	1 ~ ⊃D
3	$\sim [B \supset (A \supset B)]$✓	1 ~ ⊃D
4	B	3 ~ ⊃D
5	$\sim (A \supset B)$✓	3 ~ ⊃D
6	A	5 ~ ⊃D
7	~ B	5 ~ ⊃D
	×	

The tree is closed. So there is no truth-value assignment on which the sentence '$\sim (A \supset [B \supset (A \supset B)])$' is true. Since that sentence is a negation, there is no truth-value assignment on which the sentence of which it is a negation, '$A \supset [B \supset (A \supset B)]$', is false. That sentence is therefore truth-functionally true.

A sentence **P** of *SL* is *truth-functionally true* if and only if {~ **P**} has a closed tree.

A sentence is truth-functionally indeterminate if and only if it is neither truth-functionally true nor truth-functionally false. Therefore

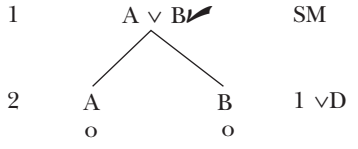
A sentence **P** of *SL* is *truth-functionally indeterminate* if and only if neither {**P**} nor {~ **P**} has a closed tree.

When we are interested in determining the truth-functional status of a sentence, the trees we construct will be trees for unit sets of sentences. However, we shall allow ourselves to talk informally of constructing a tree for **P** or for ~ **P**. Such talk is to be understood as shorthand for talk of trees for unit sets.

When determining the truth-functional status of a sentence **P**, we shall sometimes end up constructing two trees, one for **P** and one for ~ **P**. Of course, if we suspect that **P** is truth-functionally true, we should first do a truth-tree for ~ **P**; if we suspect that **P** is truth-functionally false, we should first do a truth-tree for **P** itself.

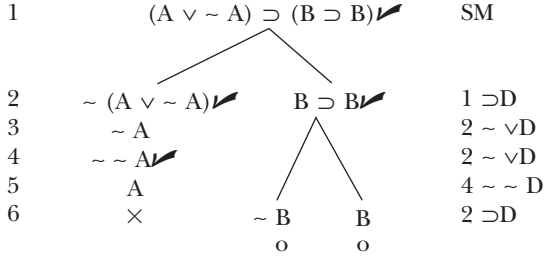
Recall that all of the branches of our tree for '$A \supset [B \supset (A \supset B)]$' were completed open branches. One might think that it follows from this alone that '$A \supset [B \supset (A \supset B)]$' is truth-functionally true, for surely, if that

sentence were not truth-functionally true, a tree for that sentence would have at least one closed branch. But this reasoning is mistaken. Many sentences that are not truth-functional truths have trees all of whose branches are completed open branches, and many truth-functional truths have trees with some closed branches. Consider the truth-tree for the simple disjunction 'A ∨ B':



Both branches of this tree are completed open branches. Yet we know that 'A ∨ B' is not a truth-functional truth. Its truth-table will mirror the characteristic truth-table for disjunctions; that is, the first three rows under its main connective will contain **T**, and the fourth row will contain **F**.

To see that not all truth-functional truths have completed truth-trees all of whose branches are open, consider the sentence '(A ∨ ~ A) ⊃ (B ⊃ B)'. This sentence is a truth-functional truth inasmuch as its consequent is a truth-functional truth (its antecedent is as well), and for this reason there is no truth-value assignment on which '(A ∨ ~ A) ⊃ (B ⊃ B)' is false. But this tree for the sentence does have one closed branch:



There is a way we can avoid constructing two truth-trees for one sentence. Suppose that we construct a tree for a sentence **P**, thinking it may be truth–functionally false, but the tree does not close. We now know that **P** is either truth-functionally true or truth-functionally indeterminate. If it is true on all truth-value assignments, it is truth-functionally true; if it is true on only some assignments, it is truth-functionally indeterminate. We can find out which is the case by counting the number of distinct sets of truth-value assignments that are recoverable from the completed open tree—for these sets correspond to the rows of the truth-table for the sentence being tested in which there is a **T** under that sentence's main connective. If **P** has **n** atomic components, we

shall recover 2^n distinct sets of truth-value assignments from our tree if and only if **P** is truth-functionally true.

Recall our tree for 'A ∨ B', which has two open and no closed branches. The only literal occurring on the left-hand branch is 'A', so from that branch we can recover two sets of truth-value assignments, one set assigning the truth-value **T** to 'B' and one set assigning the truth-value **F** to 'B':

A	B
T	**T**
T	**F**

We can also recover two sets of truth-value assignments from the right-hand open branch. But only one of these is a new set:

A	B
F	**T**

From neither branch can we recover the set of truth-value assignments that assign the truth-value **F** to both 'A' and 'B', and this is just what we expected, for a disjunction is false when (and only when) both its disjuncts are false. By identifying all of the recoverable sets of truth-value assignments—and finding that there are only three such sets—we have shown that 'A ∨ B' is truth-functionally indeterminate, without having to construct two trees for that sentence.

We can use this same procedure to verify that '(A ∨ ~ A) ⊃ (B ⊃ B)' is indeed truth-functionally true. This sentence has two atomic components, so we can expect to recover four distinct sets of truth-value assignments from the tree for this sentence, each set representing one combination of values that the atomic components 'A' and 'B' can have. The tree has two completed open branches. The only literal on the left-hand branch is '~ B', so this branch yields two sets of truth-value assignments:

A	B
T	**F**
F	**F**

The only literal occurring on the right-hand branch is 'B', so this branch yields two new fragments:

A	B
T	**T**
F	**T**

We have recovered four distinct sets of truth-value assignments, thus showing that the sentence being tested is true on every truth-value assignment. We have verified that it is truth-functionally true, even though the tree for that sentence has one closed branch.

Suppose we suspect that a sentence **P** is truth-functionally true and accordingly construct a tree for the negation of that sentence, ~ **P**. Suppose also that our tree has at least one completed open branch, and thus that in this case our suspicion were wrong: **P** is not truth-functionally true. The standard procedure would now be to construct a tree for **P** to see whether that sentence is truth-functionally false or truth-functionally indeterminate. Instead, we can see which distinct sets of truth-value assignments can be recovered from the tree we have already constructed for ~ **P**. The sets of truth-value assignments we can recover are those on which ~ **P** is true. If we can recover all sets of truth-value assignments, each set assigning a distinct combination of values to the atomic components of **P**, then we know that ~ **P** is true on every truth-value assignment and is thus truth-functionally true. And if ~ **P** is truth-functionally true, **P** is truth-functionally false. If we cannot recover all sets of truth-value assignments from our tree, we know that there is at least one set of truth-value assignments on which ~ **P** is false, and hence on which **P** is true. In this case **P** is truth-functionally indeterminate.

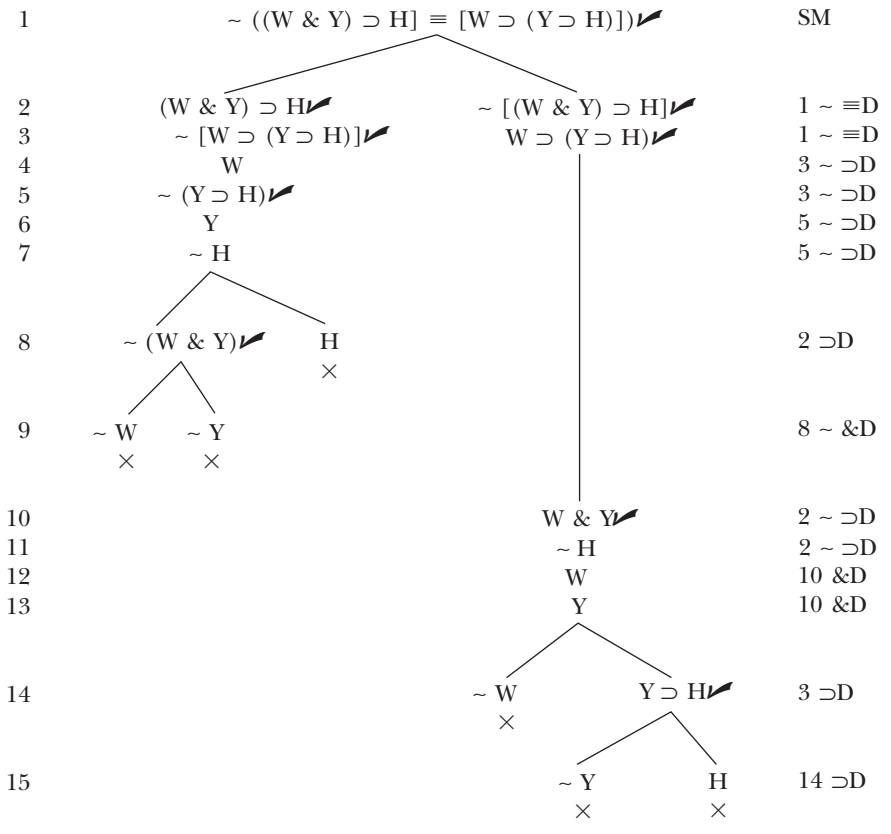
The method of recovering truth-value assignments always allows us to avoid constructing a second tree. However, to use the method, we must complete the tree we are working with (rather than stopping when we have one completed open branch). As a result, when the tree is complex and the number of distinct combinations of truth-values that can be assigned to the atomic components of a sentence is relatively large—eight, sixteen, thirty-two, or more—it is often easier to construct a second tree than to recover and count distinct sets of truth-value assignments.

Sentences **P** and **Q** of *SL* are truth-functionally equivalent if and only if there is no truth-value assignment on which **P** and **Q** have different truth-values. It follows that sentences **P** and **Q** are truth-functionally equivalent if and only if their corresponding material biconditional, **P** ≡ **Q**, is truth-functionally true. And that material biconditional is truth-functionally true if and only if its negation is truth-functionally false. Since a sentence of *SL* is truth-functionally false if and only if its unit set has a closed tree, it follows that:

> Sentences **P** and **Q** of *SL* are *truth-functionally equivalent* if and only if {~ (**P** ≡ **Q**)} has a closed tree.

In Chapter 3, we showed that '(W & Y) ⊃ H' is truth-functionally equivalent to 'W ⊃ (Y ⊃ H)' by producing a truth-table revealing that these two sentences have the same truth-value on every truth-value assignment. We can

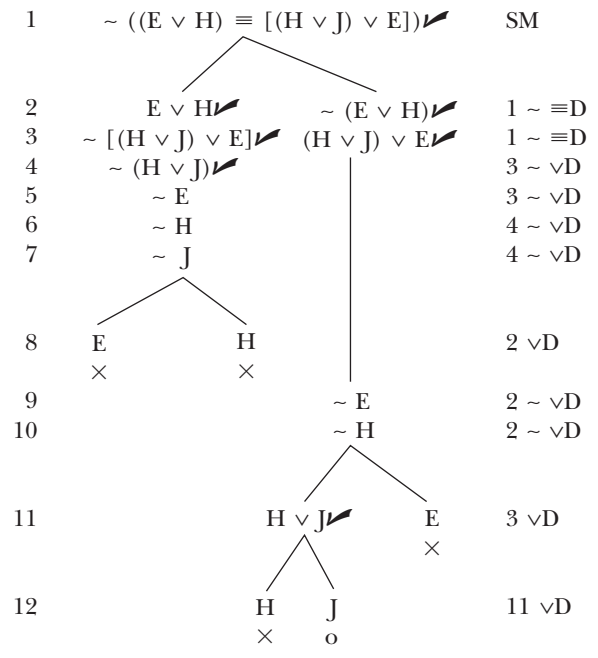
now use the truth-tree method to reach the same result. To show that these sentences are equivalent, we need show only that their corresponding material biconditional is truth-functionally true, and we can do this by showing that the negation of that biconditional has a closed truth-tree.



This tree is closed. The sentence at the top of the tree is therefore false on every truth-value assignment, and the biconditional of which it is the negation is therefore true on every truth-value assignment. So the immediate components of that biconditional, '(W & Y) ⊃ H' and 'W ⊃ (Y ⊃ H)', are truth-functionally equivalent.

In Chapter 3 we also showed that 'E ∨ H' and '(H ∨ J) ∨ E' are not truth-functionally equivalent. We can now show this by using the truth-tree method. These sentences are truth-functionally equivalent if and only if their corresponding material biconditional, '(E ∨ H) ≡ [(H ∨ J) ∨ E]', is truth-

functionally true. And that biconditional is truth-functionally true if and only if the tree for its negation closes:



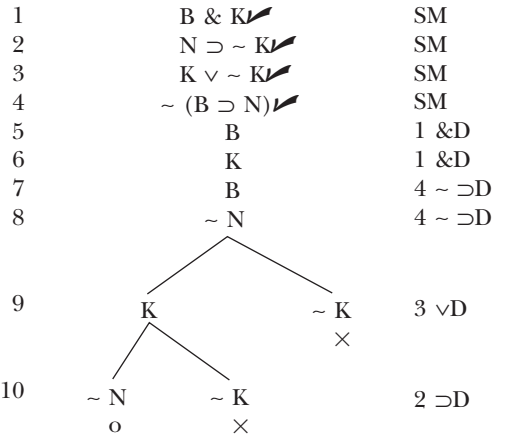
Since this truth-tree has a completed open branch, there is at least one truth-value assignment on which the sentence at the top of the tree is true. That sentence is therefore not truth-functionally false, and the biconditional of which it is the negation is thus not truth-functionally true. It follows that the sentences that are the immediate components of that biconditional, 'E ∨ H' and '(H ∨ J) ∨ E', are not truth-functionally equivalent. They have different truth-values on every truth-value assignment that assigns the following values to 'E', 'H', and 'J':

E	H	J
F	**F**	**T**

We can use truth-trees to test for truth-functional entailment. Recall that, where **P** is a sentence of *SL* and Γ is a set of sentences of *SL*, Γ truth-functionally entails **P** if and only if there is no truth-value assignment on which every member of Γ is true and **P** is false. It follows that a set Γ of sentences truth-functionally entails a sentence **P** if and only if the set of sentences Γ ∪ Γ {~**P**} is truth-functionally inconsistent. Hence, to see if a finite set Γ truth-functionally entails **P**, we construct a tree for the members of Γ ∪ {~ **P**}. Here we have to be careful to negate the allegedly entailed sentence before constructing the tree.

A finite set Γ of sentences of *SL truth-functionally entails* a sentence **P** of *SL* if and only if Γ ∪ {~ **P**} has a closed truth-tree.

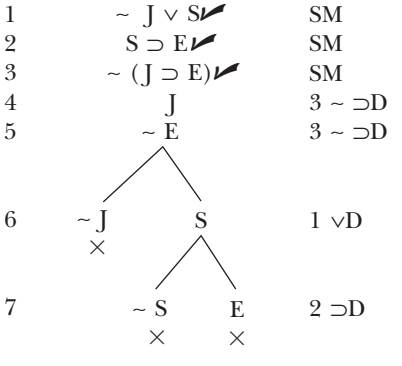
Does the set {B & K, N ⊃ ~ K, K ∨ ~ K} truth-functionally entail 'B ⊃ N'? We can find out by constructing a tree for {B & K, N ⊃ ~ K, K ∨ ~ K, ~ (B ⊃ N)}:



Since this truth-tree has a completed open branch, there is a truth-value assignment on which all the sentences we are testing are true. Hence there is an assignment on which the members of the set {B & K, N ⊃ ~ K, K ∨ ~ K} are all true and the sentence 'B ⊃ N' is false. So the entailment does not in fact hold. The set members are true while 'B ⊃ N' is false on every truth-value assignment that assigns the following values to 'B', 'K', and 'N':

B	K	N
T	**T**	**F**

On the other hand, {~ J ∨ S, S ⊃ E} does truth-functionally entail 'J ⊃ E', as the following closed truth-tree shows:



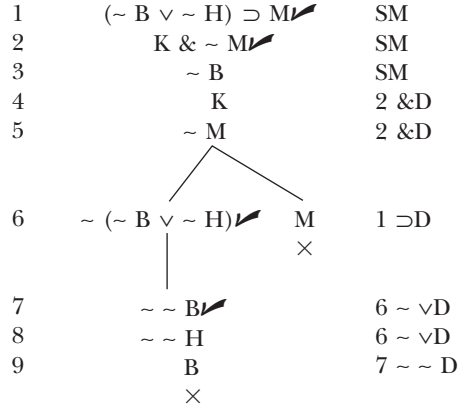
An argument is truth-functionally valid if and only if there is no truth-value assignment on which the premises are true and the conclusion false. It follows that an argument is truth-functionally valid if and only if there is no truth-value assignment on which both the premises and the *negation* of the conclusion are true. Hence an argument is truth-functionally valid if and only if the set consisting of the premises and the *negation* of the conclusion is truth-functionally inconsistent:

> An argument of *SL* with a finite number of premises is *truth-functionally valid* if and only if the set consisting of the premises and the negation of the conclusion has a closed truth-tree.

In our next example we use the tree method to determine whether the following argument is truth-functionally valid:

$$\begin{array}{l}(\sim B \vee \sim H) \supset M\\ K \,\&\, \sim M\\ \hline B\end{array}$$

Trees here are no different from the trees we have already constructed, but we must remember to construct a tree for the premises and the *negation* of the conclusion, rather than for the premises and the conclusion:

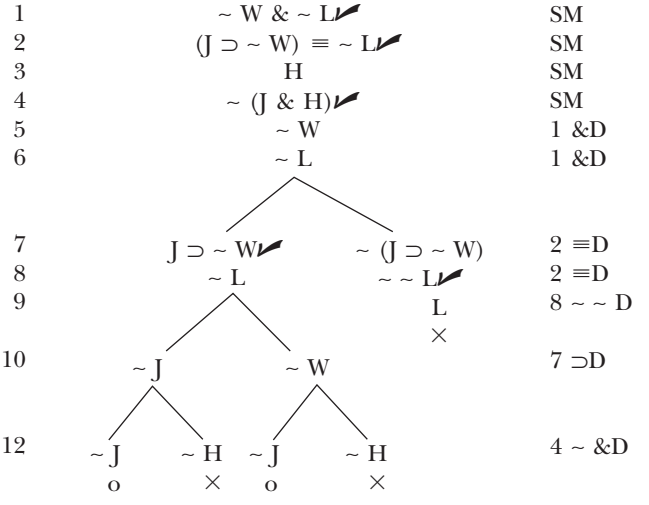


This truth-tree is closed. So we know that the set consisting of the sentences we are testing is truth-functionally inconsistent, and hence that the argument from which the set was formed is truth-functionally valid. Our reasoning is this: The closed tree shows that there is no truth-value assignment on which the premises of our argument are all true and the negation of the conclusion is also true. Therefore there is no truth-value assignment on which those premises are true and the conclusion false, so the argument is truth-functionally valid.

As another example, we'll construct a truth-tree to test the following argument:

$$\begin{array}{l} \sim W \,\&\, \sim L \\ (J \supset \sim W) \equiv \sim L \\ H \\ \hline J \,\&\, H \end{array}$$

Our tree for this argument follows. Again, it is the negation of the conclusion that we use along with the premises, not the conclusion itself:



Because this tree has at least one completed open branch, we can recover a set of truth-value assignments on which the premises and the negation of the conclusion are true, and hence on which the premises are true and the conclusion false. So the argument we are testing is truth-functionally invalid. The recoverable truth-value assignments assign the following values to the four atomic sentences that occur in the premises and conclusion:

H	J	L	W
T	**F**	**F**	**F**

Because an argument is truth-functionally valid if and only if the set consisting of the premises of that argument truth-functionally entails the conclusion of that argument, the procedures for constructing truth-trees to test for truth-functional validity and for truth-functional entailment are similar. In the case of testing for truth-functional validity, the conclusion is negated; in the case of testing for truth-functional entailment, the allegedly entailed sentence is negated.

4.2**E**. EXERCISES

1. For each of the following sentences, use the truth-tree method to determine its truth-functional status—that is, whether it is truth-functionally true, truth-functionally false, or truth-functionally indeterminate.

a. M & ~ M
*b. M ∨ ~ M
c. ~ M ∨ ~ M
*d. (C ⊃ R) ⊃ [~ R ⊃ ~ (C & J)]
e. (C ⊃ R) & [(C ⊃ ~ R) & ~ (~ C ∨ R)]
*f. (K ≡ W) ∨ (A & W)
g. (~ A ≡ ~Z) & (A & ~ Z)
*h. [L ∨ (J ∨ ~ K)] & (K & [(J ∨ L) ⊃ ~ K])
i. (A ∨ B) & ~ (A ∨ B)
*j. (A ∨ B) ⊃ ~ (A ∨ B)
k. (A ∨ B) ≡ ~ (A ∨ B)
*l. ~ (D ∨ F) ≡ ~ (D & F)
m. ~ (D ∨ F) ≡ (~ D ∨ ~ F)
*n. ~ (D ∨ F) ≡ (~ D & ~ F)

2. Construct truth-trees to determine which of the following sentences are truth-functionally true.

a. (B ⊃ L) ∨ (L ⊃ B)
*b. (B ⊃ L) & (L ⊃ B)
c. (A ≡ K) ⊃ (A ∨ K)
*d. (A ≡ K) ⊃ (~ A ∨ K)
e. [(J ⊃ Z) & ~ Z] ⊃ ~ J
*f. [(J ⊃ Z) & ~ J] ⊃ ~ Z
g. (B ⊃ (M ⊃ H)) ≡ [(B ⊃ M) ⊃ (B ⊃ H)]
*h. M ⊃ [L ≡ (~ M ≡ ~ L)]
i. [(A ⊃ B) ⊃ A] ⊃ A
*j. (A & ~ B) ⊃ ~ (A & B)
k. [(A & B) ⊃ C] ≡ [(A ⊃ ~ B) ∨ C]
*l. (D ≡ ~ E) ≡ ~ (D ≡ E)
m. [A ⊃ (B & C)] ⊃ [A ⊃ (B ⊃ C)]
*n. [A ⊃ (B & C)] ≡ [A ⊃ (B ⊃ C)]
o. [(A & B) ⊃ C] ≡ [A ⊃ (B ⊃ C)]

3. For each of the following sentences, use the truth-tree method to determine its truth-functional status—that is, whether it is truth-functionally true, truth-functionally false, or truth-functionally indeterminate. In each case construct a tree only for the given sentence. If the tree does not close, determine the truth-functional status of the sentence by recovering and counting distinct sets of truth-value assignments.

a. ~ (~ A ⊃ A)
*b. J ⊃ (K ⊃ L)
c. (A ≡ ~ A) ⊃ ~ (A ≡ ~ A)
*d. (E ≡ H) ⊃ (~ E ⊃ ~ H)
e. (~ B & ~ D) ∨ ~ (B ∨ D)
*f. ([(C ⊃ D) & (D ⊃ E)] & C) & ~ E

g. $[(A \vee B) \,\&\, (A \vee C)] \supset \sim (B \,\&\, C)$
*h. $\sim ([(A \vee B) \,\&\, (B \vee C)] \,\&\, (\sim A \,\&\, \sim B))$
i. $(J \vee \sim K) \equiv \sim \sim (K \supset J)$
*j. $\sim B \supset [(B \vee D) \supset D]$

4. Decide which of the following claims are true and which are false. In each case explain and defend your reasoning. Use examples where appropriate.

a. If a completed tree for the unit set of **P**, {**P**} has at least one open branch and at least one closed branch, then **P** is truth-functionally indeterminate.
*b. If **P** is truth-functionally true and has four atomic components, then a completed tree for {**P**} will have four open branches.
c. If a completed tree for {**P**} has all open branches, then **P** is truth-functionally true.
*d. If {**P**} has a closed tree and {**Q**} has a closed tree, then the unit set of every truth-functionally compound sentence whose immediate components are **P** and **Q** will also have a closed tree.
e. If {**P**} and {**Q**} each has a tree with at least one completed open branch, then the unit set of every truth-functionally compound sentence that has **P** and **Q** as its immediate components will have a completed tree with an open branch.
*f. If a completed truth-tree for {**P**} has exactly one open branch, then ~ **P** is truth-functionally indeterminate.
g. If **P** and **Q** are both truth-functionally true, then **P** & **Q**, **P** $\vee$ **Q**, **P** $\supset$ **Q**, and **P** $\equiv$ **Q** will each have a completed tree all of whose branches are open.
*h. If **P** and **Q** are both truth-functionally true, then **P** & **Q**, **P** $\vee$ **Q**, **P** $\supset$ **Q**, and **P** $\equiv$ **Q** will each have a tree with at least two completed open branches.
i. If **P** and **Q** are both truth-functionally false, then **P** & **Q**, **P** $\vee$ **Q**, **P** $\supset$ **Q**, and **P** $\equiv$ **Q** will each have a closed tree.
*j. If **P** and **Q** are both truth-functionally false, then **P** & **Q**, **P** $\vee$ **Q**, **P** $\supset$ **Q**, and **P** $\equiv$ **Q** will each have a completed tree with at least one closed branch.
k. If **P** is truth-functionally true and **Q** is truth-functionally false, then **P** & **Q**, **P** $\vee$ **Q**, **P** $\supset$ **Q**, and **P** $\equiv$ **Q** will each have a completed tree with at least one open branch and one closed branch.

5. Use the truth-tree method to determine whether the following pairs of sentences are truth-functionally equivalent. For those pairs that are not truth-functionally equivalent, recover a set of truth-value assignments that shows this.

a.	$\sim (Z \vee K)$	$\sim Z \,\&\, \sim K$
*b.	$\sim (Z \vee K)$	$\sim Z \vee \sim K$
c.	$(B \,\&\, C) \supset R$	$(B \supset R) \,\&\, (C \supset R)$
*d.	$(B \vee C) \supset R$	$(B \supset R) \,\&\, (C \supset R)$
e.	$A \,\&\, (B \vee C)$	$(A \,\&\, B) \vee (A \,\&\, C)$
*f.	$A \vee (B \,\&\, C)$	$(A \vee B) \,\&\, (A \vee C)$
g.	$D \supset (L \supset M)$	$(D \supset L) \supset M$
*h.	$J \supset (K \equiv L)$	$(J \supset K) \equiv (J \supset L)$
i.	$A \supset A$	$B \supset B$
*j.	$A \,\&\, \sim A$	$B \,\&\, \sim B$
k.	$A \,\&\, \sim B$	$\sim A \vee B$
*l.	$\sim (A \vee B)$	$\sim (A \,\&\, B)$
m.	$\sim (A \equiv B)$	$\sim A \equiv \sim B$

*n.	$A \supset (B \supset C)$	$(A \supset B) \supset C$
o.	$A \,\&\, (B \vee C)$	$(A \,\&\, B) \vee (A \,\&\, C)$
*p.	$A \supset (B \supset C)$	$A \supset (B \,\&\, C)$

6. Decide which of the following claims are true and which are false. In each case explain and defend your reasoning. If **P** and **Q** are truth-functionally equivalent, then

a. A completed truth-tree for $\{\mathbf{P} \equiv \mathbf{Q}\}$ will be open.
*b. A completed truth-tree for $\{\mathbf{P} \equiv \sim \mathbf{Q}\}$ will be open.
c. A completed truth-tree for the set $\{\mathbf{P}, \mathbf{Q}\}$ will be open.
*d. A completed truth-tree for $\{\sim \mathbf{P} \equiv \sim \mathbf{Q}\}$ will be open.

7. Use the truth-tree method to determine which of the following claims are true and which are false. For those that are false, recover a set of truth-value assignments that shows this.

a. $\{A \supset (B \,\&\, C), C \equiv B, \sim C\} \models \sim A$
*b. $\{K \supset H, H \supset L, L \supset M\} \models K \supset M$
c. $\{\sim (A \equiv B), \sim A, \sim B\} \models C \,\&\, \sim C$
*d. $\{\sim (\sim A \,\&\, \sim B)\} \models A \,\&\, B$
e. $\{\sim \sim F \supset \sim \sim G, \sim G \supset \sim F\} \models G \supset F$
*f. $\{A \,\&\, (B \supset C)\} \models (A \,\&\, C) \vee (A \,\&\, B)$
g. $\{[(C \vee D) \,\&\, H] \supset A, D\} \models H \supset A$
*h. $\{(G \equiv H) \vee (\sim G \equiv H)\} \models (\sim G \equiv \sim H) \vee \sim (G \equiv H)$
i. $\{(J \vee M) \supset \sim (J \,\&\, M), M \equiv (M \supset J)\} \models M \supset J$
*j. $\models [A \vee ((K \supset \sim H) \,\&\, \sim A)] \vee \sim A$
k. $\models \sim (A \equiv B) \supset (\sim A \equiv \sim B)$
*l. $\models \sim (C \equiv C) \equiv (C \vee \sim C)$
m. $\models [(A \supset B) \supset (C \supset D)] \supset [C \supset (B \supset D)]$

8. Use the truth-tree method to determine which of the following arguments are truth-functionally valid and which are truth-functionally invalid. For those that are truth-functionally invalid, recover a set of a truth-value assignments that show this.

a. $M \supset (K \supset B)$
$\sim K \supset \sim M$
$L \,\&\, M$

B

*b. $(\sim J \vee K) \supset (L \,\&\, M)$
$\sim (\sim J \vee K)$

$\sim (L \,\&\, M)$

c. $A \,\&\, (B \vee C)$
$(\sim C \vee H) \,\&\, (H \supset \sim H)$

$A \,\&\, B$

*d. $(D \equiv \sim G) \,\&\, G$
$[G \vee ((A \supset D) \,\&\, A)] \supset \sim D$

$G \supset \sim D$

e. $(M \equiv K) \vee \sim (K \,\&\, D)$
$\sim M \supset \sim K$
$\sim D \supset \sim (K \,\&\, D)$

M

*f. $(J \supset T) \supset J$
$(T \supset J) \supset T$

$\sim J \vee \sim T$

g. $B \& (H \vee Z)$
$\sim Z \supset K$
$(B \equiv Z) \supset \sim Z$
$\sim K$

$M \& N$

*h. $A \vee \sim (B \& C)$
$\sim B$
$\sim (A \vee C)$

A

i. $A \& (B \supset C)$

$(A \& C) \vee (A \& \sim B)$

*j. $(G \equiv H) \vee (\sim G \equiv H)$

$(\sim G \equiv \sim H) \vee \sim (G \equiv H)$

k. $A \supset \sim A$
$(B \supset A) \supset B$

$A \equiv \sim B$

*l. $B \vee (A \& \sim C)$
$(C \vee A) \equiv B$
$\sim B \vee A$

$\sim (A \vee C)$

9. Symbolize each of the following arguments and then use the truth-tree method to determine whether the symbolized argument is truth-functionally valid. If an argument is not truth-functionally valid, recover a set of truth-value assignments that show this.

a. The social security system will succeed if and only if more money is collected through social security taxes. Unless the social security system succeeds, many senior citizens will be in trouble. Although members of Congress claim to be sympathetic to senior citizens, more money won't be collected through social security taxes. Hence the social security system will not succeed.

*b. Either the president and the senators will support the legislation, or the president and the representatives will support it. Moreover, the representatives will support the legislation, provided that a majority of the people support it. The people don't support it. Thus the senators will support the legislation.

c. If the president acts quickly the social security system will be saved, and if the social security system is saved, senior citizens will be delighted. If the president is pressured by members of the Senate, by members of the House of Representatives, or by senior citizens, he will act quickly. However, neither members of the Senate nor members of the House will pressure the president, but senior citizens will. Therefore senior citizens will be delighted.

*d. The president won't veto the bill if Congress passes it, and Congress will pass it if and only if both the Senate passes it and the House of Representatives passes it. But the House of Representatives will pass it only if a majority of Democrats will vote for it; and indeed, a majority of Democrats will vote for it. Therefore the president won't veto the bill.

e. At most, one of the two houses of Congress will pass the bill. If either the House of Representatives or the Senate passes it, the voters will be pleased; but if both houses of Congress pass the bill, the president will not be pleased. If the president is not pleased, not all the members of the White House will be happy. Hence some members of the White House will not be happy.

10. Show that constructing a tree for the premises and conclusion (not the negation of the conclusion) of an argument of *SL* yields no useful information concerning the validity of the argument by completing the following exercises.

a. Give two arguments of *SL,* one valid and the other invalid, such that the trees for the premises and conclusion of these arguments both have at least one completed open branch. Construct the trees and explain why they are not useful in determining whether the arguments in question are truth-functionally valid.

*b. Give two arguments of *SL,* one valid and the other invalid, such that the trees for the premises and conclusion of these arguments are both closed. Construct the trees and explain why they are not useful in determining whether the arguments in question are truth-functionally valid.

c. Explain why (a) and (b) together constitute a proof that there is no useful information concerning the validity of an argument to be obtained by doing a tree for the premises and conclusion of the argument.

11. Suppose we define a new connective, '|', thus:

P	**Q**	**P|Q**
T	**T**	**F**
T	**F**	**T**
F	**T**	**T**
F	**F**	**T**

To accommodate this new connective, we have to add two new rules to our truth-tree system, one for decomposing sentences of the form **P|Q** and one for decomposing sentences of the form ~ (**P|Q**).

a. Give the rules needed for sentences of these two forms.

b. Use the new rules to test the sentences 'A|B' and '(A|A) ∨ (B|B)' for truth-functional equivalence, using the truth-tree method. State your result.

GLOSSARY

Core Logical Concepts

TRUTH-FUNCTIONAL CONSISTENCY: A finite set Γ of sentences of *SL* is *truth-functionally consistent* if and only if Γ has a truth-tree with at least one completed open branch.

TRUTH-FUNCTIONAL INCONSISTENCY: A finite set Γ of sentences of *SL* is *truth-functionally inconsistent* if and only Γ has a closed truth-tree.

TRUTH-FUNCTIONAL FALSITY: A sentence **P** of *SL* is *truth-functionally false* if and only if {**P**} has a closed truth-tree.

TRUTH-FUNCTIONAL TRUTH: A sentence **P** of *SL* is *truth-functionally true* if and only if {~ **P**} has a closed truth-tree.

TRUTH-FUNCTIONAL INDETERMINACY: A sentence **P** of *SL* is *truth-functionally indeterminate* if and only if neither {**P**} nor {~ **P**} has a closed truth-tree.

TRUTH-FUNCTIONAL EQUIVALENCE: Sentences **P** and **Q** of *SL* are *truth-functionally equivalent* if and only if {~ (**P** ≡ **Q**)} has a closed truth-tree.

TRUTH-FUNCTIONAL ENTAILMENT: A finite set Γ of sentences of *SL truth-functionally entails* a sentence **P** of *SL* if and only if $\Gamma \cup \{\sim \mathbf{P}\}$ has a closed truth-tree.

TRUTH-FUNCTIONAL VALIDITY: An argument of *SL* with a finite number of premises is *truth-functionally valid* if and only if the set consisting of the premises and the negation of the conclusion has a closed truth-tree.

Key Truth-Tree Concepts

CLOSED BRANCH: A branch on which contradictory literals occur.

OPEN BRANCH: A branch that is not closed.

COMPLETED OPEN BRANCH: An open branch on which every sentence is either a literal or has been decomposed (checked off).

CLOSED TRUTH-TREE: A truth-tree each of whose branches is closed.

OPEN TRUTH-TREE: A truth-tree that is not closed.

COMPLETED TRUTH-TREE: A truth-tree each of whose branches is either closed or a completed open branch.

Chapter 5

SENTENTIAL LOGIC: DERIVATIONS

In Section 5.1 we introduce the derivation system *SD* and the concept of a derivation. In Section 5.2 we introduce syntactic analogues of core logical concepts: **derivable in** *SD*, **valid in** *SD*, **theorem** in *SD*, **equivalent** in *SD*, and **inconsistent** in *SD*. Section 5.3 is devoted to developing strategies for constructing derivations in *SD,* and Section 5.4 introduces the derivation system *SD*+, which is an expansion of *SD*.

5.1 THE DERIVATION SYSTEM *SD*

In Chapter 3 we presented semantic accounts of consistency, validity, equivalence, entailment, logical truth, and logical falsity. The semantic truth-table and truth-tree tests we developed for these properties in Chapters 3 and 4 show whether there is or is not a truth-value assignment of a particular kind for a particular sentence or group of sentences. These test procedures can hardly be said to reflect the reasoning we do in everyday discourse when we are trying to show, for example, that an argument is valid or that a set of sentences is inconsistent. In this chapter we develop techniques that do, at least in broad outline, parallel the kind of reasoning we do make use of in everyday discourse. These techniques rely on the form or structure of sentences of *SL* and are

not intended to reveal whether there is or is not a truth-value assignment of a certain sort. These are therefore syntactic techniques.

Consider the following argument:

> If Marshall survives the current scandal and if her opponent doesn't outspend her then Marshall will be reelected. If it continues to be politics as usual Marshall will survive the latest scandal. The scandal is no longer front page news, so it is going to be politics as usual. Marshall's opponent will not outspend her. So Marshall will be reelected.

How might we, in everyday discourse, convince ourselves that the foregoing argument is valid? We will start by providing an explicit paraphrase of the premises and conclusion of this argument:

> If (Marshall will survive the current scandal and it is not the case that Marshall's opponent outspends Marshall) then Marshall will be reelected.
>
> If it continues to be politics as usual then Marshall will survive the current scandal.
>
> It is not the case that the scandal is still front page news and it continues to be politics as usual.
>
> It is not the case that Marshall's opponent outspends Marshall.
>
> ---
>
> Marshall will be reelected.

Note that we paraphrased the third premise as a conjunction. The task before us is to show that starting with the premises as assumptions we can, by a series of obvious inferences, reach the conclusion. We can do this as follows.

1	If (Marshall will survive the current scandal and it is not the case that Marshall's opponent outspends Marshall) then Marshall will be reelected.	Assumption
2	If it continues to be politics as usual then Marshall will survive the current scandal.	Assumption
3	It is not the case that the scandal is still front page news and it continues to be politics as usual.	Assumption
4	It is not the case that Marshall's opponent outspends Marshall.	Assumption
5	It continues to be politics as usual.	From 3
6	Marshall will survive the current scandal.	From 2 and 5

7	Marshall will survive the current scandal and it is not the case that Marshall's opponent outspends Marshall.	From 6 and 4
8	Marshall will be reelected.	From 1 and 7

The structure of our reasoning may be more apparent when we symbolize these paraphrases in *SL* and indicate how each step of our reasoning is justified:

1	$(S \,\&\, \sim O) \supset R$	Assumption
2	$C \supset S$	Assumption
3	$\sim F \,\&\, C$	Assumption
4	$\sim O$	Assumption
5	C	From 3
6	S	From 2 and 5
7	$S \,\&\, \sim O$	From 6 and 4
8	R	From 1 and 7

In this section we develop the derivation system *SD* ('*SD*' for 'Sentential Derivation'), which consists of eleven derivation rules. Each of the inferences represented by lines 5 through 8 above will be justified by a syntactic rule of *SD*. These rules specify that if we have a sentence or sentences of such and such form or forms, then we may infer a sentence of a specified form. The rules are called '**derivation rules**' and the structures we construct using them are called '**derivations**'.

The simplest derivation rule of *SD* is Reiteration:

Reiteration (R)

	P
▷	**P**

Here, and in the rule schema presented below, the '▷' sign indicates the sentence that can be inferred or **derived** using the rule in question. Here is a simple and admittedly uninteresting use of Reiteration:

1	C		Assumption
2	C		1 R

Reiteration is often used in strategies that involve **subderivations**, which we introduce later in this section.

The language *SL* includes five kinds of compound sentences: Negations, Conjunctions, Disjunctions, Material Conditionals, and Material Biconditionals—and there are two derivation rules of *SD* associated with each kind of compound. One rule is for deriving a sentence *from* a compound of the specified sort and

the other is for deriving a compound *of* the specified sort. The former are **elimination** rules. A sentence derived by an elimination rule may have a main connective other than that after which the rule is named, or no main connective. The latter are **introduction** rules, so called because they *yield* an *SL* sentence whose main connective is the one after which the rule is named. Some of these ten rules make use of **subderivations**. We first present the rules that *do not* use subderivations.

5.1.1 THE NON-SUBDERIVATION RULES OF SD

The derivation rules that do not make use of subderivations are

Reiteration (R)

	P
▷	**P**

Conjunction Elimination (&E)

	P & Q		**P & Q**
▷	**P**	▷	**Q**

Conjunction Introduction (&I)

	P
	Q
▷	**P & Q**

Disjunction Introduction (∨I)

	P		**P**
▷	**P ∨ Q**	▷	**Q ∨ P**

Conditional Elimination (⊃E)

	P ⊃ Q
	P
▷	**Q**

Biconditional Elimination (≡E)

	P ≡ Q		**P ≡ Q**
	P		**Q**
▷	**Q**	▷	**P**

These rules are all quite straightforward. The abbreviation for each rule is given in parentheses following the rule name. In each case the sentence the '▷' symbol points to can be derived if the one or two sentences occurring above it have already been derived. Some of these rules have two versions.

> **Conjunction Elimination** specifies that if a conjunction occurs on an earlier line of a derivation then we may enter on a subsequent line either the left conjunct or the right conjunct of the conjunction.

Conjunction Introduction specifies that if **P** and **Q** occur on earlier lines of a derivation then we may enter **P** & **Q** on a subsequent line. Here the rule template should *not* be taken as specifying the order in which **P** and **Q** must be derived before **P** & **Q** can be entered.

Disjunction Introduction specifies that if a sentence **P** occurs on an earlier line of a derivation then we may enter on a subsequent line either **P** ∨ **Q** or **Q** ∨ **P**, where **Q** is *any* sentence of *SL*.

Conditional Elimination specifies that if **P** ⊃ **Q** and **P** occur on earlier lines of a derivation, in either order, then we may enter **Q** on a subsequent line.

Biconditional Elimination specifies that if a sentence of the form **P** ≡ **Q** and one of its immediate components (**P** or **Q**) occur on earlier lines of a derivation, in either order, then we may enter on a subsequent line the other immediate component.

Reiteration, which may seem to be a somewhat strange rule, is often used in strategies that involve **subderivations**, which we introduce later in this section.

Here is a derivation that uses both Conjunction Introduction and Conjunction Elimination:

1	B	Assumption
2	C & ~ D	Assumption
3	~ D	2 &E
4	B & ~ D	1, 3 &I

The sentences on lines 1 and 2 are assumptions, as is indicated in the justification column. The sentence on line 3 is obtained from line 2 by Conjunction Elimination. And the sentence on line 4 is obtained from lines 1 and 3 by Conjunction Introduction.

Disjunction Introduction may seem to be an odd rule, for given a sentence **P** why would we want to obtain **P** ∨ **Q** or **Q** ∨ **P**, both of which are clearly weaker than the sentence from which they can be obtained? The following derivation illustrates an application of Disjunction Introduction and why it is useful, as well as an application of Conditional Elimination:

Derive: H

1	F	Assumption
2	(F ∨ G) ⊃ H	Assumption
3	F ∨ G	1 ∨I
4	H	2, 3 ⊃E

In this derivation our goal was to obtain 'H' from our two assumptions. We indicated this by entering the word 'Derive:' followed by the sentence to be derived, in this case 'H', at the top of the derivation. Hereafter we will always so specify the sentence to be derived. The sentence on line 2 is a material conditional whose

consequent is 'H'. We saw that we could derive 'H' from this line *if we also had the antecedent*, 'F ∨ G'. This was not one of our assumptions. We did have 'F', at line 1. But from 'F' we knew we could derive 'F ∨ G' by Disjunction Introduction, and we did so on line 3. 'H' then followed by Conditional Elimination on line 4.

The following derivation uses Biconditional Elimination and Conjunction Elimination (each twice) as well as Disjunction Introduction and Conjunction Introduction.

Derive: ~ A & (B ∨ C)

1	B ≡ (D ≡ ~ A)	Assumption
2	B & D	Assumption
3	B	2 &E
4	D ≡ ~ A	1, 3 ≡E
5	D	2 &E
6	~ A	4, 5 ≡E
7	B ∨ C	3 ∨I
8	~ A & (B ∨ C)	6, 7 &I

We will discuss strategies for constructing derivations at length later in this chapter. Here we note that the overall strategy we use in constructing derivations is to try to figure out how the desired sentence might be derived—which sentences we need to derive in order to derive that sentence, and then which sentences we need to derive to obtain those sentences, and so on, until we see a path from the given assumptions to the desired sentence. In the foregoing derivation we noted that the sentence to be derived is a conjunction and that conjunctions can be obtained by Conjunction Introduction. So we set about trying to derive the conjuncts of that conjunction, '~ A' and 'B ∨ C'. We reasoned that '~ A' could be derived from line 1 by two uses of Biconditional Elimination if we could derive both 'B' and 'D', and we saw that we could derive both from line 2, by two uses of Conjunction Elimination. And once we had 'B' on line 3 it was easy to derive 'B ∨ C' on line 7 by Disjunction Introduction.

Our next derivation uses all of the Introduction and Elimination rules of *SD* we have so far introduced:

Derive: ~ C

1	~ A ≡ (B & ~ C)	Assumption
2	B & D	Assumption
3	(D ∨ C) ⊃ ~ A	Assumption
4	D	2 &E
5	D ∨ C	4 ∨I
6	~ A	3, 5 ⊃E
7	B & ~ C	1, 6 ≡E
8	~ C	7 &E

Our goal in this derivation was to derive '~ C', and '~ C' is a component of the sentence on line 1. We realized that '~ C' could be derived from line 1 in two steps if we could first derive '~ A' and that because '~ A' is the consequent of

the material conditional on line 3, it could be derived by Conditional Elimination if we could first derive 'D ∨ C'. The latter sentence follows from 'D' by Disjunction Introduction, and 'D' follows from the sentence on line 2, 'B & D', by Conjunction Elimination.

5.1.1**E** EXERCISES

1. Complete the following derivations by entering justifications for the derived sentences:

a. Derive: A & B

1	A	Assumption
2	A ⊃ B	Assumption
3	B	
4	A & B	

*b. Derive: ~ C

1	A ⊃ (B & ~ C)	Assumption
2	A & B	Assumption
3	A	
4	B & ~ C	
5	~ C	

c. Derive: ~ (A ≡ ~ B)

1	~ (A ≡ ~ B) ≡ (~ C ∨ ~ D)	Assumption
2	A ⊃ (~ D & C)	Assumption
3	D & A	Assumption
4	A	
5	~ D & C	
6	~ D	
7	~ C ∨ ~ D	
8	~ (A ≡ ~ B)	

*d. Derive: (E & D) & (~ B & C)

1	~ B ⊃ (D & E)	Assumption
2	(A & ~ B) & C	Assumption
3	A & ~ B	
4	~ B	
5	D & E	
6	D	
7	E	
8	E & D	
9	C	
10	~ B & C	
11	(E & D) & (~ B & C)	

e. Derive: F ⊃ ~ G

1	(E ∨ H) ⊃ (F ⊃ ~ G)	Assumption
2	(C ∨ D) ≡ (E & ~ H)	Assumption
3	C	Assumption
4	C ∨ D	
5	E & ~ H	
6	E	
7	E ∨ H	
8	F ⊃ ~ G	

*f. Derive: ~ G

1	(H & ~ I) ⊃ ~ G	Assumption
2	(F ∨ ~ G) ≡ H	Assumption
3	F & ~ I	Assumption
4	F	
5	F ∨ ~ G	
6	H	
7	~ I	
8	H & ~ I	
9	~ G	

g. Derive: D ≡ ~ B

1	(A & ~ B) ⊃ C	Assumption
2	(C ∨ D) ⊃ (D ≡ ~ B)	Assumption
3	~ B & A	Assumption
4	A	
5	~ B	
6	A & ~ B	
7	C	
8	C ∨ D	
9	D ≡ ~ B	

*h. Derive: M & ~ N

1	(K & ~ L) & (~ I & J)	Assumption
2	~ L ⊃ M	Assumption
3	(K & ~ I) ⊃ ~ N	Assumption
4	K & ~ L	
5	~ L	
6	M	
7	K	
8	~ I & J	
9	~ I	
10	K & ~ I	
11	~ N	
12	M & ~ N	

i. Derive: ~ D & ~ F

1	(A ∨ ~ B) ≡ (A & ~ F)	Assumption
2	C ≡ ~ B	Assumption
3	C & ~ D	Assumption
4	~ D	
5	C	
6	~ B	
7	A ∨ ~ B	
8	A & ~ F	
9	~ F	
10	~ D & ~ F	

*j. Derive: ~ (A ∨ B)

1	A ⊃ [~ B ⊃ ~ (A ∨ B)]	Assumption
2	C ≡ (A & ~ B)	Assumption
3	~ D & C	Assumption
4	C	
5	A & ~ B	
6	A	
7	~ B ⊃ ~ (A ∨ B)	
8	~ B	
9	~ (A ∨ B)	

2. Complete the following derivations.

a. Derive: D & ~ B

1	A & ~ B	Assumption
2	(A ∨ ~ C) ⊃ D	Assumption

*b. Derive: F & ~ H

1	F ≡ ~ G	Assumption
2	D ⊃ ~ G	Assumption
3	~ H & D	Assumption

c. Derive: ~ D ∨ E

1	A & ~ B	Assumption
2	~ B ≡ (A ≡ ~ D)	Assumption

*d. Derive: ~ E ∨ (G & ~ F)

1	D ≡ (C & ~ E)	Assumption
2	F & (F ≡ D)	Assumption

e. Derive: H & ~ I

1	~ F & ~ G	Assumption
2	~ G ⊃ H	Assumption
3	(H & ~ F) ≡ ~ I	Assumption

f. Derive: D & ~ D

*1	(~ A & B) ⊃ (B ≡ D)	Assumption
2	B ⊃ (C & ~ A)	Assumption
3	~ D & B	Assumption

g. Derive: F & ~ G

1	(F ∨ ~ G) ⊃ (F & ~ H)	Assumption
2	~ H ⊃ ~ G	Assumption
3	(~ H ⊃ ~ G) ≡ F	Assumption

5.1.2 THE SUBDERIVATION RULES OF SD

All five of the derivation rules we are about to introduce make use of subderivations. A subderivation is useful when we want to show that if we add an assumption to those we already made then we can derive a sentence that we may not be able to derive without the additional assumption. But every time we use a subderivation—which adds a new assumption—we must eventually end that subderivation and discontinue reliance on the assumption that starts that subderivation. Each subderivation rule provides a way of ending the subderivation it relies on. Once the way subderivations work is understood, it is fairly easy to master the subderivation rules. Our explication of Conditional Introduction will illustrate how subderivations work.

Conditional Introduction (⊃I)

		P
		Q
▷	**P ⊃ Q**	

Suppose we are trying to complete the following derivation:

Derive: A ⊃ H

1	A ⊃ G	Assumption
2	G ⊃ H	Assumption
	A ⊃ H	

Here we have entered the sentence we want to derive, 'A ⊃ H', some distance below our assumptions—because we want the last line of our derivation to be

'A ⊃ H' but we don't yet know how we can get from our assumptions to 'A ⊃ H'. Intuitively, we might reason as follows. We have 'A ⊃ G' as an assumption. *If* we also had 'A', we could derive 'G' by Conditional Elimination. And once we have 'G', we can derive 'H' by Conditional Elimination. That is, *given* 'A ⊃ G' and 'G ⊃ H' we can derive 'H' *if* we also have 'A'. We can encapsulate this reasoning in a subderivation:

Derive: A ⊃ H

1	A ⊃ G	Assumption
2	G ⊃ H	Assumption
3	A	A / ⊃I
4	G	1, 3 ⊃E
5	H	2,4 ⊃E
6	A ⊃ H	3–5 ⊃I

At line 3 we started a derivation within our existing derivation—hence the name 'subderivation'. In this case our purpose in doing so was to show that once we assume 'A' we can derive 'H' (in two steps), using our original assumptions and the assumption that starts our subderivation. Lines 3–5 show that, given our original assumptions, if we have 'A' we can derive 'H'. Note that this does not show that 'H' is a consequence of our original assumptions. Rather, we have shown that the conditional 'A ⊃ H' is a consequence of the original assumptions because we have shown how to derive 'H' given 'A'. It is the entire subderivation, which occupies lines 3–5, that justifies our entering 'A ⊃ C' on line 6. We indicate this by entering, in the justification column, '3–5 ⊃I', not '3,5 ⊃I'. This notation references the entire subderivation, not just lines 3 and 5.

We often use the reasoning process that is captured by Conditional Introduction in everyday reasoning. For example, suppose we know that if Jean gets an A in Biology 400 her grade point average will be 3.8, and that if her grade point average is 3.8 she will graduate with honors. If we assume she does get an A in Biology 400 it follows that she will have a grade point average of 3.8, and from this and what we know about the requirements for graduating with honors, it follows that Jean will graduate with honors. Of course, we do not conclude that Jean *will* graduate with honors, but rather that *if* she gets an A in Biology 400 *then* she will graduate with honors. If we use 'A' to symbolize 'Jean will get an A in Biology 400', 'G' to symbolize 'Jean will have a grade point average of 3.8', and 'H' to symbolize 'Jean will graduate with honors', the derivation we constructed using Conditional Introduction formalizes this reasoning about Jean and her graduating with honors.

There are several points to note before introducing the remaining subderivation rules. First, the vertical lines in a derivation are called '**scope lines**'. Assumptions with just one scope line to their left are the **primary**

assumptions of a derivation. Primary assumptions hold and are available for the entire derivation, as is indicated by the scope line to their immediate left that continues to the end of the derivation. Each subderivation begins with an **auxiliary assumption**, and the scope line to the immediate left of the auxiliary assumption indicates how far the scope of that assumption extends; the auxiliary assumption may be appealed to only so long as the scope line to its immediate left continues. In the above example there is one subderivation, occupying lines 3 through 5. The assumption of that subderivation is in force only through line 5.

We construct subderivations so that we can use rules that require subderivations. In the above example we constructed the subderivation so that we could use the rule Conditional Introduction. This rule calls for assuming, as an auxiliary assumption, the antecedent of the material conditional we wish to derive, and then deriving the consequent of that material conditional within the subderivation. In the justification column for a sentence entered as an auxiliary assumption, we write 'A' (for 'Assumption') and the abbreviation for the rule that calls for a subderivation of the sort we are constructing (here '⊃I'), separated by a slash ('/').

We end a subderivation by using the rule indicated on the assumption line of the subderivation to derive a sentence *outside* the scope of the subderivation, citing the entire subderivation. It is the entire subderivation that justifies applying a subderivation rule. When a subderivation is ended (by using a rule that cites the entire subderivation) we say that the assumption of that subderivation has been **discharged** and is **closed.** The scope of an assumption includes the assumption itself and all sentences and subderivations that occur subsequent to the assumption but before it is discharged. Once an assumption is discharged, neither it nor any sentence or subderivation lying within its scope can be appealed to in justifying subsequent lines of a derivation. We refer to assumptions that have not been discharged as being **open,** and to those that have been discharged as being **closed.** In our example, the scope of the assumption on line 3 extends only to line 5.

We can now give an informal account of **accessibility:** A sentence or subderivation is accessible at line **n** of a derivation (can be appealed to in justifying line **n**) if and only if every scope line to the left of the sentence or subderivation is also to the left of the sentence on line **n**.

Thus, scope lines, the vertical lines to the left of the sentences of a derivation, provide a visual way of telling when a sentence or subderivation is accessible. The leftmost vertical line is the scope line for the entire derivation. Primary assumptions, if any, appear to the immediate right of this scope line at the top of the derivation. Every auxiliary assumption has its own scope line, a line that continues only so long as that assumption remains open. A sentence is accessible only as long as the scope lines to its left continue. Primary assumptions, of course, are never discharged. If a sentence or subderivation is accessible at a given line of a derivation then it can be appealed to in justifying the sentence entered on that line.

Here is another derivation that uses Conditional Introduction:

Derive: $A \supset (B \supset C)$

1	C	Assumption
2	| A	A / ⊃I
3	| | B	A / ⊃I
4	| | C	1 R
5	| B ⊃ C	3–4 ⊃I
6	A ⊃ (B ⊃ C)	2–5 ⊃I

This derivation contains two subderivations, one nested within the other. The innermost subderivation occupies lines 3–4; the outer subderivation lines 2–5. At line 4 we were able to use Reiteration to derive 'C' because every scope line to the left of 'C' on line 1 (there is only one) is also to the left of the sentence we entered on line 4. And on line 5 we were able to enter 'B ⊃ C' by Horseshoe Introduction because every scope line to the left of 'B ⊃ C' at line 5 (there are two) is also to the left of the *subderivation* occupying lines 3–4. Note that while there are three scope lines to the left of the *sentences* on lines 3 and 4, the rightmost of these is *part of* the subderivation occupying lines 3–4. So there are 2, not 3, scope lines to the left of that subderivation. Neither 'A', the auxiliary assumption that begins the subderivation occupying lines 2–5, nor 'B', the auxiliary assumption that begins the subderivation occupying lines 3–4, was appealed to in deriving the sentences 'B ⊃ C' and 'C' within those subderivations. It is often the case that the auxiliary assumption that begins a subderivation is not appealed to until the subderivation is ended (when it is appealed to as part of the entire subderivation).

The following variant of the previous derivation is also constructed in accordance with the rules of *SD*.

Derive: $A \supset (B \supset C)$

1	C	Assumption
2	| A	A / ⊃I
3	| | B	A / ⊃I
4	| | B & C	1, 3 &I
5	| | C	4 &E
6	| B ⊃ C	3–5 ⊃I
7	A ⊃ (B ⊃ C)	2–6 ⊃I

At line 4 both 'C' on line 1 and 'B' on line 3 are accessible, so our use of Conjunction Introduction is allowed. However, we did not choose to construct this derivation, because it is one line longer than our earlier derivation of 'A ⊃ (B ⊃ C)' from

'C'. On the other hand, the following variation *is not* constructed in accordance with the rules of *SD*.

Derive: $A \supset (B \supset C)$

1	C	Assumption	
2	A	A / ⊃I	
3	B	A / ⊃I	
4	C	1 R	
5	B ⊃ C	3–4 ⊃I	
6	A ⊃ (B ⊃ C)	2–5 ⊃I	
7	B & C	3, 4 &I	**MISTAKE!**

Neither the sentence on line 3 nor that on line 4 is accessible from line 7. There are two scope lines to the left of the sentences on lines 3 and 4 that *do not* extend to the left of the sentence we tried to enter on line 7 (the assumptions on lines 2 and 3 were closed before line 7, so sentences falling within the scope of either assumption cannot be appealed to on line 7).

The remaining four subderivation rules are as follows:

Negation Introduction (~ I)

	P
	Q
	~ Q
▷	**~ P**

Negation Elimination (~ E)

	~ P
	Q
	~ Q
▷	**P**

Disjunction Elimination (∨E)

	P ∨ Q
	P
	R
	Q
	R
▷	**Q**

Biconditional Introduction (≡I)

	P
	Q
	Q
	P
▷	**P ≡ Q**

Negation Introduction specifies that if we can derive a sentence and its negation, **Q** and ~ **Q**, within the scope of an auxiliary assumption **P**, then we may end the subderivation and enter ~ **P** on the following line. Here and with the remaining subderivation rules the template should not be taken as specifying the order in which sentences must be derived within the subderivation.

Negation Elimination specifies that if we can derive a sentence and its negation, **Q** and ~ **Q**, within the scope of an auxiliary assumption ~ **P**, then we may end the subderivation and enter **P** on a subsequent line.

Disjunction Elimination specifies that if **P** ∨ **Q** occurs on an earlier line of a derivation and subsequent to it there are two subderivations, one of **R** from **P** and the other of **R** from **Q**, then **R** may be entered on a subsequent line.

Biconditional Introduction specifies that if the derivation contains two subderivations, one of **Q** from **P** and one of **P** from **Q**, then **P** ≡ **Q** may be entered on a subsequent line.

Negation Introduction and Negation Elimination both parallel a pattern of reasoning we often use in everyday life, *reductio ad absurdum* reasoning. In this reasoning, we make an assumption and then show that an absurd result follows from that assumption and whatever other assumptions we may already have made. To avoid the absurdity we reject one of our assumptions. Here is an example of *reductio ad absurdum* reasoning. Suppose we know the following:

> Billings was shot to death in New York City during the evening hours of October 25. Billings' partner, Jenkins, became sole owner of their company as a result of Billing's death, and Jenkins is in dire financial straits and has always hated his partner.

We want to explore the possibility that Jenkins shot Billings, and we do so by assuming that he did. Further investigation reveals that Jenkins was seen sitting in a *Pizza Uno* restaurant in Chicago the entire evening of the shooting. We now reason as follows:

> Suppose Jenkins shot Billings. Then Jenkins was in New York on the evening of the 25th. But we know he was in Chicago that entire evening, so he was not in New York. Therefore, Jenkins did not shoot Billings.

The assumption that Jenkins shot Billings, along with our knowledge that he was in Chicago the entire evening of October 25, leads to the absurdity the Billings was both in New York City and not in New York City that evening. So we rejected our assumption and concluded that Jenkins did not shoot Billings.

Both Negation Introduction and Negation Elimination mirror this kind of reasoning. In both cases we make an assumption and then derive a sentence and its negation (**Q** and ~ **Q**). It would be absurd to remain committed to both a sentence and its negation. But we are so committed as long as we can derive both. So we reject the assumption that starts the subderivation—we close the subderivation and enter **P** (if our assumption was ~ **P**) or ~ **P** (if our assumption was **P**).

Here are some fairly simple derivations in which Negation Introduction and Negation Elimination are used:

Derive: ~ H

1	H ⊃ F	Assumption
2	~ F	Assumption
3	| H	A / ~ I
4	| F	1, 3 ⊃E
5	| ~F	2 R
6	~ H	3–5 ~ I

Derive: N

1	~ N ⊃ S	Assumption
2	S ⊃ C	Assumption
3	C ⊃ N	Assumption
4	| ~ N	A / ~ E
5	| S	1, 4 ⊃E
6	| C	2, 5 ⊃E
7	| N	3, 6 ⊃E
8	| ~N	4 R
9	N	4–8 ~ E

We noted when we introduced the rule Reiteration that it would often be useful in derivations involving subderivations, and we have used Reiteration in both of these derivations (as we did in an earlier use of Conditional Introduction). Next we present a derivation that uses both Negation Introduction and Negation Elimination:

Derive: ~ A & B

1	~ (A ~∨ B)	Assumption
2	| A	A / ~ I
3	| A ~∨ B	2 ∨I
4	| ~ (A ~∨ B)	1 R
5	~ A	2–4 ~ I
6	| ~ B	A / ~ E
7	| A ~∨ B	6 ∨I
8	| ~ (A ~∨ B)	1 R
9	B	6–8 ~ E
10	~ A & B	5, 9 &I

In this derivation the sentence to be derived is a conjunction, so we opted to derive it by Conjunction Introduction. Having made that decision, we were left

with two goals: deriving '~ A' and deriving 'B'. We derived '~ A' by Negation Introduction and 'B' by Negation Elimination. One key to constructing this derivation was recognizing that our primary assumption, '~ (A ∨ ~ B)', was a negation and hence was a candidate for serving as the negation ~ **Q** that would be needed within both our Negation Introduction and our Negation Elimination subderivations. The other key was recognizing that the sentence **Q** in this case, 'A ∨ ~ B', could be derived by Disjunction Introduction from the assumption 'A' in the first subderivation and from '~ B' in the second.

Disjunction Elimination also parallels a pattern of reasoning we use in everyday life. Here is an example:

> The CEO is incompetent and will either resign or be fired. If she resigns she will move to Boston to be near her son. If she is fired she will move to Boston to live with her son. So the CEO will move to Bostson.

In this example we know the CEO will either resign or be fired. If the first happens, she will end up in Boston, and if the second happens, she will also end up in Boston. So whichever happens, the CEO will end up in Boston. The following derivation formalizes this reasoning:

Derive: B

1	I & (R ∨ F)	Assumption
2	R ⊃ (B & N)	Assumption
3	F ⊃ (B & L)	Assumption
4	R ∨ F	1 & E
5	| R	A / ∨E
6	| B & N	2, 5 ⊃E
7	| B	6 &E
8	| F	A / ∨E
9	| B & L	3, 8 ⊃E
10	| B	9 &E
11	B	4, 5–7, 8–10 ∨E

In this derivation we derived a disjunction on line 4 and then constructed two subderivations. The first has 'R', the left disjunct of 'R ∨ F', as its auxiliary assumption. The subderivation shows that given 'R', 'B' can be derived. The second subderivation shows that 'B' can also be derived from the right disjunct, 'F'. Having derived 'B' from each disjunct, we entered 'B' on line 11. The justification for line 11 cites the disjunction on line 4 and the two subderivations, one occupying lines 5–7 and the other lines 8–10.

Material biconditionals of *SL*, sentences of the form **P** ≡ **Q**, have the force of two material conditionals, (**P** ⊃ **Q**) and (**Q** ⊃ **P**). Hence it should not be surprising that Biconditional Introduction requires two subderivations, one

in which **Q** is derived from **P** and one in which **P** is derived from **Q**. Here is a simple use of Biconditional Introduction:

Derive: A ≡ B

1	A ⊃ B	Assumption
2	B ⊃ A	Assumption
3	| A	A / ≡I
4	| B	1, 3 ⊃E
5	| B	A / ≡I
6	| A	2, 5 ⊃E
7	A ≡ B	3–4, 5–6 ≡I

The following derivation uses both Biconditional Elimination and Biconditional Introduction:

Derive: A ≡ C

1	A ≡ B	Assumption
2	B ≡ C	Assumption
3	| A	A / ≡I
4	| B	1, 3 ≡E
5	| C	2, 4 ≡E
6	| C	A / ≡I
7	| B	2, 6 ≡E
8	| A	1, 7 ≡E
9	A ≡ C	3–5, 6–8 ≡I

5.1.2**E** EXERCISES

1. Complete the following derivations by entering the appropriate justifications:

a. Derive: (A ⊃ B) & (A ⊃ ~ D)

1	A ⊃ (B & ~ D)	Assumption
2	| A	
3	| B & ~ D	
4	| B	
5	A ⊃ B	
6	| A	
7	| B & ~ D	
8	| ~ D	
9	A ⊃ ~ D	
10	(A ⊃ B) & (A ⊃ ~ D)	

*b. Derive: $A \supset [B \supset (C \vee D)]$

1	$(A \,\&\, B) \supset C$	Assumption
2	$\quad A$	
3	$\quad\quad B$	
4	$\quad\quad A \,\&\, B$	
5	$\quad\quad C$	
6	$\quad\quad C \vee D$	
7	$\quad B \supset (C \vee D)$	
8	$A \supset [B \supset (C \vee D)]$	

c. Derive: B

1	$\sim B \supset B$	Assumption
2	$\quad \sim B$	
3	$\quad B$	
4	$\quad \sim B$	
5	B	

*d. Derive: $A \supset \sim B$

1	$A \supset (B \supset C)$	Assumption
2	$\sim C$	Assumption
3	$\quad A$	
4	$\quad B \supset C$	
5	$\quad\quad B$	
6	$\quad\quad C$	
7	$\quad\quad \sim C$	
8	$\quad \sim B$	
9	$A \supset \sim B$	

e. Derive: $E \vee D$

1	$A \vee (B \,\&\, \sim C)$	Assumption
2	$A \supset D$	Assumption
3	$\sim C \supset E$	Assumption
4	$\quad A$	
5	$\quad D$	
6	$\quad E \vee D$	
7	$\quad B \,\&\, \sim C$	
8	$\quad\quad \sim C$	
9	$\quad\quad E$	
10	$\quad\quad E \vee D$	
11	$E \vee D$	

*f. Derive: E

1	$F \supset (\sim G \vee \sim H)$	Assumption
2	$(\sim G \supset E) \ \& \ (\sim E \supset H)$	Assumption
3	F	Assumption
4	$\sim G \vee \sim H$	
5	$\sim G$	
6	$\sim G \supset E$	
7	E	
8	$\sim H$	
9	$\sim E \supset H$	
10	$\sim E$	
11	H	
12	$\sim H$	
13	E	
14	E	

g. Derive: $F \equiv \sim G$

1	$(F \supset \sim G) \ \& \ (\sim G \supset F)$	Assumption
2	F	
3	$F \supset \sim G$	
4	$\sim G$	
5	$\sim G$	
6	$\sim G \supset F$	
7	F	
8	$F \equiv \sim G$	

*h. Derive: $H \equiv J$

1	$(H \ \& \ I) \equiv J$	Assumption
2	$H \equiv I$	Assumption
3	H	
4	I	
5	H & I	
6	J	
7	J	
8	H & I	
9	H	
10	$H \equiv J$	

2. Complete the following derivations.

a. Derive: A ≡ B

1	A	Assumption
2	B	Assumption

*b. Derive: ~ B

1	B ⊃ ~ B	Assumption

c. Derive: A

1	~ ~ A	Assumption

*d. Derive: H & ~ I

1	I & ~ I	Assumption

e. Derive: B

1	~ B ⊃ C	Assumption
2	~ C ≡ A	Assumption
3	A	Assumption

*f. Derive: A ≡ C

1	A ≡ ~ B	Assumption
2	~ B ≡ C	Assumption

g. Derive: ~ H

1	H ⊃ I	Assumption
2	~ I	Assumption

*h. Derive: ~ G

1	~ F ⊃ ~ G	Assumption
2	~ F ∨ H	Assumption
3	H ≡ ~ G	Assumption

i. Derive: ~ (F ∨ G)

1	(F ∨ G) ⊃ (H & I)	Assumption
2	~ H	Assumption

*j. Derive: ~ (F & G)

1	F ≡ (~ G & H)	Assumption

5.1.3 CONCLUDING COMMENTS

All the derivation rules of *SD* have been introduced. We repeat them here, for easy reference. Rather than listing the rules that do not use subderivations separately from those that do use subderivations, we here arrange the derivation rules by the kind of compound that is either appealed to or introduced. The rules can also be found on the inside front cover of this text.

Reiteration (R)

P
▷ **P**

Conjunction Introduction (&I)

P
Q
▷ **P & Q**

Conjunction Elimination (&E)

P & Q	**P & Q**
▷ **P**	▷ **Q**

Conditional Introduction (⊃I)

P	
Q	
▷ **P ⊃ Q**	

Conditional Elimination (⊃E)

P ⊃ Q
P
▷ **Q**

Negation Introduction (~ I)

P	
Q	
~ Q	
▷ **~ P**	

Negation Elimination (~ E)

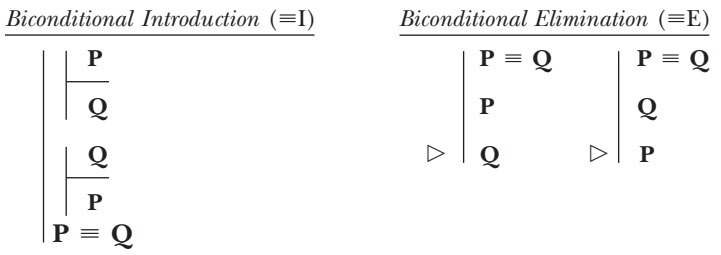
~ P	
Q	
~ Q	
▷ **P**	

Disjunction Introduction (∨I)

P	**P**
▷ **P ∨ Q**	▷ **Q ∨ P**

Disjunction Elimination (∨E)

P ∨ Q		
	P	
	R	
	Q	
	R	
▷ **R**		



We have presented the derivation rules of *SD* and constructed a fair number of derivations. But we haven't actually defined the term 'derivation in *SD*'. We do so now:

> A *derivation in SD* is a series of sentences of *SL,* each of which is either an assumption or is obtained from previous sentences by one of the rules of *SD*.

We will continue to annotate our derivations with line numbers, scope and assumption lines, and line justifications. However, these annotations are not, as the above definition makes clear, officially parts of derivations.

There are many truth-preserving templates we do not include as rules of either *SD* or *SD*+. Why are some included and others not? For *SD* the answer is fairly simple. We want a derivation system to be truth-preserving (include no rule that ever takes us from truths to a falsehood). A system that has this property, never taking us from truths to a falsehood, is said to be **sound.** We also want our derivation systems to be **complete.** A derivation system is complete if and only if every sentence that is truth-functionally entailed by a set of sentences can be derived from that set. *SD* is complete in this sense and it is a fairly minimalist derivation system—it includes only two rules for each connective.[1] *SD*+ will also be complete but includes additional derivation rules, some because they mirror reasoning patterns that are common in everyday discourse, some because they have historically been included in derivation systems. We prove that both *SD* and *SD*+ are complete in Chapter 6.

Before ending this section we will take time to caution against some mistakes that are commonly made while constructing derivations. First, the derivation rules of *SD* are rules of inference, which is to say that when they appeal to a line earlier in the derivation they appeal to the entire sentence on that line, not to a sentence that is a component of a longer sentence. Here is

[1]Two rules of *SD,* Reiteration and Negation Introduction, could be dropped without making the system incomplete. This is not true of any of the other rules of *SD*.

an attempt at a derivation that misuses Conjunction Elimination by appealing to a component of a longer sentence.

Derive A ⊃ C

1	A ⊃ (B & C)	Assumption	
2	A	A / ⊃I	
3	C	1 &E	**MISTAKE!**
4	A ⊃ C	2–3 ⊃I	

The mistake at line 3 results from trying to apply Conjunction Elimination to a component of a longer sentence. The sentence on line 1 is *not* of the form **P** & **Q**, and while a component of that sentence, 'B & C', is of that form, rules of inference work, again, on sentences that are not themselves parts of longer sentences. A correct derivation for this problem is

Derive A ⊃ C

1	A ⊃ (B & C)	Assumption
2	A	A / ⊃I
3	B & C	1, 2 ⊃E
4	C	3 &E
5	A ⊃ C	2–4 ⊃I

The sentence on line 3 is of the form **P** & **Q**. It is not part of a longer sentence on that line. So we can apply Conjunction Elimination to it and obtain 'C' at line 4.

Here is a similar misuse of a derivation rule.

Derive: C

1	B ⊃ (A ⊃ C)	Assumption	
2	A	Assumption	
3	C	1, 2 ⊃E	**MISTAKE!**

Here an attempt has been made to apply Conditional Elimination to a component, 'A ⊃ C' of the longer sentence 'B ⊃ (A ⊃ C)' and this cannot be done. In this case there is no correct derivation. 'C' does not follow from the assumptions on lines 1 and 2.

Another common mistake is to appeal to lines or subderivations that are not accessible. In a derivation a sentence or subderivation is *accessible* at line **n** (it can be appealed to when justifying a sentence on line **n**) if and only if that sentence or subderivation does not lie within the scope of a closed assumption, that is, an assumption that has been discharged prior

to line **n**. Here is an attempt at a derivation that twice violates the accessibility requirement:

Derive: B

1	B	A / ⊃I	
2	A	A / ⊃I	
3	B	1 R	
4	A ⊃ B	2–3 ⊃I	
5	B ⊃ (A ⊃ B)	1–4 ⊃I	
6	A ⊃ B	2–3 ⊃I	**MISTAKE!**
7	B	2, 6 ⊃E	**MISTAKE!**

Line 6 is a mistake because it appeals to a subderivation, that occurring on lines 2 through 3, that is no longer accessible. It is not accessible at line 6, because not every scope line to the left of that subderivation (there are two) continues to line 6. The auxiliary assumption occurring on line 2 was discharged at line 4, when Conditional Introduction was used. (We also cannot use Reiteration to obtain A ⊃ B on line 6, because the sentence on line 4 is inaccessible at that point.) Line 7 is a mistake because it appeals to a line, line 2, which is no longer accessible. Of course, it is also a mistake because it appeals to a line, line 6, which is itself a mistake. In fact, neither line 6 nor line 7 can be derived in a derivation that has no primary assumptions. On the other hand, part of the above attempt, namely the part consisting of lines 1 through 5, is correct, demonstrating that some sentences can be derived starting from no primary assumptions. 'B ⊃ (A ⊃ B)' is one such sentence.

The following derivation is correctly done.

Derive: ~ U ⊃ ~ S

1	~ U ⊃ ~ W	Assumption
2	~ W ⊃ ~ S	Assumption
3	~ U	A / ⊃I
4	~W	1, 3 ⊃E
5	~ S	2, 4 ⊃E
6	~ U ⊃ ~S	3–5 ⊃I

Line 4 cites lines 1 and 3, which are both accessible at line 4. The sentences on lines 1 and 3 do not lie within the scope of an assumption that has been discharged prior to line 4. (Neither the sentence on line 1 nor the sentence on line 3 has a scope line to its left that is not also to the left of the sentence on line 4.) Similarly line 5 cites lines 2 and 4, which are both accessible at line 5. Line 6 cites the subderivation from lines 3–5. This subderivation is accessible at line 6 because the subderivation does not lie within the scope of an assumption that has been closed prior to line 6.

Here is another example in which an inaccessible subderivation is cited:

Derive A ≡ C

1	~ C	Assumption	
2	B ⊃ C	Assumption	
3	~ A & ~ B	Assumption	
4	A	A / ≡I	
5	~ B	A / ~ E	
6	~ A	3 &E	
7	A	4 R	
8	B	5–7 ~E	
9	C	2, 8 ⊃E	
10	C	A / ≡I	
11	~ B ⊃ A	5–7 ⊃I	**MISTAKE!**
12	~ B	3 &E	
13	A	11, 12 ⊃E	
14	A ≡ C	4–9, 10–13 ≡I	

The mistake at line 11 is that of citing a subderivation that is not accessible from line 11. That it is not accessible is indicated by there being a scope line to the left of the subderivation, the scope line running from line 4 through line 9, that is not to the left of the sentence entered at line 11. More substantively, 'A' was derived at line 7 by Reiteration on line 4. The assumption at line 4 is not accessible at line 11, and neither are results obtained while it was available.

In fact, it is possible to derive 'A ≡ C' from the above primary assumptions. Here is a derivation that does so.

Derive A ≡ C

1	~ C	Assumption	
2	B ⊃ C	Assumption	
3	~ A & ~ B	Assumption	
4	A	A / ≡I	
5	~ C	A / ~ E	
6	~ A	3 &E	
7	A	4 R	
8	C	5–7 ~ E	
9	C	A / ~ I	
10	~ A	A / ~ E	
11	C	9 R	
12	~ C	1 R	
13	A	10–12 ~ E	
14	A ≡ C	4–8, 9–13 ≡I	

It is possible to use a single auxiliary assumption to generate a subderivation that allows the use of two different subderivation rules. Here is such a case:

Derive: C & (A ⊃ C)

1	A ∨ B	Assumption
2	A ⊃ D	Assumption
3	B ⊃ D	Assumption
4	~ C ⊃ ~ D	Assumption
5	A	A / ∨E / ⊃I
6	~ C	A / ~ E
7	~ D	4, 6 ⊃E
8	D	2, 5 ⊃E
9	C	6–8 ~ E
10	B	A / ∨E
11	~ C	A / ~ E
12	~ D	4, 11 ⊃E
13	D	3, 10 ⊃E
14	C	11–13 ~ E
15	C	1, 5–9, 10–14 ∨E
16	A ⊃ C	5–9 ⊃I
17	C & (A ⊃ C)	15, 16 &I

Notice that the subderivation occupying lines 5 through 9 is cited twice, once as part of an application of the rule Disjunction Elimination (at line 15) and once as the basis for entering a conditional at line 16. In the present case it is unlikely that when the assumption at line 5 is made it was foreseen that the subderivation to be constructed would be used in both of the above indicated ways. So most likely at the time the assumption was made the only notation entered in the justification column was 'A / ∨E'. It is only after reaching 'C' at line 15 and wondering how 'A ⊃ C' can be obtained that it became apparent that work already done, the subderivation on lines 5 through 9, could be reused. So the extra notation '/ ⊃I' was added to line 5 when line 16 was entered.

In the above example identical subderivations occur on lines 6 through 8 and lines 11 through 13. We had to do this work twice because when trying to get from 'B' at line 10 to 'C' on a subsequent line the subderivation occupying lines 6 through 8 is no longer accessible.

Finally, it is possible to end a subderivation at any time, without using one of the introduction rules that requires a subderivation. This is likely to occur when one decides the strategy being pursued is unproductive and simply abandons the work done within the subderivation. Here is an example:

Derive: A ⊃ (B ⊃ A)

1	A	A / ⊃I
2	~ (B ⊃ A)	A / ~ E
3	A	1 R
4	B	A / ⊃I
5	A	1 R
6	B ⊃ A	4–5 ⊃I
7	A ⊃ (B ⊃ A)	1–6 ⊃I

Here the subderivation on lines 2–3 is in effect wasted work, work we have thrown away. It does no harm, but neither does it do any good.

5.1.3**E** EXERCISES

1. Complete each of the following derivations by entering the appropriate justifications.

a. Derive: (A & C) ∨ (B & C)

1	(A ∨ B) & C
2	A ∨ B
3	C
4	A
5	A & C
6	(A & C) ∨ (B & C)
7	B
8	B & C
9	(A & C) ∨ (B & C)
10	(A & C) ∨ (B & C)

*b. Derive: A ⊃ (B ⊃ C)

1	(A & B) ⊃ C
2	A
3	B
4	A & B
5	C
6	B ⊃ C
7	A ⊃ (B ⊃ C)

c. Derive: ~ B

1	B ⊃ (A & ~ B)
2	B
3	A & ~ B
4	~ B
5	B
6	~ B

*d. Derive: A ⊃ B

1	(A & ~ B) ⊃ (~ B & C)
2	C ⊃ ~ A
3	A
4	~ B
5	A & ~ B
6	~ B & C
7	C
8	~ A
9	A
10	B
11	A ⊃ B

e. Derive: $C \supset (\sim A \mathbin{\&} B)$

1		$\sim D$
2		$C \supset (A \equiv B)$
3		$(D \vee B) \supset \sim A$
4		$(A \equiv B) \supset (D \mathbin{\&} E)$
5		$\sim B \supset D$
6	|	C
7	|	$A \equiv B$
8	|	$D \mathbin{\&} E$
9	|	D
10	|	$D \vee B$
11	|	$\sim A$
12	| |	$\sim B$
13	| |	D
14	| |	$\sim D$
15	|	B
16	|	$\sim A \mathbin{\&} B$
17		$C \supset (\sim A \mathbin{\&} B)$

*f. Derive: $A \supset (B \vee C)$

1		$(\sim B \mathbin{\&} \sim C) \supset \sim A$
2	|	A
3	| |	$\sim (B \vee C)$
4	| | |	B
5	| | |	$B \vee C$
6	| | |	$\sim (B \vee C)$
7	| |	$\sim B$
8	| | |	C
9	| | |	$B \vee C$
10	| | |	$\sim (B \vee C)$
12	| |	$\sim C$
13	| |	$\sim B \mathbin{\&} \sim C$
14	| |	$\sim A$
15	| |	A
16	|	$B \vee C$
17		$A \supset (B \vee C)$

g. Derive: $A \equiv B$

1		$\sim A \mathbin{\&} \sim B$
2	|	A
3	| |	$\sim B$
4	| |	$\sim A$
5	| |	A
6	|	B
7	|	B
8	| |	$\sim A$
9	| |	B
10	| |	$\sim B$
11	|	A
12		$A \equiv B$

*h. Derive: $A \equiv (B \vee C)$

1		$(A \equiv B) \mathbin{\&} (A \equiv C)$
2	|	A
3	|	$A \equiv B$
4	|	B
5	|	$B \vee C$
6	|	$B \vee C$
7	| |	B
8	| |	$A \equiv B$
9	| |	A
10	| |	C
11	| |	$A \equiv C$
12	| |	A
13	|	A
14		$A \equiv (B \vee C)$

5.2 BASIC CONCEPTS OF *SD*

We now define the key concepts of *SD*. These are all syntactical concepts as each is defined by reference to there being a derivation of a certain sort—no reference is made in any of these definitions either to truth-values or to truth-value assignments.

> *Derivability:* A sentence **P** of *SL* is *derivable in SD* from a set Γ of sentences of *SL* if and only if there is a derivation in *SD* in which all the primary assumptions are members of Γ and **P** occurs in the scope of only those assumptions.
>
> *Valid in SD:* An argument of *SL* is *valid in SD* if and only if the conclusion of the argument is derivable in *SD* from the set consisting of the premises. An argument of *SL* is *invalid in SD* if and only if it is not valid in *SD*.
>
> *Theorem in SD:* A sentence **P** of *SL* is a *theorem in SD* if and only if **P** is derivable in *SD* from the empty set.
>
> *Equivalence in SD:* Sentences **P** and **Q** are *equivalent in SD* if and only if **Q** is derivable in *SD* from {**P**} and **P** is derivable in *SD* from {**Q**}.
>
> *Inconsistency in SD:* A set Γ of sentences of *SL* is *inconsistent in SD* if and only if there is a sentence **P** such that both **P** and ~ **P** are derivable in *SD* from Γ. A set Γ is *consistent in SD* if and only if it is not inconsistent in *SD*.

A few additional notational conventions will be useful. We will use the single turnstile, '⊢' to assert derivability, and will read

$$\Gamma \vdash \mathbf{P}$$

as '**P** is derivable from Γ'. We will read 'Γ ⊬ **P**' as '**P** is not derivable from Γ'. This parallels our use of the double turnstile in previous chapters, where we read

$$\Gamma \models \mathbf{P}$$

as 'Γ truth-functionally entails **P**' and 'Γ ⊭ **P**' as 'Γ does not truth-functionally entail **P**'. The parallelism is for good reason. It will turn out that for any finite set Γ of sentences of *SL* and any sentence **P** of *SL,*

$\Gamma \vdash \mathbf{P}$ in *SD* if and only if $\Gamma \models \mathbf{P}$.

This is a key claim of metatheory that we prove in Chapter 6. Finally, we will read

$$\vdash \mathbf{P}$$

as '**P** is a theorem'. This notation derives from '$\varnothing \vdash$ **P**', which is read '**P** is derivable from the empty set'. And of course a sentence of *SL* is a theorem of *SD* if and only if it is derivable in *SD* from the empty set. We will also refer to a derivation of a sentence of *SL* from no primary assumptions as a **proof** of the theorem that is the last line of that derivation.

The careful reader will recall that there are seven key semantical concepts of *SL:* Truth-functional consistency, truth-functional truth, truth-functional falsity, truth-functional indeterminacy, truth-functional equivalence, truth-functional validity, and truth-functional entailment. We have syntactic parallels for only five of those concepts. These pair up as follows:

Truth-functional consistency	Consistency in *SD*
Truth-functional truth	Theorem in *SD*
Truth-functional equivalence	Equivalence in *SD*
Truth-functional validity	Valid in *SD*
Truth-functional entailment	Derivability in *SD*

There is no syntactic counterpart to either truth-functional falsity or truth-functional indeterminacy. Introducing such counterparts is easy enough—we could define an *anti-theorem* of *SD* as a sentence **P** of *SL* whose negation, ~ **P**, is a theorem of *SD*. And we could take a sentence **P** of *SL* to be *syntactically undetermined* in *SD* if and only if neither it nor its negation is a theorem of *SD*. We would then have syntactic counterparts to all seven central semantic concepts, but historically logicians have never felt the need to add these or equivalent definitions. We will follow their lead.

Below we construct a derivation that establishes that the following simple argument is valid in *SD:*

$A \supset B$
$\sim B$

$\sim A$

Derive: $\sim A$

1	$A \supset B$	Assumption
2	$\sim B$	Assumption
3	$\quad A$	A /~ I
4	$\quad B$	1, 3 ⊃E
5	$\quad \sim B$	2 R
6	$\sim A$	3–5 ~ I

This derivation establishes that the above argument is valid in *SD*. (The conclusion of the argument has been derived from the set consisting of the premises of the argument.)

On the other hand, the following *does not* establish the validity of the above argument:

Derive: ~ A

1	A ⊃ B	Assumption
2	~ B	Assumption
3	~ A	A
4	~ A	3 R

Here '~ A', the conclusion of the argument, has not been derived from the set consisting of the premises of the argument. Rather, it has been derived from those sentences and '~ A'—that is, from the primary assumptions and an auxiliary assumption. We have not shown that '~ A' is derivable from the set consisting of the premises A ⊃ B and ~ A.

Note that no notation has been made on line 3 as to the reason for assuming '~ A'. Someone constructing a derivation such as this may well have reasoned "I want to obtain '~ A'. Since I can assume anything, I will assume what I want, namely '~ A', and then use Reiteration to derive my goal, '~ A'." It is true that any sentence of *SL* can be assumed at any time. But there is no point to assuming a sentence unless one has a rule in mind for discharging that assumption. This is why we require the justification column for auxiliary assumptions to include both the indication that the sentence just entered is an assumption ('A') and an indication of what rule will be used to discharge the assumption. There are only five rules (Conditional Introduction, Disjunction Elimination, Negation Introduction, Negation Elimination, and Biconditional Introduction) that require the construction of a subderivation. These are also the only rules that involve discharging an assumption. Requiring a notation that indicates what rule will be used to discharge an assumption largely prevents the making of assumptions that do not serve a strategic purpose.

A theorem of *SD* is a sentence of *SL* that can be derived from no primary assumptions. A derivation of such a sentence is said to be a **proof** of that sentence. Here is a proof of the theorem '[A ⊃ (B ⊃ C)] ⊃ [(A & B) ⊃ C]':

Derive: [A ⊃ (B ⊃ C)] ⊃ [(A & B) ⊃ C]

1	A ⊃ (B ⊃ C)	A / ⊃I
2	A & B	A / ⊃I
3	A	2 &E
4	B ⊃ C	1, 3 ⊃E
5	B	2 &E
6	C	4, 5 ⊃E
7	(A & B) ⊃ C	2–6 ⊃I
8	[A ⊃ (B ⊃ C)] ⊃ [(A & B) ⊃ C]	1–7 ⊃I

There are no primary assumptions in this derivation, and every auxiliary assumption has been discharged. The sentence '[A ⊃ (B ⊃ C)] ⊃ [(A & B) ⊃ C]' on the last line does not lie within the scope of any assumption. Hence it has been derived from the empty set and is a theorem of *SD*.

As one would expect, the sentences 'A ≡ B' and 'B ≡ A' are equivalent in *SD*, as the following two derivations show. Establishing the equivalence in *SD* of two distinct sentences of *SL* requires *two* derivations because we must establish that each sentence is derivable from the unit set of the other.

Derive: B ≡ A

1	A ≡ B	Assumption
2	| B	A / ≡I
3	| A	1, 2 ≡E
4	| A	A / ≡I
5	| B	1, 4 ≡I
6	B ≡ A	2–3, 4–5 ≡I

Having derived 'B ≡ A' from 'A ≡ B', we next derive 'A ≡ B' from 'B ≡ A'.

Derive: A ≡ B

1	B ≡ A	Assumption
2	| A	A / ≡I
3	| B	1, 2 ≡E
4	| B	A / ≡I
5	| A	1, 4 ≡I
6	A ≡ B	2–3, 4–5 ≡I

These two derivations establish that 'A ≡ B' and 'B ≡ A' are equivalent in *SD*.

When **P** and **Q** are distinct sentences we need two derivations to show that they are equivalent, one of **Q** from {**P**} and one of **P** from {**Q**}. But when **P** and **Q** are identical, the same sentence, we need only one derivation to show they are equivalent (the one sentence is equivalent to itself) because in this case the derivation of **Q** from {**P**} is also a derivation of **P** from {**Q**}. A sentence can be derived from its own unit set in just one step, using Reiteration:

1	**P**	Assumption
2	**P**	1 Reiteration

Derivations are unlike truth-tables and truth-trees in two important respects. First, when one of the syntactic properties we have defined holds (for a sentence, a pair of sentences, an argument, etc.) there is a derivation that demonstrates that this property holds. For example, if an argument is valid in *SD* it is the existence of a derivation of the conclusion of the argument from the set consisting of the argument's premises that makes this so. But if an argument is invalid in *SD* there is no derivation that demonstrates this. Rather, it is the *absence* of a derivation that makes an argument invalid in *SD*. While one can use the derivation system *SD* to show that there is a derivation of a certain sort (by producing such a derivation), one cannot use it to show that there is no derivation of a certain sort. No number of unsuccessful attempts to construct a derivation of a certain sort proves that there is no such derivation. Hence, the system *SD* can be used to establish validity in *SD,* but not invalidity. So too for equivalence in *SD,* inconsistency in *SD,* and theoremhood in *SD*. That is, one cannot use the system *SD* to prove that the members of a pair of sentences are not equivalent in *SD,* that a set is consistent in *SD,* or that a sentence is not a theorem in *SD*. In this way the derivation system is unlike truth-tables and truth-trees, for those procedures are able to establish, for each key semantic concept of *SL,* whether that concept holds or does not hold for a sentence or set of sentences of *SL*.

A second important difference between truth-tables and truth-trees and derivations is that while it is fairly easy to see how an explicit procedure can be developed for constructing truth-tables and truth-trees such that following the procedure does not call for making any choices and always results in a truth-table or truth-tree that yields an answer to the question being asked (*e.g.,* is this set truth-functionally consistent?), it is considerably harder to specify such an explicit procedure for constructing derivations. Procedures that do determine every step of the construction process, whether for truth-tables, trees, or derivations, are said to be **mechanical** procedures. While mechanical procedures for constructing derivations in systems like *SD* (derivation systems for sentential logic)—procedures that will always produce a derivation of a certain sort when one does exist—have been formulated, they are very complex and we will make no attempt to present such a procedure here.[2] There are thus two ways in which one's efforts to construct a derivation of a certain sort might end in frustration—where there is no such derivation and where there is one but all attempts one makes to find it fail. Of course these are very different situations; the first results from trying to do what is impossible, the second from failing to find a solution that does exist.

While we will not present a mechanical procedure for constructing derivations we will provide some useful strategies, strategies that can help avoid

[2]These procedures are generally called theorem provers because what the procedure does, in the first instance, is give mechanical instructions for constructing a proof of a theorem. These procedures are very complicated. It is also important to note that such procedures, when applied to a sentence that is not a theorem of the system, will produce no result that shows the sentence in question is not a theorem.

frustration of the second sort just alluded to. The overarching strategy is that of goal analysis. In every derivation the goal is to derive a sentence, or sentences, from primary assumptions where there are such, otherwise from no assumptions. Goal analysis is the process of determining how a goal sentence can be derived, and involves working backward from the intended last line of the derivation as well as forward from the primary assumptions, if any, of the derivation.

No matter what the goal sentence is, the derivation step that produces that sentence might be the application of any of the elimination rules. To see this one need only remember that the elimination rules tell us nothing about the derived sentence—in each case it might be an atomic sentence, a conjunction, a disjunction, a conditional, a negation, or a biconditional. On the other hand, the introduction rules do tell us a lot about the sentence derived by using one of these rules. First, atomic sentences cannot be derived by using an introduction rule, for all such rules produce truth-functionally compound sentences. Second, we know, for each introduction rule, what the main connective is of a sentence obtained by that rule. Conjunction Introduction produces conjunctions, Disjunction Introduction disjunctions, and so on.

The first step in goal analysis is therefore to determine what kind of a sentence the goal sentence is. If it is an atomic sentence it must be obtained by one of the elimination rules (or by Reiteration). If it is a truth-functional compound sentence it might be obtained by any of the elimination rules or by the appropriate introduction rule, namely the introduction rule that produces sentences whose main connective is the main connective of the goal sentence. The bottom line, of course, is that there will always be multiple ways in which the goal sentence might be derived. But some ways will generally be more plausible than others, as we will soon see.

Having picked one way in which a goal sentence can be obtained, the next step is to determine whether this way of obtaining the goal sentence generates one or more new goal sentences, and then to ask of each of these how they might be obtained. The idea is that eventually the rule picked as a way of obtaining the current goal can be applied directly to currently available sentences, thus completing the derivation. Multiple examples will, we hope, make all of this much clearer.

We here enumerate the strategies we will use throughout the rest of this chapter:

- If the current goal sentence can be obtained by Reiteration, use that rule, otherwise
- If the current goal sentence can be obtained by using a non-subderivation rule, or a series of such rules, do so; otherwise
- Try to obtain the goal sentence by using an appropriate subderivation rule.
- When using a negation rule, try to use an already accessible negation (if there is one) as the ~ Q that the negation rules require be derived.

Most of the derivations we will construct and most of the derivations called for by exercises will involve using multiple strategies, as most will involve deriving one or more subgoals before the final goal can be derived. In practice, this means that most of the time in constructing derivations we work both from the bottom up and from the top down. That is, we work from the bottom up by noting what sentence or sentences we need to obtain before we can obtain the sentence we are trying to derive, and make them subgoals. We work from the top down by deriving from accessible lines and subderivations sentences that will be useful in obtaining our final goal sentence. We will shortly work through the construction of derivations that illustrate this process. Finally, it will often be the case that two or more strategies appear to be viable ways of obtaining a goal sentence. For example, if the current goal sentence is a material conditional and one of the accessible sentences is a disjunction, then both Disjunction Elimination and Conditional Introduction suggest themselves as possible strategies. Rather than puzzling over which strategy is most likely to succeed or which will produce the shortest derivation it is often wise to just pick one and pursue it.

Suppose we are trying to derive '(A & B) ⊃ C' from {A ⊃ C}. The derivation will obviously have just one primary assumption. So we start work as follows:

Derive: (A & B) ⊃ C

1	A ⊃ C	Assumption
G	(A & B) ⊃ C	

Our current goal is the sentence '(A & B) ⊃ C'. We have indicated this by writing 'G' where a line number will eventually be placed. We will follow this convention, of indicating goal sentences by writing 'G' where the number of the line will eventually be, throughout the rest of this section. Readers should follow this convention when constructing their own derivations only if they are working in pencil and can erase these goal sentence markers and replace them with line numbers as appropriate. We write this goal sentence a substantial distance below the primary assumptions, because we do not know, at this stage, how many steps it will take to derive this sentence. At this early stage we know neither the line number nor the justification for the final line of the derivation. We note that the goal sentence is a material conditional. Hence, in principle it could come by any one of the elimination rules, by Reiteration, or by Conditional Introduction. Reiteration is not plausible, as the goal sentence is not among the primary assumptions (there is only one). An elimination rule is not a likely way of generating the goal sentence because the only accessible

sentence is the conditional on line 1 and Conditional Elimination requires that we have both a conditional and the antecedent of that conditional. In this case we do not have the antecedent of 'A ⊃ C', and even if we did the result of applying Conditional Elimination would be 'C', not '(A & B) ⊃ C'. So Conditional Introduction seems to be the most likely rule to have produced our goal sentence. We now note that to use Conditional Introduction we need a subderivation whose assumption is the antecedent of our goal sentence, namely 'A & B', and we need to derive the consequent of our goal sentence, 'C', within the scope of that assumption. That is, we know our derivation will look like this:

Derive: (A & B) ⊃ C

1	A ⊃ C	Assumption
2	A & B	A / ⊃I
G	C	
G	(A & B) ⊃ C	2–__ ⊃I

We still do not know the line number of the last line of our derivation, but we do know we will use Conditional Introduction to obtain it and that we will cite a subderivation that begins on line 2. We note this in the justification column for the last line by entering '2–__ ⊃I' where the underscore marks the space where we will subsequently enter the number of the preceding line. We also know that line 2 will be an auxiliary assumption made for the purpose of doing Conditional Introduction. We are now in a position to stop wondering how '(A & B) ⊃ C' will be obtained. We have a strategy for obtaining that sentence, Conditional Introduction. Accordingly we now switch our focus to how we can complete the subderivation we have started. That is, how can we get from our two assumptions, one primary and one auxiliary, to 'C'? 'C' is an atomic sentence, so we know we will not use an introduction rule to obtain this sentence. Nor will Reiteration generate 'C'. So we are left with the elimination rules. Which elimination rule seems promising? Here it is important to learn to "see" what is available to us at this point in our work. We have two sentences to work from, 'A ⊃ C' and 'A & B'. We want 'C'. We know that 'C' can be obtained from 'A ⊃ C' by Conditional Elimination *if* we have 'A'. We do not currently have 'A'. But we do have 'A & B', and 'A' can be obtained from 'A & B' by Conjunction Elimination. So we now see a path to the completion of our derivation:

Derive: (A & B) ⊃ C

1	A ⊃ C	Assumption
2	A & B	A / ⊃I
3	A	2 &E
4	C	1, 3 ⊃E
5	(A & B) ⊃ C	2–4 ⊃I

We will spend the rest of this section illustrating how the strategies we have enumerated can be used to construct derivations. We will first construct derivations that establish validity in *SD*, then ones that establish that a sentence is a theorem in *SD*, then ones that establish the equivalence in *SD* of a pair of sentences, and finally ones that establish inconsistency in *SD*. We again note that while derivations can be used to establish such results, they cannot be used to establish that an argument is invalid in *SD*, that a pair of sentences are not equivalent in *SD*, or that a set of sentences is consistent in *SD*. Nor, except in special cases, can derivations be used to show that a sentence is not a theorem in *SD*.

ARGUMENTS

Consider next the following argument.

$$\begin{array}{l} \sim \mathrm{N} \\ (\sim \mathrm{N} \supset \mathrm{L}) \ \& \ [\mathrm{D} \equiv (\sim \mathrm{N} \vee \mathrm{A})] \\ \hline \mathrm{L} \ \& \ \mathrm{D} \end{array}$$

To show that this argument is valid in *SD* we need to derive the conclusion from the set consisting of the premises. So we start as follows:

Derive: L & D

1	~ N	Assumption
2	(~ N ⊃ L) & [D ≡ (~ N ∨ A)]	Assumption
G	L & D	__, __ &I

Our goal is a conjunction. It seems unlikely that it will be obtained by an elimination rule, in part because 'L & D' does not occur as a component of any accessible sentence. An introduction rule seems more promising, and since the main connective of our goal sentence is '&' it is Conjunction Introduction that seems most promising. We have noted this by writing '&I' in the justification column for our goal sentence, and we have indicated with two underscores that two line numbers will need to be supplied later. If we are to use Conjunction Introduction we will need to have the two conjuncts 'L' and 'D' available on accessible earlier lines. So we now add two **subgoals** to our derivation structure:

Derive: L & D

1	~ N	Assumption
2	(~ N ⊃ L) & [D ≡ (~ N ∨ A)]	Assumption
G	L	
G	D	
G	L & D	—, — &I

If we can obtain both 'L' and 'D' we can use Conjunction Introduction to obtain 'L & D'. Our new goal sentences, 'L' and 'D' are both atomic sentences, so neither will come by an introduction rule. We note that 'L' occurs as the consequent of a conditional embedded in our second primary assumption. If we could get that conditional, '~ N ⊃ L', out of line 2 we could obtain 'L' by Conditional Elimination, as we do have the antecedent of that conditional '~ N' at line 1. Conjunction Elimination does allow us to extract '~ N ⊃ L' from line 2:

Derive: L & D

1	~ N	Assumption
2	(~ N ⊃ L) & [D ≡ (~ N ∨ A)]	Assumption
3	~ N ⊃ L	2 &E
4	L	1, 3 ⊃E
G	D	
G	L & D	4, — &I

The remaining task, then, is to obtain 'D'. We note that this sentence occurs in the biconditional embedded in line 2. Since the main connective of the sentence on line 2 is '&', we can obtain the biconditional by Conjunction Elimination. To get 'D' from that biconditional we can use Biconditional Elimination, if we have '~ N ∨ A'. This reasoning allows us to add the following steps to our derivation:

Derive: L & D

1	~ N	Assumption
2	(~ N ⊃ L) & [D ≡ (~ N ∨ A)]	Assumption
3	~ N ⊃ L	2 &E
4	L	1, 3 ⊃E
5	D ≡(~ N ∨ A)	2 &E
G	~ N ∨ A	
G	D	5, — ≡E
G	L & D	4, — &I

Note that we have added '~ N ∨ A' as a new goal sentence. The main connective of this sentence is '∨', so if we had either '~ N' or 'A' we could obtain our current goal by Disjunction Introduction. As it happens, we do have '~ N'—it occurs as a primary assumption on line 1. So we can now complete our derivation.

Derive: L & D

1	~ N	Assumption
2	(~ N ⊃ L) & [D ≡ (~ N ∨ A)]	Assumption
3	~ N ⊃ L	2 &E
4	L	1, 3 ⊃E
5	D ≡ (~ N ∨ A)	2 &E
6	~ N ∨ A	1 ∨I
7	D	5, 6 ≡E
8	L & D	4, 7 &I

We will next show that the following argument is valid in *SD* by deriving its conclusion from the set consisting of its premises.

~ A ∨ B
~ A ⊃ B
B ≡ C

C

We begin as always, by taking the premises as primary assumptions and making the conclusion our primary goal.

Derive: C

1	~ A ∨ B	Assumption
2	~ A ⊃ B	Assumption
3	B ≡ C	Assumption
G	C	

After some reflection, two strategies suggest themselves: using Negation Elimination to obtain 'C' and using Disjunction Elimination to obtain 'C'. Both will, in the end, work. We choose to use Disjunction Elimination.

Derive: C

1	~ A ∨ B	Assumption
2	~ A ⊃ B	Assumption
3	B ≡ C	Assumption
4	~ A	A / ∨E
G	C	
	B	A / ∨E
G	C	
G	C	1, 4–__, __–__ ∨E

Our strategy, as the above schema indicates, is to show that the conclusion of the argument, 'C', can be derived from each disjunct of '~ A ∨ B', and hence that 'C' itself can be obtained by Disjunction Elimination. Completing the second subderivation is trivial, for 'C' can be obtained from line 3 and our second auxiliary assumption by Biconditional Elimination.

Derive: C

1	~ A ∨ B	Assumption
2	~ A ⊃ B	Assumption
3	B ≡ C	Assumption
4	~ A	A / ∨E
G	C	
	B	A / ∨E
G	C	3, __ ≡E
G	C	1, 4–__, __–__ ∨E

Completing the first subderivation is only slightly more challenging. From lines 4 and 2 we can obtain 'B' by Conditional Elimination. And we can then use Biconditional Elimination to obtain 'C'.

Derive: C

1	~ A ∨ B	Assumption
2	~ A ⊃ B	Assumption
3	B ≡ C	Assumption
4	~ A	A / ∨E
5	B	2, 4 ⊃E
6	C	3, 5 ≡E
7	B	A / ∨E
8	C	3, 7 ≡E
9	C	1, 4–6, 7–8 ∨E

THEOREMS

Next we will construct proofs of several theorems. We start with a very obvious theorem, 'A ∨ ~ A', whose proof is not obvious. Our task is to derive this sentence using no primary assumptions.

Derive: A ∨ ~ A

1	
G	A ∨ ~ A

Our goal is 'A ∨ ~ A' and here it should be obvious that though this sentence is a disjunction we will not be able to obtain it by Disjunction Introduction. Neither 'A' nor '~ A' is a theorem, and neither can be derived given no primary assumptions. So the only sensible strategy is to use Negation Elimination.

Derive: A ∨ ~ A

1	~ (A ∨ ~ A)	A / ~ E	
G	A ∨ ~ A	1–__ ~ E	

Note that the only accessible sentence, the sentence on line 1, is a negation. There is no rule of *SD* that allows us to 'take apart' a negation. In the present context, we can use Reiteration on line 1, but there is little else we can do with it. Fortunately, this will be useful. Our current strategy is to use Negation Elimination and to do so we need to derive a sentence and its negation. So we will use the assumption on line 1 as the negation and make 'A ∨ ~ A' our new goal.

Derive: A ∨ ~ A

1	|	~ (A ∨ ~ A)	A / ~ E
G	|	A ∨ ~ A	
	|	~ (A ∨ ~ A)	1 R
G	A ∨ ~ A		1–__ ~ E

We noted above that obtaining the last line of our derivation by Disjunction Introduction will not work because neither 'A' nor '~ A' is a theorem. But our current goal, which is the same sentence as that occurring on the last line of the derivation, is to be obtained with the help of the auxiliary assumption '~ (A ∨ ~ A)', and here it is reasonable to hope to use Disjunction Introduction. We will make 'A' our new goal and try to derive it by Negation Elimination.

Derive: A ∨ ~ A

1	|	~ (A ∨ ~ A)	A / ~ E
2	|	~A	A / ~ E
G	|	A	2–__ ~E
G	|	A ∨ ~ A	— ∨I
G	|	~ (A ∨ ~ A)	1 R
G	A ∨ ~ A		1–__ ~ E

One of the points we have emphasized is that when using a Negation Elimination subderivation it is wise to use as the ~ **Q** a negation that is readily available. In the present instance two negations are readily available, '~ A' and '~ (A ∨ ~ A)'. There may be a temptation to select '~ A' as ~ **Q**. But this would be a mistake, for doing so would require that **Q** be 'A' and that sentence is not readily derived from the available assumptions. (We should take a hint from the fact that the point of our current subderivation is to obtain 'A'. If there were an easy way to obtain it we would not be involved

in the current Negation Elimination subderivation.) But if we take ~ **Q** to be '~ (A ∨ ~ A)' then our new goal becomes 'A ∨ ~ A' and this sentence is readily derived—by applying Disjunction Introduction to line 2. We are now able to complete the derivation.

Derive: A ∨ ~ A

1	| |	~ (A ∨ ~ A)	A / ~ E
2	| | |	~ A	A / ~ E
3	| | |	A ∨ ~ A	2 ∨I
4	| | |	~ (A ∨ ~ A)	1 R
5	| |	A	2–4 ~ E
6	| |	A ∨ ~ A	5 ∨I
7	| |	~ (A ∨ ~ A)	1 R
8	|	A ∨ ~ A	1–7 ~ E

Next we will prove the theorem '~ (A ∨ B) ≡ (~ A & ~ B)'. This theorem is a biconditional, so it is plausible the last line will come from Biconditional Introduction, and that rule requires two subderivations, one in which we derive '~ A & ~ B' from {~ (A ∨ B)} and the other in which we derive '~ (A ∨ B)' from {~ A & ~ B} .

Derive: ~ (A ∨ B) ≡ (~ A & ~ B)

1	| |	~ (A ∨ B)	A / ≡I
	| |		
	| |		
G	| |	~ A & ~ B	
	| |	~ A & ~ B	A / ≡I
	| |		
	| |		
G	| |	~ (A ∨ B)	
G	|	~(A ∨ B) ≡ (~ A & ~ B)	1–__, __–__ ≡I

We now have two goals, '~ A & ~ B' in the first subderivation and '~ (A ∨ B)' in the second subderivation. We will work on the upper subderivation first. Since our goal is a conjunction, we will take as new subgoals the two conjuncts of that conjunction, '~ A' and '~ B', and attempt to derive each by Negation Introduction.

Derive: ~ (A ∨ B) ≡ (~ A & ~ B)

1	~ (A ∨ B)	A / ≡I
2	A	A / ~ I
3	A ∨ B	2 ∨I
4	~ (A ∨ B)	1 R
5	~ A	2–4 ~ I
6	B	A / ~ I
7	A ∨ B	6 ∨I
8	~ (A ∨ B)	1 R
9	~ B	6–8 ~ I
10	~ A & ~ B	5, 9 &I
11	~ A & ~ B	A / ≡I
G	~ (A ∨ B)	
G	~ (A ∨ B) ≡ (~ A & ~ B)	1–10, 11–__ ≡I

Note that within the first of our two main subderivations we twice use Negation Introduction, and in each case use 'A ∨ B' and '~ (A ∨ B)' as **Q** and ~ **Q**.

Completing our second main subderivation requires deriving '~ (A ∨ B)', and this invites a Negation Introduction subderivation, giving us a new assumption, 'A ∨ B', which in turn invites a Disjunction Elimination strategy:

11	~ A & ~ B	A / ≡I
12	A ∨ B	A / ~ I
13	A	A / ∨E
	B	
G	~ (A ∨ B)	12–__ ~ I
G	~ (A ∨ B) ≡ (~ A & ~ B)	1–10, 11–__ ≡I

The question now is what sentence we want to play the role of '**R**' in our Disjunction Elimination subderivation. We need a sentence and its negation to make our Negation Introduction subderivation, begun at line 12, work. Two negations are readily available, '~ A' and '~ B'. So we will arbitrarily select one of these, say '~ B', and then make 'B' the sentence we try to obtain by Disjunction Elimination:

11	~ A & ~ B	A / ≡I
12	A ∨ B	A / ~ I
13	A	A / ∨E
G	B	
	B	A / ∨E
G	B	__ R
G	B	12, 13–__, __–__ ∨E
G	~ B	11 &E
G	~ (A ∨ B)	12–__ ~ I
G	~ (A ∨ B) ≡ (~ A & ~ B)	1–10, 11–__ ≡I

We now have two subderivations to complete. The second is, in fact, already complete, for it involves deriving 'B' from an auxiliary assumption of 'B', so Reiteration will accomplish the task. The first involves deriving 'B' from the assumptions on lines 11 through 13. Fortunately a sentence, 'A', and its negation, '~ A' are both readily available. So Negation Elimination will yield the desired result:

11	~ A & ~ B	A / ≡I
12	A ∨ B	A / ~ I
13	A	A / ∨E
14	~ B	A / ~ E
15	~ A	11 &E
16	A	13 R
17	B	14–16 ~ E
18	B	A / ∨E
19	B	18 R
20	B	12, 13–17, 18–19 ∨E
21	~ B	11 &E
22	~ (A ∨ B)	12–21 ~ I
23	~ (A ∨ B) ≡ (~ A & ~ B)	1–10, 11–22 ≡I

This completes our proof of the theorem '~ (A ∨ B) ≡ (~ A & ~ B)'.

We will conclude our discussion of theorems by constructing a proof of what has become known as Peirce's Law.[3]

$[(A \supset B) \supset A] \supset A$

Since the theorem is a conditional it is plausible that we will be using Conditional Introduction as our primary strategy.

Derive: $[(A \supset B) \supset A] \supset A$

1	$(A \supset B) \supset A$	A / $\supset$I
G	A	
G	$[(A \supset B) \supset A] \supset A$	1–__ $\supset$I

But how we should proceed next may not be obvious. We could derive our current goal, 'A', from line 1 by Conditional Elimination *if* we also had 'A $\supset$ B', but we do not. So perhaps we should take the sentence 'A $\supset$ B' as our new goal, and try to obtain it by Conditional Introduction.

Derive: $[(A \supset B) \supset A] \supset A$

1	$(A \supset B) \supset A$	A / $\supset$I
2	A	A / $\supset$I
G	B	
G	$A \supset B$	2–__ $\supset$I
G	A	1, __ $\supset$E
G	$[(A \supset B) \supset A] \supset A$	1–__ $\supset$I

So far, one might think, so good. But how are we to obtain 'B' from the sentences on lines 1 and 2? We could assume '~ B' and hope to use Negation Elimination.

Derive: $[(A \supset B) \supset A] \supset A$

1	$(A \supset B) \supset A$	A / $\supset$I
2	A	A / $\supset$I
3	~ B	A / ~ E
G	B	
G	$A \supset B$	2–__ $\supset$I
G	A	1, __ $\supset$E
G	$[(A \supset B) \supset A] \supset A$	1–__ $\supset$I

[3]The first proof of this theorem was given by Charles Peirce, a nineteenth-century American philosopher.

Unfortunately, the only negation now available is '~ B', so it appears that to make Negation Elimination work we will have to derive '~ B' (by Reiteration) and 'B'. But how do we derive 'B'? We seem to be back where we were before we assumed '~ B'. That is, 'B' is again our goal sentence.

We appear to be on the wrong track. Suppose that when we had 'A' as our goal, instead of planning on deriving 'A' by Conditional Elimination we try to derive it by Negation Elimination.

Derive: $[(A \supset B) \supset A] \supset A$

1	$(A \supset B) \supset A$	A / ⊃I
2	~A	A / ~ E
G	A	2–__ ~ E
G	$[(A \supset B) \supset A] \supset A$	1–__ ⊃I

Since we have a negation available, '~ A', perhaps we should take 'A' and '~ A' as the sentences **Q** and ~ **Q** we need to use Negation Elimination and accordingly make 'A' our new goal. This may seem no more promising than was the line of reasoning recently abandoned, since deriving 'A' was our goal before assuming '~ A'. But we are, in fact, making progress.

Derive: $[(A \supset B) \supset A] \supset A$

1	$(A \supset B) \supset A$	A / ⊃I
2	~ A	A / ~ E
G	A	
	~ A	2 R
G	A	2–__ ~ E
G	$[(A \supset B) \supset A] \supset A$	1–__ ⊃I

We can obtain 'A' from line 1 by Conditional Elimination if we can first obtain 'A ⊃ B'. This is, of course, the position we were in at the start of our work. But now we have an additional assumption available to us, namely '~ A'.

Derive: [(A ⊃ B) ⊃ A] ⊃ A

1	(A ⊃ B) ⊃ A	A / ⊃I
2	~ A	A / ~ E
3	A	A / ⊃I
G	B	
G	A ⊃ B	3–__ ⊃I
G	A	1–__ ⊃E
	~ A	2 R
G	A	2–__ ~ E
G	[(A ⊃ B) ⊃ A] ⊃ A	1–__ ⊃I

And now we can see our way to the end. We need 'B' and we have a sentence and its negation readily available ('A' and '~ A'), so we can assume '~ B' and use Negation Elimination. Here is the completed derivation.

Derive: [(A ⊃ B) ⊃ A] ⊃ A

1	(A ⊃ B) ⊃ A	A / ⊃I
2	~ A	A / ~ E
3	A	A / ⊃I
4	~ B	A / ~ E
5	A	3 R
6	~A	2 R
7	B	4–6 ~ E
8	A ⊃ B	3–7 ⊃I
9	A	1, 8 ⊃E
10	~A	2 R
11	A	2–10 ~ E
12	[(A ⊃ B) ⊃ A] ⊃ A	1–11 ⊃I

It is worth noting that in this example, as is frequently the case, a strategy that at first seems obvious (using Conditional Elimination to obtain 'A' as the penultimate line of the derivation) but proves problematic can successfully be used as a secondary strategy inside an alternative strategy (here Negation Elimination).

EQUIVALENCE

Suppose we want to establish that 'A ≡ ~ B' and '~ A ≡ B' are equivalent in *SD* (they are). Two derivations are required, one deriving '~ A ≡ B' from

{A ≡ ~ B} and one deriving 'A ≡ ~ B' from {~ A ≡ B}. Here is a start for the first of these:

Derive: ~ A ≡ B

1	A ≡ ~ B	Assumption
G	~ A ≡ B	

It should be apparent that our goal, '~ A ≡ B', is not going to be obtained by an elimination rule. We have too little to work with by way of primary assumptions for that to be a viable strategy. Since the main connective of our goal sentence is '≡', Biconditional Introduction may be a viable strategy. So we continue our derivation thus:

Derive: ~ A ≡ B

1	A ≡ ~ B	Assumption
2	~ A	A / ≡I
G	B	
	B	A / ≡I
G	~ A	
G	~ A ≡ B	2–__, __–__ ≡I

We now have two subderivations to complete. The goal of the first is 'B', and it can be obtained by Negation Elimination. The goal of the second, '~ A', can be obtained by Negation Introduction:

Derive: ~ A ≡ B

1	A ≡ ~ B	Assumption
2	~ A	A / ≡I
3	~ B	A / ~ E
4	A	1, 3 ≡E
5	~ A	2 R
6	B	3–5 ~ E
7	B	A / ≡I
8	A	A / ~ I
9	~ B	1, 8 ≡E
10	B	7 R
11	~ A	8–10 ~ I
12	~ A ≡ B	2–6, 7–11 ≡I

The second half of our current task is to derive 'A ≡ ~ B' from {~A ≡ B}.

Derive: A ≡ ~ B

1	~ A ≡ B	Assumption
G	A ≡ ~ B	

Biconditional Introduction is also a good strategy in this case.

Derive: A ≡ ~ B

1	~ A ≡ B	Assumption
2	A	A / ≡I
G	~ B	A / ≡I
	~ B	
G	A	
G	A ≡ ~ B	2–__, __–__ ≡I

Here, too, negation strategies will yield the desired results:

Derive: A ≡ ~ B

1	~ A ≡ B	Assumption
2	A	A / ≡I
3	B	A / ~ I
4	~ A	1, 3 ≡E
5	A	2 R
6	~ B	3–5 ~ I
7	~ B	A / ≡I
8	~ A	A / ~ E
9	B	1, 8 ≡E
10	~ B	7 R
11	A	8–10 ~ E
12	A ≡ ~ B	2–6, 7–11 ≡I

We next show that 'A ⊃ B' and '~ A ∨ B' are equivalent in *SD*. To do so will require deriving each sentence from the unit set of the other. So we will be doing two derivations. Both of these derivations are rather difficult but also highly instructive as they will allow us to illustrate strategies associated with a number of introduction and elimination rules. We set up our first derivation as follows:

Derive: ~ A ∨ B

1	A ⊃ B	Assumption
G	~ A ∨ B	

Our goal sentence is '~ A ∨ B', a disjunction. So we might be tempted to try to obtain our goal by Disjunction Introduction. While this strategy will not work, we will explore it anyway to illustrate how one can fall into unproductive strategies. If we are to use Disjunction Introduction we will need to first obtain either '~ A' or 'B'. We will take 'B' as our new goal. (In fact, neither 'B' nor '~ A' is obtainable given just 'A ⊃ B'.)

Derive: ~ A ∨ B

1	A ⊃ B	Assumption
G	B	
G	~ A ∨ B	__ ∨I

Since our goal is now 'B', and we have 'A ⊃ B' at line 1, it might seem like a good idea to assume 'A' and then use Conditional Elimination to obtain 'B'.

Derive: ~ A ∨ B

1	A ⊃ B	Assumption	
2	A	A	
3	B	1, 2 ⊃E	
4	B	3 R	**MISTAKE!**
5	~ A ∨ B	4 ∨ I	

Line 4 is a mistake because it appeals to a sentence, 'B', on line 3 that is not accessible at line 4. There is a scope line to the left of 'B' at line 3 that does not continue through line 4. We had two chances to avoid going down this path to a mistake. First, thinking we could get '~ A ∨ B' by first deriving 'B' from the assumption on line 1 was a bad idea. That assumption is 'A ⊃ B'. We are trying to show that 'A ⊃ B' and '~ A ∨ B' are equivalent in *SD,* as indeed they are. Although we are here concerned with syntactic properties of sentences and sets of sentences, it is well to remember that for any set Γ of sentences of *SL* and any sentence **P** of *SL*,

$\Gamma \vdash \mathbf{P}$ in *SD* if and only if $\Gamma \models \mathbf{P}$.

Our ill-advised strategy involved trying to show that

$\{A \supset B\} \vdash B$

where in fact 'B' is not derivable from {A ⊃ B}. For if this derivability claim did hold then it would also have to be the case that

$\{A \supset B\} \models B$

and this claim is false. There are truth-value assignments on which 'A ⊃ B' is true and 'B' false, namely every truth-value assignment on which 'A' and 'B' are both assigned **F**.

We had a second chance to avoid going down an unpromising road when we assumed 'A' at line 2. Note that there is nothing in the justification column for line 2 indicating why we are making this assumption. Had we been paying attention at that time we would have realized that we have no good reason for assuming 'A'. There is no subderivation strategy that will allow us to assume 'A', derive some sentence or sentences, and then end the subderivation and enter 'B' as the next line.

A more promising strategy for completing our first derivation, though one that does not initially come to mind when one is first learning to do derivations, is to use Negation Elimination to obtain '~ A ∨ B'.

Derive: ~ A ∨ B

1	A ⊃ B	A / ≡I
2	~ (~ A ∨ B)	A / ~ E
G	~ A ∨ B	2–__ ~ E

This strategy will seem unpromising if one thinks either that the **Q** and ~ **Q** that need to be derived to use a negation rule must be an atomic sentence and its negation, or that a negation must be among or easily obtained from the sentences that are accessible before one makes the auxiliary assumption that begins a negation subderivation. Neither is the case. The **Q** and ~ **Q** that both negation rules require deriving can be a compound sentence and its negation as well as an atomic sentence and its negation. And the ~ **Q** that is derived can occur as the auxiliary assumption that initiates the negation subderivation. Keeping this in mind we proceed as follows:

Derive: ~ A ∨ B

1	A ⊃ B	Assumption
2	~ (~ A ∨ B)	A / ~ E
G	~ A ∨ B	
	~ (~ A ∨ B)	2 R
G	~ A ∨ B	2–__ ~ E

It certainly might appear that we are making no progress. The goal of this derivation is '~ A ∨ B'. And this same sentence is now our goal within the subderivation begun at line 2. But in fact we are making progress. We noted earlier that we cannot derive '~ A ∨ B' by Disjunction Introduction when the only accessible sentence is 'A ⊃ B'. But we now have two accessible sentences to appeal to, those at lines 1 and 2. If we can use these two assumptions to derive '~ A', we can obtain our current goal, '~ A ∨ B' by Disjunction Introduction. This suggests we try to obtain '~ A' by Negation Introduction.

Derive: ~ A ∨ B

1	A ⊃ B	Assumption
2	~ (~ A ∨ B)	A / ~ E
3	A	A / ~ I
G	~ A	3–__ ~ I
G	~ A ∨ B	
	~ (~ A ∨ B)	2 R
G	~ A ∨ B	2–__ ~ E

We are again at the point where it is essential to be able to 'see' what we can obtain from the sentences that are accessible at the point where we are working (inside the subderivation that we began at line 3). The accessible sentences are those on lines 1–3. At line 3 we have 'A'. At line 1 we have 'A ⊃ B'. From these two sentences we can obtain 'B' by Conditional Elimination. From 'B' we can obtain '~ A ∨ B' by Disjunction Introduction, and we can derive the negation of this sentence, '~ (~ A ∨ B)' by Reiteration on line 2. These steps will complete the first half of our current task, that of showing that 'A ⊃ B' and '~ A ∨ B' are equivalent in *SD*.

Derive: ~ A ∨ B

1	A ⊃ B	Assumption
2	~ (~ A ∨ B)	A / ~ E
3	A	A / ~ I
4	B	1, 3 ⊃E
5	~ A ∨ B	4 ∨I
6	~ (~ A ∨ B)	2 R
7	~ A	3–6 ~ I
8	~ A ∨ B	7 ∨I
9	~ (~ A ∨ B)	2 R
10	~ A ∨ B	2–9 ~ E

This derivation of '~ A ∨ B' from 'A ⊃ B' is instructive in several ways. First, given that a disjunction is derivable, it does not follow that the last step in that derivation is Disjunction Introduction. Second, in picking a goal sentence it is wise to consider whether it is plausible that the selected sentence is derivable from the currently accessible sentences. Third, when using a negation rule the **Q** and ~ **Q** to be derived within the scope of the assumption called for by the rule may well both be compound sentences. Fourth, it does sometimes happen that one sentence is a goal in multiple parts of a derivation. Fifth, in using a negation

rule it is advisable to use as ~ **Q** a sentence that is readily available, and it may be available as the assumption of the very subderivation in which we are working. Finally, there is nothing wrong with using two or more instances of negation rules within which the same sentences (on different lines) play the roles of **Q** and ~ **Q**.

The second part of our proof that 'A ⊃ B' and '~ A ∨ B' are equivalent in *SD,* a derivation of 'A ⊃ B' from {~ A ∨ B}, is also instructive.

Derive: A ⊃ B

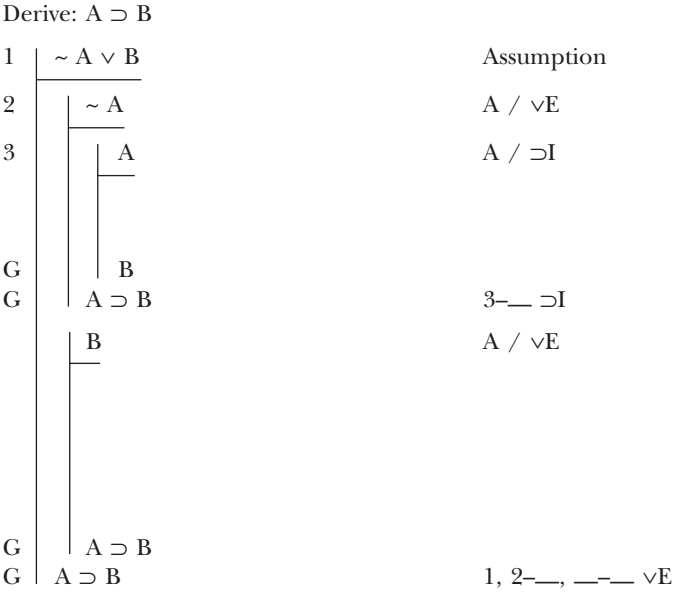
1	~ A ∨ B	Assumption
G	A ⊃ B	

We now need a strategy for getting from '~ A ∨ B' to 'A ⊃ B'. A little reflection suggests two alternative strategies. Since the goal sentence is a material conditional, we could use Conditional Introduction, and accordingly assume 'A' at line 2 for the purpose of using Conditional Introduction. Alternatively, since the only accessible sentence, the one at line 1, is a disjunction, we could plan to work to the conditional we want by using Disjunction Elimination. That is, in this case we can either let our goal sentence drive our strategy, working from the bottom up, or we can let our one accessible sentence drive our strategy, working from the top down. Here, as is often the case, both strategies will work. Moreover, whichever strategy we pick as our primary strategy we will end up using the other strategy within the first strategy. This is also often the case. Picking Disjunction Elimination as our primary strategy yields the following:

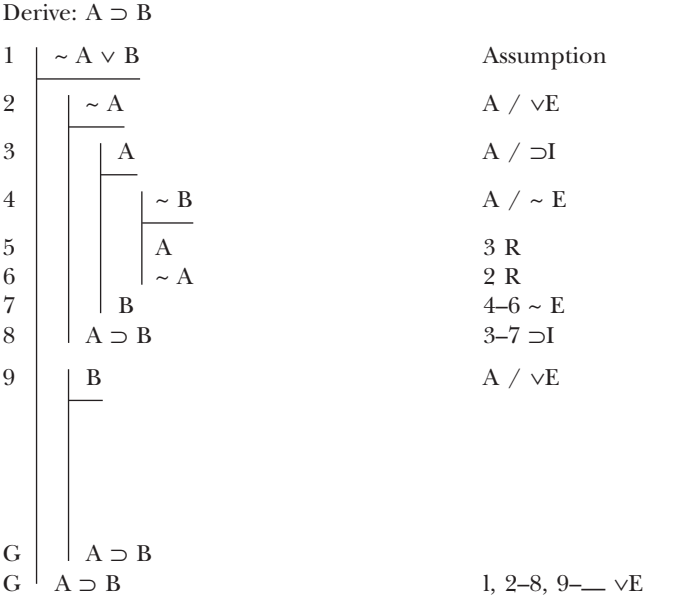
Derive: A ⊃ B

1	~ A ∨ B	Assumption
2	~ A	A / ∨E
G	A ⊃ B	
	B	A / ∨E
G	A ⊃ B	
G	A ⊃ B	1, 2–__, __–__ ∨E

Lines 1 and 2, by themselves, don't suggest a strategy for deriving 'A ⊃ B'. But 'A ⊃ B' is a material conditional and this suggests we use Conditional Introduction to obtain it.



Our goal within the subderivation beginning on line 3 is 'B'. We now note that the three accessible sentences include both 'A' and '~ A'. Their availability invites a negation strategy. To obtain 'B' we thus assume '~ B' and derive 'A' and '~ A', both by Reiteration.



What remains is to derive 'A ⊃ B' from 'B'. This is actually quite easy. We can use Conditional Introduction, assuming 'A' and deriving 'B' by Reiteration on line 9.

Derive: A ⊃ B

1	~ A ∨ B	Assumption
2	~ A	A / ∨E
3	A	A / ⊃I
4	~B	A / ~ E
5	A	3 R
6	~ A	2 R
7	B	4–6 ~ E
8	A ⊃ B	3–7 ⊃I
9	B	A / ∨E
10	A	A / ⊃I
11	B	9 R
12	A ⊃ B	10–11 ⊃I
13	A ⊃ B	1, 2–8, 9–12 ∨E

We have derived '~ A ∨ B' from {A ⊃ B} and 'A ⊃ B' from {~ A ∨ B}, thus demonstrating that these sentences are equivalent in *SD*. Two important lessons about material conditionals are illustrated in our last derivation. The first is that a conditional can be derived from the negation of its antecedent, as we did in lines 2 through 8 above. The second is that a material conditional can be derived from its consequent as we did in lines 9–12 above.

In our last derivation we used Disjunction Elimination as our primary strategy. Using Conditional Introduction as the primary strategy works just as well:

Derive: A ⊃ B

1	~ A ∨ B	Assumption
2	A	A / ⊃I
3	~ A	A / ∨E
4	~ B	A / ~ E
5	A	2 R
6	~ A	3 R
7	B	4–6 ~ E
8	B	A / ∨E
9	B	8 R
10	B	1, 3–7, 8–9 ∨E
11	A ⊃ B	2–10 ⊃I

INCONSISTENCY

We will conclude our illustration of strategies for constructing derivations in *SD* by doing several derivations that demonstrate the inconsistency of given

sets. Consider first the set {~ (A ⊃ B), B}. To show this set is inconsistent in *SD* we need to derive from it some sentence **Q** and its negation ~ **Q**. In planning a strategy it helps to remember that **Q** need not be an atomic sentence, and that it is often useful to use as ~ **Q** a sentence that is readily available. In the present case the only readily available negation is '~ (A ⊃ B)'. This suggests the following strategy:

Derive: A ⊃ B, ~ (A ⊃ B)

1	~ (A ⊃ B)	Assumption
2	B	Assumption
G	A ⊃ B	
G	~ (A ⊃ B)	1 R

Our goal is now to derive 'A ⊃ B' from our two assumptions. Since this goal sentence is a conditional, we will plan on using Conditional Introduction:

Derive: A ⊃ B, ~ (A ⊃ B)

1	~ (A ⊃ B)	Assumption
2	B	Assumption
3	A	A / ⊃I
G	B	
G	A ⊃ B	3–__ ⊃I
G	~ (A ⊃ B)	1 R

It is now apparent that our derivation is effectively done. Our only remaining goal, 'B', can be obtained by Reiteration on line 2:

Derive: A ⊃ B, ~ (A ⊃ B)

1	~ (A ⊃ B)	Assumption
2	B	Assumption
3	A	A / ⊃I
4	B	2 R
5	A ⊃ B	3–4 ⊃I
6	~ (A ⊃ B)	1 R

Establishing that the following set is inconsistent in *SD* is only modestly more challenging:

{A ≡ ~ B, B ≡ C, A ≡ C}

In this example the only negation that occurs as a component of any of the members of the set is '~ B'. So perhaps our goal should be to derive both 'B' and '~ B', even though neither can be derived by Reiteration or by any other rule in a single step.

Derive: B, ~ B

1	A ≡ ~ B	Assumption
2	B ≡ C	Assumption
3	A ≡ C	Assumption
G	B	
G	~ B	

To obtain our first goal, 'B', we might try using Negation Elimination:

Derive: B, ~ B

1	A ≡ ~ B	Assumption
2	B ≡ C	Assumption
3	A ≡ C	Assumption
4	~ B	A / ~ E
G	B	4–__ ~ E
G	~ B	4–__ ~ I

A cursory inspection of the sentences on lines 1–4 reveals that we can obtain '~ B' by Reiteration and 'B' by repeated uses of Biconditional Elimination:

Derive: B, ~ B

1	A ≡ ~ B	Assumption
2	B ≡ C	Assumption
3	A ≡ C	Assumption
4	| ~ B	A / ~ E
5	| A	1, 4 ≡E
6	| C	3, 5 ≡E
7	| B	2, 6 ≡E
8	| ~ B	4 R
9	B	4–8 ~ E
G	~ B	

The remaining task is to derive '~ B', and this too can be accomplished by repeated applications of Biconditional Elimination:

Derive: B, ~ B

1	A ≡ ~ B	Assumption
2	B ≡ C	Assumption
3	A ≡ C	Assumption
4	| ~ B	A / ~ I
5	| A	1, 4 ≡E
6	| C	3, 5 ≡E
7	| B	2, 6 ≡E
8	| ~ B	4 R
9	B	4–8 ~ I
10	C	2, 9 ≡E
11	A	3, 10 ≡E
12	~ B	1, 11 ≡E

Finally, we will show that the set {~ (A ⊃ B), ~ (B ⊃ C)} is inconsistent in *SD*. This is a challenging exercise. We do have two negations immediately available, so we will probably use one of them as ~ **Q**; which one makes no difference. So we set up our derivation this way:

Derive: A ⊃ B, ~ (A ⊃ B)

1	~ (A ⊃ B)	Assumption
2	~ (B ⊃ C)	Assumption
G	A ⊃ B	
	~ (A ⊃ B)	1 R

We cannot apply any elimination rule to either assumption since they are both negations. So we proceed by asking how our current goal, 'A ⊃ B', could be obtained by an introduction rule, and the answer is of course by Conditional Introduction:

Derive: A ⊃ B, ~ (A ⊃ B)

1	~ (A ⊃ B)	Assumption
2	~ (B ⊃ C)	Assumption
3	A	A / ⊃I
G	B	
G	A ⊃ B	3–__ ⊃I
	~ (A ⊃ B)	1 R

Our new goal is 'B'. The only strategy for obtaining 'B' that seems remotely promising is that of Negation Elimination:

Derive: A ⊃ B, ~ (A ⊃ B)

1	~ (A ⊃ B)	Assumption
2	~ (B ⊃ C)	Assumption
3	A	A / ⊃I
4	~ B	A / ~ E
G	B	4–__ ~ E
G	A ⊃ B	3–__ ⊃I
	~ (A ⊃ B)	1 R

We need to derive, within the subderivation beginning on line 4, a sentence **Q** and its negation ~ **Q**. Three negations, '~ (A ⊃ B)', '~ (B ⊃ C)', and '~B' are readily available. Since the presumed inconsistency of the set we are testing fairly clearly derives from the interplay of those two assumptions—that is, neither assumption by itself is problematic—we will eventually have to appeal to both assumptions. And we are already using '~ (A ⊃ B)' (as the last line of our derivation), so perhaps it is time to find a role for '~ (B ⊃ C)'. Accordingly we will try to obtain 'B ⊃ C' and '~ (B ⊃ C)'.

Derive: A ⊃ B, ~ (A ⊃ B)

1	~ (A ⊃ B)	Assumption
2	~ (B ⊃ C)	Assumption
3	A	A / ⊃I
4	~ B	A / ~ E
	B ⊃ C	
	~ (B ⊃ C)	2 R
	B	4–__ ~ E
	A ⊃ B	3–__ ⊃I
	~ (A ⊃ B)	1 R

Our new goal, 'B ⊃ C', is a conditional, so Conditional Introduction seems appropriate:

Derive: A ⊃ B, ~ (A ⊃ B)

1	~ (A ⊃ B)	Assumption
2	~ (B ⊃ C)	Assumption
3	A	A / ⊃I
4	~B	A / ~ E
5	B	A / ⊃I
	C	
	B ⊃ C	5–__ ⊃I
	~ (B ⊃ C)	2 R
	B	4–__ ~ E
	A ⊃ B	3–__ ⊃I
	~ (A ⊃ B)	1 R

At this point, as is often the case, the 'trick' is to be aware of what sentences are available to us—in this case the sentences on lines 1–5—and what we can do with those sentences. Note that we have both 'B' (at line 5) and '~ B' (at

line 4), and we know that whenever we can obtain a sentence and its negation we can obtain any sentence whatsoever by the appropriate negation strategy. We want 'C', so we will obtain it by Negation Elimination.

Derive: A ⊃ B, ~ (A ⊃ B)

1	~ (A ⊃ B)	Assumption
2	~ (B ⊃ C)	Assumption
3	A	A / ⊃I
4	~ B	A / ~ E
5	B	A / ⊃I
6	~ C	A / ~ E
7	B	5 R
8	~ B	4 R
9	C	6–8 ~ E
10	B ⊃ C	5–9 ⊃I
11	~ (B ⊃ C)	2 R
12	B	4–11 ~ E
13	A ⊃ B	3–12 ⊃I
14	~ (A ⊃ B)	1 R

5.3**E** EXERCISES

1. Construct derivations that establish the following derivability claims. In each case start by setting up the main structure of the derivation—with the primary assumption or assumptions at the top and the sentence to be derived at the bottom, and then identify the initial subgoal or goals. Complete the derivation, remembering to consider both the form of the current goal sentence and the content of the accessible sentences in selecting appropriate subgoals.

a. {A ⊃ B} ⊢ A ⊃ (A & B)
*b. {~ B ≡ A} ⊢ A ⊃ ~ B
c. {(K ⊃ L) & (L ⊃ K)} ⊢ L ≡ K
*d. {M ≡ T, ~ T} ⊢ ~ M
e. {B & ~ B} ⊢ C
*f. {D} ⊢ A ⊃ (B ⊃ D)
g. {A ⊃ C, (~ A ∨ C) ⊃ (D ⊃ B)} ⊢ D ⊃ B
*h. {~ A ⊃ ~ B, A ⊃ C, B ∨ D, D ⊃ E} ⊢ E ∨ C
i. {A ⊃ B, ~ (B & ~ C) ⊃ A} ⊢ B
*j. {~ A ⊃ B, C ⊃ ~ B, ~ (~ C & ~ A)} ⊢ A
k. {A ∨ (B & C), C ⊃ ~ A} ⊢ B ∨ ~ C
*l. {(A ⊃ B) ⊃ ~ B} ⊢ ~ B
m. {(A ∨ B) ⊃ C, (D ∨ E) ⊃ [(F ∨ G) ⊃ A]} ⊢ D ⊃ (F ⊃ C)
*n. {(F ∨ G) ⊃ (H & I)} ⊢ ~ F ∨ H
o. {A ⊃ ~ (B ∨ C), (C ∨ D) ⊃ A, ~ F ⊃ (D & ~ E)} ⊢ B ⊃ F
*p. {(A & B) ≡ (A ∨ B), C & (C ≡ ~ ~ A)} ⊢ B
q. {F ⊃ (G ∨ H), ~ (~ F ∨ H), ~ G} ⊢ H
*r. {~ (A ⊃ B) & (C & ~ D), (B ∨ ~ A) ∨ [(C & E) ⊃ D]} ⊢ ~ E

2. Show that each of the following arguments is valid in *SD*.

a. $A \supset \sim B$
$\sim B \supset C$
$\therefore A \supset C$

*b. $B \supset (A \,\&\, \sim B)$
$\therefore \sim B$

c. $A \equiv B$
$\sim A$
$\therefore \sim B$

*d. $A \supset (B \,\&\, C)$
$\sim C$
$\therefore \sim A$

e. D
$\therefore A \supset [B \supset (C \supset D)]$

*f. $A \equiv B$
$B \equiv C$
$\therefore A \equiv C$

g. $A \supset (B \supset C)$
$D \supset B$
$\therefore A \supset (D \supset C)$

*h. $\sim B \supset A$
$C \vee \sim B$
$\sim C$
$\therefore A$

i. $\sim A \vee B$
$B \supset C$
$\therefore A \supset C$

*j. $(E \supset T) \,\&\, (T \supset O)$
$O \supset E$
$\therefore (E \equiv O) \,\&\, (O \equiv E)$

k. $A \supset (C \supset B)$
$\sim C \supset \sim A$
A
$\therefore B$

*l. $\sim F$
$\sim G$
$\therefore \sim (F \vee G)$

m. $F \equiv G$
$F \vee G$
$\therefore F \,\&\, G$

3. Prove that each of the following is a theorem in *SD*.

a. $A \supset (A \vee B)$
*b. $A \supset (B \supset A)$
c. $A \supset [B \supset (A \,\&\, B)]$
*d. $(A \,\&\, B) \supset [(A \vee C) \,\&\, (B \vee C)]$
e. $(A \equiv B) \supset (A \supset B)$
*f. $(A \,\&\, \sim A) \supset (B \,\&\, \sim B)$
g. $(A \supset B) \supset [(C \supset A) \supset (C \supset B)]$
*h. $(A \,\&\, B) \vee (\sim A \vee \sim B)$
i. $[(A \supset B) \,\&\, \sim B] \supset \sim A$
*j. $(A \,\&\, A) \equiv A$
k. $A \supset [B \supset (A \supset B)]$
*l. $\sim A \supset [(B \,\&\, A) \supset C]$
m. $(A \supset B) \supset [\sim B \supset \sim (A \,\&\, D)]$
*n. $[\sim A \supset \sim (A \supset B)] \supset A$

4. Show that the members of each of the following pairs of sentences are equivalent in *SD*.

a.	A & ~ A	B & ~ B
*b.	A & A	A ∨ A
c.	(A ∨ B) ⊃ A	B ⊃ A
*d.	~ (A ⊃ B)	A & ~ B
e.	~ (A ≡ B)	(A & ~ B) ∨ (B & ~ A)
*f.	A ≡ ~ B	~ (A ≡ B)

5. Show that each of the following sets of sentences is inconsistent in *SD*.

a. {~ (A ⊃ A)}
*b. {A ⊃ (B & ~ B), A}
c. {A ≡ B, B ⊃ ~ A, A}
*d. {A ≡ ~ (A ≡ A), A}
e. {A ⊃ ~ A, ~ A ⊃ A}
*f. {A ⊃ (C ⊃ B), ~ C ⊃ B, A & ~ B}
g. {~ (A ∨ B), C ⊃ A, ~ C ⊃ B}
*h. {~ (B ≡ A), ~ B, ~ A}
i. {~ (F ∨ G) ≡ (A ⊃ A), H ⊃ F, ~ H ⊃ F}

6. Show that the following derivability claims hold in *SD*.

a. {A ⊃ B, ~ A ⊃ ~ B} ⊢ A ≡ B
*b. {F ≡ ~ (G ≡ ~ H), ~ (F ∨ G)} ⊢ H
c. {A ≡ (~B ∨ C), B ⊃ C} ⊢ A
*d. {G ∨ ~ H, ~ G ∨ ~ H} ⊢ ~ H
e. {B ∨ (C ∨ D), C ⊃ A, A ⊃ ~ C} ⊢ B ∨ D
*f. {(A ⊃ B) ⊃ C, (A ⊃ B) ∨ ~ C ⊢ ~ C ≡ ~ (A ⊃ B)
g. {A ⊃ (D & B), (~ D ≡ B) & (C ⊃ A)} ⊢ (A ∨ B) ⊃ ~ C
*h. {~ (A ≡ B)} ⊢ (A & ~ B) ∨ (B & ~ A)

7. Show that each of the following arguments is valid in *SD*.

a.
~ (C ∨ A)

~ (C ≡ ~ A)

*b.
C ∨ ~ D
C ⊃ E
D

E

c.
~ A & ~ B

A ≡ B

*d.
~ (F ∨ ~ G) ≡ ~ (H ∨ I)
F ∨ I

H ∨ I

e.
H ≡ ~ (I & ~ J)
~ I ≡ ~ H
J ⊃ ~ I

~ H

*f.
~ (F ⊃ G)
~ (G ⊃ H)

I

g.
(F ∨ G) ∨ (H ∨ ~ I)
F ⊃ H
I ⊃ ~ G

H ∨ ~ I

*h. ~ D

$C \supset (A \equiv B)$
$(D \vee B) \supset \sim A$
$(A \equiv B) \supset (D \mathbin{\&} E)$
$\sim B \supset D$

$C \supset (\sim A \mathbin{\&} B)$

i. $\sim (F \vee \sim G) \equiv \sim (H \vee I)$
$F \vee I$

$F \vee (I \mathbin{\&} \sim G)$

*j. $(A \vee \sim B) \supset (C \mathbin{\&} D)$
$A \equiv \sim D$
$\sim B \equiv \sim C$

$\sim (A \vee \sim B)$

k. $(\sim A \equiv \sim C) \equiv (B \equiv \sim D)$
$\sim A \supset \sim B$
$C \supset \sim D$

$(\sim A \equiv \sim C) \supset (\sim A \equiv D)$

*l. $F \supset (G \vee H)$
$\sim (\sim F \vee H)$
$\sim G$

H

m. $\sim (A \supset B) \mathbin{\&} (C \mathbin{\&} \sim D)$
$(B \vee \sim A) \vee [(C \mathbin{\&} E) \supset D]$

$\sim E$

8. Prove that each of the following is a theorem in *SD*.

a. $\sim (A \supset B) \supset \sim (A \equiv B)$
*b. $\sim (A \equiv B) \supset \sim (A \mathbin{\&} B)$
c. $(A \supset B) \vee (B \supset A)$
*d. $[A \supset (B \supset C)] \equiv [(A \supset B) \supset (A \supset C)]$
e. $[(A \vee B) \supset C] \equiv [(A \supset C) \mathbin{\&} (B \supset C)]$
*f. $[A \vee (B \vee C)] \supset [(D \supset A) \vee ((D \supset B) \vee (D \supset C))]$
g. $\sim (A \equiv B) \equiv (A \equiv \sim B)$

9. Show that the members of each of the following pairs of sentences are equivalent in *SD*.

a.	A	$\sim \sim A$	Double Negation
*b.	A	$A \mathbin{\&} A$	Idempotence
c.	A	$A \vee A$	Idempotence
*d.	$A \mathbin{\&} B$	$B \mathbin{\&} A$	Commutation
e.	$A \vee B$	$B \vee A$	Commutation
*f.	$A \mathbin{\&} (B \mathbin{\&} C)$	$(A \mathbin{\&} B) \mathbin{\&} C$	Association
g.	$A \vee (B \vee C)$	$(A \vee B) \vee C$	Association
*h.	$A \supset (B \supset C)$	$(A \mathbin{\&} B) \supset C$	Exportation
i.	$A \supset B$	$\sim B \supset \sim A$	Transposition
*j.	$A \equiv B$	$(A \supset B) \mathbin{\&} (B \supset A)$	Equivalence
k.	$A \equiv B$	$(A \mathbin{\&} B) \vee (\sim A \mathbin{\&} \sim B)$	Equivalence
*l.	$A \mathbin{\&} (B \vee C)$	$(A \mathbin{\&} B) \vee (A \mathbin{\&} C)$	Distribution
m.	$A \vee (B \mathbin{\&} C)$	$(A \vee B) \mathbin{\&} (A \vee C)$	Distribution
*n.	$\sim (A \mathbin{\&} B)$	$\sim A \vee \sim B$	De Morgan

10. Show that each of the following sets of sentences of *SL* is inconsistent in *SD*.

a. $\{(A \supset B) \mathbin{\&} (A \supset \sim B), (C \supset A) \mathbin{\&} (\sim C \supset A)\}$
*b. $\{B \equiv (A \mathbin{\&} \sim A), \sim B \supset (A \mathbin{\&} \sim A)\}$

c. $\{C \equiv \sim A, C \equiv A\}$
*d. $\{\sim (F \vee G) \equiv (\sim F \supset \sim F), \sim G \supset F\}$
e. $\{\sim [A \vee (B \vee C)], A \equiv \sim C\}$
*f. $\{F \vee (G \supset H), \sim H \,\&\, \sim (F \vee \sim G)\}$
g. $\{A \,\&\, (B \vee C), (\sim C \vee H) \,\&\, (H \supset \sim H), \sim B\}$
*h. $\{[(A \equiv B) \equiv (D \,\&\, \sim D)] \equiv B, A\}$

11. Symbolize the following arguments in *SL*. Then show that the symbolized arguments are valid in *SD*.

a. Spring has sprung, and the flowers are blooming. If the flowers are blooming, the bees are happy. If the bees are happy but aren't making honey, then spring hasn't sprung. So the bees are making honey.

*b. If Luscious Food Industries goes out of business, then food processing won't be improved. And if they go out of business, canned beans will be available if and only if Brockport Company stays in business. But Brockport Company is going out of business, and canned beans will be available. Hence Luscious Food Industries is staying in business unless food processing is improved.

c. If civil disobedience is moral, then not all resistance to the law is morally prohibited, although our legal code is correct if all resistance to the law is morally prohibited. But civil disobedience is moral if and only if either civil disobedience is moral or our legal code is correct. Our judges have acted well only if all resistance to the law is morally prohibited. So our judges haven't acted well.

*d. If oranges contain citric acid so do lemons, or if lemons don't contain citric acid neither do grapefruit. Thus, if oranges and grapefruit contain citric acid, so do lemons.

e. Neither rubber nor wood is a good conductor of electricity. But either rubber is a good conductor if and only if metal is, or if metal or glass is a good conductor then wood is a good conductor if and only if metal is. So metal isn't a good conductor of electricity.

*f. If the trains stop running then airline prices will increase, and buses will reduce their fares provided that trains don't stop running. If airline prices increase, then buses won't lose their customers. Hence buses will lose their customers only if they reduce their fares.

g. If the house is built and taxes increase, Jones will go bankrupt. If Smith becomes mayor, then the tax director will quit; and Smith will become mayor unless the tax director quits. But taxes won't increase if but only if the tax director doesn't quit and Smith becomes mayor. So if the house is built, Jones will go bankrupt.

*h. Jim is a Democrat only if Howard or Rhoda is. If Howard is a Democrat, so are Barbara and Allen. If Barbara is a Democrat, then Allen is a Democrat only if Freda is. But not both Freda and Jim are Democrats. Therefore Jim is a Democrat only if Rhoda is too.

i. If life is a carnival, then I'm a clown or a trapeze artist. But either life isn't a carnival or there are balloons, and either there aren't any balloons or I'm not a clown. So, if life is a carnival, then I'm a trapeze artist.

12. Symbolize the following passages in *SL* and show that the resulting sets of sentences are inconsistent in *SD*.

a. If motorcycling is dangerous sailboating is also dangerous, and if sailboating is dangerous parachuting is dangerous. Motorcycling is dangerous but parachuting is not.

*b. If the recipe doesn't call for flavoring or it doesn't call for eggs, it's not a recipe for tapioca. If the recipe calls for eggs, then it's a tapioca recipe and it doesn't call for flavoring. But this recipe calls for eggs.

c. Bach is popular only if Beethoven is ignored. If Bach is unpopular and Beethoven isn't ignored, then current musical tastes are hopeless. Current musical tastes aren't hopeless, and Beethoven isn't ignored.

*d. Historians are right just in case theologians are mistaken, if and only if Darwin's theory is correct. And if historians or philosophers are right, then Darwinian theory is correct and theologians are mistaken. Historians are right if and only if philosophers are wrong. But if Darwinian theory is correct, then historians are mistaken.

e. Either Martha was commissioned to write the ballet or, if the fund-raising sale was a failure, Tony was commissioned. Nancy will dance if and only if Tony wasn't commissioned. But the fund-raiser was a failure, Nancy will dance, and Martha wasn't commissioned.

13. Explain:

a. Why we would not want to include the following derivation rule in *SD*.

| **P** ∨ **Q**
|
| **P**

*b. Why Negation Introduction is a dispensable rule in *SD*. We take a rule to be dispensable in *SD* if and only if the last line of every derivation that makes use of the rule in question can also be derived from the given assumptions without using that rule.

c. Why Reiteration is a dispensable rule in *SD*.

*d. Why deriving a sentence and its negation within the scope of an auxilliary assumption *does not* show that the primary assumptions constitute an inconsistent set but *does* show that the set that consists of the primary assumptions and the assumptions of all open subderivations is inconsistent.

e. Why an argument of *SL* that has as one of its premises the negation of a theorem is valid in *SD*.

14. In Chapter 6 (see Sections 6.3 and 6.4) we prove that, for any sentence **P** and set Γ of sentences of *SL*,

$\Gamma \vdash \mathbf{P}$ in *SD* if and only if $\Gamma \models \mathbf{P}$.

Show that a-c below follow from this result.

a. An argument of *SL* is valid in *SD* if and only if the argument is truth-functionally valid.

*b. A sentence **P** of *SL* is a theorem in *SD* if and only if **P** is truth-functionally true.

c. Sentences **P** and **Q** of *SL* are equivalent in *SD* if and only if **P** and **Q** are truth-functionally equivalent.

5.4 THE DERIVATION SYSTEM *SD*+

In this section we introduce a new natural deduction system, *SD*+, which contains all the derivation rules of *SD* plus some additional rules. However, *SD*+ is not a stronger system than *SD* in the sense that more arguments of *SL* can be

shown to be valid or that more sentences of *SL* are theorems in *SD* than are in *SD*+. That is

$$\Gamma \vdash \mathbf{P} \text{ in } SD$$

if and only if

$$\Gamma \vdash \mathbf{P} \text{ in } SD+$$

However, historically a larger set of rules, such as those constituting *SD*+, have been used in many derivation systems. This larger set contains some rules absent from *SD* that do correspond to reasoning patterns commonly used in ordinary discourse, and often derivations in *SD*+ are shorter than corresponding derivations in *SD*.

RULES OF INFERENCE

Suppose that prior to line **n** of a derivation two accessible lines, **i** and **j**, contain $\mathbf{P} \supset \mathbf{Q}$ and $\sim \mathbf{Q}$, respectively. In *SD* we can derive $\sim \mathbf{P}$ as follows:

i	**P** ⊃ **Q**	
j	~ **Q**	
n	| **P**	A / ~ I
n + 1	| **Q**	i, **n** ⊃E
n + 2	| ~ **Q**	j R
n + 3	~ **P**	n − (**n** + 2) ~ I

To avoid going through this routine every time such a situation arises, we introduce the rule **Modus Tollens:**

Modus Tollens (MT)

	P ⊃ **Q**
	~ **P**
▷	~ **P**

Now suppose that prior to line **n** of a derivation two accessible lines, **i** and **j**, contain $\mathbf{P} \supset \mathbf{Q}$ and $\mathbf{Q} \supset \mathbf{R}$. A routine to derive $\mathbf{P} \supset \mathbf{R}$ in *SD* beginning at line **i** is as follows:

i	**P** ⊃ **Q**	
j	**Q** ⊃ **R**	
n	| **P**	**A** / ~ I
n + 1	| **Q**	**i, n** ⊃**E**
n + 2	| **R**	**j, n** + 1 ⊃**E**
n + 3	**P** ⊃ **R**	**n** − (**n** + 2) ⊃**I**

To avoid this routine, we introduce the rule **Hypothetical Syllogism:**

Hypothetical Syllogism (HS)

	P ⊃ **Q**
	Q ⊃ **R**
▷	**P** ⊃ **R**

Finally suppose that prior to the line **n** of a derivation two accessible lines, **i** and **j**, contain **P** ∨ **Q** and ~ **P** and that we wish to derive **Q**. A routine for accomplishing this in *SD* is as follows:

i	**P** ∨ **Q**	
j	~ **P**	
n	**P**	A / ∨E
n + 1	~ **Q**	A / ~ E
n + 2	**P**	**n** R
n + 3	~ **P**	**j** R
n + 4	**Q**	**n** + 1 − **n** + 3 ~ E
n + 5	**Q**	A / ∨E
n + 6	**Q**	**n** + 5 R
n + 7	**Q**	**i**, **n** − **n** + 4, **n** + 5 − **n** + 6 ∨E

The rule of *Disjunctive Syllogism* allows us to avoid going through this routine for this and similar cases.

Disjunctive Syllogism (DS)

P ∨ **Q**		**P** ∨ **Q**
~ **P**	or	~ **Q**
Q		**P**

The three rules of inference just introduced can be thought of as derived rules. They are added for convenience only; whatever we can derive with them, we can derive without them, using only the rules of *SD*.

RULES OF REPLACEMENT

In addition to rules of inference, there are also derivation rules known as *rules of replacement*. Rules of replacement, as their name suggests, allow us to derive

some sentences from other sentences by replacing sentential components. For example, from the sentence

$$G \vee (H \,\&\, K)$$

we can certainly infer

$$G \vee (\sim \sim H \,\&\, K)$$

In this instance the sentential component 'H' has been replaced with '~ ~ H'. Similarly from

$$G \vee (\sim \sim H \,\&\, K)$$

we can certainly infer

$$G \vee (H \,\&\, K)$$

Double Negation is the rule of replacement that licenses such moves within a derivation.

Double Negation (DN)
$\mathbf{P} \lhd \rhd \sim \sim \mathbf{P}$

That is, by using Double Negation, we can derive from a sentence **Q** that contains **P** as a sentential component another sentence that is like **Q**, except that one occurrence of the sentential component **P** has been replaced with ~ ~ **P**. And, by using Double Negation, we can derive from a sentence **Q** that contains ~ ~ **P** as a sentential component another sentence that is like **Q**, except that one occurrence of the sentential component ~ ~ **P** has been replaced with **P**.

Double Negation can be applied to any of the sentential components of a sentence. For instance, from

$$G \vee (H \,\&\, K)$$

Double Negation permits us to derive

$$G \vee \sim \sim (H \,\&\, K)$$

And from

$$G \vee \sim \sim (H \,\&\, K)$$

Double Negation allows us to derive

$$G \vee (H \,\&\, K)$$

Since every sentence is a sentential component of itself, Double Negation applies to the entire sentence as well. In a derivation Double Negation permits us to go from

$$G \vee (H \,\&\, K)$$

to

$$\sim \sim [G \vee (H \,\&\, K)]$$

and from

$$\sim \sim [G \vee (H \,\&\, K)]$$

to

$$G \vee (H \,\&\, K)$$

Here are the rules of replacement for *SD*+:

Commutation (Com)
$\mathbf{P} \,\&\, \mathbf{Q} \dashv\vdash \mathbf{Q} \,\&\, \mathbf{P}$
$\mathbf{P} \vee \mathbf{Q} \dashv\vdash \mathbf{Q} \vee \mathbf{P}$

Association (Assoc)
$\mathbf{P} \,\&\, (\mathbf{Q} \,\&\, \mathbf{R}) \dashv\vdash (\mathbf{P} \,\&\, \mathbf{Q}) \,\&\, \mathbf{R}$
$\mathbf{P} \vee (\mathbf{Q} \vee \mathbf{R}) \dashv\vdash (\mathbf{P} \vee \mathbf{Q}) \vee \mathbf{R}$

Implication (Impl)
$\mathbf{P} \supset \mathbf{Q} \dashv\vdash \sim \mathbf{P} \vee \mathbf{Q}$

Double Negation (DN)
$\mathbf{P} \dashv\vdash \sim \sim \mathbf{P}$

De Morgan (DeM)
$\sim (\mathbf{P} \,\&\, \mathbf{Q}) \dashv\vdash \sim \mathbf{P} \vee \sim \mathbf{Q}$
$\sim (\mathbf{P} \vee \mathbf{Q}) \dashv\vdash \sim \mathbf{P} \,\&\, \sim \mathbf{Q}$

Idempotence (Idem)
$\mathbf{P} \dashv\vdash \mathbf{P} \,\&\, \mathbf{P}$
$\mathbf{P} \dashv\vdash \mathbf{P} \vee \mathbf{P}$

Transposition (Trans)
$\mathbf{P} \supset \mathbf{Q} \dashv\vdash \sim \mathbf{Q} \supset \sim \mathbf{P}$

Exportation (Exp)
$\mathbf{P} \supset (\mathbf{Q} \supset \mathbf{R}) \dashv\vdash (\mathbf{P} \,\&\, \mathbf{Q}) \supset \mathbf{R}$

Distribution (Dist)
$\mathbf{P} \,\&\, (\mathbf{Q} \vee \mathbf{R}) \dashv\vdash (\mathbf{P} \,\&\, \mathbf{Q}) \vee (\mathbf{P} \,\&\, \mathbf{R})$
$\mathbf{P} \vee (\mathbf{Q} \,\&\, \mathbf{R}) \dashv\vdash (\mathbf{P} \vee \mathbf{Q}) \,\&\, (\mathbf{P} \vee \mathbf{R})$

Equivalence (Equiv)
$\mathbf{P} \equiv \mathbf{Q} \dashv\vdash (\mathbf{P} \supset \mathbf{Q}) \,\&\, (\mathbf{Q} \supset \mathbf{P})$
$\mathbf{P} \equiv \mathbf{Q} \dashv\vdash (\mathbf{P} \,\&\, \mathbf{Q}) \vee (\sim \mathbf{P} \,\&\, \sim \mathbf{Q})$

Rules of replacement always allow the replacement of sentential components. In addition, all these rules of replacement are two-way rules; that is, a sentential component that has the form of the sentence on the left of '$\triangleleft\ \triangleright$' can be replaced with a sentential component that has the form of the sentence on the right of '$\triangleleft\ \triangleright$', and vice versa.

Consider the following derivation:

Derive: $J \supset [M \vee (G \vee I)]$

1	$J \supset [K \vee (L \vee H)]$	Assumption
2	$[(K \vee L) \vee H] \supset [(M \vee G) \vee I]$	Assumption
3	$J \supset [(K \vee L) \vee H]$	1 Assoc
4	$J \supset [(M \vee G) \vee I]$	2, 3 HS
5	$J \supset [M \vee (G \vee I)]$	4 Assoc

Here the replacement rule Association has been used twice—first to replace a sentential component of the form $\mathbf{P \vee (Q \vee R)}$ with a sentential component of the form $\mathbf{(P \vee Q) \vee R}$ and then to replace a sentential component of the form $\mathbf{(P \vee Q) \vee R}$ with a sentential component of the form $\mathbf{P \vee (Q \vee R)}$.

Since all the derivation rules of *SD* are derivation rules of *SD*+, the procedures for properly applying the rules of *SD* apply to *SD*+ as well. The rules of inference of *SD*+, including Modus Tollens, Hypothetical Syllogism, and Disjunctive Syllogism, must be applied to entire sentences on a line. Rules of replacement, on the other hand, can be applied to all sentential components. The following derivation illustrates the proper use of several of the rules of replacement:

Derive: $\sim C \equiv E$

1	$(D \vee B) \vee (E \supset \sim C)$	Assumption
2	$\sim B \ \&\ [\sim D \ \&\ (\sim E \supset C)]$	Assumption
3	$(\sim B \ \&\ \sim D) \ \&\ (\sim E \supset C)$	2 Assoc
4	$\sim (B \vee D) \ \&\ (\sim E \supset C)$	3 DeM
5	$\sim (B \vee D)$	4 &E
6	$\sim (D \vee B)$	5 Com
7	$E \supset \sim C$	1, 6 DS
8	$\sim E \supset C$	3 &E
9	$\sim C \supset \sim \sim E$	8 Trans
10	$\sim C \supset E$	9 DN
11	$(\sim C \supset E) \ \&\ (E \supset \sim C)$	7, 10 &I
12	$\sim C \equiv E$	11 Equiv

Notice that each application of a derivation rule requires a separate line. Moreover, care must be taken to apply each derivation rule only to sentences that have the proper form (or, in the case of rules of replacement, sentences that have components that have the proper form).

Here is an example in which these points are ignored:

Derive: ~ A ⊃ [B ⊃ (G ∨ D)]

1	(A ∨ ~ B) ∨ ~ C	Assumption	
2	(D ∨ G) ∨ C	Assumption	
3	~ (~ A & B) ∨ ~ C	1 DeM	**MISTAKE!**
4	(~ A & B) ⊃ ~ C	3 Impl	
5	C ∨ (G ∨ D)	2 Com	**MISTAKE!**
6	~ C ⊃ (G ∨ D)	5 Impl	**MISTAKE!**
7	(~ A & B) ⊃ (G ∨ D)	4, 6 HS	
8	~ A ⊃ [B ⊃ (G ∨ D)]	7 Exp	

De Morgan does not license entering the sentence on line 3. What De Morgan does allow is the replacement of a sentential component of the form ~ **P** ∨ ~ **Q** with a sentential component of the form ~ **(P** & **Q)**, but the sentential component 'A ∨ ~ B' does not have the form ~ **P** ∨ ~ **Q**. However, by applying Double Negation to the first assumption, we can obtain '(~ ~ A ∨ ~ B) ∨ ~ C'. And this latter sentence does have a sentential component of the form ~ **P** ∨ ~ **Q**, namely, '~ ~ A ∨ ~ B'. Here **P** is '~ A', and **Q** is 'B'. Hence the derivation should begin this way:

Derive: ~ A ⊃ [B ⊃ (G ∨ D)]

1	(A ∨ ~ B) ∨ ~ C	Assumption
2	(D ∨ G) ∨ C	Assumption
3	(~ ~ A ∨ ~ B) ∨ ~ C	1 DN
4	~ (~ A & B) ∨ ~ C	3 DeM

The second mistake in our example, in line 5, is that Commutation is applied twice within the same line. Each application of a rule, even if it is the same rule, requires a separate line. Correctly done, the derivation proceeds:

5	(~ A & B) ⊃ ~ C	4 Impl
6	C ∨ (D ∨ G)	2 Com
7	C ∨ (G ∨ D)	6 Com

The third mistake, in line 6 of the example, also stems from our trying to apply a rule of replacement to a sentential component that does not have the form required by the rule. Implication permits the replacement of a sentential component of the form ~ **P** ∨ **Q** with a sentential component of the form **P** ⊃ **Q**, but 'C ∨ (G ∨ D)' does not have the form ~ **P** ∨ **Q**. However, applying Double Negation to 'C', a sentential component of 'C ∨ (G ∨ D)', generates '~ ~ C ∨ (G ∨ D)'. This latter sentence does have the form ~ **P** ∨ **Q**, where P is '~ C' and **Q** is 'G ∨ D'. Here is the entire derivation done correctly:

Derive: ~ A ⊃ [B ⊃ (G ∨ D)]

1	(A ∨ ~ B) ∨ ~ C	Assumption
2	(D ∨ G) ∨ C	Assumption
3	(~ ~ A ∨ ~ B) ∨ ~ C	1 DN
4	~ (~ A & B) ∨ ~ C	3 DeM
5	(~ A & B) ⊃ ~ C	4 Impl
6	C ∨ (D ∨ G)	2 Com
7	C ∨ (G ∨ D)	6 Com
8	~ ~ C ∨ (G ∨ D)	7 DN
9	~ C ⊃ (G ∨ D)	8 Impl
10	(~ A & B) ⊃ (G ∨ D)	5, 9 HS
11	~ A ⊃ [B ⊃ (G ∨ D)]	10 Exp

The definitions of the basic concepts of *SD*+ parallel the definitions for the basic concepts of *SD,* except that '*SD*' is replaced with '*SD*+'. For example, the concept of derivability is defined as follows:

A sentence **P** of *SL* is *derivable in SD*+ from a set Γ of sentence of *SL* if and only if there is a derivation in *SD*+ in which all the primary assumptions are members of Γ and **P** occurs within the scope of only those assumptions.

Consequently tests for the various syntactic properties in *SD*+ are analogous to those of *SD*. To show that an argument is valid in *SD*+, we construct a derivation in *SD*+ showing that the conclusion of the argument is derivable in *SD*+ from the set all of whose members are premises of the argument. To show that a sentence **P** of *SL* is a theorem in *SD*+, we show that **P** is derivable in *SD*+ from the empty set. And so on. Remember that, although *SD* and *SD*+ are different syntactic systems, whatever can be derived in one can be derived in the other.

The Derivation Rules of SD+

All the Derivation Rules of *SD* and Rules of Inference

Modus Tollens (MT)

P ⊃ **Q**
~ **Q**
∴ ~ **P**

Hypothetical Syllogism (HS)

P ⊃ **Q**
Q ⊃ **R**
∴ **P** ⊃ **R**

Disjunctive Syllogism (DS)

P ∨ **Q**
~ **P**
∴ **Q**

or

P ∨ **Q**
~ **Q**
∴ **P**

Rules of Replacement

Commutation (Com)

P & **Q** ⊲⊳ **Q** & **P**
P ∨ **Q** ⊲⊳ **Q** ∨ **P**

Association (Assoc)

P & (**Q** & **R**) ⊲⊳ (**P** & **Q**) & **R**
P ∨ (**Q** ∨ **R**) ⊲⊳ (**P** ∨ **Q**) ∨ **R**

Implication (Impl)

P ⊃ **Q** ⊲⊳ ~ **P** ∨ **Q**

Double Negation (DN)

P ⊲⊳ ~ ~ **P**

De Morgan (DeM)

~ (**P** & **Q**) ⊲⊳ ~ **P** ∨ ~ **Q**
~ (**P** ∨ **Q**) ⊲⊳ ~ **P** & ~ **Q**

Idempotence (Idem)

P ⊲⊳ **P** & **P**
P ⊲⊳ **P** ∨ **P**

Transposition (Trans)

P ⊃ **Q** ⊲⊳ ~ **Q** ⊃ ~ **P**

Exportation (Exp)

P ⊃ (**Q** ⊃ **R**) ⊲⊳ (**P** & **Q**) ⊃ **R**

Distribution (Dist)

P & (**Q** ∨ **R**) ⊲⊳ (**P** & **Q**) ∨ (**P** & **R**)
P ∨ (**Q** & **R**) ⊲⊳ (**P** ∨ **Q**) & (**P** ∨ **R**)

Equivalence (Equiv)

P ≡ **Q** ⊲⊳ (**P** ⊃ **Q**) & (**Q** ⊃ **P**)
P ≡ **Q** ⊲⊳ (**P** & **Q**) ∨ (~ **P** & ~ **Q**)

5.4**E** EXERCISES

1. Show that the following derivability claims hold in *SD*+.

a. {D ⊃ E, E ⊃ (Z & W), ~ Z ∨ ~ W} ⊢ ~ D
*b. {(H & G) ⊃ (L ∨ K), G & H} ⊢ K ∨ L
c. {(W ⊃ S) & ~ M, (~ W ⊃ H) ∨ M, (~ S ⊃ H) ⊃ K} ⊢ K
*d. {[(K & J) ∨ I] ∨ ~ Y, Y & [(I ∨ K) ⊃ F]} ⊢ F ∨ N
e. {(M ∨ B) ∨ (C ∨ G), ~ B & (~ G & ~ M)} ⊢ C
*f. {~ L ∨ (~ Z ∨ ~ U), (U & G) ∨ H, Z} ⊢ L ⊃ H

2. Show that each of the following is valid in *SD*+.

a.
~ Y ⊃ ~ Z
~ Z ⊃ ~ X
~ X ⊃ ~ Y

Y ≡ Z

*b.
(~ A & ~ B) ∨ (~ A & ~ C)
(E & D) ⊃ A

~ E ∨ ~ D

c.
(F & G) ∨ (H & ~ I)
I ⊃ ~ (F & D)

I ⊃ ~ D

*d.
F ⊃ (~ G ∨ H)
F ⊃ G
~ (H ∨ I)

F ⊃ J

e. $F \supset (G \supset H)$
$\sim I \supset (F \vee H)$
$F \supset G$

$I \vee H$

g. $[(X \,\&\, Z) \,\&\, Y] \vee (\sim X \supset \sim Y)$
$X \supset Z$
$Z \supset Y$

$X \equiv Y$

*f. $G \supset (H \,\&\, \sim K)$
$H \equiv (L \,\&\, I)$
$\sim I \vee K$

$\sim G$

3. Show that each of the following is a theorem in *SD*+.

a. $A \vee \sim A$
*b. $\sim \sim \sim \sim \sim (A \,\&\, \sim A)$
c. $A \vee [(\sim A \vee B) \,\&\, (\sim A \vee C)]$
*d. $[(A \,\&\, B) \supset (B \,\&\, A)] \,\&\, [\sim (A \,\&\, B) \supset \sim (B \,\&\, A)]$
e. $[A \supset (B \,\&\, C)] \equiv [(\sim B \vee \sim C) \supset \sim A]$
*f. $[A \vee (B \vee C)] \equiv [C \vee (B \vee A)]$
g. $[A \supset (B \equiv C)] \equiv (A \supset [(\sim B \vee C) \,\&\, (\sim C \vee B)])$
*h. $(A \vee [B \supset (A \supset B)]) \equiv (A \vee [(\sim A \vee \sim B) \vee B])$
i. $[\sim A \supset (\sim B \supset C)] \supset [(A \vee B) \vee (\sim \sim B \vee C)]$
*j. $(\sim A \equiv \sim A) \equiv [\sim (\sim A \supset A) \equiv (A \supset \sim A)]$

4. Show that the members of each of the following pairs of sentences are equivalent in *SD*+.

a. $A \vee B$
$\sim (\sim A \,\&\, \sim B)$
*b. $A \,\&\, (B \vee C)$
$(B \,\&\, A) \vee (C \,\&\, A)$
c. $(A \,\&\, B) \supset C$
$\sim (A \supset C) \supset \sim B$
*d. $(A \vee B) \vee C$
$\sim A \supset (\sim B \supset C)$
e. $A \vee (B \equiv C)$
$A \vee (\sim B \equiv \sim C)$
*f. $(A \,\&\, B) \vee [(C \,\&\, D) \vee A]$
$([(C \vee A) \,\&\, (C \vee B)] \,\&\, [(D \vee A) \,\&\, (D \vee B)]) \vee A$

5. Show that the following sets of sentences are inconsistent in *SD*+.

a. $\{[(E \,\&\, F) \vee \sim \sim G] \supset M, \sim [[(G \vee E) \,\&\, (F \vee G)] \supset (M \,\&\, M)]\}$
*b. $\{\sim [(\sim C \vee \sim \sim C) \vee \sim \sim C]\}$
c. $\{M \,\&\, L, [L \,\&\, (M \,\&\, \sim S)] \supset K, \sim K \vee \sim S, \sim (K \equiv \sim S)\}$
*d. $\{B \,\&\, (H \vee Z), \sim Z \supset K, (B \equiv Z) \supset \sim Z, \sim K\}$
e. $\{\sim [W \,\&\, (Z \vee Y)], (Z \supset Y) \supset Z, (Y \supset Z) \supset W\}$
*f. $\{[(F \supset G) \vee (\sim F \supset G)] \supset H, (A \,\&\, H) \supset \sim A, A \vee \sim H\}$

6. Symbolize the following arguments in *SL*, and show that they are valid in *SD*+.

a. If the phone rings Ed is calling, or if the beeper beeps Ed is calling. If not both Ed and Agnes are at home today, then it's not the case that if the phone rings, Ed is calling. Ed isn't home today, and he isn't calling. So the beeper won't beep.

*b. If Monday is a bad day, then I'll lose my job provided the boss doesn't call in sick. The boss won't call in sick. So I'll lose my job—since either Monday will be a bad day, or the boss won't call in sick only if I lose my job.

c. Army coats are warm only if they're either made of wool or not made of cotton or rayon. If army coats are not made of rayon, then they're made of cotton. Hence, if they're not made of wool, army coats aren't warm.

*d. If either the greenhouse is dry or the greenhouse is sunny if and only if it's not raining, the violets will wither. But if the violets wither the greenhouse is sunny, or if the violets wither the greenhouse isn't dry. It's raining, and the greenhouse isn't sunny. So the greenhouse is dry only if the violets won't wither.

e. It's not the case that John is rich and Hugo isn't. In fact, Hugo isn't rich, unless Moe is. And if Moe just emptied his bank account, then he isn't rich. Thus, if John is rich, then it's not the case that either Moe emptied his bank account or Moe isn't rich,

*f. Neither aspirin nor gin will ease my headache, unless it's psychosomatic. If it's psychosomatic and I'm really not ill, then I'll go out to a party and drink some martinis. So, if I'm not ill and don't drink any martinis, then aspirin won't ease my headache.

g. If I stay on this highway and don't slow down, I'll arrive in Montreal by 5:00. If I don't put my foot on the brake, I won't slow down. Either I won't slow down or I'll stop for a cup of coffee at the next exit. I'll stop for a cup of coffee at the next exit only if I'm falling asleep. So, if I don't arrive in Montreal by 5:00, then I'll stay on this highway only if I'm falling asleep and I put my foot on the brake.

*h. The weather is fine if and only if it's not snowing, and it's not snowing if and only if the sky is clear. So, either the weather is fine, the sky is clear, and it's not snowing; or it's snowing, the sky isn't clear, and the weather is lousy.

7. Symbolize the following passages in *SL,* and show that the resulting sets of sentences of *SL* are inconsistent in *SD*+.

a. Unless Stowe believes that all liberals are atheists, his claims about current politics are unintelligible. But if liberals are atheists only if they're not churchgoers, then Stowe's claims about current politics are nevertheless intelligible. Liberals are, in fact, churchgoers if and only if Stowe doesn't believe that they're all atheists, and if liberals aren't atheists, then Stowe doesn't believe that they are atheists. Liberals aren't atheists.

*b. Either Congress won't cut taxes or the elderly and the poor will riot, if but only if big business prospers. If the elderly don't riot, then Congress won't cut taxes. It won't happen that both the poor will riot and big business will prosper, and it won't happen that the poor don't riot and big business doesn't prosper. But if big business prospers, then Congress will cut taxes.

8. Answer the following.

a. Suppose we can derive **Q** from **P** by using only the rules of replacement. Why can we be sure that we can derive **P** from **Q**?

*b. Why must all arguments that are valid in *SD* be valid in *SD*+ as well?

c. Suppose we develop a new natural deduction system *SD**. Let *SD** contain all the derivation rules of *SD* and in addition the derivation rule Absorption.

Absorption

$$\triangleright \;\begin{array}{|l} \mathbf{P} \supset \mathbf{Q} \\ \mathbf{P} \supset (\mathbf{P} \;\&\; \mathbf{Q}) \end{array}$$

Using only the derivation rules of *SD,* develop a routine showing that any sentence derived by using Absorption could be derived in *SD* without using it.

GLOSSARY[4]

DERIVABILITY IN *SD:* A sentence **P** of *SL* is *derivable in SD* from a set Γ of sentences of *SL* if and only if there is a derivation in *SD* in which all the primary assumptions are members of Γ and **P** occurs in the scope of only those assumptions.

VALIDITY IN *SD:* An argument of *SL* is *valid in SD* if and only if the conclusion of the argument is derivable in *SD* from the set consisting of the premises. An argument of *SL* is *invalid in SD* if and only if it is not valid in *SD*.

THEOREM IN *SD:* A sentence **P** of *SL* is a *theorem in SD* if and only if **P** is derivable in *SD* from the empty set.

EQUIVALENCE IN *SD:* Sentences **P** and **Q** of *SL* are *equivalent in SD* if and only if **Q** is derivable in *SD* from {**P**} and **P** is derivable in *SD* from {**Q**} .

INCONSISTENCY IN *SD:* A set Γ of sentences of *SL* is *inconsistent in SD* if and only if both a sentence **P** of *SL* and its negation ~ **P** are derivable in *SD* from Γ. A set Γ of sentences of *SL* is *consistent in SD* if and only if it is not inconsistent in *SD*.

[4]Similar definitions hold for the derivation system *SD*+.

Chapter 6

SENTENTIAL LOGIC: METATHEORY

Section 6.1 introduces mathematical induction, a technique that we will use to establish important results about the syntax and semantics of sentential logic. Section 6.2 establishes that the five connectives of *SL* are truth-functionally complete, that is, they can be used to express any truth-function. Section 6.3 establishes that *SD* and *SD*+ are sound systems for sentential logic, that is, derivability in these systems establishes truth-functional entailment. Section 6.4 establishes that *SD* and *SD*+ are complete for sentential logic, that is, given any truth-functional entailment, there is a corresponding derivation in these systems.

6.1 MATHEMATICAL INDUCTION

In the three previous chapters we concentrated on developing and using techniques of sentential logic, both semantic and syntactic. In this chapter we step back to prove some claims *about* the semantics and syntax of sentential logic. Such results constitute the **metatheory** of sentential logic. Throughout this chapter, unless otherwise noted, when we speak of sets we are speaking of sets of sentences of *SL*, and when we speak of sentences we are speaking of sentences of *SL*. We also adopt the convention of numbering our metatheoretic results to reflect the section in which they occur.

For the language *SL*, the semantic accounts of such logical properties of sentences and sets of sentences of *SL* as validity, consistency, and equivalence

given in Chapter 3 are fundamental in the sense that they are the standards by which other accounts of these properties are judged. For instance, although the techniques of Chapter 5 are purely syntactical—all the derivation rules appeal to the structures or forms of sentences, not to their truth-conditions—those techniques are intended to yield results paralleling the results yielded by the semantic techniques of Chapter 3. One of the important metatheoretic results that we shall prove in this chapter is that this parallel does hold. We shall prove this by proving that the natural deduction system *SD* allows us to construct all and only the derivations we want to be able to construct, given the semantics of Chapter 3. Specifically we shall prove that, given any set Γ of sentences of *SL* and any sentence **P** of *SL*, **P** is derivable from Γ in *SD* if and only if **P** is truth–functionally entailed by Γ. It follows from this that all and only the truth-functionally valid arguments of *SL* are valid in *SD*, all and only the truth-functionally true sentences of *SL* are theorems in *SD*, and so on.

We shall use a very powerful method of proof known as **mathematical induction** to establish the foregoing results. We introduce mathematical induction with a simple example. We will use it to prove what appears to be an obvious result: that in every sentence of *SL* the number of left parentheses equals the number of right parentheses. Because there are an infinite number of sentences of *SL*, we cannot establish this result by looking at each sentence and counting the number of left and right parentheses that it contains. Mathematical induction allows us to establish that a claim holds for an infinite number of cases without going through them one at a time. Recall the recursive definition of 'sentence of *SL*' given in Chapter 2:

1. Every sentence letter of *SL* is a sentence of *SL*.
2. If **P** is a sentence of *SL*, then $\sim$ **P** is a sentence of *SL*.
3. If **P** and **Q** are sentences of *SL*, then (**P** & **Q**) is a sentence of *SL*.
4. If **P** and **Q** are sentences of *SL*, then (**P** $\vee$ **Q**) is a sentence of *SL*.
5. If **P** and **Q** are sentences of *SL*, then (**P** $\supset$ **Q**) is a sentence of *SL*.
6. If **P** and **Q** are sentences of *SL*, then (**P** $\equiv$ **Q**) is a sentence of *SL*.
7. Nothing is a sentence of *SL* unless it can be formed by repeated application of clauses 1–6.

It is trivial to show that every atomic sentence of *SL*—that is, every sentence formed in accordance with clause 1—has an equal number of left and right parentheses (namely, zero), because atomic sentences contain no parentheses. All other sentences of *SL* are formed in accordance with clauses 2–6. We note that in each of these cases an equal number of outermost left and right parentheses are added to those already occurring in the sentence's immediate components to form the new sentence (zero of each in clause 2, one of each in clauses 3–6). Therefore, if we can be sure that the immediate components **P** and **Q** of sentences formed in accordance with clauses 2–6 themselves contain an equal number of left and right parentheses, then we may conclude that the

sentences produced by these clauses will also contain an equal number of left and right parentheses.

How can we be sure, though, that each of the immediate components of a compound sentence *does* contain an equal number of left and right parentheses? We can start with compound sentences with one occurrence of a connective such as '~ A', '(A ⊃ B)', and '(A & B)'. Every sentence that contains one occurrence of a connective has one of the forms ~ **P**, (**P** & **Q**), (**P** ∨ **Q**), (**P** ⊃ **Q**), or (**P** ≡ **Q**). Moreover, in each case the immediate component or components are atomic. We have already noted that every atomic sentence has an equal number of left and right parentheses (namely, zero), and so, because clauses 2–6 each add an equal number of left and right parentheses to those already occurring in its immediate components, every compound sentence with one occurrence of a connective must also have an equal number of left and right parentheses.

Now consider truth-functionally compound sentences that contain two occurrences of connectives—sentences like '~ ~ A', '~ (A ∨ B)', '(A ∨ ~ B)', '((A ≡ B) ⊃ C)', and '(A ∨ (B & C))'. We may reason as we did in the previous paragraph. That is, every sentence that contains two occurrences of connectives has one of the forms ~ **P**, (**P** & **Q**), (**P** ∨ **Q**), (**P** ⊃ **Q**), or (**P** ≡ **Q**). And in each case the immediate component or components each have fewer than two occurrences of connectives. We have already found that, every sentence that contains fewer than two occurrences of connectives (atomic sentences and sentences containing one occurrence of a connective) has an equal number of left and right parentheses. Therefore, because clauses 2–6 each add an equal number of left and right parentheses to those already occurring in its immediate components, we may conclude that every compound sentence with two occurrences of connectives also has an equal number of left and right parentheses.

The same pattern of reasoning can be used for sentences with *three* occurrences of connectives, such as '~ ~ ~ A', '~ (~ A ∨ B)', '((A ⊃ B) & (A ∨ C))', and '(~ (A ≡ B) ≡ C)'. In every sentence that has three occurrences of connectives, the immediate components each contain fewer than three occurrences of connectives. We have already shown that every sentence of *SL* that contains fewer than three occurrences of connectives has an equal number of left and right parentheses. Therefore, because clauses 2–6 each add an equal number of left and right parentheses, we may conclude that every sentence that contains three occurrences of connectives has an equal number of left and right parentheses. The same reasoning can now be used to show that the claim holds for every sentence with four occurrences of connectives, then for every sentence with five occurrences of connectives, and so on. But of course we cannot consider each case individually—that is, each number of occurrences of connectives a sentence of *SL* might have, because there are an infinite number of such cases. But because there is a commonality of reasoning that can be used for each case we can use *mathematical induction* to generalize, reasoning as follows:

> Every sentence of *SL* with zero occurrences of connectives—that is, every atomic sentence of *SL*—has an equal number of left and right parentheses.

If every sentence of *SL* with **k** or fewer occurrences of connectives has an equal number of left and right parentheses, then every sentence of *SL* with **k** + 1 occurrences of connectives also has an equal number of left and right parentheses.

Therefore every sentence of *SL* has an equal number of left and right parentheses.

(Here we use '**k**' as a variable ranging over the nonnegative integers, that is, the positive integers plus zero.) This argument is logically valid—if the premises are true, then the conclusion is true as well. The first premise is our claim about sentences with no connectives, and the second premise therefore says that it follows that the claim also holds for sentences containing one occurrence of a connective. If the claim holds for all sentences containing zero or one occurrences of connectives, the second premise also assures us that the claim must also hold for sentences containing two occurrences of connectives. If the claim holds for all sentences containing zero, one, or two occurrences of connectives, the second premise also assures us that the claim also holds for sentences containing three occurrences of connectives, and so on for sentences with any number of occurrences of connectives. Because the argument is logically valid, we can establish that its conclusion is true by showing that both premises are true.

We have already shown that the first premise is true. Sentences that contain zero occurrences of connectives are atomic sentences, and atomic sentences are simply sentence letters. The first premise is called the '**basis clause**' of the argument.

The second premise of the argument is called the '**inductive step**', and its antecedent is called the **'inductive hypothesis**'. We shall prove that the inductive step is true by generalizing on the reasoning that we have already used. We shall assume that the inductive hypothesis is true—that is, that every sentence of *SL* containing **k** or fewer occurrences of connectives has an equal number of left and right parentheses—and we will show that it follows that every sentence **P** with **k** + 1 occurrences of connectives also has an equal number of left and right parentheses. Since **k** is nonnegative, **k** + 1 is positive, and hence such a sentence **P** contains at least one occurrence of a connective. So **P** will be a compound sentence that has one of the forms ~ **Q**, (**Q** & **R**), (**Q** ∨ **R**), (**Q** ⊃ **R**), or (**Q** ≡ **R**). We divide these forms into two cases.

Case 1: P has the form ~ **Q**. If ~ **Q** contains **k** + 1 occurrences of connectives, then **Q** contains **k** occurrences of connectives. By the inductive hypothesis (that every sentence containing **k** or fewer connectives has an equal number of left and right parentheses), **Q** has an equal number of left and right parentheses. But ~ **Q** contains all the parentheses occurring in **Q** and no others. So ~ **Q** contains an equal number of left and right parentheses as well.

Case 2: P has one of the forms (**Q** & **R**), (**Q** ∨ **R**), (**Q** ⊃ **R**), or (**Q** ≡ **R**). In each instance, if **P** contains **k** + 1 occurrences of connectives, then its immediate components, **Q** and **R**, must each contain **k** or fewer occurrences of connectives. By the inductive hypothesis, then:

a. **Q** has an equal number of left and right parentheses. Call this number **m**.
b. **R** has an equal number of left and right parentheses. Call this number **n**.

The number of left parentheses in **P** is therefore **m** + **n** + 1 (the 1 is for the outer-left parentheses that is added when **P** is formed from **Q** and **R**), and similarly, the number of right parentheses in **P** is **m** + **n** + 1. Therefore, the number of left parentheses in **P** equals the number of right parentheses in **P**.

This completes our proof that the second premise, the inductive step, is true. Having established that both premises are true, we may conclude that the conclusion is true as well. Every sentence of *SL* has an equal number of left and right parentheses.

We may now generally characterize arguments by mathematical induction. First, we group the items about which we wish to prove some claim into a series of cases, each associated with a nonnegative integer **k**. In our example, we arranged the sentences of *SL* into the series: sentences with zero occurrences of connectives, sentences with one occurrence of a connective, sentences with two occurrences of connectives, and so on. Every sentence of *SL* falls into one of these cases. The argument by mathematical induction then takes the following form:[1]

The claim holds for every member of the first group in the series.

If the claim holds for every member of every group associated with an integer less than **k**, then the claim holds for every member of the group associated with the integer **k**.

The thesis holds for every member of every group in the series.

All arguments of this form are valid. Of course, only those with true premises are sound. So to establish that the claim holds for every member of every group, we must show that the claim does hold for every member of the first group and that, no matter what subsequent group in the series we may choose, the claim holds for every member of that group if it holds for every member of every prior group. Again, the first premise of an argument by mathematical

[1]Strictly speaking, this is the form of arguments by *strong* mathematical induction. There is another type of mathematical induction, known as *weak* induction. We shall use only the strong variety of mathematical induction in this text. There is no loss here, for every claim that can be proven by weak mathematical induction can also be proven by strong mathematical induction.

induction is the **basis clause,** the second premise is the **inductive step,** and the antecedent of the second premise is the **inductive hypothesis.**

We'll illustrate mathematical induction with another example. Let **P** be a sentence of *SL* that contains only '~', '∨', and '&' as connectives, and let **P′** be the sentence that results from:

a. Replacing each occurrence of '∨' in **P** with '&'
b. Replacing each occurrence of '&' in **P** with '∨'
c. Adding a '~' in front of each atomic component of **P**.

We shall call a sentence that contains only '~', '∨', and '&' as connectives a **TWA** sentence (short for '*t*ilde, *w*edge, and *a*mpersand'), and we shall call the sentence **P′** that results from **P** by (a), (b), and (c) the **dual** of **P**. Here are some examples of duals for TWA sentences:

P	*Dual of* **P**
A	~ A
((A ∨ F) & G)	((~ A & ~ F) ∨ ~ G)
(((B & C) & C) ∨ D)	(((~ B ∨ ~ C) ∨ ~ C) & ~ D)
~ ((A ∨ ~ B) ∨ (~ A & ~ B))	~ ((~ A & ~ ~ B) & (~ ~ A ∨ ~ ~ B))

We shall use mathematical induction to establish the following thesis:

Every TWA sentence **P** is such that **P** and its dual **P′** have opposite truth-values on each truth-value assignment (that is, if **P** is true then **P′** is false, and if **P** is false then **P′** is true).

As in the previous example, our series will classify sentences by the number of occurrences of connectives that they contain:

Basis clause: Every TWA sentence **P** with zero occurrences of connectives is such that **P** and its dual **P′** have opposite truth-values on each truth-value assignment.

Inductive step: If every TWA sentence **P** with **k** or fewer occurrences of connectives is such that **P** and its dual **P′** have opposite truth-values on each truth-value assignment, then every TWA sentence **P** with **k** + 1 occurrences of connectives is such that **P** and its dual **P′** have opposite truth-values on each truth-value assignment.

Conclusion: Every TWA sentence **P** of *SL* is such that **P** and its dual **P′** have opposite truth-values on each truth-value assignment.

To show that the conclusion of this argument is true, we must show that the first premise, the basis clause, is true and also that the second premise, the inductive step, is true.

Proof of basis clause: A TWA sentence **P** that contains zero occurrences of connectives must be an atomic sentence, and its dual is simply ~ **P**. If **P** is true on a truth-value assignment, then according to the characteristic truth-table for the tilde, ~ **P** must be false. And if **P** is false on a truth-value assignment, then ~ **P** is true. We conclude that **P** and its dual have opposite truth-values on each truth-value assignment.

Proof of inductive step: We assume that the inductive hypothesis is true for all sentences with fewer than **k** + 1 occurrences of connectives—that is, that every TWA sentence **P** with fewer than **k** + 1 occurrences of connectives is such that **P** and its dual **P′** have opposite truth-values on each truth-value assignment. We must show that it follows from this assumption that the claim is also true of all TWA sentences **P** with **k** + 1 occurrences of connectives. A TWA sentence **P** that contains **k** + 1 occurrences of connectives must be compound, and because it is a TWA sentence, it must have one of the three forms ~ **Q**, (**Q** ∨ **R**), or (**Q** & **R**). We will consider each form.

Case 1: P has the form ~ **Q**. If ~ **Q** contains **k** + 1 occurrences of connectives, then **Q** contains **k** occurrences of connectives, and **Q** is a TWA sentence (if it were not—if it contained a horseshoe or triple bar—then ~ **Q** would not be a TWA sentence either). Let **Q′** be the dual of **Q**. Then the dual of ~ **Q** is ~ **Q′**, the sentence that results from ~ **Q** by changing **Q** in accordance with (a), (b), and (c) of our definition of dual sentences and leaving the initial tilde of ~ **Q** intact.

Because **Q** is a TWA sentence with fewer than **k** + 1 occurrences of connectives, it follows from the inductive hypothesis that **Q** and its dual **Q′** have opposite truth-values on each truth-value assignment. Therefore, by the characteristic truth-table for negation, ~ **Q** and ~ **Q′** will also have opposite truth-values on each truth-value assignment.

Case 2: P has the form (**Q** ∨ **R**). If (**Q** ∨ **R**) contains **k** + 1 occurrences of connectives, then **Q** and **R** each contain **k** or fewer occurrences of connectives. **Q** and **R** must also be TWA sentences. (Again, if either of them were not, then **P** would not be a TWA sentence.) Let **Q′** be the dual of **Q** and **R′** be the dual of **R**. Then the dual of **P** is (**Q′** & **R′**)—the result of making the changes specified by (a), (b), and (c) within **Q** and within **R** and replacing the main connective '∨' of (**Q** ∨ **R**) with '&'.

If (**Q** ∨ **R**) is true on a truth-value assignment, then by the characteristic truth-table for the wedge, either **Q** is true or **R** is true. Because **Q** and **R** each contain **k** or fewer occurrences of connectives, it follows from the inductive hypothesis that either **Q′** is false or **R′** is false. Either way, (**Q′** & **R′**), the dual of (**Q** ∨ **R**), must be false as well. On the other hand, if (**Q** ∨ **R**) is false on a truth-value assignment, then both **Q** and **R** must be false on that assignment. It follows from the inductive hypothesis that both **Q′** and **R′** are true on that assignment,

so (**Q**′ & **R**′) is true as well. We conclude that (**Q** ∨ **R**) and its dual have opposite truth-values on each truth-value assignment.

Case 3: **P** has the form (**Q** & **R**). If **P** contains **k** + 1 occurrences of connectives, then **Q** and **R** each contain **k** or fewer occurrences of connectives. And they must also be TWA sentences. Let **Q**′ be the dual of **Q** and **R**′ the dual of **R**. Then the dual of **P** is (**Q**′ ∨ **R**′)—the result of making the changes specified by (a), (b), and (c) within **Q** and within **R** and replacing the main connective of (**Q** & **R**) with '∨'.

If (**Q** & **R**) is true on a truth-value assignment, then, by the characteristic truth-table for the ampersand, both **Q** and **R** are true on that truth-value assignment. Because **Q** and **R** each contain **k** or fewer occurrences of connectives, it follows from the inductive hypothesis that **Q**′ and **R**′ are both false on that assignment, and therefore that the dual of (**Q** & **R**), (**Q**′ ∨ **R**′), is also false on that assignment. If (**Q** & **R**) is false on a truth-value assignment, then either **Q** is false or **R** is false on that assignment. If **Q** is false, then it follows by the inductive hypothesis that **Q**′ is true. If **R** is false on that assignment, then it follows by the inductive hypothesis that **R**′ is true. So at least one of **Q**′ and **R**′ is true on the assignment in question, and (**Q**′ ∨ **R**′), the dual of (**Q** & **R**), must also be true on that assignment. We conclude that (**Q** & **R**) and its dual have opposite truth-values on each truth-value assignment.

These three cases establish the inductive step of the mathematical induction, and we may now conclude that its conclusion is true as well. Our argument shows that the thesis about duals is true of every TWA sentence of *SL*.

6.1**E** EXERCISES

1. Prove the following theses by mathematical induction.

a. No sentence of *SL* that contains only binary connectives, if any, is truth–functionally false (that is, every truth-functionally false sentence of *SL* contains at least one '~').

b. Every sentence of *SL* that contains no binary connectives is truth-functionally indeterminate.

c. If two truth-value assignments **A**′ and **A**″ assign the same truth-values to the atomic components of a sentence **P**, then **P** has the same truth-value on **A**′ and **A**″.

d. An iterated conjunction (. . . (**P**$_1$ & **P**$_2$) & . . . & **P**$_\mathbf{n}$) of sentences of *SL* is true on a truth-value assignment if and only if **P**$_1$, **P**$_2$ . . . , **P**$_\mathbf{n}$ are all true on that assignment.

e. Where **P** is a sentence of *SL* and **Q** is a sentential component of **P**, let [**P**] (**Q**$_1$//**Q**) be a sentence that is the result of replacing at least one occurrence of **Q** in **P** with the sentence **Q**$_1$. If **Q** and **Q**$_1$ are truth-functionally equivalent, then **P** and [**P**] (**Q**$_1$//**Q**) are truth-functionally equivalent.

2. Consider this claim:

> No sentence of *SL* that contains only binary connectives is truth-functionally true.
>
> Show that this claim is false by producing a sentence that contains only binary connectives and that is truth-functionally true. Explain where an attempt to prove the claim by mathematical induction (in the manner of the answer to Exercise 1.a) would fail.

6.2 TRUTH-FUNCTIONAL COMPLETENESS

In Chapter 2 we defined the truth-functional use of sentential connectives as follows:

> A sentential connective, of a formal or a natural language, is used *truth-functionally* if and only if it is used to generate a compound sentence from one or more sentences in such a way that the truth-value of the generated compound is wholly determined by the truth-values of those one or more sentences from which the compound is generated, no matter what those truth-values may be.

The connectives of *SL* have only truth-functional uses since their intended interpretations are given wholly by their characteristic truth-tables. Although *SL* contains only five sentential connectives, we found in Chapter 2 that a great variety of English compounds can nevertheless be adequately symbolized using various combinations of these connectives. For instance, an English sentence of the form

> Neither **p** nor **q**

can be appropriately symbolized either by a sentence of the form

> ~ (**P** ∨ **Q**)

or by a sentence of the form

> ~ **P** & ~ **Q**

An interesting question now arises: Is *SL* capable of representing all possible truth-functionally compound sentences? We want the answer to this question to be 'yes', because we want *SL* to be an adequate vehicle for all of truth-functional logic. If there is some way of truth-functionally compounding sentences that cannot be represented in *SL*, then there may be some truth-functionally valid arguments, for example, that do not have valid symbolizations in *SL* simply because they cannot be adequately symbolized in *SL*.

To settle this question, we might try to produce complicated examples of truth-functionally compound sentences of English and then show that each can be adequately symbolized in *SL*. But obviously we cannot prove in this way that every truth-functionally compound sentence can be adequately symbolized in *SL*, for there are infinitely many possibilities. Rather, we shall use mathematical induction to show that *all* possible truth-functionally compound sentences can be adequately symbolized in *SL*.

First, though, we must formulate our question somewhat more precisely: Can every **truth-function** be expressed by a sentence of *SL*? A *truth-function* is an **n**-place function (where **n** is a positive integer) that maps each combination of **n** truth-values to a single truth-value. The truth-values that are mapped are the **arguments** of the truth-function, and the truth-value to which each combination of truth-values is mapped is the **value** of the truth-function for those arguments. For example, the characteristic truth-table for '⊃' defines the material conditional truth-function:

P	**Q**	**P ⊃ Q**
T	**T**	**T**
T	**F**	**F**
F	**T**	**T**
F	**F**	**T**

This truth-function is a truth-function of *two* arguments. Each distinct combination of arguments for the function is listed to the left of the vertical line, and the truth-value to which that combination of arguments is mapped is listed to the right of the vertical line.

A truth-function is said to be **expressed** in *SL* by any sentence whose truth-table contains (in the column under its main connective) exactly the column of **T**s and **F**s that occurs on the right-hand side of the characteristic truth-table for the truth-function in question. For example, each sentence of the form **P** ⊃ **Q**, where **P** and **Q** are atomic sentences of *SL*, expresses the material conditional truth-function—for every such sentence has a four-row truth-table in which the column under the main connective contains a **T** in the first, third, and fourth rows and an **F** in the second row. This truth-function is also expressed by other sentences of *SL*—for example, by all sentences of the form ~ **P** ∨ **Q**, where **P** and **Q** are atomic sentences. Every such sentence has a four-row truth-table in which the column under the main connective is

T
F
T
T

The important question for us is not how many sentences of *SL* express the same truth-function but rather whether there is *at least one* sentence of *SL* that expresses each truth-function. There are an infinite number of truth-functions. This is most easily seen by considering that for every positive integer **n** there are truth-functions of **n** arguments (truth-functions that map each combination of truth-values that **n** sentences of *SL* may have to a truth-value), and there are infinitely many positive integers. In Chapter 2 we defined one truth-function of one argument and four truth-functions of two arguments via the five characteristic truth-tables for the connectives of *SL*. There are three additional truth-functions of one argument that are distinct from the negation truth-function:

P	
T	**T**
F	**F**

P	
T	**T**
F	**T**

P	
T	**F**
F	**F**

And there are twelve other truth-functions of two arguments (because there are sixteen different ways of arranging **T**s and **F**s in a column of a four-row truth-table). Generally, where **n** is any positive integer, there are $2^{(2^{\mathbf{n}})}$ truth-functions of **n** arguments. So there are 256 truth-functions of three arguments, 65,536 truth-functions of four arguments, and so on. What we want to show is that, given any truth-function of any finite number of arguments, there is at least one sentence of *SL* that expresses that truth-function. In fact, we shall prove something even stronger:

Metatheorem 6.2.1: Every truth-function can be expressed by a sentence of *SL* that contains no sentential connectives other than '~', '∨', and '&'.

The connectives of a language in which every truth-function can be expressed form a **truth-functionally complete** set of connectives. Metatheorem 6.2.1 says that the set of connectives {'~', '∨', '&'}, defined as they are defined in *SL*, is truth-functionally complete.

Characteristic truth-tables define truth-functions by exhaustively listing the combinations of arguments that each truth-function takes and displaying the value to which each such combination is mapped. We will specify truth-functions in the same tabular form. Thus,

T	**T**	**F**
T	**F**	**F**
F	**T**	**F**
F	**F**	**T**

specifies a truth-function of two arguments. Since every truth-function maps only a finite number of combinations of arguments, every truth-function can be specified in such a table. We call this table a **truth-function schema.**

We shall now show that the set of connectives {'~', '&', '∨'} is truth-functionally complete by producing an **algorithm** for constructing, given any truth-function schema, a sentence of *SL* that contains no connectives other than '~', '&', and '∨' that expresses the truth-function specified by the schema. An algorithm is an **effective procedure** for producing a desired result—that is, a mechanical procedure that, when correctly followed, yields the desired result in a finite number of steps. Given a truth-function schema, our algorithm will produce a sentence whose truth-table contains, under its main connective, exactly the same column of **T**s and **F**s as occurs to the right of the vertical line in the truth-function schema. Once we produce the algorithm, Metatheorem 6.2.1 will be proven; the construction of such an algorithm will show that every truth-function can be expressed by a sentence of *SL* containing no connectives other than '~', '&', and '∨'.

To begin, we need a stock of atomic sentences. If the truth-function is a function of **n** arguments, we use the alphabetically first **n** atomic sentences of *SL*.[2] So for the truth-function schema

T	**T**	**F**
T	**F**	**F**
F	**T**	**F**
F	**F**	**T**

we use the letters 'A' and 'B':

A	B	
T	**T**	**F**
T	**F**	**F**
F	**T**	**F**
F	**F**	**T**

Next for each row of the truth-table schema, we form a sentence that is true if and only if its atomic components have the truth-values indicated in that row. This sentence is called the **characteristic sentence** for the row in question. As an example, we display the characteristic sentences for the four rows of our truth-function schema to the left of each row:

	A	B	
A & B	**T**	**T**	**F**
A & ~ B	**T**	**F**	**F**
~ A & B	**F**	**T**	**F**
~ A & ~ B	**F**	**F**	**T**

[2]We use an extended sense of alphabetical order in which all of the nonsubscripted sentence letters appear first, in the usual alphabetical order, then all of the letters (in the same order) subscripted with '1', then all of the letters subscripted with '2', and so on.

Our construction of these sentences was guided by the combinations of truth-values under 'A' and 'B': the conjunction includes the sentence letter itself if the sentence letter has the truth-value **T** in that row, and the conjunction includes the negation of the sentence letter if the sentence letter has the truth-value **F** in that row. The conjunction that has been constructed for each row has the truth-value **T** when 'A' and ' B' have the truth-values indicated in that row and has the truth-value **F** for every other combination of truth-values. In general, the characteristic sentence for row **i** of a truth-function schema is the iterated conjunction

$$(\ldots (\mathbf{P}_1 \,\&\, \mathbf{P}_2) \,\&\, \ldots \,\&\, \mathbf{P}_{\mathbf{n}})$$

in which $\mathbf{P_j}$ is the **j**th atomic sentence if the **j**th value in row **i** (to the left of the vertical bar) is **T**, and $\mathbf{P_j}$ is the negation of the **j**th atomic sentence if the **j**th value in row **i** is **F**. We leave it as an exercise to prove that the characteristic sentence for each row of a truth-function schema is true if and only if its atomic components, alphabetically ordered, have the truth-values presented in that row.

Finally we identify the rows in the truth-function schema that have a **T** to the right of the vertical bar. If there is only one such row, then the characteristic sentence for that row is a sentence that expresses the truth-function specified in the schema. In our example the fourth row is the only row that has a **T** to the right of the vertical bar, and the characteristic sentence for that row is '~ A & ~ B'. This sentence is true if and only if both 'A' and 'B' are false, and therefore this sentence expresses the truth-function specified by the truth-function schema:

			↓	
A	B	~ A	&	~ B
T	**T**	**F T**	**F**	**F T**
T	**F**	**F T**	**F**	**T F**
F	**T**	**T F**	**F**	**F T**
F	**F**	**T F**	**T**	**T F**

If the truth-function schema has more than one **T** to the right of the vertical bar, as does the following,

T	**T**	**F**
T	**F**	**T**
F	**T**	**F**
F	**F**	**T**

then we form an iterated disjunction of the characteristic sentences for the rows that have a **T** to the right of the vertical bar. In the present case the disjunction is '(A & ~ B) ∨ (~ A & ~ B)'—the disjunction of the characteristic sentences for the second and fourth rows. This sentence is true if and only if either 'A'

is true and ‘B’ is false or both ‘A’ and ‘B’ are false, and it therefore expresses the truth-function specified in this schema:

						↓				
A	B	(A	&	~ B)	∨	(~ A	&	~ B)		
T	**T**	**T**	**F**	**F T**	**F**	**F T**	**F**	**F T**		
T	**F**	**T**	**T**	**T F**	**T**	**F T**	**F**	**T F**		
F	**T**	**F**	**F**	**F T**	**F**	**T F**	**F**	**F T**		
F	**F**	**F**	**F**	**T F**	**T**	**T F**	**T**	**T F**		

And if the schema is

T	**T**	**F**
T	**F**	**T**
F	**T**	**T**
F	**F**	**T**

then the disjunction of the characteristic sentences for the last three rows, ‘((A & ~ B) ∨ (~ A & B)) ∨ (~ A & ~ B)’, expresses the truth-function in the schema:

										↓				
A	B	((A	&	~ B)	∨	(~ A	&	B))	∨	(~ A	&	~ B)		
T	**T**	**T**	**F**	**F T**	**F**	**F T**	**F**	**T**	**F**	**F T**	**F**	**F T**		
T	**F**	**T**	**T**	**T F**	**T**	**F T**	**F**	**F**	**T**	**F T**	**F**	**T F**		
F	**T**	**F**	**F**	**F T**	**T**	**T F**	**T**	**T**	**T**	**T F**	**F**	**F T**		
F	**F**	**F**	**F**	**T F**	**F**	**T F**	**F**	**F**	**T**	**T F**	**T**	**T F**		

In general, when more than one row has a **T** to the right of the vertical bar, the iterated disjunction that we form from the characteristic sentences for those rows will be true if and only if at least one of its disjuncts is true, and because each disjunct is true only in the row for which it is a characteristic sentence, it follows that the iterated disjunction is true if and only if its atomic components have the truth-values specified by one of the rows that have a **T** to the right of the vertical bar. Thus the disjunction expresses the truth-function specified by that schema.

We must consider one final case. There may be no **T**s in the column to the right of the vertical bar in a truth-function schema. In this case, we will produce a truth-functionally false sentence by conjoining the characteristic sentence for the first row of the truth-function schema with its negation. (Any other row’s characteristic sentence would have done as well.) Because this sentence has the form **P** & ~ **P**, it expresses a truth-function that maps every combination of **n** truth-values to **F**. For example, if our schema is

T	**T**	**F**
T	**F**	**F**
F	**T**	**F**
F	**F**	**F**

then the sentence '(A & B) & ~ (A & B)' expresses the truth-function specified in the schema:

					↓				
A	B	(	(	A	&	B)	&	~	(A & B))
T	**T**	**T**	**T**	**T**	**F**	**F**	**T**	**T**	**T**
T	**F**	**T**	**F**	**F**	**F**	**T**	**T**	**F**	**F**
F	**T**	**F**	**F**	**T**	**F**	**T**	**F**	**F**	**T**
F	**F**	**F**	**F**	**F**	**F**	**T**	**F**	**F**	**F**

In sum, there are three cases to consider:

1. The given truth-function schema contains exactly one row in which there is a **T** to the right of the vertical line. In this case, the truth-function is expressed by the characteristic sentence for that row.
2. The given truth-function schema contains more than one row in which there is a **T** to the right of the vertical line. In this case, the truth-function is expressed by the iterated disjunction of the characteristic sentences for those rows.
3. The given truth-function schema has no **T**s to the right of the vertical line. In this case, the truth-function is expressed by the conjunction of the characteristic sentence for the first row of the truth-function schema and the negation of that sentence.

As a final example, we will use the algorithm we have developed to construct a sentence of *SL* containing only the connectives '~', '∨', and '&' that expresses the truth-function specified in the schema

T	**T**	**T**	**F**
T	**T**	**F**	**F**
T	**F**	**T**	**T**
T	**F**	**F**	**T**
F	**T**	**T**	**F**
F	**T**	**F**	**F**
F	**F**	**T**	**T**
F	**F**	**F**	**F**

This schema falls under our second case; there is more than one **T** to the right of the vertical line. We shall use the first three sentence letters of *SL*, because the truth-function is a truth-function of three arguments. We form the characteristic sentences for rows 3, 4, and 7 and then disjoin those characteristic sentences to produce

$$((((A \mathbin{\&} {\sim} B) \mathbin{\&} C) \vee ((A \mathbin{\&} {\sim} B) \mathbin{\&} {\sim} C)) \vee (({\sim} A \mathbin{\&} {\sim} B) \mathbin{\&} C))$$

This sentence is true if and only if 'A', 'B', and 'C' have one of the combinations of truth-values represented in the third, fourth, and seventh rows of the schema.

Our algorithm shows how to construct, for any truth-function, a sentence of *SL* that expresses that truth-function. It therefore shows that for each truth-function there is *at least one* sentence of *SL* that expresses that truth-function. Moreover, because we have used only the three connectives '~', '&', and '∨', we have shown that the set of connectives {'~', '&', '∨'} is truth–functionally complete. This completes the proof of Metatheorem 6.2.1.

There is a consequence of the theorem that follows almost immediately: The smaller set {'~', '∨'} is also truth-functionally complete. Every conjunction (**P** & **Q**) is truth-functionally equivalent to ~ (~ **P** ∨ ~ **Q**), and so we may rewrite each sentence produced by the algorithm using only '~' and '∨'. For example, the sentence

(~ (~ A ∨ ~ ~ B) ∨ ~ (~ ~ A ∨ ~ ~ B))

expresses the same truth-function as

((A & ~ B) ∨ (~ A & ~ B))

Therefore, every truth-function can be expressed by a sentence that contains only '~' and '∨' as connectives. It is also a consequence of Metatheorem 6.2.1 that the sets of connectives {'~', '&'} and {'~', '⊃'} are truth-functionally complete; we leave the proofs as an exercise.

On the other hand, the set of connectives {'∨', '&'} is *not* truth-functionally complete. To prove this, we must show that there is at least one truth-function that cannot be expressed by any sentence that contains at most the connectives '∨' and '&'. We call such a sentence a **W-A** sentence (short for '*w*edge and *a*mpersand'). A little reflection suggests that, no matter how many times we conjoin and disjoin, if we do not have the tilde available we can never produce a false sentence from atomic components that are all true. That is, every W-A sentence is true whenever its atomic components are all true. And if this is the case, then there are many truth-functions that cannot be expressed by any W-A sentence. Take the negation truth-function as an example. This truth-function maps the argument **T** into the value **F**. If our reflection is correct, there is no false W-A sentence with a single atomic component when that atomic component is true.

We shall therefore show that the set of connectives {'∨', '&'} is not truth-functionally complete by proving the following thesis:

> Every W-A sentence has the truth-value **T** on every truth-value assignment on which its atomic components all have the truth-value **T**.

Because this is a general claim about *all* W-A sentences, we shall prove the thesis by mathematical induction.

The shortest W-A sentences—that is, those with zero occurrences of connectives, are simply the atomic sentences of *SL*.

Basis clause: Every atomic sentence of *SL* has the truth-value **T** on every truth-value assignment on which its atomic components all have the truth-value **T**.

Proof of basis clause: The basis clause is obviously true, since an atomic sentence is itself its only component.

Inductive step: If every W-A sentence of *SL* with **k** or fewer occurrences of connectives has the truth-value **T** on every truth-value assignment on which its atomic components all have the truth-value **T**, then every W-A sentence with **k** + 1 occurrences of connectives has the truth-value **T** on every truth-value assignment on which its atomic components all have the truth-value **T**.

Proof of inductive step: We assume that the inductive hypothesis is true for an arbitrary nonnegative integer **k**; that is, we assume that every W-A sentence with **k** or fewer occurrences of connectives is true whenever all its atomic components are true. We must show that it follows that the thesis also holds for any W-A sentence **P** with **k** + 1 occurrences of connectives. Since these sentences contain only '∨' and '&' as connectives, there are two cases to consider.

Case 1: P has the form **Q** ∨ **R**. **Q** and **R** each contain fewer than **k** + 1 occurrences of connectives, and they are also W-A sentences. So, by the inductive hypothesis, each disjunct is true on every truth-value assignment on which each of its atomic components is true. So, if all the atomic components of **Q** ∨ **R** are true, then both **Q** and **R** are true, and hence **Q** ∨ **R** is itself true.

Case 2: P has the form **Q** & **R**. Then each of **Q** and **R** is a W-A sentence with **k** or fewer occurrences of connectives. Hence the inductive hypothesis holds for both **Q** and **R**. Each conjunct is true on every truth-value assignment on which all its atomic components are true. So, if all the atomic components of **Q** & **R** are true, then both **Q** and **R** are true, and hence **Q** & **R** itself is true.

This proves the inductive step, and we can conclude that the thesis holds for every W-A sentence:

Conclusion: Every W-A sentence has the truth-value **T** on every truth-value assignment on which its atomic components all have the truth-value **T**.

It follows that no W-A sentence can express the negation truth-function as defined in the characteristic truth-table for the tilde, since no W-A sentence can express a truth-function that maps the truth-value **T** to the truth- value **F**.

6.2**E** EXERCISES

1. Show that a sentence constructed in accordance with our characteristic sentence algorithm is indeed a characteristic sentence for the row of the truth-function schema in question.

2. Using the algorithm in the proof of Metatheorem 6.2.1, construct a sentence containing at most '~', '&', and '∨' that expresses the truth-function defined in each of the following truth-function schemata.

a.

T	**T**	**F**
T	**F**	**T**
F	**T**	**F**
F	**F**	**T**

b.

T	**F**
F	**F**

*c.

T	**T**	**F**
T	**F**	**T**
F	**T**	**T**
F	**F**	**F**

d.

T	**T**	**T**	**T**
T	**T**	**F**	**T**
T	**F**	**T**	**F**
T	**F**	**F**	**F**
F	**T**	**T**	**F**
F	**T**	**F**	**F**
F	**F**	**T**	**T**
F	**F**	**F**	**F**

3. Present an algorithm analogous to that in Metatheorem 6.2.1 for constructing a characteristic sentence containing only '~' and '∨' for each row of a truth-function schema.

4. Using Metatheorem 6.2.1, prove that the sets {'~', '&'} and {'~', '⊃'} are truth-functionally complete.

5. Prove that the set consisting of the dagger '↓' is truth-functionally complete, where the dagger has the following characteristic truth-table:

P	**Q**	**P ↓ Q**
T	**T**	**F**
T	**F**	**F**
F	**T**	**F**
F	**F**	**T**

*6. Prove that the set consisting of the stroke '|' is truth-functionally complete, where the stroke has the following characteristic truth-table:

P	**Q**	**P | Q**
T	**T**	**F**
T	**F**	**T**
F	**T**	**T**
F	**F**	**T**

7. Using the results of Exercises 1.a and 1.b in Section 6.1E, prove that the following sets of connectives are not truth-functionally complete: {'~'}, {'&', '∨', '⊃', '≡'}.

8. Prove that the set {'~', '≡'} is not truth-functionally complete. *Hint:* Show that the truth-table for any sentence **P** that contains only these two connectives and just two atomic components will have an even number of **T**s and an even number of **F**s in the column under the main connective.

9. Prove that if a truth-functionally complete set of connectives consists of exactly one binary connective, then that connective has either the characteristic truth-table for '↓' or the characteristic truth-table for '|'. (That is, show that the connective must be either '↓' or '|', though possibly under a different name.) (*Hint:* In the proofs for Exercises 7 and 8 above, it became apparent that characteristic truth-tables for truth-functionally complete sets of connectives must have certain properties. Show that only two characteristic truth-tables with just four rows have these properties.)

6.3 THE SOUNDNESS OF *SD* AND *SD*+

We now turn to the results announced at the beginning of this chapter. In this section we shall prove that, if a sentence **P** of *SL* is derivable in *SD* from a set Γ of sentences of *SL*, then Γ truth-functionally entails **P**. A natural deduction system for which this result holds is said to be **sound** for sentential logic. In the next section we shall prove the converse—that if a set Γ of sentences of *SL* truth-functionally entails a sentence **P** of *SL*, then **P** is derivable in *SD* from Γ. A natural deduction system for which this second result holds is said to be **complete** for sentential logic. Soundness and completeness are important properties for natural deduction systems. A natural deduction system that is not sound will sometimes lead us from true sentences to false ones, and a natural deduction system that is not complete will not allow us to construct all the derivations that we want to construct. In either case the natural deduction system would not be adequate for the purposes of sentential logic.

Metatheorem 6.3.1 is the *Soundness Metatheorem* for *SD*.

Metatheorem 6.3.1: For any set Γ of sentences of *SL* and any sentence **P** of *SL*, if $\Gamma \vdash \mathbf{P}$ in *SD* then $\Gamma \models \mathbf{P}$[3].

Recall that $\Gamma \models \mathbf{P}$ if and only if there is no truth-value assignment on which all the members of Γ are true and **P** is false. Metatheorem 6.3.1 therefore says that the derivation rules of *SD* are *truth-preserving*.

Our proof will use mathematical induction to establish that each sentence in a derivation is true if all the open assumptions in whose scope the sentence lies are true. The basis clause will show that this claim is true of the

[3]In what follows we shall abbreviate '$\Gamma \vdash \mathbf{P}$ in *SD*' as '$\Gamma \vdash \mathbf{P}$'.

first sentence in a derivation. The inductive step will show that, if the claim is true for the first **k** sentences in a derivation, then the claim is also true for the (**k** + 1)th sentence—that is, each application of another derivation rule in the derivation is truth-preserving. We will then be able to conclude that the last sentence in any derivation, no matter how long the derivation is, is true if all the open assumptions in whose scope the sentence lies are true, which is what Metatheorem 6.3.1 says.

In the course of the proof, we shall use some set-theoretic terminology that we will explain here: Let Γ and Γ' be sets. If every member of Γ is also a member of Γ', then Γ is said to be a **subset** of Γ'. Note that every set is a subset of itself, and the empty set is trivially a subset of every set (because the empty set has no members, it has no members that are *not* members of every set). As an example, the set of sentences

{A, B, C}

has eight subsets: {A, B, C}, {A, B}, {B, C}, {A, C}, {A}, {B}, {C}, and ∅. If a set Γ is a subset of a set Γ', then Γ' is said to be a **superset** of Γ. Thus {A, B, C} is a superset of each of its eight subsets.

We will also make use of several semantic results that we prove here. First, if **P** is truth-functionally entailed by a set of sentences Γ, then **P** is truth-functionally entailed by every superset of Γ:

6.3.2: If $\Gamma \vDash \mathbf{P}$, then for every superset Γ' of Γ, $\Gamma' \vDash \mathbf{P}$.

Proof: Assume that $\Gamma \vDash \mathbf{P}$ and let Γ' be any superset of Γ. If every member of Γ' is true on some truth-value assignment, then every member of its subset Γ is true on that assignment, and so, because $\Gamma \vDash \mathbf{P}$, **P** is also true on the assignment. Therefore $\Gamma' \vDash \mathbf{P}$.

Second, we have two results that were proved in the exercises for Chapter 3:

6.3.3: If $\Gamma \cup \{\mathbf{Q}\} \vDash \mathbf{R}$, then $\Gamma \vDash \mathbf{Q} \supset \mathbf{R}$ (see Exercise 2.b in Section 3.6E).

6.3.4: If $\Gamma \vDash \mathbf{Q}$ and $\Gamma \vDash \sim \mathbf{Q}$ for some sentence **Q**, then Γ is truth-functionally inconsistent (see Exercise 3.b in Section 3.6E).

Finally, if a set of sentences is truth-functionally inconsistent, then every sentence **Q** in the set is such that the set consisting of all the *other* sentences in the set truth-functionally entails ~ **Q**:

6.3.5: If $\Gamma \cup \{\mathbf{Q}\}$ is truth-functionally inconsistent, then $\Gamma \vDash \sim \mathbf{Q}$.

Proof: Assume that $\Gamma \cup \{\mathbf{Q}\}$ is truth-functionally inconsistent. Then there is no truth-value assignment on which every member of $\Gamma \cup \{\mathbf{Q}\}$ is true. Therefore, if every member of Γ is true on some truth-value assignment, **Q** must be false on that assignment, and so ~ **Q** will be true. So $\Gamma \vDash \sim \mathbf{Q}$.

In our proof that each sentence in a derivation is truth-functionally entailed by the set of the open assumptions in whose scope the sentence lies, we use the following notation: For any derivation, let $\mathbf{P_k}$ be the **k**th sentence in the derivation, and let $\Gamma_\mathbf{k}$ be the set of open assumptions in whose scope $\mathbf{P_k}$ lies. Here is our argument by mathematical induction on the position **k** in a derivation:

Basis clause: $\Gamma_1 \vDash \mathbf{P}_1$.

Inductive step: If $\Gamma_\mathbf{i} \vDash \mathbf{P_i}$ for every positive integer $\mathbf{i} \leq \mathbf{k}$, then $\Gamma_{\mathbf{k}+1} \vDash \mathbf{P}_{\mathbf{k}+1}$.

Conclusion: For every positive integer **k**, $\Gamma_\mathbf{k} \vDash \mathbf{P_k}$.

Proof of basis clause: $\mathbf{P}_1$ is the first sentence in a derivation. Moreover, because every derivation in *SD* begins with one or more assumptions, $\mathbf{P}_1$ is an open assumption that lies in its own scope. (We remind the reader that, by definition, every assumption of a derivation lies within its own scope.) Therefore, the set Γ_1 of open assumptions in whose scope $\mathbf{P}_1$ lies is $\{\mathbf{P}_1\}$. Because $\{\mathbf{P}_1\} \vDash \mathbf{P}_1$, we conclude that the basis clause is true.

Proof of inductive step: Let **k** be an arbitrary positive integer and assume the inductive hypothesis: for every positive integer $\mathbf{i} \leq \mathbf{k}$, $\Gamma_\mathbf{i} \vDash \mathbf{P_i}$. We must show that it follows that $\Gamma_{\mathbf{k}+1} \vDash \mathbf{P}_{\mathbf{k}+1}$. We shall consider each way in which $\mathbf{P}_{\mathbf{k}+1}$ might be justified and show that our thesis holds no matter what justification is used. We now turn to cases.

Case 1: $\mathbf{P}_{\mathbf{k}+1}$ is an Assumption. Then $\mathbf{P}_{\mathbf{k}+1}$ is a member of $\Gamma_{\mathbf{k}+1}$, the set of open assumptions in whose scope $\mathbf{P}_{\mathbf{k}+1}$ lies. Therefore, if every member of $\Gamma_{\mathbf{k}+1}$ is true, $\mathbf{P}_{\mathbf{k}+1}$, being a member of the set, is true as well. So $\Gamma_{\mathbf{k}+1} \vDash \mathbf{P}_{\mathbf{k}+1}$.

Case 2: $\mathbf{P}_{\mathbf{k}+1}$ is justified by Reiteration. Then $\mathbf{P}_{\mathbf{k}+1}$ occurs earlier in the derivation as sentence $\mathbf{P_i}$ at some position **i**. Moreover every assumption that is open at position **i** must remain open at position **k** + 1—for if even one assumption in whose scope $\mathbf{P_i}$ lies were closed before position **k** + 1, $\mathbf{P_i}$ would not be accessible at position **k** + 1. Therefore, $\Gamma_\mathbf{i}$ is a subset of $\Gamma_{\mathbf{k}+1}$. By our inductive hypothesis, $\Gamma_\mathbf{i} \vDash \mathbf{P_i}$. Because $\Gamma_\mathbf{i}$ is a subset of $\Gamma_{\mathbf{k}+1}$, it follows, by 6.3.2, that $\Gamma_{\mathbf{k}+1} \vDash \mathbf{P_i}$. $\mathbf{P}_{\mathbf{k}+1}$ is the same sentence as $\mathbf{P_i}$, so $\Gamma_{\mathbf{k}+1} \vDash \mathbf{P}_{\mathbf{k}+1}$.

Case 3: $\mathbf{P}_{\mathbf{k}+1}$ is justified by Conjunction Introduction. The conjuncts of $\mathbf{P}_{\mathbf{k}+1}$ occur earlier in the derivation, say at positions **h** and **j**, both of which are accessible from position **k** + 1:

h	**Q**	
j	**R**	
k + 1	**Q** & **R** (= $\mathbf{P}_{\mathbf{k}+1}$)	**h**, **j** &I

(There may be open assumptions between positions **h** and **j** and between positions **j** and **k** + 1. Moreover it may be that **R** occurs earlier in the derivation than **Q** does—the order is immaterial.) By the inductive hypothesis, $\Gamma_{\mathbf{h}} \vDash \mathbf{Q}$ and $\Gamma_{\mathbf{j}} \vDash \mathbf{R}$. Moreover, because both **Q** and **R** are accessible at position **k** + 1, $\Gamma_{\mathbf{h}}$ and $\Gamma_{\mathbf{j}}$ are both subsets of $\Gamma_{\mathbf{k}+1}$ and so, by 6.3.2, $\Gamma_{\mathbf{k}+1} \vDash \mathbf{Q}$ and $\Gamma_{\mathbf{k}+1} \vDash \mathbf{R}$. But whenever both **Q** and **R** are true, $\mathbf{P}_{\mathbf{k}+1}$, which is **Q** & **R**, is also true. So $\Gamma_{\mathbf{k}+1} \vDash \mathbf{P}_{\mathbf{k}+1}$ as well.

Case 4: $\mathbf{P}_{\mathbf{k}+1}$ is justified by Conjunction Elimination:

	h	$\mathbf{Q} \,\&\, \mathbf{P}_{\mathbf{k}+1}$				**h**	$\mathbf{P}_{\mathbf{k}+1} \,\&\, \mathbf{Q}$
				or			
k + 1		$\mathbf{P}_{\mathbf{k}+1}$	**h** &E		**k** + 1	$\mathbf{P}_{\mathbf{k}+1}$	**h** &E

By the inductive hypothesis, $\Gamma_{\mathbf{h}}$ truth-functionally entails the conjunction at position **h**. And whenever the conjunction is true, both conjuncts must be true. So $\Gamma_{\mathbf{h}} \vDash \mathbf{P}_{\mathbf{k}+1}$. Moreover, $\Gamma_{\mathbf{h}}$ is a subset of $\Gamma_{\mathbf{k}+1}$, because the conjunction at position **h** is accessible at position **k** + 1. It follows, by 6.3.2, that $\Gamma_{\mathbf{k}+1} \vDash \mathbf{P}_{\mathbf{k}+1}$.

Case 5: $\mathbf{P}_{\mathbf{k}+1}$ is justified by Disjunction Introduction:

	h	**Q**				**h**	**R**
				or			
k + 1		$\mathbf{Q} \vee \mathbf{R}$ 3(= $\mathbf{P}_{\mathbf{k}+1}$)	**h** ∨I		**k** + 1	$\mathbf{Q} \vee \mathbf{R}$ (= $\mathbf{P}_{\mathbf{k}+1}$)	**h** ∨I

By the inductive hypothesis, $\Gamma_{\mathbf{h}}$ truth-functionally entails the sentence at position **h**. That sentence is one of the disjuncts of $\mathbf{Q} \vee \mathbf{R}$, so whenever it is true, so is $\mathbf{Q} \vee \mathbf{R}$. Thus $\Gamma_{\mathbf{h}} \vDash \mathbf{P}_{\mathbf{k}+1}$. $\Gamma_{\mathbf{h}}$ must be a subset of $\Gamma_{\mathbf{k}+1}$ if the sentence at position **h** is accessible at position **k** + 1, and so, by 6.3.2, $\Gamma_{\mathbf{k}+1} \vDash \mathbf{P}_{\mathbf{k}+1}$.

Case 6: $\mathbf{P}_{\mathbf{k}+1}$ is justified by Conditional Elimination:

	h	**Q**	
	j	$\mathbf{Q} \supset \mathbf{P}_{\mathbf{k}+1}$	
k + 1		$\mathbf{P}_{\mathbf{k}+1}$	**h**, **j** ⊃E

By the inductive hypothesis, $\Gamma_{\mathbf{h}} \vDash \mathbf{Q}$ and $\Gamma_{\mathbf{j}} \vDash \mathbf{Q} \supset \mathbf{P}_{\mathbf{k}+1}$. Both $\Gamma_{\mathbf{h}}$ and $\Gamma_{\mathbf{j}}$ must be subsets of $\Gamma_{\mathbf{k}+1}$ because the sentences at positions **h** and **j** are accessible at position **k** + 1. By 6.3.2, then, $\Gamma_{\mathbf{k}+1} \vDash \mathbf{Q}$ and $\Gamma_{\mathbf{k}+1} \vDash \mathbf{Q} \supset \mathbf{P}_{\mathbf{k}+1}$. Because $\mathbf{P}_{\mathbf{k}+1}$ must be true whenever both **Q** and $\mathbf{Q} \supset \mathbf{P}_{\mathbf{k}+1}$ are true, $\Gamma_{\mathbf{k}+1} \vDash \mathbf{P}_{\mathbf{k}+1}$ as well.

Case 7: $\mathbf{P_{k+1}}$ is justified by Biconditional Elimination:

h	**Q**			**h**	**Q**		
j	$\mathbf{Q \equiv P_{k+1}}$		or	**j**	$\mathbf{P_{k+1} \equiv Q}$		
$\mathbf{k + 1}$	$\mathbf{P_{k+1}}$	**h, j** ≡E		$\mathbf{k + 1}$	$\mathbf{P_{k+1}}$	**h, j** ≡E	

By the inductive hypothesis, $\Gamma_{\mathbf{h}} \vDash \mathbf{Q}$ and $\Gamma_{\mathbf{j}}$ truth-functionally entails the biconditional at position **j**. $\Gamma_{\mathbf{h}}$ and $\Gamma_{\mathbf{j}}$ must be subsets of $\Gamma_{\mathbf{k+1}}$ because the sentences at positions **h** and **j** are accessible at position $\mathbf{k + 1}$. By 6.3.2, then, $\Gamma_{\mathbf{k+1}}$ truth-functionally entails both **Q** and the biconditional at position **j**. Because the sentence $\mathbf{P_{k+1}}$ must be true whenever both **Q** and the biconditional at position **j** are true, $\Gamma_{\mathbf{k+1}} \vDash \mathbf{P_{k+1}}$ as well.

Case 8: $\mathbf{P_{k+1}}$ is justified by Conditional Introduction:

h	**Q**	
j	**R**	
$\mathbf{k + 1}$	$\mathbf{Q \supset R}$ $(= \mathbf{P_{k+1}})$	**h–j** ⊃I

By the inductive hypothesis, $\Gamma_{\mathbf{j}} \vDash \mathbf{R}$. Because the subderivation in which **R** is derived from **Q** is accessible at position $\mathbf{k + 1}$, every assumption that is open at position **j** is open at position $\mathbf{k + 1}$, except for the assumption **Q** that begins the subderivation. So the set of open assumptions $\Gamma_{\mathbf{j}}$ is a subset of $\Gamma_{\mathbf{k+1}} \cup \{\mathbf{Q}\}$. Because $\Gamma_{\mathbf{j}} \vDash \mathbf{R}$, it follows, by 6.3.2, that $\Gamma_{\mathbf{k+1}} \cup \{\mathbf{Q}\} \vDash \mathbf{R}$. And from this it follows, by 6.3.3, that $\Gamma_{\mathbf{k+1}} \vDash \mathbf{Q \supset R}$.

Case 9: $\mathbf{P_{k+1}}$ is justified by Negation Introduction:

h	**Q**	
j	**R**	
m	**~ R**	
$\mathbf{k + 1}$	**~ Q** $(= \mathbf{P_{k+1}})$	**h–m** ~ I

By the inductive hypothesis, $\Gamma_{\mathbf{j}} \vDash \mathbf{R}$ and $\Gamma_{\mathbf{m}} \vDash \mathbf{\sim R}$. Because the subderivation that derives **R** from **Q** is accessible at position $\mathbf{k + 1}$, every assumption that is open at position **j** is open at position $\mathbf{k + 1}$ except for the assumption **Q** that begins the subderivation. That is, the set of open assumptions $\Gamma_{\mathbf{j}}$ is a subset of $\Gamma_{\mathbf{k+1}} \cup \{\mathbf{Q}\}$. By similar reasoning $\Gamma_{\mathbf{m}}$ must be a subset of $\Gamma_{\mathbf{k+1}} \cup \{\mathbf{Q}\}$. Therefore, by 6.3.2,

$\Gamma_{k+1} \cup \{Q\} \vDash R$ and $\Gamma_{k+1} \cup \{Q\} \vDash \sim R$. From this it follows, by 6.3.4, that $\Gamma_{k+1} \cup \{Q\}$ is truth-functionally inconsistent and then, by 6.3.5, that $\Gamma_{k+1} \vDash \sim Q$.

Case 10: P_{k+1} is justified by Negation Elimination. See Exercise 3.

Case 11: P_{k+1} is justified by Disjunction Elimination:

h	$Q \vee R$	
j	| Q	
	|	
m	| P_{k+1}	
n	| R	
	|	
p	| P_{k+1}	
k + 1	P_{k+1}	h, j–m, n–p ∨E

By the inductive hypothesis, $\Gamma_h \vDash Q \vee R$, $\Gamma_m \vDash P_{k+1}$, and $\Gamma_p \vDash P_{k+1}$. Because the two subderivations are accessible at position $k + 1$, the open assumptions Γ_m form a subset of $\Gamma_{k+1} \cup \{Q\}$ and the open assumptions Γ_p form a subset of $\Gamma_{k+1} \cup \{R\}$. By 6.3.2, then, $\Gamma_{k+1} \cup \{Q\} \vDash P_{k+1}$ and $\Gamma_{k+1} \cup \{R\} \vDash P_{k+1}$. Moreover, because $Q \vee R$ at position h is accessible at position $k + 1$, Γ_h is a subset of Γ_{k+1}. So, because $\Gamma_h \vDash Q \vee R$, it follows, by 6.3.2, that $\Gamma_{k+1} \vDash Q \vee R$. Now consider any truth-value assignment on which every member of Γ_{k+1} is true. Because $\Gamma_{k+1} \vDash Q \vee R$, $Q \vee R$ is also true on this assignment. So either Q or R is true. If Q is true, then every member of $\Gamma_{k+1} \cup \{Q\}$ is true and hence P_{k+1} is true as well because $\Gamma_{k+1} \cup \{Q\} \vDash P_{k+1}$. Similarly, if R is true, then every member of $\Gamma_{k+1} \cup \{R\}$ is true, and hence P_{k+1} is true as well because $\Gamma_{k+1} \cup \{R\} \vDash P_{k+1}$. Either way, it follows that P_{k+1} must be true on any truth-value assignment on which every member of Γ_{k+1} is true. So $\Gamma_{k+1} \vDash P_{k+1}$.

Case 12: P_{k+1} is justified by Biconditional Introduction:

h	| Q	
	|	
j	| R	
m	| R	
	|	
n	| Q	
k + 1	$Q \equiv R$ $(= P_{k+1})$	h–j, m–n ≡I

By the inductive hypothesis, $\Gamma_j \vDash R$ and $\Gamma_n \vDash Q$. Because the two subderivations are accessible at position $k + 1$, Γ_j is a subset of $\Gamma_{k+1} \cup \{Q\}$ and Γ_n is a subset of $\Gamma_{k+1} \cup \{R\}$. By 6.3.2, then, $\Gamma_{k+1} \cup \{Q\} \vDash R$ and $\Gamma_{k+1} \cup \{R\} \vDash Q$. Now consider any truth-value assignment on which every member of

$\Gamma_{\mathbf{k}+1}$ is true. If **R** is also true on that assignment, then so is **Q** because $\Gamma_{\mathbf{k}+1} \cup \{\mathbf{R}\} \vDash \mathbf{Q}$. If **R** is false on that assignment, then **Q** must also be false because if **Q** were true, **R** would have to be true as well since $\Gamma_{\mathbf{k}+1} \cup \{\mathbf{Q}\} \vDash \mathbf{R}$. Either way, **Q** and **R** have the same truth-value, and so $\mathbf{Q} \equiv \mathbf{R}$ is true on every truth-value assignment on which every member of $\Gamma_{\mathbf{k}+1}$ is true. So $\Gamma_{\mathbf{k}+1} \vDash \mathbf{P}_{\mathbf{k}+1}$.

This completes the proof of the inductive step; we have considered every way in which the sentence at position $\mathbf{k} + 1$ of a derivation might be justified and have shown that in each case $\Gamma_{\mathbf{k}+1} \vDash \mathbf{P}_{\mathbf{k}+1}$ if the same is true of all earlier positions in the derivation. We have therefore established the conclusion of the mathematical induction. The sentence at any position in a derivation is truth-functionally entailed by the set of open assumptions in whose scope it lies. And this establishes the soundness metatheorem for *SD*: If $\Gamma \vdash \mathbf{P}$ in *SD*, then $\Gamma \vDash \mathbf{P}$. It follows from Metatheorem 6.3.1 that every argument of *SL* that is valid in *SD* is truth-functionally valid, every sentence of *SL* that is a theorem in *SD* is truth-functionally true, every pair of sentences of *SL* that are equivalent in *SD* are truth-functionally equivalent, and every set of sentences of *SL* that is inconsistent in *SD* is truth-functionally inconsistent. (see Exercise 14 in Exercise set 5.3E).

6.3**E** EXERCISES

1. List all the subsets of each of the following sets:
a. {A ⊃ B, C ⊃ D}
b. {C ∨ ~ D, ~ D ∨ C, C ∨ C}
c. {(B & A) ≡ K}
d. ∅

2. Of which of the following sets is {A ⊃ B, C & D, D ⊃ A} a superset?
a. {A ⊃ B}
b. {D ⊃ A, A ⊃ B}
c. {A ⊃ D, C & D}
d. ∅
e. {C & D, D ⊃ A, A ⊃ B}

***3.** Prove Case 10 of the inductive step in the proof of Metatheorem 6.3.1.

4.
a. Suppose that system *SD** is just like *SD* except that it also contains a new rule of inference:

Negated Biconditional Introduction (~≡I)

| **P**
| ~ **Q**
| ~ (**P** ≡ **Q**)

Prove that system *SD** is a sound system for sentential logic; that is, prove that if $\Gamma \vdash \mathbf{P}$ in *SD** then $\Gamma \models \mathbf{P}$. (You may use Metatheorem 6.3.1.)

*b. Suppose that system *SD** is just like *SD* except that it also contains a new rule of inference:

Backward Conditional Introduction (B⊃I)

		~ **Q**
		~ **P**
P ⊃ **Q**		

Prove that system *SD** is sound for sentential logic.

c. Suppose that system *SD** is just like *SD* except that it also contains a new rule of inference:

Crazy Disjunction Elimination (C∨E)

	P ∨ **Q**	or		**P** ∨ **Q**
▷	**P**		▷	**Q**

Prove that *SD** is not a sound system for sentential logic.

*d. Suppose that system *SD** is just like *SD* except that it also contains a new rule of inference:

Crazy Conditional Introduction (C⊃I)

		~ **P**
		Q
P ⊃ **Q**		

Prove that *SD** is not a sound system for sentential logic.

e. Suppose that the rules of a system *SD** form a subset of the rules of *SD*. Is *SD** a sound system for sentential logic? Explain.

5. Suppose that we changed the characteristic truth-table for '&' to

P	**Q**	**P** & **Q**
T	**T**	**T**
T	**F**	**T**
F	**T**	**F**
F	**F**	**F**

while the characteristic truth-tables for the other sentential connectives remained the same. Would *SD* still be a sound system for sentential logic? Explain.

6. Using Metatheorem 6.3.1 and Exercise 1.e in Section 6.1E, prove that *SD*+ is sound for sentential logic.

We proved in the last section that *SD* is sound, and so every derivation in *SD* is semantically acceptable. This fact alone does not establish that *SD* is an adequate natural deduction system for sentential logic. To establish that *SD* is such a system we must also show that if a set of sentences of *SL* truth-functionally entails a sentence **P** of *SL*, then there is a derivation in *SD* of **P** from that set of sentences. If there is even one case of truth-functional entailment for which a derivation cannot be constructed in *SD*, then *SD* is not adequate to sentential logic. Our final metatheorem assures us that we can derive all that we want to derive in *SD*; it is called the *Completeness Metatheorem:*

Metatheorem 6.4.1: For every sentence **P** of *SL* and every set Γ of sentences of *SL*, if $\Gamma \models \mathbf{P}$ then $\Gamma \vdash \mathbf{P}$ in *SD*.

That is, if a set Γ truth-functionally entails a sentence **P**, then **P** can be derived from Γ in *SD*. It follows from this metatheorem that every argument of *SL* that is truth-functionally valid is valid in *SD*, that every sentence of *SL* that is truth-functionally true is a theorem in *SD* (see Exercise 20 in Section 5.4E), that every pair of sentences of *SL* that are truth-functionally equivalent are equivalent in *SD*, and that every set of sentences of *SL* that is truth-functionally inconsistent is inconsistent in *SD*. A system for which Metatheorem 6.4.1 holds is said to be *complete* for sentential logic.

There are several well-known ways to prove completeness theorems. It may seem that the obvious approach is to show, given that $\Gamma \models \mathbf{P}$, how to construct a derivation of **P** from Γ. There are such proofs and they are termed **constructive** proofs precisely because they not only prove that there is a derivation of **P** from Γ but also provide instructions for constructing such a derivation. We will pursue a different course. The proof we will offer will establish that if $\Gamma \models \mathbf{P}$ then there is a derivation of **P** from Γ but *it will not show us how to construct that derivation.*[4]

The bulk of the work in our proof of 6.4.1 will be in establishing what may at first seem to be an unrelated result:

> **6.4.2:** For any set Γ of sentences of *SL*, if Γ is consistent in *SD* then Γ is truth-functionally consistent.

We begin by proving that Metatheorem 6.4.1 does follow from 6.4.2. To do this, we will use the following result:

> **6.4.3:** If $\Gamma \models \mathbf{P}$ then $\Gamma \cup \{\sim \mathbf{P}\}$ is truth-functionally inconsistent.
>
> **Proof**: Assume $\Gamma \models \mathbf{P}$. Then there is no truth-value assignment on which every member of Γ is true and **P** is false. Therefore, there is no truth-

[4]The proof we will present, while complex, is actually simpler than are constructive proofs. Moreover, our proof of the completeness of *SD* will serve as a model for the completeness proof for *PD* in Chapter 11, for which a constructive proof is not possible. The method that we use to prove completeness is due to Leon Henkin, "The Completeness of the First-Order Functional Calculus," *Journal of Symbolic Logic,* 14 (1949), pp. 159–166.

value assignment on which every member of Γ is true and ~ **P** is also true. So Γ ∪ {~ **P**} is truth-functionally inconsistent.

Here's how metatheorem 6.4.1 follows from 6.4.2:

- Assume that Γ ⊨ **P**.
- Then, by 6.4.3, if follows that Γ ∪ {~ **P**} is truth-functionally inconsistent.
- Since 6.4.2 (which we have not yet proven) tells us that a set of sentences of *SL* that is consistent in *SD* is also truth-functionally consistent, it follows that Γ ∪ {~ **P**} is *not* consistent in *SD*.
- And if Γ ∪ {~ **P**} is *not* consistent in *SD*, it follows from the following result that Γ ⊢ **P**:

  6.4.4: If Γ ∪ {~ **P**} is inconsistent in *SD* then Γ ⊢ **P**.

  Proof: Assume that Γ ∪ {~ **P**} is inconsistent in *SD*. Then by definition, there is a derivation **D** of some sentence **Q** and its negation ~ **Q** from Γ ∪ {~ **P**}. But then we can show that there is also a derivation **D′** of **P** from Γ.

  - The primary assumptions of derivation **D′** are the same as the primary assumptions of derivation **D**, *except that* they do not include ~ **P**.
  - Rather, ~ **P** is assumed *as an auxiliary assumption* in **D′** immediately after the primary assumptions.
  - Once we add this auxiliarly assumption, all of the primary assumptions in **D** are open and accessible assumptions in **D′**.
  - **Q** and ~ **Q** are then derived in **D′** in the same way they were derived in **D.**
  - Then Negation Elimination is used to close the auxiliary assumption ~ **P** and derive **P**.
  - Since **P** falls only within the scope of the primary assumptions in **D′**, which are the members of Γ, this establishes that Γ ⊢ **P**.

All that remains to be done to prove Metatheorem 6.4.1, then, is proving result 6.4.2.

As noted earlier, our proof of 6.4.2 is quite complex. We start by outlining the structure of that proof, proving the simpler parts of it as we go but leaving proof of the more complex section until we have completed the outline. In proving metatheorem 6.4.2 our overall strategy will be to show that if a set Γ of sentences of *SL* is consistent in *SD* then we can construct a truth-value assignment on which every member of Γ is true, thereby showing that Γ is truth-functionally consistent.

We shall construct the truth-value assignment in two steps. First, we shall form a superset of Γ (a set that includes all the members of Γ and possibly

other sentences) that is *maximally consistent in SD*. A maximally consistent set is, intuitively, a consistent set that contains as many sentences as it can without being inconsistent in *SD*:

> A set Γ of sentences of *SL* is **maximally consistent in *SD*** if and only if Γ is consistent in *SD* and, for every sentence **P** of *SL* that is not a member of Γ, $\Gamma \cup \{\mathbf{P}\}$ is inconsistent in *SD*.

If a set is maximally consistent in *SD*, then if we add to the set any sentence that is not already a member, it will be possible to derive some sentence and its negation from the augmented set.

Having constructed a maximally consistent superset of Γ, we then construct a model for the maximally consistent superset, that is, a truth-value assignment on which every member of the maximally consistent superset is true. We construct the model for a superset of Γ that is maximally consistent in *SD*, rather than simply for the original set Γ, because there is a straightforward way to construct models for maximally consistent sets. Of course, because every member of Γ will be in the maximally consistent superset, it will follow that every member of Γ is true on the model that we have constructed and therefore that Γ is truth-functionally consistent.

We now need to fill in the details of our proof. We first need to establish that if Γ is consistent in *SD* then it is a subset of a set Γ that is maximally consistent in *SD*:

> **6.4.5** (The *Maximal Consistency Lemma*): If Γ is a set of sentences of *SL* that is consistent in *SD*, then Γ is a subset of at least one set of sentences that is maximally consistent in *SD*.

In proving the Maximal Consistency Lemma (6.4.5), we shall make use of the fact that the sentences of *SL* can be *enumerated,* that is, placed in a definite order in one-to-one correspondence with the positive integers so that each sentence of *SL* is associated with exactly one positive integer. Here is one method of enumerating the sentences of *SL*. First, we associate with each symbol of *SL* the two-digit numeral occurring to its right:

Symbol	*Numeral*	*Symbol*	*Numeral*
$\sim$	10	A	30
$\vee$	11	B	31
&	12	C	32
$\supset$	13	D	33
$\equiv$	14	E	34
(	15	F	35
)	16	G	36
0	20	H	37
1	21	I	38
⋮	⋮	⋮	⋮
9	29	Z	55

(The ellipses mean that the next two-digit numeral is assigned to the next digit or letter of the alphabet.) Next we associate with each sentence of *SL*, atomic or compound, the integer designated by the numeral that consists of the numerals associated with the symbols in the sentence, in the order in which those symbols occur. For example, the integers associated with the sentences

$$(A \vee C_2) \qquad \sim \sim (A \supset (B \,\&\, \sim C))$$

are, respectively,

153011322216 10101530131531121032161 6

It is obvious that each sentence of *SL* will thus have a distinct integer associated with it. Finally we enumerate all the sentences of *SL* in the order of their associated integers: The first sentence in the enumeration is the sentence with the smallest associated integer, the second sentence is the one with the next smallest associated integer, and so on. In effect, we have imposed an alphabetical order on the sentences of *SL* so that we may freely talk of the first sentence of *SL* (which turns out to be 'A'—because only atomic sentences will have two-digit associated numbers, and the number for 'A' is the smallest of these), the second sentence of *SL* (which turns out to be 'B'), and so on.

Starting with a set Γ of sentences that is consistent in *SD* (as provided for in the antecedent of the Maximal Consistency Lemma), we use our enumeration to construct a superset of Γ that is maximally consistent in *SD*. The construction considers in sequence each sentence in the enumeration we have just described and adds the sentence to the set if and only if the resulting set is consistent in *SD*. In the end the construction will have added as many sentences as can be added to the original set without producing a set that is inconsistent in *SD*. More formally, as the construction goes through the sentences of *SL*, deciding whether to add each sentence, it produces an infinite sequence Γ_1, Γ_2, Γ_3, . . . of sets of sentences of *SL*:

1. Γ_1 is the original set Γ.
2. If $\mathbf{P_i}$ is the **i**th sentence in the enumeration, then $\Gamma_{\mathbf{i}+1}$ is $\Gamma_{\mathbf{i}} \cup \{\mathbf{P_i}\}$ if $\Gamma_{\mathbf{i}} \cup \{\mathbf{P_i}\}$ is consistent in *SD*; otherwise $\Gamma_{\mathbf{i}+1}$ is $\Gamma_{\mathbf{i}}$.

As an example, if $\Gamma_{\mathbf{i}}$ is {~ B, ~ C $\vee$ ~ B} and $\mathbf{P_i}$ is 'A', then $\Gamma_{\mathbf{i}} \cup \{\mathbf{P_i}\}$, which is {~ B, ~ C $\vee$ ~ B, A}, is consistent in *SD*. In this case $\Gamma_{\mathbf{i}+1}$ will be the expanded set $\Gamma_{\mathbf{i}} \cup \{\mathbf{P_i}\}$. If $\Gamma_{\mathbf{i}}$ is {A, ~ B, ~ C $\vee$ ~ B} and $\mathbf{P_i}$ is 'B', then $\Gamma_{\mathbf{i}} \cup \{\mathbf{P_i}\}$, which is {A, ~ B, ~ C $\vee$ ~ B, B}, is inconsistent in *SD* (this is readily verified). In this case $\mathbf{P_i}$ is not added and the set $\Gamma_{\mathbf{i}+1}$ is defined to be $\Gamma_{\mathbf{i}}$, that is, the set {A, ~ B, ~ C $\vee$ ~ B}.

Because we have an infinite sequence of sets, we cannot take the last member of the series as the maximally consistent set desired—because there is no last member! Instead, we form a set Γ^* that is the union of all the sets in the series: Γ^* is defined to contain every sentence that is a member of at least one set in the series and no other sentences. Γ^* is a superset of Γ because it follows

from the definition of Γ^* that every sentence in Γ_1 (as well as Γ_2, Γ_3, . . .) is a member of Γ^*, and Γ_1 is the original set Γ.

Having formed the set Γ^*, it remains to be proved that Γ^* is consistent in *SD* and that it is *maximally* consistent in *SD*. To prove the first claim, we first prove that every set in the sequence Γ_1, Γ_2, Γ_3, . . . is consistent in *SD*. This is easily established by mathematical induction:

Basis clause: The first member of the sequence, Γ_1, is consistent in *SD*.

Proof: Γ_1 is defined to be the original set Γ, which is consistent in *SD*.

Inductive step: If every set in the sequence prior to $\Gamma_{\mathbf{k}+1}$ is consistent in *SD*, then $\Gamma_{\mathbf{k}+1}$ is consistent in *SD*.

Proof: $\Gamma_{\mathbf{k}+1}$ was defined to be $\Gamma_{\mathbf{k}} \cup \{\mathbf{P_k}\}$ if the latter set is consistent in *SD* and to be $\Gamma_{\mathbf{k}}$ otherwise. In the first case $\Gamma_{\mathbf{k}+1}$ is obviously consistent in *SD*. In the second case $\Gamma_{\mathbf{k}+1}$ is consistent because, by the inductive hypothesis, $\Gamma_{\mathbf{k}}$ is consistent in *SD* and $\Gamma_{\mathbf{k}+1}$ just is $\Gamma_{\mathbf{k}}$.

Conclusion: Every member of the series Γ_1, Γ_2, Γ_3, . . . is consistent in *SD*.

We must next show that Γ^* is also consistent in *SD*. We will do so by using the following easily established result:

6.4.6: If a set Γ of sentences of *SL* is inconsistent in *SD*, then some finite subset of Γ is also inconsistent in *SD* (see Exercise 6.4.2).

Assume that Γ^* is inconsistent in *SD*. It then follows from 6.4.6 that there is a finite subset of Γ^*, call it Γ', that is inconsistent in *SD*. Γ' must be nonempty, for the empty set is consistent in *SD* (see Exercise 6.4.3). Moreover, because Γ' is finite, there is a sentence in Γ' that comes after all the other members of Γ' in our enumeration—call this sentence $\mathbf{P_j}$. (That is, any other member of Γ' is $\mathbf{P_h}$ for some $\mathbf{h} < \mathbf{j}$.) Then every member of Γ' is a member of $\Gamma_{\mathbf{j}+1}$, by the way we constructed the series Γ_1, Γ_2, Γ_3, ... (We have constructed the sets in such a way that if a sentence that is the **i**th sentence in our enumeration is a member of *any* set in the sequence—and hence of Γ^*—it must be in the set $\Gamma_{\mathbf{i}+1}$ and every set thereafter.) But if Γ' is inconsistent in *SD*, and every member of Γ' is a member of $\Gamma_{\mathbf{j}+1}$, then $\Gamma_{\mathbf{j}+1}$ is inconsistent in *SD* as well, by 6.4.7:

6.4.7: If Γ is inconsistent in *SD*, then every superset of Γ is inconsistent in *SD*.

Proof: Assume that Γ is inconsistent in *SD*. Then for some sentence **P** there is a derivation of **P** in which all the primary assumptions are members of Γ, and also a derivation of ~ **P** in which all the primary assumptions are members of Γ. The primary assumptions of both derivations are members of every superset of Γ, so **P** and ~ **P** are both derivable from every superset of Γ. Therefore every superset of Γ is inconsistent in *SD*.

But we have already proved by mathematical induction that every set in the infinite sequence is consistent in *SD*. So $\Gamma_{\mathbf{j}+1}$ *cannot* be inconsistent in *SD*, and

our supposition that led to this conclusion is wrong—we may conclude that Γ^* is consistent in *SD*.

Having established that Γ^* is consistent in *SD*, it remains to be shown that it is *maximally* consistent in *SD*. Suppose that Γ^* is not maximally consistent in *SD*. Then there is at least one sentence $\mathbf{P_k}$ of *SL* that is not a member of Γ^* and is such that $\Gamma^* \cup \{\mathbf{P_k}\}$ is consistent in *SD*. We showed, in 6.4.7, that every superset of a set that is inconsistent in *SD* is itself inconsistent, so every subset of a set that is *consistent* in *SD* must itself be consistent in *SD*. In particular, the subset $\Gamma_\mathbf{k} \cup \{\mathbf{P_k}\}$ of $\Gamma^* \cup \{\mathbf{P_k}\}$ must be consistent in *SD*. But then, by step 2 of the construction of the sequence of sets, $\Gamma_{\mathbf{k}+1}$ is defined to be $\Gamma_\mathbf{k} \cup \{\mathbf{P_k}\}$—$\mathbf{P_k}$ is a member of $\Gamma_{\mathbf{k}+1}$. $\mathbf{P_k}$ is therefore a member of Γ^*, contradicting our supposition that it is not a member of Γ^*. Therefore Γ^* must be maximally consistent in *SD*—every sentence that can be consistently added to Γ^* is already a member of Γ^*. This and the result of the previous paragraph establish the Maximal Consistency Lemma (6.4.5); we have shown that, given any set of sentences that is consistent in *SD*, we can construct a superset that is maximally consistent in *SD*.

Finally, we will show that we can construct a truth-value assignment for every set that is maximally consistent in *SD* such that every member of that set is true on that truth-value assignment. From this we will have the following:

> **6.4.8** (the *Consistency Lemma*): Every set of sentences of *SL* that is maximally consistent in *SD* is truth-functionally consistent.

In establishing the Consistency Lemma, we shall appeal to the following important facts about sets that are maximally consistent in *SD*:

> **6.4.9:** If $\Gamma \vdash \mathbf{P}$ and Γ^* is a maximally consistent superset of Γ, then $\mathbf{P}$ is a member of Γ^*.
>
> **Proof:** Assume that $\Gamma \vdash \mathbf{P}$ and let Γ^* be a maximally consistent superset of Γ. By the definition of derivability in *SD*, $\Gamma^* \vdash \mathbf{P}$ as well. Now suppose, contrary to what we wish to prove, that $\mathbf{P}$ is *not* a member of Γ^*. Then, by the definition of maximal consistency, $\Gamma^* \cup \{\mathbf{P}\}$ is inconsistent in *SD*. Therefore by 6.4.4 (if $\Gamma \cup \{\mathbf{P}\}$ is inconsistent in *SD*, then $\Gamma \vdash \sim \mathbf{P}$), it follows that $\Gamma^* \vdash \sim \mathbf{P}$. But then, because both $\mathbf{P}$ and $\sim \mathbf{P}$ are derivable in *SD* from Γ^*, it follows that Γ^* is inconsistent in *SD*. But this is impossible if Γ^* is maximally consistent in *SD*. We conclude that our supposition about $\mathbf{P}$, that it is not a member of Γ^*, is wrong—that is, if $\Gamma^* \vdash \mathbf{P}$, then $\mathbf{P}$ *is* a member of Γ^*.

In what follows, we will use the standard notation

$$\mathbf{P} \in \Gamma$$

to mean

$\mathbf{P}$ is a member of Γ

and the standard notation

$$\mathbf{P} \notin \Gamma$$

to mean

> **P** is not a member of Γ.

The next result concerns the composition of the membership of any set that is maximally consistent in *SD*:

> **6.4.11:** If Γ* is maximally consistent in *SD* and **P** and **Q** are sentences of *SL*, then:
>
> a. ~ **P** ∈ Γ* if and only if **P** ∉ Γ*.
> b. **P** & **Q** ∈ Γ* if and only if both **P** ∈ Γ* and **Q** ∈ Γ*.
> c. **P** ∨ **Q** ∈ Γ* if and only if either **P** ∈ Γ* or **Q** ∈ Γ*.
> d. **P** ⊃ **Q** ∈ Γ* if and only if either **P** ∉ Γ* or **Q** ∈ Γ*.
> e. **P** ≡ **Q** ∈ Γ* if and only if either **P** ∈ Γ* and **Q** ∈ Γ*, or **P** ∉ Γ* and **Q** ∉ Γ*.
>
> **Proof of (a):** Assume that ~ **P** ∈ Γ*. Then **P** ∉ Γ* for, if it were a member, then Γ* would have a finite subset that is inconsistent in *SD*, namely, {**P**, ~ **P**}, and according to 6.4.7 this is impossible if Γ* is consistent in *SD*. Now assume that **P** ∉ Γ*. Then, by the definition of maximal consistency in *SD*, Γ* ∪ {**P**} is inconsistent in *SD*. So, by reasoning similar to that used in proving 6.4.9, some finite subset Γ′ of Γ* is such that Γ′ ∪ {**P**} is inconsistent in *SD*, and therefore such that Γ′ ∪ {~ ~ **P**} is inconsistent in *SD*. So Γ′ ⊢ ~ **P**, by 6.4.4. It follows, by 6.4.9, that ~ **P** ∈ Γ*.
>
> **Proof of (b):** Assume that **P** & **Q** ∈ Γ*. Then {**P** & **Q**} is a subset of Γ*. Because {**P** & **Q**} ⊢ **P** and {**P** & **Q**} ⊢ **Q** (both by Conjunction Elimination), it follows, by 6.4.9, that **P** ∈ Γ* and **Q** ∈ Γ*. Now suppose that **P** ∈ Γ* and **Q** ∈ Γ*. Then {**P**, **Q**} is a subset of Γ* and, because {**P**, **Q**} ⊢ **P** & **Q** (by Conjunction Introduction), it follows, by 6.4.9, that **P** & **Q** ∈ Γ*.
>
> **Proof of (c):** See Exercise 6.4.5.
>
> **Proof of (d):** Assume that **P** ⊃ **Q** ∈ Γ*. If **P** ∉ Γ*, then it follows trivially that either **P** ∉ Γ* or **Q** ∈ Γ*. If **P** ∈ Γ*, then {**P**, **P** ⊃ **Q**} is a subset of Γ*. Because {**P**, **P** ⊃ **Q**} ⊢ **Q** (by Conditional Elimination), it follows, by 6.4.9, that **Q** ∈ Γ*. So, if **P** ⊃ **Q** ∈ Γ*, then either **P** ∉ Γ* or **Q** ∈ Γ*. Now assume that either **P** ∉ Γ* or **Q** ∈ Γ*. In the former case, by (a), ~ **P** ∈ Γ*. So either {~ **P**} is a subset of Γ* or {**Q**} is a subset of Γ*. **P** ⊃ **Q** is derivable from both subsets:

1	~ **P**	Assumption
2	**P**	A / ~ E
3	~ **Q**	A / ~ E
4	**P**	2 R
5	~ **P**	1 R
6	**Q**	3–5 ~ E
7	**P** ⊃ **Q**	2–6 ⊃I

1	**Q**	Assumption
2	**P**	A / ⊃I
3	**Q**	1 R
4	**P** ⊃ **Q**	2–3 ⊃I

Either way, there is a finite subset of Γ* from which $\mathbf{P} \supset \mathbf{Q}$ is derivable; so, by 6.4.9, it follows that $\mathbf{P} \supset \mathbf{Q} \in \Gamma^*$.

Proof of (e): See Exercise 6.4.5.

Turning now to the Consistency Lemma (6.4.8), let Γ be a set of sentences that is maximally consistent in *SD*. We said earlier that it is easy to construct a truth-value assignment on which every member of a maximally consistent set is true, and it is; we need only consider the atomic sentences in the set. Let **A*** be the truth-value assignment that assigns the truth-value **T** to every atomic sentence of *SL* that is a member of Γ* and assigns the truth-value **F** to every other atomic sentence of *SL*. We shall prove by mathematical induction that each sentence of *SL* is true on the truth-value assignment **A*** if and only if it is a member of Γ*—from which it follows that every member of Γ* is true on **A***, thus establishing truth-functional consistency. The induction will be based on the number of occurrences of connectives in the sentences of *SL*:

Basis clause: Each atomic sentence of *SL* is true on **A*** if and only if it is a member of Γ*.

Inductive step: If every sentence of *SL* with **k** or fewer occurrences of connectives is such that it is true on **A*** if and only if it is a member of Γ*, then every sentence of *SL* with $\mathbf{k} + 1$ occurrences of connectives is such that it is true on **A*** if and only if it is a member of Γ*.

Conclusion: Every sentence of *SL* is such that it is true on **A*** if and only if it is a member of Γ*.

The basis clause is obviously true; we defined **A*** to be an assignment that assigns **T** to all and only the atomic sentences of *SL* that are members of Γ*. To prove the inductive step, we will assume that the inductive hypothesis holds for an arbitrary integer **k**: that each sentence containing **k** or fewer occurrences of connectives is true on **A*** if and only if it is a member of Γ*. We must now show that the same holds true for every sentence **P** containing $\mathbf{k} + 1$ occurrences of connectives. We consider five cases, reflecting the five forms that a compound sentence of *SL* might have.

Case 1: P has the form $\sim \mathbf{Q}$. If $\sim \mathbf{Q}$ is true on **A***, then **Q** is false on **A***. Because **Q** contains fewer than $\mathbf{k} + 1$ occurrences of connectives, it follows by the inductive hypothesis that $\mathbf{Q} \notin \Gamma^*$. Therefore, by 6.4.11(a), $\sim \mathbf{Q} \in \Gamma^*$. If $\sim \mathbf{Q}$ is false on **A***, then **Q** is true on **A***. It follows by the inductive hypothesis that $\mathbf{Q} \in \Gamma^*$. Therefore, by 6.4.11(a), $\sim \mathbf{Q} \notin \Gamma^*$.

Case 2: P has the form $\mathbf{Q} \,\&\, \mathbf{R}$. If $\mathbf{Q} \,\&\, \mathbf{R}$ is true on **A***, then both **Q** and **R** are true on **A***. Because **Q** and **R** each contain fewer than $\mathbf{k} + 1$ occurrences of connectives, it follows by the inductive hypothesis that $\mathbf{Q} \in \Gamma^*$ and $\mathbf{R} \in \Gamma^*$. Therefore, by 6.4.11(b), $\mathbf{Q} \,\&\, \mathbf{R} \in \Gamma^*$. If $\mathbf{Q} \,\&\, \mathbf{R}$ is false on **A***, then either **Q** is false on **A*** or **R** is false

on **A***. Therefore, by the inductive hypothesis, either $\mathbf{Q} \notin \Gamma^*$ or $\mathbf{R} \notin \Gamma^*$ and so, by 6.4.11(b), $\mathbf{Q} \,\&\, \mathbf{R} \notin \Gamma^*$.

Case 3: **P** has the form $\mathbf{Q} \vee \mathbf{R}$. See Exercise 6.4.6.

Case 4: **P** has the form $\mathbf{Q} \supset \mathbf{R}$. If $\mathbf{Q} \supset \mathbf{R}$ is true on **A***, then either **Q** is false on **A*** or **R** is true on **A***. Because **Q** and **R** each contain fewer than $\mathbf{k} + 1$ occurrences of connectives, it follows from the inductive hypothesis that either $\mathbf{Q} \notin \Gamma^*$ or $\mathbf{R} \in \Gamma^*$. By 6.4.11(d), then, $\mathbf{Q} \supset \mathbf{R} \in \Gamma^*$. If $\mathbf{Q} \supset \mathbf{R}$ is false on **A***, then **Q** is true on **A*** and **R** is false on **A***. By the inductive hypothesis, then, $\mathbf{Q} \in \Gamma^*$ and $\mathbf{R} \notin \Gamma^*$. And by 6.4.11(d), it follows that $\mathbf{Q} \supset \mathbf{R} \notin \Gamma^*$.

Case 5: See Exercise 6.4.6.

This completes the proof of the inductive step. Hence we may conclude that each sentence of *SL* is a member of Γ^* if and only if it is true on **A***. So every member of a set Γ^* that is maximally consistent in *SD* is true on **A***, and the set Γ^* is therefore truth-functionally consistent. This establishes the Consistency Lemma (6.4.8).

We now know that result 6.4.2, which we repeat here, is true:

6.4.2: For any set Γ of sentences of *SL*, if Γ is consistent in *SD* then Γ is truth-functionally consistent.

Because every set of sentences Γ that is consistent in *SD* is a subset of a set of sentences that is maximally consistent in *SD* (the Maximal Consistency Lemma (6.4.5)), and because every set of sentences that is maximally consistent in *SD* is truth-functionally consistent (the Consistency Lemma (6.4.8)), it follows that every set of sentences that is consistent in *SD* is a subset of a truth-functionally consistent set and is therefore itself truth-functionally consistent.

And Metatheorem 6.4.1 follows from 6.4.2:

If $\Gamma \vDash \mathbf{P}$, then $\Gamma \vdash \mathbf{P}$.

For if $\Gamma \vDash \mathbf{P}$, then, by 6.4.3, $\Gamma \cup \{\sim \mathbf{P}\}$ is truth-functionally inconsistent. Then, by result 6.4.2, $\Gamma \cup \{\sim \mathbf{P}\}$ is inconsistent in *SD*. And if $\Gamma \cup \{\sim \mathbf{P}\}$ is inconsistent in *SD*, then, by 6.4.4, $\Gamma \vdash \mathbf{P}$ in *SD*. So *SD* is complete for sentential logic—for every truth-functional entailment, at least one corresponding derivation can be constructed in *SD*. This, together with the proof of the Soundness Metatheorem in Section 6.3, shows that *SD* is an adequate system for sentential logic.

We conclude by noting that another important result, the *Compactness Theorem* for sentential logic, follows from Result 6.4.2 and Metatheorem 6.3.1:

Metatheorem 6.4.12: A set Γ of sentences of *SL* is truth-functionally consistent if and only if every finite subset of Γ is truth-functionally consistent.

And, as a consequence, a set of sentences of *SL* is truth-functionally inconsistent if and only if at least one finite subset of Γ is inconsistent.

6.4**E** EXERCISES

1. Prove 6.4.4 and 6.4.10.

2. Prove 6.4.6.

***3.** Prove that the empty set is consistent in *SD*.

4. Using Metatheorem 6.4.1, prove that *SD*+ is complete for sentential logic.

***5.** Prove that every set that is maximally consistent in *SD* has the following properties:
a. $\mathbf{P} \vee \mathbf{Q} \in \Gamma^*$ if and only if either $\mathbf{P} \in \Gamma^*$ or $\mathbf{Q} \in \Gamma^*$.
b. $\mathbf{P} \equiv \mathbf{Q} \in \Gamma^*$ if and only if either $\mathbf{P} \in \Gamma^*$ and $\mathbf{Q} \in \Gamma^*$, or $\mathbf{P} \notin \Gamma^*$ and $\mathbf{Q} \notin \Gamma^*$. (These are clauses c and e of 6.4.11.)

***6.** Establish Cases 3 and 5 of the inductive step in the proof of the Consistency Lemma 6.4.8.

7.a. Suppose that *SD** is like *SD* except that it lacks Reiteration. Show that *SD** is complete for sentential logic.
*b. Suppose that *SD** is like *SD* except that it lacks Negation Introduction. Show that *SD** is complete for sentential logic.

8. Suppose that *SD** is like *SD* except that it lacks Conjunction Elimination. Show where our completeness proof for *SD* will fail as a completeness proof for *SD**.

9. Using Result 6.4.2 and Metatheorem 6.3.1, prove Metatheorem 6.4.12.

Chapter 7

PREDICATE LOGIC: SYNTAX AND SYMBOLIZATION

Section 7.1 discusses the aspects of English syntax that cannot be captured by *SL* but are mirrored in *PL*. In Section 7.2 we present the formal syntax of *PL*. In Section 7.3 we symbolize a wide range of English sentences in *PL*. In Section 7.4 we explore a variety of issues bearing on how we symbolize sentences in *PL*. In Section 7.5 we present *PLE*, an extension of *PL* that includes identity and functors.

7.1 PREDICATES, SINGULAR TERMS, AND QUANTITY EXPRESSIONS OF ENGLISH

As we noted in Chapter 2, the syntax of English (and every natural language) is much more complicated than is the syntax of *SL*. *SL* is a language for sentential logic and uses sentence letters to symbolize whole sentences of English. Consequently, subsentential components of English sentences have no counterparts in *SL*. Among the subsentential components of English sentences are

singular terms, predicates, and quantity expressions. Consider the following fairly simple argument:

> None of David's friends supports Republicans. Sarah supports Breitlow and Breitlow is a Republican. So Sarah is no friend of David's.

This is a valid argument. Although we can symbolize this argument in *SL* using the following symbolization key:

N: None of David's friends supports Republicans.
S: Sarah supports Breitlow
B: Breitlow is a Republican
F: Sarah is a friend of David's

the resulting argument is *not* valid in *SL*:

N
S & B
~ F

The problem is that the atomic sentences of *SL* are logically independent of each other—the truth-value of an atomic sentence on a truth-value assignment has no bearing on the truth-value of other atomic sentences on that truth-value assignment. But the English sentences we are symbolizing *are not* logically independent of each other. If the sentences symbolized by 'S' and 'B' are true, then it is also true that Sarah supports a Republican. And if that is so, and the first premise of our English argument is true, it follows that Sarah is not a friend of David's. That is, it is the interconnections between the predicates 'supports Republicans', 'supports Breitlow', 'is a Republican', and 'is a friend of David's', the singular terms 'Sarah', 'David', and 'Breitlow', and the quantity expression 'None' that make the English language argument valid.

The sentence

Each citizen will vote or will not vote

provides another illustration of the limitations of *SL*. This sentence is not presenting two alternatives, that every citizen votes or that no citizen votes. Rather it is expressing the logical truth that the predicate 'will vote or will not vote' is true of each citizen. This generalization about citizens applies to each citizen individually, not to citizens as a collective group. For example, it applies to Cynthia (presuming she is a citizen) and says of Cynthia that she will either vote or not vote. This claim about Cynthia, or any other specified citizen, can readily be symbolized as a truth-functional truth of *SL*. Where 'C' abbreviates 'Cynthia will vote', 'C ∨ ~ C' says of Cynthia what the general claim says of each citizen. But there is, barring heroic measures, no symbolization of the general claim in *SL* that is truth-functionally true.[1]

[1]Since there are presumably only finitely many citizens, we could construct a very long iterated conjunction with as many conjuncts of the sort 'C ∨ ~ C' as there are citizens. But even such heroic measures fail when the items about which we wish to talk (for example, the positive integers) constitute an infinite, and not just an exceedingly large, set.

What we need is a symbolic language whose syntax does mirror the use of singular terms, predicates, and quantity expressions in English. *PL* is such a language. However, before introducing *PL* we will explore how singular terms, predicates, and quantity expressions function in English. A **singular term** is any word or phrase that designates or purports to designate (or denote or refer to) some one thing. Singular terms are of three sorts: proper names, definite descriptions, and pronouns used in place of proper names or definite descriptions, that is, pronouns that make pronominal cross-reference to proper names or definite descriptions. Examples of proper names include 'George Washington', 'Marie Curie', 'Sir Arthur Conon Doyle', 'Rhoda', and 'Henry'. Generally speaking, proper names are attached to the things they name by simple convention. On the other hand, definite descriptions—for example, 'the discoverer of radium', 'the person Henry is talking to', 'Mary's best friend', and 'James' only brother'—pick out or purport to pick out a thing by providing a unique description of that thing.[2] A definite description is a description that, by its grammatical structure, describes or specifies at most one thing. Thus 'James' only brother' is a definite description whereas 'James' brother' is not—the latter could accurately apply to more than one person because James may have many brothers, whereas the former can apply to at most one. Pronouns that bear pronominal cross-reference to singular terms refer to the things those singular terms designate.

The sentence 'If Sue has read Darwin's works, she's no creationist' contains three singular terms, the proper name 'Sue', the definite description 'Darwin's works', and the pronoun 'she'. Each of these singular terms does refer to something. 'Sue' refers to Sue because, by convention, it is her name. 'Darwin's works' refers to the works of Darwin because it is a definite description of those works. And the pronoun 'she' refers to Sue because in this sentence 'she' is going proxy for 'Sue' (it bears pronominal cross-reference to 'Sue' and hence refers to the same entity as does 'Sue').

Predicates, such as 'supports Republicans', can be thought of as incomplete sentences that contain gaps or holes such that when those gaps are filled with singular terms the result is a complete sentence. However, writing predicates as we have just done does not visibly display the gap into which a singular term can be placed, and it is not suitable for displaying predicates containing more than one gap or hole. We could display predicates by indicating the gaps with underscores, as in

___ supports Republicans
___ is located between ___ and ___

[2]As these examples illustrate, definite descriptions can themselves contain singular terms. But we are here concerned only with singular terms that do not occur as constituents of other singular terms. For example, we here take 'The Roman general who defeated Pompey invaded both Gaul and Germany' to contain just three singular terms: 'The Roman general who defeated Pompey', 'Gaul', and 'Germany'. In Section 7.5 we shall introduce techniques that allow us to recognize and symbolize singular terms that are themselves constituents of singular terms—including 'Pompey' as it occurs in 'The Roman general who defeated Pompey'.

But for reasons that will emerge, it is useful to use the lowercase letters 'w', 'x', 'y', and 'z' (called 'variables' for reasons to be subsequently explained) to mark the places in predicates where singular terms can be placed. Using this convention we can specify the two predicates displayed above as 'x supports Republicans' and 'x is located between y and z'. A predicate with one gap is a one-place predicate, a predicate containing two gaps is a two-place predicate, and in general a predicate containing **n** gaps is an **n**-place predicate.

One way of generating a predicate is to start with a complete sentence of English containing one or more singular terms and delete one or more of those terms. And one way of generating a sentence from a predicate is to fill all the holes that are marked by variables with singular terms.

Because the gaps in predicates that are marked by variables can be filled with referring expressions—proper names, definite descriptions, and some uses of pronouns—we will say that these gaps are 'referential positions' and that expressions occurring in these positions, including variables, '**occur in referential position**'. But not all expressions that occur in referential positions do refer. Consider

If you play with fire you are likely to get burned.

In most contexts this sentence is used to comment about what is likely to happen to one, anyone, who plays with fire. Hence it would be a mistake to ask whom 'you', in either occurrence in the sentence, refers to. The sentence is a warning to all persons but does not refer to any particular person. Similarly, though 'Nobody' is a pronoun and does occur in referential position in

Nobody knows where Tom is

it does not make reference to anyone. There is no one whose name is 'nobody', nor does 'nobody' describe someone. We shall shortly discuss at length the use of pronouns that occur in referential position but do not in fact refer to anyone or anything.

It is not the case that all the singular terms of natural languages do refer or denote some one thing. For example, the only singular term in

Sherlock Holmes was a great detective

does not refer to a nineteenth-century English detective named 'Sherlock Holmes' who lived at 221B Baker Street, because there was no such detective. There are also definite descriptions that occur in referential positions but do not refer. Two examples are 'the present prime minister of the United States' and 'the largest prime number'. There is no prime minister of the United States and there is no largest prime number. This is a matter of some importance because by stipulation all of the individual constants of *PL*, the analogues of proper names and definite descriptions of English, do refer. Various strategies have been advanced for dealing with singular terms that do not refer, and we will explore one of them later in this chapter. But for the present we stipulate that all the singular terms we use in examples and exercises should be taken to refer.

Of course, what a singular term refers to is often context dependent. In its most familiar use 'George Washington' refers to the first president of the United States. But the U.S. Navy has an aircraft carrier named after the first president and so there are contexts in which 'George Washington' refers to a ship, not a man. Similarly, at a cocktail party where there is only one Henry, 'the person Henry is talking to' may refer to different persons at different times. Hereafter, when we use a sentence of English as an example or in an exercise set, we are assuming that sentence is being used in a context such that it is clear who or what the singular terms in that sentence refer to. We also note that when we are working with a group of sentences, the context that is assumed must be the same for all the sentences in the group. That is, we assume that a singular term that occurs several times in the piece of English discourse under discussion designates the same thing in each of its occurrences.

Predicates may contain multiple singular terms; in generating a predicate from a sentence containing multiple singular terms we may, but need not, delete all the singular terms. For example, 'New York City is north of Philadelphia' contains two singular terms, 'New York City' and 'Philadelphia', and we can obtain three distinct predicates by deleting one or both of these singular terms:

x is north of Philadelphia
New York City is north of x
x is north of y

As far as grammar is concerned, any singular term can be used to replace a variable in a predicate. Hence among the sentences we can generate from the two-place predicate 'x is north of y' and the singular terms 'Minneapolis', 'Chicago', and '3' are

Minneapolis is north of Chicago.
Chicago is north of Minneapolis.
Chicago is north of Chicago.
Chicago is north of 3.

The semantics we will adopt will assign a truth-value to each of these sentences.[3]

Given a stock of predicates, singular terms, and the sentential connectives '. . . and . . .', '. . . or . . .', 'if . . . then . . .', '. . . if and only if . . .', and 'it is not the case that . . .', we can generate a wide variety of sentences of English. For example, from these sentential connectives, the singular terms 'Henry', 'Sue', 'Rita', and 'Michael', and the predicates

x is easygoing
x likes y
x is taller than y

[3]It may be suggested that the fourth sentence is neither true nor false, as it "makes no sense". Numbers do not have location, so 3 is not located anywhere. But the semantics we will provide in Chapter 8 does allow for such sentences and counts them as false. Precisely because numbers do not have location it is false that any given number is spatially related to anything.

we can generate the following sentences

Michael is easygoing.
Sue is easygoing.
Michael is taller than Sue and Sue is taller than Henry.
Sue likes Henry and Michael likes Rita.
If Rita likes Henry, then Rita is taller than Henry.
Michael is easygoing if and only if it is not the case that Rita is easygoing.

If we allow the use of quantity expressions as well as singular terms to generate sentences from predicates, that is expressions such as 'everything', 'something', 'nothing', 'everyone', someone', and 'no one', we can also generate the following sentences from these same predicates:

Everyone is easygoing.
No one is easygoing.
Someone is easygoing.
Michael likes everyone.
Someone likes Sue.
No one is taller than her- or himself.

Note that the quantity expressions we used to generate these sentences, 'everyone', 'no one', and 'someone', all occur in referential positions, in positions where singular terms can occur. But these and other quantity expressions *are not* singular terms. 'No one' obviously does not refer to anyone. And neither does 'someone'. Consider 'Someone will win tonight's lottery'. We can all agree that this is true—there will be a winner. Suppose the winner turns out to be Henry Jacobson. It is not the case that when we asserted, in the morning, that someone would win the lottery we were referring to Henry Jacobson. All 'Someone will win tonight's lottery' asserts is that there is a person, identity presumably unknown, who will win tonight's lottery.

Similarly, 'Everyone' in 'Everyone is easygoing' does not refer to the totality of people or the set of all people, for it is individuals, not collections or sets of individuals, that are claimed to be easygoing. Rather, the force of 'Everyone is easygoing' is just that 'is easygoing' can be truly predicated of each and every individual.

7.1 EXERCISES

1. Identify the singular terms in the following sentences, and then specify all the predicates that can be obtained from each sentence by deleting one or more singular terms.

a. The president is a Democrat.
*b. The speaker of the house is a Republican
c. Sarah attends Smith College.

*d. Bob flunked out of U Mass.
e. Charles and Rita are siblings.
*f. 2 is greater than 1 and less than 4.

7.2 THE FORMAL SYNTAX OF *PL*

The language *PL* is far more powerful than the language *SL* because it includes constituents whose functions largely mirror the functions of **n**-place predicates, singular terms, and quantity terms ('every', 'some', 'no', . . .) in English.[4] In this section we present the formal syntax of *PL*, which is somewhat complicated (though not as complicated as the syntax of English).

The vocabulary of *PL* consists of:

Sentence Letters: The capital Roman letters 'A' through 'Z', with or without positive-integer subscripts	A, B, C, . . . , Z, A_1, B_1, C_1, . . . , Z_1, . . .
Predicates: The capital Roman letters 'A' through 'Z', with or without positive-integer subscripts, followed by one or more primes	A′, B′, C′, . . . , Z′, A_1', B_1', C_1', . . . , Z_1', . . .
Individual terms:	a, b, c, . . . , v,
Individual constants: The lowercase Roman letters 'a' through 'v', with or without positive-integer subscripts	a_1, b_1, c_1, . . . , v_1, . . .
Individual variables: The lowercase Roman letters 'w' through 'z', with or without positive-integer subscripts	w, x, y, z, w_1, x_1, y_1, z_1, . . .
Truth-functional connectives:	~, &, ∨, ⊃, ≡
Quantifier symbols:	∀, ∃
Punctuation marks:	( )

The sentence letters of *PL* are just the sentence letters of *SL*. This makes every sentence of *SL* a sentence of *PL*. Officially, that a predicate of *PL* is an **n**-place predicate is indicated by the presence of **n** primes. So 'A′' is a one-place predicate and 'B″' is a two-place predicate of *PL*. But we also adopt the informal convention of allowing the omission of primes when the context makes it clear whether the predicate in question is a 1-place, 2-place, 3-place, . . . predicate, as when we write an **n**-place predicate with **n** distinct variables

[4] *PL* does not mirror all the subsentential relations present in English and other natural languages. Consequently, there are, as one might expect, English language arguments that are deductively valid but whose symbolizations in *PL* are not valid, English sentences that are logically true but whose symbolizations in *PL* do not reflect this, and so on. To deal with natural language discourse that cannot be adequately represented in *PL*, even more powerful formal systems are available—for example, tense logic and modal logic. A discussion of these systems is beyond the scope of this text.

following the predicate. So we will often write 'Ax' (or 'Ay' or 'Az' . . .) rather than 'A′', 'Bxy' rather than 'B″', and so on.

An **expression** of *PL* is a sequence of not necessarily distinct elements of the vocabulary of *PL*. All of the following are expressions of *PL*:

Lab
(∀∃xy
(∀w)(∃y)Fwy
((a ⊃ B)

In each case every character in the expression is an element of the vocabulary of *PL*. But the following are not expressions of *PL*:

{AbcB
(A ⊃ π)
(∀x/W

Each of these expressions contains a character that is not part of the vocabulary of *PL*. These are, respectively, '{', 'π' (the Greek letter pi), and '/'.

In what follows we will use the boldface capital letters '**P**', '**Q**', '**R**', and '**S**' as metavariables ranging over expressions of *PL*. We will also use boldface '**a**' as a metavariable ranging over individual constants of *PL* and boldface '**x**' as a metavariable ranging over individual variables of *PL*.

> *Quantifier of PL:* An expression of *PL* of the form (∀**x**) or (∃**x**). An expression of the first form is a **universal** quantifier, and one of the second form is an **existential** quantifier.

We will say that a quantifier *contains* a variable. Thus '(∀y)' and '(∃y)' both contain the variable 'y' (and are 'y-quantifiers'); '(∀z)' and '(∃z)' both contain the variable 'z' (and are 'z-quantifiers').

> *Atomic formulas of PL*: Every expression of *PL* that is either a sentence letter of *PL* or an **n**-place predicate of *PL* followed by **n** individual terms of *PL*.

We are now able to recursively define 'formula of *PL*':

1. Every atomic formula of *PL* is a formula of *PL*.
2. If **P** is a formula of *PL*, so is ~ **P**.
3. If **P** and **Q** are formulas of *PL*, so are (**P** & **Q**), (**P** ∨ **Q**), (**P** ⊃ **Q**), and (**P** ≡ **Q**).
4. If **P** is a formula of *PL* that contains at least one occurrence of **x** and no **x**-quantifier, then (∀**x**)**P** and (∃**x**)**P** are both formulas of *PL*.

5. Nothing is a formula of *PL* unless it can be formed by repeated applications of clauses 1–4.

Lastly, we specify the **logical operators** of *PL*:

> An expression of *PL* that is either a quantifier or a truth-functional connective is a *logical operator* of *PL*.

It will emerge that not every formula of *PL* is a sentence of *PL*. However, before we can specify which formulas of *PL* are sentences of *PL* we need to define the terms 'subformula' and 'main logical operator':

1. Every formula is a subformula of itself.
2. If **P** is an atomic formula of *PL*, then **P** contains no logical operator, and hence no main logical operator, and **P** has no immediate subformula.
3. If **P** is a formula of *PL* of the form ~ **Q**, then the tilde ('~') that precedes **Q** is the main logical operator of **P**, and **Q** is the immediate subformula of **P**.
4. If **P** is a formula of *PL* of the form (**Q** & **R**), (**Q** ∨ **R**), (**Q** ⊃ **R**), or (**Q** ≡ **R**), then the binary connective between **Q** and **R** is the main logical operator of **P**, and **Q** and **R** are the immediate subformulas of **P**.
5. If **P** is a formula of *PL* of the form (∀**x**)**Q** or of the form (∃**x**)**Q**, then the quantifier that occurs before **Q** is the main logical operator of **P**, and **Q** is the immediate subformula of **P**.

The *subformulas* of a formula **P** of *PL* are

- **P** itself,
- The immediate subformulas of **P**,
- The subformulas of **P**'s immediate subformulas.

We can classify formulas of *PL* (and later sentences) by their main logical operator. Atomic formulas have no main logical operator. Quantified formulas have a quantifier as their main logical operator. Formulas whose main logical operator is a sentential connective are truth-functional formulas. Below we display several formulas and all of their subformulas. Remember that every formula is a subformula of itself.

Formula	Main Logical Operator	Formula Type
1. Rabz	None	Atomic
2. ~ (Rabz & Hxy)	~	Truth-functional
(Rabz & Hxy)	&	Truth-functional
Rabz	None	Atomic
Hxy	None	Atomic

3. (Hab ≡ (∀z)(Fz ⊃ Gza))	≡	Truth-functional
Hab	None	Atomic
(∀z)(Fz ⊃ Gza)	(∀x)	Quantified
(Fz ⊃ Gza)	⊃	Truth-functional
Fz	None	Atomic
Gza	None	Atomic
4. (∀y)(Hay ∨ (Fy ⊃ Gya))	(∀y)	Quantified
(Hay ∨ (Fy ⊃ Gya))	∨	Truth-functional
Hay	None	Atomic
(Fy ⊃ Gya)	⊃	Truth-functional
Fy	None	Atomic
Gya	None	Atomic

Quantifiers interpret the variables that fall within their **scope**.

> *Scope of a quantifier:* The scope of a quantifier in a formula **P** of *PL* is the quantifier itself and the subformula **Q** that immediately follows the quantifier.

In other words, the scope of a quantifier is all of the formula of which the quantifier is the main logical operator, including the quantifier itself. Some examples will be helpful here. In the formula '(∃y)(Fyz & Gzy)' the subformula that immediately follows the quantifier '(∃y)' is '(Fyz & Gzy)' and accordingly the scope of that quantifier is all of '(∃y)(Fyz & Gzy)', and all of the variables in that formula, including the 'y' following '∃', fall within the scope of that quantifier. But in the formula 'Hx ⊃ (∀y)Fxy' the formula immediately following the quantifier '(∀y)' is 'Fxy' and the scope of that quantifier is therefore all of '(∀y)Fxy'. The first occurrence of 'x' (in 'Hx') does not fall within the scope of '(∀y)'. Similarly in '(∃w)(Gwa ⊃ Fa) ≡ Hw' the scope of '(∃w)' does not include the whole formula, for the formula that immediately follows that quantifier is 'Gwa ⊃ Fa'. Hence the first and second occurrences of 'w' in '((∃w)(Gwa ⊃ Fa) ≡ Hw)' fall within the scope of '(∃w)' but the third, in 'Hw', does not.

The final concepts we need to introduce before we define 'sentence of *PL*' are those of **free** and **bound** variables.

> *Bound variable:* An occurrence of a variable **x** in a formula **P** of *PL* that is within the scope of an **x**-quantifier.
> *Free variable:* An occurrence of a variable **x** in a formula **P** of *PL* that is not bound.

At long last we are ready to formally introduce the notion of a **sentence of *PL:***

> *Sentence of PL:* A formula **P** of *PL* is a sentence of *PL* if and only if no occurrence of a variable in **P** is free.

The formula '(Hx ⊃ (∀y)Fxy)' is not a sentence of *PL* because it contains a free variable. In fact, both occurrences of 'x' in this formula are free.

The first occurrence of 'x' does not fall within the scope of any quantifier and is therefore free, and the second occurrence of 'x', while falling within the scope of a quantifier, does not fall within the scope of an x-quantifier and is therefore free. The formula '$(\forall z)Gz \supset \sim Hz$' is not a sentence of *PL* because the third occurrence of 'z' does not fall within the scope of a z-quantifier. The scope of '$(\forall z)$' is limited to the subformula of which it is the main logical operator—that is, to '$(\forall z)Gz$'.

Earlier we considered the following four formulas of *PL*:

Rabz

$\sim$ (Rabz & Hxy)

(Hab $\equiv$ $(\forall z)$(Fz $\supset$ Ga))

$(\forall y)$(Hay $\vee$ (Fy $\supset$ Gya))

The first formula is not a sentence of *PL* because it contains 'z' as a free variable. We can construct a sentence from this formula by prefacing it with a z-quantifier; both '$(\exists z)$Rabz' and '$(\forall z)$Rabz' are sentences of *PL*. The second formula in our list is not a sentence of *PL* because 'z', 'x', and 'y' all occur free in that formula. This formula can be converted to a sentence by prefacing it with three distinct quantifiers, as in '$(\exists z)(\exists x)(\exists y) \sim$ (Rabz & Hxy)'. The third formula in our list is a sentence of *PL*. The only variable in '(Hab $\equiv$ $(\forall z)$(Fz $\supset$ Ga))' is 'z' and its two occurrences both fall within the scope of the quantifier '$(\forall z)$'. The fourth formula is also a sentence of *PL*. The formula of which '$(\forall y)$' is the main logical operator is the entire formula, and hence all four occurrences of 'y' fall within the scope of that quantifier.

There are formulas of *PL* that cannot be transformed into sentences of *PL* by adding quantifiers within whose scope the entire original formula falls. Consider '(Fy $\supset$ $(\exists y)$Gy)'. The first occurrence of 'y' in this formula is free as it does not fall within the scope of any quantifier. The result of attaching a universal quantifier to this entire formula is '$(\forall y)$(Fy $\supset$ $(\exists y)$Gy)'. But this expression is neither a formula nor a sentence of *SL*. The only way quantifiers become attached to formulas is in accordance with the fourth clause of the recursive definition of 'formula of *PL*', which is

4. If **P** is a formula of *PL* that contains at least one occurrence of **x** and no **x**-quantifier, then $(\forall \mathbf{x})\mathbf{P}$ and $(\exists \mathbf{x})\mathbf{P}$ are both formulas of *PL*.

This clause does not allow attaching '$(\forall y)$' to '(Fy $\supset$ $(\exists y)$Gy)' because the latter already contains a y-quantifier, '$(\exists y)$'.

We have been omitting the primes that, by the formal requirements of *PL*, are parts of the predicates of *PL*, and we will continue to do so. We will also frequently omit the outermost parentheses of a formula of *PL*, as we did with *SL*. In our usage outermost parentheses are a pair of left and right parentheses that are added, as a pair, when a binary connective is inserted between two formulas of *PL*.

The omission of outermost parentheses should cause no confusion. Note, however, that when outermost parentheses are customarily dropped, it is not safe to assume that every sentence that begins with a quantifier is a quantified sentence. Consider

$$(\forall x)(Fx \supset Ga)$$

and

$$(\forall x)Fx \supset Ga$$

Both begin with quantifiers, but only the first is a quantified sentence. The scope of the x-quantifier in this sentence is the whole formula. The second sentence is a truth-functional compound; the scope of the x-quantifier is just '$(\forall x)Fx$'. It turns out that these two sentences say very different things.

To make complicated formulas of *PL* easier to read, we also allow the use of square brackets, '[' and ']', in place of the parentheses required by clause 3 of the recursive definition of 'formula of *PL*'. (But we will not allow square brackets in place of parentheses in quantifiers.) So, instead of

$$\sim (\forall y)((\exists z)Fzy \supset (\exists x)Gxy)$$

we can write

$$\sim(\forall y)[(\exists z)Fzy \supset (\exists x)Gxy]$$

In later chapters we shall require one further syntactic concept, that of a **substitution instance** of a quantified sentence. We use the notation $\mathbf{P}(\mathbf{a}/\mathbf{x})$ to specify the formula of *PL* that is like **P** except that it contains the individual constant **a** wherever **P** contains the individual variable **x**. Thus if **P** is

$$(Fza \vee \sim Gz)$$

then $\mathbf{P}(c/z)$ is

$$(Fca \vee \sim Gc)$$

> *Substitution instance of* **P**: If **P** is a sentence of *PL* of the form $(\forall\mathbf{x})\mathbf{Q}$ or $(\exists\mathbf{x})\mathbf{Q}$, and **a** is an individual constant, then $\mathbf{Q}(\mathbf{a}/\mathbf{x})$ is a substitution instance of **P**. The constant **a** is the **instantiating constant**.

For example, 'Ga', 'Gb', and 'Gc' are all substitution instances of '$(\exists z)Gz$'. And 'Fab', 'Fbb', and 'Fcb' are all substitution instances of '$(\forall z)Fzb$'. 'Fab' is the result of substituting 'a' for 'z' in 'Fzb', 'Fbb' is the result of substituting 'b' for 'z' in 'Fzb', and 'Fcb' is the result of substituting 'c' for 'z' in 'Fzb'. In forming a substitution instance of a quantified sentence, we drop the initial

quantifier and replace all remaining occurrences of the now free variable with some one constant. Thus '$(\exists y)$Hay' and '$(\exists y)$Hgy' are both substitution instances of '$(\forall x)(\exists y)$Hxy', but 'Hab' is not. (In forming substitution instances *only* the initial quantifier is dropped, and every occurrence of the variable that becomes free when that quantifier is dropped is replaced by the *same* constant.) All the following are substitution instances of '$(\exists w)[Fw \supset (\forall y)(\sim Dwy \equiv Ry)]$':

$$Fd \supset (\forall y)(\sim Ddy \equiv Ry)$$
$$Fa \supset (\forall y)(\sim Day \equiv Ry)$$
$$Fn \supset (\forall y)(\sim Dny \equiv Ry)$$

but

$$Fd \supset (\forall y)(\sim Dny \equiv Ry)$$

is not—for here we have used one constant to replace the first occurrence of 'w' and a different constant to replace the second occurrence of 'w'. Again, in generating substitution instances, each occurrence of the variable being replaced must be replaced by the same individual constant.

Only quantified sentences have substitution instances, and the substitution instances are formed by dropping the initial quantifier. Thus '~ Fa' is *not* a substitution instance of '~ $(\forall x)$Fx'. '~ $(\forall x)$Fx' is a negation, not a quantified sentence, and hence has no substitution instances. '$(\forall x)$Fxb' is not a substitution instance of '$(\forall x)(\forall y)$Fxy' because, while the latter is a quantified sentence, only the initial quantifier can be dropped in forming substitution instances, and here the initial quantifier is '$(\forall x)$', not '$(\forall y)$'.

7.2E EXERCISES

1. Determine, for each of the following, whether it is a formula of *PL*, and if it is, whether it is a sentence of *PL*. If it is not a formula, explain why not. If it is a formula but not a sentence, explain why it is not a sentence. Then, if it is a formula of *PL*, list all of its components and identify the main logical operator, if any, of each by circling it, and for each subformula indicate whether it is an atomic, truth-functionally compound, or quantified formula. We here allow the deletion of outer parentheses and the use of square brackets in place of parentheses around binary compounds.

For example, a correct answer for the expression '~ $(\exists z)$Fz & Hz' would be Formula but not a sentence. The 'z' in 'Hz' is free.

~ $(\exists z)$Fz & Hz	Truth-functional
~ $(\exists z)$Fz	Truth-functional
Hz	Atomic
$(\exists z)$Fz	Quantified
Fz	Atomic

a. Ba & Hc
*b. (∃x)(Fx & Gx)
c. ~ (∀y)Fya
*d. Fz ⊃ (∀z)Fz
e. (∃a)Ga
*f. Hxx ≡ (∃w)Fw
g. (∀x)(∀y) ~ Hxy
*h. (∃y) ~ Hyy & Ga
i. (∀y) ~ Fy ≡ ~ (∃w)Fw
*j. (∀x)Faa
k. (∃z)(Fz & ~ Baz)
*l. (∃x)[Fx & (∀x)(Fx ⊃ Gx)]
m. (∃x)Fx ∨ ~ (∃x)Fx
*n. ~ (∀x)(Gx ≡ Fx)
o. (∃x)(∃y)Lxx
*p. (∀x)[(∃y)Fyx ⊃ (∃y)Fxy]
q. Fa ⊃ (∃x)Fx
*r. Fa ≡ (∀x)Fa
s. ~ Fw ⊃ ~ (∃w)Gww

2. Indicate, for each of the following sentences of *PL*, whether it is an atomic sentence, a truth-functionally compound sentence, or a quantified sentence. Circle the main logical operator, if any.

a. (∀x) (Fx ⊃ Ga)
*b. (∀x) ~ (Fx ⊃ Ga)
c. ~ (∀x)(Fx ⊃ Ga)
*d. (∃w)Raw ∨ (∃w)Rwa
e. ~ (∃x)Hx
*f. Habc
g. (∀x)(Fx ≡ (∃w)Gw)
*h. (∀x)Fx ≡ (∃w)Gw
i. (∃w)(Pw ⊃ (∀y)(Hy ≡ ~ Kyw))
*j. ~ (∃w)(Jw ∨ Nw) ∨ (∃w)(Mw ∨ Lw)
k. ~ [(∃w)(Jw ∨ Nw) ∨ (∃w)(Mw ∨ Lw)]
*l. Da
m. (∀z)Gza ⊃ (∃z)Fz
*n. ~ (∃x)(Fx & ~ Gxa)
o. (∃z) ~ Hza
*p. (∀w)(~ Hw ⊃ (∃y)Gwy)
q. (∀x) ~ Fx ≡ (∀z) ~ Hza

3. Give a substitution instance of each of the following sentences in which 'a' is the instantiating term.

a. (∀w)(Mww & Fw)
*b. (∃y)(Mby ⊃ Mya)
c. (∃z) ~ (Cz ~ Cz)
*d. (∀x)[(Laa & Lab) ⊃ Lax]
e. (∃z)[Fz & ~ Gb) ⊃ (Bzb ∨ Bbz)]
*f. (∃w)[Fw & (∀y)(Cyw ⊃ Cwa)]
g. (∀y)[~ (∃z)Nyz ≡ (∀w)(Mww & Nyw)]
*h. (∀y)[(Fy & Hy) ⊃ [(∃z)(Fz & Gz) ⊃ Gy]]

i. $(\exists x)(Fxb \equiv Gbx)$
*j. $(\forall x)(\forall y)[(\exists z)Hzx \supset (\exists z)Hzy]$
k. $(\forall x) \sim (\exists y)(Hxy \,\&\, Hyx)$
*l. $(\forall z)[Fz \supset (\exists w)(\sim Fw \,\&\, Gwaz)]$
m. $(\forall w)(\forall y)[(Hwy \,\&\, Hyw) \supset (\exists z)Gzw]$
*n. $(\exists z)(\exists w)(\exists y)[(Fzwy \equiv Fwzy) \equiv Fyzw]$

4. Which of the following examples are substitution instances of the sentence '$(\exists w)(\forall y)(Rwy \supset Byy)$'?
a. $(\forall y)Ray \supset Byy$
*b. $(\forall y)(Ray \supset Byy)$
c. $(\forall y)(Rwy \supset Byy)$
*d. $(\forall y)(Rcy \supset Byy)$
e. $(\forall y)(Ryy \supset Byy)$
*f. $(\exists y)(Ray \supset Byy)$
g. $(Ray \supset Byy)$
*h. $(\forall y)(Ray \supset Baa)$

7.3 INTRODUCTION TO SYMBOLIZATION

Recall the sentences about Michael and his co-workers that we discussed in Section 7.1:

> Michael is easygoing.
> Sue is easygoing.
> Michael is taller than Sue and Sue is taller than Henry.
> Sue likes Henry and Michael likes Rita.
> If Rita likes Henry, then Rita is taller than Henry.
> Michael is easygoing if and only if it is not the case that Rita is easygoing.

We can now symbolize these sentences in *PL*. Here, as will frequently be the case throughout the rest of this chapter, we will use a symbolization key. A symbolization key specifies the **universe of discourse** ('UD' for short) or set of things we are talking about. Every UD is a *nonempty* set. A symbolization key also gives the English readings of the predicates of *PL* we will use in our symbolizations and assigns members of the UD to the individual constants we will use. Our symbolization keys will also assign truth-values to any sentence letters of *PL* that we will use in our symbolizations. We will specify the set we are using as the UD either as we do below, by listing the members inside curly brackets, or by using a description of the set, for example 'The set of positive integers'. In symbolizing our sentences about Michael, Sue, Henry, and Rita we will use the following symbolization key:

> UD: The set {Michael, Sue, Henry, Rita}
> Ex: x is easygoing
> Txy: x is taller than y

Lxy: x likes y
m: Michael
s: Sue
h: Henry
r: Rita

The English sentences we are symbolizing and our symbolizations of them in *PL* are as follows:

Michael is easygoing.	Em
Sue is easygoing.	Es
Michael is taller than Sue and Sue is taller than Henry.	Tms & Tsh
Sue likes Henry and Michael likes Rita.	Lsh & Lmr
If Rita likes Henry, then Rita is taller than Henry.	Lrh ⊃ Trh
Michael is easygoing if and only if it is not the case that Rita is easygoing.	Em ≡ ~ Er

In constructing our symbolization key we selected predicate letters and individual constants that may help us remember what English predicates and singular terms they symbolize. We will follow this practice throughout this chapter, but we note that a strong mnemonic connection between the predicates and singular terms of *PL* and the expressions of English they symbolize is not always possible.

We can also use symbolization keys to provide English readings of sentences of *PL*. For example, using our current symbolization key we can read

Lrh ≡ (Lhr & ~ Lhs)

as

Rita likes Henry if and only if Henry likes Rita and does not like Sue.

Using the same symbolization key we can also provide English readings for the following sentences of *PL*:

Lhr & ~ Lrh
Lrh ⊃ Lrm
Trh & ~ Trs
Tsh ⊃ Lhs
(Lmh ∨ Lms) ⊃ (Lmh & Lms)

In English these become, respectively,

Henry likes Rita and Rita does not like Henry.
If Rita likes Henry, then Rita likes Michael.

Rita is taller than Henry and Rita is not taller than Sue.

If Sue is taller than Henry, then Henry likes Sue.

If Michael likes Henry or Michael likes Sue, then Michael likes Henry and Michael likes Sue.

We can, of course, improve on the English. For example, the last sentence of *PL* can be more colloquially read as

If Michael likes either Henry or Sue he likes both of them.

Earlier we gave several examples of English sentences having quantity expressions in positions that singular terms can also occupy. Among these were

Everyone is easygoing.
No one is easygoing.

We can symbolize such sentences in *PL* without using quantifiers provided the discourse within which such sentences occur is about a finite, and for practical purposes, a reasonably small number, of things or individuals. For example, suppose we are again talking about only Michael, Sue, Henry, and Rita. Given this context, 'Everyone is easygoing' is equivalent to 'Michael, Sue, Henry, and Rita are easygoing' and this claim can be symbolized as an iterated conjunction:

(Em & Es) & (Eh & Er)

And 'No one is easygoing' is in this context equivalent to 'Neither Michael nor Sue nor Henry nor Rita is easygoing' and can be symbolized as the negation of an iterated disjunction:

$\sim [(Em \vee Es) \vee (Eh \vee Er)]$

But these techniques are impractical when the number of things or individuals we are talking about is even moderately large. And we cannot, even in principle, symbolize quantity claims about an infinite number of things, say the positive integers, by using iterated conjunctions and disjunctions. For these purposes we do need the quantifiers of *PL*.

Suppose substantially more than four people work in Michael's office and we want to symbolize sentences about this larger group of individuals. We will use the following symbolization key to do so.

UD:	The set of people who work in Michael's office
Lxy:	x likes y
Rxy:	x respects y
m:	Michael
r:	Rita
h:	Henry

The first sentence we will symbolize is

Everyone likes Michael.

In symbolizing English sentences in *PL* it will often be useful to first paraphrase those sentences. Our paraphrases will be analogous to those we used in Chapter 2. We will use the terms 'each' and 'there is a(n)' followed by a variable to specify where quantifiers will occur in *PL,* and we will underline all expressions that are counterparts to the logical operators of *PL*. As 'Everyone likes Michael' is a claim about everyone in the UD, we will paraphrase it as:

Each x is such that x likes Michael.

Using 'Lxy' to symbolize 'x likes y' and using 'm' to designate Michael, our symbolization is

(∀x)Lxm

This sentence says that 'Lxm' is true of each thing in the UD, that is, each thing in the UD likes Michael. Our symbolization key includes

Lxy: x likes y

That we chose, arbitrarily, to use 'x' and 'y' in assigning an English reading to this 2-place predicate *does not mean* that whenever we use this predicate it must be followed by 'x' and then 'y'. A predicate of *PL* can be followed by *any* combination of the appropriate number of variables and individual constants. More generally, in symbolization keys, variables are used to mark the gaps in **n**-place predicates, not to specify what variables are to be used in symbolizations containing those predicates. We also could have used any variable in our paraphrase and any variable in our symbolization. For example, '(∀y)Lym', '(∀z)Lzm', and '(∀w)Lwm' are all correct symbolizations of 'Everyone likes Michael'.

Having symbolized 'Everyone likes Michael' it is easy to symbolize 'Michael likes everyone'. An appropriate paraphrase is

Each x is such that Michael likes x.

Our symbolization is '(∀x)Lmx'. Note that since Michael is part of the UD, it follows both from 'Everyone likes Michael' and from 'Michael likes everyone' that Michael likes Michael, that is, that Michael likes himself.

The sentence

Someone likes Michael and someone does not like Michael

can be paraphrased and symbolized as a conjunction. Our paraphrase is

> There is a y such that y likes Michael and there is a y such that it is not the case that y likes Michael.

Our paraphrase is readily symbolized as

> $(\exists y)\text{Lym} \;\&\; (\exists y) \sim \text{Lym}$

Again, there is no requirement that the variables we use in symbolization keys also be used in corresponding positions in symbolizations based on those symbolization keys. And there is no requirement that the variable we use in one subformula of a sentence of *PL* also be used in other subformulas unless those variables are being interpreted by the same quantifier. So we could equally correctly have symbolized 'Someone likes Michael and someone does not like Michael' as

> $(\exists x)\text{Lxm} \;\&\; (\exists z) \sim \text{Lzm}$

Note that '$(\exists y)(\text{Lym} \;\&\; \sim \text{Lym})$' says something very different from '$(\exists y)\text{Lym} \;\&\; (\exists y) \sim \text{Lym}$'. The former sentence says that there is someone who both likes Michael and does not like Michael.

The sentence

> Everyone who likes Michael also respects him

is readily paraphrased and symbolized as follows:

> Each x is such that (if x likes Michael then x respects Michael).
>
> $(\forall w)(\text{Lwm} \supset \text{Rwm})$

And 'Someone likes and respects Michael' can be paraphrased and symbolized as

> There is a y such that (y likes Michael and y respects Michael).
>
> $(\exists y)(\text{Lym} \;\&\; \text{Rym})$

It is important to understand why the main logical operator of the immediate subformula of '$(\forall w)(\text{Lwm} \supset \text{Rwm})$' is a '$\supset$' while that of '$(\exists x)(\text{Lxm} \;\&\; \text{Rxm})$' is an '&'. '$(\forall w)(\text{Lwm} \;\&\; \text{Rwm})$' and '$(\forall w)(\text{Lwm} \supset \text{Rwm})$' say quite different things. The former says that each member of the UD both likes and respects Michael. The latter attributes 'respects Michael' *only* to those members of the UD who do like Michael. When the UD is heterogeneous and we want to attribute some property to members of the UD that are of a particular sort, the most common way of doing so is to use a universally quantified sentence whose immediate subformula is a material conditional,

that is, to use a sentence of the form $(\forall \mathbf{x})(\mathbf{P} \supset \mathbf{Q})$. Such a sentence does not say that every member of the UD is of the sort **P**, nor does it say that every member of the UD is of the sort **Q**. Rather, it says of those members of the UD that *are* of the sort **P** that they are *also* of the sort **Q**. So, while a sentence of the form $(\forall \mathbf{x})(\mathbf{P} \supset \mathbf{Q})$ does say of every member of the UD that it is of the sort $\mathbf{P} \supset \mathbf{Q}$, when a member of the UD is not of the sort **P** this comes to naught.

On the other hand, when we do want to say that one or more members of the UD are of the sort **P** and of the sort **Q** our symbolization will be a sentence of the form $(\exists \mathbf{x})(\mathbf{P} \,\&\, \mathbf{Q})$. Note that '$\supset$' is not appropriate in this case, for $(\exists \mathbf{x})(\mathbf{P} \supset \mathbf{Q})$ says that there is at least one member of the UD such that if it is of the sort **P** then it is also of the sort **Q**. If a member of the UD is not of the sort **P**, then trivially it is such that *if* it is of the sort **P** (which it is not) *then* it is also of the sort **Q**. This is a much weaker claim than the claim made by a sentence of the form $(\exists \mathbf{x})(\mathbf{P} \,\&\, \mathbf{Q})$.

For the reasons just given, many of our symbolizations of English sentences will be either of the form $(\forall \mathbf{x})(\mathbf{P} \supset \mathbf{Q})$ or of the form $(\exists \mathbf{x})(\mathbf{P} \,\&\, \mathbf{Q})$ (where the variable **x** occurs in both **P** and **Q**). Sentences of the form $(\forall \mathbf{x})(\mathbf{P} \,\&\, \mathbf{Q})$ as well as those of the form $(\exists \mathbf{x})(\mathbf{P} \supset \mathbf{Q})$ are far less common as symbolizations of English sentences. Sentences of the form $(\forall \mathbf{x})(\mathbf{P} \,\&\, \mathbf{Q})$ are very strong. They say each thing in the UD is both of the sort **P** and of the sort **Q**. On the other hand, sentences of the form $(\exists \mathbf{x})(\mathbf{P} \supset \mathbf{Q})$ are extremely weak. On truth-functional grounds such sentences are equivalent to sentences of the form $(\exists \mathbf{x})(\sim \mathbf{P} \vee \mathbf{Q})$, which means all they say is that there is at least one thing that either is not of the sort **P** or is of the sort **Q**. The moral in both cases is that when we find we have constructed a symbolization that is of the form $(\forall \mathbf{x})(\mathbf{P} \,\&\, \mathbf{Q})$ or of the form $(\exists \mathbf{x})(\mathbf{P} \supset \mathbf{Q})$ it is a good idea to double-check to make sure our symbolization is correct.

The quantity terms 'any' and 'anyone' are often appropriately symbolized by universal quantifiers. Such is the case in the sentence

> Anyone who respects Michael also respects Rita.

Our paraphrase and symbolization are

> Each x is such that (if x respects Michael then x respects Rita)
>
> $(\forall x)(Rxm \supset Rxr)$

But some uses of 'any' *can* be symbolized by an existential quantifier. Consider 'If anyone respects Rita, Henry does'. This sentence is a material conditional and the consequent says that a specific person, Henry, respects Rita. This sentence can be paraphrased in two different ways:

> If there is an x such that x respects Rita then Henry respects Rita,
> Each x is such that (if x respects Rita then Henry respects Rita).

These can be symbolized, respectively, as '$(\exists x)Rxr \supset Rhr$' and '$(\forall x)(Rxr \supset Rhr)$'. The first sentence of *PL* is a material conditional, while the second is a universally quantified sentence whose immediate component is a material conditional. These sentences are equivalent, as are our paraphrases. If it is true that if there is a person that respects Rita, then Henry does, it is also true of each person that if that person respects Rita (which means that at least one person respects Rita) then Henry does, and vice versa. But neither of the following is a correct symbolization of 'If anyone respects Rita, Henry does':

$(\forall x)Rxr \supset Rhr$
$(\exists x)(Rxr \supset Rhr)$

The first of these sentences of *PL* says that <u>if</u> <u>each person x is such that</u> x respects Rita <u>then</u> Henry respects Rita, that is, that if *everyone* respects Rita then Henry does. This is not news, for Henry, being one of 'everyone', of course respects Rita if everyone does. That is, the sentence is logically true.

The second of these sentences of *PL* is an existentially quantified sentence whose immediate subformula, '$Rxr \supset Rhr$', is a material conditional. As pointed out above, a sentence of a form such as '$(\exists x)(Rxr \supset Rhr)$' is equivalent to '$(\exists x)(\sim Rxr \sim Rhr)$', which says that there is someone such that either that person does not respect Rita or Henry respects Rita. This is also a logical truth—and not surprisingly, because it is equivalent to the symbolization '$(\forall x)Rxr \supset Rhr$' that we discussed in the previous paragraph (later in this section we will explain why the equivalence holds). Not all sentences of the form $(\exists \mathbf{x})(\mathbf{P} \supset \mathbf{Q})$, where **P** contains **x** but **Q** does not, are logically true but, as we have noted, no such sentence makes a very strong claim.

In English there are a fair number of different ways we can say that everything of this sort is also of that sort. For example, if we are talking about the people in Michael's office, all of the following sentences can be used to make the same claim:

Everyone who respects Henry also respects Rita.
Each person who respects Henry also respects Rita.
All those who respect Henry also respect Rita.
Anyone who respects Henry also respects Rita.
Those who respect Henry also respect Rita.
A person who respects Henry also respects Rita.

All of these can appropriately be paraphrased and symbolized as follows:

<u>Each x is such that</u> (<u>if</u> x respects Henry <u>then</u> x respects Rita).

$(\forall x)(Rxh \supset Rxr)$

Examples of English sentences that are appropriately symbolized as existentially quantified sentences are

> There is someone who respects Henry and Rita,
> Someone respects Henry and Rita,

and

> At least one person respects Henry and Rita

These can all be paraphrased and symbolized as

> There is an x such that (x respects Henry and x respects Rita)
>
> $(\exists x)(Rxh \ \& \ Rxr)$

It is at least arguable that there are some uses of 'some' in English where 'some' means 'at least two'. We here note that the existential quantifier of *PL* always means 'there is at least one'. In Section 7.5 we will introduce an expansion of *PL*, *PLE*, and in that language we will be able to adequately symbolize such expressions as 'there are at least two' and 'there are exactly two' and thus accommodate those uses of 'some' in English where 'some' means' 'at least two'.

We next symbolize some sentences about the animals in the Saint Louis Zoo.[5]

- The dolphins want to swim with us.
- The jaguars prance on tree limbs.
- The grizzlies are discontent when forced to dine without wine.
- The alligators sup in sullen silence and the polar bears sunbathe without swim suits.
- The gorillas stare mutely but intently as the rhinos dance divinely.
- The great horned owls see and know all but say nothing.
- Neither the tigers nor the zebras ever change their stripes.

Our symbolization key will be

> UD: The set consisting of animals in the Saint Louis Zoo
> Az: z is an alligator
> Bz: z is a grizzly bear
> Cz: z sometimes changes its stripes
> Dz: z is a dolphin
> Ez: z sees everything
> Fz: z is discontent when forced to dine without wine
> Gz: z is a gorilla

[5]Some readers will recognize the influence of Simon and Garfunkel's whimsical song *At the Zoo*.

Iz: z stares intently
Jz: z is a jaguar
Kz: z knows everything
Mz: z stares mutely
Nz: z says nothing
Oz: z is a great horned owl
Pz: z is a polar bear
Rz: z is a rhinoceros
Sz: z sups in sullen silence
Tz: z is a tiger
Uz: z prances upon tree limbs
Vz: z dances divinely
Wz: z wants to swim with us
Xz: z sunbathes without a swim suit
Zz: z is a zebra

As is to be expected with this large a symbolization key, not all of the predicate letters we have selected are mnemonic reminders of what they symbolize. Our first three symbolizations are straightforward:

- The dolphins want to swim with us.

  $(\forall x)\,(Dx \supset Wx)$

- The jaguars prance upon tree limbs.

  $(\forall y)\,(Jy \supset Uy)$

We can paraphrase our third example

- The grizzlies are discontent when forced to dine without wine. Each x is such that if x is a grizzly then x is discontent when forced to dine without wine and symbolize it as

  $(\forall w)\,(Bw \supset Fw)$

Our next three examples can be paraphrased and symbolized as conjunctions.

- The alligators sup in sullen silence and the polar bears sunbathe without swimsuits.

  Each w is such that if w is an alligator then x sups in sullen silence and each x is such that if x is a polar bear then x sunbathes without a swimsuit.

  $(\forall w)\,(Aw \supset Sw)\ \&\ (\forall x)\,(Px \supset Xx)$

Our current example can also (and equivalently) be paraphrased and symbolized as

Each w is such that [(if w is an alligator then w sups in sullen silence) and (if w is a polar bear then w sunbathes without a swimsuit)]

$(\forall w)[(Aw \supset Sw) \ \& \ (Pw \supset Xw)]$

- The gorillas stare mutely but intently; moreover, the rhinos dance divinely.

  Each z is such that [if z is a gorilla then (z stares mutely and z stares intently)] and each z is such that (if z is a rhino then z dances divinely)

  $(\forall z)[Gz \supset (Mz \ \& \ Iz)] \ \& \ (\forall z)(Rz \supset Vz)$

An alternative paraphrase and symbolization are equally appropriate:

Each x is such that ([if x is a gorilla then (x stares mutely and x stares intently)] and (if x is a rhino then x dances divinely))

$(\forall x)([Gx \supset (Mx \ \& \ Ix)] \ \& \ (Rx \supset Vx))$

Note that in our paraphrases we have used 'and' in place of both 'but' and 'moreover'.

- The great horned owls see and know all but say nothing.

  Each y is such that (if y is a great horned owl then [(y sees all and y knows all) and y says nothing])

  $(\forall y)(Oy \supset [(Ey \ \& \ Ky) \ \& \ Ny])$

- Neither the tigers nor the zebras ever change their stripes

can be paraphrased and symbolized in various ways, including as a conjunction of two universally quantified sentences and as a quantified sentence whose immediate component is a material conditional whose antecedent is a disjunction:

Each x is such that (if x is a tiger then it is not the case that x sometimes changes its stripes) and each y is such that (if y is a zebra then it is not the case that y sometimes changes its stripes).

$(\forall x)(Tx \supset \sim Cx) \ \& \ (\forall y)(Zy \supset \sim Cy)$

Each x is such that [if (x is a tiger or x is a zebra) then it is not the case that x sometimes changes its stripes]

$(\forall x)[(Tx \vee Zx) \supset \sim Cx]$

We specified that the sentences we have just symbolized are about a specific group of animals—those at the Saint Louis Zoo. This makes the use of 'the' ('The dolphins . . .', 'The grizzlies . . .') appropriate. In English, as we have already seen, we don't always use 'all', 'every', 'each' or other quantity terms when making universal claims. For example, 'Dolphins are good swimmers' is appropriately used to make a claim about all dolphins, everywhere. But in 'The dolphins want to swim with us' the use of 'the' indicates we are talking about some specific group of dolphins.

In our symbolization key we used 'z' in interpreting our one-place predicates. But in our symbolizations we sometimes used 'x', sometimes 'y', sometimes 'w', and sometimes 'z'. As we noted earlier, the variables we use in symbolization keys to interpret predicates need not be the variables we use in quantified sentences containing those predicates. What matters is that the variable we use in a quantifier *matches* the variables that the quantifier is intended to interpret. '$(\forall y)(Dy \supset Wy)$' and '$(\forall w)(Dw \supset Ww)$' are equally good symbolizations of 'The dolphins want to swim with us' but '$(\forall x)(Dx \supset Wy)$' is not a symbolization of that sentence at all. It is, in fact, not a sentence of *PL* because it contains a free variable, 'y'.

In symbolizing our sentences about zoo animals we constructed universally quantified sentences whose immediate components are truth-functional compounds, often material conditionals. Because these symbolizations are *universally* quantified sentences their immediate components—truth-functional compounds—are attributed to each and every member of the UD. We again note that the attribution is vacuous, comes to nothing, when the attribution is to a member of the UD that is *not* of the sort specified by the antecedent of the material conditional. So while '$(\forall w)(Jw \supset Uw)$' attributes '$Jw \supset Uw$' to all members of the UD, it attributes 'Uw' ('w prances upon tree branches') only to those members that 'Jw' ('w is a jaguar') is true of and says nothing of the non-jaguars.

We now augment our present symbolization key by adding the following two-place predicates to symbolize the sentences that follow:

Hxy: x is heavier than y
Lxy: x likes y

- Every animal likes every animal

  Each x and each y are such that (or each pair x and y is such that) x likes y.

  $(\forall x)(\forall y)Lxy$

- Every animal likes at least one animal.

  Each x is such that there is a y such that x likes y.

  $(\forall x)(\exists y)Lxy$

- There is an animal that likes all the animals.

  There is a z such that each w is such that z likes w.

  $(\exists z)(\forall w)Lzw$

- No animal is heavier than every animal.

  It is not the case that there is an x such that each y is such that x is heavier than y.

  $\sim (\exists x)(\forall y)Hxy$

- If an animal is heavier than another, then the second is not heavier than the first.

  Each x and each y are such that (if x is heavier than y then it is not the case that y is heavier than x).

  $(\forall x)(\forall y)(Hxy \supset \sim Hyx)$

- No animal is heavier than itself.

  It is not the case that there is an x such that x is heavier than x.

  $\sim (\exists x)Hxx$

- Every gorilla likes every rhinoceros.

  Each x is such that [if x is gorilla then each y is such that (if y is a rhinoceros then x likes y)].

  $(\forall x)[Gx \supset (\forall y)(Ry \supset Lxy)]$

'Every gorilla likes every rhinoceros' can also be equivalently paraphrased and symbolized as:

Each x and each y are such that if [(x is a gorilla and y is a rhinoceros) then x likes y].

$(\forall x)(\forall y)[(Gx \ \& \ Ry) \supset Lxy]$

- Every gorilla likes at least one rhinoceros.

Each x is such that [if x is a gorilla then there is a y such that (y is a rhinoceros and x likes y)].

$(\forall x)[Gx \supset (\exists y)(Ry \ \& \ Lxy)]$

- Every gorilla likes at least one rhinoceros and does not like at least one jaguar.

  Each z is such that (if z is a gorilla then [there is a w such that (w is a rhinoceros and z likes w) and there is a y such that (y is a jaguar and it is not the case that z likes y)]).

  $(\forall z)(Gz \supset [(\exists w)(Rw \ \& \ Lzw) \ \& \ (\exists y)(Jy \ \& \sim Lzy)])$

- The dolphins don't like the grizzly bears.

  Each y is such that [if y is a dolphin then each w is such that (if w is a grizzly bear then it is not the case that y likes w)].

  $(\forall y)[Dy \supset (\forall w)(Bw \supset \sim Lyw)]$

- Some tigers like all the jaguars but no tiger likes any grizzly bear.

  There is an x such that [x is a tiger and each y is such that (if y is a jaguar then x likes y)] and it is not the case that there is a w such that [w is a tiger and there is a z such that (z is a grizzly bear and w likes z)].

  $(\exists x)[Tx \ \& \ (\forall y)(Jy \supset Lxy)] \ \& \sim (\exists w)[Tw \ \& \ (\exists z)(Bz \ \& \ Lwz)]$

The right conjunct of our paraphrase can also be correctly symbolized as '$\sim (\exists w)(\exists z)[(Tw \ \& \ Bz) \ \& \ Lwz)]$'.

- Every dolphin is heavier than every great horned owl but no dolphin is heavier than any rhinoceros.

  Each x and each y are such that [if (x is a dolphin and y is a great horned owl) then x is heavier than y] and it is not the case that there is an x and there is a y such that [(x is a dolphin and y is an rhinoceros) and x is heavier than y].

  $(\forall x)(\forall y)[(Dx \ \& \ Oy) \supset Hxy] \ \& \sim (\exists x)(\exists y)[(Dx \ \& \ Ry) \ \& \ Hxy]$

- Anything that is heavier than every gorilla is a rhinoceros.

  Each w is such that [if each y is such that (if y is a gorilla then w is heavier than y) then w is a rhinoceros].

  $(\forall w)[(\forall y)(Gy \supset Hwy) \supset Rw]$

Notice that the scope of '(∀w)' is the entire sentence, while the scope of '(∀y)' is just '(∀y)(Gy ⊃ Hwy)'. Compare this sentence of *PL* to

$$(\forall w)(\forall y)[Gy \supset (Hwy \supset Rw)],$$

which says that each pair of members of the UD is such that if one is a gorilla then if the other is heavier than that gorilla then it is a rhinoceros. In better English, this comes to 'Anything that is heavier than *any* gorilla [even one gorilla] is a rhinoceros'.

We can also use our symbolization key to construct English readings of the following sentences of *PL*:

- $(\forall y)[Jy \supset (\forall x)(Tx \supset \sim Lyx)]$

  Each y is such that if [y is a jaguar then each x is such that (if x is a tiger then it is not the case that y likes x)].

Since we are talking about each y and each x, this comes to

> Each jaguar and each tiger are such that it is not the case that the jaguar likes the tiger,

or more idiomatically:

> The jaguars do not like the tigers.

Note that it would be a mistake to read the sentence of *PL* we are currently considering as 'All the jaguars don't like all the tigers', for this English sentence is ambiguous. It can be taken to mean that it is not the case that all the jaguars like all the tigers, which is consistent with some of the jaguars liking some or all of the tigers. Our next example is

- $(\exists w)[Gw \ \& \ (\forall x)(Bx \supset Lwx)] \ \& \sim (\forall z)[Gz \supset (\forall x)(Bx \supset Lzx)]$

The left conjunct can be read as

> There is a w such that [w is a gorilla and each x is such that (if x is a grizzly bear then w likes x)]

which comes to 'There is a gorilla that likes every grizzly bear'. The right conjunct can be read as

> It is not the case that each z is such that [if z is a gorilla then each x is such that (if x is a grizzly bear then z likes x)]

which comes to 'Not every gorilla likes every grizzly bear'. An appropriate reading of the entire conjunction is

Some, but not all, of the gorillas like all the grizzly bears.

While *PL* can be used to symbolize claims about almost anything, including people, animals, all living things, countries, numbers, and whatever else our ontology (those things we take to exist) includes, the positive integers (the whole numbers 1, 2, 3, . . .) constitute an especially interesting UD for at least two reasons, and we will frequently use them as our UD in examples and exercises in the rest of this chapter as well as throughout Chapter 8. First, once one is familiar with the basic nature of the positive integers, symbolizing claims about them becomes fairly straightforward. Many claims about the positive integers and the relations among them are clear and unambiguous. This is often not true of sentences about other kinds of things. Second, if there is an interpretation of a set of sentences of *PL* on which all the members of the set are true then there is such an interpretation that uses the positive integers as the universe of discourse. This will be of considerable importance when we are working with the semantics of *PL*, as we will see in Chapter 8.

We will next symbolize a number of sentences about the positive integers. Readers may find it useful to consult Appendix 1, which details some simple facts about the positive integers, before proceeding. We will use the following symbolization key:

UD:	The set of positive integers
Lxy:	x is less than y
Ox:	x is odd
Ex:	x is even
Exy:	x times y is even
Oxy:	x times y is odd
Px:	x is a prime number
Sxy:	x is the successor of y (x = y + 1)
a:	2

- There is a smallest positive integer.

Symbolizing this sentence is fairly straightforward. All we need say is that there is a positive integer such that no positive integer is smaller than it. And this is what

$$(\exists y) \sim (\exists x)\,Lxy$$

says.

- There is no largest positive integer.

We do not have a predicate for 'x is larger than y' in our symbolization key, and we do not need one to symbolize this sentence. What would a largest positive integer be? It would be an integer such that there is no positive integer it is

less than. And we do have a predicate for 'x is less than y': 'Lxy'. So the following says that there is a positive integer y such that there is no positive integer x such that y is less than x. That is, it says there is a largest positive integer.

$$(\exists y) \sim (\exists x) Lyx$$

So the negation of this sentence

$$\sim (\exists y) \sim (\exists x) Lxy$$

symbolizes 'There is no largest positive integer'.

- An odd positive integer times an odd positive integer is odd.

We can paraphrase and symbolize this sentence as follows:

Each x and each y are such that [if (x is odd and y is odd) then x times y is odd]

$$(\forall x)(\forall y)[(Ox \ \& \ Oy) \supset Oxy]$$

- There is a pair of primes such that one member of the pair is the successor of the other member of the pair.

Our paraphrase and symbolization are

There is an x and there is a y such that [(x is prime and y is prime) and y is the successor of x]

$$(\exists x)(\exists y)[(Px \ \& \ Py) \ \& \ Syx]$$

(This claim is true; 2 and 3 are both primes and 3 is the successor of 2.)

- An even positive integer times an even positive integer is an even positive integer.

Our paraphrase and symbolization are

Each x and each y are such that [(if x is even and y is even) then x times y is even]

$$(\forall x)(\forall y)[(Ex \ \& \ Ey) \supset Exy]$$

- An even positive integer times an odd positive integer is an even positive integer.

Our symbolization of this sentence is like that of the preceding one, substituting 'Oy' for 'Ey', thus specifying that the second member of the pair is an odd, not an even, positive integer:

$$(\forall x)(\forall y)[(Ex \ \& \ Oy) \supset Exy]$$

Our next example combines the claims of the preceding two examples:

- An even positive integer times an even or an odd positive integer is even.

Our symbolization is

$$(\forall x)(\forall y)[(Ex \ \& \ (Ey \vee Oy)) \supset Exy]$$

Our final three sentences concern prime numbers. The first is

- 2 is prime and 2 has a prime successor.

$$Pa \ \& \ (\exists y)(Sya \ \& \ Py)$$

A literal reading of this sentence is '2 is prime and there is a successor of 2 and it is prime'.

Our second sentence about primes is

- 2 is prime and no prime number is less than 2.

Our symbolization is straightforward:

$$Pa \ \& \sim (\exists x)(Px \ \& \ Lxa)$$

Our last sentence concerning primes is

- 2 is an even prime and every prime greater than 2 is odd.

We can symbolize this sentence as a conjunction:

$$(Pa \ \& \ Ea) \ \& \ (\forall y)[(Py \ \& \ Lay) \supset Oy]$$

Before concluding this section, we note a limited parallel between *PL* and Aristotelian logic. Aristotelian logic recognizes four kinds of quantity claims, traditionally termed '**A-**', '**E-**', '**I-**', and '**O-sentences**':[6]

A-sentences	All **A**s are **B**s.
E-sentences	No **A**s are **B**s.

[6]The use of 'A', 'E', 'I', and 'O' to designate kinds of sentences apparently dates to the Middle Ages. A- and I-sentences are thought of as affirmations and match the first two vowels in the Latin verb 'affirmo' (which means 'I affirm') while E- and O-sentences are thought of as denials and match the first two vowels in the Latin verb 'nego' (which means 'I deny'). See Francis Garden, *Outline of Logic: For the Use of Teachers and Students*, 2nd ed. (Oxford and London: Rivingtons, 1871, p. 65).

I-sentences	Some **A**s are **B**s.
O-sentences	Some **A**s are not **B**s.

Here '**A**' and '**B**' are metavariables ranging over general terms, that is, terms such as 'people', 'horses', 'orators', 'fish', 'voters', and 'Athenians'. Here are examples of each kind of sentence:

A-sentence	All horses are mammals.
E-sentence	No horses are mammals.
I-sentence	Some horses are mammals.
O-sentence	Some horses are not mammals.

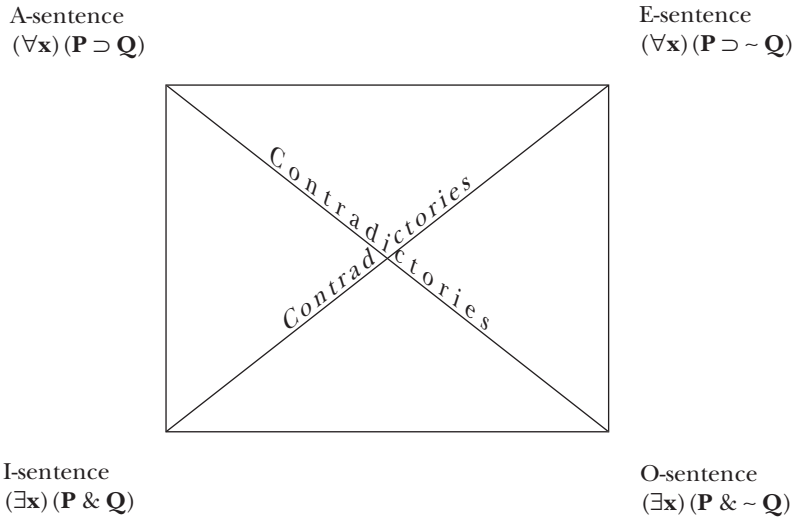
PL contains analogues to each of these kinds of sentences. Where **x** is a variable of *PL* and **P** and **Q** are open sentences of *PL*, each of which contains at least one occurrence of **x** and no **x**-quantifier, the *PL* analogues are

A-sentence	$(\forall \mathbf{x})\,(\mathbf{P} \supset \mathbf{Q})$
E-sentence	$(\forall \mathbf{x})\,(\mathbf{P} \supset \sim \mathbf{Q})$
I-sentence	$(\exists \mathbf{x})\,(\mathbf{P} \,\&\, \mathbf{Q})$
O-sentence	$(\exists \mathbf{x})\,(\mathbf{P} \,\&\, \sim \mathbf{Q})$

We can use these templates to provide symbolizations of the above four claims about horses:

A-sentence	$(\forall x)\,(Hx \supset Mx)$
E-sentence	$(\forall x)\,(Hx \supset \sim Mx)$
I-sentence	$(\exists x)\,(Hx \,\&\, Mx)$
O-sentence	$(\exists x)\,(Hx \,\&\, \sim Mx)$

We are here taking our UD to be the set of living things, and using 'Hx' to symbolize 'x is a horse' and 'Mx' to symbolize 'x is a mammal'. The relations among these kinds of claims are often presented through a **square of opposition:**



Aristotle held that there are several interesting relationships among the four types of sentences displayed at the corners of the square of opposition. In *PL* the interesting relations are those between sentence types at opposite ends of the diagonal lines. These constitute contradictory sentence pairs. That is, an A-sentence is equivalent to the *negation* of the corresponding O-sentence and vice versa. And an E-sentence is equivalent to the *negation* of the corresponding I-sentence, and vice versa, yielding the following pairs of equivalent sentence forms:

$$(\forall \mathbf{x})(\mathbf{P} \supset \mathbf{Q}) \quad \text{and} \quad \sim (\exists \mathbf{x})(\mathbf{P} \,\&\, \sim \mathbf{Q})$$
$$(\forall \mathbf{x})(\mathbf{P} \supset \sim \mathbf{Q}) \quad \text{and} \quad \sim (\exists \mathbf{x})(\mathbf{P} \,\&\, \mathbf{Q})$$
$$(\exists \mathbf{x})(\mathbf{P} \,\&\, \mathbf{Q}) \quad \text{and} \quad \sim (\forall \mathbf{x})(\mathbf{P} \supset \sim \mathbf{Q})$$
$$(\exists \mathbf{x})(\mathbf{P} \,\&\, \sim \mathbf{Q}) \quad \text{and} \quad \sim (\forall \mathbf{x})(\mathbf{P} \supset \mathbf{Q})$$

Knowing the foregoing equivalences can be helpful in symbolizing English sentences in *PL*, for these equivalences provide alternative patterns for symbolizing sentences that can be symbolized as A-, E-, I-, or O-sentences.

7.3**E** EXERCISES

1. Symbolize the following sentences in *PL*, without using quantifiers, using the following symbolization key:

UD: The set {Bob, Carol, David, Emily}
Gy: y will graduate
Jy: y will get a job
Ay: y will join the Army
Ly: y will become a longshoreman
Mxy: x will make more money than y
b: Bob
c: Carol
d: David
e: Emily

a. Bob and Carol will graduate and so will either David or Emily.
*b. If David doesn't graduate he will join the Army and if Emily doesn't graduate she will become a longshoreman.
c. If David joins the Army and Emily becomes a longshoreman, she will make more money than he will.
*d. All of those who graduate will get jobs.
e. If David will graduate they will all graduate.
*f. If at least one of them graduates they will all graduate.

2. Symbolize the following sentences in *PL*.

UD: The set of positive integers
Ex: x is even
Ox: x is odd

Lxy: x is less than y
Px: x is prime
a: 1
b: 2
c: 4
d: 100

a. Some positive integers are odd and some are even.
*b. Some positive integers are prime but not all positive integers are prime.
c. No positive integer is less than 1.
*d. No positive integer is less than itself.
e. 2 is less than 4 and 4 is less than some positive integer.
*f. Not every positive integer is less than 100.
g. Not all positive integers are prime and not all positive integers are even.
*h. Not all positive integers are primes and not all positive integers are non-primes.
i. All positive integers are even if and only if all positive integers are not odd.
*j. 1 is not prime and no positive integer is less than 1.
k. There is a positive integer that is less than 100.

3. Symbolize the following sentences in *PL*, using quantifiers wherever appropriate, using the following symbolization key:

UD: The set of seniors at Dartmouth College
Gy: y will graduate
Jy: y will get a job
Ay: y will join the Army
Ly: y will become a longshoreman
Mxy: x will make more money than y
b: Bob
c: Carol
d: David
e: Emily

a. All of those who graduate will get jobs.
*b. If David will graduate, all seniors will graduate.
c. If at least one senior graduates, they will all graduate.
*d. Everyone who doesn't graduate will join the Army.
e. If anyone joins the Army both Carol and David will.
*f. Everyone will graduate or no one will graduate.
g. Each senior will either graduate or not graduate.
*h. If anyone who graduates becomes a longshoreman Emily will become a longshoreman.
i. Everyone who becomes a longshoreman will make more money than will everyone who does not.
*j. Each senior will join the Army if and only if he or she does not graduate.

4. Using the following symbolization key, symbolize the following sentences in *PL*. (Note: Not all of these sentences are true.)

UD: The set of positive integers
Px: x is a prime

Ox: x is odd
Ex: x is even
Lxy: x is less than y
Txy: x times y is prime
Dxy: x is evenly divisible by y (x is divisible by y without remainder)
a: 2

a. There is a positive integer that is less than all primes.
*b. A positive integer is even if and only if it is evenly divisible by 2.
c. A prime times a prime is not prime.
*d. A prime times an even positive integer is not prime.
e. A prime times any positive integer greater than 1 is not prime.
*f. If a pair of positive integers is such that the first is evenly divisible by the second, then either both integers are even or both are odd.
g. If a pair of positive integers is such that the first is evenly divisible by the second and the second is greater than 1, then either both integers are even or both are odd.
*h. For each prime, there is a greater non-prime.

7.4 SYMBOLIZATION FINE-TUNED

In this section we discuss some missteps that need to be avoided in symbolizing English sentences in *PL*, and we symbolize some sentences that are more complex than the ones we have so far dealt with.

There are contexts in English and other natural languages in which singular terms cannot be interpreted as denoting or referring to anything, and there are contexts in which predicates cannot be interpreted as we have been interpreting them. These contexts arise because we can think, dream, speculate, hunt for, and believe in (and give names to) things that do not exist. Consider, for example, the following claims:

Ponce de Leon is hunting for the Fountain of Youth.
Max is looking for trolls.

Ponce de Leon was a Spanish explorer of the fifteenth century who allegedly spent a lot of time looking for the Fountain of Youth. But of course there is no such thing. The nonexistence of such a fountain does not keep people from looking for it, though of course that nonexistence does prevent anyone from finding it. So too, although Norse mythology contains numerous descriptions of trolls there are no trolls. Nonetheless, it may well be true that our benighted friend Max is out looking for trolls.

Because there is no Fountain of Youth we cannot symbolize the sentence concerning Ponce de Leon as

Hpf

where ‘Hxy’ symbolizes ‘x is hunting for y’, ‘p’ designates Ponce de Leon, and ‘f’ designates the Fountain of Youth, because there is no such thing for ‘f’ to designate. Nor can we symbolize ‘Max is looking for trolls’ as

$$(\exists y)\,(Ty \;\&\; Lmy)$$

where ‘Tx’ symbolizes ‘x is a troll’, ‘Lxy’ symbolizes ‘x is looking for y’, and ‘m’ designates Max, because the English sentence ‘Max is looking for trolls’ does not entail ‘There are trolls’. For these reasons, we should instead symbolize the claim about Ponce de Leon as an atomic sentence of *PL* such as

$$Fp$$

where ‘Fx’ symbolizes ‘x is looking for the Fountain of Youth’ and ‘p’ designates Ponce de Leon. And we should symbolize our sentence about Max as an atomic sentence such as

$$Tm$$

where ‘m’ designates ‘Max’ and ‘Tx’ symbolizes ‘x is looking for trolls’. In the first case we have embedded the non-referring expression ‘the Fountain of Youth’ in a predicate, thus keeping it out of referential position. In the second case we have embedded ‘trolls’ in a larger predicate to avoid the problematic existential quantification.

A related problem arises when someone is looking for or seeking an object of a kind of which there are instances, but no particular instance is being looked for. Suppose that an orangutan—Sally, to be specific—has gone missing from the Saint Louis Zoo and the zookeeper, Mike by name, is in pursuit of her. In this situation the zookeeper is looking for a particular orangutan. Finding another orangutan might be a surprise, and perhaps even a pleasant surprise (for the zoo is short on orangutans), but this will not bring Mike’s search to an end. He is after Sally, not just any orangutan. In this situation, we can symbolize ‘Mike is looking for an orangutan missing from the Saint Louis Zoo’ as

$$(\exists z)\,[(Oz \;\&\; Mz) \;\&\; Lmz],$$

where ‘Oz’ symbolizes ‘z is an orangutan’, ‘Mz’ symbolizes ‘z is missing from the Saint Louis Zoo’, ‘Lwz’ symbolizes ‘w is looking for z’, and ‘m’ designates Mike. Similarly, if all of the zoo’s orangutans have gone missing and Mike is in pursuit, we can accurately say that Mike is looking for all of the missing orangutans and symbolize this claim as

$$(\forall y)\,[(Oy \;\&\; My) \supset Lmy]$$

In the envisioned situation this sentence of *PL* accurately says ‘Each y is such that if y is an orangutan and y is missing from the Saint Louis Zoo then Mike

is looking for y'. But the situation is quite different if Mike has been sent to Indonesia to acquire an orangutan for the zoo. In this context

Mike is looking for an orangutan
cannot be symbolized as
$(\exists x)(Ox \ \& \ Lmx)$

because it is not true that there is a particular orangutan that Mike is looking for. Nor is

$$(\forall x)(Ox \supset Lmx)$$

an appropriate characterization of Mike's activity, for this sentence says that he is looking for *all* orangutans, and he is not. Mike is neither looking for one particular orangutan nor looking for all orangutans. He does want to acquire an orangutan, but any orangutan will suffice. So we should symbolize 'Mike is looking for an orangutan' as an atomic sentence of *PL,* say, 'Lm, where 'Lx' symbolizes 'x is looking for an orangutan' and 'm' again designates Mike.

The general point is that we can look for, believe in, and dream about things that do not exist and we can look for, speculate about, and hope to find a certain sort of thing without there being a particular thing that we are looking for, speculating about, or hoping to find. We must symbolize sentences concerning these activities as we have just done, by embedding the problematic language in predicates that specify the relevant activity (thinking about, searching for, hoping to find, and so on).

We turn now to a more general discussion of how to decide what predicates it is appropriate to use in symbolizing sentences in *PL.* Usually this is a straightforward matter. But consider sentences such as the following:

There are rabid bats in the attic.

In symbolizing sentences such as this, where an adjective modifies a noun, we must decide how many predicates we should use. Should we use a single predicate, 'x is a rabid bat in the attic', two predicates, 'x is a rabid bat' and 'x is in the attic', or three, 'x is a bat, 'x is rabid', and 'x is in the attic'? Using just one predicate will yield '$(\exists x)Ix$' where 'Ix' symbolizes 'x is a rabid bat in the attic'. If we use two predicates our symbolization might be

$$(\exists x)(Bx \ \& \ Ax),$$

where 'Bx' symbolizes 'x is a rabid bat' and 'Ax' symbolizes 'x is in the attic'. An appropriate symbolization using three predicates is

$$(\exists x)[(Rx \ \& \ Bx) \ \& \ Ax],$$

here using 'Rx' to symbolize 'x is rabid', 'Bx' to symbolize 'x is a bat', and 'Ax' to symbolize 'x is in the attic'.

When the sentence ‘There are rabid bats in the attic’ is taken in isolation, or as part of a set of symbolization exercises, all three ways of symbolizing the sentence are correct symbolizations of ‘There are rabid bats in the attic’. But there are contexts in which one symbolization is clearly preferable to the others. Consider this simple and clearly valid argument:

Rabid animals are dangerous.
There are rabid bats in the attic.

There are dangerous animals in the attic.

The symbolization key

UD:	The set of all animals
Rx:	x is rabid
Dx:	x is dangerous
Bx:	x is a rabid bat
Ax:	x is in the attic

yields an argument that is not valid in *PL*:

$(\forall x)(Rx \supset Dx)$
$(\exists x)(Bx \ \& \ Ax)$

$(\exists x)(Dx \ \& \ Ax)$

This argument is invalid because its component sentences do not reveal the connection between there being rabid bats in the attic and there being rabid animals in the attic. But if we use ‘Bx’ to symbolize ‘x is a bat’, rather than ‘x is a rabid bat’ the resulting symbolization of our argument *is* valid in *PL*:

$(\forall x)(Rx \supset Dx)$
$(\exists x)[(Rx \ \& \ Bx) \ \& \ Ax]$

$(\exists x)(Dx \ \& \ Ax)$

The lesson to be learned here is that when we are symbolizing a number of sentences and are interested in the relations among them it is advisable to select predicates that will capture as many of the connections among the English sentences as possible. But we must be careful. It is not always correct to extract two separate predicates when an adjective modifies a noun. Consider:

Sue is a ninety-eight pound gymnast.
Ed is an attractive candidate.
Stan is a meticulous accountant.

We can extract two predicates from the first of these sentences, parsing it as 'Sue is a gymnast and Sue weighs ninety-eight pounds', and symbolize it as 'Gs & Ns', using 's' to designate Sue, 'Gx' to symbolize 'x is a gymnast', and 'Nx' to symbolize 'x weighs ninety-eight pounds'. But we cannot similarly parse 'Ed is an attractive candidate', at least not in every context. Suppose Ed is running for state office and that he is fiscally conservative and a wounded war veteran. These traits may well make him an attractive candidate for state office, but they don't have anything to do with his being attractive in the sense of being a handsome man. In this case, we must treat 'is an attractive candidate' as *one* predicate. Similarly, it is probably unwise to parse 'Stan is a meticulous accountant' as 'Stan is meticulous and Stan is an accountant', for although Stan is a meticulous accountant, he may be anything but meticulous in the rest of his life.

We now turn our attention to finer issues concerning quantifiers. The syntax of *PL* requires that each variable occurring in a sentence of *PL* be bound, that is, fall within the scope of a matching quantifier. So '$(\forall x)(Fx \supset Gy)$' is a formula but not a sentence of *PL*, because 'y' is free in '$(\forall x)(Fx \supset Gy)$'. We have also seen that quantifiers can have overlapping scope. For example, we can transform '$(\forall x)(Fx \supset Gy)$' into a sentence by adding a universal y-quantifier. The three sentences we can obtain in this way are

$$(\forall x)(Fx \supset (\forall y)Gy)$$
$$(\forall x)(\forall y)(Fx \supset Gy)$$
$$(\forall y)(\forall x)(Fx \supset Gy)$$

The question now arises: are these three sentences equivalent? In subsequent chapters we will present techniques for answering this question but at present our concern is with symbolizing sentences in *PL*, and in symbolizing sentences in *PL* we need to understand the effects of placing quantifiers in different positions.

In fact, the three sentences of *PL* are equivalent. The second and third sentences are equivalent because whenever a sentence begins with multiple universal quantifiers or with multiple existential quantifiers and the rest of the sentence is in the scope of all of these quantifiers, the order in which the quantifiers appear does not matter. That is, changing the order does not change what the sentence says. So

$$(\exists x)(\exists y)(\exists z)Fxyz$$
$$(\exists y)(\exists x)(\exists z)Fxyz$$
$$(\exists z)(\exists y)(\exists x)Fxyz$$

are also equivalent sentences of *PL*, as are the results of placing the three existential quantifiers in any order. And as all of '$Fy \supset Gx$' falls within the scope of both quantifiers in

$$(\exists x)(\exists y)(Fy \supset Gx)$$

reversing the order of the quantifiers produces an equivalent sentence:

$$(\exists y)(\exists x)(Fy \supset Gx)$$

But our rule about reversing the order of initial quantifiers does not apply to '$(\exists x)[(\exists y)Fy \supset Gx]$', as this sentence does not begin with two existential quantifiers both having scope over the rest of the sentence. The scope of '$(\exists y)$' in '$(\exists x)[(\exists y)Fy \supset Gx]$' is just '$(\exists y)Fy$'.

When a sentence contains consecutive quantifiers of different types, existential and universal, such as '$(\forall x)(\exists y)$', we cannot in general change the order of those quantifiers. '$(\forall x)(\exists y)Lxy$' and '$(\exists y)(\forall x)Lxy$' *are not* equivalent sentences. Suppose we are using the set of positive integers as the UD and using 'Lxy' to symbolize 'x is less than y'. Then the first sentence says that every positive integer is less than some positive integer, while the second sentence says that there is a specific positive integer such that every positive integer is less than it.

Quantifiers can often be moved without producing nonequivalent sentences. They can, of course, only be moved if the result is *not* a formula containing a free variable. Consider the following pairs of sentences:

$Fa \;\&\; (\exists x)Gx$	$(\exists x)(Fa \;\&\; Gx)$
$Fa \;\&\; (\forall x)Gx$	$(\forall x)(Fa \;\&\; Gx)$
$Fa \vee (\exists x)Gx$	$(\exists x)(Fa \vee Gx)$
$Fa \vee (\forall x)Gx$	$(\forall x)(Fa \vee Gx)$
$Fa \supset (\exists x)Gx$	$(\exists x)(Fa \supset Gx)$
$Fa \supset (\forall x)Gx$	$(\forall x)(Fa \supset Gx)$

Careful reflection should convince the reader that all of these are pairs of equivalent sentences.

But there are two cases in which changing the scope of a quantifier requires changing the quantifier: in these cases if we broaden the scope of an existential quantifier we must replace it with a universal quantifier, and if we broaden the scope of a universal quantifier we must replace it with an existential quantifier. Here is an example of the first case:

$(\exists x)Gx \supset Fa$	$(\forall x)(Gx \supset Fa)$

These sentences are equivalent and it is fairly easy to see why they are. We discussed such a case when we symbolized 'If anyone respects Rita, Henry does'. We saw that this sentence can be correctly symbolized either as '$(\exists x)Rxr \supset Rhr$' or as '$(\forall x)(Rxr \supset Rhr)$'. Both will be true if either the UD does not contain anyone who respects Rita, or it contains at least one person who respects Rita and Henry respects Rita. So we can add '$(\exists x)Gx \supset Fa$' and '$(\forall x)(Gx \supset Fa)$' to our list of pairs of equivalent sentences.

The second case in which extending the scope of a quantifier requires changing the quantifier is illustrated by the following pair of sentences:

$$(\forall x)Gx \supset Fa \qquad\qquad (\exists x)(Gx \supset Fa)$$

It turns out, perhaps surprisingly, that these sentences are equivalent. It should be apparent that the first of these sentences is equivalent to '$\sim (\forall x)Gx \vee Fa$' on truth-functional grounds. Because '$\sim (\forall x)Gx$' is equivalent to '$(\exists x) \sim Gx$', '$\sim (\forall x)Gx \vee Fa$' is equivalent to '$(\exists x) \sim Gx \vee Fa$'. And since we can extend the scope of an existential quantifier over a wedge (providing the result *is* a sentence of *PL*), this sentence is equivalent to '$(\exists x)(\sim Gx \vee Fa)$', which, again on truth-functional grounds, is equivalent to '$(\exists x)(Gx \supset Fa)$'. So '$(\forall x)Gx \supset Fa$' and '$(\exists x)(Gx \supset Fa)$' are equivalent sentences.

The following table displays equivalent sentence forms. Here **P** is a formula containing at least one free occurrence of **x** and **Q** is a sentence of *PL* in which **x** *does not* occur.

$(\exists x)\mathbf{P} \supset \mathbf{Q}$	$(\forall x)(\mathbf{P} \supset \mathbf{Q})$
$(\forall x)\mathbf{P} \supset \mathbf{Q}$	$(\exists x)(\mathbf{P} \supset \mathbf{Q})$
$\mathbf{Q} \supset (\exists x)\mathbf{P}$	$(\exists x)(\mathbf{Q} \supset \mathbf{P})$
$\mathbf{Q} \supset (\forall x)\mathbf{P}$	$(\forall x)(\mathbf{Q} \supset \mathbf{P})$
$(\exists x)\mathbf{P} \vee \mathbf{Q}$	$(\exists x)(\mathbf{P} \vee \mathbf{Q})$
$(\forall x)\mathbf{P} \vee \mathbf{Q}$	$(\forall x)(\mathbf{P} \vee \mathbf{Q})$
$\mathbf{Q} \vee (\exists x)\mathbf{P}$	$(\exists x)(\mathbf{Q} \vee \mathbf{P})$
$\mathbf{Q} \vee (\forall x)\mathbf{P}$	$(\forall x)(\mathbf{Q} \vee \mathbf{P})$
$(\exists x)\mathbf{P} \,\&\, \mathbf{Q}$	$(\exists x)(\mathbf{P} \,\&\, \mathbf{Q})$
$(\forall x)\mathbf{P} \,\&\, \mathbf{Q}$	$(\forall x)(\mathbf{P} \,\&\, \mathbf{Q})$
$\mathbf{Q} \,\&\, (\exists x)\mathbf{P}$	$(\exists x)(\mathbf{Q} \,\&\, \mathbf{P})$
$\mathbf{Q} \,\&\, (\forall x)\mathbf{P}$	$(\forall x)(\mathbf{Q} \,\&\, \mathbf{P})$

Conspicuously absent from this table are sentence forms containing the triple bar. It turns out that in general, a sentence of the form $(\forall \mathbf{x})\mathbf{Px} \equiv \mathbf{Q}$ is equivalent neither to the corresponding sentence of the form $(\forall \mathbf{x})(\mathbf{Px} \equiv \mathbf{Q})$ nor to the corresponding sentence of the form $(\exists \mathbf{x})(\mathbf{Px} \equiv \mathbf{Q})$. Hence, the scope of a quantifier that includes only one side of a material biconditional *cannot* in general be broadened to have scope over the entire biconditional without creating a nonequivalent sentence.

We now turn to more complex symbolizations. Recall the argument we considered at the beginning of this chapter:

> None of David's friends supports Republicans. Sarah supports Breitlow and Breitlow is a Republican. So Sarah is no friend of David's.

We saw that symbolizations of this argument in *SL* are not valid. We are now in a position to provide a symbolization in *PL* that is valid. We will use the following symbolization key:

UD: The set of all people
Fxy: x is a friend of y
Sxy: x supports y
Rx: x is a Republican
d: David
b: Breitlow
s: Sarah

The second premise is readily symbolized as the conjunction 'Ssb & Rb'. The conclusion is also easy to symbolize since it simply amounts to the claim that Sarah is not a friend of David's: '~ Fsd'. The first premise, however, may pose difficulties. An appropriate paraphrase is

> It is not the case that there is an x such that [x is a friend of David's and (there is a y such that y is a Republican and x supports y)].

The expressions 'there is an x' and 'there is a y' are standing proxy for existential quantifiers. The structure of our paraphrase indicates that our symbolization will be a negation containing two existential quantifiers and two occurrences of '&'. Our symbolization mirrors the syntax of our paraphrase:

$$\sim (\exists x)[Fxd \ \& \ (\exists y)(Ry \ \& \ Sxy)]$$

This is a somewhat complicated case of a negated I-sentence. Our English argument can thus be symbolized as the following argument of *PL*:

$$\begin{array}{l} \sim (\exists x)[Fxd \ \& \ (\exists y)(Ry \ \& \ Sxy)] \\ Ssb \ \& \ Rb \\ \hline \sim Fsd \end{array}$$

The techniques presented in subsequent chapters can be used to show that this is a valid argument of *PL*.

We know that the negation of an I-sentence is equivalent to the corresponding E-sentence. This suggests that there is an alternative but equally correct symbolization of the first premise of our argument that has the form $(\forall \mathbf{x})(\mathbf{P} \supset \sim \mathbf{Q})$, and there is. We can alternatively paraphrase the argument's first premise as

> Each x is such that [if x is a friend of David's then it is not the case that there is a y such that (y is a Republican and x supports y)].

Our symbolization of this paraphrase is

$$(\forall x)[Fxd \supset \sim (\exists y)(Ry \ \& \ Sxy)]$$

Here is a somewhat more interesting, and convoluted, argument:

> Anyone who is proud of anyone is proud of Samantha. Rhoda isn't proud of anyone who's proud of him- or herself, but she is proud of everyone who has mastered calculus. Therefore if Art has mastered calculus, Samantha isn't proud of herself.

We will use the following symbolization key:

UD:	The set of students in Samantha's class
Pxy:	x is proud of y
Mx:	x has mastered calculus
a:	Art
r:	Rhoda
s:	Samantha

The first premise can be paraphrased as

> Each x is such that [if there is a y such that x is proud of y then x is proud of Samantha]

and can be symbolized as

$$(\forall x)[(\exists y)Pxy \supset Pxs]$$

The second premise of our argument is a conjunction. The first conjunct is

> Rhoda isn't proud of anyone who's proud of him- or herself.

Although the quantity expression 'anyone' does not occur at the beginning of this sentence, it is clear that the sentence is saying something about anyone who is proud of him- or herself. And 'anyone' in this sentence will go over to a universal quantifier in our symbolic sentence, for the sentence says something about *all* those individuals in Samantha's class who are proud of themselves. Our paraphrase is

> Each x is such that (if x is proud of x then it is not the case that Rhoda is proud of x).

The second conjunct of the second premise is

> she (Rhoda) is proud of everyone who has mastered calculus

This is a claim about everyone in Samantha's class who has mastered calculus. Our paraphrase is

Each x is such that (if x has mastered calculus then Rhoda is proud of x).

Our symbolization of the entire second premise is thus

$$(\forall x)(Pxx \supset \sim Prx) \ \& \ (\forall x)(Mx \supset Prx)$$

The conclusion, 'If Art has mastered calculus, Samantha isn't proud of herself' is a simple truth-functional claim and can be symbolized as

$$Ma \supset \sim Pss$$

Our complete argument of *PL* is therefore

$$\begin{array}{l} (\forall x)[(\exists y)Pxy \supset Pxs] \\ \underline{(\forall x)(Pxx \supset \sim Prx) \ \& \ (\forall x)(Mx \supset Prx)} \\ Ma \supset \sim Pss \end{array}$$

Techniques developed in subsequent chapters can be used to show that this is a valid argument of *PL*.

We will next paraphrase and symbolize a number of sentences about the positive integers. We will paraphrase each sentence before we symbolize it, and we will classify our paraphrases and symbolizations according to the Aristotelian classification system introduced at the end of the last section. But our classification is arbitrary in the sense that, as the square of opposition illustrates, an English sentence that can be symbolized as an A-sentence can also be symbolized as the negation of an O-sentence, one that can be symbolized as an E-sentence can also be symbolized as the negation of an I-sentence, and so on. Some readers will find identifying sentences in terms of the Aristotelian classification system useful; others will not. We will use the following symbolization key:

UD:	The set of positive integers
Px:	x is prime
Ex:	x is even
Ox:	x is odd
Gxy:	x is greater than y
Dxy:	x is evenly divisible by y
Sxyz:	x is the sum of y and z
Txyz:	x is the product of y and z
a:	1
b:	2

- Not every positive integer is prime.
  Negation of an A-sentence

It is not the case that each x is such that x is prime.

$\sim (\forall x)Px$

- Every prime greater than 2 is odd.
  A-sentence

  Each x is such that [(if x is prime and x is greater than 2) then x is odd].

  $(\forall x)[(Px \ \& \ Gxb) \supset Ox]$

- The sum of two primes each of which is greater than 2 is even.
  A-sentence

  Each x, each y, and each z are such that if [([(x is prime and y is prime) and (x is greater than 2 and y is greater than 2)] and z is the sum of x and y) then z is even].

  $(\forall x)(\forall y)(\forall z)[([(Px \ \& \ Py) \ \& \ (Gxb \ \& \ Gyb)] \ \& \ Szxy) \supset Ez]$

- The sum of 2 and a prime greater than 2 is odd.
  A-sentence

  Each x and each y are such that if ([(x is prime and x is greater than 2) and y is the sum of 2 and x] then y is odd).

  $(\forall x)(\forall y)([(Px \ \& \ Gxb) \ \& \ Sybx] \supset Oy)$

- No product of primes is a prime.
  Negation of an I-sentence

  It is not the case that there is an x and a y and a z such that [(x is prime and y is prime) and (z is the product of x and y and z is prime)].

  $\sim (\exists x)(\exists y)(\exists z)[(Px \ \& \ Py) \ \& \ (Tzxy \ \& \ Pz)]$

- No product of a prime and a non-prime greater than 1 is prime.
  Negation of an I-sentence

  It is not the case that there is an x and a y and a z such that [(x is prime and it is not the case that y is prime) and [(y is greater than 1 and (z is the product of x and y and z is prime)]].

  $\sim (\exists x)(\exists y)(\exists z)[(Px \ \& \sim Py) \ \& \ [Gya \ \& \ (Tzxy \ \& \ Pz)]]$

- No prime greater than 2 is evenly divisible by 2.
  Negation of an I-sentence

It is not the case that there is a y such that [(y is prime and y is greater than 2) and y is evenly divisible by 2].

~ (∃y)[(Py & Gyb) & Dyb]

- No prime greater than 2 is evenly divisible by an even number.
  E-sentence

  Each w is such that [if (w is prime and w is greater than 2) then it is not the case that there is a z such that (z is even and w is evenly divisible by z)].

  (∀w)[(Pw & Gwb) ⊃ ~ (∃z)(Ez & Dwz)].

- There are pairs of primes whose sum is prime.
  I-sentence

  There is an x and a y and a z such that [(x is prime and y is prime) and (z is the sum of x and y and z is prime)].

  (∃x)(∃y)(∃z)[(Px & Py) & (Szxy & Pz)]

The last group of sentences we symbolize are more complex than those we have so far dealt with, and the last of these is very complex. We will use the symbolization key:

UD:	The set of all books and all people
Uxy:	x understands y
Lxy:	x likes y
Axy:	x admires y
Rxy:	x reads y
Lx:	x is a logician
Px:	x is a person
p:	*Principia Mathematica*
a:	*Alice in Wonderland*
g:	*Green Eggs and Ham*

We note that the universe of discourse does not consist exclusively of people. This means that when we want to say something about people we will have to use the predicate 'Px' to distinguish them from other members of the UD.

Our first example can be symbolized as an I-sentence; our second example can be symbolized as the conjunction of two I-sentences:

- Someone understands *Principia Mathematica* and *Alice in Wonderland*.

  There is an x such that [x is a person and (x understands *Principia Mathematica* and x understands *Alice in Wonderland*)].

  (∃x)[Px & (Uxp & Uxa)]

- Someone understands *Principia Mathematica* and someone understands *Alice in Wonderland*.

  There is an x such that (x is a person and x understands *Principia Mathematica*) and there is an x such that (x is a person and x understands *Alice in Wonderland*).

  $(\exists x)(Px \ \& \ Uxp) \ \& \ (\exists x)(Px \ \& \ Uxa)$

The difference between the two sentences we have just paraphrased and symbolized is that the first says that there is some one person who understands both the books in question. The second sentence says only that there is someone who understands *Principia Mathematica* and that there is someone who understands *Alice in Wonderland*. It does not say whether these are one and the same person.

The third example can be symbolized as an A-sentence, and the fourth as the negation of an I-sentence:

- Everyone who reads *Green Eggs and Ham* both understands it and likes it.

  Each y is such that [if (y is a person and y reads *Green Eggs and Ham*) then (y is understands *Green Eggs and Ham* and y likes *Green Eggs and Ham*)].

  $(\forall y)[(Py \ \& \ Ryg) \supset (Uyg \ \& \ Lyg)]$

- No one who reads *Principia Mathematica* either understands it or likes it.

  It is not the case that there is a w such that [(w is a person and w reads *Principia Mathematica*) & (w understands *Principia Mathematica* or w likes *Principia Mathematica*)].

  $\sim (\exists w)[(Pw \ \& \ Rwp) \ \& \ (Uwp \vee Lwp)]$

Our fifth example can be symbolized as an E-sentence and our sixth example as an A-sentence:

- Anyone who reads *Green Eggs and Ham* and likes it doesn't understand anyone who reads it and doesn't like it.

  Each z is such that if ([z is a person and (z reads *Green Eggs and Ham* and z likes *Green Eggs and Ham*)] then it is not the case that there is a w such that ([(w is a person and w reads *Green Eggs and Ham*) and it is not the case that w likes *Green Eggs and Ham*] and z understands w)]).

  $(\forall z)([Pz \ \& \ (Rzg \ \& \ Lzg)] \supset \sim (\exists w)([(Pw \ \& \ Rwg) \ \& \sim Lwg] \ \& \ Uzw))$

- Anyone who understands both *Principia Mathematica* and *Alice in Wonderland* is admired by every logician.

  Each x is such that if ([x is a person and (x understands *Principia Mathematica* and x understands *Alice in Wonderland*) then each y is such that (if y is a logician then y admires x))

  $(\forall x)([Px \ \& \ (Uxp \ \& \ Uxa)] \supset (\forall y)(Ly \supset Ayx))$

Our seventh example can be paraphrased and symbolized as the negation of an I-sentence:

- No one who is not a logician understands either *Principia Mathematica* or *Alice in Wonderland*.

  It is not the case that there is an x such that [x is a person and it is not the case that x is a logician and (x understands *Principia Mathematica* or x understands *Alice in Wonderland*)].

  $\sim (\exists x)[(Px \ \& \sim Lx) \ \& \ (Uxp \vee Uxa)]$

We can symbolize our eighth example as a conjunction of an O-sentence and the negation of an O-sentence:

- There are logicians who understand but do not like *Principia Mathematica* but there are no logicians who understand but do not like *Alice in Wonderland*.

  There is a z such that [(z is a logician and z understands *Principia MathematicaI)* and it is not the case that z likes *Principia Mathematica*] and it is not the case that there is a y such that [(y is a logician and y understands *Alice in Wonderland*) and it is not the case that y likes *Alice in Wonderland*]

  $(\exists z)[Lz \ \& \ (Uzp \ \& \sim Lza)] \ \& \sim (\exists y)[Ly \ \& \ (Uya \ \& \sim Lya)]$

We can symbolize our last example as a conjunction whose left conjunct is a negation of an A-sentence and whose right conjunct is an A-sentence.

- Not everyone admires those who understand *Principia Mathematica*, but those who do also admire those who understand *Alice in Wonderland*.

  It is not the case that each w is such that [if w is a person then each z is such that (if (z is a person and z understands *Principia Mathematica*) then w admires z)] and each x is such that [if [x is a person and each y is such

that [if (y is a person and y understands *Principia Mathematica*) then x admires y]] then each z is such that [if (z is a person and z understands *Alice in Wonderland*) then x admires z]].

$\sim (\forall w)[Pw \supset (\forall z)((Pz \ \& \ Uzp) \supset Awz)] \ \& \ (\forall x)[[Px \ \& \ (\forall y)[(Py \ \& \ Uyp) \supset Axy]] \supset (\forall z)[(Pz \ \& \ Uza) \supset Pxz]]$

7.5 THE LANGUAGE *PLE* (PREDICATE LOGIC EXTENDED)

The language *PLE* is an expansion of *PL* and as such includes all the vocabulary of *PL*. In addition, *PLE* includes a two-place predicate that is defined as the identity predicate, and functors (used to express functions).

Our standard reading of 'some' is 'at least one'. Some may object that this is not an accurate reading, that 'some' sometimes means something like 'at least two'. It is alleged, for example, that to say

There are still some apples in the basket

when there is only one apple in the basket is at best misleading and at worst false. In any event we clearly do want a means of symbolizing such claims as

There are at least two apples in the basket.

We can do this by interpreting one of the two-place predicates of *PL* as expressing the identity relation. For example, we could interpret 'Ixy' as 'x is identical with y'. Given the symbolization key

UD:	The set of items in a basket of fruit
Nxy:	x is in y
Ixy:	x is identical with y
Ax:	x is an apple
b:	the basket

both

$(\exists x)(Ax \ \& \ Nxb)$

and

$(\exists x)[(Ax \ \& \ Nxb) \ \& \ (\exists y)(Ay \ \& \ Nyb)]$

say 'There is at least one apple in the basket'. The latter merely says it twice. But

$(\exists x)(\exists y)([(Ax \ \& \ Ay) \ \& \ (Nxb \ \& \ Nyb)] \ \& \sim Ixy)$

does say 'There are at least two apples in the basket'. This sentence of *PL* can be paraphrased as 'There is an x and there is a y such that ([(x is an apple and y is an apple) and (x is in the basket and y is in the basket)] and it is not the case that x is identical to y)'. This last clause is not redundant because using different variables does not commit us to there being more than one thing of the specified sort.

THE IDENTITY PREDICATE

An alternative to interpreting one of the two-place predicates of *PL* as expressing identity is to introduce a special two-place predicate and specify that it *always* be interpreted as expressing the identity relation. This is the course we shall follow. In adding this predicate to *PL*, we generate a new language, *PLE*. As an extension of *PL*, it includes all the vocabulary of *PL* and an additional two-place predicate. *PLE* also includes, as we detail later in this section, functors (used to express functions). The formulas and sentences of *PL* are also formulas and sentences of *PLE*.

The new two-place predicate that is distinctive of *PLE* is the **identity predicate,**

$=''$

When using this predicate we shall, as we have been doing with other predicates, omit the two primes as the number of individual terms used (two) will show that this is a two-place predicate. This predicate is always interpreted as the identity predicate. For example, '= ab' says that a is identical to b. However, it is customary to write, informally, 'a = b', rather than '= ab'—that is to place one individual term before the predicate and one after it—and we shall follow this custom.

So, instead of '= ab', ' = xy', and ' = aa', we write 'a = b', 'x = y', and 'a = a'. And in place of, for example, '~ = ab', we write '~ a = b'. Since the interpretation of '=' is fixed, we never have to include an interpretation of this predicate in a symbolization key.

We can now symbolize 'There are at least two apples in the basket' in *PLE*, using the preceding symbolization key (but dispensing with the now superfluous 'Ixy'), as

$$(\exists x)(\exists y)([(Ax \ \& \ Ay) \ \& \ (Nxb \ \& \ Nyb)] \ \& \sim x = y)$$

In *PLE* we can also say that there are just so many apples in the basket and no more—for example, that there is exactly one apple in the basket. An appropriate paraphrase is

> There is a y such that [(y is an apple and y is in the basket) and each thing z is such that [(if z is an apple and z is in the basket) then z is identical to y]].

A full symbolization is

$$(\exists y)[(Ay \ \& \ Nyb) \ \& \ (\forall z)[(Az \ \& \ Nzb) \supset z = y]]$$

What we are saying is that there is at least one apple in the basket and that anything that is an apple and is in the basket is *that very apple*.

Consider next

Henry hasn't read *Alice in Wonderland* but everyone else in the class has.

If we limit our universe of discourse to the students in the class in question, let 'h' designate Henry, and interpret 'Ax' as 'x has read *Alice in Wonderland*', we can symbolize this claim as

$$\sim Ah \;\&\; (\forall y)[\sim y = h \supset Ay]$$

And, using 'b' to designate Bob, we can symbolize 'Only Henry and Bob have not read *Alice in Wonderland*', as

$$\sim (Ah \vee Ab) \;\&\; (\forall x)[\sim (x = h \vee x = b) \supset Ax]$$

This says that neither Henry nor Bob has read *Alice in Wonderland* and that everyone else—that is, each person in the class who is neither identical to Henry nor identical to Bob—has read it.

We can also use the identity predicate to symbolize the following sentences of *PLE*:

1. There are apples and pears in the basket.
2. The only pear in the basket is rotten.
3. There are at least two apples in the basket.
4. There are two (and only two) apples in the basket.
5. There are no more than two pears in the basket.
6. There are at least three apples in the basket.

UD:	The set of items in a fruit bowl
Ax:	x is an apple
Nxy:	x is in y
Px:	x is a pear
Rx:	x is rotten
b:	the basket

We can recast sentence 1 as 'There is at least one apple and at least one pear in the basket', and symbolize it without using the identity predicate:

$$(\exists x)(\exists y)[(Ax \;\&\; Py) \;\&\; (Nxb \;\&\; Nyb)]$$

However, if we take sentence 1 to assert that there are at least two apples and at least two pears in the basket, we do need the identity predicate:

$$(\exists x)(\exists y)[((Ax \;\&\; Ay) \;\&\; (Nxb \;\&\; Nyb)) \;\&\; \sim x = y] \;\&$$
$$(\exists x)(\exists y)[((Px \;\&\; Py) \;\&\; (Nxb \;\&\; Nyb)) \;\&\; \sim x = y]$$

Sentence 2 says that there is one and only one pear in the basket and that that one pear is rotten:

$$(\exists x)[((Px \ \& \ Nxb) \ \& \ Rx) \ \& \ (\forall y)[(Py \ \& \ Nyb) \supset y = x]]$$

Sentence 3 says only that there are *at least* two apples in the basket, not that there are *exactly* two. Hence

$$(\exists x)(\exists y)[((Ax \ \& \ Ay) \ \& \ (Nxb \ \& \ Nyb)) \ \& \sim x = y]$$

To symbolize sentence 4 we start with the symbolization for sentence 3 and add a clause saying there are no additional apples in the basket:

$$(\exists x)(\exists y)([((Ax \ \& \ Ay) \ \& \ (Nxb \ \& \ Nyb)) \ \& \sim x = y] \ \&$$
$$(\forall z)[(Az \ \& \ Nzb) \supset (z = x \vee z = y)])$$

The added clause says, in effect, 'and anything that is an apple and is in the basket is either x or y'. Sentence 5 does not say that there are two pears in the basket; rather, it says that there are *at most* two pears in the basket. We can express this in *PLE* by saying that of any pears, x, y, and z that are in the basket these are really at most two; that is, either x is identical to y, or x is identical to z, or y is identical to z. In other words

$$(\forall x)(\forall y)(\forall z)[([(Px \ \& \ Py) \ \& \ Pz] \ \& \ [(Nxb \ \& \ Nyb) \ \& \ Nzb]) \supset$$
$$((x = y \vee x = z) \vee y = z)]$$

Finally sentence 6 can be symbolized by building on the symbolization for sentence 3:

$$(\exists x)(\exists y)(\exists z)(([(Ax \ \& \ Ay) \ \& \ Az] \ \& \ [(Nxb \ \& \ Nyb) \ \& \ Nzb]) \ \&$$
$$[(\sim x = y \ \& \sim y = z) \ \& \sim x = z)]$$

We now return to our discussion of positive integers. This time we will use this symbolization key for the sentences that follow.

UD:	The set of positive integers
Bxyz:	x is between y and z
Lxy:	x is larger than y
Sxy:	x is a successor of y
Ex:	x is even
Px:	x is prime
a:	1
b:	2
c:	10
d:	14

1. There is no largest positive integer.
2. There is a unique smallest positive integer.
3. 2 is the only even prime.
4. Every positive integer has exactly one successor.
5. 2 is the only prime whose successor is prime.

As we saw in our earlier discussion, we can symbolize sentence 1 without using the identity predicate, for to say that there is no largest positive integer it suffices to say that for every integer there is a larger integer (no matter what integer one might pick, there is an integer larger than it):

$$(\forall x)(\exists y)Lyx$$

It is also tempting to symbolize sentence 2 without using the identity predicate, for to say that there is a smallest positive integer seems to be to say that there is an integer that is not larger than any integer:

$$(\exists x) \sim (\exists y)Lxy$$

But while the foregoing does say that there is a smallest positive integer, it does not say that there is a unique such integer. So a better symbolization is

$$(\exists x)(\forall y)(\sim y = x \supset Lyx)$$

This sentence of *PL* says that there is an integer such that every integer not identical to it is larger than it. This does imply uniqueness.

Sentence 3, '2 is the only even prime', says that 2 is prime and is even and that all other primes are not even:

> 2 is prime and 2 is even, and each z is such that if z is prime and z is not identical with 2 then z is not even.

In *PLE*

$$(Pb \,\&\, Eb) \,\&\, (\forall z)[(Pz \,\&\, \sim z = b) \supset \sim Ez]$$

This is equivalent to

$$(Pb \,\&\, Eb) \,\&\, (\forall z)[(Pz \,\&\, Ez) \supset z = b]$$

Notice that we could equally well have paraphrased and symbolized sentence 3 as

> 2 is prime and 2 is even, and it is not the case that there is a z such that z is prime and z is even, and z is not identical with 2

and symbolized this claim as

$$(Pb \,\&\, Eb) \,\&\, \sim (\exists z)[(Pz \,\&\, Ez) \,\&\, \sim z = b]$$

Notice, too, that all three symbolic versions of sentence 3 are truth-functional compounds, not quantified sentences.

Sentence 4, 'Every positive integer has exactly one successor', can be symbolized as

$$(\forall x)(\exists y)[Syx \,\&\, (\forall z)(Szx \supset z = y)]$$

This says that each positive integer x has a successor y and that any integer that is a successor of x is identical to y—that is, that each positive integer has exactly one successor.

Sentence 5, '2 is the only prime whose (only) successor is prime', can be paraphrased as a conjunction:

> (2 is prime and there is an x such that [(x is the successor of 2 and each y is such that (if y is the successor of 2 then y = x)) and x is prime]) and each x and each y are such that [(if x is the successor of y and (y is prime and it is not the case that y = b)) then it is not the case that x is prime]

The first conjunct can be symbolized as

$$Pb \,\&\, (\exists x)[(Sxb \,\&\, (\forall y)(Syb \supset y = x)) \,\&\, Px]$$

The second conjunct can be symbolized as

$$(\forall x)(\forall y)[(Sxy \,\&\, (Py \,\&\, \sim y = b)) \supset \sim Px]$$

Putting these together we obtain

$$(Pb \,\&\, (\exists x)[(Sxb \,\&\, (\forall y)(Syb \supset y = x)) \,\&\, Px]) \,\&\, (\forall x)(\forall y)$$
$$[(Sxy \,\&\, (Py \,\&\, \sim y = b)) \supset \sim Px]$$

DEFINITE DESCRIPTIONS

In Section 7.1 we discussed three kinds of singular terms of English: proper names, pronouns, and **definite descriptions.** We subsequently noted that individual constants of *PL* can be used analogously to singular terms of English that do refer. But following this practice means that the internal structure of definite descriptions is not represented in *PL*. Consider, by way of illustration, this argument:

> The Roman general who defeated Pompey invaded both Gaul and Germany. Therefore Pompey was defeated by someone who invaded both Gaul and Germany.

This is fairly obviously a valid argument. But its symbolization in *PL* is not valid:

UD:	The set of persons and countries
Ixy:	x invaded y
Dxy:	x defeated y
r:	The Roman general who defeated Pompey
p:	Pompey
g:	Gaul
e:	Germany

Treating 'The Roman general who defeated Pompey' as an unanalyzable unit, to be symbolized by 'r,' and paraphrasing the conclusion as 'There is an x such that [x defeated Pompey and (x invaded Gaul and x invaded Germany)]' yields the following symbolization:

$$\frac{\text{Irg} \;\&\; \text{Ire}}{(\exists x)[\text{Dxp} \;\&\; (\text{Ixg} \;\&\; \text{Ixe})]}$$

The techniques we develop for testing arguments of *PL* will show that this argument of *PL* is invalid. This should not be surprising, for the premise tells us only that the thing designated by 'r' invaded both Gaul and Germany; it does not tell us that that thing is a thing that defeated Pompey, as the conclusion claims.

By using the identity predicate we can capture the structure of definite descriptions within *PLE*. Suppose we paraphrase the first premise of the preceding argument as

> There is an x such that [[(x is a Roman general and x defeated Pompey) and each y is such that [if (y is a Roman general and y defeated Pompey) then y = x]] and (x invaded Gaul and x invaded Germany)].

Definite descriptions are, after all, descriptions that purport to specify conditions that are satisfied by exactly one thing. Using our current symbolization key, plus 'Rx' for 'x is a Roman general', we can symbolize the first premise as

$$(\exists x)[[(\text{Rx} \;\&\; \text{Dxp}) \;\&\; (\forall y)[(\text{Ry} \;\&\; \text{Dyp}) \supset y = x]] \;\&\; (\text{Ixg} \;\&\; \text{Ixe})]$$

We shall later show that in *PLE* the conclusion '$(\exists x)[\text{Dxp} \;\&\; (\text{Ixg} \;\&\; \text{Ixe})]$' does follow from this premise.

By transforming definite descriptions into unique existence claims, that is, claims that there is exactly one object of such-and-such a sort, we gain the further benefit of being able to symbolize English language definite descriptions that may, in fact, not designate anything. For example, taking the UD to be persons and using 'Dxy' for 'x is a daughter of y', 'Bx' for 'x is a biochemist', and 'j' to designate John, we might symbolize 'John's only daughter is a biochemist' as

$$(\exists x)[(\text{Dxj} \;\&\; (\forall y)(\text{Dyj} \supset y = x)) \;\&\; \text{Bx}]$$

If it turns out that John has no, or more than one, daughter, or that his only daughter is not a biochemist, the above sentence of *PLE* will be false, not meaningless or truth-valueless. This is an acceptable result.

PROPERTIES OF RELATIONS

Identity is a relation with three rather special properties. First, identity is a **transitive** relation. That is, if an object x is identical with an object y, and y is identical with an object z, then x is identical with z. The following sentence of *PLE* says, in effect, that identity is transitive:

$$(\forall x)(\forall y)(\forall z)[(x = y \ \& \ y = z) \supset x = z]$$

Many relations other than identity are also transitive relations. The predicates

x is larger than y
x is taller than y
x is an ancestor of y
x is heavier than y
x occurs before y

all express transitive relations. But, 'x is a friend of y' does not represent a transitive relation. That is, 'Any friend of a friend of mine is a friend of mine' is a substantive claim, and one that is generally false. Where x, y, and z are all variables of *PL* or *PLE* and A is a two-place predicate of *PL* or *PLE*, the following says that A expresses a transitive relation:

$$(\forall x)(\forall y)(\forall z)[(Axy \ \& \ Ayz) \supset Axz]$$

Identity is also a **symmetric** relation; that is, if an object x is identical with an object y, then y is identical with x. The following says that A is a symmetric relation:

$$(\forall x)(\forall y)(Axy \supset Ayx)$$

The following predicates also express symmetric relations:

x is a sibling of y
x is a classmate of y
x is a relative of y
x has the same father as does y

Note that neither 'x is a sister of y' nor 'x loves y' expresses a symmetric relation. Jane Fonda is a sister of Peter Fonda, but Peter Fonda is not a sister of

Jane Fonda. And, alas, it may be that Manfred loves Hildegard even though Hildegard does not love Manfred.

A relation is **reflexive** if and only if each object stands in that relation to itself. In *PL* and *PLE* the following says that A expresses a reflexive relation:

$(\forall x)Axx$

Identity is a reflexive relation. In an unrestricted UD it is rather hard to find other reflexive relations. For example, a little thought should show that none of the following expresses a reflexive relation in an unrestricted universe of discourse:

x is the same age as y
x is the same height as y
x is in the same place as y

Since the number 48 is not of any age, it is not the same age as itself nor the same height as itself. Numbers have neither age nor height, though inscriptions of numerals usually have both. So, too, neither the number 93 nor the set of human beings is in any place. Numbers and sets do not have spatial positions; hence neither is in the same place as itself. However, the relations just discussed are reflexive relations in suitably restricted universes of discourse. For example, if the universe of discourse consists exclusively of people, then

x is the same age as y

expresses a reflexive relation (it is also transitive and symmetric). Every person is the same age as him- or herself. In this restricted universe 'x is the same height as y' and 'x is in the same place as y' also represent reflexive relations. Each person is the same height as him- or herself and is in the same place as him- or herself. And, if the universe of discourse is restricted to the positive integers, then

x is evenly divisible by y

expresses a reflexive relation, for every positive integer is evenly divisible by itself. This relation is not symmetric (not every positive integer evenly divides all the positive integers it is evenly divisible by). However, 'x is evenly divisible by y' does express a transitive relation.

FUNCTIONS

A **function** is an operation that takes one or more element of a set as arguments and returns a single value. *Addition, subtraction, multiplication, square,* and *successor* are all common functions of arithmetic. Each returns, for each number or pair of numbers, a single value. *Addition* takes a pair of numbers

as arguments and returns their sum; *multiplication* takes a pair of numbers and returns the product of those numbers; *subtraction* returns, for each pair of numbers, the first number minus the second. The *square* function returns, for each number, the result of multiplying that number by itself; the *successor* function returns, for any positive integer **n**, the integer **n** + 1.

Not all functions are arithmetic functions. We have already encountered truth-functions—functions that map values from the set consisting of the truth-values (the set {**T**, **F**}) to truth-values. *Negation* is a function of one argument that returns **F** when given **T** as an argument and returns **T** when given **F** as an argument. *Conjunction, disjunction,* the *material conditional,* and the *material biconditional* are all functions that take two arguments (two truth-values) and return a single truth-value. Characteristic truth-tables display the value of each of these functions for each pair of truth-values.

Functions are also found outside of formal logic and mathematics. Consider a set of monogamously married individuals.[7] Here *spouse* is a function that takes a single member of the set as an argument and returns that person's spouse as its value. For the set of all twins, the function *twin* returns, for each member of the set, that member's twin. In *PLE* we shall use lowercase italicized Roman letters *a*–*z*, with or without a positive-integer subscript, followed by one or more prime marks to symbolize functions. We call these symbols **functors.** Where **n** is the number of prime marks after the functor, the function assigned to the functor takes **n** arguments. For example, in talking about the set of positive integers, we might assign the successor function to the functor *f*.[8] We specify this assignment in a symbolization key much the way we have been assigning interpretations to predicates. The following symbolization key assigns the successor function to f':

UD:	The set of positive integers
f'(x):	the successor of x
Ex:	x is even
Ox:	x is odd
a:	2
b:	3

The variable x in parentheses indicates that we are assigning to f' a function that takes a single argument. The expression to the right of the colon assigns the successor function to f'. Given the above symbolization key,

Ob

says 3 is odd. The sentence

Of'(a)

[7]The example is from Geoffrey Hunter, *Metalogic: An Introduction to the Metatheory of Standard First Order Logic*, Paperback ed. (Berkeley: University of California Press, 1996).

[8]It is customary to use, where only a few functors are needed, the letters 'f'', 'g', 'h', . . . We will follow this custom.

says the successor of 2, which is 3, is odd. Both claims are, of course, true. And

$$f'(a) = b$$

says the successor of 2 is 3, which it is. Similarly,

$$(\exists x)Of'(x) \ \& \ (\exists x)Ef'(x)$$

says there is a positive integer whose successor is odd and there is a positive integer whose successor is even. We can also use the symbolization key to symbolize 'The successor of an even number is odd'. A first step is the quasi-English

$$(\forall x)(Ex \supset \text{the successor of x is odd})$$

The successor of x is $f'(x)$, so the full symbolization is

$$(\forall x)(Ex \supset Of'(x))$$

We can add the following to our symbolization key

$$h''(x,y): \quad \text{the sum of x and y}$$

and symbolize 'The sum of an even number and an odd number is odd' as

$$(\forall x)(\forall y)[(Ex \ \& \ Oy) \supset Oh''(x,y)]$$

Since the number of distinct individual terms occurring within the parentheses after a functor indicates how many arguments the function assigned to that functor takes, we can informally omit the primes that officially follow functors, just as we do for predicate letters. Hereafter we will do so.

Returning to our example of the set of twins, we can use the following symbolization key

UD: The set of all twins
$f(x)$: the twin of x
c: Cathy
h: Henry
j: Jose
s: Simone

to symbolize

Simone is Henry's twin

as

$$s = f(h)$$

and

Jose is Cathy's twin

as

$$j = f(c)$$

Using 'Bx' for 'x is bald', we can symbolize 'A twin is bald if and only if her or his twin is bald' as

$$(\forall x)[Bx \equiv Bf(x)]$$

and 'Some bald twins have twins that are not bald' as

$$(\exists x)[Bx \ \& \sim Bf(x)]$$

The symbolization

$$(\forall x)(\forall y)[(\exists z)(z = f(x) \ \& \ z = f(y)) \supset x = y]$$

says, in quasi-English, 'Any members of the UD x and y who are such that if there is a z who is both a twin of x and a twin of y then x and y are identical', or 'No one is a twin of two different twins'.

We require that the functions we symbolize with functors have the following characteristics:

1. An **n**-place function must yield one and only one value for each **n**-tuple of arguments.[9]
2. The value of a function for an **n**-tuple of members of a UD must be a member of that UD.

If the UD is the set of integers, the square root operation does not meet condition 1 because it can yield more than one value for its arguments (there are two square roots of 4, 2 and −2.). (It also fails to meet condition 2 because not all square roots of integers are integers.) If the UD is the set of positive integers, the subtraction function does not meet condition 2, because when y is greater than x, x minus y yields a value that is not a positive integer (3 minus 9 is −6, and −6 is not a positive integer). Subtraction *does* meet condition 2 when the UD is the set of all integers—positive, zero, and negative. If the UD is the set of positive integers, division also fails to meet condition 2 (3 divided by 9 yields 1/3, which is not a positive integer). Division *does* meet condition 2 when the UD is the set of positive rational numbers (positive integers plus numbers expressible as the ratio between positive integers). Finally division does not meet condition 1 when the UD is the set of *all* integers because it is undefined when the divisor is zero.

[9]An **n**-tuple is an ordered set containing **n** members.

As we have just seen, functors can be used to generate a new kind of individual term (in addition to the individual constants and variables of *PL*). We call these new terms **complex terms.** Complex terms are of the form

$$\boldsymbol{f}(\mathbf{t}_1, \mathbf{t}_2, \ldots \mathbf{t}_n)$$

where $\boldsymbol{f}$ is an **n**-place functor and $\mathbf{t}_1, \mathbf{t}_2, \ldots \mathbf{t}_n$ are individual terms. Further examples of complex terms include

f(a,b)
h(a,b,c)
g(a)
f(b,b)
f(x,y)
f(a,y)
f(y,a)
g(x)
f(g(a),b)
f(a,g(x))

Complex terms are complex in that they are always formed from a functor and at least one individual term. Some complex terms contain variables, and some do not. We call individual terms that do not contain variables **closed terms,** and those that do **open terms.** This makes *both* individual constants and complex terms that contain no variables closed terms. Complex terms that do contain at least one variable, as well as variables themselves, are open terms. Individual terms that are not complex terms (the individual constants and individual variables) are **simple individual terms.** In the above list, the first four complex terms are closed, the next four open, the ninth closed, and the last open. Note the last two examples. In each, one of the individual terms from which the example is built is itself a complex term. This is wholly in order, as complex terms are individual terms and can occur anywhere a constant can occur. The kinds of individual terms included in *PLE* are summarized in the following table:

INDIVIDUAL TERMS OF *PLE*

	Open	*Closed*
Simple	Individual variables	Individual constants
Complex	Individual term formed from a functor and *at least one* individual variable—for example, f(x), f(a,x), g(f(a),y), g(h(x,y),a)	Individual term formed from a functor and containing *no* individual variable—for example, f(a), g(a,b), f(g(a,f(a,c)))

All of the following are formulas of *PLE*:

$Fa\mathit{f}(x)$
$F\mathit{f}(x)a$
$F\mathit{f}(a)b$
$(\forall x)Fa\mathit{f}(x)$
$(\forall x)(\exists y)Fx\mathit{f}(y)$

In each of these examples 'F' is a two-place predicate. The first and second are formulas of *PLE* but are not sentences (because the x in '$\mathit{f}(x)$' is not bound). The third, fourth, and fifth examples are all both formulas and sentences of *PLE*. The third says that $\mathit{f}(a)$ bears the relation F to b. The fourth says that each thing x in the UD is such that a bears the relation F to $\mathit{f}(x)$, that is, to the value of the function *f* as applied to x. The fifth says that each thing x in the UD is such that there is a thing y such that x bears the relation F to $\mathit{f}(y)$. Every example contains a complex individual term, and all but the third an open complex individual term.

Consider this symbolization key:

UD:	The set of positive integers
Ox:	x is odd
Ex:	x is even
Px:	x is prime
Gxy:	x is greater than y
$h(x,y)$:	the sum of x and y
$\mathit{f}(x)$:	the successor of x
a:	1
b:	2

The sentence

$$(\forall x)[Ex \supset O\mathit{f}(x)]$$

says, truly, that each positive integer is such that if it is even then its successor is odd. And

$$(\forall x)[Ex \supset E\mathit{f}(\mathit{f}(x))]$$

says, truly, that each positive integer is such that if it is even then the successor of its successor is also even. The sentence

$$(\forall x)(\forall y)[(Ex \,\&\, Ey) \supset Eh(x,y)]$$

can be read in quasi-English as '<u>Each x and each y are such that</u> [<u>if</u> (x is even <u>and</u> y is even) <u>then</u> the sum of x and y is even]. This is, of course, true.

Here are further sentences of *PLE* that can be read in English using the above symbolization key. The sentence

$$(\forall x)(\forall y)[Gh(x,y)x \ \& \ Gh(x,y)y]$$

says that for any positive integers x and y the sum of x and y is greater than x, and the sum of x and y is greater than y. This is true. The sentence

$$(\exists x)Gxh(a,b)$$

says that there is a positive integer, x, that is greater than the sum of 1 and 2—that is, there is a positive integer that is greater than 3. This is also true. The sentence

$$(\forall x)(\forall y)[(Ex \ \& \ Oy) \supset Oh(x,y)]$$

says that, for any pair of positive integers x and y, if the first is even and the second is odd, then their sum is odd. This is true as well. Finally the sentence

$$(\forall x)(\forall y)[Ph(x,y) \supset \sim (Px \ \& \ Py)]$$

says that, for any pair of positive integers, if their sum is prime then it is not the case that they are both prime, or, in other words, that there are no prime numbers x and y such that their sum is also prime. This sentence is false; 2 and 3 are both prime, and so is their sum, 5.

THE SYNTAX OF PLE

In addition to the vocabulary of *PL*, the vocabulary of *PLE* also includes

> $=''$: The two-place identity predicate (fixed interpretation)
>
> *Functors of PLE:* Lowercase italicized Roman letters *a*, *b*, *c*, . . . , with or without a numeric subscript, followed by **n** primes.
>
> *Individual terms of PLE:*
>
> > Individual constants are individual terms of *PLE*
> >
> > Individual variables are individual terms of *PLE*
> >
> > Expressions of the form $\boldsymbol{f}(\mathbf{t}_1, \mathbf{t}_2, \ldots \mathbf{t}_n)$, where $\boldsymbol{f}$ is an **n**-place functor and $\mathbf{t}_1, \mathbf{t}_2, \ldots, \mathbf{t}_n$ are individual terms of *PLE*

We can classify the individual terms of *PLE* as follows:

> *Simple terms of PLE:* The individual constants and individual variables of *PLE*

Complex terms of PLE: Individual terms of the form $\boldsymbol{f}(\mathbf{t}_1, \mathbf{t}_2, \ldots, \mathbf{t}_n)$, where $\boldsymbol{f}$ is an **n**-place functor

Closed individual term: An individual term in which no variable occurs

Open individual term: An individual term in which at least one variable occurs

Individual variables and functors that contain at least one individual variable are thus open terms. Individual constants and functors that contain no variables are thus closed terms.

In *PLE* a substitution instance is defined as follows:

Substitution instance of **P**: If **P** is a sentence of *PLE* of the form $(\forall \mathbf{x})\mathbf{Q}$ or $(\exists \mathbf{x})\mathbf{Q}$ and **t** is a closed individual term, then $\mathbf{Q}(\mathbf{t}/\mathbf{x})$ is a substitution instance of **P**. The individual term **t** is the *instantiating individual term.*

Note that every substitution instance of a sentence of *PL* is also a substitution instance of that same sentence in *PLE.*

7.5E EXERCISES

1. Symbolize the following sentences in *PLE* using the following symbolization key:

UD:	The set of all people
Sx:	x is a sailor
Lx:	x is lucky
Cx:	x is careless
Yx:	x dies young
Sxy:	x is a son of y
Dxy:	x is a daughter of y
Wx:	x is a Wilcox
d:	Daniel Wilcox
j:	Jacob Wilcox
r:	Rebecca Wilcox

a. Every Wilcox except Daniel is a sailor.
*b. Every Wilcox except Daniel is the offspring of a sailor.
c. Every Wilcox except Daniel is either a sailor or the offspring of sailor.
*d. Daniel is the only son of Jacob.
e. Daniel is the only child of Jacob.
*f. All the Wilcoxes except Daniel are sailors.
g. Rebecca's only son is Jacob's only son.
*h. Rebecca Wilcox has only one son who is a sailor.
i. Rebecca Wilcox has at least two daughters who are sailors.

*j. There are two and only two sailors in the Wilcox family.
k. Jacob Wilcox has one son and two daughters, and they are all sailors.

2. Give fluent English readings for the following sentences of *PLE* using the given symbolization key.

UD:	The set of positive integers
Lxy:	x is less than y
Gxy:	x is greater than y
Ex:	x is even
Ox:	x is odd
Px:	x is prime
f(x,y):	the product of x and y
t:	2
f:	5
n:	9

a. $(\forall x)(\exists y)Lxy$
*b. $(\exists x)(\forall y)(\sim x = y \supset Lxy)$
c. $(\exists x)(\forall y) \sim Lyx$
*d. $\sim (\exists x)(Ex \ \& \ Lxt)$
e. $(Pt \ \& \ Et) \ \& \ (\forall x)[(Px \ \& \ Ex) \supset x = t]$
*f. $\sim (\exists x)(\exists y)[(Px \ \& \ Py) \ \& \ Pf(x,y)]$
g. $(\forall y)(\forall z)[(Oy \ \& \ Oz) \supset Of(y,z)]$
*h. $(\forall y)(\forall z)[(Ey \ \& \ Ez) \supset Ef(z,y)]$
i. $(\forall y)(\forall z)[(Ey \vee Ez) \supset Ef(y,z)]$
*j. $(\forall x)[Ex \supset (\exists y)(Oy \ \& \ Gxy)] \ \& \sim (\forall x)[Ox \supset (\exists y)(Ey \ \& \ Gxy)]$
k. $(\exists x)[[Px \ \& \ (Gxf \ \& \ Lxn)] \ \& \ (\forall y)([Py \ \& \ (Gyf \ \& \ Lyn)] \supset y = x)]$

3. For a–p, decide whether the specified relation is reflexive, whether it is symmetric, and whether it is transitive (in suitably restricted universes of discourse). In each case give the sentences of *PL* that assert the appropriate properties of the relation in question. If the relation is reflexive, symmetric, or transitive only in a restricted universe of discourse, specify such a universe of discourse.

a. Nxy: x is a neighbor of y
*b. Mxy: x is married to y
c. Axy: x admires y
*d. Nxy: x is north of y
e. Rxy: x is a relative of y
*f. Sxy: x is the same size as y
g. Txy: x is at least as tall as y
*h. Cxy: x coauthors a book with y
i. Exy: x enrolls in the same course as y
*j. Fxy: x fights y
k. Wxy: x weighs the same as y
*l. Cxy: x contracts with y
m. Axy: x is an ancestor of y
*n. Cxy: x is a cousin of y
o. Lxy: x and y have the same taste in food
*p. Rxy: x respects y

4. Symbolize the following sentences in *PLE* using the given symbolization key.

UD:	The set of people in Doreen's hometown
Dxy:	x is a daughter of y
Sxy:	x is a son of y
Bxy:	x is a brother of y
Oxy:	x is older than y
Mxy:	x is married to y
Txy:	x is taller than y
Px:	x is a physician
Bx:	x is a baseball player
Mx:	x is a marine biologist
d:	Doreen
c:	Cory
j:	Jeremy
h:	Hal

a. Jeremy is Cory's son.
*b. Jeremy is Cory's only son.
c. Jeremy is Cory's oldest son.
*d. Doreen's only daughter is a physician.
e. Doreen's eldest daughter is a physician.
*f. Doreen is a physician and so is her eldest daughter.
g. Cory is Doreen's eldest daughter.
*h. Cory is married to Hal's only son.
i. Cory is married to Hal's tallest son.
*j. Doreen's eldest daughter is married to Hal's only son.
k. The only baseball player in town is the only marine biologist in town.
*l. The only baseball player in town is married to one of Jeremy's daughters.
m. Cory's husband is Jeremy's only brother.

5. Symbolize the following sentences in *PLE* using the given symbolization key.

UD:	The set of positive integers
Ox:	x is odd
Ex:	x is even
Px:	x is prime
a:	1
b:	2
f(x):	the successor of x
q(x):	x squared
t(x,y):	the product of x and y
s(x,y):	the sum of x and y

a. One is not the successor of any integer.
*b. One is not prime but its successor is.
c. There is a prime that is even.
*d. There is one and only one even prime.
e. Every integer has a successor.

*f. The square of a prime is not prime.
g. The successor of an odd integer is even.
*h. The successor of an even integer is odd.
i. If the product of a pair of positive integers is odd, then the product of the successors of those integers is even.
*j. If the product of a pair of positive integers is even, then one of those integers is even.
k. If the sum of a pair of positive integers is odd, then one member of the pair is odd and the other member is even.
*l. If the sum of a pair of positive integers is even, then either both members of the pair are even or both members are odd.
m. The product of a pair of prime integers is not prime.
*n. There are no primes such that their product is prime.
o. The square of an even number is even and the square of an odd number is odd.
*p. The successor of the square of an even number is odd.
q. The successor of the square of an odd number is even.
*r. 2 is the only even prime.
s. The sum of 2 and a prime other than 2 is odd.
*t. There is exactly one integer that is prime and is the successor of a prime.
u. There is a pair of primes such that their product is the successor of their sum.

Chapter 8

PREDICATE LOGIC: SEMANTICS

Section 8.1 of this chapter introduces interpretations, which are the foundation for the quantificational semantics for *PL*. Sections 8.2 through 8.4 present the quantificational versions of the core logical concepts: quantificational truth, falsehood, and indeterminacy, quantificational equivalence, quantificational consistency, and quantificational entailment validity. Section 8.5 presents truth-functional expansions, which can sometimes be used to reason about quantificational semantics. Section 8.6 augments *PL*'s quantificational semantics for *PLE,* the language of predicate logic with identity and functors.

8.1 INTERPRETATIONS

The basic semantic concept for the language of sentential logic, *SL*, is that of a truth-value assignment. The semantics for *PL* is more complex than truth-functional semantics. One source of the added complexity is this: Whereas atomic sentences of *SL* are not analyzable in terms of more basic linguistic expressions of *SL*, this is not true of all of the atomic sentences of *PL*. Atomic sentences such as 'Fa' are complex expressions composed of predicates and individual constants. Consequently, we directly assign

truth-values only to the atomic sentences of *PL* that are sentence letters. The truth-values of complex atomic sentences like 'Fa' will be determined by the interpretations of the predicates and individual constants that constitute such sentences.

The basic semantic concept of *PL*, in terms of which other semantic concepts are defined, is that of an **interpretation.** Just as truth-value assignments for *SL* assign truth-values to every sentence of *SL*, an interpretation for *PL* interprets every individual constant, predicate, and sentence letter of *PL*. There is a sense in which the symbolization keys presented in Chapter 7 provide interpretations for the symbolic sentences of *PL*, for the truth-conditions of sentences of *PL* depend on the universe of discourse we choose, the interpretations that we provide for the predicates and individual constants that occur in the sentences, and the truth-values that we assign to the sentence letters that occur in those sentences. Consider, for example, the symbolization key

UD:	The set {1, 2, 3, 4, 5}
Ex:	x is even
Gxy:	x is greater than or equal to y
Pxyz:	the sum of x, y, and z is odd
a:	1
b:	2
c:	3
d:	4
e:	5

Given this symbolization key,

- 'Eb' and 'Ed' are true because 2 and 4 are even
- 'Ea', 'Ec', and 'Ee' are false because 1, 3, and 5 are not even
- 'Gca' is true because 3 is greater than or equal to 1
- 'Gbe' is false because 2 is not greater than or equal to 5
- 'Pace' and 'Pabd' are true because the sum of 1, 3, and 5 is an odd number (9), and the sum of 1, 2, and 4 is an odd number (7)
- 'Pabc' is false because the sum of 1, 2, and 3 (6) is not an odd number.

However, the symbolization key provides more information than is needed for an actual *interpretation*. When we interpret predicates, we do not need to know what they mean, but only what they do or do not apply to. To understand this, consider the following symbolization key:

UD:	The set {1, 2, 3, 4, 5}
Ex:	x is even
Gxy:	x is greater than or equal to y
Pxyz:	either x, y, and z are all odd or exactly one of x, y, and z is odd

a: 1
b: 2
c: 3
d: 4
e: 5

Given this symbolization key, 'Pace' and 'Pabd' are still true, because 1, 3, and 5 are all odd and exactly one of the integers 1, 2, and 4 is odd. Moreover, 'Pabc' is still false because 1, 2, and 3 are not all odd, nor is it the case that exactly one of these integers is odd. In fact, no matter which combination of three (not necessarily distinct) individual constants we write after 'P', the atomic sentence that is formed will have the same truth-value with respect to both symbolization keys. The reason is that the sum of 3 positive integers is odd if and only if either exactly one or all three of the positive integers are themselves odd. Thus, given either symbolization key, 'P' applies to, or is true of, the same integers. Since the purpose of our interpretations is to provide truth-conditions for sentences of *PL*, our interpretations only need to specify the things of which the predicates are true, rather than provide English-language readings for the predicates. (Similarly, to provide truth-conditions for sentences of *SL*, we only needed to assign truth-values to the language's atomic sentences rather than provide English readings of those sentences).

To this end, we need the concept of an **n-tuple**, which is an ordered set containing **n** members. An **n**-tuple is ordered in the sense that one member is designated as the first member, another as the second (assuming that $\mathbf{n} > 1$), and so on. Moreover, because each member is associated with a position, the same item can occur in more than one position in a single **n**-tuple. We designate an **n**-tuple by listing its members, in the order in which they occur in the **n**-tuple, between the angle brackets '<' and '>'. For example, we designate the 3-tuple whose members are 1, 2, and 3 *in that order* as

$$<1, 2, 3>.$$

Because **n**-tuples are ordered, this 3-tuple is distinct from all of the following 3-tuples (which are also distinct from one another):

$$<1, 3, 2>, <2, 1, 3>, <2, 3, 1>, <3, 1, 2>, <3, 2, 1>.$$

Here are examples of 3-tuples in which the same integer occurs more than once:

$$<3, 3, 4>, <3, 4, 3>, <4, 3, 3>, <3, 3, 3>.$$

To interpret an **n**-place predicate, we will assign a set of **n**-tuples to that predicate. We call this set the '**extension**' of the predicate. For example, if

we want to produce an interpretation that corresponds to either symbolization key above, we would assign the following set of 3-tuples to 'P' as its extension:

{<1, 1, 1>, <3, 3, 3>, <5, 5, 5>, <1, 1, 3>, <1, 3, 1>, <3, 1, 1>,
<1, 1, 5>, <1, 5, 1>, <5, 1, 1>, <3, 3, 1>, <3, 1, 3>, <1, 3, 3>,
<3, 3, 5>, <3, 5, 3>, <5, 3, 3>, <5, 5, 1>, <5, 1, 5>, <1, 5, 5>,
<1, 3, 5>, <1, 5, 3>, <3, 1, 5>, <3, 5, 1>, <5, 1, 3>, <5, 3, 1>,
<1, 2, 2>, <2, 1, 2>, <2, 2, 1>, <1, 4, 4>, <4, 1, 4>, <4, 4, 1>,
<1, 2, 4>, <1, 4, 2>, <2, 1, 4>, <4, 1, 2>, <2, 4, 1>, <4, 2, 1>,
<3, 2, 2>, <2, 3, 2>, <2, 2, 3>, <3, 4, 4>, <4, 3, 4>, <4, 4, 3>,
<3, 2, 4>, <3, 4, 2>, <2, 3, 4>, <4, 3, 2>, <2, 4, 3>, <4, 2, 3>,
<5, 2, 2>, <2, 5, 2>, <2, 2, 5>, <5, 4, 4>, <4, 5, 4>, <4, 4, 5>,
<5, 2, 4>, <5, 4, 2>, <2, 5, 4>, <4, 5, 2>, <2, 4, 5>, <4, 2, 5>}.

That is, the three members of each of these 3-tuples are such that their sum is odd or, alternatively, they are such that either all three are odd or exactly one is odd; and these are the *only* 3-tuples of members of the UD that satisfy the conditions.

The ordering in **n**-tuples allows us to characterize relations that are asymmetric and antisymmetric. The 'greater than or equal to' relation is antisymmetric, that is, if x is greater than or equal to y and y is greater than or equal to x, then x and y are identical. To capture this, we want to be able to distinguish between the 2-tuple <2, 1> and the 2-tuple <1, 2>, for example, since 2 is greater than or equal to 1, but not vice versa. To capture the reading of 'G' in the above symbolization keys, we assign the following set of 2-tuples to 'G' as its extension:

{<1, 1>, <2, 1>, <3, 1>, <4, 1>, <5, 1>, <2, 2>, <3, 2>, <4, 2>,
<5, 2>, <3, 3>, <4, 3>, <5, 3>, <4, 4>, <5, 4>, <5, 5>}.

The two members of each 2-tuple are such that the first is greater than or equal to the second, and these are the only 2-tuples of members of the UD of which this is true.

1-tuples are sometimes called singletons (like single-member sets). To capture the truth-conditions for 'E' in the above symbolization keys, we would assign the set of singletons

{<2>, <4>}

as its extension. It may seem odd to assign this set as an extension of 'E', rather than merely the set {2, 4}, so we shall explain that the reason for doing so in the formal semantics is that defining *all* extensions of predicates as sets of **n**-tuples allows us to state truth-conditions more simply than if we had one exception to the general rule. Finally, the empty set may be assigned to any predicate as its extension. Such an extension is appropriate when we want a predicate to apply to, or be true of, nothing in the UD.

Interpretations for *PL* can now be defined as follows:
An *interpretation for PL* specifies a nonempty set as a UD and assigns a truth-value to each sentence letter of *PL*, a member of the UD to each

individual constant of *PL*, and a set of **n**-tuples of members of the UD to each **n**-place predicate of *PL*.

Here is an example of how we will specify interpretations in the remainder of this section:

UD: The set {1, 2, 3, 4, 5}
a: 1
b: 2
c: 3
d: 4
e: 5
E: {<2>, <4>}
G: {<1, 1>, <2, 1>, <3, 1>, <4, 1>, <5, 1>, <2, 2>, <3, 2>, <4, 2>, <5, 2>, <3, 3>, <4, 3>, <5, 3>, <4, 4>, <5, 4>, <5, 5>}
P: {<$\mathbf{u}_1$, $\mathbf{u}_2$, $\mathbf{u}_3$> : the sum of $\mathbf{u}_1$, $\mathbf{u}_2$, and $\mathbf{u}_3$ is odd}
where $\boldsymbol{u}_1$, $\boldsymbol{u}_2$, *and* $\boldsymbol{u}_3$ *range over members of the UD*

This example shows two ways that we can specify the extensions that are assigned to predicates. We can list the members of the extension explicitly, as we have for the extensions of 'E' and 'G', or we can describe the members, as we have for 'P'. The notation '{<$\mathbf{u}_1$, $\mathbf{u}_2$, $\mathbf{u}_3$> : the sum of $\mathbf{u}_1$, $\mathbf{u}_2$, and $\mathbf{u}_3$ is odd}' means: *the set of ordered triples in which the sum of the first, second, and third members (or the sum of the three members) is odd.* This is a useful way to specify a large extension or one that has an infinite number of members. We can also specify small extensions in this way, if we so desire—that is, we could also have written:

E: {<**u**> : **u** is even}
G: {<$\mathbf{u}_1$, $\mathbf{u}_2$> : $\mathbf{u}_1$ is greater than or equal to $\mathbf{u}_2$}

In what follows, we will adopt the convention that when an interpretation specifies extensions in this second way, the lowercase boldface letter '**u**', with or without a subscript, always ranges over members of the interpretation's UD. With this convention, we do not need the last (italicized) line in our example interpretation. Finally, in this example, we have only specified part of an interpretation; an interpretation must interpret *every* individual constant, predicate, and sentence letter of *PL*. Throughout this chapter and the next, we will display only those parts of an interpretation that are relevant for the sentences that we are discussing.

We note that a single interpretation can interpret two or more individual constants as designating the same member of the UD. This is no different from two or more names or descriptions in a natural language applying to a single individual. For example, 'George Washington' and 'the first president of the United States' refer to the same person. So we could have interpreted both 'a' and 'b', for example, as designating the integer 1. However, while one name can apply to more than one individual in a natural language, an interpretation for *PL cannot* use a single individual constant to designate more than one member of the UD.

We also note that interpretations do not assign anything to individual variables! As Chapter 7 has made clear, the purpose of individual variables is to allow us to speak generally about members of a UD rather than about particular things.

We can now state the truth-conditions for atomic sentences of *PL* (we will revise the clauses somewhat after we discuss quantified sentences, but the end result will be the same). In stating truth-conditions, we use the expression

I(**X**)

to mean: *the value that interpretation **I** assigns to the symbol **X***. We adopt the convention that when a symbol of *PL* occurs inside the parentheses, it is being mentioned, so that we will not need to indicate this with quotation marks (the same convention will apply within square brackets in notation that we shall introduce below). As we did in Chapter 7, we will use boldface letters as metavariables as follows:

- '**P**', '**Q**', and '**R**' will be used to range over formulas of *PL*,
- '**A**' will be used to range over predicates of *PL*,
- '**a**' will be used to range over individual constants of *PL*,
- '**x**' will be used to range over individual variables of *PL*, and
- '**t**' will be used to range over individual terms (individual constants and individual variables) of *PL*.

All of these may be subscripted. The first clause of our definition of truth-conditions is:

> If **P** is a sentence letter, **P** has the truth-value **T** on an interpretation **I** if and only if **I**(**P**) = **T**.

Since every sentence of *PL* will have either the truth-value **T** or the truth-value **F** on an interpretation, it follows that a sentence letter **P** has the truth-value **F** on an interpretation **I** if and only if **I**(**P**) = **F**.

In writing our next clause, we use the set membership symbol '$\in$'. For example,

$$<2, 1> \in \mathrm{I(G)}$$

indicates that $<2, 1>$ is a member of the set I(G). We shall also use the symbol '$\notin$' to indicate that something is *not* a member of a set. The clause for atomic sentences that are formed from predicates and individual constants is

> If **P** is an atomic sentence of the form $\mathbf{Aa}_1 \ldots \mathbf{a_n}$ (where A is an **n**-place predicate), then **P** has the truth-value **T** on interpretation **I** if and only if $<\mathbf{I}(\mathbf{a}_1), \mathbf{I}(\mathbf{a}_2), \ldots, \mathbf{I}(\mathbf{a_n})> \in \mathrm{I}(\mathbf{A})$.

It follows from this clause that an atomic sentence of the form $\mathbf{Aa}_1 \ldots \mathbf{a_n}$ will have the truth-value **F** on an interpretation **I** if and only if $<\mathbf{I}(\mathbf{a}_1), \mathbf{I}(\mathbf{a}_2), \ldots, \mathbf{I}(\mathbf{a_n})> \notin \mathbf{I}(\mathbf{A})$. Note that

$$<\mathbf{I}(\mathbf{a}_1), \mathbf{I}(\mathbf{a}_2), \ldots, \mathbf{I}(\mathbf{a_n})>,$$

which may look daunting, simply means:

> *the **n**-tuple whose first member is the member of the UD that interpretation **I** has assigned to $\boldsymbol{a}_1$, . . ., and whose **n**th member is the member of the UD that interpretation **I** has assigned to $\boldsymbol{a_n}$.*

Let's see how this clause works, using the previous interpretation, which we shall number and refer to as 'interpretation 1' (and we will subsequently subscript 'I' with 1 to indicate that we are talking about this interpretation):

1. UD: The set {1, 2, 3, 4, 5}
a: 1
b: 2
c: 3
d: 4
e: 5
E: {<2>, <4>}
G: {<1, 1>, <2, 1>, <2, 2>, <3, 1>, <3, 2>, <3, 3>, <4, 1>, <4, 2>, <4, 3>, <4, 4>, <5, 1>, <5, 2>, <5, 3>, <5, 4>, <5, 5>}
P: {$<\mathbf{u}_1, \mathbf{u}_2, \mathbf{u}_3>$: the sum of $\mathbf{u}_1$, $\mathbf{u}_2$, and $\mathbf{u}_3$ is odd}

- 'Eb' has the truth-value **T** on interpretation 1 because $<I_1(b)>$, which is <2>, is a member of $I_1(E)$.
- 'Ea' has the truth-value **F** on interpretation 1 because $< I_1(a)>$, which is <1>, is not a member of $I_1(E)$.
- 'Gca' has the truth-value **T** on interpretation 1 because $< I_1(c), I_1(a)>$, which is <3, 1>, is a member of $I_1(G)$.
- 'Gdd' has the truth-value **T** on interpretation 1 because $<I_1(d), I_1(d)>$, which is <4, 4>, is a member of **I**(G).
- 'Gbe' has the truth-value **F** on interpretation 1 because $< I_1(b), I_1(e)>$, which is <2, 5>, is not a member of $I_1(G)$.

Note that in the previous cases, we simply had to examine the lists of members of the extensions of these predicates to determine whether the relevant 1- or 2-tuple was included in those extensions. In the next cases, we will need to consult the description that interpretation 1 used to specify the 3-tuples that are members of the extension assigned to 'P'.

- 'Pace' and 'Pabd' both have the truth-value **T** on interpretation 1 because $< I_1(a), I_1(c), I_1(e)>$, which is <1, 3, 5>, and $< I_1(a), I_1(b),$

$I_1(d)>$, which is $<1, 2, 4>$, are both members of $I_1(P)$—the sum of 1, 3, and 5 and the sum of 1, 2, and 4 are both odd numbers.

- 'Pabc' has the truth-value **F** on interpretation 1 because $< I_1(a), I_1(b), I_1(c)>$, which is $<1, 2, 3>$, is not a member of $I_1(P)$; the sum of 1, 2, and 3 is not an odd number.

The truth-conditions for negations, conjunctions, disjunctions, material conditionals, and material biconditionals of *PL* will be based on the characteristic truth-tables for these connectives. Because these will be straightforward, we turn to a discussion of the truth-conditions for quantified sentences of *PL*. We will be defining the truth-conditions of sentences like '$(\forall x)(Fx \vee Gx)$' in terms of the semantics of their subformulas. But '$Fx \vee Gx$' is an open sentence, and we consider open sentences to be neither true nor false. Free variables are not names, and interpretations therefore do not assign values to them. To formally explain how free variables work, we will need some additional definitions and notation to capture a fairly clear intuitive understanding of the semantics of quantified sentences. We will talk about that intuitive understanding first.

Consider the sentence '$(\forall x)(Ex \supset Gxx)$', which we may read as 'Each x is such that if x is F then x stands in the relation G to itself ' or 'Everything that is F stands in the relation G to itself'. When we specify a UD for interpreting this sentence, we thereby specify what 'everything' is, namely, everything in the UD. The part of the sentence that follows the universal quantifier, the open sentence '$Ex \supset Gxx$', specifies a condition that may or may not hold of the individual members of the UD. The function of the universal quantifier is to state that this condition holds for each member of the UD, and consequently, the sentence is true if and only if every member of the UD does meet that condition. In the case of interpretation 1 and English readings of the predicates 'E' and 'G' as 'is even' and 'is greater than or equal to', 'everything' is comprised of the integers 1 through 5, and the sentence may be read as: 'Each of the integers 1 through 5 is such that if it is even, then it is greater than or equal to itself'—which is true. Conversely, given interpretation 1, the sentence '$(\forall x)(Gxx \supset Ex)$' may be read as: 'Each of the positive integers 1 through 5 is such that if it is greater than or equal to itself then it is even', and this is false. It is false because three members of the UD fail to satisfy the condition specified by the open sentence '$Gxx \supset Ex$': the integers 1, 3, and 5 all satisfy the antecedent but fail to satisfy the consequent, and therefore they all fail to satisfy the open sentence.

Existential quantifers function in *PL* to indicate that at least one member of the UD satisfies a certain condition. So the sentence '$(\exists x)(Gxx \ \& \ Ex)$' says that at least one member of the UD both stands in the relation G to itself and has the property E. This sentence is true with respect to interpretation 1. In fact, there are two such members of the UD: both 2 and 4 satisfy 'Gxx' and 'Ex', as each is both greater than or equal to itself and even. On the other hand, the sentence '$(\exists x)(Ex \ \& \ Pxxx)$' is false on interpretation 1 because not even one member **u** of the UD satisfies the condition specified by '$Ex \ \& \ Pxxx$'.

1, 3, and 5 fail to satisfy this condition because they are not even, while 2 and 4 fail to satisfy this condition because neither 2 + 2 + 2 nor 4 + 4 + 4 is odd.

We will now dive into the formal semantics. Some readers may wish to skip or skim the following discussion and perhaps return to it later; we have written the rest of this chapter on the assumption that some readers will do so. However, those readers who plan to skip the following and proceed to Section 8.2 should first read the last four paragraphs of Section 8.1.

To consider the different values that variables range over, we will use **variable assignments**. A ***variable assignment for an interpretation I*** assigns to each individual variable of *PL* a member of the UD. For example, given interpretation 1, each variable assignment will assign values from the set {1, 2, 3, 4, 5} to the individual variables of *PL*. Intuitively, a variable assignment captures one way in which the variables of *PL*, in their role as pronouns, can be used to refer to objects in an interpretation's UD. We use the notation

$$\mathbf{d_I},\ \mathbf{d_I(x)}$$

to denote, respectively, a variable assignment for interpretation **I** (think of '$\mathbf{d_I}$' as shorthand for 'designates' or 'denotes') and the value that $\mathbf{d_I}$ assigns to **x.** So

$$d_1(x) = 5$$
$$d_1(y) = 1$$
$$d_1(z) = 3$$
$$d_1(x_1) = 2$$
$$d_1(y_1) = 5$$

mean that the variable assignment d_1 assigns the value 5 to 'x', the value 1 to 'y', the value 3 to 'z', the value 2 to 'x_1', and the value 5 to 'y_1'**.** We will use this example below, so we (arbitrarily) stipulate that d_1 assigns 3 to all of the other individual variables of *PL*. Note that a variable assignment can assign the same value to more than one variable. Given the infinitely many individual variables in the language *PL*, this will certainly be the case for variable assignments for interpretations that have finite UDs. Also note that variable assignments are not required to assign each member of the UD to at least one variable; they can leave some members of the UD unassigned.

In defining the truth-conditions for sentences of *PL*, we will define the conditions under which a variable assignment **satisfies** a sentence or open sentence of *PL*, and this will capture the intuitive concept of objects satisying what an open sentence says.[1] We'll look at a simple example of how we'll use

[1]Some authors allow all open sentences to be true or false on interpretations, so the concept of satisfaction is not needed in their truth-definitions. Other authors use a type of semantics known as *substitution semantics*; in this type of semantics the concept of satisfaction is also unnecessary. For obvious reasons, the semantics we present in this section is known as *satisfaction semantics* (or sometimes as *referential* or *objectual semantics*). Satisfaction semantics were proposed by Alfred Tarski in "der Wahrheitsbegriff in den formalisierten Sprachen," *Studia Philosophica*, 1 (1936), 261–405.

variable assignments before putting together the full formal definition of the truth-conditions for quantified (and other) sentences of *PL*. Here are the satisfaction conditions for an atomic formula in which all of the individual terms are variables:

> If **P** is an atomic formula of the form $\mathbf{Ax_1 \ldots x_n}$ (where **A** is an **n**-place predicate), then the variable assignment $\mathbf{d_I}$ satisfies **P** on interpretation **I** if and only if $<\mathbf{d_I}(\mathbf{x_1}), \mathbf{d_I}(\mathbf{x_2}), \ldots, \mathbf{d_I}(\mathbf{x_n})> \in \mathbf{I}(\mathbf{A})$.

The variable assignment $\mathbf{d}_1$ that is partially displayed above satisfies 'Gxy' on interpretation 1, because $<\mathbf{d}_1(x), \mathbf{d}_1(y)>$, which is $<5, 1>$, is a member of $I_1(G)$. On the other hand, a variable assignment that assigns 2 to 'x' and 3 to 'y' *will not* satisfy 'Gxy'.

Now consider the quantified sentence '$(\exists x)(\forall y)$Gxy'. To capture the conditions under which a variable assignment $\mathbf{d_I}$ satisfies this sentence, we need to know whether '$(\forall y)$Gxy' is satisfied by *at least one* variable assignment that is exactly like the assignment $\mathbf{d_I}$ *except that* it may assign any value to 'x'. Put more intuitively, if at least one member x of the UD satisfies '$(\forall y)$Gxy', then '$(\exists x)(\forall y)$Gxy' will be satisfied. To formalize the intuitive idea, we introduce the notation

$$\mathbf{d_I}[\mathbf{u}/\mathbf{x}]$$

which means: *the variable assignment for interpretation* ***I*** *that assigns the same values as* $\boldsymbol{d_I}$ *to all variables other than* ***x****, and that assigns the value* ***u*** *to* ***x***. $\mathbf{d_I}[\mathbf{u}/\mathbf{x}]$ is called a **variant** of the assignment $\mathbf{d_I}$. So, for example, given the sample variable assignment d_1 above, the variant $d_1[2/x]$ makes the following assignments:

$$d_1[2/x](x) = 2$$
$$d_1[2/x](y) = 1$$
$$d_1[2/x](z) = 3$$
$$d_1[2/x](x_1) = 2$$
$$d_1[2/x](y_1) = 5,$$

and $d_1[2/x]$ assigns 3 to all of the other individual variables of *PL* (because d_1 does).

The following discussion will be easier to understand if the reader remembers the following two simple points about the variants of a variable assignment $\mathbf{d_I}$:

- $\mathbf{d_I}[\mathbf{u}/\mathbf{x}](\mathbf{x})$ is simply **u**.
- For every variable **y** that is distinct from **x**, $\mathbf{d_I}[\mathbf{u}/\mathbf{x}]$ assigns to **y** the same member of the UD that $\mathbf{d_I}$ assigns to **y**, that is, $\mathbf{d_I}[\mathbf{u}/\mathbf{x}](\mathbf{y})$ is simply $\mathbf{d_I}(\mathbf{y})$.

Here's how variable assignments work in the semantics. The satisfaction clauses for quantified formulas are

> If **P** is a formula of the form $(\forall \mathbf{x})\mathbf{Q}$, then $\mathbf{d_I}$ satisfies **P** on interpretation **I** if and only if each member **u** of the UD is such that $\mathbf{d_I}[\mathbf{u}/\mathbf{x}]$ satisfies **Q** on interpretation **I**

and

> If **P** is a formula of the form $(\exists \mathbf{x})\mathbf{Q}$, then $\mathbf{d_I}$ satisfies **P** on interpretation **I** if and only if there is at least one member **u** of the UD such that $\mathbf{d_I}[\mathbf{u}/\mathbf{x}]$ satisfies **Q** on interpretation **I**.

To illustrate these clauses, we first consider the singly quantified sentences '$(\forall x)Ex$' and '$(\exists x)Ex$', using interpretation 1 and our sample variable assignment d_1. For these examples, we note that given the five-member UD of interpretation 1, there are five variants $d_1[\mathbf{u}/x]$ of d_1, only four of which are distinct from d_1. The variants are are $d_1[1/x]$, $d_1[2/x]$, $d_1[3/x]$, $d_1[4/x]$, and $d_1[5/x]$ (the last of these is the same as d_1). So d_1 satisfies '$(\forall x)Ex$' on interpretation 1 if and only if all five of $d_1[1/x]$, $d_1[2/x]$, $d_1[3/x]$, $d_1[4/x]$, and $d_1[5/x]$ satisfy 'Ex'. Not all of them do:

- $d_1[1/x]$ does not satisfy 'Ex' because $<d_1[1/x]>$ is $<1>$, and $<1> \notin I(E)$.
- $d_1[2/x]$ does satisfy 'Ex' because $<d_1[2/x]>$ is $<2>$, and $<2> \in I(E)$.
- $d_1[3/x]$ does not satisfy 'Ex' because $<d_1[3/x]>$ is $<3>$, and $<3> \notin I(E)$.
- $d_1[4/x]$ does satisfy 'Ex' because $<d_1[4/x]>$ is $<4>$, and $<4> \in I(E)$.
- $d_1[5/x]$ does not satisfy 'Ex' because $<d_1[5/x]>$ is $<5>$, and $<5> \notin I(E)$.

Each of the variants $d_1[1/x]$, $d_1[3/x]$, and $d_1[5/x]$ taken alone is sufficient to establish that d_1 does not satisfy '$(\forall x)Ex$'. On the either hand, d_1 does satisfy '$(\exists x)Ex$' because at least one of the five variants of d_1 satisfies 'Ex'—in fact, both $d_1[2/x]$ and $d_1[4/x]$ do.

Now we'll point out a very important fact about the variable assignment d_1: the value that d_1 itself assigns to 'x' or to any other other variable has no bearing on whether d_1 satisfies '$(\forall x)Ex$' or '$(\exists x)Ex$'! The values assigned to other variables don't matter, because they don't occur in these sentences. And the value assigned to 'x' doesn't matter because the satisfaction conditions call for looking at variants that assign different values to 'x'. In each case, what mattered was whether all or some of the five variants $d_1[1/x]$, $d_1[2/x]$, $d_1[3/x]$, $d_1[4/x]$, and $d_1[5/x]$ satisfy 'Ex'. And in determining this, we looked at the value that the variant assigned to 'x', rather than the value that d_1 assigns to 'x'. The assignment d_1 itself merely gave us a starting point for constructing

the variants. We could have started with any other variable assignment for interpretation 1, and we would have found that it also satifies '$(\exists x)Ex$' and fails to satisfy '$(\forall x)Ex$'. Because we get the same results regardless of which variable assignement we start with, we will define truth across the board as satisfaction by *all* variable assignments and falsehood as satisfaction by *none*.

We will formally define satisfaction and truth and falsehood momentarily, but we will first introduce an additional piece of notation that is useful for writing the satisfaction clause for atomic formulas. The satisfaction clauses that we have examined so far for atomic formulas apply to formulas in which all the individual terms are constants or all the individual terms are variables. We need to generalize and provide a single satisfaction clause for atomic formulas that contain any mixture of individual constants and/or variables. To this end, we define

$$\text{den}_{\mathbf{I},\mathbf{d_I}}$$

(read this as: *the denotation with respect to interpretation* ***I*** *and variable assignment* $\boldsymbol{d_I}$) as follows:

1. If $\mathbf{t}$ is a variable, then $\text{den}_{\mathbf{I},\mathbf{d_I}}(\mathbf{t}) = \mathbf{d_I}(\mathbf{t})$.
2. If $\mathbf{t}$ is an individual constant, then $\text{den}_{\mathbf{I},\mathbf{d_I}}(\mathbf{t}) = \mathbf{I}(\mathbf{t})$.

The denotation of a term is simply the member of the UD that the variable assignment or interpretation says it designates.

We can now recursively define the concept of satisfaction of formulas and then define truth and falsehood for the sentences of *PL*. Let **I** be an interpretation, $\mathbf{d_I}$ a variable assignment for **I**, and **P** a formula of *PL*. Then

1. If **P** is a sentence letter, $\mathbf{d_I}$ satisfies **P** on interpretation **I** if and only if $\mathbf{I}(\mathbf{P}) = \mathbf{T}$.
2. If **P** is an atomic formula of the form $\mathbf{At_1} \ldots \mathbf{t_n}$ (where **A** is an **n**-place predicate), $\mathbf{d_I}$ satisfies **P** on interpretation **I** if and only if $<\text{den}_{\mathbf{I},\mathbf{d_I}}(\mathbf{t}_1), \text{den}_{\mathbf{I},\mathbf{d_I}}(\mathbf{t}_2), \ldots, \text{den}_{\mathbf{I},\mathbf{d_I}}(\mathbf{t_n})> \in \mathbf{I}(\mathbf{A})$.
3. If **P** is a formula of the form $\sim \mathbf{Q}$, $\mathbf{d_I}$ satisfies **P** on interpretation **I** if and only if $\mathbf{d_I}$ does not satisfy **Q** on interpretation **I**.
4. If **P** is a formula of the form $\mathbf{Q} \,\&\, \mathbf{R}$, $\mathbf{d_I}$ satisfies **P** on interpretation **I** if and only if $\mathbf{d_I}$ satisfies **Q** on interpretation **I** and $\mathbf{d_I}$ satisfies **R** on interpretation **I**.
5. If **P** is a formula of the form $\mathbf{Q} \vee \mathbf{R}$, $\mathbf{d_I}$ satisfies **P** on interpretation **I** if and only if either $\mathbf{d_I}$ satisfies **Q** on interpretation **I** or $\mathbf{d_I}$ satisfies **R** on interpretation **I**.
6. If **P** is a formula of the form $\mathbf{Q} \supset \mathbf{R}$, $\mathbf{d_I}$ satisfies **P** on interpretation **I** if and only if either $\mathbf{d_I}$ does not satisfy **Q** on interpretation **I** or $\mathbf{d_I}$ satisfies **R** on interpretation **I**.

7. If **P** is a formula of the form **Q** ≡ **R**, $\mathbf{d_I}$ satisfies **P** on interpretation **I** if and only if either $\mathbf{d_I}$ satisfies **Q** on interpretation **I** and $\mathbf{d_I}$ satisfies **R** on interpretation **I**, or $\mathbf{d_I}$ does not satisfy **Q** on interpretation **I** and $\mathbf{d_I}$ does not satisfy **R** on interpretation **I**.
8. If **P** is a formula of the form (∀**x**)**Q**, $\mathbf{d_I}$ satisfies **P** on interpretation **I** if and only if each member **u** of the UD is such that $\mathbf{d_I}$[**u**/**x**] satisfies **Q** on interpretation **I**.
9. If **P** is a formula of the form (∃**x**)**Q**, $\mathbf{d_I}$ satisfies **P** on interpretation **I** if and only if there is at least one member **u** of the UD such that $\mathbf{d_I}$[**u**/**x**] satisfies **Q** on interpretation **I**.

We have already discussed clauses 8 and 9, with an example. Note that the values that $\mathbf{d_I}$ assigns to variables play no role in clause 1; all that matters in the case of sentence letters are the truth-values that the interpretation assigns to them. The second clause is a generalization of the restricted clauses that we gave earlier for atomic formulas, and it says, for example, that d_1 satisfies 'Gxa' if $<d_1(x), I_1(a)>$—the 2-tuple whose first member is the object that d_I assigns to the variable 'x' and whose second member is the object that I_1 assigns to the constant 'a'—is a member of I_1('G'). Clauses 3–7 are straightforward.

Finally, the definitions of truth and falsehood are

> A sentence **P** of *PL* is **true on an interpretation I** if and only if every variable assignment $\mathbf{d_I}$ satisfies **P** on **I**. A sentence **P** of *PL* is **false on an interpretation I** if and only if no variable assignment $\mathbf{d_I}$ satisfies **P** on **I**.

In Chapter 11 we shall formally prove that given a sentence **P** and interpretation **I**, either all variable assignments for **I** satisfy **P** or none do. (This is not generally true for *open* sentences.) Therefore, each sentence of *PL* will be true or false on any interpretation.

We'll end this section with a series of examples that use interpretation 1 and a few other interpretations. The sentence '(∃x)Ex & (∃x)Pxxx' is true on interpretation 1. Consider an arbitary variable assignment d_1 for interpretation 1 (here and below, d_1 can be any variable assignment for interpretation 1, not necessarily the specific assignment we used in previous examples):

- By clause 4, d_1 satisfies '(∃x)Ex & (∃x)Pxxx' if and only if d_1 satisfies both '(∃x)Ex' and '(∃x)Pxxx'.
- By clause 9, d_1 satisfies '(∃x)Ex' if and only if at least one variant of d_1 satisfies 'Ex'. Both the variant d_1[2/x] and the variant d_1[4/x] satisfy 'Ex' because both <2> and <4> are members of I_1(E).
- By clause 9, d_1 satisfies '(∃x)Pxxx' if and only if at least one variant of d_1 satisfies 'Pxxx'. The variants d_1[1/x], d_1[3/x], and d_1[5/x] all satisfy 'Pxxx', because <1, 1, 1>, <3, 3, 3>, and <5, 5, 5> are all members of I_1(P).
- Therefore, d_1 satisfies the sentence '(∃x)Ex & (∃x)Pxxx'.

It follows that '(∃x)Ex & (∃x)Pxxx' is true on interpretation 1. Note that the satisfaction conditions for the two conjuncts are examined independently, and that the variants of d_1 that satisfy '(∃x)Ex' are distinct from the variants that satisfy '(∃x)Pxxx'.

In contrast, the sentence '(∃x)(Ex & Pxxx)', in which everything falls within the scope of a single existential quantifier, is false on interpretation 1. Let d_1 be an arbitrary variable assignment for this interpretation.

- By clause 9, d_1 satisfies '(∃x)(Ex & Pxxx)' if and only if there is at least one member **u** of the UD such that $d_1[\mathbf{u}/x]$ satisfies 'Ex & Pxxx'.
- By clause 4, a variant $d_1[\mathbf{u}/x]$ satisfies 'Ex & Pxxx' if and only if it satisfies both 'Ex' and 'Pxxx'. We shall show that no variant satisfies both conjuncts.
- $d_1[1/x]$ does not satisfy 'Ex' because $<1> \notin I(E)$.
- $d_1[2/x]$ does not satisfy 'Pxxx' because $<2, 2, 2> \notin I(P)$.
- $d_1[3/x]$ does not satisfy 'Ex' because $<3> \notin I(E)$.
- $d_1[4/x]$ does not satisfy 'Pxxx' because $<4, 4, 4> \notin I(E)$.
- $d_1[5/x]$ does not satisfy 'Ex' because $<5> \notin I(E)$.
- Therefore, d_1 does not satisfy '(∃x)(Ex & Pxxx)'.

We may conclude that '(∃x)(Ex & Pxxx)' is false on interpretation 1.

Next we'll look at some simple sentences with nested quantifiers, still using interpretation 1, to see how these work. Our first sentence is '(∀x)(∀y)Gxy'.

- By clause 8, d_1 satisfies '(∀x)(∀y)Gxy' if and only if all of the variants $d_1[1/x]$, $d_1[2/x]$, $d_1[3/x]$, $d_1[4/x]$, and $d_1[5/x]$ satisfy '(∀y)Gxy'.

To determine whether these variants satisfy '(∀y)Gxy', we need to know whether, given the specified value for 'x', 'Gxy' will be satisfied no matter what value 'y' may have. So we will need to look at variants of these variants. For this example and those that follow, we introduce a convention that makes it easier to write variants of variants: $\mathbf{d_I}[\mathbf{u}_1/\mathbf{x}_1, \mathbf{u}_2/\mathbf{x}_2, \ldots, \mathbf{u_n}/\mathbf{x_n}]$ is shorthand for the variable assignment $\mathbf{d_I}[\mathbf{u}_1/\mathbf{x}_1]\ [\mathbf{u}_2/\mathbf{x}_2] \ldots [\mathbf{u_n}/\mathbf{x_n}]$—the variable assignment that starts out like $\mathbf{d_I}$ and results from successive stipulations that $\mathbf{u}_1$ will be assigned to $\mathbf{x}_1$, $\mathbf{u}_2$ to $\mathbf{x}_2$, . . . , and $\mathbf{u_n}$ to $\mathbf{x_n}$. To continue the example, according to clause 8:

- $d_1[1/x]$ satisfies '(∀y)Gxy' if and only if all five of $d_1[1/x, 1/y]$, $d_1[1/x, 2/y]$, $d_1[1/x, 3/y]$, $d_1[1/x, 4/y]$, and $d_1[1/x, 5/y]$ satisfy 'Gxy'
- $d_1[2/x]$ satisfies '(∀y)Gxy' if and only if all five of $d_1[2/x, 1/y]$, $d_1[2/x, 2/y]$, $d_1[2/x, 3/y]$, $d_1[2/x, 4/y]$, and $d_1[2/x, 5/y]$ satisfy 'Gxy'
- $d_1[3/x]$ satisfies '(∀y)Gxy' if and only if all five of $d_1[3/x, 1/y]$, $d_1[3/x, 2/y]$, $d_1[3/x, 3/y]$, $d_1[3/x, 4/y]$, and $d_1[3/x, 5/y]$ satisfy 'Gxy'

- $d_1[4/x]$ satisfies '$(\forall y)Gxy$' if and only if all five of $d_1[4/x, 1/y]$, $d_1[4/x, 2/y]$, $d_1[4/x, 3/y]$, $d_1[4/x, 4/y]$, and $d_1[4/x, 5/y]$ satisfy 'Gxy'
- $d_1[5/x]$ satisfies '$(\forall y)Gxy$' if and only if all five of $d_1[5/x, 1/y]$, $d_1[5/x, 2/y]$, $d_1[5/x, 3/y]$, $d_1[5/x, 4/y]$, and $d_1[5/x, 5/y]$ satisfy 'Gxy'.

Given the satisfaction clause for atomic formulas, the first item in the previous list amounts to

- $d_1[1/x]$ satisfies '$(\forall y)Gxy$' if and only if $<d_1[1/x, 1/y](x), d_1[1/x, 1/y](y)>$, $<d_1[1/x, 2/y](x), d_1[1/x, 2/y](y)>$, $<d_1[1/x, 3/y](x), d_1[1/x, 3/y](y)>$, $<d_1[1/x, 4/y](x), d_1[1/x, 4/y](y)>$, and $<d_1[1/x, 5/y](x), d_1[1/x, 5/y](y)>$ are all members of $I_1(G)$.

But we can write this more simply. Because $\mathbf{d_I}[\mathbf{u}_1/x, \mathbf{u}_2/y](x)$ is just $\mathbf{u}_1$ and $\mathbf{d_I}[\mathbf{u}_1/x, \mathbf{u}_2/y](y)$ is just $\mathbf{u}_2$, the 2-tuple $<\mathbf{d_I}[\mathbf{u}_1/x, \mathbf{u}_2/y](x), \mathbf{d_I}[\mathbf{u}_1/x, \mathbf{u}_2/y](y)>$ is simply $<\mathbf{u}_1, \mathbf{u}_2>$. So the previous five items are equivalent to

- $d_1[1/x]$ satisfies '$(\forall y)Gxy$' if and only if $<1, 1>$, $<1, 2>$, $<1, 3>$, $<1, 4>$, and $<1, 5>$ are all members of $I_1(G)$
- $d_1[2/x]$ satisfies '$(\forall y)Gxy$' if and only if $<2, 1>$, $<2, 2>$, $<2, 3>$, $<2, 4>$, and $<2, 5>$ are all members of $I_1(G)$
- $d_1[3/x]$ satisfies '$(\forall y)Gxy$' if and only if $<3, 1>$, $<3, 2>$, $<3, 3>$, $<3, 4>$, and $<3, 5>$ are all members of $I_1(G)$
- $d_1[4/x]$ satisfies '$(\forall y)Gxy$' if and only if $<4, 1>$, $<4, 2>$, $<4, 3>$, $<4, 4>$, and $<4, 5>$ are all members of $I_1(G)$
- $d_1[5/x]$ satisfies '$(\forall y)Gxy$' if and only if $<5, 1>$, $<5, 2>$, $<5, 3>$, $<5, 4>$, and $<5, 5>$ are all members of $I_1(G)$.

Only $d_1[5/x]$ satisfies the specified condition. Because $d_1[1/x]$, $d_1[2/x]$, $d_1[3/x]$, and $d_1[4/x]$ do not satisfy '$(\forall y)Gxy$', d_1 does not satisfy '$(\forall x)(\forall y)Gxy$' (actually, any one of these taken alone suffices), and this sentence is therefore false on interpretation 1. Looking at the steps we have gone through along with the 5-tuples that appear in the last bulleted list, we can see that the sentence '$(\forall x)(\forall y)Gxy$' is true on interpretation 1 if and only if the extension of 'G' includes every 2-tuple of members of the UD. Of course, this is exactly what we want, given the universal quantifiers in this sentence.

We'll be briefer in our discussions of the remaining examples, omitting detailed explicit statements of every step. Consider '$(\exists x)(\exists y)Gxy$':

- By clause 9, d_1 satisfies '$(\exists x)(\exists y)Gxy$' if and only if at least one of the five variants $d_1[1/x]$, $d_1[2/x]$, $d_1[3/x]$, $d_1[4/x]$, and $d_1[5/x]$ satisfies '$(\exists y)Gxy$'. In fact, they all do. For example,
- $d_1[1/x]$ satisfies '$(\exists y)Gxy$' if and only if at least one of the variants $d_1[1/x, 1/y]$, $d_1[1/x, 2/y]$, $d_1[1/x, 3/y]$, $d_1[1/x, 4/y]$, and $d_1[1/x, 5/y]$ satisfies 'Gxy'. One of these does:

- $d_1[1/x, 1/y]$ satisfies 'Gxy' because $<1, 1> \in I_1(G)$.
- So $d_1[1/x]$ satisfies '$(\exists y)$Gxy', and d_1 therefore satisfies '$(\exists x)(\exists y)$Gxy'.

We conclude that '$(\exists x)(\exists y)$Gxy' is true on interpretation 1. This is because we were able to identify a 2-tuple that is in the extension of 'G'. Although other 2-tuples are also members of this extension, the nested existential quantifiers only require that at least one 2-tuple is a member.

Next, consider '$(\exists x)(\forall y)$Gxy':

- d_1 satisfies '$(\exists x)(\forall y)$Gxy' if and only if at least one of the variants $d_1[1/x]$, $d_1[2/x]$, $d_1[3/x]$, $d_1[4/x]$, and $d_1[5/x]$ satisfies '$(\forall y)$Gxy'.
- In the example before the last, we explained that $d_1[5/x]$ satisfies '$(\forall y)$Gxy'. So d_1 satisfies '$(\exists x)(\forall y)$Gxy'.

Because the satisfaction assignment d_1 satisfies '$(\exists x)(\forall y)$Gxy', the sentence is true on interpretation 1. Note that we were able to conclude that this sentence is true because we were able to identify one member of the UD, 5, that is greater than or equal to all members of the UD.

The sentence '$(\forall y)(\exists x)$Gxy' reverses the order of quantifiers in the previous sentence:

- d_1 satisfies '$(\forall y)(\exists x)$Gxy' if and only if all of the variants $d_1[1/y]$, $d_1[2/y]$, $d_1[3/y]$, $d_1[4/y]$, and $d_1[5/y]$ satisfy '$(\exists x)$Gxy'.
- $d_1[1/y]$ satisfies '$(\exists x)$Gxy' if and only if at least one of the variants $d_1[1/y, 1/x]$, $d_1[1/y, 2/x]$, $d_1[1/y, 3/x]$, $d_1[1/y, 4/x]$, and $d_1[1/y, 5/x]$ satisfies 'Gxy'. They all do.
- $d_1[2/y]$ satisfies '$(\exists x)$Gxy' if and only if at least one of the variants $d_1[2/y, 1/x]$, $d_1[2/y, 2/x]$, $d_1[2/y, 3/x]$, $d_1[2/y, 4/x]$, and $d_1[2/y, 5/x]$ satisfies 'Gxy'. The last four do.
- $d_1[3/y]$ satisfies '$(\exists x)$Gxy' if and only if at least one of the variants $d_1[3/y, 1/x]$, $d_1[3/y, 2/x]$, $d_1[3/y, 3/x]$, $d_1[3/y, 4/x]$, and $d_1[3/y, 5/x]$ satisfies 'Gxy'. The last three do.
- $d_1[4/y]$ satisfies '$(\exists x)$Gxy' if and only if at least one of the variants $d_1[4/y, 1/x]$, $d_1[4/y, 2/x]$, $d_1[4/y, 3/x]$, $d_1[4/y, 4/x]$, and $d_1[4/y, 5/x]$ satisfies 'Gxy'. The last two do.
- $d_1[5/y]$ satisfies '$(\exists x)$Gxy' if and only if at least one of the variants $d_1[5/y, 1/x]$, $d_1[5/y, 2/x]$, $d_1[5/y, 3/x]$, $d_1[5/y, 4/x]$, and $d_1[5/y, 5/x]$ satisfies 'Gxy'. The last does.
- Therefore, d_1 satisfies '$(\forall y)(\exists x)$Gxy'.

And the sentence '$(\forall y)(\exists x)$Gxy' is therfore true on interpretation 1. Note that in this case the sentence is true because each member of the UD is such that there is at least one member that is greater than or equal to that member. This reflects the order of the initial quantifiers.

The sentence '$(\exists x)(Ex \ \& \ (\forall y)Gxy) \supset Pebe$' is true on interpretation 1. Consider an arbitary variable assignment d_1 for interpretation 1.

- By clause 6, d_1 satisfies '$(\exists x)(Ex \ \& \ (\forall y)Gxy) \supset Pebe$' if and only if either d_1 does not satisfy '$(\exists x)(Ex \ \& \ (\forall y)Gxy)$' or d_1 does satisfy 'Pbeb'.
- Looking at the consequent first, d_1 satisfies 'Pebe' if and only if $<den_{\mathbf{I_1,d_1}}(e), den_{\mathbf{I_1,d_1}}(b), den_{\mathbf{I_1,d_1}}(e)> \in I(P)$.
- By definition, $<den_{\mathbf{I_1,d_1}}(e), den_{\mathbf{I_1,d_1}}(b), den_{\mathbf{I_1,d_1}}(e)>$ is $<I_1(e), I_1(b), I_1(e)>$, or $<5, 2, 5>$, which is not a member of I(P). So d_1 does not satisfy 'Pebe'.
- Looking at the antecedent, d_1 satisfies '$(\exists x)(Ex \ \& \ (\forall y)Gxy)$' if and only if there is at least one member **u** of the UD such that $d_1[\mathbf{u}/x]$ satisfies '$Ex \ \& \ (\forall y)Gxy$', which will be the case if and only if $d_1[\mathbf{u}/x]$ satisfies both 'Ex' and '$(\forall y)Gxy$'. We shall show that there is no such member.
- If **u** is 1, 3, or 5, then $d_1[\mathbf{u}/x]$ does not satisfy 'Ex' because $<d_1[\mathbf{u}/x](x)>$, which is either $<1>$, $<3>$, or $<5>$ in this case, is not a member of $I_1(E)$.
- If **u** is 2 or 4, then $d_1[\mathbf{u}/x]$ does not satisfy '$(\forall y)Gxy$'. We explained this case earlier when we looked at the sentence '$(\forall x)(\forall y)Gxy$'.
- Therefore, d_1 does not satisfy '$(\exists x)(Ex \ \& \ (\forall y)Gxy)$', and so it does satisfy the conditional '$(\exists x)(Ex \ \& \ (\forall y)Gxy) \supset Pebe$'.

We conclude that '$(\exists x)(Ex \ \& \ (\forall y)Gxy) \supset Pebe$' is true on interpretation 1.

We'll now look at examples with other interpretations. Consider the sentence

$(\forall y)(By \supset \sim (\exists z)Dyz)$

and interpretation 2:

2. UD: The set {2, 4}
   B: {<2>}
   D: {<4, 2>}

Consider any variable assignment d_2 for this interpretation.

- By clause 8, d_2 satisfies the sentence if and only if both $d_2[2/y]$ and $d_2[4/y]$ satisfy the open sentence '$By \supset \sim (\exists z)Dyz$'.
- $d_2[2/y]$ satisfies the consequent of the open sentence (in addition to its antecedent) because it fails to satisfy '$(\exists z)Dyz$'. The latter is because neither $d_2[2/y, 2/z]$ nor $d_2[2/y, 4/z]$ satisfies 'Dyz':
  - $d_2[2/y, 2/z]$ does not satisfy 'Dyz' because $<2, 2> \notin I_2(D)$.
  - $d_2[2/y, 4/z]$ does not satisfy 'Dyz' because $<2, 4> \notin I_2(D)$.

- Because $d_2[2/y]$ satisfies ~ (∃z)Dyz', it also satisfies 'By ⊃ ~ (∃z)Dyz'.
- The variant $d_2[4/y]$ satisfies 'By ⊃ ~ (∃z)Dyz' because it fails to satisfy the antecedent (it also fails to satisfy the consequent): $<4> \notin I_2(B)$. So $d_2[4/y]$ satisfies 'By ⊃ ~ (∃z)Dyz'.
- d_2 therefore satisfies '(∀y)(By ⊃ ~ (∃z)Dyz)'.

Thus, the sentence '(∀y)(By ⊃ ~ (∃z)Dyz)' is true on interpretation 2.

However, '(∀y)(By ⊃ ~ (∃z)Dyz)' is false on the following interpretation:

3. UD: The set of positive integers
   B: {<**u**>: **u** is prime}
   D: {<$\mathbf{u}_1$, $\mathbf{u}_2$>: $\mathbf{u}_1$ is greater than $\mathbf{u}_2$}

As we apply our definitions, it may be helpful to keep in mind the reading of the sentence given this interpretation: 'Every positive integer is such that if it is prime then there is no positive integer than which it is greater'. The sentence is obviously false on this interpretation. Let d_3 be any variable assignment for interpretation 3.

- By clause 8, d_3 satisfies '(∀y)(By ⊃ ~ (∃z)Dyz)' if and only if every member **u** of the UD is such that $d_3[\mathbf{u}/y]$ satisfies 'By ⊃ ~ (∃z)Dyz'. We'll look at the case where **u** is 2—for this will be sufficient to show that the sentence is false on interpretation 3.
- The variant $d_3[2/y]$ satisfies 'By ⊃ ~ (∃z)Dyz' if and only if either $d_3[2/y]$ does not satisfy 'By' or $d_3[2/y]$ does satisfy '~ (∃z)Dyz'.
- $d_3[2/y]$ satisfies 'By' if and only if $<2> \in I_3(B)$. It is a member.
- $d_3[2/y]$ satisfies '~ (∃z)Dyz' if and only if it does not satisfy '(∃z)Dyz'.
- $d_3[2/y]$ satisfies '(∃z)Dyz' if and only if there is at least one member **u** of the UD such that $d_3[2/y, \mathbf{u}/z]$ satisfies 'Dyz', that is, if and only if there is at least one member **u** of the UD such that $<2, \mathbf{u}> \in I_3(D)$.
- $<2, 1> \in I(D)$, so $d_3[2/y]$ satisfies '(∃z)Dyz' and therefore does not satisfy '~ (∃z)Dyz'.
- Hence $d_3[2/y]$ does not satisfy 'By ⊃ ~ (∃z)Dyz', because it satisfies the antecedent but fails to satisfy the consequent, and so d_3 does not satisfy '(∀y)(By ⊃ ~ (∃z)Dyz)'.

We conclude that the sentence '(∀y)(By ⊃ ~ (∃z)Dyz)' is therefore false on interpretation 3. It is false because there is at least one member of the UD that is a prime number and that *is* greater than at least one member of the UD.

On the other hand, the sentence '(∃y)(By ⊃ ~ (∃z)Dyz)', in which the universal quantifier of the previous sentence '(∀y)(By ⊃ ~ (∃z)Dyz)' has been

replaced by an existential quantifier, is true on interpretation 3. Again, let d_3 be an arbitrary variable assignment for interpretation 3.

- d_3 satisfies '$(\exists y)(By \supset \sim (\exists z)Dyz)$' if and only if there is at least one member **u** of the UD such that $d_3[\mathbf{u}/y]$ satisfies the open sentence '$By \supset \sim (\exists z)Dyz$', and the latter is the case if and only if either $d_3[\mathbf{u}/y]$ fails to satisfy 'By' or $d_3[\mathbf{u}/y]$ does satisfy '$(\exists z)Dyz$'.
- $d_3[4/y]$ fails to satisfy 'By', as does any other variant that assigns a non-prime positive integer to 'y'. $d_3[4/y]$ fails to satisfy 'By' because $<d_3[4/y](y)>$, which is $<4>$, is not a member of $I_3(B)$.

Therefore, d_3 satisfies '$(\exists y)(By \supset \sim (\exists z)Dyz)$', so the sentence is true on interpretation 3. We have shown that it is true by finding one member of the UD that does not satisfy the condition expressed by 'By' and that therefore does satisfy the condition expressed by the open sentence '$By \supset \sim (\exists z)Dyz$'. Thus we see that the truth-conditions for the universally quantified sentence '$(\forall y)(By \supset \sim (\exists z)Dyz)$' and the existentially quantified sentence '$(\exists y)(By \supset \sim (\exists z)Dyz)$' are significantly different.

The universally quantified sentence '$(\forall y)(By \supset \sim (\exists z)Dyz)$' is true on interpretation 4:

4. UD: The set of positive integers
   B: $\varnothing$
   D: $\{<\mathbf{u}_1, \mathbf{u}_2>: \mathbf{u}_1 \text{ is less than } \mathbf{u}_2\}$

Consider any variable assignment d_4 for this interpretation:

- d_4 satisfies '$(\forall y)(By \supset \sim (\exists z)Dyz)$' if every variant $d_4[\mathbf{u}/x]$ satisfies '$By \supset \sim (\exists z)Dyz$'.
- Every variant $d_4[\mathbf{u}/x]$ satisfies '$By \supset \sim (\exists z)Dyz$' because every variant $d_4[\mathbf{u}/x]$ fails to satisfy 'By'—$I_4(B)$ is the empty set.

Therefore, '$(\forall y)(By \supset \sim (\exists z)Dyz)$' is true on this interpretation. Note that because no variants satisfy the antecedent, it does not matter whether any variants do or do not satisfy '$\sim (\exists z)Dyz$'. We say that the universally quantified conditional is "trivially true" in cases like this.

As a final example, we may use our definitions to prove that the sentence '$(\forall x)(Bx \equiv \sim Bx)$' is false on *every* interpretation. We will do this by assuming that there is an interpretation on which the sentence is true, and then show that this is impossible. So:

- Suppose that **I** is an interpretation on which '$(\forall x)(Bx \equiv \sim Bx)$' is true. By definition, every variable assignment $\mathbf{d_I}$ for **I** must therefore satisfy '$(\forall x)(Bx \equiv \sim Bx)$'.

- A variable assignment $\mathbf{d_I}$ for **I** satisfies '$(\forall x)(Bx \equiv \sim Bx)$' if and only if each member **u** of **I**'s UD is such that the variant $\mathbf{d_I}[\mathbf{u}/x]$ satisfies '$Bx \equiv \sim Bx$' (by clause 8).
- No variant $\mathbf{d_I}[\mathbf{u}/x]$ of $\mathbf{d_I}$ can satisfy '$Bx \equiv \sim Bx$', because any variant that satisfies 'Bx' will fail to satisfy '$\sim Bx$', and any variant that fails to satisfy 'Bx' will satisfy '$\sim Bx$'. By clause 7, therefore, no variant can satisfy '$Bx \equiv \sim Bx$'.

So no variable assignment $\mathbf{d_I}$ can satisfy '$(\forall x)(Bx \equiv \sim Bx)$' on interpretation **I**, no matter what interpretation **I** may be, and this sentence must therefore be false on every interpretation.

In the rest of this chapter, we will generally not pause to talk about variable assignments or the **n**-tuples of objects that constitute the interpretations of predicates. It should be clear that these constructs allow us to rigorously capture an informal notion of objects satisfying open sentences, and henceforth, we shall generally speak in this more informal manner. Do remember, however, that there is a full-fledged formal semantics for *PL* that rigorously captures the informal talk.

Although we have used sets of positive integers as the UDs for each of the interpretations in this section, it was not necessary to do so. Any nonempty set of "things", abstract or concrete, can serve as the UD for an interpretation. In fact, the exercises for this section include interpretations that have UDs containing things other than positive integers. However, because the focus of this chapter is semantics rather than symbolization, it is convenient to stick with a single type of UD with well-defined properties in our examples, and we shall continue to do so in the remainder of this chapter. We have collected some simple facts about the positive integers in Appendix 1, and the reader may find it particularly useful to consult this appendix when attempting to construct interpretations for sentences and sets of sentences of *PL*. In the next section, we will provide an additional reason for using sets of positive integers for the UDs in our interpretations.

Also, for the sake of readability, we will usually present interpretations of predicates in the manner of symbolization keys, rather than using the explicit **n**-tuple notation. For example, instead of writing

D: $\{<\mathbf{u}_1, \mathbf{u}_2>$: $\mathbf{u}_1$ is less than $\mathbf{u}_2\}$,

we will write

Dxy: x is less than y.

Again, however, do remember that we are specifying extensions for predicates and that although the English reading is used to specify the extension, it is not officially part of the interpretation.

Finally, we note that an interpretation on which a sentence is true is called a '**model**' of that sentence, and we shall sometimes make use of this term in the chapters that follow.

8.1**E** EXERCISES

1. Determine the truth-value of the following sentences on this interpretation:

UD: The set of all integers
A: {<**u**>: **u** is positive}
C: {<**u**>: **u** is negative}
B: {<$\mathbf{u}_1$, $\mathbf{u}_2$>: $\mathbf{u}_1$ is a square root of $\mathbf{u}_2$}
a: 0
b: 39
c: −4

a. Cc & (Ac ∨ Bca)
*b. Ab ⊃ Ab
c. ~ Bcb ⊃ (Bba ∨ ~ Ac)
*d. Cb ≡ (~ Ab ≡ Ac)
e. (Cb & Cc) & ~ Baa
*f. ~ (~ Ab ∨ Cb) ⊃ Baa
g. Baa ≡ [Bca ⊃ (Cb ∨ ~ Ab)]
*h. ~ (Ab ∨ Bcc) & (Cc ⊃ ~ Ac)

2. Determine the truth-value of the following sentences on this interpretation:

UD: The set of countries, cities, and people
B: {<$\mathbf{u}_1$, $\mathbf{u}_2$, $\mathbf{u}_3$>: $\mathbf{u}_1$ is between $\mathbf{u}_2$ and $\mathbf{u}_3$}
D: {<$\mathbf{u}_1$, $\mathbf{u}_2$>: $\mathbf{u}_1$ lives in $\mathbf{u}_2$}
F: {<**u**>: **u** is a large city}
a: Germany
b: the United States
c: Italy
d: the U.S. president
e: Tokyo
f: Rome

a. Fa ⊃ Dda
*b. Ff ⊃ Ddb
c. (~ Babc ∨ ~ Bbac) ∨ ~ Bcab
*d. (Fa ≡ Fe) ⊃ Dde
e. (~ Fe ∨ Ddf) & (Fe ∨ Fb)
*f. Baaa ⊃ Bfff
g. (Dda ∨ Ddc) ∨ (Dde ∨ Ddf)
*h. (Fa ≡ Dda) & ~ (Ddb ⊃ Bccc)

3. For each of the following sentences, construct an interpretation on which the sentence is true.
a. Nad ⊃ ~ Nda
*b. Da ≡ ~ (Fb ∨ Gc)
c. (Lm & ~ Lm) ∨ Chm
*d. ~ (Wab ⊃ (Wbb & Eb))
e. (Ma ∨ Na) ∨ (Mb ∨ Nb)
*f. ~ Fc & [(Fa ⊃ Na) & (Fb ⊃ Nb)]

4. For each of the following sentences, construct an interpretation on which the sentence is false.

a. $(Crs \vee Csr) \vee (Css \vee Crr)$
*b. $(Ka \equiv \sim Ma) \equiv Gh$
c. $(Li \vee Lj) \vee Lm$
*d. $Iap \supset (Ipa \supset Iaa)$
e. $(\sim Ja \equiv Jb) \ \& \ (\sim Jc \equiv \sim Jd)$
*f. $(Ha \vee \sim Ha) \supset (Fbb \supset Fba)$

5. For each of the following pairs of sentences, construct an interpretation on which one sentence is true and the other false.

a. $Fab \supset Fba$, $Fba \supset Fab$
*b. $(Caa \ \& \ Cab) \vee Da$, $\sim Da \equiv \sim (Caa \ \& \ Cab)$
c. $\sim Ma \vee Cpqr$, $Capq \vee \sim Mr$
*d. $Kac \vee Kad$, $Kac \ \& \ Kad$
e. $\sim Ljk \equiv (Mjk \vee Mkj)$, $(Mjk \ \& \ Mkj) \ \& \ Ljk$
*f. $Fab \supset (Fbc \supset Fac)$, $Fac \supset (Fcb \supset Fab)$

6. Determine the truth-value of the following sentences on this interpretation:

UD: The set of people
B: $\{<\mathbf{u}>$: **u** is a child$\}$
C: $\{<\mathbf{u}>$: **u** is over 40 years old$\}$
D: $\{<\mathbf{u}_1, \mathbf{u}_2>$: $\mathbf{u}_1$ and $\mathbf{u}_2$ are sisters$\}$
F: $\{<\mathbf{u}_1, \mathbf{u}_2>$: $\mathbf{u}_1$ and $\mathbf{u}_2$ are brothers$\}$

a. $(\forall w)(Cw \supset (\exists x)Dxx)$
*b. $(\exists x)(\exists y)(Fxy \ \& \ Cx)$
c. $(\exists x)(\forall y)(By \vee Fxy)$
*d. $(\forall x)(\forall y)(Dxy \equiv Fxy)$
e. $(\exists x)Cx \supset ((\exists x)(\exists y)Fxy \supset (\exists y)By)$
*f. $\sim (\forall w)(Cw \vee Bw)$
g. $(\forall x)Bx \supset (\forall x)Cx$
*h. $(\forall x)[(\exists y)(Dxy \vee Fxy) \supset Bx]$
i. $(\exists x)[Cx \vee (\exists y)(Dxy \ \& \ Cy)]$
*j. $(\forall w)((Cw \vee Bw) \supset (\exists y)Fwy)$

7. Determine the truth-value of each of the following sentences on this interpretation:

UD: The set of U.S. presidents
A: $\{<\mathbf{u}>$: **u** was the first U.S. president$\}$
B: $\{<\mathbf{u}>$: **u** is a female$\}$
U: $\{<\mathbf{u}>$: **u** is a U.S. citizen$\}$
D: $\{<\mathbf{u}_1, \mathbf{u}_2>$: $\mathbf{u}_1$ held office after $\mathbf{u}_2$'s first term of office$\}$
g: George Washington

a. $(\forall w)Dwg$
*b. $(\forall x)(\forall y)((Bx \ \& \ Ay) \supset Dyx)$
c. $(\exists x)(Ax \ \& \ (\exists y)Dyx)$
*d. $((\exists x)Ax \ \& \sim (\exists z)Bz) \ \& \ (Ag \supset (\forall y)Uy)$
e. $(\forall y)(Uy \supset (\exists x)(Dyx \vee Dxy))$

*f. $(\forall w)(Bw \equiv \sim Uw)$
g. $(\forall x)(Dxg \supset (\exists y)(\sim Uy \ \& \ Dxy))$
*h. $(\exists x)(Ax \ \& \ Bx) \equiv (\forall y)(Ay \supset Uy)$
i. $\sim (Bg \vee (\exists x)(\forall y)Dxy)$
*j. $(\forall y)((By \ \& \ Ay) \supset Dgy)$

8. Determine the truth-value of each of the following sentences on this interpretation:

UD: The set of positive integers
B: $\{<\mathbf{u}>: \mathbf{u}$ is even$\}$
G: $\{<\mathbf{u}_1, \mathbf{u}_2>: \mathbf{u}_1$ is greater than $\mathbf{u}_2\}$
E: $\{<\mathbf{u}_1, \mathbf{u}_2>: \mathbf{u}_1$ equals $\mathbf{u}_2\}$
M: $\{<\mathbf{u}_1, \mathbf{u}_2, \mathbf{u}_3>: \mathbf{u}_1$ minus $\mathbf{u}_2$ equals $\mathbf{u}_3\}$
a: 1
b: 2
c: 3

a. $Bb \ \& \ (\forall w)(Gwb \supset \sim Ewb)$
*b. $(\forall x)(\forall z)(\sim Exz \equiv Gxz)$
c. $(\forall x)(\forall z)(Gxz \supset \sim Exz)$
*d. $(\forall x)(\exists w)(Gwx \ \& \ (\exists z)Mzxw)$
e. $\sim (\forall w)(\forall y)Gwy \supset Mcba$
*f. $(\forall y)(Eya \vee Gya)$
g. $(\forall z)(Bz \supset \sim (\exists y)(By \ \& \ Mzay))$
*h. $(\forall y)[(Bb \ \& \ (\exists x)Exb) \supset Mcby]$
i. $(\forall x)(Exx \equiv \sim (\exists y)(\exists z)Myzx)$
*j. $(\exists x)((Bx \ \& \ Gxc) \ \& \sim (\exists z)Mxcz)$

8.2 QUANTIFICATIONAL TRUTH, FALSEHOOD, AND INDETERMINACY

Using the concept of an interpretation, we may now specify the quantificational versions of the core logical concepts. Here are the quantificational properties that individual sentences of *PL* may have:

A sentence **P** of *PL* is *quantificationally true* if and only if **P** is true on every interpretation.

A sentence **P** of *PL* is *quantificationally false* if and only if **P** is false on every interpretation.

A sentence **P** of *PL* is *quantificationally indeterminate* if and only if **P** is neither quantificationally true nor quantificationally false.

The sentence '$(\exists x)(Gx \vee \sim Gx)$' is quantificationally true. We cannot hope to show this by going through every interpretation of the sentence,

since there are infinitely many. (To see this, it suffices to note that there are infinitely many possible universes of discourse for the sentence.) However, we may reason about the sentence as follows: Because the sentence is existentially quantified, it is true on an interpretation just in case at least one member x of the UD satisfies the condition specified by 'Gx ∨ ~ Gx'—that is, just in case at least one member of the UD either is or is not in the extension of 'G'. Without knowing what the interpretation of 'G' is, we know that every member x of the UD satisfies this condition. And since by definition every interpretation has a nonempty UD, we know that the UD for any interpretation has at least one member and thus at least one member that satisfies the condition specified by the open sentence 'Gx ∨ ~ Gx'. Therefore, '(∃x)(Gx ∨ ~ Gx)' is true on every interpretation.

In general, to show that a sentence of *PL* is quantificationally true, we must use reasoning that shows that, no matter what the UD is and no matter how the sentence letters, predicates, and individual constants are interpreted, the sentence always turns out to be true. Here is another example. The sentence

(∃x)(∃y)(Gxy ⊃ (∀z)(∀w)Gzw)

is quantificationally true. That is, given any interpretation, there are always members x and y of the UD that satisfy the condition specified by 'Gxy ⊃ (∀z)(∀w)Gzw'. The sentence claims that there is a pair x and y of members of the UD such that <u>if</u> x and y are in the extension of 'G' <u>then</u> *all* pairs of members of the UD are in the extension of 'G'. We will consider two possibilities: Either every pair of members of the UD is in the extension of 'G' or not every pair is in the extension of 'G'.

If every pair *is* in the extension of 'G', then every pair x and y (and thus at least one pair) satisfies the condition specified by 'Gxy ⊃ (∀z)(∀w)Gzw' because the consequent is true in this case. Now consider the other possibility—that some (at least one) pair is not in the extension of 'G'. In this case, x and y satisfy the condition specified by 'Gxy ⊃ (∀z)(∀w)Gzw' because they *fail* to satisfy the antecedent 'Gxy'. Because either the extension of 'G' includes every pair of members of the UD or it does not, we have just shown that whatever the extension of 'G' may be, there will always be at least one pair of members of the UD that satisfies 'Gxy ⊃ (∀z)(∀w)Gzw'. This being so, the sentence '(∃x)(∃y)(Gxy ⊃ (∀z)(∀w)Gzw)' is true on every interpretation and is therefore quantificationally true.

The sentence

(∀y)By & (∃z) ~ Bz

is quantificationally false. If an interpretation makes the first conjunct true, then every 1-tuple containing a member of the UD will be in the extension of 'B'. But if this is so, then no member of the UD satisfies the condition specified by '~ Bz', and so the existentially quantified second conjunct is false. So

on any interpretation on which the first conjunct is true, the entire sentence is false. Obviously, the sentence is also false on any interpretation on which the first conjunct is false. It follows that the sentence '$(\forall y)By \ \& \ (\exists z) \sim Bz$' is false on every interpretation.

The sentence

$$(\forall x)(\exists y)(Fx \supset Gy) \equiv ((\exists x)Fx \ \& \ (\forall y) \sim Gy)$$

is also quantificationally false. Because the sentence is a biconditional, it is false on any interpretation on which its immediate components have different truth-values, and we can show that this is the case for every interpretation. Consider first an interpretation on which the immediate component '$(\forall x)(\exists y)(Fx \supset Gy)$' is true. In this case, every member x of the UD must satisfy the condition specified by '$(\exists y)(Fx \supset Gy)$'. That is, every member x must be such that <u>if</u> x is in the extension of 'F' <u>then</u> there is some member y of the UD that is in the extension of 'G'. In this case, the second immediate component of the biconditional, '$(\exists x)Fx \ \& \ (\forall y) \sim Gy$', cannot be true. If it *were* true, then some member x of the UD would be in the extension of 'F' (to satisfy the first conjunct), and no member y of the UD would be in the extension of 'G'. But the truth of '$(\forall x)(\exists y)(Fx \supset Gy)$', as we have seen, requires that if any object is in the extension of 'F' then at least one object must be in the extension of 'G'. It follows that if '$(\forall x)(\exists y)(Fx \supset Gy)$' is true on an interpretation, then '$((\exists x)Fx \ \& \ (\forall y) \sim Gy)$' is false on that interpretation.

Now let us consider an interpretation on which '$(\forall x)(\exists y)(Fx \supset Gy)$' is false. In this case some member x of the UD must fail to satisfy the condition specified by '$(\exists y)(Fx \supset Gy)$'—x must be in the extension of 'F' (to satisfy the antecedent of the conditional), and the extension of 'G' must be empty (so the consequent is not satisfied). But in this case '$((\exists x)Fx \ \& \ (\forall y) \sim Gy)$' must be true because both conjuncts are true. '$(\exists x)Fx$' is true because something is in the extension of 'F', and '$(\forall y) \sim Gy$' is true because the extension of 'G' is empty. So any interpretation that makes '$(\forall x)(\exists y)(Fx \supset Gy)$' false makes '$((\exists x)Fx \ \& \ (\forall y) \sim Gy)$' true. Combined with the results of the previous paragraph, this establishes that on any interpretation the immediate components of '$(\forall x)(\exists y)(Fx \supset Gy) \equiv ((\exists x)Fx \ \& \ (\forall y) \sim Gy)$' have different truth-values. So the biconditional is false on every interpretation and therefore is quantificationally false.

Unfortunately, it is not always so easy to show that a sentence is quantificationally true or that it is quantificationally false. However, because a quantificationally true sentence must be true on every interpretation, we can show that a sentence is *not* quantificationally true by showing that it is false on at least one interpretation. Take as an example the sentence

$$(Ga \ \& \ (\exists z)Bz) \supset (\forall x)Bx$$

This sentence is not quantificationally true. To show this, we shall construct an interpretation on which the sentence is false. The sentence is a material

conditional, and so our interpretation must make its antecedent true and its consequent false. For the antecedent to be true, 'Ga' must be true and at least one member of the UD must be in the extension of 'B'. For the consequent to be false, at least one member of the UD must fail to be in the extension of 'B'. Using the set of positive integers as our UD, we shall interpret 'G' and 'a' so that 'Ga' comes out true, and we shall interpret 'B' so that at least one member of the UD, but not all, falls into the extension of 'B'. The following interpretation will do the trick:

5. UD: The set of positive integers
Gx: x is odd
Bx: x is prime
a: 1

The antecedent '(Ga & (∃z)Bz)' is true because the integer 1 is odd and at least one positive integer is prime, but '(∀x)Bx' is false because not all positive integers are prime.

As a second example, '(∀x)[(Fx ∨ Gx) ∨ (∃y)Hxy]' is not quantificationally true. We shall show this by constructing an interpretation on which the sentence is false. Because the sentence is universally quantified, the UD must have at least one member that fails to satisfy the condition specified by '(Fx ∨ Gx) ∨ (∃y)Hxy'. We choose the set of positive integers as our UD and choose 2 as the member of the UD that does not satisfy the condition. (There is no particular reason for using 2, but choosing an integer helps us develop the rest of the interpretation.) We interpret 'F' and 'G' so that the integer 2 has neither property (otherwise it would satisfy either 'Fx' or 'Gx'). We must also interpret 'H' so that the integer 2 does not stand in the relation H to any positive integer:

6. UD: The set of positive integers
Fx: x is odd
Gx: x is greater than 4
Hxy: x is equal to y squared

Because 2 is neither odd nor greater than 4, and it is not the square of any positive integer, it fails to satisfy the condition specified by '(Fx ∨ Gx) ∨ (∃y)Hxy'. Therefore the universally quantified sentence is false on interpretation 6. Having shown that there is at least one interpretation on which the sentence is false, we may conclude that it is not quantificationally true.

We may show that a sentence is not quantificationally *false* by constructing an interpretation on which it is true. The sentence

~ (~ Ga & (∃y)Gy)

is not quantificationally false. To construct an interpretation on which it is true, we must make '~ Ga & (∃y)Gy' false. To do so, we must make one or

both conjuncts false. We choose the former and interpret 'G' and 'a' so that '~ Ga' is false:

7. UD: The set of positive integers
Gx: x is even
a: 2

Because the integer 2 is even, 'Ga' is true. Hence '~ Ga' is false and so is '~ Ga & (∃y)Gy'. (The fact that the second conjunct turns out to be true on our interpretation is irrelevant—the conjunction as a whole is still false.) Therefore '~ (~ Ga & (∃y)Gy)' is true on interpretation 7, and we may conclude that the sentence is not quantificationally false.

It is important to remember that we cannot show that a sentence *is* quantificationally true or that it *is* quantificationally false by constructing a single interpretation. To show that a sentence is quantificationally true, we must demonstrate that it is true on every interpretation, and to show that a sentence is quantificationally false, we must show that it is false on every interpretation.

A quantificationally indeterminate sentence is one that is neither quantificationally true nor quantificationally false. We may show that a sentence is quantificationally indeterminate by constructing two interpretations: one on which it is true and one on which it is false. The sentence

~ (~ Ga & (∃y)Gy)

is quantificationally indeterminate. We have already constructed an interpretation (interpretation 7) on which it is true; all that is left is to construct an interpretation on which it is false. For the sentence to be false, '~ Ga & (∃y) Gy' must be true. To make '~ Ga' true, our UD must contain at least one member that is not in the extension of 'G', and 'a' will designate this member. But the UD must also contain a member that *is* in the extension of 'G', to make '(∃y)Gy' true:

8. UD: The set of positive integers
Gx: x is odd
a: 2

The integer 2 is not odd, but at least one positive integer is, and so '~ Ga & (∃y)Gy' is true and '~ (~ Ga & (∃y)Gy)' is false on interpretation 8. The sentence is therefore not quantificationally true. Having shown that the sentence is neither quantificationally true nor quantificationally false, we may conclude that it is quantificationally indeterminate.

Sometimes it takes ingenuity to find an interpretation on which a sentence is true or an interpretation on which a sentence is false. Examine the sentence itself for guidelines, as we have just done. If it is a truth-functional compound, then use your knowledge of the truth-conditions for that type of compound. If the sentence is universally quantified, then the sentence will be

true if and only if the condition specified after the quantifier is satisfied by all members of the UD you choose. If the sentence is existentially quantified, then it will be true if and only if the condition specified after the quantifier is satisfied by at least one member of the UD. As you examine the components of the sentence, you may reason in the same way—are they truth-functional compounds or quantified? Sometimes the desired interpretation cannot be obtained. For example, a quantificationally true sentence is not false on any interpretation; therefore any attempt to construct an interpretation that makes the sentence false will fail.

Two theoretical points are of interest here. The first is that, if a sentence of predicate logic without identity is true on at least one interpretation, then it is true on some interpretation that has the set of positive integers as its UD. This result is known as the *Löwenheim Theorem* (it will be proved in the exercises in Chapter 11). It follows from this result that, if a sentence of *PL* is true on some interpretation with a finite UD, then it is true on some interpretation that has the set of positive integers as its UD. And if a sentence of *PL* is true on some interpretation for which the UD is *larger* than the set of positive integers (for example, the set of real numbers), then it is true on at least one interpretation that has the set of positive integers as its UD.

Note that this result means that the set of positive integers is always a good choice for our UD as we construct interpretations. In fact, there are sentences of *PL* that are not quantificationally true but that are nevertheless true on every interpretation with a finite UD, and there are sentences of *PL* that are not quantificationally false but are false on every interpretation with a finite UD. For example, the following sentence is not quantificationally false:

$$(\forall x)(\forall y)(\forall z)[(Bxy \ \& \ Byz) \supset Bxz] \ \& \ [(\forall x)(\exists y)Bxy \ \& \ (\forall z) \sim Bzz]$$

But it is false on every interpretation with a finite UD. To show that it is not quantificationally false, then, we must choose a UD that has infinitely many members—and the set of positive integers is a good choice.

The second point is that there is no decision procedure that will tell us, for each sentence of *PL*, whether that sentence is quantificationally true, quantificationally false, or quantificationally indeterminate. (We shall not prove the result in this text.) This is a very important way in which the semantics for *PL* differs from the semantics for *SL*. For *SL,* the construction of truth-tables gives a decision procedure for determining whether a sentence is truth-functionally true, false, or indeterminate. That is, in a finite number of mechanical steps, we can always correctly answer the questions 'Is this sentence truth-functionally true?', 'Is this sentence truth-functionally false?', and 'Is this sentence truth-functionally indeterminate?' Alonzo Church proved that there is no analogous method for predicate logic—we have no such general method now, and no such general method will ever be found. This result does not mean that we cannot ever show that some sentences of *PL* are quantificationally true, false, or indeterminate; rather, it shows that there is no *decision* procedure (mechanical, certain, and requiring only a finite number of steps) for determining the quantificational

status of *every* sentence of *PL*. However, it is interesting to note that there *is* such a procedure for determining the quantificational status of sentences of *PL* that contain no many-place predicates, that is, in which the predicates are all one-place predicates. This follows from a result by the logicians Bernays and Schönfinkel.[2]

8.2.**E** EXERCISES

For these exercises, when you are asked to construct interpretations, you may specify the interpretations either as they are specified in Section 8.1 or in the manner of symbolization keys.

1. Show that each of the following sentences is not quantificationally true by constructing an interpretation on which it is false.

a. $(\forall x)(Fx \supset Gx) \supset (\forall x)Gx$
*b. $(\exists x)(Fx \vee Gx) \supset ((\exists x)Fx \supset (\exists x) \sim Gx)$
c. $(\forall x)(\exists y)Bxy \supset (\exists y)(\forall x)Bxy$
*d. $(\forall x)(Fxb \vee Gx) \supset [(\forall x)Fxb \vee (\forall x)Gx]$
e. $[(\forall x)Fx \supset (\forall w)Gw] \supset (\forall z)(Fz \supset Gz)$
*f. $(\forall x)(Ax \supset (\forall y)By) \supset (\forall y)(By \supset (\forall x)Ax)$
g. $\sim (\exists x)Gx \supset (\forall y)(Fyy \supset Gy)$
*h. $(\forall x)(Bx \equiv Hx) \supset (\exists x)(Bx \mathbin{\&} Hx)$

2. Show that each of the following sentences is not quantificationally false by constructing an interpretation on which it is true.

a. $\sim (\forall w)(\forall y)Bwy \equiv (\forall z)Bzz$
*b. $(\exists x)Fx \mathbin{\&} (\exists x) \sim Fx$
c. $((\exists x)Fx \mathbin{\&} (\exists x)Gx) \mathbin{\&} \sim (\exists x)(Fx \mathbin{\&} Gx)$
*d. $(\exists x)((\exists y)Fy \supset \sim Fx)$
e. $(\forall x)(Fx \supset Gx) \mathbin{\&} (\forall x)(Gx \supset \sim Fx)$
*f. $(\exists x)(\forall y)(Dyx \supset \sim Dxy)$
g. $(\forall x)(Bx \equiv Hx) \supset (\exists x)(Bx \mathbin{\&} Hx)$
*h. $(\exists x)(\forall y)Dxy \vee \sim (\forall y)(\exists x)Dxy$
i. $(\forall x)(\forall y)(\forall z)\ [(Bxy \mathbin{\&} Byz) \supset Bxz] \mathbin{\&} [(\forall x)(\exists y)Bxy \mathbin{\&} (\forall z) \sim Bzz]$

3. Show that each of the following sentences is quantificationally indeterminate by constructing an interpretation on which it is true and an interpretation on which it is false.

a. $(\exists x)(Fx \mathbin{\&} Gx) \supset (\exists x) \sim (Fx \vee Gx)$
*b. $(\exists x)Fx \supset (\forall w)(Cw \supset Fw)$
c. $(\forall x)Bnx \supset (\forall x) \sim Bnx$
*d. $(\exists x)(Fx \supset Gx) \supset (\exists x)(Fx \mathbin{\&} Gx)$
e. $(\forall x)(\forall w)[(Nwx \vee Nxw) \supset Nww]$
*f. $(Ma \mathbin{\&} Mb) \mathbin{\&} (\exists x) \sim Mx$
g. $(\forall x)(Cx \vee Dx) \equiv (\exists y)(Cy \mathbin{\&} Dy)$
*h. $[\sim (\exists x)Hx \vee \sim (\exists x)Gx] \vee (\forall x)(Hx \mathbin{\&} Gx)$

[2]"Zum Entscheidungsproblem der mathematischen Logik," *Mathematische Annalen,* 99 (1928), 342–372. Alonzo Church's proof about the general case appeared in "A Note on the Entscheidungsproblem," *Journal of Symbolic Logic*, 1 (1936), 40–41, 101–102.

4. Each of the following sentences is quantificationally true. Explain why.

a. $(\exists x)(\forall y)Bxy \supset (\forall y)(\exists x)Bxy$

*b. $[(\forall x)Fx \vee (\forall x)Gx] \supset (\forall x)(Fx \vee Gx)$

c. $Fa \vee [(\forall x)Fx \supset Ga]$

*d. $(\forall x)(\exists y)Mxy \supset (\exists x)(\exists y)Mxy$

e. $(\exists x)Hx \vee (\forall x)(Hx \supset Jx)$

5. Each of the following sentences is quantificationally false. Explain why.

a. $(\exists w)(Bw \equiv \sim Bw)$

*b. $(\forall w)(Fw \supset Gw) \;\&\; [(\forall w)(Fw \supset \sim Gw) \;\&\; (\exists w)Fw]$

c. $[(\forall x)Fx \supset (\exists y)Gy] \;\&\; [\sim (\exists x)Gx \;\&\; \sim (\exists x) \sim Fx]$

*d. $(\exists x)(Fx \;\&\; \sim Gx) \;\&\; (\forall x)(Fx \supset Gx)$

e. $((\forall w)(Aw \supset Bw) \;\&\; (\forall w)(Bw \supset Cw)) \;\&\; (\exists y)(Ay \;\&\; \sim Cy)$

6. For each of the following sentences, decide whether it is quantificationally true, quantificationally false, or quantificationally indeterminate. If the sentence is quantificationally true or quantificationally false, explain why. If it is quantificationally indeterminate, construct interpretations that establish this.

a. $((\exists x)Gx \;\&\; (\exists y)Hy) \;\&\; (\exists z) \sim (Gz \;\&\; Hz)$

*b. $((\exists x)Gx \;\&\; (\exists y)Hy) \;\&\; \sim (\exists z)(Gz \;\&\; Hz)$

c. $(\forall x)(Fx \supset Gx) \supset (\forall x)(\sim Gx \supset \sim Fx)$

*d. $(\forall x)Fx \supset \sim (\exists x) \sim (Fx \vee Gx)$

e. $(\forall x)(Dx \supset (\exists z)Hxz) \supset (\exists z)(\forall x)(Dx \supset Hxz)$

*f. $(\exists z)(\forall x)(Dx \supset Hxz) \supset (\forall x)(Dx \supset (\exists z)Hxz)$

8.3 QUANTIFICATIONAL EQUIVALENCE AND CONSISTENCY

The next concept to be introduced is that of quantificational equivalence.

> Sentences **P** and **Q** of *PL* are *quantificationally equivalent* if and only if there is no interpretation on which **P** and **Q** have different truth-values.

The sentences

$$(\exists x)Fx \supset Ga$$

and

$$(\forall x)(Fx \supset Ga)$$

are quantificationally equivalent. We may reason as follows: First suppose that '$(\exists x)Fx \supset Ga$' is true on some interpretation. Then '$(\exists x)Fx$' is either true or false on this interpretation. If it is true, then so is 'Ga' (by our assumption that '$(\exists x)Fx \supset Ga$' is true). But then, since 'Ga' is true, every object x in the UD is such that <u>if</u> x is F <u>then</u> a is G. So '$(\forall x)(Fx \supset Ga)$' is true. If '$(\exists x)Fx$' is false, however, then every object x in the UD is such that <u>if</u> x is F (which, on

our assumption, it is not) then a is G. Again '$(\forall x)(Fx \supset Ga)$' is true. Hence, if '$(\exists x)Fx \supset Ga$' is true on an interpretation, '$(\forall x)(Fx \supset Ga)$' is also true on that interpretation.

Now suppose that '$(\exists x)Fx \supset Ga$' is false on some interpretation. Since the sentence is a conditional, it follows that '$(\exists x)Fx$' is true and 'Ga' is false. But if '$(\exists x)Fx$' is true, then there is at least one object x in the UD in the extension of 'F'. This object then does not satisfy the condition that if it is F (which it is) then a is G (which is false on our present assumption). So '$(\forall x)(Fx \supset Ga)$' is false if '$(\exists x)Fx \supset Ga$' is. Taken together with our previous result, this demonstrates that the two sentences are quantificationally equivalent.

The sentences

$$\sim (\exists x)(\forall y)(Gxy \vee Gyx)$$

and

$$(\forall x)(\exists y)(\sim Gxy \,\&\, \sim Gyx)$$

are also quantificationally equivalent. As in the previous example, we will show that if the first sentence is true on an interpretation then so is the second sentence, and that if the first sentence is false on an interpretation then so is the second sentence. First consider an interpretation on which '$\sim (\exists x)(\forall y)(Gxy \vee Gyx)$' is true. '$(\exists x)(\forall y)(Gxy \vee Gyx)$' must be false on this interpretation, so no member x of the UD satisfies the condition specified by '$(\forall y)(Gxy \vee Gyx)$'. That is, no member x of the UD is such that for every object y either the pair x and y or the pair y and x is in the extension of 'G'. Put another way, for each member x of the UD, there is at least one object y such that both '$\sim Gxy$' and '$\sim Gyx$' hold. And that is exactly what the second sentence says, so it is true as well.

Now consider an interpretation on which the first sentence is false; '$(\exists x)(\forall y)(Gxy \vee Gyx)$' is true on such an interpretation. So there is at least one member x of the UD such that for every object y, either 'Gxy' or 'Gyx' holds. Therefore, x does *not* satisfy the condition specified by '$(\exists y)(\sim Gxy \,\&\, \sim Gyx)$' (because there is *no* y such that neither 'Gxy' nor 'Gyx' holds). And so the universally quantified sentence '$(\forall x)(\exists y)(\sim Gxy \,\&\, \sim Gyx)$' is also false. From this and the result of the preceding paragraph, we conclude that the two sentences are quantificationally equivalent.

If we want to establish that two sentences are *not* quantificationally equivalent, we can construct an interpretation to show this. The interpretation must make one of the sentences true and the other sentence false. For example, the sentences

$$(\forall x)(Fx \supset Ga)$$

and

$$(\forall x)Fx \supset Ga$$

are not quantificationally equivalent. We shall construct an interpretation on which the first sentence is false and the second sentence is true. To make the first sentence false, 'Ga' has to be false, and at least one object x must be in the extension of 'F'—for then x will fail to satisfy 'Fx ⊃ Ga'. But we can still make '(∀x)Fx ⊃ Ga' true—because then the antecedent '(∀x)Fx' will be false. Here is our interpretation:

9. UD: The set of positive integers
Fx: x is prime
Gx: x is even
a: 1

The integer 3 (for example) does not satisfy the condition that <u>if</u> it is prime (which it is) <u>then</u> the number 1 is even (which is false). So '(∀x)(Fx ⊃ Ga)' is false on the interpretation. But '(∀x)Fx ⊃ Ga' is true because its antecedent, '(∀x)Fx', is false—not every positive integer is prime. Once again we see that the scope of quantifiers is very important in determining the truth-conditions of sentences of *PL*.

The sentences

(∀x)(∃y)(Hy ⊃ Lx)

and

(∀x)[(∃y)Hy ⊃ Lx]

are also not quantificationally equivalent. We shall show this by constructing an interpretation on which the first sentence is true and the second sentence is false. To make '(∀x)[(∃y)Hy ⊃ Lx]' false, some member of the UD must fail to satisfy '(∃y)Hy ⊃ Lx'. Therefore the UD must contain at least one object in the extension of 'H' (so that '(∃y)Hy' is satisfied) and at least one object x that is not in the extension of 'L' (so that this object does not satisfy 'Lx'). To make '(∀x)(∃y)(Hy ⊃ Lx)' true, every member of the UD must satisfy '(∃y)(Hy ⊃ Lx)'—for every member x of the UD, there must be an object y such that <u>if</u> y is H <u>then</u> x is L. We have already decided that at least one object x will not be in the extension of 'L'. So, if x (along with all other members of the UD) is to satisfy '(∃y)(Hy ⊃ Lx)', then there must be at least one member y of the UD that is not in the extension of 'H'—for then y will be such that if it is H (which it will not be) then x is L.

To summarize, we need at least one object that is in the extension of 'L' and at least one object that is not in the extension of 'H'. Here is our interpretation:

10. UD: The set of positive integers
Hx: x is odd
Lx: x is prime

The sentence '$(\forall x)[(\exists y)Hy \supset Lx]$' is false—every positive integer x that is not prime fails to satisfy the condition that if some positive integer is odd (which at least one positive integer is) then x is prime. The sentence '$(\forall x)(\exists y)(Hy \supset Lx)$' is true because at least one positive integer is not odd. So for any positive integer x there is at least one positive integer y that is not odd, and hence at least one positive integer y such that if y is odd (which y is not) then x is prime.

While we may construct single interpretations to show that two sentences are not quantificationally equivalent, we may not use the same method to show that sentences *are* quantificationally equivalent. In the latter case we must reason about every interpretation as we did in the examples at the beginning of this section.

Quantificational consistency is our next concept.

> A set of sentences of *PL* is *quantificationally consistent* if and only if there is at least one interpretation on which all the members of the set are true. A set of sentences of *PL* is *quantificationally inconsistent* if and only if the set is not quantificationally consistent.

The set of sentences

$$\{(\forall x)Gax, \sim Gba \vee (\exists x) \sim Gax\}$$

is quantificationally consistent. The following interpretation shows this:

11. UD: The set of positive integers
Gxy: x is less than or equal to y
a: 1
b: 2

On this interpretation '$(\forall x)Gax$' is true since 1 is less than or equal to every positive integer. '$\sim Gba$' is true since 2 is neither less than nor equal to 1; so '$\sim Gba \vee (\exists x) \sim Gax$' is true. Since both members of the set are true on this interpretation, the set is quantificationally consistent.

The set

$$\{(\forall w)(Fw \supset Gw), (\forall w)(Fw \supset \sim Gw)\}$$

is also quantificationally consistent. This may seem surprising since the first sentence says that everything that is F is G and the second sentence says that everything that is F is not G. But the set is consistent because, if 'F' has an empty extension, then every object w in the UD will be such that if w is F (which w is not) then w is both G and not G. The following interpretation illustrates this.

12. UD: The set of positive integers
Fx: x is negative
Gx: x is even

Each positive integer w is such that if w is negative (which w is not) then w is even, and each positive integer w is such that if w is negative (which w is not) then w is not even. Both '$(\forall w)(Fw \supset Gw)$' and '$(\forall w)(Fw \supset \sim Gw)$' are true on this interpretation.

Note that, while a single interpretation may be produced to show that a set of sentences is quantificationally consistent, a single interpretation *cannot* be used to show that a set of sentences is quantificationally *in*consistent. To show that a set is quantificationally inconsistent, we must show that on every interpretation at least one sentence in the set is false. In some cases simple reasoning shows that a set of sentences is quantificationally inconsistent. The set

$$\{(\exists y)(Fy \ \& \sim Ny),\ (\forall y)(Fy \supset Ny)\}$$

is quantificationally inconsistent. For if '$(\exists y)(Fy \ \& \sim Ny)$' is true on some interpretation then some member y of the UD is F and is not N. But then that member is *not* such that if it is F (which it is) then it is N (which it is not). Hence the universally quantified sentence '$(\forall y)(Fy \supset Ny)$' is false on such an interpretation. So there is no interpretation on which both set members are true; the set is quantificationally inconsistent.

8.3**E** EXERCISES

For these exercises, when you are asked to construct interpretations, you may specify the interpretations either as they are specified in Section 8.1 or in the manner of symbolization keys.

1. Show that the sentences in each of the following pairs are not quantificationally equivalent by constructing an interpretation on which one of the sentences is true and the other is false.

a. $(\exists x)Fx \supset Ga,\ (\exists x)(Fx \supset Ga)$
*b. $(\exists x)Fx \ \& \ (\exists x)Gx,\ (\exists x)(Fx \ \& \ Gx)$
c. $(\forall x)Fx \vee (\forall x)Gx,\ (\forall x)(Fx \vee Gx)$
*d. $(\exists x)(Fx \vee Ga),\ (\exists x)(Fx \vee Gb)$
e. $(\forall x)(Fx \equiv Gx),\ (\exists x)Fx \equiv (\exists x)Gx$
*f. $(\forall x)(Fx \supset Gx),\ (\forall y)((\forall x)Fx \supset Gy)$
g. $(\exists x)(Bx \ \& \ (\forall y)Dyx),\ (\forall x)(Bx \supset (\forall y)Dyx)$
*h. $(\exists y)(My \equiv Ny),\ (\exists y)My \equiv (\exists y)Ny$
i. $(\forall x)(\exists y)(Fx \supset Kyx),\ (\exists x)(\exists y)(Fx \supset Kyx)$

2. In each of the following pairs the sentences are quantificationally equivalent. Explain why.

a. $(\forall x)Fx \supset Ga,\ (\exists x)(Fx \supset Ga)$
*b. $(\forall x)(Fx \supset Gx),\ \sim(\exists x)(Fx \ \& \sim Gx)$
c. $(\exists x)(Fx \vee Gx),\ \sim(\forall y)(\sim Fy \ \& \sim Gy)$
*d. $(\forall x)(\forall y)(Mxy \ \& \ Myx),\ \sim(\exists x)(\exists y)(\sim Mxy \vee \sim Myx)$
e. $(\forall x)(\forall y)Gxy,\ (\forall y)(\forall x)Gxy$
*f. $(\forall x)(\forall y)(Fxy \supset Hyx),\ \sim(\exists x)(\exists y)(Fxy \ \& \sim Hyx)$

3. Decide, for each of the following pairs of sentences, whether the sentences are quantificationally equivalent. If they are quantificationally equivalent, explain why. If they are not quantificationally equivalent, construct an interpretation that shows this.

a. $(\exists x)(Fx \vee Gx)$, $(\forall x) \sim (Fx \,\&\, Gx)$
*b. $(\exists x)(Fx \,\&\, Gx)$, $\sim (\forall x) \sim (Fx \vee Gx)$
c. $(\forall w)(\forall y)(Gyw \vee Gwy)$, $(\forall w)(\forall y)(Gww \vee Gwy)$
*d. $(\forall y)((\exists z)Hzy \supset Hyy)$, $(\forall y)((\exists z)(Hzz \supset Hzy)$

4. Show that each of the following sets of sentences is quantificationally consistent by constructing an interpretation on which every member of the set is true.

a. $\{(\exists x)Bx, (\exists x)Cx, \sim (\forall x)(Bx \vee Cx)\}$
*b. $\{(\exists x)Fx \vee (\exists x)Gx, (\exists x) \sim Fx, (\exists x) \sim Gx\}$
c. $\{(\forall x)(Fx \supset Gx), (\forall x)(Nx \supset Mx), (\forall x)(Gx \supset \sim Mx)\}$
*d. $\{(\forall x)(Dax \equiv Bax), \sim Dab, \sim Bba\}$
e. $\{(\forall w)(Nw \supset (\exists z)(Mz \,\&\, Cwz)), (\forall z)(\forall w)(Mz \supset \sim Cwz)\}$
*f. $\{(\exists w)Fw, (\forall w)(Fw \supset (\exists x)Bxw), (\forall x) \sim Bxx\}$
g. $\{\sim (\forall y)(Ny \supset My), \sim (\forall y) \sim (Ny \supset My)\}$
*h. $\{(\forall x)(Bx \equiv (\forall y)Cxy), (\exists x) \sim Bx, (\exists x)(\exists y)Cxy\}$
i. $\{(\exists y)Fay, (\exists y) \sim Gay, (\forall y)(Fay \vee Gay)\}$

5. Each of the following sets of sentences is quantificationally inconsistent. Explain why.

a. $\{(\exists x)(Bx \,\&\, Cx), (\forall x) \sim (Bx \vee Cx)\}$
*b. $\{(\exists x)(\exists y)(Bxy \vee Byx), \sim (\exists x)(\exists y)Bxy\}$
c. $\{(\forall x)(\forall y)(Byx \vee Bxy), (\exists y) \sim Byy\}$
*d. $\{Ba, (\exists y)Day, (\forall x)(Bx \supset (\forall y) \sim Dxy)\}$
e. $\{(\exists x)(\forall y)Gxy, (\forall x)(\forall y) \sim Gxy\}$
*f. $\{(\forall x)Fx \vee (\forall x) \sim Fx, (\exists x)Fx \equiv (\exists x) \sim Fx\}$

6. Decide, for each of the following sets of sentences, whether the set is quantificationally consistent. If the set is quantificationally consistent, construct an interpretation that shows this. If it is quantificationally inconsistent, explain why.

a. $\{(\exists x)Fx \supset (\forall x)Fx, (\exists x) \sim Fx, (\exists x)\ Fx\}$
*b. $\{(\exists x)(\exists y)Gxy, (\forall y) \sim Gyy\}$
c. $\{(\forall x) \sim (\forall y)Gxy, (\forall x)Gxx\}$
*d. $\{(\exists x)Px, (\forall y)(Py \supset Hya), \sim (\forall x) \sim Hxa\}$

7. Explain why sentences **P** and **Q** of *PL* are quantificationally equivalent if and only if $\mathbf{P} \equiv \mathbf{Q}$ is quantificationally true.

8.4 QUANTIFICATIONAL ENTAILMENT AND VALIDITY

Our last two semantic concepts for the language *PL* are the concepts of quantificational entailment and quantificational validity.

> A set Γ of sentences of *PL quantificationally entails* a sentence **P** of *PL* if and only if there is no interpretation on which every member of Γ is true and **P** is false.

An argument of *PL* is *quantificationally valid* if and only if there is no interpretation on which every premise is true and the conclusion is false. An argument of *PL* is *quantificationally invalid* if and only if the argument is not quantificationally valid.

The set

$$\{(\forall x)(Bx \supset Ga),\ (\exists x)Bx\}$$

quantificationally entails the sentence 'Ga'. As in *SL*, we may use the double turnstile and write this as

$$\{(\forall x)(Bx \supset Ga),\ (\exists x)Bx\} \models Ga$$

Suppose that '$(\forall x)(Bx \supset Ga)$' and '$(\exists x)Bx$' are both true on some interpretation. Since '$(\forall x)(Bx \supset Ga)$' is true, we know that every object x in the UD is such that if x is B then a is G. Since '$(\exists x)Bx$' is true, we know that at least one object x in the UD is in the extension of 'B'. Since it is true that if x is B (which it is) then a is G, 'Ga' must therefore be true. So, on any interpretation on which '$(\forall x)(Bx \supset Ga)$' and '$(\exists x)Bx$' are both true, 'Ga' is also true. So the entailment does hold.

The argument

$$\begin{array}{l}(\exists x)(Fx \vee Gx)\\(\forall x) \sim Fx\\\hline(\exists x)Gx\end{array}$$

is quantificationally valid. Suppose that on some interpretation both premises are true. If the first premise is true, then some member x of the UD is either F or G. If the premise '$(\forall x) \sim Fx$' is true, then no member of the UD is F. Therefore, because the member that is either F or G is not F, it must be G. Thus '$(\exists x)Gx$' will also be true on such an interpretation.

We can show that a set of sentences does *not* quantificationally entail a sentence by constructing an interpretation. For example, the set

$$\{\sim(\forall x)(Gx \equiv Fx),\ \sim Fb\}$$

does not quantificationally entail the sentence

$$(\forall x) \sim Gx$$

We will construct an interpretation on which the members of the set are true and '$(\forall x) \sim Gx$' is false. For the sentence '$\sim (\forall x)(Gx \equiv Fx)$' to be true, the UD must contain at least one member x that fails to satisfy '$Gx \equiv Fx$'—x must

be in the extension of one of the two predicates but not in the extension of the other. For '~ Fb' to be true, 'b' must designate an object that is not in the extension of 'F'. And '$(\forall x) \sim Gx$', which claims that everything is not G, will be false if at least one object in the UD is in the extension of 'G'. Here is an interpretation that satisfies these conditions:

13. UD: The set of positive integers
Fx: x is greater than 5
Gx: x is prime
b: 3

Not all positive integers are prime if and only if they are greater than 5—take 2 as an example—and 3 is not greater than 5. Therefore the set members are both true on this interpretation. On the other hand, '$(\forall x) \sim Gx$' is false, because some positive integers are prime.

To show that an argument is quantificationally invalid, we can construct an interpretation on which its premises are true and its conclusion is false. The argument

$$\begin{array}{l} (\exists x)[(\exists y)Fy \supset Fx] \\ (\exists y) \sim Fy \\ \hline \sim (\exists x)Fx \end{array}$$

is quantificationally invalid. We can make the first premise true by interpreting 'F' so that at least one member of the UD is in its extension—for then that object will satisfy the condition specified by '$(\exists y)Fy \supset Fx$' because it will satisfy its consequent. This object will also make the conclusion false. The second premise will be true if at least one member of the UD is not in the extension of 'F'. So the extension of 'F' will include some, but not all, members of the UD. The following interpretation will do the trick:

14. UD: The set of positive integers
Fx: x is prime

Some positive integer x is such that if there exists a prime positive integer then x is prime—for example, the integer 5 satisfies this condition—and some positive integer is not prime, but it is false that no positive integer is prime.

Note that we cannot prove that a quantificational entailment *does* hold or that an argument *is* quantificationally valid by constructing a single interpretation. Proving either of these involves proving something about the truth-value of sentences on every interpretation, not just a select few.

And, once again, there are limitations on deciding questions of quantificational equivalence, consistency, entailment, and validity. Owing to Church's result, mentioned at the end of Section 8.2, we know that there is no procedure for deciding these questions for every group of sentences of

PL.[3] However, our method of producing interpretations to establish quantificational consistency, nonequivalence, nonentailment, and invalidity, although not a decision procedure, often produces the desired result.

8.4**E** EXERCISES

For these exercises, when you are asked to construct interpretations, you may specify the interpretations either as they are specified in Section 8.1 or in the manner of symbolization keys.

1. Establish each of the following by constructing an appropriate interpretation.

a. $\{(\forall x)(Fx \supset Gx),\ (\forall x)(Hx \supset Gx)\} \nvDash (\exists x)(Hx \mathbin{\&} Fx)$

*b. $\{(\forall y)(Fy \equiv Fa),\ Fa\} \nvDash {\sim} Fb$

c. $\{(\exists x)Fx\} \nvDash Fa$

*d. $\{(\forall x)(Bx \supset Cx),\ (\exists x)Bx\} \nvDash (\forall x)Cx$

e. $\{(\exists x)(Bx \supset Cx),\ (\exists x)Cx\} \nvDash (\exists x)Bx$

*f. $\{(\forall x)(Fx \supset Gx),\ (\forall x)(Hx \supset {\sim} Fx)\} \nvDash (\forall x)(Hx \supset Gx)$

g. $\{(\forall x)(\exists y) {\sim} Lxy\} \nvDash (\forall x) {\sim} Lxx$

*h. $\{(\exists x)(\forall y)(Hxy \vee Jxy),\ (\exists x)(\forall y) {\sim} Hxy\} \nvDash (\exists x)(\forall y)Jxy$

2. Show that each of the following arguments is quantificationally invalid by constructing an appropriate interpretation.

a. $(\forall x)(Fx \supset Gx) \supset (\exists x)Nx$
$(\forall x)(Nx \supset Gx)$
$\therefore (\forall x)({\sim} Fx \vee Gx)$

*b. $({\sim} (\exists y)Fy \supset (\exists y)Fy) \vee {\sim} Fa$
$\therefore (\exists z)Fz$

c. $(\exists x)(Fx \mathbin{\&} Gx)$
$(\exists x)(Fx \mathbin{\&} Hx)$
$\therefore (\exists x)(Gx \mathbin{\&} Hx)$

*d. $(\forall x)(Fx \supset Gx)$
Ga
$\therefore Fa$

e. $(\forall x)(Fx \supset Gx)$
${\sim} (\exists x)Fx$
$\therefore {\sim} (\exists x)Gx$

[3]Moreover, some arguments can be proved quantificationally invalid and some sets quantificationally consistent only by means of interpretations with universes of discourse containing an infinite number of members. However, there is a result for sets of sentences analogous to the Löwenheim Theorem (mentioned in Section 8.2), which says that if a set of sentences is quantificationally consistent—or an argument quantificationally invalid—then there are interpretations with the set of positive integers as the UD that show this. It is not necessary in any case to check interpretations with larger universes of discourse. This result is known as the *Löwenheim-Skolem Theorem* and is assigned as an exercise in Chapter 11.

*f. $(\forall x)(\forall y)(Mxy \supset Nxy)$

$(\forall x)(\forall y)(Mxy \supset (Nxy \ \& \ Nyx))$

g. $(\exists x)Gx$

$(\forall x)(Gx \supset Dxx)$

$(\exists x)(\forall y)(Gx \ \& \ Dxy)$

*h. $Fa \vee (\exists y)Gya$

$Fb \vee (\exists y) \sim Gyb$

$(\exists y)Gya$

i. $(\forall x)(Fx \supset Gx)$

$(\forall x)(Hx \supset Gx)$

$(\forall x)(Fx \vee Hx)$

3. Using the given symbolization keys, symbolize the following arguments in *PL*. Then show that the first symbolized argument in each pair is quantificationally valid while the second is not.

a. UD: The set consisting of all living creatures
Bx: x is beautiful
Px: x is a person

Every living creature is beautiful. Therefore some living creature is beautiful.

Everyone is beautiful. Therefore someone is beautiful.

*b. UD: The set of people
Rx: x roller skates
Dx: x can dance

Not everyone can dance. Therefore someone can't dance.

No one who roller skates can dance. Therefore some roller skater can't dance.

c. UD: The set of people
Lxy: x loves y

There is a person who loves everyone. Therefore everyone is loved by someone.

Everyone is loved by someone. Therefore there is a person who loves everyone.

*d. UD: The set of integers
Ex: x is even
Dx: x is divisible by 2

Some integers are even and some integers are divisible by 2. Therefore some integers are even if and only if some integers are divisible by 2.

An integer is even if and only if it is divisible by 2. Therefore some integer is even.

e. UD: The set of people
Tx: x is a student
Sx: x is smart
Hx: x is happy

Some students are smart and some students are not happy. Therefore there is a student who is smart or not happy.

All students are smart, and no student is happy. Therefore there is a student who is smart or not happy.

*f. UD: The set of people
Sx: x is a senator
Rx: x is a Republican
Dx: x is a Democrat

Any senator who is not a Republican is a Democrat. There is a senator who is not a Republican. Therefore some senator is a Democrat.

There is a senator who is not a Republican. Therefore some senator is a Democrat.

g. UD: The set of people
Ax: x likes asparagus
Sx: x likes spinach
Cx: x is crazy

Anyone who likes asparagus is crazy, and anyone who is crazy likes spinach. Therefore anyone who likes asparagus also likes spinach.

Anyone who likes spinach is crazy, and anyone who is crazy likes asparagus. Therefore anyone who likes asparagus also likes spinach.

4. Decide, for each of the following arguments, whether it is quantificationally valid. If the argument is quantificationally valid, explain why. If the argument is not quantificationally valid, construct an interpretation that shows this.

a. $(\forall x)((Lx \ \& \ Dx) \supset Fx)$
$(\exists x)(Dx \ \& \sim Fx)$

$\sim(\exists x)Lx$

*b. $(\forall x)(Sx \supset (Gx \vee Bx))$
$(\exists x)(Sx \ \& \sim Bx)$

$(\exists x)Gx$

c. $(\exists x)(Hx \equiv (Rx \vee Sx))$

$(\exists x)((Hx \ \& \ Rx) \vee (Hx \ \& \ Sx))$

*d. $(\exists x)(\exists y)((Rx \ \& \ Sy) \ \& \ Pxy)$
$(\forall x)(Rx \supset Tx)$

$(\exists x)(\exists y)(Tx \ \& \ Pxy)$

8.5 TRUTH-FUNCTIONAL EXPANSIONS

In the preceding sections we constructed interpretations for sentences of *PL* to establish various semantic results: A sentence is not quantificationally true, a set of sentences is quantificationally consistent, and so on. When we give an interpretation for a sentence or a set of sentences of *PL*, the UD we select may be very large or even infinite. However, when we ask whether certain sentences have various semantic properties, we can often find the answer by considering only interpretations with relatively small UDs. **Truth-functional expansions** enable us to reason about the truth-values of sentences on interpretations with small UDs.

The principles behind truth-functional expansions are simple. A universally quantified sentence says something about each member of the UD. If we have a finite UD and a set of constants such that each member of the UD is designated by at least one of these constants, then we can reexpress a universally quantified sentence as an iterated conjunction of its substitution instances formed from the constants (substitution instances were defined in Section 7.2 of Chapter 7). As long as every member of the UD is designated by at least one of the constants, the conjunction ends up saying the same thing as the universally quantified sentence—that every member of the UD satisfies some condition. An existentially quantified sentence says that there is at least one member of the UD of which such-and-such is true and can be reexpressed as an iterated disjunction of its substitution instances: The sentence says that such-and-such is true of *this* object or of *that* object or . . . As long as every member of the UD is designated by at least one of the constants, the disjunction of substitution instances makes the same claim about the UD as did the existentially quantified sentence.

To construct a truth-functional expansion, we first choose a set of individual constants. If the sentence contains any constants, they must be included in this set. To expand a universally quantified sentence $(\forall \mathbf{x})\mathbf{P}$, we remove the initial quantifier from the sentence and replace the resulting open sentence with the iterated conjunction

$$(\ldots (\mathbf{P}(\mathbf{a}_1/\mathbf{x}) \ \& \ \mathbf{P}(\mathbf{a}_2/\mathbf{x})) \ \& \ \ldots \ \& \ \mathbf{P}(\mathbf{a}_\mathbf{n}/\mathbf{x}))$$

where $\mathbf{a}_1, \ldots, \mathbf{a}_\mathbf{n}$ are the chosen constants and each $\mathbf{P}(\mathbf{a}_i/\mathbf{x})$ is a substitution instance of $(\forall \mathbf{x})\mathbf{P}$.

We shall expand the sentence '$(\forall x)Nx$' for the set of constants {'a', 'b'}. Removing the quantifier gives us the open sentence 'Nx', and we replace 'Nx' with the conjunction 'Na & Nb'. We can expand '$(\forall y)(My \supset Jyy)$' for the same set of constants by first dropping the quantifier and then replacing 'My ⊃ Jyy' with a conjunction in which 'a' replaces the free variable 'y' in the first conjunct, and 'b' replaces that variable in the second conjunct. The truth-functional expansion is

$$(Ma \supset Jaa) \ \& \ (Mb \supset Jbb),$$

and it has the same truth-value as the unexpanded sentence on every interpretation in which each member of the UD is named by at least one of the two constants. If we have an interpretation with a two-member UD, for example, in which 'a' designates one member and 'b' designates the other, then '(Ma ⊃ Jaa) & (Mb ⊃ Jbb)' makes the same claim about the UD as '(∀y)(My ⊃ Jyy)'—namely, that each of the two members is such that if it is M then it stands in the relation J to itself.

We have claimed that a truth-functional expansion has the same truth-value as the unexpanded sentence on any interpretation on which each member of the UD is named *by at least one* of the constants used in the expansion. We note two points about this claim using the previous example to illustrate. The first is that the interpretations in question may assign the same object to several of the constants as long as each object in the UD is named by at least one of them. So, if we have an interpretation with a one-member UD, both 'a' and 'b' must refer to that one member. In this case every object in the UD is named by at least one of the two constants. Our expanded sentence says twice that the one member of the UD is such that if it is M then it stands in the relation J to itself, and this is equivalent to the universal claim that every member of the UD satisfies that condition.

The second point is that, if a UD for an interpretation has even one member that is not designated by one of the two constants used in the truth-functional expansion, then that expansion and the sentence being expanded may not have the same truth-value. This is because in this case the truth-functional expansion does not talk about every member of the UD, whereas a universally quantified sentence always does so.

Now we shall expand the sentence

$$(\forall x)(Gac \vee Fx)$$

We have stipulated that the set of constants we use for an expansion must include all the individual constants that occur in the sentence being expanded. So any set of constants for which we expand the sentence must include 'a' and 'c'. We can expand the sentence for the set containing just those constants, in which case removing the initial quantifier and replacing 'x' with each constant in turn results in the expansion

$$(Gac \vee Fa) \ \& \ (Gac \vee Fc)$$

If we expand the sentence for the larger set {'a', 'c', 'e'} we obtain

$$((Gac \vee Fa) \ \& \ (Gac \vee Fc)) \ \& \ (Gac \vee Fe)$$

If the sentence we want to expand contains more than one universal quantifier, we start with the leftmost one and remove each in turn. To expand

$$(\forall y)(Ly \ \& \ (\forall z)Bzy)$$

for the set of constants {'a', 'b'}, we first eliminate the quantifier '$(\forall y)$' and expand the resulting open sentence, 'Ly & $(\forall z)$Bzy', to obtain

$$[La \;\&\; (\forall z)Bza] \;\&\; [Lb \;\&\; (\forall z)Bzb]$$

We now expand each of the universally quantified sentences that are components of '[La & $(\forall z)$Bza] & [Lb & $(\forall z)$Bzb]' by eliminating each occurrence of '$(\forall z)$' and expanding the resulting open sentences, to obtain first

$$[La \;\&\; (Baa \;\&\; Bba)] \;\&\; [Lb \;\&\; (\forall z)Bzb]$$

and then

$$[La \;\&\; (Baa \;\&\; Bba)] \;\&\; [Lb \;\&\; (Bab \;\&\; Bbb)]$$

Here we replaced '$(\forall z)$Bza' with '(Baa & Bba)' and '$(\forall z)$Bzb' with '(Bab & Bbb)'. Note that when we expand a quantified sentence that is a component of another sentence—as with '$(\forall z)$Bza' and '$(\forall z)$Bzb'—we replace that component exactly where it occurred in the sentence being expanded.

We may expand existentially quantified sentences in a similar way, except in this case we construct iterated *disjunctions* rather than iterated conjunctions. A sentence of the form $(\exists \mathbf{x})\mathbf{P}$ expands to the disjunction

$$(\ldots (\mathbf{P}(\mathbf{a}_1/\mathbf{x}) \vee \mathbf{P}(\mathbf{a}_2/\mathbf{x})) \vee \ldots \vee \mathbf{P}(\mathbf{a}_n/\mathbf{x}))$$

where $\mathbf{a}_1, \ldots, \mathbf{a}_n$ are the constants in the chosen set and each $\mathbf{P}(\mathbf{a}_i/\mathbf{x})$ is a substitution instance of $(\exists \mathbf{x})\mathbf{P}$. We construct an iterated disjunction because an existential quantification indicates that at least one member of the UD satisfies the specified condition: *This* member satisfies the condition, or *that* member satisfies the condition, and so on.

We can expand the sentence

$$(\exists x)(Fx \supset Gx)$$

for the set of constants {'a', 'b', 'c'} as

$$[(Fa \supset Ga) \vee (Fb \supset Gb)] \vee (Fc \supset Gc)$$

On any interpretation on which each of the members of the UD is named by at least one of 'a', 'b', and 'c', the expanded sentence will have the same truth-value as the existentially quantified sentence.

We expand the sentence

$$(\exists x)(\exists w)Zwx$$

for the set of constants {'a', 'b'} as follows: First, we eliminate '$(\exists x)$' and replace '$(\exists w)Zwx$' with an iterated disjunction:

$$(\exists w)Zwa \vee (\exists w)Zwb$$

Then we eliminate '$(\exists w)$' in each of its occurrences, first to obtain

$$(Zaa \vee Zba) \vee (\exists w)Zwb$$

and then to obtain

$$(Zaa \vee Zba) \vee (Zab \vee Zbb)$$

To expand the sentence

$$(\exists w)[Gw \supset \sim (Fw \vee (\exists z)Bz)]$$

for the set of constants {'a', 'b'}, we first eliminate '$(\exists w)$' to obtain

$$[Ga \supset \sim (Fa \vee (\exists z)Bz)] \vee [Gb \supset \sim (Fb \vee (\exists z)Bz)]$$

and then eliminate both occurrences of '$(\exists z)$' to obtain

$$[Ga \supset \sim (Fa \vee (Ba \vee Bb))] \vee [Gb \supset \sim (Fb \vee (Ba \vee Bb))]$$

The sentence

$$(\forall x)(Fx \vee (\exists z)[Fz \,\&\, \sim Izx])$$

can also be expanded by systematic elimination of its quantifiers. We shall expand it for the set of constants {'b', 'f'}. First, the universal quantifier is eliminated to obtain the conjunction

$$[Fb \vee (\exists z)[Fz \,\&\, \sim Izb]] \,\&\, (Ff \vee (\exists z)[Fz \,\&\, \sim Izf])$$

Next we eliminate the first occurrence of '$(\exists z)$' to obtain

$$(Fb \vee [(Fb \,\&\, \sim Ibb) \vee (Ff \,\&\, \sim Ifb)] \,\&\, (Ff \vee (\exists z)[Fz \,\&\, \sim Izf])$$

and then we eliminate the second occurrence of '$(\exists z)$' to obtain

$$(Fb \vee [(Fb \,\&\, \sim Ibb) \vee (Ff \,\&\, \sim Ifb)]) \,\&\,$$
$$(Ff \vee [(Fb \,\&\, \sim Ibf) \vee (Ff \,\&\, \sim Iff)])$$

When we expand a sentence, we may choose a set containing only one constant for the expansion. In this case we simply remove all quantifiers and replace the free variables in the resulting open sentence with that constant. '$(\forall x)Fx$' is expanded for the set of constants {'a'} as 'Fa', and '$(\exists x)Fx$' is also

expanded as ‘Fa’. With the same set of constants, ‘(∃y)Gyy’ is expanded as ‘Gaa’ and ‘(∀x)(Fx ∨ (∃y)Gyy)’ is expanded as ‘(Fa ∨ Gaa)’.

As a final example we expand the sentence

Dg ∨ (∀y)(∃x)Cyx

for the set of constants {‘a’, ‘g’} (we must include ‘g’ in the set because it occurs in the sentence to be expanded). We first replace ‘(∀y)(∃x)Cyx’ with its expansion to obtain

Dg ∨ [(∃x)Cax & (∃x)Cgx]

Then we replace ‘(∃x)Cax’ with its expansion to obtain

Dg ∨ [(Caa ∨ Cag) & (∃x)Cgx]

Finally we replace ‘(∃x)Cgx’ with its expansion to obtain

Dg ∨ [(Caa ∨ Cag) & (Cga ∨ Cgg)]

When we have expanded a sentence of *PL* to eliminate every quantifier, the truth-functional expansion that results is always an atomic sentence or a truth-functional compound of atomic sentences of *PL*. Because of this, we can construct truth-tables for truth-functional expansions, and these truth-tables will tell us something about the truth-conditions of the sentences that have been expanded. For example, the truth-functional expansion of the sentence ‘(∀x)(Fx & ~ Bx)’ for the set of constants {‘a’, ‘b’} is ‘(Fa & ~ Ba) & (Fb & ~ Bb)’. Here is a truth-table for the expansion:

								↓			
Ba	Bb	Fa	Fb	(Fa	&	~ Ba)		&	(Fb	&	~ Bb)
T	**T**	**T**	**T**	**T**	**F**	**F**	**T**	**F**	**T**	**F**	**F** **T**
T	**T**	**T**	**F**	**T**	**F**	**F**	**T**	**F**	**F**	**F**	**F** **T**
T	**T**	**F**	**T**	**F**	**F**	**F**	**T**	**F**	**T**	**F**	**F** **T**
T	**T**	**F**	**F**	**F**	**F**	**F**	**T**	**F**	**F**	**F**	**F** **T**
T	**F**	**T**	**T**	**T**	**F**	**F**	**T**	**F**	**T**	**T**	**T** **F**
T	**F**	**T**	**F**	**T**	**F**	**F**	**T**	**F**	**F**	**F**	**T** **F**
T	**F**	**F**	**T**	**F**	**F**	**F**	**T**	**F**	**T**	**T**	**T** **F**
T	**F**	**F**	**F**	**F**	**F**	**F**	**T**	**F**	**F**	**F**	**T** **F**
F	**T**	**T**	**T**	**T**	**T**	**T**	**F**	**F**	**T**	**F**	**F** **T**
F	**T**	**T**	**F**	**T**	**T**	**T**	**F**	**F**	**F**	**F**	**F** **T**
F	**T**	**F**	**T**	**F**	**F**	**T**	**F**	**F**	**T**	**F**	**F** **T**
F	**T**	**F**	**F**	**F**	**F**	**T**	**F**	**F**	**F**	**F**	**F** **T**
F	**F**	**T**	**T**	**T**	**T**	**T**	**F**	**T**	**T**	**T**	**T** **F**
F	**F**	**T**	**F**	**T**	**T**	**T**	**F**	**F**	**F**	**F**	**T** **F**
F	**F**	**F**	**T**	**F**	**F**	**T**	**F**	**F**	**T**	**T**	**T** **F**
F	**F**	**F**	**F**	**F**	**F**	**T**	**F**	**F**	**F**	**F**	**T** **F**

This truth-table tells us that the quantified sentence is true on some interpretations with one- or two-member UDs and false on some interpretations with one- or two-member UDs. We shall now explain why.

If each object in a UD is designated by either 'a' or 'b', then each of the combinations of truth-values to the left of the vertical line represents an interpretation of 'B' and 'F'. (This assumption means that we are restricting our attention to UDs with at most two members because the number of constants is not enough for naming more than two members.) For example, the first row represents interpretations with one- or two-member UDs in which all objects are in the extension of 'B' and also are in the extension of 'F'. If all objects in the UD are designated by either 'a' or 'b', then the assignment of **T** to both 'Ba' and 'Bb' means that all objects are in the extension of 'B', and the assignment of **T** to both 'Fa' and 'Fb' means that all objects are in the extension of 'F'. The second row represents interpretations with two-member UDs in which both objects are in the extension of 'B' (because both 'Ba' and 'Bb' are true), one object is in the extension of 'F' (because 'Fa' is true), and one object is not in the extension of 'F' (because 'Fb' is false). This row corresponds only to two-member UDs, because 'Fa' and 'Fb' do not have the same truth-value; we need at least two objects to make one of these true and the other false.

In fact, the sixteen rows between them represent all the combinations of extensions that the two predicates may have for interpretations with one- or two-member UDs. For example, we have the following possibilities for an interpretation with a one-member UD {**u**}: **u** is in the extension of both 'B' and 'F' (row 1), **u** is in the extension of 'B' but not of 'F' (row 4), **u** is in the extension of 'F' but not of 'B' (row 13), or **u** is not in the extension of either predicate (row 16). For a two-member UD {$\mathbf{u}_1$, $\mathbf{u}_2$} we have the following possibilities: Both $\mathbf{u}_1$ and $\mathbf{u}_2$ are in the extensions of both 'B' and 'F' (row 1), both $\mathbf{u}_1$ and $\mathbf{u}_2$ are in the extension of 'B' but only one of $\mathbf{u}_1$ and $\mathbf{u}_2$ is in the extension of 'F' (rows 2 and 3), both $\mathbf{u}_1$ and $\mathbf{u}_2$ are in the extension of 'B' but neither is in the extension of 'F' (row 4), and so on.

The truth-value assigned to the truth-functional expansion in each row is the truth-value that '$(\forall x)(Fx \,\&\, \sim Bx)$' receives for the interpretations of 'B' and 'F' represented by that row. The expansion has the truth-value **F** in the first row, from which we may conclude that, on every interpretation with a one- or two-member UD in which every member **u** is such that **u** is in the extension of 'B' and also in the extension of 'F', '$(\forall x)(Fx \,\&\, \sim Bx)$' is false. The expansion also has the truth-value **F** in the second row, from which we may conclude that, on every interpretation with a two-member UD {$\mathbf{u}_1$, $\mathbf{u}_2$} (recall that this row cannot represent interpretations with one-member UDs) such that both $\mathbf{u}_1$ and $\mathbf{u}_2$ are in the extension of 'B' but only one of these is in the extension of 'F', the sentence '$(\forall x)(Fx \,\&\, \sim Bx)$' is false.

The thirteenth row is the only one in which the expansion is true. From this we may conclude that '$(\forall x)(Fx \,\&\, \sim Bx)$' is true on every interpretation with a one- or two-member UD in which the extension of 'B' is empty and every member is in the extension of 'F' and that it is false on every other interpretation with a one- or two-member UD.

We can use the information in the thirteenth row to construct an interpretation on which the unexpanded quantified sentence is true. Because neither 'a' nor 'b' appears in the quantified sentence, we need only specify a UD (we will choose one with two members rather than one), an empty extension for 'B' (because 'Ba' and 'Bb' are both false), and an extension for 'F' that includes both members of the UD (because 'Fa' and 'Fb' are both true). We will specify this interpretation purely extensionally:

15. UD: The set {1, 2}
B: ∅
F: {<1>, <2>}.

Of course, it is also possible to specify an interpretation in the style of symbolization keys, such as

UD: The set {1, 2}
Bx: x is a negative integer
Fx: x is a positive integer

We specify interpretations purely extensionally in this section because it is straightforward to do so, given the truth-tables for truth-functional expansions. In fact, this can be made into a mechanical process if we agree that 'a' will always designate 1, 'b' will always designate 2, and so on.

We can use the information in the first row to construct an interpretation on which the sentence is false. We'll use the same UD and set the extensions of both predicates to include <1> and <2>—because the four atomic sentences in the first row are all true:

16. UD: The set {1, 2}
B: {<1>, <2>}
F: {<1>, <2>}

Any row in the truth-table in which 'Ba' has the same truth-value as 'Bb' and 'Fa' has the same truth-value as 'Fb' can be used to construct an interpretation with a one-member UD. For example, using the first row, we can construct an interpretation on which our quantified sentence is false by making sure that the 1-tuple of the UD's single object is in the extension of both 'B' and 'F':

17. UD: The set {1}
Bx: {<1>}
Fx: {<1>}

A truth-functional expansion of the sentence '$(\exists x)(\forall y)Nyx$' for the set of constants {'a', 'b'} is '$(Naa \ \& \ Nba) \vee (Nab \ \& \ Nbb)$'. We may show that the

sentence '(∃x)(∀y)Nyx' is true on at least one interpretation with a two-member UD by producing a shortened truth-table in which the expansion is true:

								↓					
Naa	Nab	Nba	Nbb	(Naa	&	Nba)	∨	(Nab	&	Nbb)			
T	**T**	**F**	**T**	**T**	**F**	**F**	**T**	**T**	**T**	**T**			

(The table in this case gives us information only about two-member UDs, because if there were only one member in the UD then it would be named by both 'a' and 'b', and hence the four atomic sentences would have to have the same truth-value.) We do not have to give an actual interpretation on which the sentence is true; the shortened truth-table suffices to show that *there is* such an interpretation. It shows that the quantified sentence is true on any interpretation with a two-member UD in which both members stand in the relation N to themselves and one stands in the relation N to the other, but not vice versa. And the following shortened truth-table shows that the quantified sentence is false on at least one interpretation with a one- or two-member UD:

								↓					
Naa	Nab	Nba	Nbb	(Naa	&	Nba)	∨	(Nab	&	Nbb)			
F	**F**	**F**	**F**	**F**	**F**	**F**	**F**	**F**	**F**	**F**			

(Because the four atomic sentences have the same truth-value in this table, the row of assignments may represent an interpretation with a one-member UD.) From these two shortened truth-tables we may conclude that '(∃x)(∀y)Nyx' is quantificationally indeterminate. The tables show that the sentence is true on at least one interpretation and false on at least one interpretation.

We may use truth-functional expansions and truth-tables to demonstrate that sentences and sets of sentences of *PL* have, or fail to have, some other semantic properties as well. For example, to show that a sentence is not quantificationally true, we must show that the sentence is false on at least one interpretation. And we can show this by producing a shortened truth-table in which a truth-functional expansion of the sentence is false. We will have to choose a set of constants first—ideally a small set, to save us work. An expansion of the sentence '(Ga & (∃z)Bz) ⊃ (∀x)Bx' for the set of constants {'a', 'b'} is '(Ga & (Ba ∨ Bb) ⊃ (Ba & Bb)', and the expansion is false in the following shortened truth-table:

										↓					
Ba	Bb	Ga	(Ga	&	(Ba	∨	Bb))	⊃	(Ba	&	Bb)				
T	**F**	**T**	**T**	**T**	**T**	**T**	**F**	**F**	**T**	**F**	**F**				

The table shows that there is at least one interpretation on which the sentence '(Ga & (∃z)Bz) ⊃ (∀x)Bx' is false. This sentence is therefore not quantificationally true.

Note that we cannot in general use truth-functional expansions to show that a sentence *is* quantificationally true. Even if we construct a full truth-table for a truth-functional expansion and find that the expansion is true in every row of the truth-table, all that we may generally conclude is that the sentence is true on every interpretation with a UD that is the same size as or smaller than the number of constants in the set that was used for the expansion. (An exception will be noted at the end of this section.)

The sentence '~ (~ Ga & (∃y)Gy)' is not quantificationally false. The truth-functional expansion of this sentence for the set of constants {'a'}('a' must be in this set because it occurs in the sentence) is '~ (~ Ga & Ga)', and this expansion is true in the following shortened truth-table:

		↓				
Ga	∣	~	(~ Ga	&	Ga)	
T	∣	**T**	**F** **T**	**F**	**T**	

This shows that the sentence '~ (~ Ga & (∃y)Gy)' is true on at least one interpretation and hence that the sentence is not quantificationally false. As with quantificational truth we cannot in general use truth-functional expansions to show that a sentence is quantificationally false, for that would involve showing that the sentence is false on every interpretation, not just those with a particular size UD.

The sentences

(∀x)(Fx ⊃ Ga)

and

(∀x)Fx ⊃ Ga

are not quantificationally equivalent. To show this, we can expand both sentences for the same set of constants (which must include 'a') and produce a shortened truth-table in which the expansions have different truth-values:

						↓								↓		
Fa	Fb	Ga	∣	(Fa	⊃	Ga)	&	(Fb	⊃	Ga)		(Fa	&	Fb)	⊃	Ga
T	**F**	**F**	∣	**T**	**F**	**F**	**F**	**F**	**T**	**F**		**T**	**F**	**F**	**T**	**F**

This shows that there is at least one interpretation on which '(∀x)(Fx ⊃ Ga)' is false and '(∀x)Fx ⊃ Ga' is true.

The set of sentences

{(∀x)Gax, ~ Gba ∨ (∃x) ~ Gax}

is quantificationally consistent. The truth-functional expansions of these sentences for the set {'a', 'b'} are both true in the following shortened truth-table,

and so we may conclude that there is at least one interpretation on which both members of the set are true:

					↓				↓				
Gaa	Gab	Gba	Gaa	&	Gab	~ Gba	∨	(~ Gaa	∨	~ Gab)			
T	**T**	**F**	**T**	**T**	**T**	**TF**	**T**	**FT**	**F**	**FT**			

The set of sentences

$\{\sim (\forall x)(Ga \equiv Fx), \sim Fb\}$

does not quantificationally entail the sentence

$(\forall x) \sim Gx$

The sentences have been expanded for the set of constants {'a', 'b'} in the following shortened truth-table:

					↓								↓			↓	
Fa	Fb	Ga	Gb	~	[(Ga	≡	Fa)	&	(Gb	≡	Fb)]	~ Fb	~ Ga	&	~ Gb		
F	**F**	**F**	**T**	**T**	**F**	**T**	**F**	**F**	**T**	**F**	**F**	**TF**	**TF**	**F**	**FT**		

The displayed row shows that there is at least one interpretation on which the set members are both true and '$(\forall x) \sim Gx$' is false, so '$(\forall x) \sim Gx$' is not quantificationally entailed by the set.

Finally we may use truth-functional expansions to show that some arguments are not quantificationally valid. The premises and conclusion of the argument

$(\exists x)[(\exists y)Fy \supset Fx]$
$(\exists y) \sim Fy$
$\sim (\exists x)Fx$

have been expanded for the set of constants {'a', 'b'} in the following shortened truth-table:

								↓								↓		↓					
Fa	Fb	[(Fa	∨	Fb)	⊃	Fa]	∨	[(Fa	∨	Fb)	⊃	Fb]	~ Fa	∨	~ Fb	~	(Fa	∨	Fb)				
T	**F**	**T**	**T**	**F**	**T**	**T**	**T**	**T**	**T**	**F**	**F**	**F**	**FT**	**T**	**TF**	**F**	**T**	**T**	**F**				

This shows that there is an interpretation on which the premises of the original argument are true and the conclusion is false.

Note once again that truth-functional expansions cannot generally be used to show that a set of sentences of *PL* is quantificationally inconsistent, that a set of sentences *does* quantificationally entail some sentence, or that an argument of *PL* is quantificationally valid. In each of these cases, we must prove

something about every interpretation, not just those represented in the truth-table for a particular set of expansions.

However, there is an exception to our claims about the limitations of using truth-functional expansions to test for semantic properties. We noted at the end of Section 8.2 that there is a decision procedure (based on a result by Bernays and Schönfinkel) for determining the quantificational status of sentences of *PL* that contain no many-place predicates, that is, in which the predicates are all one-place predicates. A decision procedure allows us to answer correctly in a finite number of mechanical steps the question 'Is this sentence quantificationally true?' and hence also questions like 'Is this sentence quantificationally false?' (it is if its negation is quantificationally true) and 'Is this finite set of sentences quantificationally consistent?' (it is if the conjunction of the sentences in the set is not quantificationally false). It allows us to answer these questions correctly for sentences that do not contain many-place predicates.

Bernays and Schönfinkel's result is that a sentence that contains no many-place predicates and that contains **k** distinct one-place predicates is quantificationally true if and only if the sentence is true on every interpretation with a UD containing exactly 2^k members. This being the case, we can truth-functionally expand the sentence for a set of at least 2^k constants, produce a truth-table for the expanded sentence, and determine whether it is quantificationally true by examining the truth-table. If the expanded sentence is true in every row of the truth-table, we may conclude that the sentence is true on every interpretation with a UD that contains exactly 2^k members (as well as all interpretations with smaller UDs). Given Bernays and Schönfinkel's result, we may then conclude that the sentence is quantificationally true.

8.5**E** EXERCISES

1. Produce a truth-functional expansion of each of the sentences in Exercise 7 in Section 8.1E for a set containing one constant.

2. Produce a truth-functional expansion of each of the sentences in Exercise 8 in Section 8.1E for a set containing two constants.

3. Produce a truth-functional expansion of each of the following sentences for the set {'a', 'b', 'c'}.

a. $(\forall w)(Gw \supset Nww)$
*b. $(Na \vee (\exists z)Bz)$
c. $(\exists z)(Na \equiv Bz)$
*d. $(\forall w)Bw \vee \sim (\exists w)Bw$

4. Construct truth-functional expansions of the sentence

'$((\exists x)Fx \ \& \ (\exists y) \sim Fy) \supset (\forall x) \sim Fx$'

for the sets {'a'} and {'a', 'b'}. Construct a truth-table for each expansion. What information does the first truth-table give you about this sentence? What information does the second truth-table give you?

5. For each of the following sentences, construct a truth-functional expansion for the set of constants {'a', 'n'}. Show that the expansion is true on at least one truth-value assignment. Then use the information in the truth-table to construct an interpretation on which the original sentence is true. You may specify the interpretation either purely extensionally or in the style of symbolization keys.

a. $(\forall x)(Nxx \vee (\exists y)Nxy)$

*b. $(\exists x)Fx \equiv (\forall x)Fx$

c. $(\forall y)Syyn$

6. Show that each of the sentences in Exercise 1 in Section 8.2E is not quantificationally true by producing a shortened truth-table in which a truth-functional expansion of the sentence is false.

7. Show that each of the sentences in Exercise 2 in Section 8.2E is not quantificationally false by producing a shortened truth-table in which a truth-functional expansion of the sentence is true.

8. Show that each of the sentences in Exercise 3 in Section 8.2E is quantificationally indeterminate by producing a shortened truth-table in which a truth-functional expansion of the sentence is true and a shortened truth-table in which a truth-functional expansion of the sentence is false.

***9.** In this section it was claimed that in general a sentence of *PL* that contains quantifiers cannot be shown to be quantificationally true by producing truth-tables for truth-functional expansions. Does the claim hold for sentences of *PL* that do not contain quantifiers, such as 'Fa $\supset$ (Gb $\supset$ Fa)'? Explain.

10. The truth-functional expansion of the sentence '$(\exists y)Gy$ & $(\exists y) \sim Gy$' for the set {'a'} is 'Ga & ~ Ga'. The expanded sentence is quantificationally false. Explain this and then explain why this does *not* show that the original sentence '$(\exists y)Gy$ & $(\exists y) \sim Gy$' is quantificationally false.

11. Show that the sentences in each pair in Exercise 1 in Section 8.3E are not quantificationally equivalent by producing a shortened truth-table in which a truth-functional expansion of one sentence of the pair is true and a truth-functional expansion of the other sentence (for the same set of constants) is false.

12. Show that each set of sentences in Exercise 4 in Section 8.3E is quantificationally consistent by producing a shortened truth-table in which a truth-functional expansion of each sentence in the set (for the same set of constants) is true.

***13.** a. Is the set {Ba, Bb, Bc, Bd, Be, Bf, Bg, ~ $(\forall x)Bx$} quantificationally consistent? Explain.

b. For the set in Exercise 13.a, what is the minimum size set of constants for which the sentences in the set must be expanded in order to show that the set is quantificationally consistent? Explain.

c. Can all the sentences in the set in Exercise 13.a be true on an interpretation with a UD smaller than the set of constants indicated in the answer to Exercise 13.b? Explain.

14. Show that each argument in Exercise 2 in Section 8.4E is quantificationally invalid by producing a shortened truth-table in which truth-functional expansions of the premises are true and a truth-functional expansion of the conclusion for the same set of constants is false.

In *PLE*, interpretations for sentences containing the identity predicate but no functors are the same as interpretations for *PL*, because the *identity* predicate, '=', is *not* explicitly given an interpretation. This is because we always want its extension to be the set of 2-tuples of members of the UD in which the first member is identical to the second, no matter what the UD is. Rather, we add the following clause to our definition of satisfaction:

> 10. If **P** is an atomic formula of the form $\mathbf{t}_1 = \mathbf{t}_2$, then $\mathbf{d_I}$ satisfies **P** on interpretation **I** if and only if $\text{den}_{\mathbf{I},\mathbf{d_I}}(\mathbf{t}_1) = \text{den}_{\mathbf{I},\mathbf{d_I}}(\mathbf{t}_2)$.

This clause implicitly defines an extension for the identity predicate on each interpretation: The extension includes <**u**, **u**> for each member **u** of the UD, and that is all it includes.

Obviously, every atomic sentence of the form **a** = **a**, where **a** is an arbitrary individual constant, is true on every interpretation. More generally, every sentence of the form $(\forall\mathbf{x})\mathbf{x} = \mathbf{x}$ is true on every interpretation. In both cases the reason is that the same constant/variable appears on both sides of the identify predicate, so that the same member of the UD is denoted, as required by the satisfaction clause for identity formulas. On the other hand, the truth-value of an atomic sentence of the form **a** = **b**, where **a** and **b** are different individual constants, depends on the interpretations of **a** and **b**. Interpretation 18 makes the sentence 'g = k' true, while interpretation 19 makes the sentence false:

> 18. UD: The set of positive integers
> g: 1
> k: 1
>
> 19. UD: The set of positive integers
> g: 1
> k: 2

The sentence '$(\forall x)(\forall y)(\sim x = y \supset Gxy)$' is true on interpretation 20 and false on interpretation 21:

> 20. UD: The set of positive integers
> Gxy: the sum of x and y is positive
>
> 21. UD: The set of positive integers
> Gxy: x is greater than y

On interpretation 20 the sentence may be read as 'The sum of any pair of nonidentical positive integers is a positive integer'—which is true. On interpretation 21 the sentence claims that for any pair of nonidentical positive integers

the first is greater than the second. This is false—1 and 2 are nonidentical positive integers, for example, but 1 is not greater than 2.

The sentence '$(\exists x)(Fx \ \& \ (\forall y)(Fy \supset y = x))$' is false on interpretation 22:

22. UD: Set of positive integers
F: $\{<\mathbf{u}>: \mathbf{u}$ is odd$\}$

On this interpretation, the sentence may be read as 'Exactly one positive integer is odd'. We'll show how the formal semantics works in this case. Let d_{22} be an arbitrary variable assignment for this interpretation:

- d_{22} satisfies '$(\exists x)(Fx \ \& \ (\forall y)(Fy \supset y = x))$' if and only if there is at least one member $\mathbf{u}_1$ of the UD such that the variant $d_{22}[\mathbf{u}_1/x]$ satisfies '$Fx \ \& \ (\forall y)(Fy \supset y = x)$', and this will be the case if $d_{22}[\mathbf{u}_1/x]$ satisfies both 'Fx' and '$(\forall y)(Fy \supset y = x)$'.
- If $d_{22}[\mathbf{u}_1/x]$ satisfies 'Fx', then $<\mathbf{u}_1> \in I_{22}(F)$.
- But if $<\mathbf{u}_1> \in I_{22}(F)$, $d_{22}[\mathbf{u}_1/x]$ does not satisfy '$(\forall y)(Fy \supset y = x)$' because for every odd integer $\mathbf{u}_2$ that is distinct from $\mathbf{u}_1$, $d_{22}[\mathbf{u}_1/x, \mathbf{u}_2/y]$ does not satisfy '$Fy \supset y = x$'. Although $d_{22}[\mathbf{u}_1/x, \mathbf{u}_2/y]$ satisfies the antecedent if $\mathbf{u}_2$ is odd, it does not satisfy the consequent if $\mathbf{u}_2$ is not identical to $\mathbf{u}_1$.

We can show that various sentences and sets of sentences that contain the identity predicate have, or fail to have, semantic properties much as we did for *PL*. We shall give a few examples for the semantic properties of quantificational truth and quantificational validity. We can show that the sentence

$$(\forall x)(\forall y)(\sim x = y \vee (Fx \supset Fy))$$

is quantificationally true by reasoning generally about interpretations. The sentence is universally quantified and is true on an interpretation if each pair x and y of members of the UD is such that either the pair satisfies '$\sim x = y$' or it satisfies '$Fx \supset Fy$'. So let us consider members x and y of an arbitrary UD. If x and y are not the same member, then the first disjunct '$\sim x = y$' is satisfied. If, however, x and y *are* the same member of the UD (and hence do not satisfy the first disjunct), they satisfy the second disjunct. If x is in the extension of 'F', then so is y—because y is identical to x, and so x and y satisfy the condition '$Fx \supset Fy$'. Therefore the sentence '$(\forall x)(\forall y)(\sim x = y \vee (Fx \supset Fy))$' must be true on any interpretation; it is quantificationally true.

On the other hand, the sentence

$$(\forall x)(\forall y)(x = y \vee (Fx \supset Fy))$$

is not quantificationally true. To show this, we construct an interpretation on which the sentence is false. The sentence claims that every pair of members

of the UD x and y satisfies '$x = y \vee (Fx \supset Fy)$'—that is, either x and y are the same member or if x is F then so is y. If we choose a two-member UD, then a pair consisting of the two distinct members will not satisfy the condition '$x = y$'. If the first member is F but the other is not, then this pair also will not satisfy '$Fx \supset Fy$'. Here is our interpretation:

23. UD: The set {1, 2}
F: x is odd

The pair consisting of the numbers 1 and 2 does not satisfy '$x = y \vee (Fx \supset Fy)$'. The two numbers are not identical, and it is not true that if 1 is odd (which it is) then 2 is odd (it is not).

The argument

$$\begin{array}{l} (\forall x)(Fx \equiv Gx) \\ (\forall x)(\forall y)x = y \\ Ga \\ \hline (\forall x)Fx \end{array}$$

is quantificationally valid. We can reason that any interpretation that makes the three premises true also makes '$(\forall x)Fx$' true. If '$(\forall x)(Fx \equiv Gx)$' is true, then every member of the UD that is F is also G, and every member of the UD that is G is also F. If '$(\forall x)(\forall y)x = y$' is also true, then there is exactly one object in the UD. The sentence says that, for any object x and any object y, x and y are identical—and this cannot be the case if there is more than one member of the UD. If 'Ga' is also true, then because there is exactly one object in the UD this object must be designated by 'a' and must therefore be in the extension of 'G'. It follows, from the truth of the first sentence, that this object is also in the extension of 'F'. Therefore it follows that '$(\forall x)Fx$' is true—every object in our single-member UD is F.

On the other hand, the argument

$$\begin{array}{l} (\forall x)(\exists y)x = y \\ \hline a = b \end{array}$$

is not quantificationally valid. The premise, it turns out, is quantificationally true—every member of any UD is identical to something (namely, itself). But the conclusion is false on any interpretation on which 'a' and 'b' designate different objects, such as the following:

24. UD: The set of positive integers
a: 6
b: 7

It is true that every positive integer is identical to some positive integer, but it is false that 6 is identical to 7.

Readers who have worked through the section on truth-functional expansions may wonder whether sentences containing the identity predicate may be expanded and truth-tables used to check for various semantic properties. The answer is yes, although we shall see that there is a complication. Sentences that contain the identity predicate are expanded in the same way as sentences without the identity predicate: Quantifiers are eliminated in favor of iterated conjunctions or disjunctions. The sentence '$(\forall x)(\exists y)x = y$' can be expanded for the set of constants {'a', 'b'} first to obtain

$$(\exists y)a = y \;\&\; (\exists y)b = y$$

and then to obtain

$$(a = a \vee a = b) \;\&\; (b = a \vee b = b)$$

But if we freely assign truth-values to the atomic components of this sentence, we end up with this truth-table:

							↓			
a = a	a = b	b = a	b = b	(a = a	∨	a = b)	&	(b = a	∨	b = b)
T	**T**	**T**	**T**	**T**	**T**	**T**	**T**	**T**	**T**	**T**
T	**T**	**T**	**F**	**T**	**T**	**T**	**T**	**T**	**T**	**F**
T	**T**	**F**	**T**	**T**	**T**	**T**	**T**	**F**	**T**	**T**
T	**T**	**F**	**F**	**T**	**T**	**T**	**F**	**F**	**F**	**F**
T	**F**	**T**	**T**	**T**	**T**	**F**	**T**	**T**	**T**	**T**
T	**F**	**T**	**F**	**T**	**T**	**F**	**T**	**T**	**T**	**F**
T	**F**	**F**	**T**	**T**	**T**	**F**	**T**	**F**	**T**	**T**
T	**F**	**F**	**F**	**T**	**T**	**F**	**F**	**F**	**F**	**F**
F	**T**	**T**	**T**	**F**	**T**	**T**	**T**	**T**	**T**	**T**
F	**T**	**T**	**F**	**F**	**T**	**T**	**T**	**T**	**T**	**F**
F	**T**	**F**	**T**	**F**	**T**	**T**	**T**	**F**	**T**	**T**
F	**T**	**F**	**F**	**F**	**T**	**T**	**F**	**F**	**F**	**F**
F	**F**	**T**	**T**	**F**	**F**	**F**	**F**	**T**	**T**	**T**
F	**F**	**T**	**F**	**F**	**F**	**F**	**F**	**T**	**T**	**F**
F	**F**	**F**	**T**	**F**	**F**	**F**	**F**	**F**	**T**	**T**
F	**F**	**F**	**F**	**F**	**F**	**F**	**F**	**F**	**F**	**F**

MISTAKE!

There is something wrong with this truth-table! The sentence '$(\forall x)(\exists y)x = y$' is quantificationally true, and yet we have assigned its expansion the truth-value **F** in seven rows. Let us look at the first row where this happened: row 4. In this row we have assigned **T** to 'a = b' and **F** to 'b = a', and that is a problem (There is another problem with this row, which we will get to below.) If an interpretation makes 'a = b' true then it must make 'b = a' true as well;

a and b are the same object. So row 4 does not correspond to any interpretation at all. By the same reasoning we find that none of rows 3–6 or 11–14 correspond to interpretations, for each of these rows assigns different truth-values to 'a = b' and 'b = a'. However, this still leaves us with problematic rows 8, 15, and 16—all of which make the expanded sentence false. The problem with each of these rows is that the truth-value **F** has been assigned to one or both of 'a = a' and 'b = b'—thus claiming that some object is not the same as itself (This is also a problem in the row we first examined, row 4.). Because every interpretation makes every sentence of the form **a** = **a** true, rows 8, 15, and 16, as well as all other rows that make one or both of 'a = a' and 'b = b' false, do not correspond to interpretations. In fact, we have just ruled out all rows in the truth-table except rows 1 and 7. These are the only rows in which 'a = a' and 'b = b' are both true and in which 'a = b' and 'b = a' have the same truth-value—and, as we should have expected for a quantificationally true sentence, the expanded sentence is true in both rows.

The rows of a truth-table that do not correspond to any interpretation cannot be used to establish semantic properties of the sentence that has been expanded. We therefore require that each row in the truth-table we construct for an expansion of a sentence containing the identity predicate must meet two conditions:

1. Every sentence of the form **a** = **a** has the truth-value **T**.
2. If a sentence of the form **a** = **b** has the truth-value **T**, then for each atomic sentence **P** that contains **a**, every atomic sentence **P**(**b**//**a**) that results from replacing one or more occurrences of **a** in **P** with **b** must have the same truth-value as **P**.

If conditions 1 and 2 are met, then if a sentence containing **a** = **b** has the truth-value **T** in a row, **b** = **a** will also have the truth-value **T**. Condition 1 requires that **a** = **a** have the truth-value **T**, and because **b** = **a** can be obtained from **a** = **a** by replacing the first occurrence of **a** with **b**, condition 2 requires that **b** = **a** is true since **a** = **b** and **a** = **a** are. Condition 2 also rules out rows like the one in the following shortened truth-table for the expansion

$$[(a = a \supset (Fa \supset Fa)) \;\&\; (a = b \supset (Fa \supset Fb))] \;\&\; [(b = a \supset (Fb \supset Fa)) \;\&\; (b = b \supset (Fb \supset Fb))]$$

of the sentence '$(\forall x)(\forall y)(x = y \supset (Fx \supset Fy))$' for the set of constants {'a', 'b'}:

a = a	a = b	b = a	b = b	Fa	Fb	[(a = a ⊃ (Fa ⊃ Fa)) & (a = b ⊃ (Fa ⊃ Fb))]
T	**T**	**T**	**T**	**T**	**F**	**T T T T T F T F T F F**

↓

& [(b = a ⊃ (Fb ⊃ Fa)) & (b = b ⊃ (Fb ⊃ Fb))]

F T T F T T T T T F T F

MISTAKE!

Once again we have expanded a quantificationally true sentence and produced a row of a truth-table in which the truth-functional expansion is false. We have ensured that both sentences 'a = a' and 'b = b' are true and that 'a = b' and 'b = a' have the same truth-value. The problem is that we have assigned 'Fa' and 'Fb' different truth-values, although 'a = b' is true. Condition 2 rules out this combination: 'Fb' results from replacing 'a' in 'Fa' with 'b,' and so, because 'a = b' is true, 'Fb' must have the same truth-value as 'Fa'. Our second condition reflects the fact that when the identity predicate occurs in a truth-functional expansion the atomic sentences that are components of the expansion may not be truth-functionally independent. Once a sentence of the form **a** = **b**, where **a** and **b** are different constants, has been made true, certain other atomic sentences must agree in truth-value.

The following truth-table shows the only combinations of truth-values for the atomic components of our sentence that correspond to interpretations with one- or two-member UDs:

a = a	a = b	b = a	b = b	Fa	Fb	[(a = a ⊃ (Fa ⊃ Fa)) & (a = b ⊃ (Fa ⊃ Fb))]
T	**T**	**T**	**T**	**T**	**T**	**T T T T T T T T T T**
T	**T**	**T**	**T**	**F**	**F**	**T T F T F T T T F T F**
T	**F**	**F**	**T**	**T**	**T**	**T T T T T T F T T T T**
T	**F**	**F**	**T**	**T**	**F**	**T F T T T T F T T F F**
T	**F**	**F**	**T**	**F**	**T**	**T T F T F T F T F T T**
T	**F**	**F**	**T**	**F**	**F**	**T T F T F T F T F T F**

↓

& [(b = a ⊃ (Fb ⊃ Fa)) & (b = b ⊃ (Fb ⊃ Fb))]
T T T T T T T T T T T T
T T T F T F T T T F T F
T F T T T T T T T T T T
T F T F T T T T T F T F
T F T T F F T T T T T T
T F T F T F T T T F T F

All other rows are excluded by one or both of our conditions. And again we find that the expanded sentence is true in all six rows—we have shown that there are no interpretations with one- or two-member UDs on which the sentence is false.

Adhering to our two conditions, we now produce a shortened truth-table that shows that the sentence

$$(\forall z)[Fz \ \& \ (\exists y)z = y] \supset (\forall x)Fx)$$

is not quantificationally true. The sentence claims that, for each member of the UD, if it is F and is identical to something then everything is F. Certainly the sentence will be true if the UD contains exactly one object—but for larger UDs it will be true only if either no member is F or all members are. We shall

expand the sentence for the set of constants {'a', 'b'} and produce a shortened truth-table in which the expansion is false:

a = a	a = b	b = a	b = b	Fa	Fb	[(Fa	&	(a = a	∨	a = b))	⊃	(Fa	&	Fb)]
T	**F**	**F**	**T**	**T**	**F**	**T**	**T**	**T**	**T**	**F**	**F**	**T**	**F**	**F**

↓

&	[(Fb	&	(b = a	∨	b = b))	⊃	(Fa	&	Fb)]
F	**F**	**F**	**F**	**T**	**T**	**T**	**T**	**F**	**F**

Condition 1 has been met—both 'a = a' and 'b = b' are true. Condition 2 has also been met, trivially. The two identity statements that are true are 'a = a' and 'b = b', and the result of substituting 'a' for 'a' in any sentence is just that sentence itself and the same holds for 'b'. Here is an interpretation that has been constructed using the truth-values in the truth-table as a guide:

25. UD: The set {2, 3}
F: {<2>}

We have chosen a UD with two members because the identity statements 'a = b' and 'b = a' are false in the shortened table, and so 'a' and 'b' must designate different objects. We have interpreted 'F' so that it is true of one member of the UD, but not the other.

Finally we shall complete our semantics for sentences of *PLE* that contain functors by amending our definition of an interpretation and of the denotation of a term with respect to an interpretation **I** and variable assignment $\mathbf{d_I}$. We first define precisely the concept of an **n**-*place function on a UD:* An **n**-place function on a UD maps each **n**-tuple of members of the UD to a single member of the UD (not necessarily the same one in each case). So, if the UD consists of the integers 1 and 2, for example, there are four distinct 1-place functions: the function that maps each of 1 and 2 to itself, the function that maps both 1 and 2 to 1, the function that maps both 1 and 2 to 2, and the function that maps 1 to 2 and 2 to 1. We can represent each of these functions as a set of ordered pairs, where for each member **u** of the UD there is exactly one ordered pair with **u** as its first member, and the second member of that ordered pair is the value of the function for **u**. Here are the four sets of ordered pairs:

{<1, 1>, <2, 2>} (the function that maps each of 1 and 2 to itself)
{<1, 1>, <2, 1>} (the function that maps both 1 and 2 to 1)
{<1, 2>, <2, 2>} (the function that maps both 1 and 2 to 2)
{<1, 2>, <2, 1>} (the function that maps 1 to 2 and 2 to 1)

Similarly a 2-place function can be represented as a set of ordered triples, where for each pair of members $\mathbf{u}_1$ and $\mathbf{u}_2$ of the UD there is one ordered

triple with $\mathbf{u}_1$ and $\mathbf{u}_2$ as the first two members, and the third member of that ordered triple is the value of the function for $\mathbf{u}_1$ and $\mathbf{u}_2$. So the multiplication function for the set of integers {0, 1} can be represented as the set {<0,0,0>, <0,1,0>, <1,0,0>, <1,1,1>}. (Note that multiplication is not a function on the set {1, 2} because we require that the value of the function for each pair of members of the UD itself be a member of the UD, but 2×2 is not a member of {1, 2}. On the other hand, multiplication *is* a function on the set of positive integers.)

We may now amend our definition of an interpretation:

> An interpretation for *PLE* consists in the specification of a UD and the assignment of a truth-value to each sentence letter of *PLE*, a member of the UD to each individual constant of *PLE*, an **n**-place function on the UD to each **n**-place functor of *PLE*, and a set of **n**-tuples of members of the UD to each **n**-place predicate of *PLE*.

We extend the definition of the *denotation of a term with respect to an interpretation* $\boldsymbol{I}$ *and variable assignment* $\boldsymbol{d_I}$ as follows, to accommodate terms containing functors:

1. If $\mathbf{t}$ is a variable, then $\text{den}_{\mathbf{I},\mathbf{d_I}}(\mathbf{t}) = \mathbf{d}(\mathbf{t})$.
2. If $\mathbf{t}$ is an individual constant, then $\text{den}_{\mathbf{I},\mathbf{d_I}}(\mathbf{t}) = \mathbf{I}(\mathbf{t})$.
3. If $\mathbf{t}$ is a term $f(\mathbf{t}_1, \ldots, \mathbf{t}_n)$ where f is an **n**-place functor and $\mathbf{t}_1, \ldots, \mathbf{t}_n$ are terms, then if $<\text{den}_{\mathbf{I},\mathbf{d_I}}(\mathbf{t}_1), \ldots, \text{den}_{\mathbf{I},\mathbf{d_I}}(\mathbf{t}_n), \mathbf{u}>$ is a member of $\mathbf{I}(f)$, $\text{den}_{\mathbf{I},\mathbf{d_I}}(\mathbf{t}) = \mathbf{u}$.

Recall that for any members $\mathbf{u}_1, \ldots, \mathbf{u}_n$ of the UD, there will be a unique **n**-tuple $<\mathbf{u}_1, \ldots, \mathbf{u}_n, \mathbf{u}>$ that is a member of the function $\mathbf{I}(f)$, so clause 3 identifies exactly one member of the UD as $\text{den}_{\mathbf{I},\mathbf{d_I}}(\mathbf{t})$.

Here is an interpretation for the sentence '$(\forall x)(Px \supset Hf(x))$':

26. UD: The set of positive integers
P: {<**u**>: **u** is even}
H: {<**u**>: **u** is odd}
f: {<$\mathbf{u}_1$, $\mathbf{u}_2$>: $\mathbf{u}_2$ is the successor of $\mathbf{u}_1$ (the number that results from adding 1 to $\mathbf{u}_1$)}

On this interpretation the sentence '$(\forall x)(Px \supset Hf(x))$' may be read as 'The successor of any even positive integer is an odd positive integer', which is true. To see how satisfaction works here, consider an arbitrary variable assignment d_{26} for this interpretation:

- d_{26} satisfies '$(\forall x)(Px \supset Hf(x))$' if every variant $d_{26}[\mathbf{u}_1/x]$ satisfies '$Px \supset Hf(x)$'.
- If $\mathbf{u}_1 \notin I_{26}(P)$, then $d_{26}[\mathbf{u}_1/x]$ does not satisfy 'Px' and therefore does satisfy '$Px \supset Hf(x)$'.

- If $\mathbf{u}_1 \in I_{26}(P)$, then $d_{26}[\mathbf{u}_1/x]$ satisfies 'Px'. But $d_{26}[\mathbf{u}_1/x]$ also satisfies 'Hf(x)' because $<den_{\mathbf{I},d_{26}}(f(x))>$, which is $\mathbf{u}_1 + 1$, is odd if $\mathbf{u}_1$ is even.

We conclude that the sentence '$(\forall x)(Px \supset Hf(x))$' is true on interpretation 26.

Here is an interpretation for the sentence '$(\forall x)(\forall y)f(x,y) = f(y,x)$':

27. UD: The set of positive integers:
f: $\{<\mathbf{u}_1, \mathbf{u}_2, \mathbf{u}_3>: \mathbf{u}_3$ is the sum of $\mathbf{u}_1$ and $\mathbf{u}_2\}$

On this interpretation the sentence may be read as 'The sum of any two positive integers x and y is equal to the sum of y and x', which is true. Consider an arbitrary variable assignment d_{27} for this interpretation:

- d_{27} satisfies '$(\forall x)(\forall y)(f(x,y) = f(y,x))$' if every variant $d_{27}[\mathbf{u}_1/x, \mathbf{u}_2/y]$ satisfies '$f(x,y) = f(y,x)$'.
- Every variant $d_{27}[\mathbf{u}_1/x, \mathbf{u}_2/y]$ does satisfy '$f(x,y) = f(y,x)$' because $den_{\mathbf{I},d_{27}}(f(x,y))$ is $\mathbf{u}_1 + \mathbf{u}_2$ and $den_{\mathbf{I},d_{27}}(f(y,x))$ is $\mathbf{u}_2 + \mathbf{u}_1$, and these sums are obviously equal.

The same sentence is false on the following interpretation:

28. UD: The set of positive integers
$f(x,y)$: x raised to the power y

It is not true that, for any two positive integers x and y, x raised to the power y equals y raised to the power x. For example, 2 cubed equals 8, but 3 squared equals 9.

The sentence '$(\forall x)Dh(x,f(x))$' is true on interpretation 29 and false on interpretation 30:

29. UD: The set of positive integers
D: $\{<\mathbf{u}>: \mathbf{u}$ is even$\}$
f: $\{<\mathbf{u}_1, \mathbf{u}_2>: \mathbf{u}_2$ is $\mathbf{u}_1$ cubed$\}$
h: $\{<\mathbf{u}_1, \mathbf{u}_2, \mathbf{u}_3>: \mathbf{u}_3$ is the sum of $\mathbf{u}_1$ and $\mathbf{u}_2\}$

30. UD: The set of positive integers
D: $\{<\mathbf{u}>: \mathbf{u}$ is even$\}$
f: $\{<\mathbf{u}_1, \mathbf{u}_2>: \mathbf{u}_2$ is $\mathbf{u}_1$ doubled$\}$
h: $\{<\mathbf{u}_1, \mathbf{u}_2, \mathbf{u}_3>: \mathbf{u}_3$ is the sum of $\mathbf{u}_1$ and $\mathbf{u}_2\}$

It is true that the sum of any positive integer and that same integer cubed is even, for the cube of an even integer is even and the cube of an odd integer is odd. But it is false that the sum of any positive integer and that same integer doubled is even, for the result of doubling an odd integer is even, and so the sum of an odd integer and its double is odd.

When we produce an interpretation for sentences containing functors, it is important that we really have interpreted the functors as functions. For example, it may be tempting to come up with an interpretation with the set of positive integers as the UD on which 'f(x)' means 'the integer greater than x'. But this is not a function, for there are (infinitely!) many integers greater than any positive integer. A one-place function cannot map a member of the UD to more than one value. Similarly we cannot interpret 'h(x,y)' (with the same UD) as 'the integer that is a factor of both x and y', because two positive integers can have more than one factor in common.

It is also important, when we produce an interpretation for sentences containing functors, that the interpretation assigns a function that meets the following two conditions. First, an **n**-place function that is used to interpret an **n**-place functor must be defined for every **n**-tuple of members of the UD. For example, with the set of positive integers as the UD, we cannot interpret 'f(x)' to mean 'the integer that is the square root of x', since not every positive integer has an integral square root. Similarly we cannot interpret 'h(x,y)' to mean 'the integer that is the result of dividing x by y', since, for example, no integer is the result of dividing 5 by 3.

Second, even when an **n**-place function is defined for every **n**-tuple of members of the UD, we also require that the value of the function in each case be a member of the UD. So, if our UD is the set of positive integers, we also cannot interpret 'h(x,y)' to mean 'the *number* that is the result of dividing x by y'. Although the division function is defined for every pair of positive integers, the resulting value is not in every case a positive integer. For example, the result of dividing 5 by 3, namely, $\frac{5}{3}$, is not a positive integer. Nor can we interpret 'h(x,y)' to mean 'x minus y' if our UD is the set of positive integers, because, for example, 2 minus 3 is not a positive integer. Similarly, with the UD of positive integers, we cannot interpret 'f(x)' to mean 'the predecessor of x'. Not every positive integer has a positive integer as its predecessor, for the predecessor of the positive integer 1 is 0.

We may show semantic results for sentences containing functors either by producing interpretations that are sufficient to prove the result or by arguing generally that the sentences will have certain truth-values on every interpretation. For example, interpretations 27 and 28 establish that the sentence '$(\forall x)(\forall y)f(x,y) = f(y,x)$' is quantificationally indeterminate. On the other hand, the sentence '$(\forall x)(\exists y)y = f(x)$' is quantificationally true. The sentence is universally quantified and is true on an interpretation if for each x there is at least one y such that the pair x and y satisfies '$y = f(x)$'. This must be the case for any interpretation, since 'f' must be interpreted as a function that maps each member x of the UD to a member y of the UD.

The argument

$$\frac{(\forall x)\ Pf(x)}{Pf(f(a))}$$

is quantificationally valid. We must show that any interpretation that makes the premise true also makes the conclusion true. If '$(\forall x)\ Pf(x)$' is true, then every member x of the UD is such that $f(x)$ has the property P. Now, $f(a)$ is a member of the UD by our requirements for functor interpretations, so it follows from the universally quantified sentence that $f(f(a))$ must also have the property P, making the conclusion true as well.

The similar argument

$$\frac{(\forall x)\ Pf(f(x))}{Pf(a)}$$

is quantificationally invalid. Here is an interpretation on which the premise is true and the conclusion false:

31. UD: The set of positive integers
P: {<**u**>: **u** is greater than or equal to 3}
f: {$\mathbf{u}_1$, $\mathbf{u}_2$>: $\mathbf{u}_2$ is the successor of $\mathbf{u}_1$}
a: 1

For any positive integer x the successor of the successor of x is greater than or equal to 3, but the successor of 1 is 2, which is not greater than or equal to 3.

We may also expand sentences containing functors in order to use truth-tables to check for various properties, although again there will be a complication. We first note that the rules for expanding sentences containing complex terms are the same as the rules for expanding sentences without complex terms. For example, the sentence '$(\forall x)(Px \supset Hf(x))$' is expanded for the set of constants {'a'} by eliminating the universal quantifier and substituting 'a' for 'x' to obtain

$$Pa \supset Hf(a)$$

and expanding the same sentence for the set of constants {'a', 'b'} results in the conjunction

$$(Pa \supset Hf(a)) \ \& \ (Pb \supset Hf(b))$$

To expand the sentence

$$(\forall x)(\exists y)y = g(x) \supset (\forall x)(\exists y)Pf(x,y)$$

for the set of constants {'a', 'b'}, we expand the antecedent first to obtain

$$(\exists y)y = g(a) \ \& \ (\exists y)y = g(b)$$

and then to obtain

$$(a = g(a) \vee b = g(a)) \ \& \ (a = g(b) \vee b = g(b))$$

and we expand the consequent first to obtain

$$(\exists y)Pf(a,y) \;\&\; (\exists y)Pf(b,y)$$

and then to obtain

$$(Pf(a,a) \vee Pf(a,b)) \;\&\; (Pf(b,a) \vee Pf(b,b))$$

resulting in the expansion

$$((a = g(a) \vee b = g(a)) \;\&\; (a = g(b) \vee b = g(b))) \supset$$
$$((Pf(a,a) \vee Pf(a,b)) \;\&\; (Pf(b,a) \vee Pf(b,b)))$$

for the entire sentence.

Suppose now that we want to develop a truth-table for the expansion 'Pa $\supset$ P*f*(a)' of the sentence '$(\forall x)(Px \supset Pf(x))$' for the set of constants {'a'} and that, since 'Pa' and 'P*f*(a)' are distinct sentences involving the distinct individual terms 'a' and '*f*(a)', we decide that we can assign **T** to the antecedent and **F** to the consequent:

		↓
Pa	P*f*(a)	Pa $\supset$ P*f*(a)
T	**F**	**T F F**

MISTAKE!

Something is wrong here—because the sentence '$(\forall x)(Px \supset Pf(x))$' *cannot* be false on any interpretation with a one-member UD. If there is only one member a of the UD, then the only candidate for the value of *f*(a) is that one member—since we require that the value of a function applied to any member of a UD must be a member of the UD.

Our method of using truth-functional expansions to determine possible truth-values assumes that every member of the UD is named by one of the constants used in the expansion. For this reason we cannot assume that terms containing functors might refer to individuals other than those referred to by the constants used in the expansion. To the contrary, we must assume that each term containing a functor refers to the same individual as at least one constant. Thus in the above example we must assume that 'a' refers to the same individual as '*f*(a)', if the expansion is to tell us something about one-member UDs. We will make this explicit in our truth-table: The truth-value assignment must make the sentence 'a = *f*(a)' true.

		↓	↓
Pa	P*f*(a)	Pa $\supset$ P*f*(a)	a = *f*(a)
			T

And now our conditions 1 and 2 for truth-tables containing expansions of sentences with the identity predicate must apply. In particular, condition 2 requires

that, since 'a = f(a)' is true, the sentences 'Pa' and 'Pf(a)' must have the same truth-value. So the only shortened truth-tables we can obtain are

a = f(a)	Pa	Pf(a)	Pa ⊃ Pf(a)	a = f(a)
			↓	↓
T	**T**	**T**	**T T T**	**T**

and

a = f(a)	Pa	Pf(a)	Pa ⊃ Pf(a)	a = f(a)
			↓	↓
T	**F**	**F**	**F T F**	**T**

The shortened truth-tables show that the sentence '(∀x)(Px ⊃ Pf(x))' must be true on any interpretation with a one-member UD.

Generalizing, when we construct a truth-table for the truth-functional expansion of a sentence or set of sentences containing functors, the following condition must be met in addition to those for sentences containing the identity predicate:

3. For each **n**-place functor f occurring in one or more of the sentences being expanded and each sequence of **n** constants $\mathbf{a}_1, \ldots, \mathbf{a}_n$ from the set of constants $\{\mathbf{b}_1, \ldots, \mathbf{b}_m\}$ for which the sentence(s) is (are) being expanded, the sentence $(\ldots (f(\mathbf{a}_1, \ldots, \mathbf{a}_n) = \mathbf{b}_1 \vee f(\mathbf{a}_1, \ldots, \mathbf{a}_n) = \mathbf{b}_2) \vee \ldots \vee f(\mathbf{a}_1, \ldots, \mathbf{a}_n) = \mathbf{b}_m)$ must be true.

That is, the value that the function produces when applied to $\mathbf{a}_1, \ldots, \mathbf{a}_n$ must be named by one of the constants in the set of constants for which we are producing an expansion.

Let us now construct a truth-table for the truth-functional expansion of '(∀x)(Px ⊃ Pf(x))' for the set of constants {'a', 'b'}. We begin by adding two sentences to the right of the vertical line in order to satisfy condition 3, and we add the atomic components of those sentences to the left of the vertical line:

f(a) = a	f(a) = b	f(b) = a	f(b) = b	Pa	Pb	Pf(a)	Pf(b)

(Pa ⊃ Pf(a))	&	(Pb ⊃ Pf(b))	f(a) = a	∨	f(a) = b	f(b) = a	∨	f(b) = b
				↓			↓	
				T			**T**	

Let us now assign truth-values to the four identity sentences:

f(a) = a	f(a) = b	f(b) = a	f(b) = b	Pa	Pb	Pf(a)	Pf(b)
T	**F**	**F**	**T**				

					↓			↓			↓	
(Pa ⊃ P*f*(a))	&	(Pb ⊃ P*f*(b))	*f*(a) = a	∨	*f*(a) = b		*f*(b) = a	∨	*f*(b) = b		
			T	**T**	**F**		**F**	**T**	**T**		

By condition 2 for truth-tables for the expansions of sentences containing the identity predicate, 'Pa' and 'P*f*(a)' must have the same truth-value, because we have made '*f*(a) = a' true. And since we have made '*f*(b) = b' true, both 'Pb' and 'P*f*(b)' must have the same truth-value. Here, then, is one way of completing the assignment of values:

f(a) = a	*f*(a) = b	*f*(b) = a	*f*(b) = b	Pa	Pb	P*f*(a)	P*f*(b)
T	**F**	**F**	**T**	**T**	**F**	**T**	**F**

					↓			↓			↓	
(Pa ⊃ P*f*(a))	&	(Pb ⊃ P*f*(b))	*f*(a) = a	∨	*f*(a) = b		*f*(b) = a	∨	*f*(b) = b		
T T T	**T**	**F T F**	**T**	**T**	**F**		**F**	**T**	**T**		

And here is another (there are two additional ways, which we won't display):

f(a) = a	*f*(a) = b	*f*(b) = a	*f*(b) = b	Pa	Pb	P*f*(a)	P*f*(b)
T	**F**	**F**	**T**	**F**	**F**	**F**	**F**

					↓			↓			↓	
(Pa ⊃ P*f*(a))	&	(Pb ⊃ P*f*(b))	*f*(a) = a	∨	*f*(a) = b		*f*(b) = a	∨	*f*(b) = b		
F T F	**T**	**F T F**	**T**	**T**	**F**		**F**	**T**	**T**		

Note that the expansion '(Pa ⊃ P*f*(a)) & (Pb ⊃ P*f*(b))' had to come out true in both cases, since we have decided that a and *f*(a) are the same member of the UD and that b and *f*(b) are the same member of the UD.

Other ways of assigning truth-values to the identity sentences will make the expansion false—for example,

f(a) = a	*f*(a) = b	*f*(b) = a	*f*(b) = b	Pa	Pb	P*f*(a)	P*f*(b)
F	**T**	**T**	**F**	**T**	**F**	**F**	**T**

					↓			↓			↓	
(Pa ⊃ P*f*(a))	&	(Pb ⊃ P*f*(b))	*f*(a) = a	∨	*f*(a) = b		*f*(b) = a	∨	*f*(b) = b		
T F F	**F**	**F T T**	**F**	**T**	**T**		**T**	**T**	**F**		

We may also choose to make '*f*(a) = a', '*f*(a) = b', '*f*(b) = a', and '*f*(b) = b' all true. In this case we are required also to make the sentence 'a = b' true, because of the former two identities and condition 2; to make

'f(a) = f(b)' true, because of the latter two identities; and to make 'f(b) = a' true, by virtue of the truth of 'f(b) = b' and 'a = b'. As a consequence, 'Pa', 'Pb', 'Pf(a)', and 'Pf(b)' must all have the same truth-value, so in this case there are only two distinct shortened truth-tables:

f(a) = a	f(a) = b	f(b) = a	f(b) = b	Pa	Pb	Pf(a)	Pf(b)
T	**T**	**T**	**T**	**T**	**T**	**T**	**T**

		↓				↓				↓	
(Pa ⊃ Pf(a))	&	(Pb ⊃ Pf(b))	f(a) = a	∨	f(a) = b	f(b) = a	∨	f(b) = b
T T T	**T**	**T T T**	**T**	**T**	**T**	**T**	**T**	**T**

and

f(a) = a	f(a) = b	f(b) = a	f(b) = b	Pa	Pb	Pf(a)	Pf(b)
T	**T**	**T**	**T**	**F**	**F**	**F**	**F**

		↓				↓				↓	
(Pa ⊃ Pf(a))	&	(Pb ⊃ Pf(b))	f(a) = a	∨	f(a) = b	f(b) = a	∨	f(b) = b
F T F	**T**	**F T F**	**T**	**T**	**T**	**T**	**T**	**T**

The expanded sentence is true in both cases because the truth of 'a = b' means that our UD contains only one member, given the requirement that every member of the UD be named by one of the constants.

As a second and final example, we expand the sentence '(∀x)(∀y)(Dg(f(x), h(y)) ⊃ Dx)' for the set of constants {'a', 'b'} to obtain

((Dg(f(a), h(a)) ⊃ Da) & (Dg(f(a), h(b)) ⊃ Da)) &
((Dg(f(b), h(a)) ⊃ Db) & (Dg(f(b), h(b)) ⊃ Db))

Condition 3 requires us to make all of the following sentences true:

f(a) = a ∨ f(a) = b
f(b) = a ∨ f(b) = b
h(a) = a ∨ h(a) = b
h(b) = a ∨ h(b) = b
g(a,a) = a ∨ g(a,a) = b
g(a,b) = a ∨ g(a,b) = b
g(b,a) = a ∨ g(b,a) = b
g(b,b) = a ∨ g(b,b) = b

Let us suppose that we make all of these true by making the following identity sentences true:

1. $f(a) = a$
2. $f(b) = b$
3. $h(a) = b$
4. $h(b) = b$
5. $g(a,a) = a$
6. $g(b,a) = a$
7. $g(a,b) = b$
8. $g(b,b) = b$

and the rest of the atomic identity statements false. By conditions 1 and 2 we will then have the following true identities as well:

9.	$g(f(a), h(a)) = g(a,b)$	from $g(a,b) = g(a,b)$ and 1 and 3
10.	$g(f(a), h(b)) = g(a,b)$	from $g(a,b) = g(a,b)$ and 1 and 4
11.	$g(f(b), h(a)) = g(b,b)$	from $g(b,b) = g(b,b)$ and 2 and 3
12.	$g(f(b), h(b)) = g(b,b)$	from $g(b,b) = g(b,b)$ and 2 and 4
13.	$g(f(a), h(a)) = g(f(a), h(b))$	from 9 and 10
14.	$g(f(b), h(a)) = g(f(b), h(b))$	from 11 and 12

So '$Dg(f(a), h(a))$' and '$Dg(f(a), h(b))$' must have the same truth-value, and '$Dg(f(b), h(a))$' and '$Dg(f(b), h(b))$' must have the same truth-value. Here, then, is one shortened truth-table for the truth-functional expansion reflecting our choice of identities 1–8 and the consequences that follow by condition 2:

$f(a) = a$	$f(a) = b$	$f(b) = a$	$f(b) = b$	$h(a) = a$	$h(a) = b$	$h(b) = a$	$h(b) = b$
T	**F**	**F**	**T**	**F**	**T**	**F**	**T**

$g(a,a) = a$	$g(a,a) = b$	$g(a,b) = a$	$g(a,b) = b$	$g(b,a) = a$	$g(b,a) = b$	$g(b,b) = a$
T	**F**	**F**	**T**	**T**	**F**	**F**

$g(b,b) = b$	Da	Db	$Dg(f(a), h(a))$	$Dg(f(a), h(b))$	$Dg(f(b), h(a))$	$Dg(f(b), h(b))$
T	**T**	**F**	**F**	**F**	**T**	**T**

$((Dg(f(a), h(a)) \supset Da)$	&	$(Dg(f(a), h(b)) \supset Da))$	&	$((Dg(f(b), h(a)) \supset Db)$	&
			↓		
F **T T**	**T**	**F**	**T T**	**F**	**T**	**F F**	**F**

$$\begin{array}{llllllllllll}
 & & & & & & \downarrow & & & & \downarrow & & \\
(\mathrm{D}g(f(\mathrm{b}),\ h(\mathrm{b}))\ \supset\ \mathrm{D}\mathrm{b}))) & & f(\mathrm{a}) = \mathrm{a} & & \vee & & f(\mathrm{a}) = \mathrm{b} & & f(\mathrm{b}) = \mathrm{a} & \vee & f(\mathrm{b}) = \mathrm{b} \\
\hline
\mathbf{T} & & & \mathbf{F} & \mathbf{F} & & \mathbf{T} & \mathbf{T} & \mathbf{F} & & \mathbf{F} & \mathbf{T} & \mathbf{T}
\end{array}$$

$$\begin{array}{llllllllllll}
 & & \downarrow & & & & \downarrow & & & & \downarrow & & \\
h(\mathrm{a}) = \mathrm{a} & \vee & h(\mathrm{a}) = \mathrm{b} & & h(\mathrm{b}) = \mathrm{a} & \vee & h(\mathrm{b}) = \mathrm{b} & & g(\mathrm{a},\mathrm{a}) = \mathrm{a} & \vee & g(\mathrm{a},\mathrm{a}) = \mathrm{b} & \\
\hline
\mathbf{F} & \mathbf{T} & \mathbf{T} & & \mathbf{F} & \mathbf{T} & \mathbf{T} & & \mathbf{T} & \mathbf{T} & \mathbf{F} & &
\end{array}$$

$$\begin{array}{llllllllllll}
 & & \downarrow & & & & \downarrow & & & & \downarrow & & \\
g(\mathrm{a},\mathrm{b}) = \mathrm{a} & \vee & g(\mathrm{a},\mathrm{b}) = \mathrm{b} & & g(\mathrm{b},\mathrm{a}) = \mathrm{a} & \vee & g(\mathrm{b},\mathrm{a}) = \mathrm{b} & & g(\mathrm{b},\mathrm{b}) = \mathrm{a} & \vee & g(\mathrm{b},\mathrm{b}) = \mathrm{b} & \\
\hline
\mathbf{F} & \mathbf{T} & \mathbf{T} & & \mathbf{T} & \mathbf{T} & \mathbf{F} & & \mathbf{F} & \mathbf{T} & \mathbf{T} & &
\end{array}$$

This shortened truth-table, albeit not very short, shows that the sentence '$(\forall x)(\forall y)(\mathrm{D}g(f(x),\ h(y)) \supset \mathrm{D}x)$' is false on at least one interpretation with a one- or two-member UD. There are other shortened truth-tables showing that the sentence is true on at least one interpretation with a one- or two-member UD; and producing one of those will suffice to establish that the sentence is truth-functionally indeterminate.

8.6**E** EXERCISES

For these exercises, when you are asked to construct interpretations, you may specify the interpretations of predicates either by showing the **n**-tuples that comprise their extensions or in the manner of symbolization keys.

1. Determine the truth-values of the following sentences on this interpretation:

UD: The set of positive integers
E: {<**u**>: **u** is even}
G: {<$\mathbf{u}_1$, $\mathbf{u}_2$>: $\mathbf{u}_1$ is greater than $\mathbf{u}_2$}
O: {<**u**>: **u** is odd}
P: {$\mathbf{u}_1$, $\mathbf{u}_2$, $\mathbf{u}_3$>: $\mathbf{u}_1$ plus $\mathbf{u}_2$ equals $\mathbf{u}_3$}

a. $(\exists x)(\forall y)(x = y \supset Gxy)$
*b. $(\forall x)(\forall y) \sim x = y$
c. $(\forall x)(\exists y)(Oy \supset Gyx)$
*d. $(\forall x)(\forall y)(\forall z)[(Gxy \ \& \ Gyz) \supset \sim x = z]$
e. $(\exists w)[Ew \ \& \ (\forall y)(Oy \supset \sim w = y)]$
*f. $(\forall y)(\forall z)[(Oy \ \& \ y = z) \supset \sim Ez]$
g. $(\exists z)(\exists w)(z = w \ \& \ Gzw)$
*h. $(\forall x)(\forall y)(\exists z)[(Pxyz \ \& \sim x = z) \ \& \sim y = z]$
i. $(\forall x)(\forall y)(Pxyy \vee \sim x = y)$

2. Show that each of the following sentences is not quantificationally true by producing an interpretation on which it is false.
a. $(\exists x)(\forall y)x = y$
*b. $(\forall w)(w = b \supset Fw)$

c. $(\forall x)(\forall y)(\forall z)[(x = y \vee y = z) \vee x = z]$
*d. $(\exists w)[Gw \ \& \ (\forall z)(\sim Hzw \supset z = w)]$
e. $(\exists x)(\exists y)(\sim x = y \vee Gxy)$
*f. $(\forall x)(\forall y)(\exists z)(x = y \supset \sim x = z)$

3. Each of the following sentences is quantificationally true. Explain why.
a. $(\forall x)(\forall y)(\forall z)[(x = y \ \& \ y = z) \supset x = z]$
*b. $(\forall x)(\forall y)(\exists z)(x = z \vee y = z)$
c. $(\forall x)(\forall y)[x = y \supset (Gxy \equiv Gyx)]$

4. Show that the sentences in each of the following pairs are not quantificationally equivalent by constructing an interpretation on which one sentence is true and the other is false.
a. $(\forall x)(\exists y)\ x = y$, $(\forall x)(\forall y)x = y$
*b. $(\forall x)(\forall y)[x = y \supset (Fx \equiv Fy)]$, $(\forall x)(\forall y)[(Fx \equiv Fy) \supset x = y]$
c. $(a = b \vee a = c) \supset a = d$, $a = c \supset (a = b \vee a = d)$
*d. $(\exists x)(\forall y)(\sim x = y \supset Gy)$, $(\exists x)(\forall y)(Gy \supset \sim x = y)$

5. Show that each of the following sets of sentences is quantificationally consistent by constructing an interpretation on which each sentence in the set is true.
a. $\{a = b, a = c, \sim a = d\}$
*b. $\{(\forall x)(\forall y)x = y, (\exists x)Fx, (\forall y)Gy\}$
c. $\{(\exists x)(\exists z) \sim x = z, (\forall x)(\exists z)(\exists w)(x = z \vee x = w)\}$
*d. $\{(\forall x)(Gx \supset (\forall y)(\sim y = x \supset Gy)), (\forall x)(Hx \supset Gx), (\exists z)Hz\}$

6. Establish each of the following by producing an interpretation on which the set members are true and the sentence after the double turnstile is false.
a. $\{(\forall x)(\forall y)(\forall z)[(x = y \vee x = z) \vee y = z]\} \nvDash (\forall x)(\forall y)(x = y)$
*b. $\{(\exists w)(\exists z) \sim w = z, (\exists w)Hw\} \nvDash (\exists w) \sim Hw$
c. $\{(\exists w)(\forall y)Gwy, (\exists w)(\forall y)(\sim w = y \supset \sim Gwy)\} \nvDash (\exists z) \sim Gzz$
*d. $\{(\forall x)(\forall y)[(Fx \equiv Fy) \equiv x = y], (\exists z)Fz\} \nvDash (\exists x)(\exists y)[\sim x = y \ \& \ (Fx \ \& \sim Fy)]$

7. Using the given symbolization key, symbolize each of the following arguments in *PLE*. Then, for each symbolized argument, decide whether it is quantificationally valid and defend your answer.

UD: The set of all people
Fx: x is female
Mx: x is male
Lxy: x loves y
Pxy: x is a parent of y

a. Every male loves someone other than himself, and every male loves his children. Therefore no male is his own parent.
*b. Everyone loves her or his parents, and everyone has two parents. Therefore everyone loves at least two people.
c. A female who loves her children loves herself as well. Therefore every female loves at least two people.
*d. Everybody has exactly two parents. Therefore everybody has exactly four grandparents.
e. Nobody has three parents. Everybody has more than one parent. Therefore everybody has two parents.

8. Use truth-functional expansions to establish each of the following claims. Be sure that the truth-value assignments you produce meet the first two conditions discussed in this section.

a. The sentence '$(\exists x)(\exists y) \sim x = y$' is quantificationally indeterminate.

*b. The sentence '$(\forall w)(Fw \supset (\exists y) \sim y = w) \ \& \ (\exists w)Fw$' is quantificationally indeterminate.

c. The sentences '$(\forall y)(\forall z)[(Gyz \vee Gzy) \vee y = z]$' and '$(\forall y)(\exists z)Gyz$' are not quantificationally equivalent.

*d. The set of sentences $\{(\forall x)(\forall y)(\forall z)[(Gxy \vee Gyz) \vee x = z], (\forall y)(\exists z)Gyz\}$ is quantificationally consistent.

e. The set of sentences $\{(\forall y)y = y, (\exists z)(\exists w) \sim w = z\}$ does not quantificationally entail the sentence '$(\exists z)(\forall w) \sim z = w$'.

*f. The argument

$$\frac{(\forall y)(\forall z)(Gyz \supset y = z)}{(\forall y)(\forall z)(y = z \supset Gyz)}$$

is quantificationally invalid.

9. Determine the truth-values of the following sentences on this interpretation:

UD: The set of positive integers
E: $\{\langle \mathbf{u} \rangle$: $\mathbf{u}$ is even$\}$
G: $\{\langle \mathbf{u}_1, \mathbf{u}_2 \rangle$: $\mathbf{u}_1$ is greater than $\mathbf{u}_2\}$
f: $\{\langle \mathbf{u}_1, \mathbf{u}_2 \rangle$: $\mathbf{u}_2$ is the successor of $\mathbf{u}_1\}$
g: $\{\langle \mathbf{u}_1, \mathbf{u}_2 \rangle$: $\mathbf{u}_2$ is $\mathbf{u}_1$ squared$\}$
h: $\{\langle \mathbf{u}_1, \mathbf{u}_2, \mathbf{u}_3 \rangle$: $\mathbf{u}_3$ is the sum of $\mathbf{u}_1$ and $\mathbf{u}_2\}$

a. $(\forall x)Gf(x)x$

*b. $(\forall x)Eg(x)$

c. $(\forall x)(\exists y)y = h(x,x)$

*d. $(\forall x)(\forall y)(y = h(x,x) \supset Ey)$

e. $(\exists x)(\exists y)((Ex \ \& \sim Ey) \ \& \ Eh(x,y))$

*f. $(\forall x)(\forall y)(\forall z)(Eh(h(x,y), z) \supset ((Ex \vee Ey) \vee Ez))$

g. $(\forall x)(\exists z)(Eh(g(x), z) \vee Eh(x,g(z)))$

*h. $(\forall x)(\forall y)Gh(f(x), f(y)), h(x,y)$

10. Show that each of the following sentences is not quantificationally true by producing an interpretation on which it is false.

a. $(\forall x)(Pf(x) \supset Px)$

*b. $(\forall x)(\forall y)(x = g(y) \vee y = g(x))$

c. $(\exists x)(\forall y)x = g(y)$

*d. $(\forall x)(\forall y)(\forall z)((x = f(y) \ \& \ y = f(z)) \supset x = f(z))$

e. $(\forall x) \sim x = f(x)$

*f. $(\forall x)(\forall y)(Dh(x,y) \supset Dh(y,x))$

11. Each of the following sentences is quantificationally true. Explain why.

a. $(\forall x)(\exists y)y = f(f(x))$

*b. $(\forall x)(\forall y)(\forall z)((y = f(x) \ \& \ z = f(x)) \supset y = z)$

c. $((\forall x) Hxf(x) \ \& \ (\forall x)(\forall y)(\forall z)((Hxy \ \& \ Hyz) \supset Hxz)) \supset (\forall x)Hxf(f(x))$

12. Show that the sentences in each of the following pairs are not quantificationally equivalent by constructing an interpretation on which one sentence is true and the other false.

a. $\mathrm{Lab}f(\mathrm{b})$, $\mathrm{La}f(\mathrm{b})\mathrm{b}$

*b. $(\forall \mathrm{x})\mathrm{B}(h(\mathrm{x}), \mathrm{x})$, $(\forall \mathrm{x})\mathrm{B}(\mathrm{x},h(\mathrm{x}))$

c. $(\forall \mathrm{x})(\exists \mathrm{y})\mathrm{y} = f(h(\mathrm{x}))$, $(\exists \mathrm{z})\mathrm{z} = f(h(\mathrm{z}))$

*d. $(\exists \mathrm{x})(\exists \mathrm{y})(\exists \mathrm{z})(\mathrm{x} = f(\mathrm{y}) \ \& \ \mathrm{y} = f(\mathrm{z}))$, $(\forall \mathrm{x})(\exists \mathrm{y})(\exists \mathrm{z})(\mathrm{x} = f(\mathrm{y}) \ \& \ \mathrm{y} = f(\mathrm{z}))$

13. Show that each of the following sets of sentences is quantificationally consistent by constructing an interpretation on which each sentence in the set is true.

a. $\{\mathrm{a} = f(\mathrm{b}), \mathrm{b} = f(\mathrm{c}), \mathrm{c} = f(\mathrm{a})\}$

*b. $\{(\forall \mathrm{x})\mathrm{L}\mathrm{x}f(\mathrm{x}), (\exists \mathrm{y}) \sim \mathrm{L}f(\mathrm{y})\mathrm{y}\}$

c. $\{(\exists \mathrm{x})(\forall \mathrm{y})\mathrm{x} = f(\mathrm{y}), (\exists \mathrm{x})(\forall \mathrm{y}) \sim \mathrm{x} = f(\mathrm{y})\}$

*d. $\{(\forall \mathrm{x})(\mathrm{Gx} \supset \sim \mathrm{G}h(\mathrm{x}), (\exists \mathrm{x})(\sim \mathrm{Gx} \ \& \sim \mathrm{G}h(\mathrm{x})\}$

14. For each of the following arguments, decide whether it is quantificationally valid. If it is quantificationally valid, explain why. If it is not quantificationally valid, construct an interpretation on which the premises are true and the conclusion false.

a. $\dfrac{(\forall \mathrm{x})(\mathrm{Fx} \vee \mathrm{F}g(\mathrm{x}))}{(\forall \mathrm{x})(\mathrm{Fx} \vee \mathrm{F}g(g(\mathrm{x})))}$

*b. $\dfrac{(\forall \mathrm{x})(\mathrm{Fx} \vee \mathrm{F}g(\mathrm{x}))}{(\forall \mathrm{x})(\mathrm{F}g(\mathrm{x}) \vee \mathrm{F}g(g(\mathrm{x})))}$

c. $\dfrac{(\forall \mathrm{x})(\exists \mathrm{y})(\exists \mathrm{z})\mathrm{L}f(\mathrm{x})\mathrm{yz}}{(\exists \mathrm{x})(\forall \mathrm{y})(\forall \mathrm{z})\mathrm{L}\mathrm{x}f(\mathrm{y})f(\mathrm{z})}$

*d. $\dfrac{(\forall \mathrm{x})(\mathrm{L}\mathrm{x}f(\mathrm{x}) \ \& \sim \mathrm{L}f(\mathrm{x})\mathrm{x})}{(\forall \mathrm{x})(\forall \mathrm{y})(\mathrm{y} = f(\mathrm{x}) \supset (\mathrm{Lxy} \vee \mathrm{Lyx}))}$

e. $\dfrac{(\forall \mathrm{x})(\mathrm{B}g(\mathrm{x}) \supset (\forall \mathrm{y}) \sim \mathrm{H}\mathrm{y}g(\mathrm{x}))}{(\mathrm{a} = g(\mathrm{b}) \ \& \ \mathrm{Hca}) \supset \sim \mathrm{Ba}}$

15. Use truth-functional expansions to establish each of the following. Be sure that the truth-value assignments you produce meet all three conditions discussed in this section.

a. The sentence '$(\forall \mathrm{x})(\mathrm{Fx} \vee \mathrm{F}g(\mathrm{x}))$' is quantificationally indeterminate.

*b. The sentences '$(\exists \mathrm{x})(\exists \mathrm{y})\mathrm{H}g(\mathrm{x},\mathrm{y})\mathrm{x}$' and '$(\exists \mathrm{x})(\exists \mathrm{y})\mathrm{H}g(\mathrm{y},\mathrm{x})\mathrm{x}$' are not quantificationally equivalent.

c. The set of sentences $\{(\forall \mathrm{x}) \sim \mathrm{x} = f(\mathrm{x}), (\exists \mathrm{x})\mathrm{x} = f(f(\mathrm{x}))\}$ is quantificationally consistent.

*d. The argument

$$\frac{\mathrm{a} = f(\mathrm{b}) \ \& \ \mathrm{b} = f(\mathrm{a})}{(\exists \mathrm{x})(\exists \mathrm{y}) \sim \mathrm{x} = \mathrm{y}}$$

is quantificationally invalid.

GLOSSARY

QUANTIFICATIONAL TRUTH: A sentence **P** of *PL/PLE* is *quantificationally true* if and only if **P** is true on every interpretation.

QUANTIFICATIONAL FALSITY: A sentence **P** of *PL/PLE* is *quantificationally false* if and only if **P** is false on every interpretation.

QUANTIFICATIONAL INDETERMINACY: A sentence **P** of *PL/PLE* is *quantificationally indeterminate* if and only if **P** is neither quantificationally true nor quantificationally false.

QUANTIFICATIONAL EQUIVALENCE: Sentences **P** and **Q** of *PL/PLE* are *quantificationally equivalent* if and only if there is no interpretation on which **P** and **Q** have different truth-values.

QUANTIFICATIONAL CONSISTENCY: A set of sentences of *PL/PLE* is *quantificationally consistent* if and only if there is at least one interpretation on which all the members of the set are true. A set of sentences of *PL/PLE* is *quantificationally inconsistent* if and only if the set is not quantificationally consistent.

QUANTIFICATIONAL ENTAILMENT: A set Γ of sentences of *PL/PLE quantificationally entails* a sentence **P** of *PL/PLE* if and only if there is no interpretation on which every member of Γ is true and **P** is false.

QUANTIFICATIONAL VALIDITY: An argument of *PL/PLE* is *quantificationally valid* if and only if there is no interpretation on which all the premises are true and the conclusion is false. An argument of *PL/PLE* is *quantificationally invalid* if and only if the argument is not quantificationally valid.

Chapter 9

PREDICATE LOGIC: TRUTH-TREES

Section 9.1 augments the set of tree rules presented in Chapter 4 to include four rules for quantified sentences and negations of quantified sentences of *PL* and discusses guidelines for constructing trees for sentences and sets of sentences of *PL*. Section 9.2 redefines 'completed open branch' for truth-trees for *PL* and discusses the use of the tree method to test sets of sentences of *PL* for quantificational consistency, including how to recover interpretations from completed open branches. Section 9.3 discusses using truth-trees to test sentences and sets of sentences of *PL* for other quantificational properties. Section 9.4 revises one of the truth-tree rules for *PL* and presents a systematic method for constructing truth-trees. Section 9.5 presents truth-tree rules for *PLE*, and Section 9.6 fine-tunes the systematic method for constructing truth-trees, to guide the construction of truth-trees for sentences and sets of sentences of *PLE*.

9.1 TRUTH-TREE RULES FOR *PL*

Truth-trees, as developed in Chapter 4, provide the basis for an effective method of testing finite sets of sentences of *SL* for truth-functional consistency and thus for all the properties of sentences and finite sets of sentences that can be explicated in terms of truth-functional consistency. In this chapter we shall

augment the truth-tree method to make it applicable to sets of sentences of *PL* and of *PLE*. The result will be a method of testing finite sets of sentences of *PL* and of *PLE* for quantificational consistency and thus for those properties of sentences and finite sets of sentences that can be explicated in terms of quantificational consistency.

In addition to the decomposition rules for sentences of *SL*, which can also be used for *PL*, we need rules for decomposing sentences of *PL* that have any of the following four forms:

$(\forall \mathbf{x})\mathbf{P}$
$(\exists \mathbf{x})\mathbf{P}$
$\sim (\forall \mathbf{x})\mathbf{P}$
$\sim (\exists \mathbf{x})\mathbf{P}$

We begin with the rules for negations of quantified sentences. Both are nonbranching rules:

Negated Existential Decomposition ($\sim \exists$D)

$$\frac{\sim (\exists \mathbf{x})\mathbf{P}\checkmark}{(\forall \mathbf{x}) \sim \mathbf{P}}$$

Negated Universal Decomposition ($\sim \forall$D)

$$\frac{\sim (\forall \mathbf{x})\mathbf{P}\checkmark}{(\exists \mathbf{x}) \sim \mathbf{P}}$$

In each case the sentence entered is equivalent to the sentence being decomposed. 'It is not the case that something is such-and-such' is equivalent to 'Everything is such that it is not such-and-such', and 'It is not the case that everything is such-and-such' is equivalent to 'Something is not such-and-such'.

If a universally quantified sentence $(\forall \mathbf{x})\mathbf{P}$ is true on an interpretation, then so is each substitution instance $\mathbf{P}(\mathbf{a}/\mathbf{x})$ of that sentence. We therefore want a rule that allows us to "decompose" a universally quantified sentence to its substitution instances. The rule is

Universal Decomposition ($\forall$D)

$$\frac{(\forall \mathbf{x})\mathbf{P}}{\mathbf{P}(\mathbf{a}/\mathbf{x})}$$

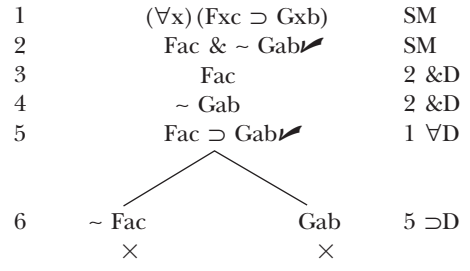
where **a** is any individual constant

At any point in the construction of a tree, a universally quantified sentence $(\forall \mathbf{x})\mathbf{P}$ may be decomposed by entering any substitution instance $\mathbf{P}(\mathbf{a}/\mathbf{x})$ of that sentence on *one or more* open branches passing through $(\forall \mathbf{x})\mathbf{P}$. Because a universally quantified sentence has an infinite number of substitution instances,

we can never "finish" decomposing such a sentence. Consequently universally quantified sentences are never checked off.

Universal Decomposition does not require that a selected substitution instance be entered on *every* open branch passing through the universally quantified sentence being decomposed. A substitution instance is often useful on one open branch passing through the sentence being decomposed but not on another. And, because universally quantified sentences are never checked off, we can always later add more substitution instances of a universally quantified sentence to any open branch passing through that sentence.

Here is a tree for the set of sentences {(∀x)(Fxc ⊃ Gxb), Fac & ~ Gab)}:



At line 5 we entered 'Fac ⊃ Gab' by Universal Decomposition. We could have entered any substitution instance of '(∀x)(Fxc ⊃ Gxb)', but only the one we did enter will end up producing a closed tree. Recall that we do not check off the universally quantified sentence that is being decomposed.

The rule for decomposing existentially quantified sentences is:

Existential Decomposition (∃D)

$$\frac{(\exists \mathbf{x})\mathbf{P}\checkmark}{\mathbf{P}(\mathbf{a}/\mathbf{x})}$$

where **a** is a constant foreign to the branch

A constant is foreign to a branch of a tree if and only if it does not occur in any sentence on that branch. Unlike universally quantified sentences, existentially quantified sentences are checked off when they are decomposed. This is because we know that if an existentially quantified sentence (∃**x**)**P** is true then there is at least one thing that is of the sort specified by **P**, but there need not be more than one such thing. When we choose an individual constant for the substitution instance **P**(**a**/**x**), the constant **a** that we choose must be foreign to the branch to which we plan to add **P**(**a**/**x**) because otherwise it would already have a role on that branch, and quite possibly a conflicting role. For example, the set {(∃x)Bx, (∃x) ~ Bx} is quantificationally consistent and should have only open truth-trees. However, if we were to drop the Existential

Decomposition restriction that **a** be foreign to the branch on which the substitution instance is entered, we could produce a closed tree for this set:

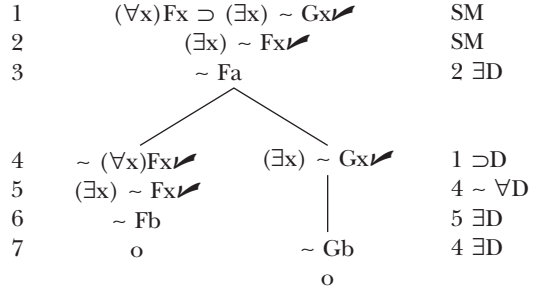
1	$(\exists x)\ Bx$	SM	
2	$(\exists x) \sim Bx$	SM	
3	Ba	1 $\exists$D	
4	$\sim Ba$	2 $\exists$D	**MISTAKE!**
	$\times$		

Line 4 is a mistake because the individual constant 'a' used in Existential Decomposition at line 4 was not foreign to the single branch of the tree prior to line 4. A correct tree uses an instantiating constant on line 4 that is different from that used on line 3:

1	$(\exists x)\ Bx$	SM
2	$(\exists x) \sim Bx$	SM
3	Ba	1 $\exists$D
4	$\sim Bb$	2 $\exists$D
	o	

The single branch is completed and this shows that the set is indeed quantificationally consistent.

The following tree contains three uses of Existential Decomposition:



At line 3 Existential Decomposition is used for the first time. Since no constant occurs on the single branch that constitutes the tree at that point, we used 'a' as the instantiating constant. The next use of Existential Decomposition is at line 6 on the left-hand branch. At that point 'a' already occurs on the branch (at line 3—remember that the sentences on lines 1–3 occur on both branches of this tree). So we use a new instantiating constant, 'b'. The final use of Existential Decomposition is at line 7 on the right-hand branch. The constant 'a' cannot be used because it occurs on line 3. But 'b' can be used, for although it already occurs on the left-hand branch, it does not occur before line 7 on the right-hand branch.

The preceding tree has two open branches, each of which contains only literals and decomposed nonliteral sentences. The complexities of predicate logic

will force us to complicate the account of 'completed open branch' given in Chapter 4. However, an open branch that contains only literals and decomposed nonliterals that have been checked off will, as in Chapter 4, count as a completed open branch. So both branches of the tree are completed open branches.

Moreover, a completed open branch guarantees that we can construct an interpretation on which every member of the set being tested is true, that is, a model for that set, so this tree demonstrates that the set {(∀x)Fx ⊃ (∃x) ~ Gx, (∃x) ~ Fx} is quantificationally consistent. We'll show how interpretations can be constructed from each of the two completed open branches of the tree.[1] An interpretation that makes all of the literals on a completed open branch true will make all of the other sentences on that branch, including the members of the set being tested, true. Starting with the left branch, we see that the branch contains two literals, '~ Fa' and '~ Fb'. To make both of these true we will construct an interpretation with a two-member UD, letting the constant 'a' denote one member and 'b' the other, and we will interpret the predicate 'F' so that neither of these members is in its extension:

UD: The set {1, 2}
a: 1
b: 2
F: ∅

Here we have shown the extension assigned to the predicate 'F' rather than construct a reading for 'F'. We shall continue this practice throughout this chapter, as the UDs for our interpretations will all be finite, and indeed quite small, and so there is no need to come up with a reading to specify the extension of a predicate. '~ Fa' and '~ Fb' are both true on any interpretation with this UD that assigns these values to 'a', 'b', and 'F'. The sentence '(∀x)Fx ⊃ (∃x) ~ Gx' is true on any interpretation that assigns these values because the antecedent is false (so it doesn't matter how the predicate 'G' is interpreted). The sentence '(∃x) ~ Fx' is true because at least one member of the UD is excluded (in fact, both are) from the extension of the predicate 'F'.

Note that the truth-values of the sentences '(∀x)Fx ⊃ (∃x) ~ Gx' and '(∃x) ~ Fx' don't depend on the assignments made to 'a' and 'b', since those constants don't appear in these sentences. So more generally we can say that the set members will be true on any interpretation that includes the following assignments:

UD: The set {1, 2}
F: ∅

Interestingly, in this case we do not need a two-member UD either. This is because the literals on the complete open branch, '~ Fa' and '~ Fb', agree

[1] In Chapter 11 we will *prove* that such an interpretation can always be found corresponding to a completed open branch of a tree for sentences of *PL*.

in what they say about the individual denoted by ‘a’ and the individual denoted by ‘b’—so the UD for an interpretation that makes the literals on the open branch true need not contain more than one individual. But our purpose in this chapter is to show how completed open branches can be used as the basis for constructing models, rather than to explore the finer points thereof, so while working with the language *PL* (rather than *PLE*) our practice will be to present a UD with exactly as many members as there are constants on the open branch, to assign distinct members of the UD to those constants, and to interpret predicates so as to make the literals occurring on the branch true.

The right-hand open branch of the previous tree contains two constants as well, and the literals ‘~ Fa’ and ‘~ Gb’, so the members of the set {(∀x)Fx ⊃ (∃x) ~ Gx, (∃x) ~ Fx} will both be true on any interpretation with a two-member UD such that ‘a’ and ‘b’ denote distinct individuals and the two literals are true. Any interpretation that includes the following assignments will satisfy these criteria:

UD:	{1, 2}
a:	1
b:	2
Fx:	∅
Gx:	∅

The sentence ‘(∀x)Fx ⊃ (∃x) ~ Gx’ will be true because the antecedent is false and the consequent is true, while ‘(∃x) ~ Fx’ will be true because at least one member of the UD (in fact, both) is excluded from the extension of the predicate ‘F’.

Except for Universal Decomposition, the truth-tree rules introduced in this section are like the tree rules of Chapter 4 in that the results of applying one of them must be entered on *every* open branch running through the sentence being decomposed. Also as in Chapter 4, it is generally wise to apply decomposition rules that do not produce new branches before applying those that do. In using Universal Decomposition it is a good idea to select substitution instances in which the instantiating constant already occurs on the open branch in question. It is also wise to try to use Existential Decomposition before using Universal Decomposition, for the former rule but not the latter places a restriction on the individual constant that can be used in the substitution instance that is added to the tree. We illustrate these last two points by constructing a tree for {(∀x)(∀y) ~ Mxy, (∃x)Mxb}:

1	(∀x)(∀y) ~ Mxy	SM
2	(∃x)Mxb✓	SM
3	Mab	2 ∃D
4	(∀y) ~ May	1 ∀D
5	~ Mab	4 ∀D
	×	

Note that we used Existential Decomposition before Universal Decomposition. At line 4 we entered '(∀y) ~ May' rather than, say, '(∀y) ~ Mgy', because 'a' occurs earlier on the tree on line 3. And although 'b' also occurs on line 3 we entered '(∀y) ~ May' rather than '(∀y) ~ Mby' because the former but *not* the latter will, when appropriately decomposed, produce the negation of the sentence on line 3.

Using Universal Decomposition before Existential Decomposition—that is, decomposing the sentence on line 1 before the sentence on line 2—will also produce a closed tree, but a tree that is more complex:

1	(∀x)(∀y) ~ Mxy	SM
2	(∃x)Mxb✓	SM
3	(∀y) ~ Mby	1 ∀D
4	~ Mbb	3 ∀D
5	Mab	2 ∃D
6	(∀y) ~ May	1 ∀D
7	~ Mab	6 ∀D
	×	

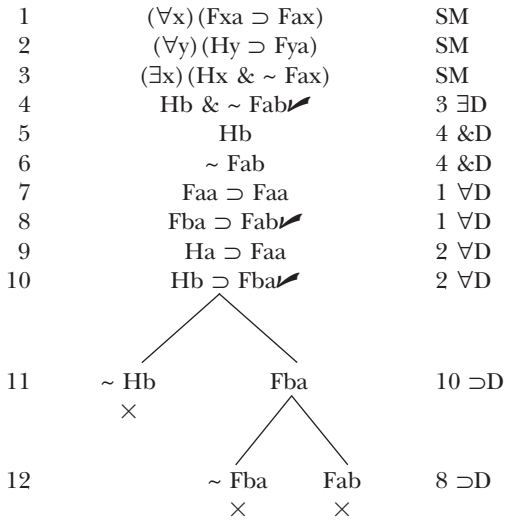
In this tree we had to enter 'Mab', rather than 'Mbb', at line 5 because Existential Decomposition requires that the instantiating constant be foreign to the branch. At line 5, the constant 'b' was not foreign to the branch. But now, having entered 'Mab' at line 5, we were able to close the tree only by reapplying Universal Decomposition to the sentence on line 1. Lines 3 and 4 of the tree are thus superfluous.

In Chapter 4 we developed four guidelines for keeping truth-trees for sets of sentences of *SL* as concise as possible. Those guidelines are also applicable here. We repeat them, along with the two new guidelines just discussed (suitably rearranged):

Guidelines for Constructing Truth-Trees

1. Stop when a tree yields the answer to the question being asked.
2. Give priority to decomposing sentences whose decomposition does not require branching.
3. Give priority to decomposing sentences whose decompositions result in the closing of one or more branches.
4. Give priority to decomposing existentially quantified sentences over universally quantified sentences.
5. When applying Universal Decomposition, try to use a substitution instance in which the instantiating constant already occurs on the branch to which the substitution instance will be added.
6. When guidelines 1–6 are not applicable, decompose the more complex sentences first.

Guideline 1 should be used with care when dealing with universally quantified sentences. Consider the following tree in which Universal Decomposition is used before Conditional Decomposition:



At line 7 we used Universal Decomposition and continued using it until each universally quantified sentence (there are two) was decomposed to every substitution instance that could be formed from a constant already on the branch. The idea is that these are the substitution instances that may be useful later on. As it turns out, lines 7 and 9 are unnecessary, but this was not completely obvious at the point where we had a choice between applying Universal Decomposition and Material Conditional Decomposition. On the other hand, we might have noticed that 'Hb' and '~Fab' already occurred on lines 5 and 6, and we might therefore have decided to first instantiate the universally quantified sentences on lines 1 and 2 with the constant 'b'—to see whether these instantiations might produce sentences that can be decomposed to close some branches.

Generally, it is a good policy to stick with guideline 1, but with the caveat that, when a shorter route to a closed tree is apparent, it should be pursued.[2]

9.I**E** EXERCISES

1. Construct truth-trees for the following sets of sentences. For each, note whether the tree you construct has a completed open branch or is closed (by the accounts of 'completed open branch' and 'closed tree' given in Chapter 4).

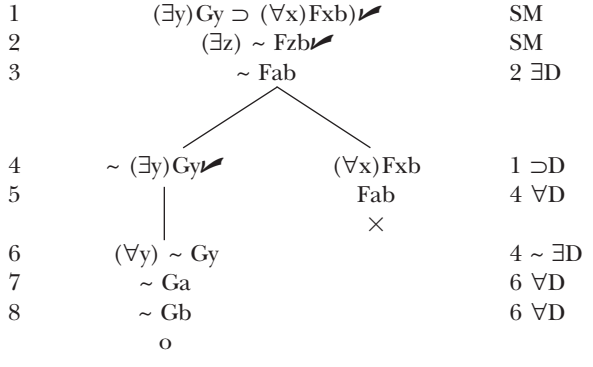
a. {(∃x)Fx, (∃x) ~ Fx}
*b. {(∃x)Fx, (∀x) ~ Fx}

[2]A further caveat will be required when we introduce systematic trees in Section 9.4, for the routine for constructing such trees requires abandoning guideline 1 altogether as it applies to universally quantified sentences.

c. {(∃x)(Fx & ~ Gx), (∀x)(Fx ⊃ Gx)}
*d. {(∃x)(Fx & ~ Gx), (∀x)Fx ⊃ (∀x)Gx}
e. {~ (∀x)(Fx ⊃ Gx), ~ (∃x)Fx, ~ (∃x)Gx}
*f. {~ (∀x)(Fx & Gx), (∃y)(Fy & Gy)}
g. {(∃x)Fx, (∃y)Gy, (∃z)(Fz & Gz)}
*h. {(∀x)(Fx ⊃ Gx), (∀x)(Gx ⊃ Hx), (∃x)(Fx & ~ Hx)}
i. {(∀x)(∀y)(Fxy ⊃ Fyx), (∃x)(∃y)(Fxy & ~ Fyx)}
*j. {(∀x)(∃y)Lxy, Lta & ~ Lat, ~ (∃y)Lay}
k. {(∃x)Fx ⊃ (∀x)Fx, ~ (∀x)[Fx ⊃ (∀y)Fy]}
*l. {(∀x)(Fx ⊃ Gx), ~ (∀x) ~ Fx, (∀x) ~ Gx}
m. {(∀x)[Fx ⊃ (∃y)Gyx], ~ (∀x) ~ Fx, (∀x)(∀y) ~ Gxy}
*n. {(∃x)Gx ⊃ (∀x)Gx, (∃z)Gz & (∃y) ~ Gy}
o. {(∃x)Lxx, ~ (∃x)(∃y)(Lxy & Lyx)}
*p. {(∃y)(Fy ∨ Gy), ~ (∀y)Fy & ~ (∀y)Gy, ~ (∀x)(Fx & Gx)}
q. {(∃x)(Fx ∨ Gx), (∀x)(Fx ⊃ ~ Gx), (∀x)(Gx ⊃ ~ Fx), ~ (∃x)(~ Fx ∨ ~ Gx)}

9.2 TRUTH-TREES AND QUANTIFICATIONAL CONSISTENCY

In Chapter 4 we defined a *completed open branch* to be an open branch on which every sentence either is a literal or has been decomposed, so that no new sentence can be added to the branch. We will have to revise this definition for trees for sets of sentences of *PL*. Consider the following tree for the set {(∃y)Gy ⊃ (∀x)Fxb, (∃z) ~ Fzb}:



This tree has one closed branch and one open branch. Further sentences could be added to the open branch, for it contains the universally quantified sentence '(∀y) ~ Gy', and there is no limit to the number of times a universally quantified sentence can be decomposed—such sentences are never checked off. In this tree we added substitution instances of '(∀y) ~ Gy' on

lines 7 and 8, using individual constants that already appeared on the branch. While further substitution instances can be added, it is clear that, no matter how many substitution instances we add, the branch will remain open. The truth-values of substitution instances formed from individual constants that do not already occur on the open branch will not have any bearing on the truth-values of literals that already occur on the branch, so there is no point in entering '~ Gh', for example. The leftmost branch extending through line 8 is sufficient for concluding that the set being tested is quantificationally consistent—we can use the literals '~ Fab', '~ Ga', and '~ Gb' that occur on this branch to construct an interpretation on which all of the set members are true. They'll be true on any interpretation that includes the following assignments:

UD: The set {1, 2}
a: 1
b: 2
F: {<2,1>}
G: ∅

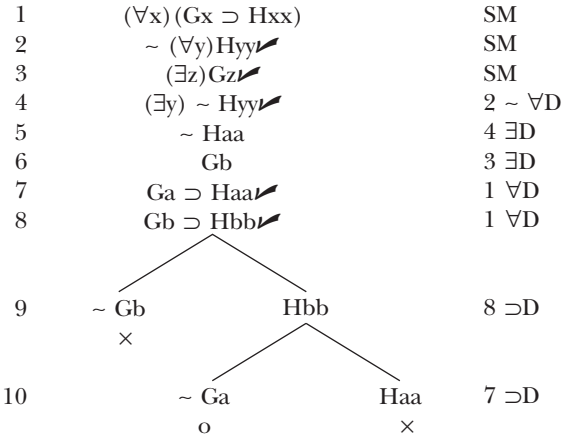
'$(\exists y)Gy \supset (\forall x)Fxb$' will be true because the antecedent is false—the extension of 'G' is empty—while '$(\exists z) \sim Fzb$' will be true because <1, 2>, for example, is not in the extension of 'F'.

We want open branches such as the left branch on the preceding tree—open branches to which no additional useful sentences can be added—to count as completed open branches. We therefore modify our definition of a completed open branch as follows:

> A branch of a truth-tree for a set of sentences of *PL* is a **completed open branch** if and only if it is a finite open branch (that is, an open branch with a finite number of sentences) and each sentence occurring on the branch is one of the following:
>
> 1. A literal (an atomic sentence or the negation of an atomic sentence)
> 2. A nonliteral sentence that is not a universally quantified sentence and is decomposed
> 3. A universally quantified sentence $(\forall \mathbf{x})\mathbf{P}$ such that at least one substitution instance occurs on the branch, and for each constant **a** occurring on the branch, the substitution instance $\mathbf{P}(\mathbf{a}/\mathbf{x})$ also occurs on the branch.

By this revised account the leftmost branch of the preceding tree is a completed open branch.

Here is another tree that contains a completed open branch:



The open branch is completed because each compound sentence that is not a universal quantification has been checked off, and the single universally quantified sentence has been decomposed using each of the two constants on the branch, at lines 7 and 8. The branch contains sufficient information for constructing a model of the set being tested. To make the literals on the completed open branch true, we need to interpret the predicates 'G' and 'H' in such a way that 'Gb' and 'Hbb' are true (since 'Gb' and 'Hbb' occur on the open branch) and 'Ga' and 'Haa' are false (since '~ Ga' and '~ Haa' occur on the open branch). The following assignments will do the trick:

UD: {1, 2}
a: 1
b: 2
G: {<2>}
H: {<2, 2>}

Any interpretation that includes these assignments will make the three sentences in the set {(∀x)(Gx ⊃ Hxx), ~ (∀y)Hyy, (∃z)Gz} true, and this establishes that the set is quantificationally consistent. An interpretation on which all the members of the set being tested are true can always be constructed from a completed open branch (we shall prove this in Chapter 11), so we shall take the presence of a completed open branch as a guarantee that the set being tested is quantificationally consistent.

To see why we require that a completed open branch on which a universally quantified sentence occurs must contain *at least one* substitution instance

of that sentence, consider the unit set {(∀x)(Fx & ~ Fx)}. The sole member of this set is quantificationally false. We therefore want every tree for the unit set of this sentence to close. One tree is as follows:

1	(∀x)(Fx & ~ Fx)	SM
2	Fa & ~ Fa	1 ∀D
3	Fa	2 &D
4	~ Fa	2 &D
	×	

If we did not require that a completed open branch contain at least one substitution instance of every universally quantified sentence occurring on that branch, we would have a completed open branch at line 1. A completed open branch is supposed to signal a consistent set, but the set {(∀x)(Fx & ~ Fx)} is not consistent. Given the requirement that a completed open branch must have at least one substitution instance of each universally quantified sentence occurring on that branch, we entered such an instance on line 2 and this led to a closed tree. Note that the tree would close no matter what substitution instance of '(∀x)(Fx & ~ Fx)' is entered at line 2.

We summarize here the important properties of truth-trees for sets of sentences of *PL*. With the exception of the concept of a completed open branch, these definitions strictly parallel those given in Chapter 4:

Closed branch:	A branch on which contradictory literals occur.
Open branch:	A branch that is not closed.
Completed open branch:	A finite open branch on which each sentence is one of the following:

1. A literal (an atomic sentence or the negation of an atomic sentence)
2. A compound sentence that is not a universally quantified sentence and is decomposed
3. A universally quantified sentence (∀**x**)**P** such that at least one substitution instance occurs on the branch, and for each constant **a** occurring on the branch, the substitution instance **P**(**a**/**x**) also occurs on the branch.

Closed truth-tree:	A truth-tree each of whose branches is closed.
Open truth-tree:	A truth-tree that is not closed.
Completed truth-tree:	A truth-tree each of whose branches is either closed or a completed open branch.

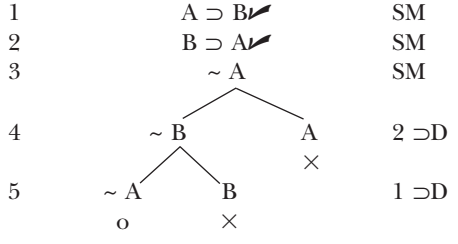
As noted, we will prove the following claims in Chapter 11:

> A finite set Γ of sentences of *PL* is *quantificationally inconsistent* if and only if Γ has a closed truth-tree.

> A finite set Γ of sentences of *PL* is *quantificationally consistent* if and only if Γ does not have a closed truth-tree.

If we can construct a closed tree for a finite set of sentences, then we can conclude that that set is quantificationally inconsistent. If we can construct a tree with a completed open branch for a finite set of sentences of *PL*, we may conclude that the set is quantificationally consistent.[3] However, in *PL,* unlike *SL,* not all consistent finite sets have trees with completed open branches: some such sets have trees all of whose open branches are infinite (we require a completed open branch to be finite).[4] That is why our second box, characterizing consistency, does so in negative terms: a finite set is quantificationally consistent if and only if it does not have a closed truth-tree (rather than if and only if it has a tree with a completed open branch).

There is another importance difference between quantificational and sentential truth-trees. In the sentential case, we can say that if all of the members of the set we are testing are true on some truth-value assignment, then all of the sentences on at least one completed open branch must be true on that assignment. So, for example, the following tree for the set {A ⊃ B, B ⊃ A, ~ A} has exactly one completed open branch:



If all of the sentences in the set are true on some truth-value assignment, then the final two sentences on the single open branch, '~ B' and '~ A', must also

[3]As we noted in Chapter 4, truth-trees can only be used to test finite sets of sentences. In Chapter 11 we shall prove that an infinite set of sentences of *PL* is quantificationally consistent if and only if every finite subset of that set is quantificationally consistent. Therefore, we can also say that an infinite set Γ of sentences of *PL* is quantificationally inconsistent if and only if at least one finite subset of Γ has a closed truth-tree and that an infinite set Γ of sentences of *PL* is quantificationally consistent if and only if no finite subset of Γ has a closed truth-tree.

[4]We discuss such trees in detail in Section 9.5.

be true on that assignment. A similar claim does not hold for quantificational trees. Consider the following simple tree for the set {(∃x)Fx}:

1	(∃x)Fx✓	SM
2	Fa	1 ∃D
	o	

The tree for this set has only one completed open branch, and from this open branch we can conclude that there is a model for the set, for example, any interpretation that includes the following assignments:

UD: {1}
a: 1
F: {<1>}

for such an interpretation will make the single literal on the branch true. But unlike the situation for *SL,* the truth of '(∃x)Fx' *does not require* the truth of 'Fa'. For example, '(∃x)Fx' is also true on any interpretation that includes the following assignments:

UD: {1, 2}
a: 2
Fx: {<1>}

'Fa' is false on such an interpretation. The completed open branch shows that the set {(∃x)Fx} is consistent not because the truth of '(∃x)Fx' requires the truth of 'Fa' but rather because the truth of 'Fa' is sufficient to guarantee the truth of '(∃x)Fx'—as illustrated by the first of these interpretations.

Although the truth of '(∃x)Fx' does not require the truth of 'Fa', it *is* the case that if '(∃x)Fx' is true on some interpretation, then there must be an intrepretation on which 'Fa' is true. If there is an interpretion on which something is F then we can construct an interpretation in which 'a' designates that thing (leaving the interpretation of 'F' unchanged) so that 'Fa' will be true. It is for this reason that the following tree establishes the quantificational inconsistency of {(∃x)(Fx & ~ Fx), even though the truth of an existentially quantified formula does not require the truth of any particular one of its substitution instances:

1	(∃x)(Fx & ~ Fx)✓	SM
2	Fa & ~ Fa✓	1 ∃D
3	Fa	2 &D
4	~ Fa	2 &D
	×	

If there is indeed something that both is and is not F, then there is an interpretation that assigns that thing to 'a' and 'Fa & ~ Fa' will be true on that interpretation. Because the single branch of this tree is closed, we may conclude that there is no interpretation on which '(∃x)(Fx & ~ Fx)' is true.

9.2**E** EXERCISES

1. Use the truth-tree method to test the following sets of sentences for quantificational consistency. State your result, and specify what it is about the tree that establishes this result. In addition, if your tree establishes consistency, show the relevant part of an interpretation that will make all of the literals on one completed open branch true.

a. {(∀x)Fx ∨ (∃y)Gy, (∃x)(Fx & Gb)}
*b. {(∀x)Fx ∨ (∃y)Gy, (∃x)(~ Fx & Gx)}
c. {(∀x)(Fx ⊃ Gxa), (∃x)Fx, (∀y) ~ Gya}
*d. {(∀x)(Fx ⊃ Gxa), (∃x)Fx}
e. {(∀x)(Fx ⊃ Gxa), (∃x)Fx, (∀y)Gya}
*f. {(∀x)(Fx ⊃ Gxa), (∃x)Fx, (∀x)(∀y)Gxy}
g. {(∀x)(Fx ∨ Gx), ~ (∃y)(Fy ∨ Gy)}
*h. {(∀x)(Fx ∨ Gx), ~ (∃y)(Fy ∨ Gy), Fa & ~ Gb}
i. {(∀z)Hz, (∃x)Hx ⊃ (∀y)Fy}
*j. {(∀z) ~ Hzb, (∃y)Fy ⊃ (∃x)Hxc}
k. {(∀x)(∀y)Lxy, (∃z) ~ Lza ⊃ (∀z) ~ Lza}
*l. {(∀x)(∀y)Lxy, (∃z) ~ Lza ⊃ (∀z) ~ Lzb}
m. {(∀x)(Rx ≡ ~ Hxa), ~ (∀y) ~ Hby, Ra}
*n. {(∀x)Fxa ≡ ~ (∀x)Gxb, (∃x)(Fxa & ~ Gxb)}

9.3 TRUTH-TREES AND OTHER SEMANTIC PROPERTIES

To facilitate the use of truth-trees to test sentences and sets of sentences for properties other than consistency, we first specify those other properties in terms of open and closed truth-trees.

A sentence **P** of *PL* is *quantificationally true* if and only if {~ **P**} has a closed truth-tree.

A sentence **P** of *PL* is *quantificationally false* if and only if {**P**} has a closed truth-tree.

A sentence **P** of *PL* is *quantificationally indeterminate* if and only if neither {**P**} nor {~ **P**} has a closed truth-tree.

Quantificational equivalence, quantificational entailment, and quantificational validity are specified analogously:

> Sentences **P** and **Q** of *PL* are *quantificationally equivalent* if and only if {~ (**P** ≡ **Q**)} has a closed truth-tree.

> A finite set Γ of sentences of *PL quantificationally entails* a sentence **P** of *PL* if and only if Γ ∪ {~ **P**} has a closed truth-tree.

> An argument of *PL* with a finite set of premises is *quantificationally valid* if and only if the set consisting of the premises and the negation of the conclusion has a closed truth-tree.

Consider the sentence '(∀x)(Fx & (∃y) ~ Fy)', which says 'Each thing is F and at least one thing is not F', a claim for which we should not hold out much hope. To verify that this sentence is quantificationally false, we construct a tree for the unit set of this sentence, and indeed this tree closes:

1	(∀x)(Fx & (∃y) ~ Fy)	SM
2	Fa & (∃y) ~ Fy✓	1 ∀D
3	Fa	2 &D
4	(∃y) ~ Fy✓	2 &D
5	~ Fb	4 ∃D
6	Fb & (∃y) ~ Fy✓	1 ∀D
7	Fb	6 &D
8	(∃y) ~ Fy	6 &D
	×	

Since the tree closes, we know that the set being tested is quantificationally inconsistent. Therefore there is no interpretation on which the single sentence in that set, '(∀x)(Fx & (∃y) ~ Fy)', is true. Hence the sentence is indeed quantificationally false. Note that we applied Universal Decomposition to the sentence on line 1 twice—once to obtain the sentence on line 2 and once to obtain the sentence on line 6. This was necessary because, by the time we reached line 5, we had introduced a new constant with which the universally quantified sentence on line 1 had not yet been decomposed.

Now consider the sentence '(∃x) ~ Fx ⊃ ~ (∀x)Fx', which says 'If there is something that is not F, then not everything is F' and is obviously quantificationally true. To verify this, we construct a tree for the unit set of its negation, {~ [(∃x) ~ Fx ⊃ ~ (∀x)Fx]} (note that in forming the negation of this

truth-functionally compound sentence we were careful to reinstate the outer brackets that had been omitted):

1	~ [(∃x) ~ Fx ⊃ ~ (∀x)Fx]✓	SM
2	(∃x) ~ Fx✓	1 ~ ⊃D
3	~ ~ (∀x)Fx✓	1 ~ ⊃D
4	(∀x)Fx	3 ~ ~ D
5	~ Fa	2 ∃D
6	Fa	4 ∀D
	×	

This tree is closed, so the set being tested is quantificationally inconsistent and we may conclude that '(∃x) ~ Fx ⊃ ~ (∀x)Fx' is quantificationally true.

Of course, we do not always have a clear intuition about a sentence's quantificational status. Consider, for example, the sentence '(∃x)(Fx ⊃ (∀y) Fy)', which may appear on first encounter to be quantificationally indeterminate. It is if and only if both the tree for '(∃x)(Fx ⊃ (∀y)Fy)' and the tree for its negation have at least one completed open branch. We begin with a tree for the unit set of the sentence, to determine whether the sentence is quantificationally false:

1	(∃x)(Fx ⊃ (∀y)Fy)✓		SM
2	Fa ⊃ (∀y)Fy✓		1 ∃D
3	~ Fa	(∀y)Fy	2 ⊃D
4	o	Fa	3 ∀D
		o	

As expected, the tree is open, so the sentence is not quantificationally false. We next construct a tree for the unit set of the negation of the sentence, to determine whether the sentence is quantificationally true:

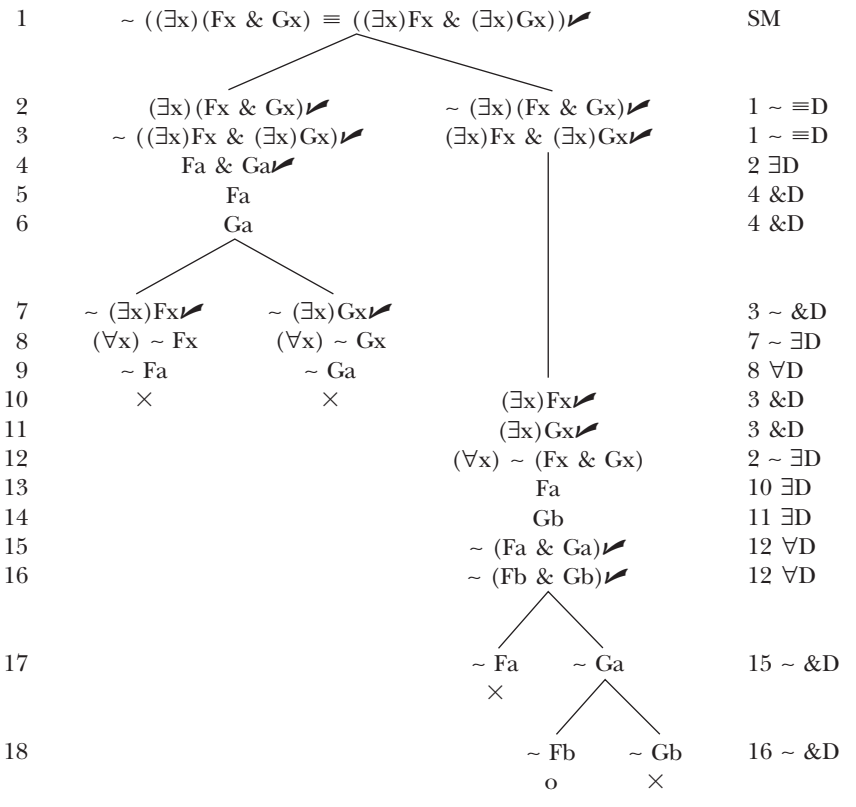
1	~ (∃x)(Fx ⊃ (∀y)Fy)✓	SM
2	(∀x) ~ (Fx ⊃ (∀y)Fy)	1 ~ ∃D
3	~ (Fa ⊃ (∀y)Fy)✓	2 ∀D
4	Fa	3 ~ ⊃D
5	~ (∀y)Fy✓	3 ~ ⊃D
6	(∃y) ~ Fy✓	5 ~ ∀D
7	~ Fb	6 ∃D
8	~ (Fb ⊃ (∀y)Fy)✓	2 ∀D
9	Fb	8 ~ ⊃D
10	~ (∀y)Fy	8 ~ ⊃D
	×	

Perhaps surprisingly, this tree is closed. So the sentence '$(\exists x)(Fx \supset (\forall y)Fy)$' is in fact quantificationally true.

Insufficient attention to the scope of quantifiers might lead one to think that the sentences '$(\exists x)(Fx \,\&\, Gx)$' and '$(\exists x)Fx \,\&\, (\exists x)Gx$' are quantificationally equivalent and hence that

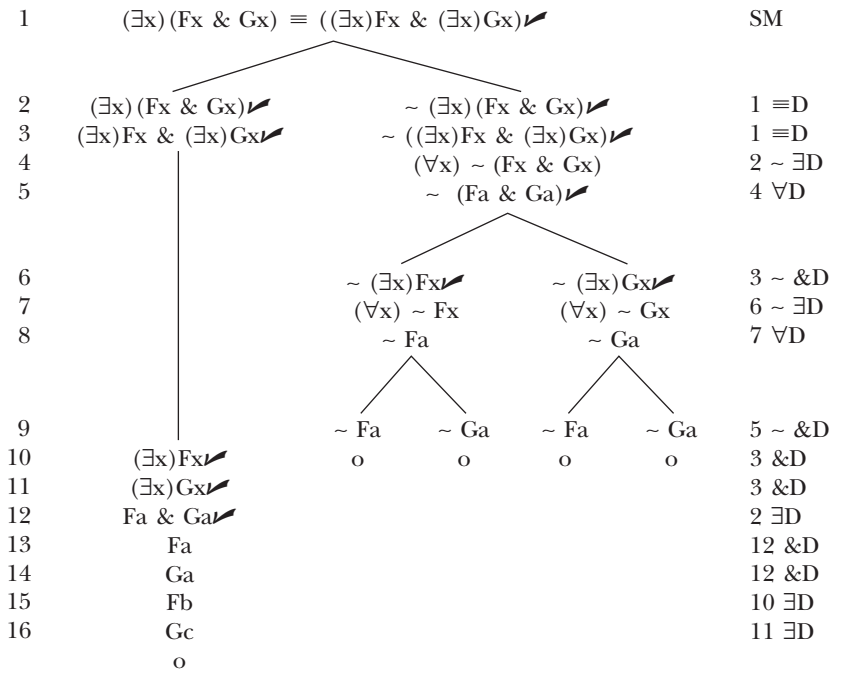
$$(\exists x)(Fx \,\&\, Gx) \equiv ((\exists x)Fx \,\&\, (\exists x)Gx)$$

is quantificationally true. To test this supposition, we construct a tree for the negation of the biconditional:



The tree has a completed open branch, so the sentences '$(\exists x)(Fx \,\&\, Gx)$' and '$(\exists x)Fx \,\&\, (\exists x)Gx$' are not quantificationally equivalent. If we are interested in using the tree method to establish that the biconditional '$(\exists x)(Fx \,\&\, Gx) \equiv$

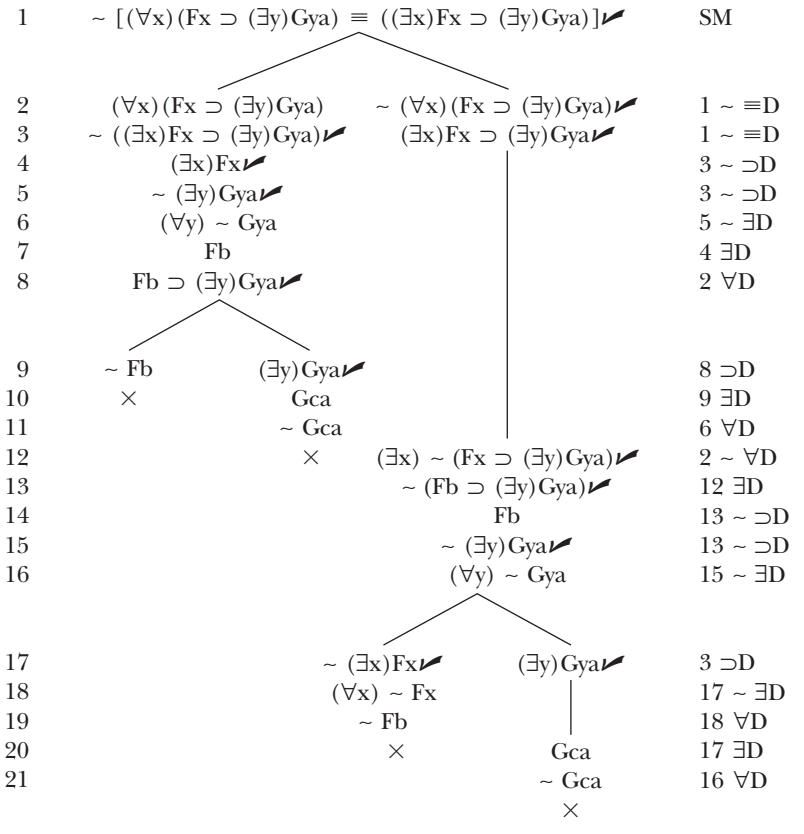
((∃x)Fx & (∃x)Gx)' is quantificationally indeterminate (and not quantificationally false), we must construct a tree for this biconditional:



It is surely not surprising that this tree has at least one completed open branch, establishing that the biconditional is not quantificationally false and is therefore quantificationally indeterminate, given the previous tree, which also had a completed open branch.

The sentences '(∀x)(Fx ⊃ (∃y)Gya)' and '(∃x)Fx ⊃ (∃y)Gya' are quantificationally equivalent, as the following closed tree for the negation of their

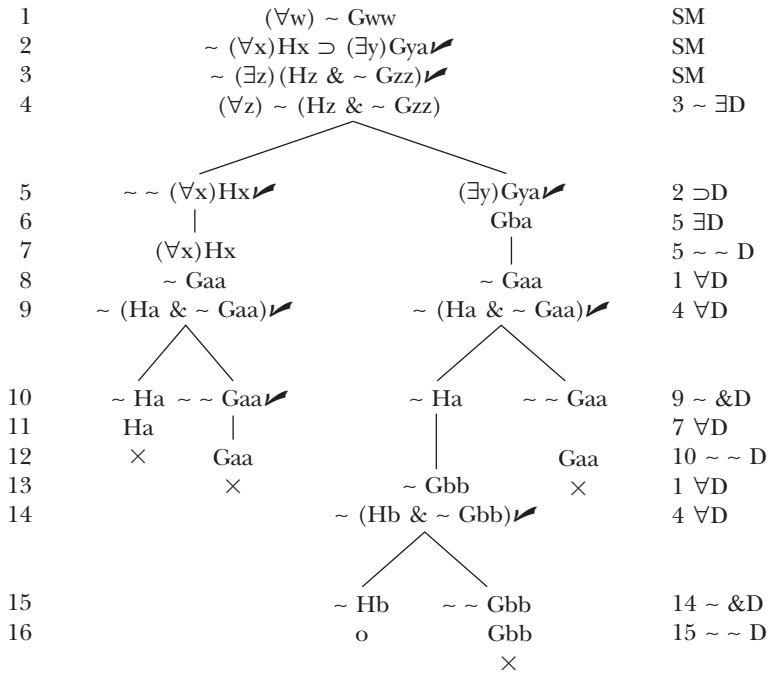
corresponding material biconditional establishes:



To use the tree method to test for quantificational validity, we construct a tree for the premises and the negation of the conclusion of the argument in question. A tree for the argument

$$\frac{\begin{array}{l}(\forall w) \sim Gww \\ \sim (\forall x)Hx \supset (\exists y)Gya\end{array}}{(\exists z)(Hz \ \& \sim Gzz)}$$

follows:



The tree has a completed open branch, so the argument is quantificationally invalid.

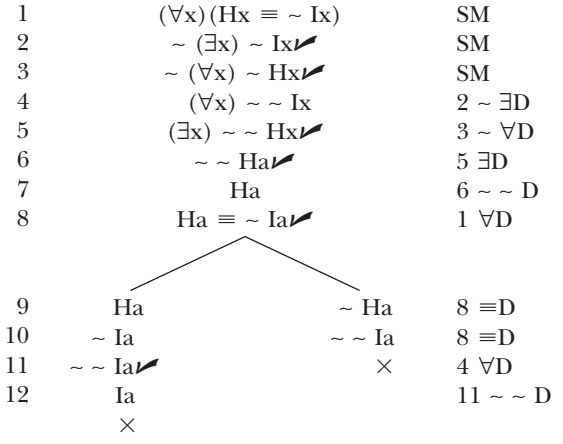
As with truth-trees for sentential logic, the procedure for testing alleged entailments parallels that for testing for validity. To test the entailment claim:

$$\{(\forall x)(Hx \equiv \sim Ix), \sim (\exists x) \sim Ix\} \vDash (\forall x) \sim Hx$$

we construct a truth-tree to determine whether the set

$$\{(\forall x)(Hx \equiv \sim Ix), \sim (\exists x) \sim Ix, \sim (\forall x) \sim Hx\}$$

is quantificationally consistent:



The tree is closed, and so we may conclude that the entailment does hold.

9.3**E** EXERCISES

Construct truth-trees as necessary to provide the requested information. In each case state your result, and specify what it is about your tree that establishes this result.

1. Which of the following sentences are quantificationally true?

a. $(\exists x)Fx \vee \sim (\exists x)Fx$
*b. $(\exists x)Fx \vee (\exists x) \sim Fx$
c. $(\forall x)Fx \vee (\forall x) \sim Fx$
*d. $(\forall x)Fx \vee \sim (\forall x)Fx$
e. $(\forall x)Fx \vee (\exists x) \sim Fx$
*f. $(\forall x)(Fx \vee Gx) \supset [(\exists x)Fx \vee (\exists x)Gx]$
g. $(\forall x)(Fx \vee Gx) \supset [(\exists x) \sim Fx \supset (\exists x)Gx]$
*h. $(\forall x)(Fx \vee Gx) \supset [(\exists x)Fx \vee (\forall x)Gx]$
i. $[(\forall x)Fx \vee (\forall x)Gx] \supset (\forall x)(Fx \vee Gx)$
*j. $(\forall x)(Fx \vee Gx) \supset [(\forall x)Fx \vee (\forall x)Gx]$
k. $(\exists x)(Fx \,\&\, Gx) \supset [(\exists x)Fx \,\&\, (\exists x)Gx]$
*l. $[(\exists x)Fx \,\&\, (\exists x)Gx] \supset (\exists x)(Fx \,\&\, Gx)$
m. $\sim (\exists x)Fx \vee (\forall x) \sim Fx$
*n. $(\forall x)[Fx \supset (Gx \,\&\, Hx)] \supset (\forall x)[(Fx \,\&\, Gx) \supset Hx]$
o. $(\forall x)[(Fx \,\&\, Gx) \supset Hx] \supset (\forall x)[Fx \supset (Gx \,\&\, Hx)]$

*p. $(\forall x)(Fx \mathbin{\&} \sim Gx) \vee (\exists x)(\sim Fx \vee Gx)$
q. $(\forall x)(Fx \supset Gx) \supset (\forall x)(Fx \supset (\forall y)Gy)$
*r. $(\forall x)(\forall y)Gxy \supset (\forall x)Gxx$
s. $(\forall x)Gxx \supset (\forall x)(\forall y)Gxy$
*t. $(\forall x)Fxx \supset (\forall x)(\exists y)Fxy$
u. $(\exists x)(\forall y)Gxy \supset (\forall x)(\exists y)Gyx$
*v. $(\exists x)(\exists y)(Lxy \equiv Lyx)$
w. $((\exists x)Lxx \supset (\forall y)Lyy) \supset (Laa \supset Lgg)$

2. Which of the following sentences are quantificationally false?

a. $(\forall x)Fx \mathbin{\&} (\exists x) \sim Fx$
*b. $(\forall x)Fx \mathbin{\&} \sim (\exists x)Fx$
c. $(\exists x)Fx \mathbin{\&} (\exists x) \sim Fx$
*d. $(\exists x)Fx \mathbin{\&} \sim (\forall x)Fx$
e. $(\forall x)(Fx \supset (\forall y) \sim Fy)$
*f. $(\forall x)(Fx \supset \sim Fx)$
g. $(\forall x)(Fx \equiv \sim Fx)$
*h. $(\exists x)Fx \supset (\forall x) \sim Fx$
i. $(\exists x)(\exists y)(Fxy \mathbin{\&} \sim Fyx)$
*j. $(\exists x)Fx \mathbin{\&} \sim (\exists y)Fy$
k. $(\forall x)(\forall y)(Fxy \supset \sim Fyx)$
*l. $(\forall x)(Gx \equiv \sim Fx) \mathbin{\&} \sim (\forall x) \sim (Gx \equiv Fx)$
m. $(\exists x)(\forall y)Gxy \mathbin{\&} \sim (\forall y)(\exists x)Gxy$

3. What is the quantificational status (quantificationally true, quantificationally false, or quantificationally indeterminate) of each of the following sentences?

a. $(\exists x)Fxx \supset (\exists x)(\exists y)Fxy$
*b. $(\exists x)(\exists y)Fxy \supset (\exists x)Fxx$
c. $(\exists x)(\forall y)Lxy \supset (\exists x)Lxx$
*d. $(\forall x)(Fx \supset (\exists y)Gyx) \supset ((\exists x)Fx \supset (\exists x)(\exists y)Gxy)$
e. $(\forall x)(Fx \supset (\exists y)Gya) \supset (Fb \supset (\exists y)Gya)$
*f. $((\exists x)Lxx \supset (\forall y)Lyy) \supset (Laa \supset Lgg)$
g. $(\forall x)(Fx \supset (\forall y)Gxy) \supset (\exists x)(Fx \supset \sim (\forall y)Gxy)$

4. Which of the following pairs of sentences are quantificationally equivalent?

a.	$(\forall x)Mxx$	$\sim (\exists x) \sim Mxx$
*b.	$(\exists x)(Fx \supset Ga)$	$(\exists x)Fx \supset Ga$
c.	$(\forall x)(Fa \supset Gx)$	$Fa \supset (\forall x)Gx$
*d.	$Ls \equiv (\forall x)Lx$	$(\exists x)Lx$
e.	$(\exists x)Fx \supset Ga$	$(\exists x)(Fx \supset Ga)$
*f.	$(\forall x)(Fx \vee Gx)$	$(\forall x)Fx \vee (\forall x)Gx$
g.	$(\forall x)Fx \supset Ga$	$(\exists x)(Fx \supset Ga)$
*h.	$(\exists x)(Ax \mathbin{\&} Bx)$	$(\exists x)Ax \mathbin{\&} (\exists x)Bx$
i.	$(\forall x)(\forall y)(Fx \supset Gy)$	$(\forall x)(Fx \supset (\forall y)Gy)$
*j.	$(\forall x)(Fx \equiv \sim Gx)$	$(\forall x) \sim (Fx \equiv Gx)$
k.	$(\forall x)(Fx \equiv Gx)$	$Fa \equiv (\forall x)Gx$
*l.	$(\forall x)(Fx \vee (\exists y)Gy)$	$(\forall x)(\exists y)(Fx \vee Gy)$
m.	$(\forall x)(Fx \supset (\forall y)Gy)$	$(\forall x)(\forall y)(Fx \supset Gy)$

5. Which of the following arguments are quantificationally valid?

a. $(\forall x)(Fx \supset Gx)$
Ga
$\overline{\quad\quad}$
Fa

*b. $(\forall x)(Tx \supset Lx)$
$\sim Lb$
$\overline{\quad\quad}$
$\sim Tb$

c. $(\forall x)(Kx \supset Lx)$
$(\forall x)(Lx \supset Mx)$
$\overline{\quad\quad}$
$(\forall x)(Kx \supset Mx)$

*d. $(\forall x)(Fx \supset Gx)$
$(\forall x)(Hx \supset Gx)$
$\overline{\quad\quad}$
$(\forall x)((Fx \vee Hx) \supset Gx)$

e. $(\forall x)(Fx \supset Gx) \supset (\exists x)Nx$
$(\forall x)(Nx \supset Gx)$
$\overline{\quad\quad}$
$(\forall x)(\sim Fx \vee Gx)$

*f. $(\sim (\exists y)Fy \supset (\exists y)Fy) \vee \sim Fa$
$\overline{\quad\quad}$
$(\exists z)Fz$

g. $(\forall x)(\sim Ax \supset Kx)$
$(\exists y) \sim Ky$
$\overline{\quad\quad}$
$(\exists w)(Aw \vee \sim Lwf)$

*h. $(\forall y)(Hy \,\&\, (Jyy \,\&\, My))$
$\overline{\quad\quad}$
$(\exists x)Jxb \,\&\, (\forall x)Mx$

i. $(\forall x)(\forall y)Cxy$
$\overline{\quad\quad}$
$(Caa \,\&\, Cab) \,\&\, (Cba \,\&\, Cbb)$

*j. $(\exists x)(Fx \,\&\, Gx)$
$(\exists x)(Fx \,\&\, Hx)$
$\overline{\quad\quad}$
$(\exists x)(Gx \,\&\, Hx)$

k. $(\forall x)(Fx \supset Gx)$
$\sim (\exists x)Fx$
$\overline{\quad\quad}$
$\sim (\exists x)Gx$

*l. $(\exists z)Bzz$
$(\forall x)(Sx \supset Bxx)$
$\overline{\quad\quad}$
$\sim Sg$

m. $(\exists x)Cx \supset Ch$
$\overline{\quad\quad}$
$(\exists x)Cx \equiv Ch$

*n. $Fa \vee (\exists y)Gya$
$Fb \vee (\exists y) \sim Gyb$
$\overline{\quad\quad}$
$(\exists y)Gya$

6. Which of the following alleged entailments hold?

a. $\{(\forall x) \sim Jx, (\exists y)(Hby \vee Ryy) \supset (\exists x)Jx\} \models (\forall y) \sim (Hby \vee Ryy)$
*b. $\{(\forall x)(\forall y)(Mxy \supset Nxy)\} \models (\forall x)(\forall y)(Mxy \supset (Nxy \,\&\, Nyx))$
c. $\{(\forall y)((Hy \,\&\, Fy) \supset Gy), (\forall z)Fz \,\&\, \sim (\forall x)Kxb\} \models (\forall x)(Hx \supset Gx)$
*d. $\{(\forall x)(Fx \supset Gx), (\forall x)(Hx \supset Gx)\} \models (\forall x)(Fx \vee Hx)$
e. $\{(\forall z)(Lz \equiv Hz), (\forall x) \sim (Hx \vee \sim Bx)\} \models \sim Lb$

9.4 FINE-TUNING THE TREE METHOD FOR *PL*

In Chapter 8 we noted that there is no decision procedure for deciding, for each sentence of *PL,* whether that sentence is quantificationally true, quantificationally false, or quantificationally indeterminate. That is, there is no mechanical test procedure that always yields, in a finite number of steps, a "yes" or "no" answer to the question 'Is this sentence of *PL* quantificationally true, and if not is it quantificationally false, and if not is it quantificationally indeterminate?'

Nor is there such a decision procedure for equivalence, consistency, validity, or entailment: the system *PL* is *undecidable*. In the current context this means that we cannot produce a mechanical method for constructing trees that will always provide correct "yes" or "no" answers in a finite number of steps. The problem here is that not every finite set of sentences of *PL* has a finite truth-tree, where we define a **finite truth-tree** to be a truth-tree that either is closed or has a completed open branch. It is an unavoidable result that not every finite set of sentences of *PL/PLE* has a finite tree. There are, however, two ways in which the tree method we have developed for *PL* can be significantly improved, and our task in this section is to do so.

First, we would like our set of rules to be capable of producing a finite tree for any finite set that has a **finite model,** that is, any set for which there is an interpretation with a finite UD on which all of the members of the set are true. Our present tree rules do not ensure that there is a finite tree for every such set. Consider, for example, the start of a tree for {(∀y)(∃z)Fyz}:

1	(∀y)(∃z)Fyz	SM
2	(∃z)Faz✓	1 ∀D
3	Fab	2 ∃D
4	(∃z)Fbz✓	1 ∀D
5	Fbc	4 ∃D
6	(∃z)Fcz✓	1 ∀D
7	Fcd	6 ∃D
	•	
	•	
	•	

The dots here indicate that the tree will continue indefinitely. There is no hope of closing the one open branch on this tree. At every other step after the first, a new atomic sentence is added to the open branch, using Existential Decomposition, and since every atomic sentence is quantificationally consistent with every other atomic sentence, continuing to add more atomic sentences will never close the tree. But this branch also can never become a *completed* open branch. Every time Universal Decomposition is applied to the sentence on line 1, a new existentially quantified sentence is added to the branch. And decomposing that sentence adds a new individual constant to the branch, necessitating a further application of Universal Decomposition to '(∀y)(∃z)Fyz', resuming the cycle. We call an open branch that cannot be completed—one that never closes and will never, in a finite number of steps, become a completed open branch—a ***nonterminating branch***.

Assuming we retain the requirement that every universally quantified sentence on a completed open branch must be decomposed using every constant on that branch, the only way to avoid the inevitability of a nonterminating branch in the preceding tree is to revise our Existential Decomposition rule. That rule currently stipulates that a sentence (∃**x**)**P** must be decomposed to a substitution instance **P**(**a**/**x**) in which **a** is *foreign* to the branch in question. Let

us recall the reason for this restriction: it is that using a constant that *already* occurs on the branch would be to make the unwarranted assumption that the thing that is of the sort **P** is also of the sort specified by the formulas in which it occurs elsewhere on the branch. The following tree illustrates this:

1	(∃x)Fx✓	SM	
2	(∃x) ~ Fx✓	SM	
3	Fa	1 ∃D	
4	~ Fa	2 ∃D	**MISTAKE!**
	×		

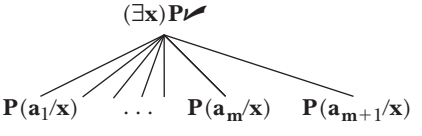
There is no interpretation on which something is F and that very same thing is not F, but in using 'b' at line 4 as well as line 3 we are, in effect, looking for such an interpretation. Yet the set {(∃x)Fx, (∃x) ~ Fx} is consistent, as the following correct tree verifies:

1	(∃x)Fx✓	SM
2	(∃x) ~ Fx✓	SM
3	Fa	1 ∃D
4	~ Fb	2 ∃D
	o	

However, as the example in the previous paragraph shows, the requirement that a new constant be used to instantiate an existentially quantified sentence sometimes leads to nonterminating branches.

Fortunately, there is another way to think about decomposing an existentially quantified sentence (∃**x**)**P**. Rather than avoid altogether the use of constants that already occur on the branch that contains (∃**x**)**P**, we shall introduce a second, *branching,* Existential Decomposition rule. The branching will allow us to consider substitution instances formed from constants already on the branch as well as a substitution instance formed from a new constant. We will call the new rule '**Existential Decomposition-2**':

Existential Decomposition-2 (∃D2)

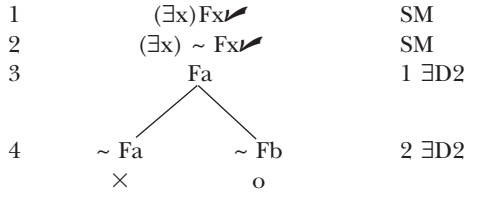


where $\mathbf{a_1}$, . . . , $\mathbf{a_m}$ are the constants that already occur on the branch on which Existential Decomposition-2 is being applied and $\mathbf{a_{m+1}}$ is a constant that is foreign to that branch.[5]

[5]This Existential Decomposition rule is due to George Boolos, "Trees and Finite Satisfiability: Proof of a Conjecture of Burgess," *Notre Dame Journal of Formal Logic,* 25(3)(1984), 193–197.

This rule requires that when we decompose an existentially quantified sentence $(\exists \mathbf{x})\mathbf{P}$ we must branch out to the relevant substitution instances. If $\mathbf{a_1}$ through $\mathbf{a_m}$ are the constants that occur on the branch that contains the sentence $(\exists \mathbf{x})\mathbf{P}$ that is being decomposed, then substitution instances formed from those constants are to be entered, each on a distinct branch, and $\mathbf{P}(\mathbf{a_{m+1}}/\mathbf{x})$, where $\mathbf{a_{m+1}}$ is any constant foreign to the branch in question, is to be added on a further branch. Thus Existential Decomposition-2 produces a varying number of new branches, depending on how many constants already occur on the branch to which it is applied.

Here is a tree for the set {(∃x)Fx, (∃x) ~ Fx} in which Existential Decomposition-2 is used:

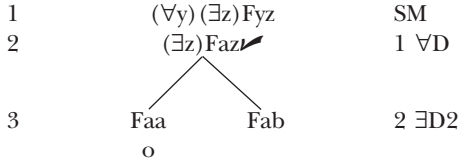


When we first used Existential Decomposition-2, on line 3, there were no constants already occurring on the open branch containing '(∃x)Fx'. In this case the rule requires adding only one substitution instance, formed from any constant. That is, if no constants previously occur on a branch, Existential Decomposition and Existential Decomposition-2 produce the same result. The second use of Existential Decompostion-2 required branching to two substitution instances of '(∃x) ~ Fx'. On the left branch we formed a substitution instance using the constant 'a' that already occurred on the single branch, and on the right branch we formed a substitution instance using a new constant, 'b'. The idea behind branching to these two possibilities is that the individual by virtue of which '(∃x) ~ Fx' is true might be the individual that we have already chosen to designate with the constant 'a', or it might be another individual. To allow for the latter possibility we form a substitution instance with a new constant. The left-hand branch closes because the individual by virtue of which '(∃x) ~ Fx' is true cannot be the individual denoted by 'a'; there is no interpretation on which 'Fa' and '~ Fa' are both true. But the open right-hand branch is complete. This branch contains the literals 'Fa' and 'Fb' and shows that there is an interpretation on which 'Fa' and '~ Fb', and consequently both '(∃x)Fx' and '(∃x) ~ Fx', are true. Any interpretation that includes the following assignments will do:

UD: the set {1, 2}
a: 1
b: 2
F: {<1>}

So far it may look like Existential Decomposition-2 makes for more work than is necessary, since using the earlier rule Existential Decomposition

resulted in a completed open four-line tree for the set {(∃x)Fx, (∃x) ~ Fx} without any branching. But it allows us to produce closed truth-trees in cases where Existential Decomposition cannot produce closed trees. Consider again the set {(∀y)(∃z)Fyz}. We saw that with the rule Existential Decomposition we could only produce a tree with a nonterminating branch for this set, even though the set is quantificationally consistent and has a finite model. Using Existential Decomposition-2 we can produce a truth-tree with a completed open branch:



At line 3 we branched to two substitution instances of the existentially quantified sentence on line 2. The instantiating constant in the substitution instance on the left-hand branch is 'a', which occurred earlier on that branch. We chose 'b' as the instantiating constant in the substitution instance on the right-hand branch because it is foreign to the branch so far. The sentence on line 2 says that the individual designated by 'a' bears the relation F to something. The branching indicates that that something might be the very same individual (this is the left-hand branch) or it might be a different individual (this is the right-hand branch). If both branches close, we will know neither is the case. But if one of these branches becomes a completed open branch, we will know that there is a model—indeed, a finite model—for the set being tested. Here the left-hand branch is completed and contains the single literal 'Faa'. So there is a finite model for the set being tested. Any interpretation that includes the following assignments will be such a model:

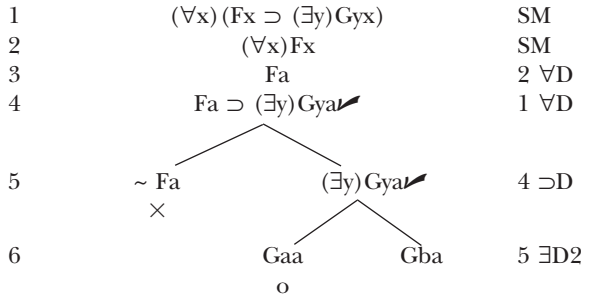
UD: {1}
F: {<1, 1>}

The right-hand branch is open but is *not* a completed open branch because the universally quantified sentence on line 1 has not been decomposed to a substitution instance formed from 'b'.

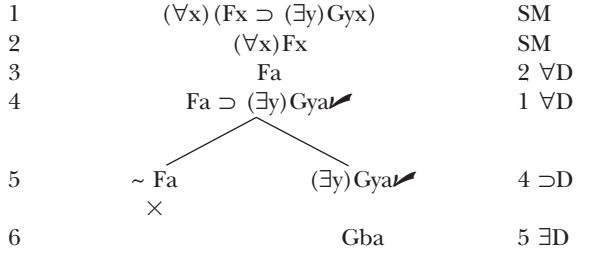
Note that if we were to continue the right-hand branch by instantiating the universal quantification on line 1 with 'b' we would need to branch to three substitution instances when decomposing the existentially quantified sentence '(∃z)Fbz', namely the substitution instances 'Fba' and 'Fbb' (because 'a' and 'b' already occur on that branch), and a substitution instance with a new constant such as 'Fbc'. The first two of these branches will also be completed open branches, while the third will require us to once again decompose the universal quantification on line 1 using the new constant 'c'. It should be

clear that we will never be able to complete the tree. But we don't need to, because the new rule Existential-Decomposition-2 produced a completed open branch at line 3, showing that the set is quantificationally consistent. Using Existential Decomposition-2 rather than Existential Decomposition ensures that the interplay between universal and existential quantifiers will not produce trees with only nonterminating open branches for sets that do have finite models. In fact, we shall prove in Chapter 11 that *using the rule ∃D2 in place of ∃D, every finite set of sentences of PL with a finite model will have a finite tree with a completed open branch.*

Here is a tree using Existential Decomposition-2 for the set {(∀x)(Fx ⊃ (∃y)Gyx), (∀x)Fx}:

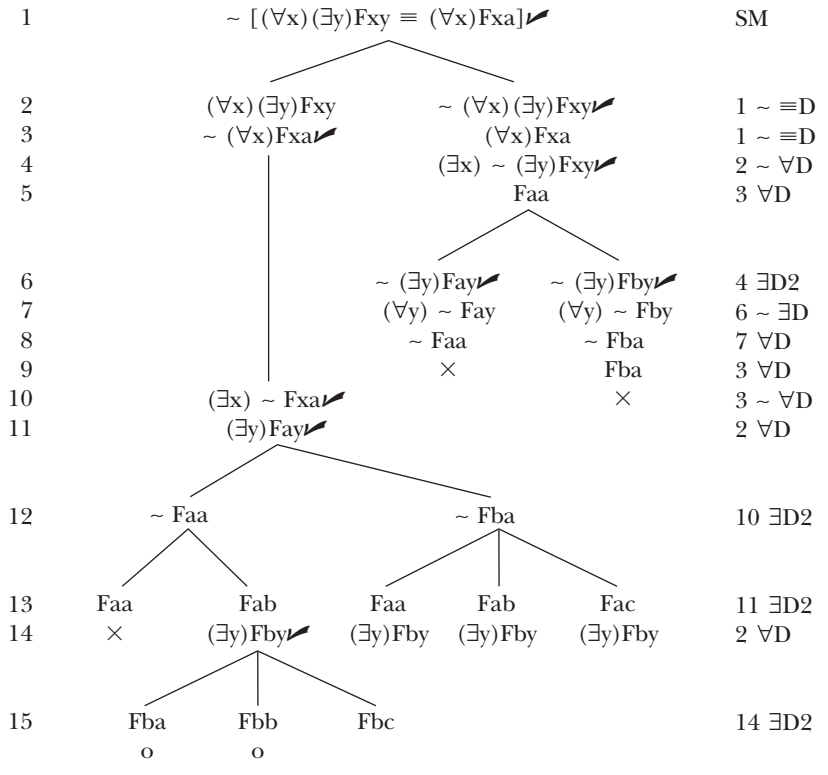


This tree has one completed open branch. The two universally quantified sentences on the branch have been decomposed to the single constant 'a' that occurs on the branch. Every other sentence on the branch is either a literal or has been decomposed, so we may conclude that the set is quantificationally consistent. Had we used Existential Decomposition at line 6, the tree would have had only two branches at that point:



Of course, the left branch still closes, but repeated uses of ∃D on the right branch will only go on to produce more closed branches and a single nonterminating branch.

The following tree shows that the set {~ [(∀x)(∃y)Fxy ≡ (∀x)Fxa]} is quantificationally consistent:



Note that using Existential Decomposition-2 at line 13 resulted in adding two new branches to the existing leftmost branch (one with the constant 'a' that already occurred on the leftmost branch and one with the constant 'b' that was foreign to that branch), and adding three new branches to the other existing open branch ending at line 12 (two with the constants 'a' and 'b' that already occurred on that branch and one with the constant 'c' that was foreign to that branch). At line 15, Existential Decomposition-2 produced three branches from the leftmost open branch of line 14: two substitution instances use the constants 'a' and 'b' that already occur there, and a third uses the constant

'c' that was foreign to that branch. Although the tree is not complete, it has two completed open branches, the one ending in 'Fba' and the one ending in 'Fbb'. The branch ending in 'Fbc' is not complete, as '$(\forall x)(\exists y)Fxy$' has not been decomposed using the constant 'c'. The branches to line 14 that end in undecomposed existentially quantified sentences are also, for that reason, not complete.

We turn now to the second improvement in our tree method for *PL*. We would like to be assured that when a set does have a finite tree we will eventually find it. The tree rules we have presented do not themselves guarantee this. For example, they allow the construction of trees such as the following one for the set $\{(\forall x)Fx, (\forall x) \sim Fx\}$:

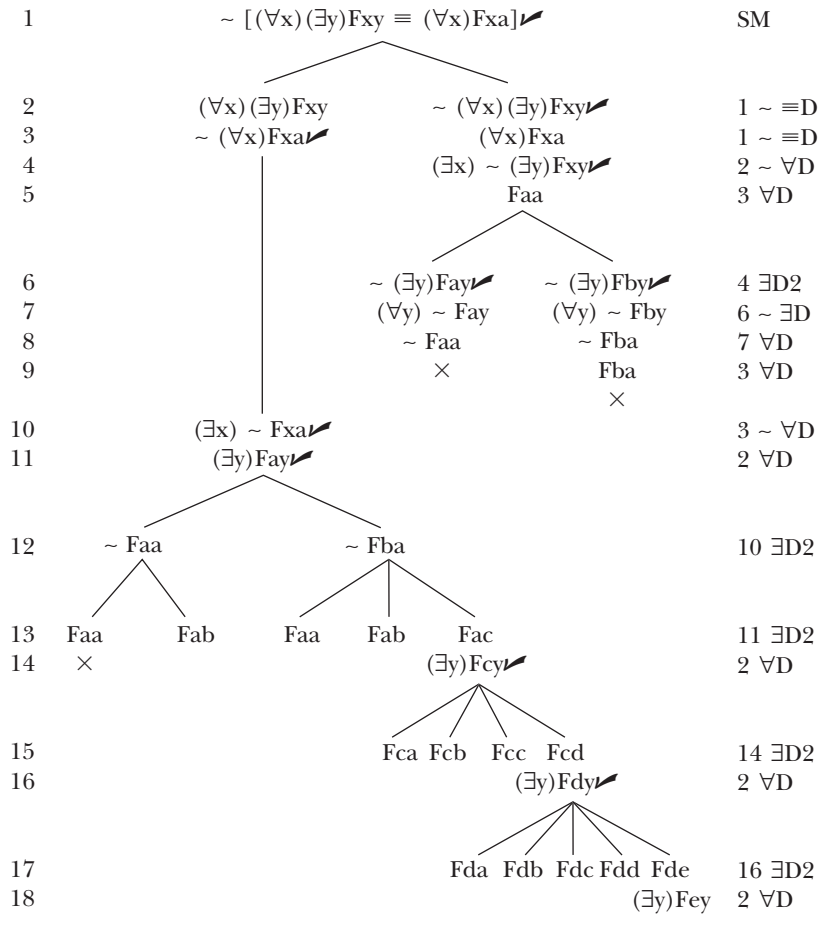
1	$(\forall x)Fx$	SM
2	$(\forall x) \sim Fx$	SM
3	Fa	1 $\forall$D
4	Fb	1 $\forall$D
5	Fc	1 $\forall$D
6	Fd	1 $\forall$D
	•	
	•	
	•	

Continuing in this way—adding substitution instances that result from applying $\forall$D to the sentence on line 1—does not involve misusing any tree rule but will never produce either a closed tree or a completed open branch. Yet a closed tree for the set can be produced in just four lines:

1	$(\forall x)Fx$	SM
2	$(\forall x) \sim Fx$	SM
3	Fa	1 $\forall$D
4	$\sim Fa$	2 $\forall$D
	×	

What we need is a procedure for applying the decomposition rules that is guaranteed to yield a finite tree where one exists, not only in this case but also in much more complicated ones. For example, earlier we produced a tree with a completed open branch for the set $\{\sim [(\forall x)(\exists y)Fxy \equiv (\forall x)Fxa]\}$, showing that

it is quantificationally consistent. But we could just as well have begun the tree for this set as follows:



In this case, after line 13 we have repeatedly added a substitution instance of the universal quantification on line 2 to the rightmost open branch, using the new constant just introduced by ∃D2, then applied ∃D2 again, generating a new instantiating constant on the rightmost branch, and so on. Clearly we can

continue this process forever. So unless care is taken we can work continuously on a branch that won't terminate while ignoring a branch that, if continued, will become a completed open branch.

To guarantee that a finite branch will be found if it exists, we introduce a procedure for constructing trees for *PL* that pursues all possibilities in a systematic fashion, such that a completed open branch will be found if one exists and also such that a tree that can be closed will in fact close:

The System for PL

List the members of the set to be tested.

Exit Conditions: Stop if

a. the tree closes, or
b. an open branch becomes a completed open branch.

Construction Procedures:

Stage 1: Decompose all truth-functionally compound and existentially quantified sentences and each resulting sentence that is itself either a truth-functional compound or an existentially quantified sentence.

Stage 2: For each universally quantified sentence $(\forall \mathbf{x})\mathbf{P}$ on the tree, enter $\mathbf{P}(\mathbf{a}/\mathbf{x})$ on every open branch passing through $(\forall \mathbf{x})\mathbf{P}$ for every constant **a** on the branch. On each open branch passing through $(\forall \mathbf{x})\mathbf{P}$ on which no constant occurs, enter $\mathbf{P}(\mathrm{a}/\mathbf{x})$.

Repeat this process until every universally quantified sentence on the tree, including those added as a result of this process, has been so decomposed.

Return to Stage 1.

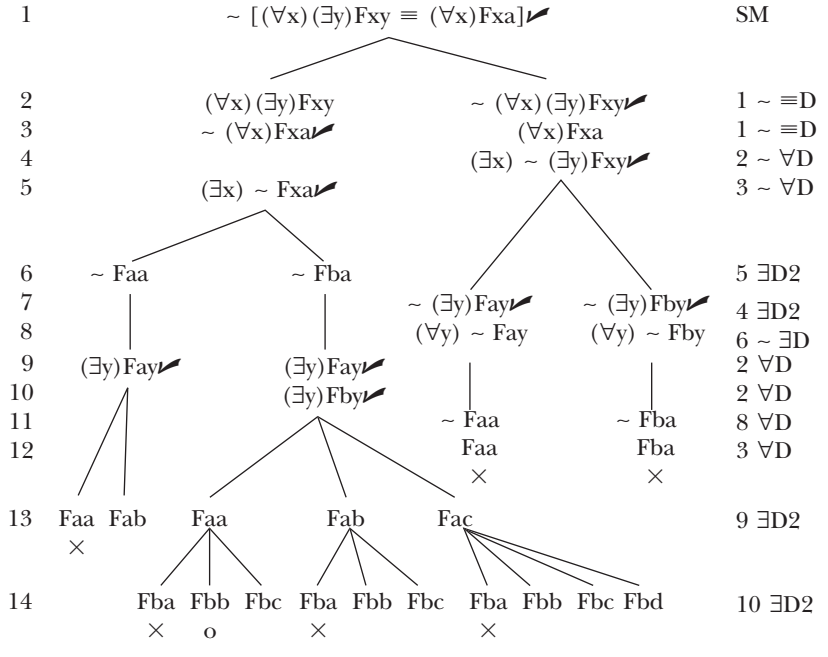
We call trees that have been constructed in accordance with the System '**systematic trees**'. In all systematic trees, Existential Decomposition-2 is used rather than Existential Decomposition. To construct a systematic tree, proceed through the tree construction stages in the order specified, being sure to exhaust the work that must be done at each stage before proceeding to the next stage. Stage 1 is complete if the only sentences on the tree that are not checked off are either literals or universally quantified sentences. Stage 2 is complete only when every universally quantified sentence on the tree has been decomposed in the required manner, including both those that were on the tree when we passed from Stage 1 to Stage 2 and those that are entered as a result of work at Stage 2. The first tree that we began to construct for the set $\{(\forall x)Fx, (\forall x) \sim Fx\}$ was not a systematic tree because it violated Stage 2. In that tree the universally quantified sentence on line 1 was decomposed at line 4 with the new constant 'b', violating

the requirement that all universally quantified sentences must be decomposed using constants already on the branch, and only those (except when there is no constant on a branch; in this case a substitution instance with the constant 'a' must be added.). However, following The System produces a closed tree for this set:

1	$(\forall x)Fx$	SM
2	$(\forall x) \sim Fx$	SM
3	Fa	1 $\forall$D
4	$\sim Fa$	2 $\forall$D
	$\times$	

At line 4 we decomposed the universally quantified sentence from line 2 with the constant 'a' that already occurred on the branch. Doing so not only completed Stage 2 but also terminated construction because an Exit Condition was met: the tree closed.

Although our first tree for the set $\{\sim [(\forall x)(\exists y)Fxy \equiv (\forall x)Fxa]\}$ included a completed open branch, that tree is not a systematic tree. It first violates the method set out in The System at line 5, where it decomposes a universally quantified sentence before all truth-functionally compound and existentially quantified sentences have been decomposed. These latter include the negated universally quantified sentence on line 3 and the existentially quantified sentence on line 4. Here is a *systematic* tree for the same set:

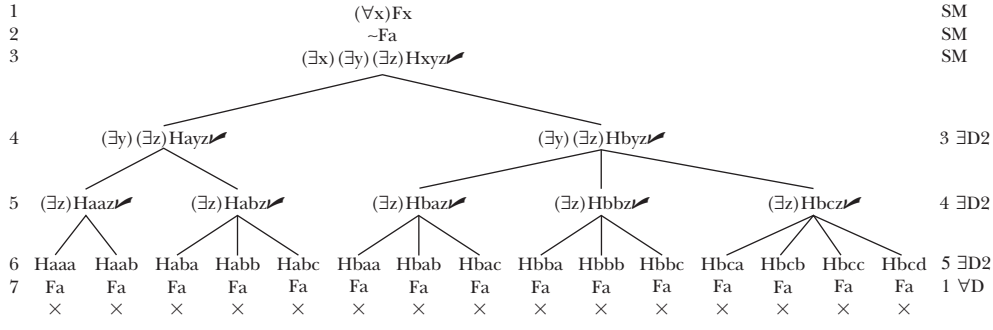


At line 8 all of the truth-functionally compound and existentially quantified sentences occurring on lines 1–8 have been decomposed, so Stage 1 has been

completed and Stage 2 commences. Stage 2 continues through line 12, at which point all universally quantified sentences have been decomposed using all of the relevant constants, so we return to Stage 1 to continue at line 13. Stage 1 produces a completed open branch on line 14, so the procedure terminates. Note that the second branch from the left at line 13, on which 'Fab' occurs, is not a completed open branch. The universally quantified sentence occurring on the left-hand branch at line 2 has not been decomposed to 'b'. Interestingly, this tree is one line shorter than our earlier tree with a completed open branch for the same set. Unfortunately, as we shall shortly see, systematic trees are not always so economical.

Systematic trees differ from nonsystematic trees in three important respects. First, in systematic trees Existential Decomposition-2 is always used to decompose existentially quantified sentences. In nonsystematic trees either Existential Decomposition or Existential Decomposition-2 (or both) may be used. Second, The System does not allow work on one branch to be continued to the point of excluding all work on another open branch. Third, it does not allow us to ignore constants already occurring on a tree when we apply Universal Decomposition in Stage 2, so that if a substitution instance using such a constant can close a branch, we will be sure that the branch *does* close.

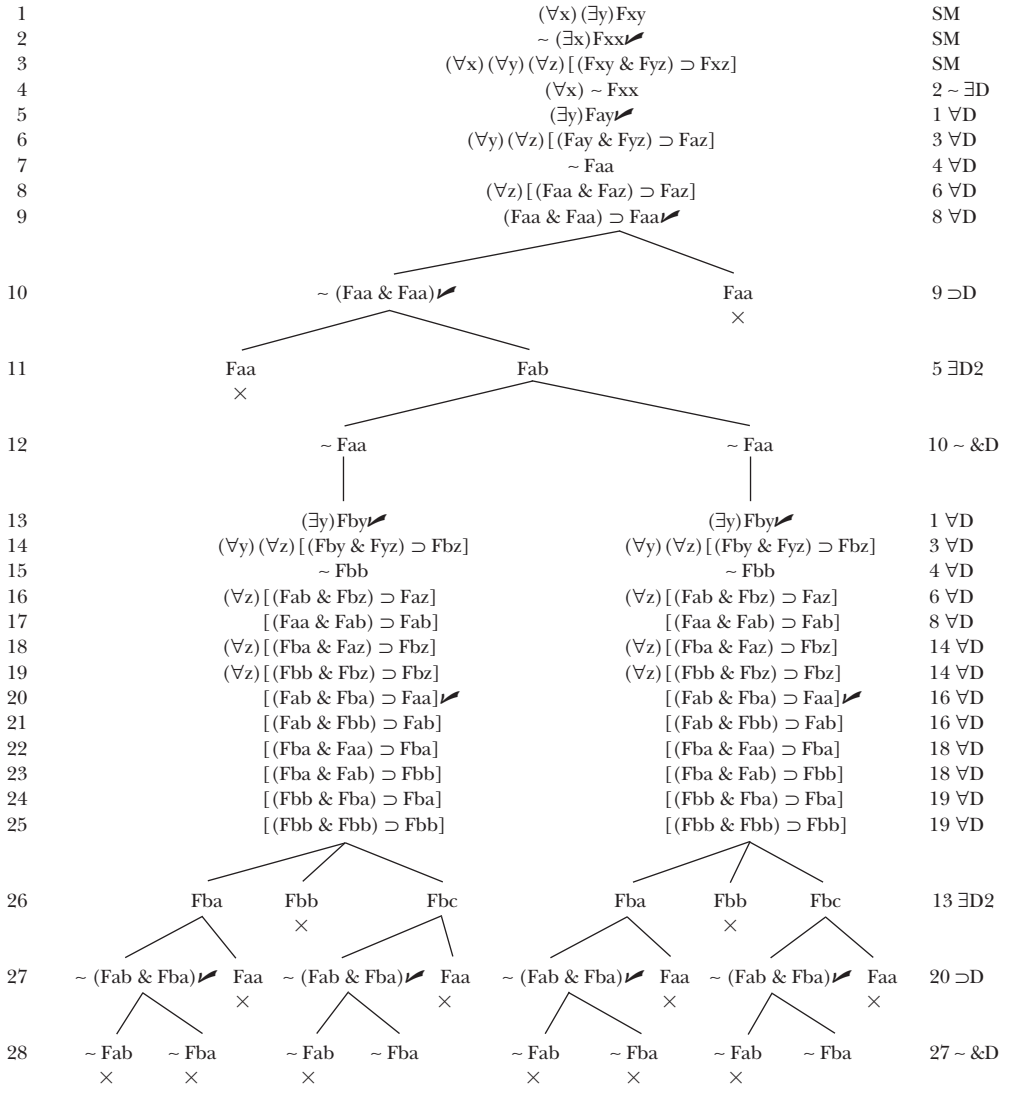
The advantage of constructing systematic trees is that doing so will always lead, in a finite number of steps, to a completed open branch when one exists, and, also in a finite number of steps, to a closed tree when one exists. The disadvantage is that systematic trees can frequently be much larger than are nonsystematic trees. For example, here is a systematic tree for the set $\{(\forall x)Fx, \sim Fa, (\exists x)(\exists y)(\exists z)Hxyz\}$:



Obviously, we could have produced a closed tree in four lines, by entering 'Fa' (obtained by decomposing the sentence on line 1) at line 4. But the result would not be a systematic tree. The System requires us first to decompose the sentence on line 3, then those on line 4, and then those on line 5, before decomposing the universally quantified sentence from line 1. So we do not claim that The System always produces the smallest trees possible, and wherever we see a more economical way to produce either a closed tree or a completed open branch we should do so. What The System *does* guarantee is that if there is a tree with a completed open branch, The System will generate such a tree, and if there is a closed tree, The System will generate a closed tree. Because The System is

reliable in this sense, it should be used when one does not see how to close a branch or produce a completed open branch without using The System.

It is important to bear in mind that The System will not always result either in a closed tree or in a tree with a completed open branch. Consider, for example, the set {(∀x)(∃y)Fxy, ~ (∃x)Fxx, (∀x)(∀y)(∀z)[(Fxy & Fyz ⊃ Fxz]}. This set is consistent, but every model for the set (every interpretation on which all of the set members are true) has an infinite UD. The System will always produce a completed open branch for a finite set that has a finite model, but it will not do so in the case of consistent sets that have only infinite models—for the very good reason that there are no trees with completed open branches for such sets. Here is the start of a systematic tree for this set:



After listing the set members, we move to Stage 1.

- Once '~ (∃x)Fxx' has been decomposed to '(∀x) ~ Fxx', every truth-functional compound (the sentence on line 2) and every existentially quantified sentence on the tree (there are none) has been decomposed.
- Proceeding to Stage 2 we decompose each universally quantified sentence on the tree to a substitution instance formed from 'a' taking us through line 9.
- Returning to Stage 1, there are two sentences to be decomposed: the existentially quantified sentence on line 5 and the truth-functional compound on line 9. We choose to decompose the latter first, as it yields one closed branch (the right branch), at line 10. Next we decompose the existentially quantified sentence from line 5. This yields one closed branch and one open branch. At this point we still have one undecomposed truth-functional compound, on line 10. Decomposing this sentence yields two open branches. As it happens these are identical—exactly the same sentences occur on each branch—but The System requires us to pursue both branches.
- Now we proceed to Stage 2, adding lines 13–25: although all the universally quantified sentences on the tree have already been decomposed to substitution instances formed from 'a', they must now also all be decomposed to substitution instances formed from 'b' since 'Fab' occurs on both branches.
- Back to Stage 1 at line 26, we decompose the existentially quantified sentences from line 13 first, splitting each of the existing two branches into three branches. Two of these branches close. At line 27 we branch again when we decompose the material conditional occurring on line 20. Four of the eight resulting branches close. Decomposing '~ (Fab & Fba)', which occurs four times on line 27, results in eight branches, six of which close.

Were we to continue, we would next decompose the remaining truth-functional compounds. Some, but not all branches, would close. Eventually we would return to Stage 1 and decompose all universally quantified sentences to substitution instances formed from 'c', since that now appears on the open branches (at line 26). This would produce a new existentially quantified sentence that would eventually be decomposed to substitution instances formed from 'a', 'b', 'c', and 'd', respectively. This cycle will repeat forever, always closing all branches except for the one that includes a new instantiating constant as a result of applying ∃D2.

We have not *proved* that this tree will never close and will never have a completed open branch, but this is the case (the only way to demonstrate this is to show, independently of the tree method, that our set is quantificationally consistent, that it has only infinite models, and that no set with only infinite

models has a finite tree). Here the point is that the tree method cannot be used to show that sets such as this one are quantificationally consistent. We abandon the tree; we do not complete it. However, having used The System, we can be sure that we have not, as far as we have gone, missed a completed open branch or a chance to close the tree.

While instructions for identifying (without fail) a systematic tree that is caught in an endless cycle of decompositions and is such that it has only nonterminating branches would be desirable, there can be no such instructions because there is no decision procedure for quantificational consistency. We can only say that, if one has cycled through the stages of The System several times and there appears to be a pattern that will continually keep at least one branch open, one should consider the possibility that the set has only infinite models and consider abandoning the tree. Abandoning a tree constitutes a failure to find an answer to the question being asked. Having abandoned a tree, we can of course try to directly establish the consistency of the set in question by trying to find an interpretation on which all the members of the set are true.

9.4**E** EXERCISES

1. Construct systematic trees to determine, for each of the following sets, whether that set is quantificationally consistent. State your result. If you abandon a tree, explain why.

a. $\{(\forall x)Jx, (\forall x)(Jx \equiv (\exists y)(Gyx \vee Ky))\}$
*b. $\{(\forall x)(Fx \supset Cx), \sim (\forall x)(Fx \,\&\, Cx)\}$
c. $\{(\exists x)Fx, (\exists x) \sim Fx\}$
*d. $\{\sim (\forall x) \sim Hx, (\forall x)(Hx \supset Kx), \sim (\exists x)(Kx \,\&\, Hx)\}$
e. $\{(\exists x)Fx \,\&\, (\exists x) \sim Fx, (\exists x)Fx \supset (\forall x) \sim Fx\}$
*f. $\{(\exists x)Fx \,\&\, (\exists x) \sim Fx, (\forall x)Fx \supset (\forall x) \sim Fx\}$
g. $\{(\forall x)(\exists y)Fxy, (\exists y)(\forall x) \sim Fyx\}$
*h. $\{(\forall x)(\sim Gx \supset Fx), (\exists x)(Fx \,\&\, \sim Gx), Fa \supset \sim Ga\}$
i. $\{(\exists x)Hx, \sim (\forall x)Hx, (\forall x)(Hx \supset Kx), (\exists x)(Kx \,\&\, Hx)\}$
*j. $\{(\exists x)(\forall y)Lxy, (\exists x)(\forall y) \sim Lxy\}$
k. $\{(\forall x)(\exists y)Lxy, (\forall x)(\exists y) \sim Lxy\}$
*l. $\{(\forall x) \sim (\exists y)Lxy, (\forall w)(\forall y)(Swy \vee \sim Lwy), \sim (\exists x) \sim (\exists z)Sxz\}$
m. $\{(\forall x)(\exists y)Fxy, (\exists x)(\exists y) \sim Fxy\}$
*n. $\{(\forall x)(\forall y)(\forall z)((Hxy \,\&\, Hyz) \supset Hxz), (\forall x)(\forall y)(Hxy \supset Hyx), (\exists x) \sim Hxx\}$
o. $\{\sim (\forall x)(Kx \supset (\forall y)(Ky \vee Lxy)), (\forall y)(Ky \supset (\forall x)(Rx \supset Lyx)), (\forall x)Rx\}$

2. Construct systematic trees to determine, for each of the following sentences, whether that sentence is quantificationally true, quantificationally false, or quantificationally indeterminate. In each case state your result. If you abandon a tree, explain why.

a. $(\forall x)(Fax \supset (\exists y)Fya)$
*b. $(\exists x) \sim Fx \supset (Fa \supset \sim Fb)$
c. $(\forall x)[Fx \supset (\forall y)(Hy \supset Fy)]$
*d. $(\exists y)(\forall x)Fxy \supset (\forall x)(\exists y)Fxy$

e. $(\exists x)(Fx \vee \sim Fx) \equiv ((\exists x)Fx \vee (\exists x) \sim Fx)$
*f. $(\forall x)(Fx \equiv [(\exists y)Gyx \supset H]) \supset (\forall x)[Fx \supset (\exists y)(Gyx \supset H)]$
g. $(\forall x)(Fx \supset [(\exists y)Gyx \supset H]) \supset (\forall x)[Fx \supset (\exists y)(Gyx \supset H)]$

3. Construct systematic trees to determine which of the following arguments are quantificationally valid. In each case state your result. If you abandon a tree, explain why.

a. Fa
$(\forall x)(Fx \supset Cx)$
$(\forall x)(Fx \ \& \ Cx)$

*b. $(\forall x)(Jx \vee Ixb) \vee (\forall x)(\exists y)(Hxy \supset Mx)$
Iab

c. Fa
$(\forall x)(Fx \supset Cx)$
$(\exists x)(Fx \ \& \ Cx)$

*d. $\sim (\forall y)Kyy \vee (\forall x)Hxx$
$(\exists x)(\sim Hxx \supset \sim Kxx)$

e. $(\forall x)(\forall y)(\forall z)[(Lxy \ \& \ Lyz) \supset Lxz]$
$(\forall x)(\forall y)(Lxy \supset Lyx)$
$(\forall x)Lxx$

*f. $(\forall x)(\forall y)(Fx \vee Gxy)$
$(\exists x)Fx$
$(\exists x)(\exists y)Gxy$

g. $(\exists x)[(Lx \vee Sx) \vee Kx]$
$(\forall y) \sim (Ly \vee Ky)$
$(\exists x)Sx$

*h. $(\exists x)((Lx \vee Sx) \vee Kx)$
$(\forall y) \sim (Ly \vee Ky)$
$(\forall x)Sx$

i. $(\forall x)(Hx \supset Kcx)$
$(\forall x)(Lx \supset \sim Kcx)$
Ld
$(\exists y) \sim Hy$

4. Construct systematic trees to determine which of the following pairs of sentences are quantificationally equivalent. In each case state your result. If you abandon a tree, explain why.

a.	$(\forall x)(\forall y) \sim Sxy$	$\sim (\exists x)(\exists y)Sxy$
*b.	$(\forall x)(\exists y)Lxy$	$(\exists y)(\forall x)Lyx$
c.	$(\exists x)(Ax \supset B)$	$(\forall x)Ax \supset B$
*d.	$(\forall x)(Ax \supset B)$	$(\forall x)Ax \supset B$
e.	$(\forall x)(Ax \supset B)$	$(\exists x)Ax \supset B$
*f.	$(\exists x)(Ax \supset B)$	$(\exists x)Ax \supset B$
g.	$(\exists x)(\exists y)Hxy$	$(\exists y)(\exists x)Hxy$

5. Construct systematic trees to determine which of the following alleged entailments hold. In each case state your result. If you abandon a tree, explain why.

a. $\{(\forall x)(Fax \supset Fxa)\} \vDash Fab \vee Fba$
*b. $\{(\forall x)(\forall y)(Fx \vee Gxy), (\exists x)Fx\} \vDash (\exists x)(\exists y)Gxy$
c. $\{\sim Fa, (\forall x)(Fa \supset (\exists y)Gxy)\} \vDash \sim (\exists y)Gay$
*d. $\{(\exists x)(\forall y)Gxy\} \vDash (\forall y)(\exists x)Gxy$

e. $\{(\exists x)Gx, (\forall x)(Gx \supset Dxx)\} \vDash (\exists x)(Gx \,\&\, (\forall y)Dxy)$
*f. $\{(\forall y)(\exists x)Gxy\} \vDash (\exists x)(\forall y)Gxy$

***6.** Show that if the members of a set Γ of sentences of *PL* contain only '~' and universal and existential quantifiers as logical operators, then Γ has no tree with more than one branch if the rule $\exists$D is used but may have a tree with more than one branch if $\exists$D2 is used.

7. Show that no closed truth-tree can have an infinite branch.

***8.** Could we replace Universal Decomposition and Existential Decomposition with the following two rules? Explain.

$(\forall \mathbf{x})\mathbf{P}\checkmark$ $\quad$ $(\exists \mathbf{x})\mathbf{P}\checkmark$
$\sim(\exists \mathbf{x}) \sim \mathbf{P}$ $\quad$ $\sim(\forall \mathbf{x}) \sim \mathbf{P}$

9. Let $\mathbf{P}(\mathbf{a}/\mathbf{x})$ be a substitution instance of some sentence $(\exists \mathbf{x})\mathbf{P}$ such that $\{\mathbf{P}(\mathbf{a}/\mathbf{x})\}$ has a closed tree. Does it follow that $\{(\exists \mathbf{x})\mathbf{P}\}$ has a closed tree? Explain.

***10.** Let $(\forall \mathbf{x})\mathbf{P}$ be a sentence such that, for every substitution instance $\mathbf{P}(\mathbf{a}/\mathbf{x})$, $\{\mathbf{P}(\mathbf{a}/\mathbf{x})\}$ has a closed tree. Does it follow that a systematic tree for $\{(\forall \mathbf{x})\mathbf{P}\}$ will close? Explain.

11. What would have to be done to make The System a mechanical procedure?

***12.** Suppose a tree for a set Γ of sentences of *PL* is abandoned without either closing or having a completed open branch. Suppose also that we find a model on which all the members of Γ are true. Suppose the model is an infinite model. Does it follow that all the open branches on the abandoned tree are nonterminating branches? Suppose the model is finite. Does anything follow regarding the abandoned tree?

9.5 TRUTH-TREES FOR *PLE*

To apply the tree method to sentences of the language *PLE,* we will modify the tree system developed in Sections 9.1–9.3 to accommodate the additional features of *PLE:* the identity predicate and complex terms. We shall introduce one new decomposition rule (Identity Decomposition), modify the definitions of a closed branch and of a completed open branch, and revise the Universal Decomposition rule to accommodate complex terms.[6]

We begin with the modification to Universal Decomposition, which is straightforward. The set $\{(\forall x) \sim Bx, Bf(c)\}$, which contains a closed complex term, 'f(c)', is clearly quantificationally inconsistent and so we want it to have a closed truth-tree. To this end, we need to allow Universal Decomposition to

[6]It is also possible to use the rule Existential Decomposition-2, developed in Section 9.4, for trees for *PLE*. We shall in fact do so in Section 9.6. But because Existential Decomposition is simpler in many cases (and because some readers may have chosen to skip 9.4), we revert to this rule for the purposes of this section.

yield substitution instances formed from *any* closed term, not just constants. For example, we want Universal Decomposition to license step 3 in the following tree:

1	$(\forall x) \sim Bx$	SM
2	$Bf(c)$	SM
3	$\sim Bf(c)$	1 $\forall$D
	$\times$	

We therefore revise Universal Decomposition as follows:

Universal Decomposition ($\forall$D)

$$(\forall \mathbf{x})\mathbf{P}$$

$$\mathbf{P}(\mathbf{t}/\mathbf{x})$$

where **t** is a closed term

This change allows the use of Universal Decomposition at line 3 of the previous tree, closing the tree and thus establishing that the set being tested is quantificationally inconsistent. We must also amend our definition of a completed open branch so as to require every universally quantified sentence to be decomposed to every substitution instance that can be formed from a *closed individual term* (individual constant or closed complex term) occurring on the branch in question.

We hasten to add that the rule Existential Decomposition remains the same for *PLE;* when we decompose an existentially quantified sentence $(\exists \mathbf{x})\mathbf{P}$ we will always use an individual constant **a** that is foreign to the branch on which the substitution instance $\mathbf{P(a/x)}$ will be entered, just as we did for *PL*. We will *not* use complex terms in these instantiations, because a complex term such as 'h(a)' carries information about the individual it denotes, namely, that the individual is related to some individual (that denoted by 'a') by the function h.

Here is the decomposition rule for identity sentences:

Identity Decomposition (=D)

$$\mathbf{t}_1 = \mathbf{t}_2$$

$$\mathbf{P}$$

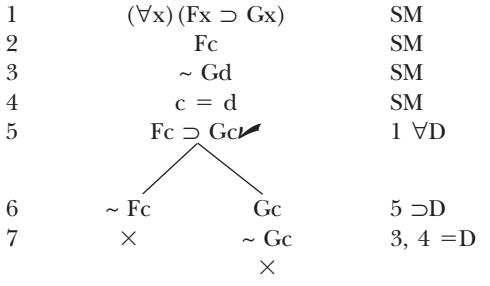
$$\mathbf{P}(\mathbf{t}_1//\mathbf{t}_2)$$

where $\mathbf{t}_1$ and $\mathbf{t}_2$ are closed individual terms and **P** is a literal containing $\mathbf{t}_2$

This rule is to be understood as follows: If a branch contains both a sentence of the form $\mathbf{t}_1 = \mathbf{t}_2$ (where $\mathbf{t}_1$ and $\mathbf{t}_2$ are closed individual terms) and a literal **P** containing the term $\mathbf{t}_2$, $\mathbf{P}(\mathbf{t}_1//\mathbf{t}_2)$—which is like **P** except that it contains

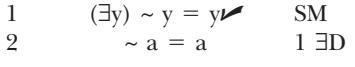
$\mathbf{t}_1$ *in at least one place* where **P** contains $\mathbf{t}_2$, may be entered on that branch. The rationale behind this rule is that if $\mathbf{t}_1$ and $\mathbf{t}_2$ designate one and the same thing then whatever is true of the individual designated by $\mathbf{t}_1$ must thereby be true of the individual designated by $\mathbf{t}_2$. Note that the identity sentence $\mathbf{t}_1 = \mathbf{t}_2$ is not checked off because it can be decomposed again and again.

Identity Decomposition is used at line 7 in the following tree:



Here $\mathbf{t}_1 = \mathbf{t}_2$ is 'c = d', **P** is '~ Gd', and $\mathbf{P}(\mathbf{t}_1//\mathbf{t}_2)$ is '~ Gc', the sentence entered on line 7, which is the result of substituting 'c' for 'd' in '~ Gd'. Note that the justification column for line 7 contains two line numbers, because Identity Decomposition licenses the entry of a sentence on a branch based on the presence of two other sentences.

Now that we have added a rule for Identity Decomposition we will need to modify the definition of a closed branch for *PLE*. To see why, consider the sentence '(∃y) ~ y = y'. This sentence says 'There is something that is not identical with itself' and is clearly quantificationally false. So we want the tree for the unit set of this sentence to close:

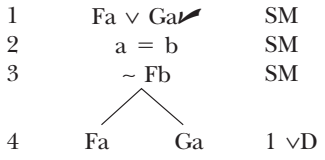


The one branch on this tree does not contain a pair of contradictory literals. So it is not, by our present account, a closed branch. But obviously, there can be no interpretation on which a sentence of the form ~ **t** = **t** is true; this is a consequence of the fixed interpretation of the identity predicate. So we modify our definition of a closed branch for *PLE* as follows:

> **Closed branch:** A branch on which contradictory literals occur or on which a sentence of the form ~ **t** = **t** occurs

By this revised account the single branch of the above tree is closed, so the tree itself is closed and we may conclude that the sentence '(∃y) ~ y = y' is indeed quantificationally false.

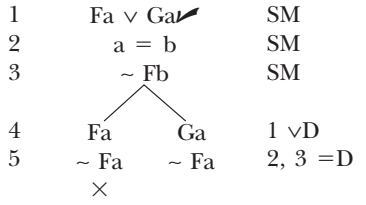
We noted earlier that we must amend our definition of a completed open branch so as to require every universally quantified sentence to be decomposed to every substitution instance that can be formed from a closed individual term (individual constant or closed complex term) occurring on the branch in question. But we also need to modify our definition of complete open branches in another way. To see why, consider the following tree for the set {Fa ∨ Ga, a = b, ~ Fb}, which is quantificationally consistent.



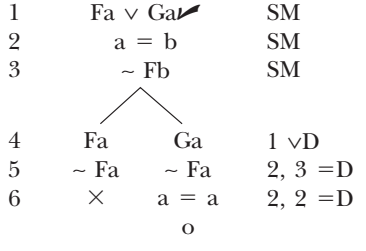
If we simply adopt the definition of completed open branches that we gave for trees for sentences of *PL,* both of the open branches on this tree will count as complete: on each branch every sentence is either a literal, or a sentence that is decomposed. But this result is not welcome, even though the set we are testing is quantificationally consistent, for the three literals 'a = b', '~ Fb', and 'Fa' on the left branch cannot all be true on a single interpretation. This branch should not count as a completed open branch from which an interpretation can be constructed. If we apply Identity Decomposition to the sentences on lines 3 and 4 we will add '~ Fa' to both branches, causing the left branch to close. So we want to be certain that Identity Decomposition is applied exhaustively. We therefore modify our account of completed open branches for trees for *PLE* by qualifying the first and third clauses and adding a fourth:

> A branch of a truth-tree for a set of sentences of *PLE* is a **completed open branch** if and only if it is a finite open branch (that is, an open branch with a finite number of sentences) and each sentence occurring on that branch is either
>
> 1. A literal that is not an identity sentence
> 2. A compound sentence that is not a universally quantified sentence and is decomposed
> 3. A universally quantified sentence $(\forall \mathbf{x})\mathbf{P}$ such that at least one substitution instance occurs on the branch, and *for each closed individual term* **t** occurring on the branch, the substitution instance $\mathbf{P}(\mathbf{t}/\mathbf{x})$ also occurs on the branch
> 4. A sentence of the form $\mathbf{t}_1 = \mathbf{t}_2$, where $\mathbf{t}_1$ and $\mathbf{t}_2$ are closed terms, such that for every literal **P** on that branch containing $\mathbf{t}_2$, the branch contains every sentence $\mathbf{P}(\mathbf{t}_1//\mathbf{t}_2)$ that can be obtained from **P** by Identity Decomposition.

Clause 4 requires that we continue work on our last tree, using Identity Decomposition to add '~ Fa' to both branches:



Here the left-hand branch has closed. The right-hand branch may appear to be a completed open branch, but it is not. This is because we must, by clause 4, replace 'b' with 'a' in *every* literal that occurs on the right branch, and 'a = b' is itself such a literal. Replacing 'b' with 'a' in this literal produces 'a = a'. This final application of Identity Decomposition yields the following completed tree, a tree with one closed branch and one completed open branch:



Adding a sentence of the form **t** = **t** will never, of course, bring about the closure of a branch, for if the negation of that sentence, ~ **t** = **t** were already on a branch, or were later added to a branch, the presence of that sentence by itself would close all the branches on which it occurs. For this reason we shall informally allow the omission of applications of Identity Decomposition that result in adding sentences of the form **t** = **t** to a branch. However, we will have to drop this informal practice when we develop *systematic trees* for *PLE* in Section 9.6, for the metatheory of Chapter 11 assumes that Identity Decomposition is rigorously applied in all such trees.

As we did for *PL,* we can use the literals on a completed open branch as a guide for constructing an interpretation on which all of the members of the set for which the tree was constructed are true. The open branch on the preceding tree has five literals: 'a = b', '~ Fb', 'Ga', '~ Fa', and 'a = a'. The last will be true on any interpretation, so it will not play a role in constructing a model for the set members. On the other hand, the identity 'a = b' tells us that 'a' and 'b' must designate the same individual. That suggests

that we try using a single-member UD, letting the constants 'a' and 'b' both designate that single member. To make the other tree literals true we need to intepret the predicate 'G' so that the single member is in its extension, and 'F' so that its extension *excludes* that single member:

UD:	{1}
a:	1
b:	1
F:	∅
G:	{<1>}

Any such interpretation is a model for the set {Fa ∨ Ga, a = b, ~ Fb}. Because *PLE* includes identity sentences such as 'a = b', when we construct interpretations of sets of sentences of *PLE* (rather than of *PL*) from completed open branches we must sometimes assign the same member of the UD to distinct constants.

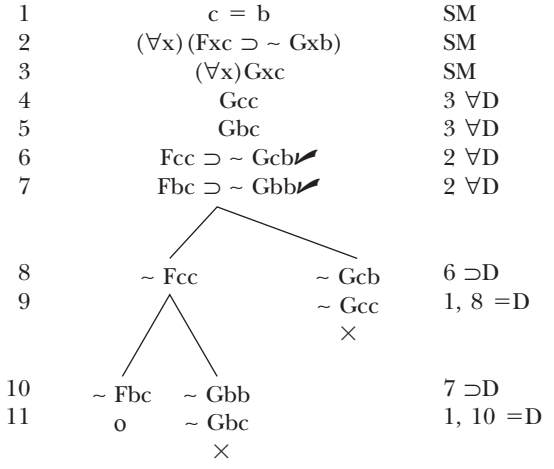
Given our expanded set of rules and revised definitions of closed and completed open branches, the explications developed in Sections 9.2 and 9.3 of semantic properties in terms of open and closed trees also hold for *PLE*. We therefore adopt them for *PLE* without repeating them here.

The set {a = b, (∀x)(Fbx & ~ Fxa)} is quantificationally inconsistent:

1	a = b	SM
2	(∀x)(Fbx & ~ Fax)	SM
3	Fbb & ~ Fab✓	2 ∀D
4	Fbb	3 &D
5	~ Fab	3 &D
6	Fab	1, 4 =D
	×	

What is interesting here is the use of Identity Decomposition at line 6. We generated 'Fab' from 'a = b' and 'Fbb' by replacing only the first occurrence of 'b' in the latter with 'a'. (Recall that when generating $\mathbf{P}(\mathbf{t}_1//\mathbf{t}_2)$ from $\mathbf{P}$, given $\mathbf{t}_1 = \mathbf{t}_2$, it is not required that *every* occurrence of $\mathbf{t}_2$ in $\mathbf{P}$ be replaced with $\mathbf{t}_1$ but only that at least one occurrence be so replaced.) We could also have closed the tree by using Identity Decomposition to enter 'Faa' line 6 (replacing both occurrences of 'b' in 'Fbb' with 'a') and then adding '~ Faa' as a new line 7 (replacing 'b' in '~ Fab' with 'a').

Consider now the quantificationally consistent set {c = b, (∀x)(Fxc ⊃ ~ Gxb), (∀x)Gxc}. Here is a tree for this set:



The left-hand branch of this tree is a completed open branch and hence establishes that the set is quantificationally consistent. The left-hand branch contains every required sentence that can be generated from the identity on line 1 and a literal containing 'b' (excepting the sentence 'c = c'). We could generate 'Gcc' by applying Identity Decomposition to the sentences at lines 1 and 5, but it already occurs on line 4 so there is no need to do so. Similarly we could generate '~ Fcc' from lines 1 and 10, but it already occurs at line 8. Given the literals 'c = b', 'Gcc', 'Gbc', '~ Fcc', and '~ Fbc', we know that any interpretation that includes the following assignments will be a model for the set {c = b, (∀x)(Fxc ⊃ ~ Gxb), (∀x)Gxc}:

UD: {1}
b: 1
c: 1
F: ∅
G: {<1, 1>}

Note that while Identity Decomposition allows, given an identity sentence $\mathbf{t}_1 = \mathbf{t}_2$, the generation of literals in which one or more occurrences of $\mathbf{t}_2$ in an existing literal have been replaced with $\mathbf{t}_1$, it does not license the generation of literals in which one or more occurrences of $\mathbf{t}_1$ have been replaced

with $\mathbf{t}_2$. We could rewrite Identity Decomposition so as to allow this, but it is not necessary to do so. That is, if a set is inconsistent it will have a closed tree given the rules as presently written. Rewriting Identity Decomposition in the suggested way would allow adding more literals to many trees, but for no useful purpose.

As explained in Chapter 7, identity is a reflexive, symmetric, and transitive relation. Accordingly, we expect the following sentences of *PLE,* which assert, respectively, the reflexivity, symmetry, and transitivity of identity, to be quantificationally true:

$$(\forall x)x = x$$
$$(\forall x)(\forall y)(x = y \supset y = x)$$
$$(\forall x)(\forall y)(\forall z)[(x = y \,\&\, y = z) \supset x = z]$$

The truth-tree method will indeed produce closed trees for the negations of these sentences. Here is the relevant tree for the claim that identity is reflexive:

1	$\sim (\forall x)x = x$ ✓	SM
2	$(\exists x) \sim x = x$ ✓	1 ~ ∀D
3	$\sim a = a$	2 ∃D
	×	

It should be noted that, when we earlier modified the definition of a closed branch so as to count every branch containing a sentence of the form '$\sim \mathbf{t}_1 = \mathbf{t}_1$' as a closed branch, we were, in effect, presupposing the reflexivity of identity. The present result is therefore neither a surprising one nor an independent proof of the reflexivity of identity. The relevant tree for the claim that identity is symmeric is

1	$\sim (\forall x)(\forall y)(x = y \supset y = x)$ ✓	SM
2	$(\exists x) \sim (\forall y)(x = y \supset y = x)$ ✓	1 ~ ∀D
3	$\sim (\forall y)(a = y \supset y = a)$ ✓	2 ∃D
4	$(\exists y) \sim (a = y \supset y = a)$ ✓	3 ~ ∀D
5	$\sim (a = b \supset b = a)$ ✓	4 ∃D
6	$a = b$	5 ~ ⊃D
7	$\sim b = a$	5 ~ ⊃D
8	$\sim a = a$	6, 7 =D
	×	

Finally, we consider transitivity. The relevant tree is

1	~ (∀x)(∀y)(∀z)[(x = y & y = z) ⊃ x = z]✓	SM
2	(∃x) ~ (∀y)(∀z)[(x = y & y = z) ⊃ x = z]✓	1 ~ ∀D
3	~ (∀y)(∀z)[(a = y & y = z) ⊃ a = z]✓	2 ∃D
4	(∃y) ~ (∀z)[(a = y & y = z) ⊃ a = z]✓	3 ~ ∀D
5	~ (∀z)[(a = b & b = z) ⊃ a = z]✓	4 ∃D
6	(∃z) ~ [(a = b & b = z) ⊃ a = z]✓	5 ~ ∀D
7	~ [(a = b & b = c) ⊃ a = c]✓	6 ∃D
8	(a = b & b = c)✓	7 ~ ⊃D
9	~ a = c	7 ~ ⊃D
10	a = b	8 &D
11	b = c	8 &D
12	a = c	10, 11 =D
	×	

Here we closed the tree by applying Identity Decomposition to lines 10 and 11, taking 'a = b' as $\mathbf{t}_1 = \mathbf{t}_2$ and 'b = c' as **P**, producing 'a = c' as $\mathbf{P}(\mathbf{t}_1//\mathbf{t}_2)$. At this point the one branch of the tree contains a pair of contradictory literals, 'a = c' and '~ a = c', and is therefore closed.

Consider now the sentence '(∀x)(∀y)[(Fxx & ~ Fyy) ⊃ ~ x = y]'. We expect this sentence to be quantificationally true (if x but not y bears a relation F to itself, then x and y are not identical). The following truth-tree confirms this expectation:

1	~ (∀x)(∀y)[(Fxx & ~ Fyy) ⊃ ~ x = y]✓	SM
2	(∃x) ~ (∀y)[(Fxx & ~ Fyy) ⊃ ~ x = y]✓	1 ~ ∀D
3	~ (∀y)[(Faa & ~ Fyy) ⊃ ~ a = y]✓	2 ∃D
4	(∃y) ~ [(Faa & ~ Fyy) ⊃ ~ a = y]✓	3 ~ ∀D
5	~ [(Faa & ~ Fbb) ⊃ ~ a = b]✓	4 ∃D
6	Faa & ~ Fbb✓	5 ~ ⊃D
7	~ ~ a = b✓	5 ~ ⊃D
8	a = b	7 ~ ~ D
9	Faa	6 &D
10	~ Fbb	6 &D
11	~ Faa	8, 10 =D
	×	

At line 11 we replaced both occurrences of 'b' in '~ Fbb' with 'a' to generate '~ Faa'. Replacing just one occurrence, while allowed, would not have produced a closed tree.

We now test the argument

$$\frac{\begin{array}{l}(\exists x)Gxa \;\&\; \sim (\exists x)Gax \\ (\forall x)(Gxb \supset x = b)\end{array}}{\sim a = b}$$

for quantificational validity by constructing a tree for the premises and the negation of the conclusion:

1	$(\exists x)Gxa \;\&\; \sim (\exists x)Gax$ ✓		SM
2	$(\forall x)(Gxb \supset x = b)$		SM
3	$\sim \sim a = b$ ✓		SM
4	$a = b$		3 ~ ~ D
5	$(\exists x)Gxa$ ✓		1 &D
6	$\sim (\exists x)Gax$ ✓		1 &D
7	$(\forall x) \sim Gax$		6 ~ ∃D
8	Gca		5 ∃D
9	$Gab \supset a = b$		2 ∀D
10	$Gbb \supset b = b$		2 ∀D
11	$Gcb \supset c = b$ ✓		2 ∀D
12	$\sim Gaa$		7 ∀D
13	$\sim Gab$		7 ∀D
14	$\sim Gac$		7 ∀D
15	$\sim Gcb$	$c = b$	11 ⊃D
16	$\sim Gca$	|	4, 15 =D
17	×	$a = c$	4, 15 =D
18		Gaa	8, 17 =D
		×	

This tree is closed. Therefore the argument is quantificationally valid. The secret to keeping this tree reasonably concise—for it could have grown quite large—came from carefully studying the sentences on lines 9–11 to determine which should be decomposed first.

- Line 11 yields '~ Gcb' on the left branch when decomposed, and replacing 'b' with 'a' in '~ Gcb' by virtue of the identity on line 4 then yields '~ Gca' at line 16, closing the left branch.
- At line 15 the right branch is still open and contains the identity 'c = b' in addition to 'a = b'. From these identities and the other literals on the branch ('Gca', '~ Gaa', '~ Gab', and '~ Gac') a host of sentences can be obtained by using Identity Decomposition, replacing 'b' with either 'a' or 'c'. Careful study reveals that any of 'Gac', 'Gab', 'Gaa', or '~ Gca' would close the branch. But none of these

can be directly obtained from the existing literals by Identity Decomposition using the two identities 'a = b' and 'c = b'.

- However, we were able to obtain the additional identity 'a = c' on line 17 by applying Identity Decomposition to the two identities themselves, replacing 'b' with 'c' in 'a = b' (as licensed by 'c = b'). This identity allowed us to obtain 'Gaa' on line 18, which closed the branch and the tree.

In the remainder of this section, we shall work through examples involving functors and identity. Consider first the sentence '~ (∃x)x = *g*(a)'. This sentence is fairly obviously quantificationally false, for it says that there is nothing that is identical to *g*(a); but, of course, we know that something *is* identical to *g*(a), namely, *g*(a) itself. Here is the start of a tree:

1	~ (∃x)x = *g*(a)✓	SM
2	(∀x) ~ x = *g*(a)	1 ~ ∃D
3	~ a = *g*(a)	2 ∀D

As of line 3 this tree has one open branch. It might seem that this branch is a completed open branch, and hence that our intuitions about the sentence we are testing must have been misguided. But the branch is not completed, for in addition to 'a', there is a closed individual term on the branch, '*g*(a)', and the universally quantified sentence on line 2 has not been decomposed to a substitution instance formed from this latter term. Adding this decomposition results in the following *closed* tree:

1	~ (∃x)x = *g*(a)✓	SM
2	(∀x) ~ x = *g*(a)	1 ~ ∃D
3	~ a = *g*(a)	2 ∀D
4	~ *g*(a) = *g*(a)	2 ∀D
	×	

It is now apparent that line 3 was unnecessary—the branch would close without that step.

Consider next the sentence '(∀x)*f*(x) = x'. This sentence is clearly not quantificationally true—because a one-place function does not always return its argument as its value. For example, the successor function returns x + 1 for any value x, not x itself. The following tree establishes that this sentence is not quantificationally true:

1	~ (∀x)*f*(x) = x✓	SM
2	(∃x) ~ *f*(x) = x✓	1 ~ ∀D
3	~ *f*(a) = a	2 ∃D
	o	

The single branch on this tree is a completed open branch. The sentences on lines 1 and 2 have been checked off, and the sentence on line 3 is a literal that is not an identity sentence. The sentence '$(\forall x)f(x) = x$' is therefore not quantificationally true.[7] Moreover, we can use the literals on the branch as a guide to constructing a model for '$\sim (\forall x)f(x) = x$' (which will be an interpretation on which the unnegated sentence is false). There is one literal on the single open branch: '$\sim f(a) = a$'. Because '$f(a)$' and 'a' must denote distinct individuals for this literal to be true, we choose a two-member UD, let 'a' designate one member, and interpret 'f' so that '$f(a)$' designates the other. Any interpretation that includes the following assignments will be a model for '$\sim (\forall x)f(x) = x$':

UD: {1, 2}
a: 2
f: {<1, 2>, <2, 1>}

The sentence '$(\forall x)(\forall y)f(x,y) = f(y,x)$' is also not quantificationally true, as the following tree shows:

1	$\sim (\forall x)(\forall y)f(x,y) = f(y,x)$ ✓	SM
2	$(\exists x) \sim (\forall y)f(x,y) = f(y,x)$ ✓	1 ~ ∀D
3	$\sim (\forall y)f(a,y) = f(y,a)$ ✓	2 ∃D
4	$(\exists y) \sim f(a,y) = f(y,a)$ ✓	3 ~ ∃D
5	$\sim f(a,b) = f(b,a)$	4 ∃D

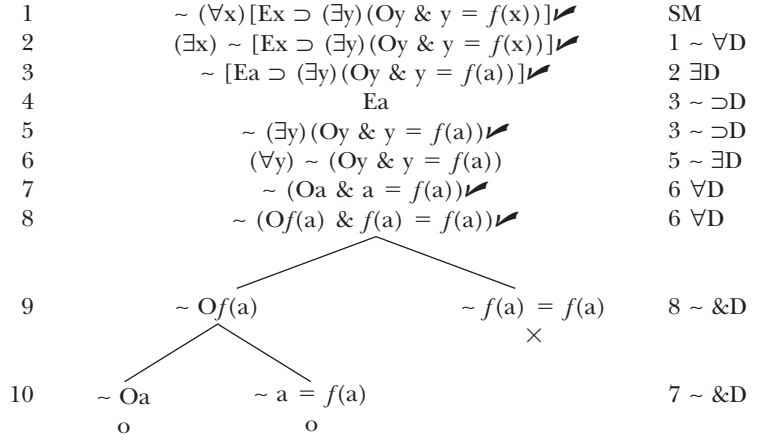
o

The completed open branch contains the literal '$\sim f(a,b) = f(b,a)$', so any interpretation that makes this literal true will also make the sentence '$\sim (\forall x)(\forall y)f(x,y) = f(y,x)$' true. To find such an interpretation we'll choose a two-member UD and interpret 'f' so that '$f(a,b)$' and '$f(b,a)$' do *not* denote the same member. Any interpretation that includes the following assignments will be a model for the sentence '$\sim (\forall x)(\forall y)f(x,y) = f(y,x)$' and hence an interpretation on which the unnegated sentence '$(\forall x)(\forall y)f(x,y) = f(y,x)$' is false:

UD: {1,2}
a: 1
b: 2
f: {<1, 1, 1>, <1, 2, 1>, <2, 2, 2>, <2, 1, 2>}

[7]We shall produce a tree in Section 9.6 that shows that this sentence is not quantificationally false.

Next consider the sentence '$(\forall x)[Ex \supset (\exists y)(Oy \,\&\, y = f(x))]$'. We can use the truth-tree method to show that this sentence is not quantificationally true:



This tree has two completed open branches. The tree contains no (nonnegated) identity sentences, and every sentence on each of these branches either is a literal, or has been checked off, or is a universally quantified sentence. There is only one of the latter, at line 6, and it has been decomposed to every closed term on the relevant branch (each branch contains only the closed terms 'a' and 'f(a)'). Because this tree has at least one completed open branch, the sentence '$(\forall x)[Ex \supset (\exists y)(Oy \,\&\, y = f(x))]$' is not quantificationally true. From the three literals 'Ea', '~ Of(a)', and '~ Oa' on the left open branch we know that '$\sim (\forall x)[Ex \supset (\exists y)(Oy \,\&\, y = f(x))]$' will be true on any interpretation that includes the following assignments:

UD: The set {1}
a: 1
f: {<1, 1>}
E: {<1>}
O: ∅

From the three literals 'Ea', '~ Of(a)', and '~ a = f(a)' on the right open branch we also know that '$\sim (\forall x)[Ex \supset (\exists y)(Oy \,\&\, y = f(x))]$' will be true on any interpretation that includes the following assignments:

UD: The set {1, 2}
a: 1
f: {<1, 2>, <2, 2>}
E: {<1>}
O: ∅

By definition a one-place function returns exactly one value for each argument. So the following sentence is quantificationally true:

$$(\forall x)(\exists y)[y = f(x) \ \& \ (\forall z)(z = f(x) \supset z = y)]$$

Here is a tree that establishes this:

1	$\sim (\forall x)(\exists y)[y = f(x) \ \& \ (\forall z)(z = f(x) \supset z = y)]$ ✓	SM
2	$(\exists x) \sim (\exists y)[y = f(x) \ \& \ (\forall z)(z = f(x) \supset z = y)]$ ✓	1 ~ ∀D
3	$\sim (\exists y)[y = f(a) \ \& \ (\forall z)(z = f(a) \supset z = y)]$ ✓	2 ∃D
4	$(\forall y) \sim [y = f(a) \ \& \ (\forall z)(z = f(a) \supset z = y)]$	3 ~ ∃D
5	$\sim [f(a) = f(a) \ \& \ (\forall z)(z = f(a) \supset z = f(a))]$ ✓	4 ∀D
6	$\sim f(a) = f(a)$	$\sim (\forall z)(z = f(a) \supset z = f(a))$ ✓	5 ~ &D
7	×	$(\exists z) \sim (z = f(a) \supset z = f(a))$ ✓	6 ~ ∀D
8		$\sim (b = f(a) \supset b = f(a))$ ✓	7 ∃D
9		$b = f(a)$	8 ~ ⊃D
10		$\sim b = f(a)$	8 ~ ⊃D
		×	

Note that at line 5 we chose to replace 'y' with 'f(a)' rather than with 'a'. Both are individual terms already occurring on the branch, but using the former generates a closed branch on the left at line 6 and, a few steps later, a closed branch on the right.

Finally, the following argument is quantificationally invalid:

$$\frac{(\forall y)y = f(y) \supset (\forall x)(\exists y)y = f(x)}{(\forall y)y = f(y)}$$

1	$(\forall y)y = f(y) \supset (\forall x)(\exists y)y = f(x)$ ✓	SM
2	$\sim (\forall y)y = f(y)$ ✓	SM
3	$(\exists y) \sim y = f(y)$ ✓	2 ~ ∀D
4	$\sim a = f(a)$	3 ∃D
5	$\sim (\forall y)y = f(y)$ ✓	$(\forall x)(\exists y)y = f(x)$	1 ⊃D
6	$(\exists y) \sim y = f(y)$ ✓	5 ~ ∀D
7	$\sim b = f(b)$	6 ∃D
	o	

The completed open branch establishes that the argument is quantificationally invalid. Note that every sentence on that branch either is checked off (lines 1, 2, 3, 5, and 6), or is a negated identity sentence (thus a literal that is not itself an identity sentence). It is worth noting that the right branch of this tree is the beginning of an infinite branch because of the interplay of the existential quantifier within the scope of the universal quantifier of the sentence on line 5

of that branch. From the completed open branch, however, which contains the literals '~ a = f(a)' and '~ b = f(b)', we know that any interpretation that includes the following assignments is a model for the set {(∀y) y = f(y) ⊃ (∀x) (∃y) y = f(x), ~ (∀y) y = f(y)}:

UD: The set {1, 2}
a: 1
b: 2
f: {<1, 2>, <2, 1>}

9.5**E** EXERCISES

Construct truth-trees as necessary to provide the requested information. In each case state your result, and specify what it is about your tree that establishes this result.

1. Determine, for each of the following sets, whether the set is quantificationally consistent. In addition, if your tree establishes consistency, show the relevant part of an interpretation that will make all of the literals on one completed branch, and therefore all of the members of the set being tested, true. (Be sure to list the literals that you are using in this case.)

a. {(∀x)Fxx, (∃x)(∃y) ~ Fxy, (∀x)x = a}
*b. {(∀x)(Fxc ⊃ x = a), ~ c = a, (∃x)Fxc}
c. {(∀x)(x = a ⊃ Gxb), ~ (∃x)Gxx, a = b}
*d. {(∃x)(∃y) ~ x = y, (∀x)(Gxx ⊃ x = b), Gaa}
e. {(∀x)((Fx & ~ Gx) ⊃ ~ x = a), Fa & ~ Ga}
*f. {(∃y)(∀x)Fxy, ~ (∀x)(∀y)x = y, Fab & ~ Fba}
g. {(∀x)(x = a ⊃ Gxf(b)), ~ (∃x)Gxf(x), f(a) = f(b)}
*h. {(∀x)(Gxx ⊃ x = f(x,b)), Gaa, (∀x) ~ f(a,x) = a}
i. {(∃x) ~ x = g(x), (∀x)(∀y)x = g(y)}
*j. {(∃x)(∃y)f(x,y) = f(y,x), (∀x) [f(x,a) = f(a,x) ⊃ ~ a = x]}
k. {(∀x)[Hx ⊃ (∀y)Txy], (∃x)Hf(x), ~ (∃x)Txx}
*l. {Hf(a,b), (∀x)(Hx ⊃ ~ Gx), (∃y)Gy}
m. {(∃x)Fx ⊃ (∃x)(∃y)f(y) = x, (∃x)Fx}
*n. {~(∃x)[x = f(s) & (∀y)(y = f(s) ⊃ y = x)]}

2. Determine, for each of the following sentences, whether it is quantificationally true, quantificationally false, or quantificationally indeterminate.

a. a = b ≡ b = a
*b. (~ a = b & ~ b = c) ⊃ ~ a = c
c. (Gab & ~ Gba) ⊃ ~ a = b
*d. (∀x)(∃y)x = y
e. Fa ≡ (∃x)(Fx & x = a)
*f. ~ (∃x)x = a
g. (∀x)x = a ⊃ [(∃x)Fx ⊃ (∀x)Fx]
*h. (∀x)(∀y)x = y
i. (∀x)(∀y) ~ x = y
*j. (∃x)(∃y)x = y

k. $(\exists x)(\exists y) \sim x = y$
*l. $(\forall x)(\forall y)[x = y \supset (Fx \equiv Fy)]$
m. $(\forall x)(\forall y)[(Fx \equiv Fy) \supset x = y]$
*n. $(\forall x)(\forall y)[x = y \supset (\forall z)(Fxz \equiv Fyz)]$
o. $[(\exists x)Gax \;\&\; \sim (\exists x)Gxa] \supset (\forall x)(Gxa \supset \sim x = a)$

3. Determine which of the following sentences are quantificationally true.

a. $(\exists x)x = f(a)$
*b. $(\forall x)(\exists y)y = f(x)$
c. $(\exists x)(\exists y)x = y$
*d. $(\exists x)(\exists y)x = f(y)$
e. $(\forall x)[Gx \supset (\exists y)f(x) = y]$
*f. $(\forall x)(\forall y)[x = y \supset f(x) = f(y)]$
g. $(\forall y) \sim [(\forall x)x = y \vee (\forall x)f(x) = y]$
*h. $(\forall x)(\exists y)[y = f(x) \;\&\; (\forall z)(z = f(x) \supset z = y)]$

4. Determine which of the following pairs of sentences are quantificationally equivalent.

a.	$\sim a = b$	$\sim b = a$
*b.	$(\exists x) \sim x = a$	$(\exists x) \sim x = b$
c.	$(\forall x)x = a$	$(\forall x)x = b$
*d.	$a = b \;\&\; b = c$	$a = c \;\&\; b = c$
e.	$(\forall x)(\forall y)x = y$	$(\forall x)x = a$
*f.	$(\forall x)(\exists y)x = y$	$(\forall y)(\exists x)x = y$
g.	$(\forall x)(Fx \supset x = a)$	$(\forall x)(Fa \supset x = a)$
*h.	$(\forall x)(x = a \vee x = b)$	$(\forall x)x = a \vee (\forall x)x = b$
i.	$(\forall x)Fx \vee (\forall x) \sim Fx$	$(\forall y)(Fy \supset y = b)$
*j.	$a = b$	$(\forall y)(y = a \supset y = b)$
k.	$(\exists x)(x = a \;\&\; x = b)$	$a = b$

5. Determine which of the following arguments are quantificationally valid.

a. $a = b \;\&\; \sim Bab$
$\therefore \sim (\forall x)Bxx$

*b. $Ge \supset d = e$
$Ge \supset He$
$\therefore Ge \supset Hd$

c. $(\forall z)(Gz \supset (\forall y)(Ky \supset Hzy))$
$(Ki \;\&\; Gj) \;\&\; i = j$
$\therefore Hii$

*d. $(\exists x)(Hx \;\&\; Mx)$
$Ms \;\&\; \sim Hs$
$\therefore (\exists x)((Hx \;\&\; Mx) \;\&\; \sim x = s)$

e. $a = b$
$\therefore Ka \vee \sim Kb$

*f. $(\exists x) \sim Pxx \supset \sim a = a$
$a = c$
$\therefore Pac$

g. $(\forall x)(x = a \vee x = b)$
$(\exists x)(Fxa \;\&\; Fbx)$
$\therefore (\exists x)Fxx$

*h. $(\exists x)Fxa$
$(\forall y)(y = a \supset y = b)$
$\therefore (\exists y)Fyy$

i. $(\forall x)(\forall y)(Fxy \vee Fyx)$
$a = b$
$(\forall x)(Fxa \vee Fbx)$

*j. $(\exists x)Fxa \ \& \ (\exists x)Fxb$
$\sim a = b$
$(\forall x)(\forall y)((Fxa \ \& \ Fyb) \supset \sim x = y)$

k. $(\forall x)(Fx \equiv \sim Gx)$
Fa
Gb
$\sim a = b$

*l. $\sim (\exists x)Fxx$
$(\forall x)(\forall y)(Fxy \supset \sim x = y)$

m. $(\forall x)(\forall y)x = y$
$\sim (\exists x)(\exists y)(Fx \ \& \sim Fy)$

*n. $(\forall x)(\sim x = a \equiv (\exists y)Gyx)$
Gbc
$\sim c = a$

o. $(\forall x)(Hx \supset Hf(x))$
$(\exists z) \sim Hf(z)$
$\sim (\forall x)Hx$

*p. $(\forall y)(Hy \supset g(y) = y)$
$(\exists x) \sim g(x) = x$
$(\exists x) \sim Hx$

q. $(\forall x)(\forall y)(Hxy \equiv \sim Hyx)$
$(\exists x)[Hxf(x) \ \& \sim Hf(x)x]$
$\sim (\forall x)f(x) = x$

*r. $(\exists x)h(x) = x$
$(\forall x)(Fx \supset \sim Fh(x))$
$(\exists x) \sim Fx$

s. $(\forall x)[Px \supset (Ox \vee \sim x = f(b))]$
$(\exists x)[(Px \ \& \sim Ox) \ \& \ x = f(b)]$
Ob

*t. $(\forall x)(\forall y)(Hxy \supset f(x) = y)$
$(\exists x)Hxx$
$(\exists x)f(x) = x$

6. Determine which of the following claims are true.

a. $\{(\forall x)(Fx \supset (\exists y)(Gyx \ \& \sim y = x)), (\exists x)Fx)\} \models (\exists x)(\exists y) \sim x = y$
*b. $\{\sim (\exists x)(Fxa \vee Fxb), (\forall x)(\forall y)(Fxy \supset \sim x = y)\} \models \sim a = b$
c. $\{(\forall x)(Fx \supset \sim x = a), (\exists x)Fx\} \models (\exists x)(\exists y) \sim x = y$
*d. $\{(\forall x)(\exists y)(Fxy \ \& \sim x = y), a = b, Fab\} \models (\exists y)(Fay \ \& \ y = b)$
e. $\{(\exists w)(\exists z) \sim w = z, (\exists w)Hw\} \models (\exists w) \sim Hw$
*f. $\{(\exists w)(\forall y)Gwy, (\exists w)(\forall y)(\sim w = y \supset \sim Gwy)\} \models (\exists z) \sim Gzz$
g. $\{(\forall x)(\forall y)((Fx \equiv Fy) \equiv x = y), (\exists z)Fz\} \models (\exists x)(\exists y)(\sim x = y \ \& \ (Fx \ \& \sim Fy))$
*h. $\{(\forall x)(\exists y)y = f(x)\} \models (\exists z)z = f(a)$
i. $\{(\forall x)(\forall y)[\sim x = g(y) \supset Gxy], \sim (\exists x)Gax\} \models (\exists x)a = g(x)$

9.6 FINE-TUNING THE TREE METHOD FOR *PLE*

The last tree that we presented in Section 9.5 contained an unending branch in the making, due to a sentence that contained an existential quantifier within the scope of a universal quantifier. We introduced a new rule in

Section 9.4, Existential Decomposition-2, as well as a systematic method of constructing trees for *PL,* to ensure that such branches would neither prevent discovering completed open branches (where they exist) nor prevent closing trees for inconsistent sets. Because the tree method for *PLE* includes all of the rules for *PL,* we will clearly need to address infinite branches arising from the interplay between existential and universal quantifiers here as well. But the inclusion of functors in *PLE* creates an additional source of nonterminating branches in trees for finite sets of sentences. Consider a tree for the set {(∀x)H*f*(x)}:

1	(∀x)H*f*(x)	SM
2	H*f*(a)	1 ∀D
3	H*f*(*f*(a))	1 ∀D
4	H*f*(*f*(*f*(a)))	1 ∀D
	•	
	•	
	•	

To qualify as a completed open branch, every universally quantified sentence on that branch must be decomposed at least once and must be decomposed to every closed term occurring on the branch. To satisfy the first requirement, we first decompose the universally quantified sentence occurring on line 1 using the constant 'a'. This introduces *two* new closed terms to the one branch of the tree: 'a' and '*f*(a)' (both occurring within the formula 'H*f*(a)'). The universal quantification has just been decomposed using 'a', but now we must also decompose it using the closed term '*f*(a)'. This produces line 3, and a new closed term, '*f*(*f*(a))', which triggers another decomposition of the universally quantified sentence, producing a new closed term, '*f*(*f*(*f*(a)))', at line 4, and so on. Clearly this branch will never close and will never become a completed open branch.

As for *PL,* we would like to have a tree system for *PLE* such that every finite inconsistent set has a closed tree *and* every finite set with a finite model has a finite tree with a completed open branch. We just saw that we cannot produce a finite tree for the set {(∀x)H*f*(x)}, given the methods presented for *PLE* in Section 9.5. Yet this set has a finite model, for example, any interpretation that includes the following assignments:

UD:	The set {2}
f:	{<2, 2>}
H:	{<2>}

In this section we will modify our definition of completed open branches, add a new decomposition rule for constructing *PLE* trees, and then present a systematic method for constructing trees such that our desiderata are satisfied.

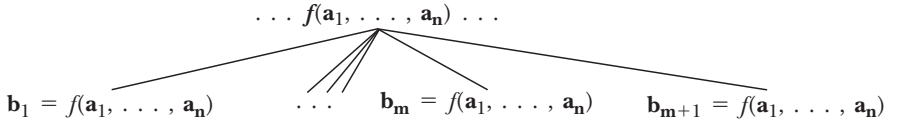
We will modify the definition of completed open branches in several ways. The first of these will be to drop the requirement that universally quantified sentences be decomposed using *every* closed term occurring on a branch, and adopt the weaker requirement that universally quantified sentences must be decomposed using every *constant* occurring on a branch (and that at least one constant must be so used). This will cut short the infinite branch that we just saw in progress for the set {(∀x)H*f*(x)}, but without other changes it will also count the single branch of the following tree as a completed open branch:

1	(∀x)Bx	SM
2	~ B*g*(a)	SM
3	Ba	1 ∀D

We clearly don't want to count this as a completed open branch, for the set {(∀x)Bx, ~ B*g*(a)} is quantificationally inconsistent. Because we are required to decompose universally quantified sentences only with constants, we will add a new rule that identifies the individuals denoted by closed complex terms with individuals identified by contants, a rule that will lead to a closed tree for this set once we make a final modification of the definition of open branches.

The new rule is:[8]

Complex Term Decomposition (CTD)

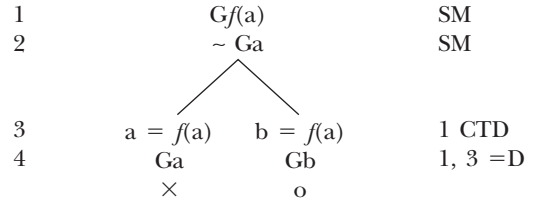


> where $f(\mathbf{a}_1, \ldots, \mathbf{a}_n)$ is a closed complex term occurring within a literal and whose arguments $\mathbf{a}_1, \ldots, \mathbf{a}_n$ are individual constants; $\mathbf{b}_1, \ldots, \mathbf{b}_m$ are the constants that already occur on the branch that contains the literal, and $\mathbf{b}_{m+1}$ is a constant that is foreign to that branch.

The expression '. . . $f(\mathbf{a}_1, \ldots, \mathbf{a}_n)$ . . .' stands for any literal that contains the complex term '$f(\mathbf{a}_1, \ldots, \mathbf{a}_n)$'. This rule bears an obvious affinity to Existential Decomposition-2: it branches out based on the constants that occur on the branch containing the complex term being decomposed, and generates one additional branch with a constant that was foreign to that branch. At the end of each of the new branches is an identity sentence, with one of the constants on the left-hand side and the complex term being decomposed on the right-hand side.

[8]This is a slight variation of the rule introduced in Merrie Bergmann, "Finite Tree Property for First-Order Logic with Identity and Functions," *Notre Dame Journal of Formal Logic,* 46 (2005), pp. 173–180.

The following tree for the set {G*f*(a), ~ Ga} illustrates the use of Complex Term Decomposition:

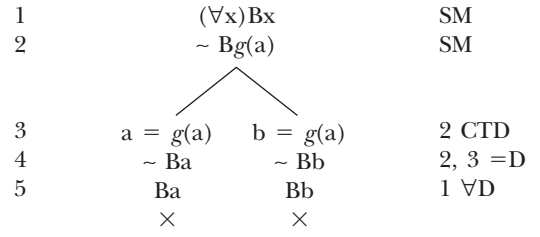


The tree begins with the set members on the first two lines. The sentence on line 1 is a literal containing the closed complex term '*f*(a)', so Complex Term Decomposition must be applied to this term. Prior to applying the rule there is one constant, 'a', that occurs on the single branch. So one of the new branches must end with the identity sentence 'a = *f*(a)', while the other ends with an identity sentence that has a new constant on the left-hand side. (Note that we do not check off the sentence containing the closed term that is being decomposed. Checks will continue to indicate completed *sentence* decomposition only.) The tree is then extended to line 4, because the identity sentences on line 3 must be decomposed by substituting the constants for '*f*(a)' in the literal 'G*f*(a)'. The left branch closes, as it should, because if 'a' and '*f*(a)' denote the same individual, then the set members state that this individual both does and does not have the property G. The right branch is a completed open branch, confirming that the set {G*f*(a), ~ Ga} is quantificationally consistent. (Strictly speaking, the formula 'b = b' should also appear on the right branch by virtue of applying Identity Decomposition to the single formula on line 3 of that branch. But here, as in Section 9.5, we omit identity formulas in which the same term appears on both sides of the identity predicate because such formulas will never cause a branch to close.) Because the open branch contains two individual constants, it indicates that we can construct a model for the set using a UD with at least two members, for example, any interpretation that includes the following assignments:

UD: {1, 2}
a: 1
b: 2
f: {<1, 2>, <2, 1>}
G: {<2>}

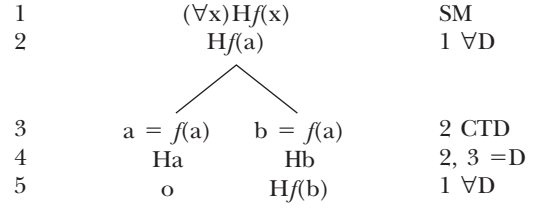
(In addition, the fact that the left branch, the only branch that contains exactly one individual constant, closes tells us that any model for this set *must* have at least two members in its UD.)

Using Complex Term Decomposition we can produce a closed tree for the quantificationally inconsistent set {(∀x)Bx, ~ B*g*(a)}:



Although we dropped the requirement that universally quantified sentences must be decomposed using closed complex terms as well as constants, this tree nevertheless closes because of the identity sentences that were generated on line 3 by Complex Term Decomposition. Branching to those sentences says that either the constant 'a' or the constant 'b' denotes the individual that the complex term '*g*(a)' denotes. Further, the identity sentences must themselves be decomposed with Identity Decomposition, and respectively substituting the constants 'a' and 'b' for '*g*(a)' in the sentence '~ B*g*(a)' produces '~ Ba' on the left branch and '~ Bb' on the right branch. Finally, because universally quantified sentences *must* be decomposed using all the *constants* occurring on a branch, we add the substitution instances on line 5 that respectively contradict the sentences on line 4 of the two branches. So the tree closes.

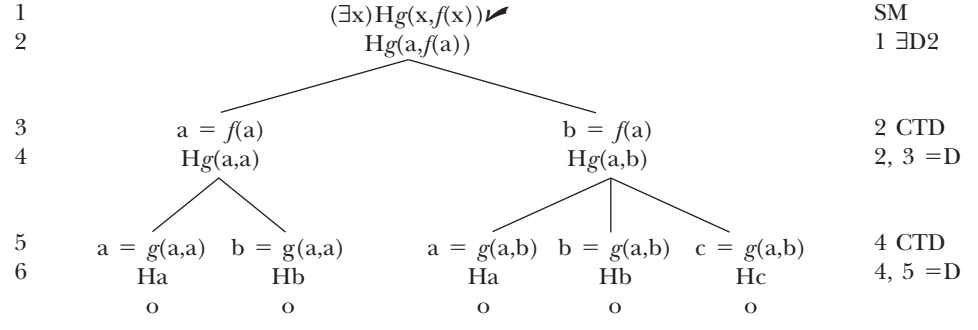
On the other hand, Complex Term Decomposition and our modified requirement for decomposing universally quantified sentences produce a completed open branch on a tree for the set {(∀x)H*f*(x)}:



After adding the sentences on line 3 with Complex Term Decomposition, we substitute the constants for the complex term '*f*(a)' in the literal on line 2 to generate the literals on line 4. The left branch is now completed: We have decomposed the universal quantification on line 1 with the only constant on the branch (this occurs at line 2), we have decomposed the complex term on line 2 (although we haven't yet stated so, there will be a requirement that all complex terms must be decomposed), and the identity on line 3 has also been decomposed as many times as it can be (it produces only the sentence on line 4). This branch is also open, indicating that the set being tested is quantificationally consistent. The right branch is also open, but it

is not completed. On line 5 we have added another substitution instance of the universal quantification, because the right branch now contains the constant 'b' along with the constant 'a'. This generates a new closed complex term, to which Complex Term Decomposition must be applied. An infinite branch is in the making here, for Complex Term Decomposition will produce a new branch with the constant 'c', which must be used to form a substitution instance of the universal quantification on line 1, and so on. The important point is, however, that we have managed to produce a completed open branch. That branch contains exactly one constant, confirming (as we already saw) that there is a model for the set {(∀x)H*f*(x)} in which the UD has exactly one member.

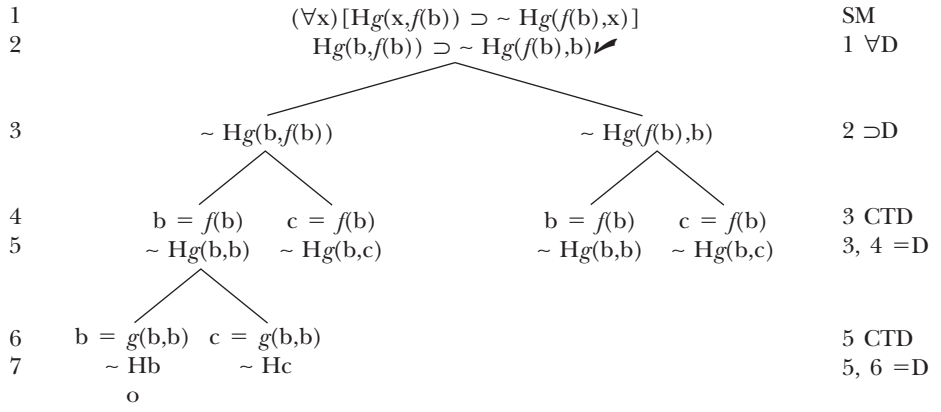
Having seen Complex Term Decomposition in action, we will now examine a tree that illustrates some of the finer points of this rule:



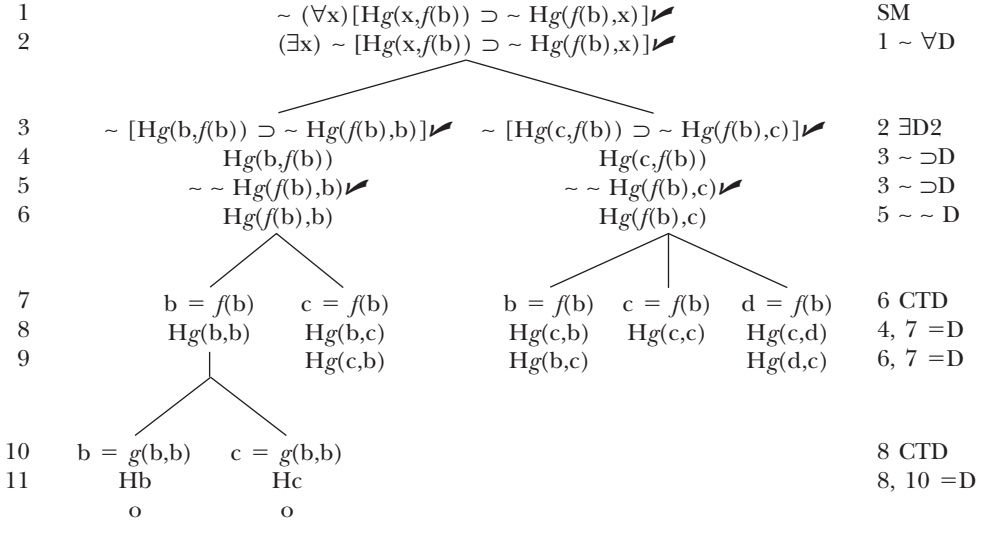
(In this section, as in Section 9.4, we will always use Existential Decomposition-2 to decompose existentially quantified sentences.) The sentence on line 1 contains the complex term '*g*(x,*f*(x))' but that term is not decomposed because it is not a *closed* term. The sentence on line 2 contains two complex terms: '*f*(a)' and '*g*(a,*f*(a))'. Both are closed, but only '*f*(a)' gets decomposed (on line 3). The term '*g*(a,*f*(a))' does not get decomposed because Complex Term Decomposition only applies if all of the arguments to the main functor are constants, but the argument '*f*(a)' is not a constant. After Identity Decomposition adds identity sentences at line 4, the tree contains two new complex terms: '*g*(a,a)' and '*g*(a,b)'. These terms are closed, and the arguments in both are exclusively constants, so they must be decomposed—this is done on line 5. Identity Decomposition then produces the sentences on line 6, and all five branches become completed open branches.

Now we will use Complex Term Decomposition to produce trees that show that the sentence '(∀x)[H*g*(x,*f*(b)) ⊃ ~ H*g*(*f*(b),x)]' is quantificationally

indeterminate. The first tree shows that this sentence is not quantificationally false:



The leftmost branch is a completed open branch. The remaining branches are incomplete, but since the tree does have at least one completed open branch there is no need to complete them. In constructing the tree we reached the sentences '~ Hb' and '~ Hc' through repeated applications of Complex Term Decomposition and Identity Decomposition, applying CTD first to line 3 to produce the identities on line 4, then to '*g*(b,b)' as that closed term occurs in a literal on line 5. The following tree establishes that '(∀x)[H*g*(x,*f*(b)) ⊃ ~ H*g*(*f*(b),x)]' also is not quantificationally true, and hence that it is quantificationally indeterminate:



The left two branches of this tree are completed open branches, so there is no point in continuing to work on the other branches. Note that the closed term '*f*(b)' occurs in literals on lines 4 and 6. Nonetheless we applied CTD to this closed term only once—at line 7, citing line 6. We could equally well have cited line 4. (There is no point to applying CTD twice to the same closed term, as the results will always be the same.) Also note that we applied Identity Decomposition at line 8 to lines 4 and 7 on all five branches. At line 9 we applied it only to the branches where a new literal is yielded.

Before we give our official definition of open branches for trees of *PLE,* we pause to consider the restriction, in the statement of Complex Term Decomposition, that the complex term being decomposed must be a closed term *whose arguments are individual constants.* In each of the preceding three trees there are closed complex terms whose arguments include complex terms. Why are we not required to decompose these complex terms? Keep in mind that the point of Complex Term Decomposition was to ensure that for every closed complex term occurring on a branch, there is a constant on that branch that denotes the same individual (so that when decomposing universally quantified sentences, we only need to generate substitution instances that are formed from individual constants that occur on the branch). It turns out that this requirement is met as long as we are careful to apply CTD to all complex terms in which the arguments are all constants, and to decompose identity sentences wherever they occur. To see this, take as an example the term '*g*(b,*f*(b))' that occurs at line 4 in the literal on the left branch of the preceding tree. The *required* decompositions on the two completed open branches below line 4 guarantee that on each of these branches there is an individual constant that denotes the same member of the UD as '*g*(b,*f*(b))'. More specifically, consider the completed open branch on the left. The identity sentence on line 7 states that 'b' and '*f*(b)' denote the same individual, and from this it follows that '*g*(b,b)' must denote the same individual as the more complex term '*g*(b,*f*(b))'. Moreover, the identity sentence on line 10 states that 'b' and '*g*(b,b)' denote the same individual—from which it follows that on any interpretation represented by this branch, 'b' denotes the same individual as the complex term '*g*(b,*f*(b))'. Similarly, it follows from the identity sentences on the right completed open branch that on any interpretation represented by that branch, 'c' denotes the same individual as '*g*(b,*f*(b))'.

The availability of CTD justifies our relaxing clause 3 of our definition of a completed open branch so as *not* to require that universally quantified sentences be decomposed to substitution instances formed from closed complex terms. Such decompositions are still allowed and will sometimes produce closed trees sooner than will using CTD, but they are not required for a branch to be a completed open branch. It turns out that we can similarly loosen the requirement concerning the use of Identity Decomposition. So our revised definition of a completed open branch for trees for *PLE* is:

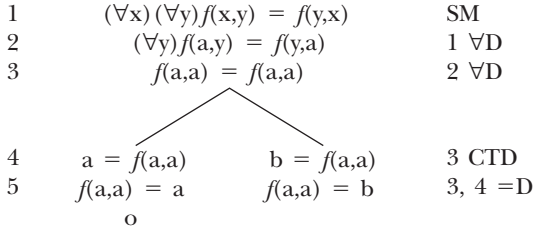
A **completed open branch** is a finite open branch on which each sentence is one of the following:

1. A literal that is not an identity sentence of the form **a** = **t**, where **a** is an individual constant and **t** is a closed term
2. A sentence of the form **a** = **t**, where **a** is an individual constant and **t** is a closed term such that the branch also contains, for every literal **P** on the branch containing **t**, every sentence **P**(**a**//**t**) that can be obtained from **P** by Identity Decomposition
3. A nonliteral sentence that is not a universally quantified sentence and is decomposed
4. A universally quantified sentence (∀**x**)**P** such that at **P**(**a**/**x**) also occurs on the branch for at least one constant **a**, and *for each constant* **a** occurring on the branch, **P**(**a**/**x**) also occurs.

and on which Complex Term Decomposition has been applied to all closed complex terms whose arguments are all individual constants that occur in literals on the branch.

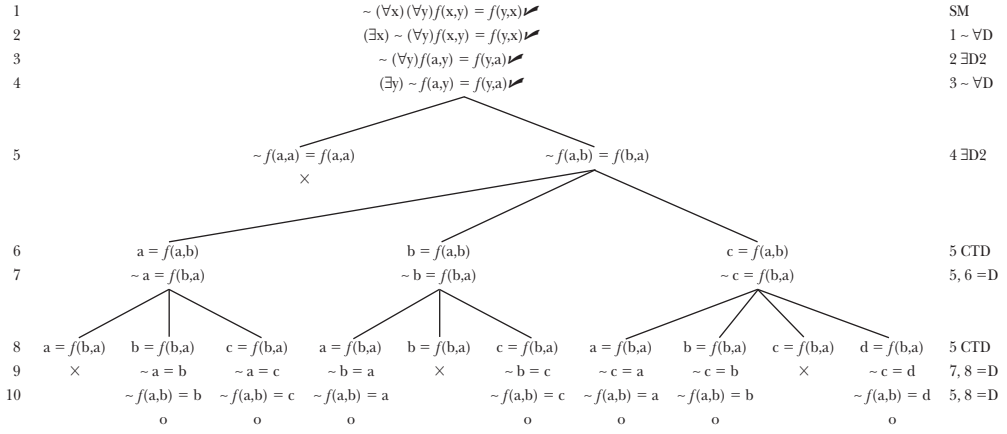
All branches that we have marked as completed open branches on trees in this section meet this definition (except that we do not bother to show the required identity sentences in which the same term occurs on both sides of the identity predicate). Note that Identity Decomposition can still be used on sentences of the form $\mathbf{t}_1 = \mathbf{t}_2$ when $\mathbf{t}_1$ is a complex term, but doing so is not necessary for producing closed branches. So, where $\mathbf{t}_1 = \mathbf{t}_2$ occurs on a branch and $\mathbf{t}_1$ is a complex term, Identity Decomposition should be used if we suspect that doing so will quickly produce a closed branch, but not otherwise.

We'll now use this revised definition of completed open branches as we construct trees that show that the sentence '(∀x)(∀y)*f*(x,y) = *f*(y,x)' is quantificationally indeterminate. We start by constructing a tree for the unit set of the sentence:



The left-hand branch is, as of line 5, a completed open branch. So the sentence is true on at least one interpretation and is therefore not quantificationally false. The right-hand branch is not completed, as it does not contain substitution instances of

the sentences on lines 1 and 2 that can be formed using the constant 'b' that occurs on that branch. Here is a tree for the unit set of the negation of our sentence:



This tree has seven completed open branches, so the sentence at line 1 is not quantificationally false, and the sentence '$(\forall x)(\forall y)f(x,y) = f(y,x)$' is not quantificationally true. This establishes that '$(\forall x)(\forall y)f(x,y) = f(y,x)$' is quantificationally indeterminate.

As we did for *PL,* we will present a procedure for constructing trees for *PLE* in a systematic fashion that will always, in a finite number of steps, find a completed open branch if one exists or close the tree if it can be closed. The System for *PLE* is somewhat more complicated than that for *PL,* owing to the presence of identity sentences and complex terms:

The System for PLE

List the members of the set to be tested.

Exit Conditions: Stop if

a. the tree closes, or

b. an open branch becomes a completed open branch.

Construction Procedures:

Stage 1: Decompose all truth-functionally compound and existentially quantified sentences and each resulting sentence that is itself either a truth-functional compound or an existentially quantified sentence.

Stage 2: For each universally quantified sentence $(\forall \mathbf{x})\mathbf{P}$ on the tree:

i) Enter $\mathbf{P}(\mathbf{a}/\mathbf{x})$ on each open branch passing through $(\forall \mathbf{x})\mathbf{P}$ for each individual constant **a** already occurring on that branch.

ii) On each open branch passing through $(\forall \mathbf{x})\mathbf{P}$ on which no constant occurs, enter $\mathbf{P}(a/\mathbf{x})$.

iii) Enter $\mathbf{P}(\mathbf{t}/\mathbf{x})$ on an open branch passing through $(\forall \mathbf{x})\mathbf{P}$ for a closed complex term $\mathbf{t}$ if and only if doing so closes the branch.

Repeat this process until every universally quantified sentence on the tree, including those added as a result of this process, has been so decomposed.

Stage 3: Apply Complex Term Decomposition to every complex term on an open branch whose arguments are all constants and to which Complex Term Decomposition has not already been applied.

Stage 4: For every sentence of the form $\mathbf{t}_1 = \mathbf{t}_2$ occurring on an open branch, apply Identity Decomposition as follows:

i) Where $\mathbf{t}_1$ is an individual constant, apply Identity Decomposition until every open branch passing through $\mathbf{t}_1 = \mathbf{t}_2$ also contains, for every literal $\mathbf{P}$ containing $\mathbf{t}_2$ on that branch, every sentence $\mathbf{P}(\mathbf{t}_1//\mathbf{t}_2)$ that can be obtained from $\mathbf{P}$ by Identity Decomposition.

ii) Where $\mathbf{t}_1$ is a closed complex term, apply Identity Decomposition to $\mathbf{t}_1 = \mathbf{t}_2$ and a literal $\mathbf{P}$ containing $\mathbf{t}_2$ that occurs on a branch passing through $\mathbf{t}_1 = \mathbf{t}_2$ if and only if doing so closes the branch.

Return to Stage 1.

Note that Stage 3 does not require us to apply Complex Term Decomposition to the same complex term on a branch more than once, even though the term may occur in more than one literal on that branch. Stage 4 ensures that after passing through that stage every sentence of the form $\mathbf{a} = \mathbf{t}$ on every open branch meets the requirements of clause 4 of the definition of a completed open branch. That is, if the branch is not completed at this point, this will *not* be because we have failed to apply Identity Decomposition the required number of times. Stages 2 and 4 each contain instructions to apply a decomposition rule in certain cases only if doing so closes a branch. The decompositions in question do not need to be done to meet the requirements for having a completed open branch, and doing such decompositions when the result is not a closed branch can produce infinite branches in cases where completed open branches are possible. Nor are such decompositions needed to produce closed branches—the branches in question will eventually close even without these decompositions—but they can result in less complex trees than would otherwise be produced by The System.

We conclude with some examples that use The System for *PLE*. The sentence '$(\forall x)(\exists y)y = f(f(x))$' is quantificationally true. So the truth-tree for the negation of that sentence should close, and it does:

1	~ (∀x)(∃y)y = *f*(*f*(x))✓	SM
2	(∃x) ~ (∃y)y = *f*(*f*(x))✓	1 ~ ∀D
3	~ (∃y)y = *f*(*f*(a))✓	2 ∃D2
4	(∀y) ~ y = *f*(*f*(a))	3 ~ ∃D
5	~ a = *f*(*f*(a))	4 ∀D
6	~ *f*(*f*(a)) = *f*(*f*(a))	4 ∀D
	×	

We closed this systematic tree by taking advantage of the instruction that a universally quantified sentence be decomposed to a substitution instance formed from a complex term if and only if doing so closes the branch.

On the other hand, the sentence '(∀x)(∀y)y = *f*(*f*(x))' is quantificationally indeterminate. The following tree shows that the sentence is not quantificationally true:

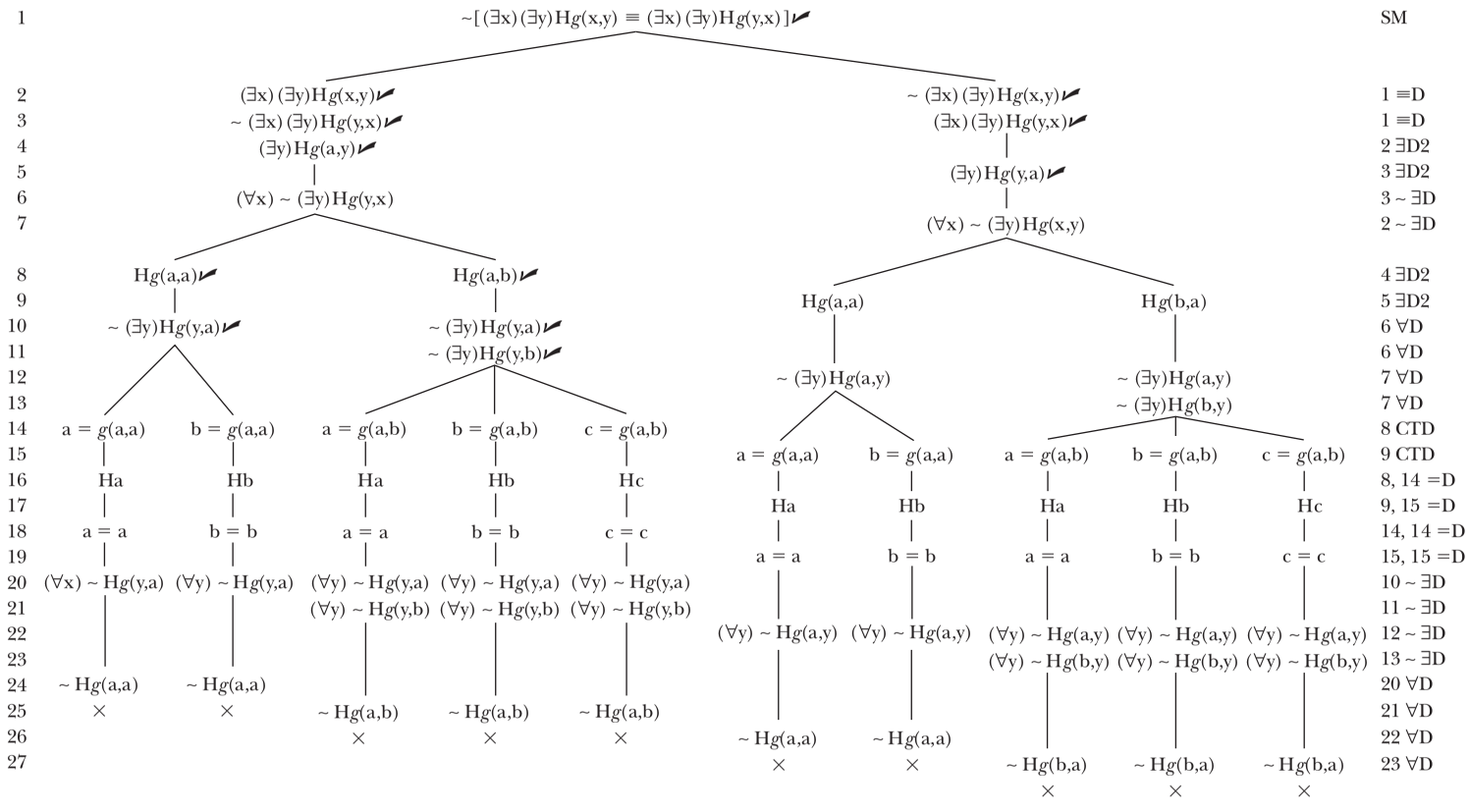
1	~ (∀x)(∀y)y = *f*(*f*(x))✓					SM
2	(∃x) ~ (∀y)y = *f*(*f*(x))✓					1 ~ ∀D
3	~ (∀y)y = *f*(*f*(a))✓					2 ∃D2
4	(∃y) ~ y = *f*(*f*(a))✓					3 ~ ∀D
5	~ a = *f*(*f*(a))		~ b = *f*(*f*(a))			4 ∃D2
6	a = *f*(a)	b = *f*(a)	a = *f*(a)	b = *f*(a)	c = *f*(a)	3 CTD
7	~ a = *f*(a)	~ a = *f*(b)	~ b = *f*(a)	~ b = *f*(b)	~ b = *f*(c)	5, 6 =D
8	a = a	b = b	a = a	b = b	c = c	6, 6 =D
9	~ a = a		~ b = a			6, 7 =D
	×		o			

Recall that the identity sentences on line 8 are officially required on completed open branches. At line 9 the leftmost branch closes owing to the sentence just entered on that branch, while the third branch becomes a completed open branch. The following tree shows that the sentence '(∀x)(∀y)y = *f*(*f*(x))' is also not quantificationally false:

1	(∀x)(∀y)y = *f*(*f*(x))			SM
2	(∀y)y = *f*(*f*(a))			1 ∀D
3	a = *f*(*f*(a))			2 ∀D
4	a = *f*(a)		b = *f*(a)	3 CTD
5	|		a = *f*(b)	3, 4 =D
6	a = a		b = b	4, 4 =D
	o			

Here, when we applied Identity Decomposition at line 5 using the sentences from lines 3 and 4, we did not add 'a = *f*(a)' to the left branch since it already occurred on that branch. The left branch (but not the right one) became a completed open branch after adding the final sentence at line 6.

As a final example we shall construct a substantially more complicated systematic tree. This tree establishes that the sentence '(∃x)(∃y)H*g*(x,y) ≡ (∃x)(∃y)H*g*(y,x)' is quantificationally true:


1 ~[(∃x)(∃y)Hg(x,y) ≡ (∃x)(∃y)Hg(y,x)] SM
2 (∃x)(∃y)Hg(x,y) 1 ≡D
3 ~(∃x)(∃y)Hg(y,x) 1 ≡D
4 (∃y)Hg(a,y) 2 ∃D2
5 (∀x) ~ (∃y)Hg(y,x) 3 ∃D2
6 3 ~ ∃D
7 2 ~ ∃D
8 Hg(a,a) Hg(a,b) 4 ∃D2
9 5 ∃D2
10 ~(∃y)Hg(y,a) ~(∃y)Hg(y,a) 6 ∀D
11 ~(∃y)Hg(y,b) 6 ∀D
12 7 ∀D
13 7 ∀D
14 a = g(a,a) b = g(a,a) a = g(a,b) b = g(a,b) c = g(a,b) 8 CTD
15 9 CTD
16 Ha Hb Ha Hb Hc 8, 14 =D
17 9, 15 =D
18 a = a b = b a = a b = b c = c 14, 14 =D
19 15, 15 =D
20 (∀x) ~ Hg(y,a) (∀y) ~ Hg(y,a) (∀y) ~ Hg(y,a) (∀y) ~ Hg(y,a) (∀y) ~ Hg(y,a) 10 ~ ∃D
21 (∀y) ~ Hg(y,b) (∀y) ~ Hg(y,b) (∀y) ~ Hg(y,b) 11 ~ ∃D
22 12 ~ ∃D
23 13 ~ ∃D
24 ~ Hg(a,a) ~ Hg(a,a) 20 ∀D
25 × × ~ Hg(a,b) ~ Hg(a,b) ~ Hg(a,b) 21 ∀D
26 × × × 22 ∀D
27 23 ∀D
~(∃x)(∃y)Hg(x,y)
(∃x)(∃y)Hg(y,x)
Hg(a,a) Hg(b,a)
~(∃y)Hg(a,y)
~(∃y)Hg(a,y)
~(∃y)Hg(b,y)
a = g(a,a) b = g(a,a) a = g(a,b) b = g(a,b) c = g(a,b)
Ha Hb Ha Hb Hc
a = a b = b a = a b = b c = c
(∀y) ~ Hg(a,y) (∀y) ~ Hg(a,y) (∀y) ~ Hg(a,y) (∀y) ~ Hg(a,y) (∀y) ~ Hg(a,y)
(∀y) ~ Hg(b,y) (∀y) ~ Hg(b,y) (∀y) ~ Hg(b,y)
~ Hg(a,a) ~ Hg(a,a) ~ Hg(b,a) ~ Hg(b,a) ~ Hg(b,a)
× × × × ×

- We begin work on the tree by decomposing all the truth-functionally compound sentences and existentially quantified sentences on the tree, and all such sentences that are added through those decompositions. This Stage 1 work takes us through line 9 of the tree.
- Next, we move to Stage 2 of The System and decompose all universally quantified sentences on each branch using constants already on the branch. This takes us through line 13
- Then we move to Stage 3 and apply Complex Term Decomposition to all complex terms that occur in literals. We complete this stage at line 15.
- Moving to Stage 4, we apply Identity Decomposition as directed at lines 16 through 19.
- At this point we have moved through The System once, but the tree has not closed and does not have a completed open branch.
- So we return, as instructed, to Stage 1. We apply Negated Existential Decomposition, which yields lines 20–23.
- We then move to Stage 2 and apply Universal Decomposition, which yields lines 24–27. After these four applications of Universal Decomposition, every branch is closed and we have a closed tree.

Note that The System specifies, at Stage 3, that we apply Complex Term Decomposition. We did so in this tree, even though the results (the introduction of the literals 'Ha', 'Hb', and 'Hc' on appropriate branches) play no role in closing any branch. In systematic trees, it is common for a tree to contain entries that are not germane to the final result. This is the price we pay for ensuring that we explore all possibilities.

9.6**E** EXERCISES

1. Construct systematic trees to determine, for each of the following sets, whether that set is quantificationally consistent. State your result. If you abandon a tree, explain why.

a. $\{(\forall x)(\forall y)[\sim x = g(y) \supset Gxy], \sim (\exists x)Gax\}$
*b. $\{(\forall x)(Gx \supset Gh(x)), (\exists x)(Gx \,\&\, \sim Gh(x))\}$
c. $\{(\exists x)(\exists y)Hf(x,y), \sim (\exists x)Hx\}$
*d. $\{(\exists x)(\forall y)\ x = f(y), (\exists x)(\forall y) \sim x = f(y)\}$
e. $(\forall x)\ Lxf(x), (\exists y) \sim Lf(y)y\}$
*f. $\{(\forall x) \sim x = f(x), (\exists x)\ x = f(f(x))\}$
g. $\{(\forall x)(Gx \supset \sim Gh(x)), (\exists x)(\sim Gx \,\&\, \sim Gh(x))\}$
*h. $\{(\forall x)\ x = f(x), (\exists x) \sim x = f(f(x))\}$

2. Show that the following sentences are not quantificationally true by constructing an appropriate systematic truth-tree.

a. $(\forall x)(Pf(x) \supset Px)$
*b. $(\forall x)(\forall y)(x = g(y) \vee y = g(x))$
c. $(\exists x)(\forall y)x = g(y)$
*d. $(\forall x) \sim x = f(x)$
e. $(\forall x)(\forall y)(Dh(x,y) \supset Dh(y,x))$
*f. $(\exists x)(\exists y) \sim (x = f(y) \vee y = f(x))$

3. Show that the following sentences are quantificationally true by constructing an appropriate systematic tree.

a. $(\forall x)(\exists y)y = f(f(x))$
*b. $(\forall x)(\forall y)(\forall z)((y = f(x) \ \& \ z = f(x)) \supset y = z)$
c. $(\forall x)Lf(x) \supset (\forall x)Lf(f(x))$
*d. $(\exists y)y = g(f(a))$

4. Construct systematic trees to determine, for each of the following sentences, whether that sentence is quantificationally true, quantificationally false, or quantificationally indeterminate. In each case state your result. If you abandon a tree, explain why.

a. $(\exists x)f(x) = x$
*b. $(\forall x)Gf(x)x$
c. $(\forall x)(\exists y)y = f(f(x))$
*d. $(\forall x)(Fx \vee \sim Fg(x))$

5. Construct systematic trees to determine which of the following arguments are quantificationally valid. In each case state your result. If you abandon a tree, explain why.

a. $(\exists x)(Fg(x) \ \& \sim Hg(x))$
$(\forall x)(Fx \supset Hx)$

$\sim Ra$

c. $a = f(b) \ \& \ b = f(a)$

$(\exists x)(\sim x = a \ \& \sim x = b)$

*b. $(\forall x) \ Pf(f(x))$

$Pf(a)$

*d. $(\forall x)(Fx \vee Fg(x))$

$(\forall x)(Fx \vee Fg(g(x)))$

6. Construct systematic trees to determine which of the following pairs of sentences are quantificationally equivalent. In each case state your result. If you abandon a tree, explain why.

a.	$(\forall x)(\exists y)y = f(x)$	$(\forall x)(\exists y)x = f(y)$
*b.	$Labf(b)$	$Laf(b)b$
c.	$(\exists x)x = x$	$(\exists x)x = f(x)$
*d.	$(\forall x)Bh(x)x$	$(\forall x)Bxh(x)$

7. Construct systematic trees to determine which of the following alleged entailments hold. In each case state your result. If you abandon a tree, explain why.

a. $\{(\forall x)(\forall y)x = g(x,y)\} \models (\forall x)x = g(x,x)$
*b. $\{(\exists x)(\forall y) \ x = g(y)\} \models h(a) = g(a)$
c. $\{(\forall x)x = f(f(x))\} \models (\forall x)x = f(x)$
*d. $\{(\forall x)x = f(x)\} \models (\forall x)x = f(f(x))$

GLOSSARY

Core Logical Concepts

QUANTIFICATIONAL INCONSISTENCY: A finite set Γ of sentences of *PL/PLE* is *quantificationally inconsistent* if and only if Γ has a closed truth-tree.

QUANTIFICATIONAL CONSISTENCY: A finite set Γ of sentences of *PL/PLE* is *quantificationally consistent* if and only if Γ does not have a closed truth-tree.

QUANTIFICATIONAL TRUTH: A sentence **P** of *PL/PLE* is *quantificationally true* if and only if {~ **P**} has a closed truth-tree.

QUANTIFICATIONAL FALSITY: A sentence **P** of *PL/PLE* is *quantificationally false* if and only if {**P**} has a closed truth-tree.

QUANTIFICATIONAL INDETERMINACY: A sentence **P** of *PL/PLE* is *quantificationally indeterminate* if and only if neither {**P**} nor {~ **P**} has a closed truth-tree.

QUANTIFICATIONAL EQUIVALENCE: Sentences **P** and **Q** of *PL/PLE* are *quantificationally equivalent* if and only if {~ (**P** ≡ **Q**)} has a closed truth-tree.

QUANTIFICATIONAL ENTAILMENT: A finite set Γ of sentences of *PL/PLE* *quantificationally entails* a sentence **P** of *PL/PLE* if and only if $\Gamma \cup$ {~ **P**} has a closed truth-tree.

QUANTIFICATIONAL VALIDITY: An argument of *PL/PLE* with a finite number of premises is *quantificationally valid* if and only if the set consisting of the premises and the negation of the conclusion has a closed truth-tree.

Key Truth-Tree Concepts

CLOSED BRANCH OF A TRUTH-TREE FOR A SET OF SENTENCES OF *PL:* A branch containing a contradictory pair of literals.

CLOSED BRANCH OF A TRUTH-TREE FOR A SET OF SENTENCES OF *PLE:* A branch containing a contradictory pair of literals or a sentence of the form ~ **t** = **t**.

CLOSED TRUTH-TREE: A truth-tree each of whose branches is closed.

OPEN BRANCH: A branch that is not closed.

COMPLETED OPEN BRANCH OF A TRUTH-TREE FOR A SET OF SENTENCES OF *PL:* A finite open branch on which each sentence is one of the following:

1. A literal (an atomic sentence or the negation of an atomic sentence)
2. A compound sentence that is not a universally quantified sentence and is decomposed
3. A universally quantified sentence (∀**x**)**P** such that **P**(**a**/**x**) also occurs on that branch *for each constant* **a** occurring on the branch and at least one substitution instance **P**(**a**/**x**) occurs on the branch

(SECTION 9.5 ACCOUNT) COMPLETED OPEN BRANCH OF A TRUTH-TREE FOR A SET OF SENTENCES OF *PLE:* A finite open branch on which each sentence is one of the following:

1. A literal that is not an identity sentence
2. A compound sentence that is not a universally quantified sentence and is decomposed
3. A universally quantified sentence (∀**x**)**P** such that **P**(**t**/**x**) also occurs on that branch *for each closed individual term* **t** occurring on the branch and at least one substitution instance **P**(**t**/**x**) occurs on the branch

4. A sentence of the form $\mathbf{t}_1 = \mathbf{t}_2$, where $\mathbf{t}_1$ and $\mathbf{t}_2$ are closed terms such that the branch also contains, for every literal **P** on that branch containing $\mathbf{t}_2$, every sentence $\mathbf{P}(\mathbf{t}_1//\mathbf{t}_2)$ that can be obtained from **P** by Identity Decomposition

(SECTION 9.6 ACCOUNT) COMPLETED OPEN BRANCH OF A TRUTH-TREE FOR A SET OF SENTENCES OF *PLE:* A finite open branch on which each sentence is one of the following:

1. A literal that is not an identity sentence
2. A nonliteral sentence that is not a universally quantified sentence and is decomposed
3. A universally quantified sentence $(\forall\mathbf{x})\mathbf{P}$ such that $\mathbf{P}(\mathbf{a}/\mathbf{x})$ also occurs on that branch *for each constant* **a** occurring on the branch and $\mathbf{P}(\mathbf{a}/\mathbf{x})$ occurs on the branch for at least one constant **a**
4. A sentence of the form $\mathbf{a} = \mathbf{t}$, where **a** is an individual constant and **t** is a closed term such that the branch also contains, for every literal **P** on that branch containing **t**, every sentence $\mathbf{P}(\mathbf{a}//\mathbf{t})$ that can be obtained from **P** by Identity Decomposition

and on which Complex Term Decomposition has been applied to every closed complex term occurring in a literal on the branch whose arguments are all individual constants.

NONTERMINATING BRANCH: An open branch that never closes and will never, in a finite number of steps, become a completed open branch.

COMPLETED TRUTH-TREE: A truth-tree each of whose branches is either closed or is a completed open branch.

OPEN TRUTH-TREE: A truth-tree that is not closed.

Chapter 10

PREDICATE LOGIC: DERIVATIONS

10.1 THE DERIVATION SYSTEM *PD*

In this chapter we develop natural deduction systems for predicate logic. The first system, *PD* (for *p*redicate *d*erivations), contains exactly two rules for each logical operator, just as *SD* contains exactly two rules for each sentential connective. It provides syntactic methods for evaluating sentences and sets of sentences of *PL,* just as the natural deduction system *SD* provides methods for evaluating sentences and sets of sentences of *SL*. *PD* is both complete and sound: for any set Γ of sentences of *PL* and any sentence **P** of *PL*

$\Gamma \models \mathbf{P}$ if and only if $\Gamma \vdash \mathbf{P}$ in *PD*.

That is, a sentence **P** of *PL* is quantificationally entailed by a set Γ of sentences of *PL* if and only if **P** is derivable from Γ in *PD*. We prove this in Chapter 11.

The derivation rules of *PD* include all the derivation rules of *SD,* with the understanding that they apply to sentences of *PL*. So the following is a derivation in *PD:*

Derive: ~ (∀x)Hx

1	(∀x)Hx ⊃ ~ (∃y)Py	Assumption
2	(∃y)Py	Assumption
3	| (∀x)Hx	A / ~ I
4	| ~ (∃y)Py	1, 3 ⊃E
5	| (∃y)Py	2 R
6	~ (∀x)Hx	3–5 ~ I

The strategies we used with *SD* are also useful when working in *PD*. Those strategies are based on careful analyses of the goal or goals of a derivation—the structure of the sentence or sentences to be derived—and the structure of accessible sentences. They can be summarized thus:

- If the current goal sentence can be obtained by Reiteration, use that rule, otherwise
- If the current goal sentence can be obtained by using a non-subderivation rule, or a series of such rules, do so; otherwise
- Try to obtain the goal sentence by using an appropriate subderivation rule.
- When using a negation rule, try to use an already accessible negation (if there is one) as the ~ **Q** that the negation rules require be derived.

The new rules of *PD* call for some new strategies. We will introduce these as we introduce the new derivation rules of *PD*. *PD* contains four new rules, Universal Elimination, Universal Introduction, Existential Elimination, and Existential Introduction. Each of the new rules involves a quantified sentence and a substitution instance of that sentence. The elimination rule for the universal quantifier is Universal Elimination:

Universal Elimination (∀E)

	$(\forall \mathbf{x})\mathbf{P}$
▷	$\mathbf{P}(\mathbf{a}/\mathbf{x})$

Here we use the expression '$\mathbf{P}(\mathbf{a}/\mathbf{x})$' to stand for a substitution instance of the quantified sentence $(\forall \mathbf{x})\mathbf{P}$. $\mathbf{P}(\mathbf{a}/\mathbf{x})$ is obtained from the quantified sentence by dropping the initial quantifier and replacing every occurrence of **x** with **a**. We will refer to the constant **a** that is substituted for the variable **x** as the **instantiating constant** for the rule ∀E (and similarly for the other rules introduced on the following pages).

Universal Elimination allows us, given a universal generalization, to infer a sentence that says of a particular thing what the given universal generalization says of everything. Consider the following argument:

All philosophers are somewhat strange.

Socrates is a philosopher.

Socrates is somewhat strange.

The first premise makes a universal claim: it says that each thing is such that if it is a philosopher then it is somewhat strange. We can symbolize this claim as '$(\forall y)(Py \supset Sy)$'. The second premise can be symbolized as 'Ps' and the conclusion as 'Ss'. Here is a derivation of the conclusion from the premises.

Derive: Ss

1	(∀y)(Py ⊃ Sy)	Assumption
2	Ps	Assumption
3	Ps ⊃ Ss	1 ∀E
4	Ss	2, 3 ⊃E

The sentence on line 3 is a substitution instance of the quantified sentence on line 1. When we remove the initial (and only) quantifier from '(∀y)(Py ⊃ Sy)' we get the open sentence 'Py ⊃ Sy', which contains two free occurrences of 'y'. Replacing both occurrences with the constant 's' yields the substitution instance 'Ps ⊃ Ss' on line 3, justified by ∀E. We then use Conditional Elimination to obtain 'Ss'.

This simple derivation illustrates the first new strategy for constructing derivations in *PD:*

- When using Universal Elimination use goal sentences as guides to which constant to use in forming the substitution instance of the universally quantified sentence.

At line 3 in the above derivation we could have entered 'Pa ⊃ Sa', or any other substitution instance of '(∀y)(Py ⊃ Sy)'. But obviously only the substitution instance using 's' is of any use in completing the derivation.

The instantiating constant employed in Universal Elimination may or may not already occur in the quantified sentence. The following is a correct use of Universal Elimination:

1	(∀x)Lxa	Assumption
2	Lta	1 ∀E

If we take our one assumption to symbolize 'Everyone loves Alice', with 'a' designating Alice, then clearly it follows that Tom, or whomever t designates, loves Alice. The following is also a correct use of Universal Elimination:

1	(∀x)Lxa	Assumption
2	Laa	1 ∀E

If everyone loves Alice, then it follows that Alice loves Alice, that is, that Alice loves herself.

The introduction rule for existential quantifiers is Existential Introduction:

Existential Introduction (∃I)

	P(a/x)
▷	**(∃x)P**

This rule allows us to infer an existentially quantified sentence from any one of its substitution instances. Here is an example:

1	Fa	Assumption
2	(∃y)Fy	1 ∃I

That Existential Introduction is truth-preserving should also be obvious. If the thing designated by the constant 'a' is F, then at least one thing is F. For example, if Alfred is a father, then it follows that someone is a father.

The following derivation uses Existential Introduction three times:

1	Faa	Assumption
2	(∃y)Fya	1 ∃I
3	(∃y)Fyy	1 ∃I
4	(∃y)Fay	1 ∃I

These uses are all correct because the sentence on line 1 is a substitution instance of the sentence on line 2, and of the sentence on line 3, and of the sentence on line 4. If Alice is fond of herself, then it follows that someone is fond of Alice, that someone is fond of her/himself, and that Alice is fond of someone.

The strategy for using Existential Introduction is straightforward:

- When the goal to be derived is an existentially quantified sentence establish a substitution instance of that sentence as a subgoal, with the intent of applying Existential Introduction to that subgoal to obtain the goal.

The rules Universal Introduction and Existential Elimination are somewhat more complicated. We begin with Universal Introduction:

Universal Introduction (∀I)

	P(**a**/**x**)
▷	(∀**x**)**P**

provided that

(i) **a** does not occur in an open assumption.
(ii) **a** does not occur in (∀**x**)**P**.

Here, again, we will call the constant **a** in **P**(**a**/**x**) the *instantiating constant*. This rule specifies that under certain conditions we can infer a universally quantified sentence from one of its substitution instances. At first glance this might seem implausible, for how can we infer, from a claim that a particular thing is of a certain sort, that *everything* is of that sort? The answer, of course, lies in the restrictions specified in the "provided that" clause.

Here is a very simple example. The sentences '$(\forall x)Fx$' and '$(\forall y)Fy$' are equivalent; they are simply notational variants of each other. They both say that everything is F. So we should be able to derive each from the other. Below we derive the second from the first:

Derive: $(\forall y)Fy$

1	$(\forall x)Fx$	Assumption
2	Fb	1 $\forall$E
3	$(\forall y)Fy$	2 $\forall$I

The use of Universal Introduction at line 3 meets both the restrictions on that rule. The instantiating constant 'b' does not occur in an open assumption and does not occur in the universal generalization entered on line 3.

The kind of reasoning that Universal Introduction is based on is common in mathematics. Suppose we want to establish that no even positive integer greater than 2 is prime. [A prime is a positive integer that is evenly divisible only by itself and 1, and is not 1.] We might reason thus:

> Consider any even positive integer *i* greater than 2. Because *i* is even, *i* must be evenly divisible by 2. But since *i* is not 2 (it is greater than 2), it follows that *i* is evenly divisible by at least three positive integers: 1, 2, and *i* itself. So it is not the case that *i* is evenly divisible only by itself and 1, and *i* cannot be prime. Therefore no even positive integer greater than 2 is prime.

It would exhibit a misunderstanding of this reasoning to reply "but the positive integer *i* you considered might have been 4, and while the reasoning does hold of 4—it is not prime—that fact alone doesn't show that the reasoning holds of every even positive integer greater than 2. You haven't considered 6 and 8 and 10 and. . . ." It would be a misunderstanding because in saying 'Consider any even positive integer *i* greater than 2' we don't mean 'Pick one'. We say 'Consider any even positive integer *i* . . .' because it is easier to construct the argument when we are speaking, grammatically, in the singular ('*i* is . . .', '*i* is not . . .'). But what we are really saying is 'Consider what we know about all positive integers that are even and greater than two . . .' So the proof is a proof about all such integers. Similarly, in derivations we often use an individual constant to reason about all cases of a certain sort.

Suppose we want to establish that '$(\forall x)[Fx \supset (Fx \vee Gx)]$' can be derived from no assumptions. (This will, of course, establish that this sentence of *PL* is a theorem in *PD*.) Here is one such derivation:

Derive: $(\forall x)[Fx \supset (Fx \vee Gx)]$

1	Fc	A / $\supset$I
2	$Fc \vee Gc$	1 $\vee$I
3	$Fc \supset (Fc \vee Gc)$	1–2 $\supset$I
4	$(\forall x)[Fx \supset (Fx \vee Gx)]$	3 $\forall$I

The sentence on line 3 follows from the subderivation on lines 1–2, no matter what the constant 'c' designates. The subderivation establishes that no matter what c is, if it is F then it is F or G. Hence we are justified in deriving the universal quantification on line 4. Note that although 'c' occurs in the assumption on line 1, that assumption is not open at line 4, so we have not run afoul of the first restriction on the rule Universal Introduction.

On the other hand, Universal Introduction is misused in the following attempted derivation:

Derive: $(\forall y)Fy$

1	$Fb \ \& \sim Fc$	Assumption	
2	Fb	1 &E	
3	$(\forall y)Fy$	2 $\forall$I	**MISTAKE**!

'Fb' does follow from line 1. But line 3 does not follow from line 2, and the restriction that the instantiating constant, in this case 'b', not occur in an open assumption prevents us from using Universal Introduction at line 3. (From the fact that Beth is a faculty member and Carl is not it does not follow that everyone is a faculty member.)

The rule Universal Introduction contains a second restriction, namely that the instantiating constant not occur in the derived universally quantified sentence. The following attempt at a derivation illustrates why this restriction is needed:

Derive: $(\forall x)Lxh$

1	$(\forall x)Lxx$	Assumption	
2	Lhh	1 $\forall$E	
3	$(\forall x)Lxh$	2 $\forall$I	**MISTAKE!**

The sentence on line 1 tells us that everying bears L to itself. It certainly follows that h bears L to itself. But it does not follow that everything bears L to h, and the second restriction on Universal Introduction disallows the use of that rule to obtain the sentence on line 3, because the instantiating constant, 'h', does occur in that sentence.

The strategy associated with Universal Introduction is

- When the current goal is a universally quantified sentence make a substitution instance of that quantified sentence a subgoal, with the intent of applying Universal Introduction to derive the goal from the subgoal. Make sure that the two restrictions on Universal Introduction will be met: use an instantiating constant in the substitution instance that does not occur in the universally quantified goal sentence and that does not occur in any assumption that is open at the line where the substitution instance is entered.

Here is the elimination rule for existential quantifiers:

Existential Elimination (∃E)

| (∃**x**)**P**
|
| | **P**(**a**/**x**)
| |---
| | **Q**
▷ | **Q**

provided that

(i) **a** does not occur in an open assumption.
(ii) **a** does not occur in (∃**x**)**P**.
(iii) **a** does not occur in **Q**.

The idea behind this rule is that if we have an existentially quantified sentence (∃**x**)**P** then we know that something is of the sort specified by **P**, though not which thing. If, by assuming an arbitrary substitution instance **P**(**a**/**x**) of (∃**x**)**P**, we can derive a sentence **Q** that does not contain the instantiating constant **a** in **P**(**a**/**x**), then we can end the subderivation and enter **Q** on the next line of the derivation.

We illustrate a simple use of Existential Elimination by deriving '(∃x)(Gx ∨ Fx)' from {(∃z)Fz & (∀y)Hy}.

Derive: (∃x)(Gx ∨ Fx)

1	(∃z)Fz & (∀y)Hy	Assumption
2	(∃z)Fz	1 &E
3	Fb	A / ∃E
4	Gb ∨ Fb	3 ∨I
5	(∃x)(Gx ∨ Fx)	4 ∃I
6	(∃x)(Gx ∨ Fx)	2, 3–5 ∃E

'Existential Elimination' may seem like an odd name for the rule we used at line 6 of the above derivation, because the sentence entered at line 6 is itself an existentially quantified sentence. But remember that what is common to all elimination rules is that they are rules that *start* with a sentence with a specified main logical operator and produce a sentence that may or may not have that operator as a main logical operator. Here Existential Elimination cites the existentially quantified sentence at line 2, along with the subderivation beginning with a substitution instance of that sentence. Note that we have met all the restrictions on Existential Elimination. The instantiating constant 'b' does not occur in an assumption that is open as of line 6. Nor does 'b' occur in '(∃z)Fz'. Finally, 'b' does not occur in the sentence that is derived, at line 6, by Existential Elimination. All three of these restrictions are necessary, as we will now illustrate.

Two specific strategies are associated with the rule Existential Elimination. The first is this:

- When one or more of the currently accessible sentences in a derivation is an existentially quantified sentence, consider using Existential Elimination to obtain the current goal. Assume a substitution instance that contains a constant that does not occur in the existential quantification, in an open assumption, or in the current goal. Work within the Existential Elimination subderivation to derive the current goal.

In other words, whenever an existentially quantified sentence is accessible consider making Existential Elimination the primary strategy for obtaining the current goal, doing the work required to obtain the current goal within the scope of the Existential Elimination subderivation. This is often necessary to avoid violating the restrictions on Existential Elimination. For example, in the previous derivation we had to use Existential Introduction within the scope of the assumption on line 3—because trying to derive '$Gb \vee Fb$' by Existential Elimination at line 5, prior to applying Existential Introduction, would violate the third restriction on Existential Elimination:

Derive: $(\exists x)(Gx \vee Fx)$

1	$(\exists z)Fz \;\&\; (\forall y)Hy$	Assumption	
2	$(\exists z)Fz$	1 &E	
3	$\quad Fb$	A / $\exists$E	
4	$\quad Gb \vee Fb$	3 $\vee$I	
5	$Gb \vee Fb$	2, 3–4 $\exists$E	**MISTAKE!**
6	$(\exists x)(Gx \vee Fx)$	5 $\exists$I	

Line 5 is a mistake because the instantiating constant 'b' occurs in the sentence we are trying to obtain by Existential Elimination, in violation of the third restriction on Existential Elimination. From the truth of '$(\exists z)Fz$' it does not follow that the individual designated by 'b' is either G or F—although it does follow, as in the previous derivation, that *something* is either G or F. This is why, in the correctly done derivation, we used Existential Introduction inside of the Existential Elimination subderivation. Doing so results in a sentence that does not contain the instantiating constant 'b' and that therefore can correctly be moved out of the subderivation by Existential Elimination.

Here is another example in which the third restriction on Existential Elimination is violated:

Derive: $(\exists z)Fbz$

1	$(\exists z)Fzz$	Assumption	
2	$\quad Fbb$	A / $\exists$E	
3	$\quad (\exists z)Fbz$	3 $\exists$I	
4	$(\exists z)Fbz$	1, 2–3 $\exists$E	**MISTAKE!**

The instantiating constant 'b' occurs in the sentence on line 4, in violation of restriction (iii) on Existential Elimination. It is clear that we don't want the above to count as a derivation. Given the assumption on line 1 we know that something bears F to itself. At line 2 we assume that thing is b (knowing that this may not be the case). Line 3 certainly follows from line 2. *If* b bears F to itself then b does bear F to something. But line 4, where we have given up the assumption that it is b that bears F to itself, does not follow from the sentence at line 1, which is the single open assumption as of line 4. Contrast the preceding derivation with the following:

Derive: $(\exists y)Fyy$

1	$(\exists z)Fzz$	Assumption
2	$\quad Fbb$	A / $\exists$E
3	$\quad (\exists y)Fyy$	2 $\exists$I
4	$(\exists y)Fyy$	1, 2–3 $\exists$E

Here 'b' does not occur in the sentence at line 4, so the third restriction on Existential Elimination is not violated. We have used Existential Elimination to show that '$(\exists y)Fyy$' follows from '$(\exists z)Fzz$', which should be no surprise since these sentences are clearly equivalent.

We will now examine some misuses of Existential Elimination that illustrate why the two other restrictions on Existential Elimination are also necessary.

Derive: $(\forall x)Fx$

1	$Gb \supset (\forall x)Fx$	Assumption	
2	$(\exists z)Gz$	Assumption	
3	$\quad Gb$	A / $\exists$E	
4	$\quad (\forall x)Fx$	1, 3 $\supset$E	
5	$(\forall x)Fx$	2, 3–4 $\exists$E	**MISTAKE!**

From line 1 we know that if a particular thing, namely b, is G, then everything is F. And from line 2 we know that something is G. But we do not know that it is b that is G. So we should not be able to infer, as we have here tried to do at line 5, that everything is F. Line 5 is a mistaken application of Existential Elimination because restriction (i) has not been met. The assumption at line 1, which contains the instantiating constant 'b', is still open as of line 5. The rationale for restriction (i) should now be clear. Existential Elimination uses a substitution instance of an existentially quantified claim to show what follows from the existentially quantified claim. But the constant used in the substitution instance, the instantiating constant, should be arbitrary, in the sense that no assumptions have been made concerning the thing designated by that constant. If the instantiating constant occurs in an open assumption then it is not arbitrarily selected, because the open assumption provides information about b (that if it is G everything is F). It may be the case that if Bob graduates then

everyone is happy and the case that someone does graduate. But it does not follow from this that everyone is happy, for the someone who graduates may not be Bob.

We now turn to the rationale for the second restriction. Consider the following attempt at a derivation:

Derive: (∃w)Lww

1	(∀y)(∃z)Lzy	Assumption	
2	(∃z)Lza	1 ∀E	
3	Laa	A / ∃E	
4	(∃w)Lww	3 ∃I	
5	(∃w)Lww	2, 3–4 ∃E	**MISTAKE!**

The problem is that the instantiating constant 'a' used at line 3 to form a substitution instance of the sentence '(∃z)Lza' occurs in '(∃x)Lza', violating the second restriction. If we only know that *something* stands in the relation L to a, we should not assume that that something is in fact a itself.

Universal Elimination produces a substitution of the universally quantified sentence to which it is applied. Existential Elimination does not, in general, produce a substitution instance of the existentially quantified sentence to which it is applied. Indeed the sentence it produces may bear no resemblance, by any normal standard of resemblance, to the existentially quantified sentence to which it is applied. Here is a case in point:

Derive: (∃x)Hx

1	(∃z)Gz	Assumption
2	(∀y)(Gy ⊃ Hc)	Assumption
3	Gb	A / ∃E
4	Gb ⊃ Hc	2 ∀E
5	Hc	3, 4 ⊃E
6	Hc	1, 3–5 ∃E
7	(∃x)Hx	6 ∃I

Here the sentence derived at line 6 has no obvious connection to the existentially quantified sentence at line 1. The existentially quantified sentence tells us that something is G. At line 3 we assume that that thing is b. The constant 'b' is not used earlier in the derivation, so we are committed to nothing about b other than its being G. At line 4 we use Universal Elimination to obtain 'Gb ⊃ Hc', and then we use Conditional Elimination at line 5 to obtain 'Hc'. At the point we apply Existential Elimination (line 6) there is here no open assumption that contains 'b'—the only open assumptions are those on line 1 and line 2—so the first restriction on Existential Elimination is met. The second and third restrictions are also met since 'b' occurs in neither '(∃z)Gz' nor 'Hc'. We can, therefore, derive 'Hc' by Existential Elimination at line 6.

Note that in this case we were able to move 'Hc' out of the Existential Elimination subderivation prior to using Existential Introduction. We could do this because 'c' was not the instantiating constant for our use of Existential Elimination. However, we could also have applied Existential Introduction within the subderivation;

Derive: $(\exists x)Hx$

1	$(\exists z)Gz$	Assumption
2	$(\forall y)(Gy \supset Hc)$	Assumption
3	$\quad Gb$	A / $\exists$E
4	$\quad Gb \supset Hc$	2 $\forall$E
5	$\quad Hc$	3, 4 $\supset$E
6	$\quad (\exists x)Hx$	5 $\exists$I
7	$(\exists x)Hx$	1, 3–6 $\exists$E

Existential Elimination provides a strategy for working from a substitution instance of an existentially quantified sentence to a sentence that does not contain the instantiating constant of the substitution instance. If the other restrictions on Existential Elimination are also met the subderivation can be ended and the derived sentence entered on the next line of the derivation.

There is a second important strategy associated with Existential Elimination. We will use it to show that the set $\{(\exists x) \sim Fx, (\forall x)Fx\}$ is inconsistent in *PD*. The foregoing set obviously is inconsistent, but demonstrating this is not as easy as it might seem. We might start as follows:

Derive: Fa, ~ Fa

1	$(\exists x) \sim Fx$	Assumption	
2	$(\forall x)Fx$	Assumption	
3	$\quad Fa$	2 $\forall$E	
4	$\quad \sim Fa$	1 $\exists$E	**MISTAKE!**

Line 4 is an obvious misuse of Existential Elimination. A more promising approach might be as follows:

Derive: Fa, ~ Fa

1	$(\exists x) \sim Fx$	Assumption
2	$(\forall x)Fx$	Assumption
3	$\quad \sim Fa$	A / $\exists$E
4	$\quad Fa$	2 $\forall$E
5	$\quad \sim Fa$	3 R

We have derived a sentence and its negation ('Fa' and '~ Fa'), but only within the scope of our Existential Elimination subderivation. And since 'a' is the instantiating constant of the assumption at line 3, we cannot hope to move either 'Fa' or '~ Fa' out from the scope of the assumption at line 3 by Existential Elimination. The

situation we are in is not an uncommon one. We need to use Existential Elimination, and we can derive a contradiction within the Existential Elimination subderivation, but the contradictory sentences we derive cannot be moved outside that subderivation because they contain the instantiating constant of the assumption.

The strategy we will use in situations such as this makes use of the fact that we can derive contradictory sentences within the Existential Elimination subderivation. Since we can do this, we can also derive any sentence we want by use of the appropriate negation rule. In our present case we want to derive a sentence and its negation, to show that the set we are working from is inconsistent in *PD*. There are no negations among our primary assumptions. We know taking 'Fa' and '$\sim Fa$' as our ultimate goals will not work (so long as '$\sim Fa$' remains as our Existential Elimination assumption at line 3). So we will take a sentence that is accessable, '$(\forall x)Fx$', and its negation as our ultimate goals, and we will derive '$\sim (\forall x)Fx$' by Negation Introduction within our Existential Elimination subderivation, and then move it out of that subderivation by Existential Elimination:

Derive: $(\forall x)Fx$, $\sim (\forall x)Fx$

1	$(\exists x) \sim Fx$	Assumption
2	$(\forall x)Fx$	Assumption
3	$\sim Fa$	A / $\exists$E
4	$(\forall x)Fx$	A / $\sim$ I
5	Fa	4 $\forall$E
6	$\sim Fa$	3 R
7	$\sim (\forall x)Fx$	4–6 $\sim$ I
8	$\sim (\forall x)Fx$	1, 3–7 $\exists$E
9	$(\forall x)Fx$	2 R

What may strike one as odd about this derivation is that we are assuming, at line 4, a sentence that is already accessible (as the assumption on line 2). But the point of making this assumption of a sentence we already have is to derive its negation, which we do at line 7. Negation Introduction requires us to assume this sentence, even though it also occurs at line 2, before we can apply that rule. At line 4 we could, of course, have equally well assumed '$(\exists x) \sim Fx$', in which case our ultimate goals would have been '$(\exists x) \sim Fx$' and '$\sim (\exists x) \sim Fx$'.

The strategy we are illustrating can be put thus:

- When contradictory sentences are available within an Existential Elimination subderivation but cannot be moved out of that subderivation without violating the restrictions on Existential Elimination, derive another sentence—one that is contradictory to a sentence accessible outside the Existential Elimination subderivation and one that can be moved out. That sentence will be derivable by the appropriate negation strategy (because contradictory sentences are available within the Existential Elimination subderivation).

Using this strategy will frequently involve assuming, as the assumption of a negation strategy, a sentence that is already accessible outside the Existential Elimination subderivation.

Consider next the following failed attempt at a derivation of '~ (∃x)Fx' from {~ (∃x)Fx}:

Derive: ~ (∃x)Fx

1	(∀x) ~ Fx	Assumption	
2	(∃x)Fx	A / ~ I	
3	Fa	A / ∃E	
4	~ Fa	1 ∀E	
5	Fa	3 R	
6	~ (∃x)Fx	3–5 ~ I	**MISTAKE!**
7	~ (∃x)Fx	2, 3–6 ∃E	**MISTAKE!**

We are trying to derive a negation, '~ (∃x)Fx', and so assume '(∃x)Fx' at line 2. Clearly an Existential Elimination strategy is now called for, and accordingly we assume 'Fa' at line 3. It is now easy to derive the contradictory sentences 'Fa' and '~ Fa', and we do so at lines 4 and 5. But line 6 is a mistake. Our primary strategy is Negation Introduction and we have derived a sentence and its negation; but we have done so only within the scope of an additional assumption, the one at line 3 that begins our Existential Elimination strategy. Line 6 is a mistake because 'Fa' and '~ Fa' have been derived, not from just the assumptions on lines 1 and 2, but also using the assumption on line 3. We need to complete our Existential Elimination strategy before using Negation Introduction. And what we want our Existential Elimination strategy to yield is a sentence that can serve as one of the contradictory sentences we need to complete the Negation Introduction subderivation we began at line 2.

Two sentences are accessible outside our Existential Elimination subderivation—those on lines 1 and 2 ('(∀x) ~ Fx' and '(∃x)Fx') and obtaining the negation of either one of these by Existental Elimination will allow us to complete the derivation. Here is a successful derivation in which we derive '~ (∀x) ~ Fx' by Existential Elimination.

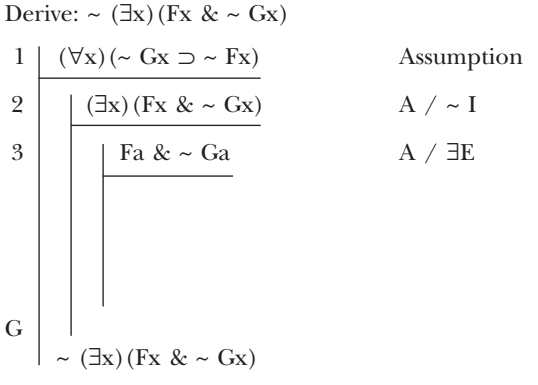
Derive: ~ (∃x)Fx

1	(∀x) ~ Fx	Assumption
2	(∃x)Fx	A / ~ I
3	Fa	A / ∃E
4	(∀x) ~ Fx	A / ~ I
5	~ Fa	1 ∀E
6	Fa	3 R
7	~ (∀x) ~ Fx	4–6 ~ I
8	~ (∀x) ~ Fx	2, 3–7 ∃E
9	(∀x) ~ Fx	1 R
10	~ (∃x)Fx	2–9 ~ I

After making the assumption at line 3 we realize we can derive the contradictory sentences 'Fa' and '~ Fa'. Because we want to obtain '~ (∀x) ~ Fx' by Existential Elimination, we assume '(∀x) ~ Fx' at line 4 and derive '~ Fa' and 'Fa' within the scope of that assumption, allowing us to then derive '~ (∀x) ~ Fx' by Negation Introduction.

Alternatively, we could have used '(∃x)Fx' as an assumption at line 4, derived 'Fa' and '~ Fa', obtained '~ (∃x)Fx' by Negation Elimination, moved that sentence out of the scope of the assumption made at line 3 by Existential Elimination, and then reiterated '(∃x)Fx' within the scope of the assumption '(∃x)Fx' so as to have the contradictory sentences we need to finish the derivation with Negation Introduction. Note also that the assumption at line 4 is necessary to obtain its negation even though the sentence we assume is already available as an earlier assumption (on line 1). As noted earlier this process of making an assumption of a sentence that is already available outside the scope of an Existential Elimination strategy within that strategy *in order to obtain its negation* is extremely useful and frequently called for, as we will see in examples and exercises later in this chapter.

As another example, suppose we want to derive '~ (∃x) (Fx & ~ Gx)' from {(∀x) (~ Gx ⊃ ~ Fx)}. Since our primary goal is a negation, we plan to use Negation Introduction, and since the assumption of that strategy will be an existentially quantified sentence, we will use Existential Elimination within the Negation Introduction subderivation:



Following our new strategy we will begin a Negation Introduction subderivation inside of the Existential Elimination subderivation, assuming one of the sentences that is accessible from outside of that subderivation. In this example there are again two such sentences, '(∀x) (~ Gx ⊃ ~ Fx)' and '(∃x) (Fx & ~ Gx)'. We arbitrarily select the latter as the assumption of the inner Negation Introduction

subderivation and complete the derivation as follows:

Derive: ~ (∃x) (Fx & ~ Gx)

1	(∀x) (~ Gx ⊃ ~ Fx)	Assumption
2	(∃x) (Fx & ~ Gx)	A / ~ I
3	Fa & ~ Ga	A / ∃E
4	(∃x) (Fx & ~ Gx)	A / ~ I
5	~ Ga ⊃ ~ Fa	1 ∀E
6	~ Ga	3 &E
7	~ Fa	5, 6 ⊃E
8	Fa	3 &E
9	~ (∃x) (Fx & ~ Gx)	4–8 ~ I
10	~ (∃x) (Fx & ~ Gx)	2, 3–9 ∃E
11	(∃x) (Fx & ~ Gx)	2 R
12	~ (∃x) (Fx & ~ Gx)	2–11 ~ I

Although the assumption at line 4 is an existentially quantified sentence, there is no need for a second use of Existential Elimination. We can derive the contradictory pair of sentences 'Fa' and '~ Fa' without making any additional assumptions.

We have specified strategies for using each of the four new quantifier rules. Now that we have introduced all the rules of *PD* a note about applying those rules is in order. The quantifier introduction and elimination rules, like all the rules of *PD,* are *rules of inference.* That is, they apply only to whole sentences, *not* to subsentential components of sentences that may or may not themselves be sentences. The only sentences that quantifier elimination rules can be applied to are sentences whose main logical operators are quantifiers. Moreover, the quantifier *introduction* rules generate only sentences whose main logical operators are quantifiers. The following examples illustrate some common types of mistakes that ignore these points about the quantifier rules of *PD.*

Derive: Fa ⊃ Ha

1	(∀x)Fx ⊃ Ha	Assumption	
2	Fa ⊃ Ha	1 ∀E	**MISTAKE!**

The sentence on line 1 is not a universally quantified sentence. Rather, it is a material conditional, so Universal Elimination cannot be applied to it. Obviously, the sentence on line 2 does not follow from the sentence on line 1. From that fact that if everything is F then a is H it does not follow that if a (which is only one thing) is F then a is H.

Here is another example illustrating a similar mistake:

Derive: Ga

1	Fa	Assumption	
2	$(\forall x)(Fx \supset (\forall y)Gy)$	Assumption	
3	$(\forall x)(Fx \supset Ga)$	2 $\forall$E	**MISTAKE!**
4	$Fa \supset Ga$	3 $\forall$E	
5	Ga	1, 4 $\supset$E	

Line 3 is a mistake even though the sentence it cites, '$(\forall x)(Fx \supset (\forall y)Gy)$', is a universally quantified sentence. It is a mistake because it attempts to apply Universal Elimination to '$(\forall y)Gy$', which occurs only as a component of the sentence on line 2. Rules of inference can only be applied to sentences that are not components of larger sentences. Universal Elimination can only produce a substitution instance, for example '$Fa \supset (\forall y)Gy$', of the entire sentence on line 2.

We hasten to add that it *is* possible to derive 'Ga' from the sentences on lines 1 and 2 but a different strategy is required:

Derive: Ga

1	Fa	Assumption
2	$(\forall x)(Fx \supset (\forall y)Gy)$	Assumption
3	$Fa \supset (\forall y)Gy$	2 $\forall$E
4	$(\forall y)Gy$	1, 3 $\supset$E
5	Ga	4 $\forall$E

Here Universal Elimination has only been applied to entire sentences occurring on earlier lines.

The following also illustrates a misuse of a quantifier rule:

Derive: $(\exists z)Fz \supset Gb$

1	$Fa \supset Gb$	Assumption	
2	$(\exists z)Fz \supset Gb$	1 $\exists$I	**MISTAKE!**

Existential Introduction produces existentially quantified sentences, and the sentence on line 2 is a material conditional, not an existentially quantified sentence. Nor do we want to be able to derive the sentence on line 2 from the sentence on line 1. From 'If Alfred wins the election then Bob will be happy' it does not follow that if *someone* wins the election then Bob will be happy. A correct use of the rule would be

Derive: $(\exists z)(Fz \supset Gb)$

1	$Fa \supset Gb$	Assumption	
2	$(\exists z)(Fz \supset Gb)$	1 $\exists$I	

In the following failed derivation, the use of Universal Elimination is incorrect because the sentence on line 1 is not a universally quantified sentence. Rather, it is the *negation* of a universally quantified sentence:

Derive: ~ Fb

1	~ (∀y)Fy	Assumption	
2	~ Fb	1 ∀E	**MISTAKE!**

Having introduced all the rules of *PD* we can now define syntactic analogues of core logical concepts for *PD:*

Derivability in PD: A sentence **P** of *PL* is *derivable in PD* from a set Γ of sentences of *PL* if and only if there is a derivation in *PD* in which all the primary assumptions are members of Γ and **P** occurs within the scope of only the primary assumptions.

Validity in PD: An argument of *PL* is *valid in PD* if and only if the conclusion of the argument is derivable in *PD* from the set consisting of the premises. An argument of *PL* is *invalid in PD* if and only if it is not valid in *PD*.

Theorem in PD: A sentence **P** of *PL* is a *theorem in PD* if and only if **P** is derivable in *PD* from the empty set.

Equivalence in PD: Sentences **P** and **Q** of *PL* are *equivalent in PD* if and only if **Q** is derivable in *PD* from {**P**} and **P** is derivable in *PD* from {**Q**}.

Inconsistency in PD: A set Γ of sentences of *PL* is *inconsistent in PD* if and only if there is a sentence **P** such that both **P** and ~ **P** are derivable in *PD* from Γ. A set Γ is *consistent in PD* if and only if it is not inconsistent in *PD*.

10.1**E** EXERCISES

1. Construct derivations that establish the following claims:

a. {(∀x)Fx} ⊢ (∀y)Fy
*b. {Fb, Gb} ⊢ (∃x)(Fx & Gx)
c. {(∀x)(∀y)Hxy} ⊢ (∃x)(∃y)Hxy
*d. {(∃x)(Fx & Gx)} ⊢ (∃y)Fy & (∃w)Gw
e. {(∀x)(∀y)Hxy, Hab ⊃ Kg} ⊢ Kg
*f. {(∀x)(Fx ≡ Gx), (∀y)(Gy ≡ Hy)} ⊢ (∀x)(Fx ≡ Hx)
g. {(∀x)Sx, (∃y)Sy ⊃ (∀w)Ww} ⊢ (∃y)Wy
*h. {(∀y)Hyy, (∃z)Bz} ⊢ (∃x)(Bx & Hxx)
i. {(∀x)(∀y)Lxy, (∃w)Hww} ⊢ (∃x)(Lxx & Hxx)
*j. {(∀x)(Fx ⊃ Lx), (∃y)Fy} ⊢ (∃x)Lx

2. Identify the mistake in each of the following attempted derivations, and explain why it is a mistake.

a. Derive: Na

1	(∀x)Hx ⊃ ~ (∃y)Ky	Assumption
2	Ha ⊃ Na	Assumption
3	Ha	1 ∀E
4	Na	2, 3 ⊃E

*b. Derive: (∀x) (Bx & Mx)

1	Bk	Assumption
2	(∀x)Mx	Assumption
3	Mk	2 ∀E
4	Bk & Mk	1, 3 &I
5	(∀x) (Bx & Mx)	4 ∀I

c. Derive: (∃x)Cx

1	(∃y)Fy	Assumption
2	(∀w) (Fw ≡ Cw)	Assumption
3	Fa	1 ∃E
4	Fa ≡ Ca	2 ∀∃E
5	Ca	3, 4 ≡E
6	(∃x)Cx	5 ∃I

*d Derive: (∃z)Gz

1	(∀x) (Fx ⊃ Gx)	Assumption
2	(∃y)Fy	Assumption
3	Fa	A / ∃E
4	Fa ⊃ Ga	1 ∀E
5	Ga	3, 4 ⊃E
6	Ga	2, 3–5 ∃E
7	(∃z)Gz	6 ∃I

e. Derive: (∃y) (∀x)Ayx

1	(∀x) (∃y)Ayx	Assumption
2	(∀x)Aax	1 ∀E
3	(∃y) (∀x)Ayx	2 ∃I

*f. Derive: ~ Rba

1	(∃x)Rxx	Assumption
2	(∀x) (∀y) (Rxy ⊃ ~ Ryx)	Assumption
3	Raa	A / ∃E
4	(∀y) (Ray ⊃ ~ Rya)	2 ∀E
5	Raa ⊃ ~ Raa	2 ∀E
6	~ Raa	3, 5 ⊃E
7	(∀x) ~ Rxx	6 ∀I
8	(∀x) ~ Rxx	1, 3–7 ∃E

10.2 USING DERIVATIONS TO ESTABLISH SYNTACTIC PROPERTIES OF *PD*

In this section we will work through a series of derivations, illustrating both strategies that are useful in constructing derivations in *PD* and how derivations are used to establish that various syntactic properties of *PD* hold of sentences and sets of sentences of *PL*.

We begin by repeating the strategies we have enumerated as useful in constructing derivations:

- If the current goal sentence can be obtained by Reiteration, use that rule, otherwise
- If the current goal sentence can be obtained by using a non-subderivation rule, or a series of such rules, do so; otherwise
- Try to obtain the goal sentence by using an appropriate subderivation rule.
- When using a negation rule, try to use an already accessible negation (if there is one) as the ~ **Q** that the negation rules require be derived.
- When using Universal Elimination use goal sentences as guides when choosing the instantiating constant.
- When the goal to be derived is an existentially quantified sentence make a substitution instance of that sentence a subgoal, with the intent of applying Existential Introduction to that subgoal to obtain the goal.
- When the current goal is a universally quantified sentence make a substitution instance of that quantified sentence a subgoal, with the intent of applying Universal Introduction to that subgoal. Make sure the two restrictions on the instantiating constant for the use of Universal Introduction are met. Be sure to choose an instantiating constant that does not occur in the universally quantified sentence that is the goal and that does not occur in any assumption that will be open when Universal Introduction is applied to derive that goal.
- When one of the accessible assumptions is an existentially quantified sentence, consider using Existential Elimination to obtain the current goal. Set up an Existential Elimination subderivation, and continue working within that subderivation until a sentence that does not contain the constant used to form the substitution instance that is the assumption of that subderivation is derived.
- When contradictory sentences are available within an Existential Elimination subderivation but cannot be moved out of that subderivation without violating the restrictions on Existential Elimination, derive another sentence—one that is contradictory to a sentence

accessible outside the Existential Elimination subderivation and that does not contain the instantiating constant for this use of Existential Elimination. That sentence will be derivable by the appropriate negation strategy (using the contradictory sentences that are available within the Existential Elimination subderivation).

- There will often be more than one plausible strategy, and often more than one will lead to success. Rather than trying to figure out which of these is the most promising it is often wise to just pick one and pursue it.

ARGUMENTS

An argument of *PL* is valid in *PD* if and only if the conclusion can be derived from the set consisting of the argument's premises. The following argument is valid in *PD,* as we will now show.

$$\frac{(\exists x)(Fx \ \& \ Gx)}{(\exists y)Fy \ \& \ (\exists z)Gz}$$

The single premise is an existentially quantified sentence—which suggests using Existential Elimination. The conclusion is a conjunction, suggesting Conjunction Introduction as a strategy. We will use both strategies, and since it is in general wise to do as much work as possible within an Existential Elimination strategy (so as to avoid violating the third restriction on Existential Elimination), we will make that strategy our primary strategy. We begin as follows:

Derive: $(\exists y)Fy \ \& \ (\exists z)Gz$

1	$(\exists x)(Fx \ \& \ Gx)$	Assumption
2	$Fb \ \& \ Gb$	A / $\exists$E
G	$(\exists y)Fy \ \& \ (\exists z)Gz$	—, — &I
G	$(\exists y)Fy \ \& \ (\exists z)Gz$	1, 2–— $\exists$E

We will try to derive the conclusion of the argument *within* the scope of the Existential Elimination subderivation because doing so will avoid violating the third restriction on Existential Elimination, that the instantiating constant not occur in the derived sentence. In our derivation 'a' is the instantiating constant and it does not occur in the conclusion of the argument.

Our current goal is a conjunction and can be obtained by Conjunction Introduction. The completed derivation is

Derive: $(\exists y)Fy \;\&\; (\exists z)Gz$

1	$(\exists x)(Fx \;\&\; Gx)$	Assumption
2	$\quad Fa \;\&\; Ga$	A / $\exists$E
3	$\quad Fa$	2 &E
4	$\quad (\exists y)Fy$	3 $\exists$I
5	$\quad Ga$	2 &E
6	$\quad (\exists z)Gz$	5 $\exists$I
7	$\quad (\exists y)Fy \;\&\; (\exists z)Gz$	4, 6 &I
8	$(\exists y)Fy \;\&\; (\exists z)Gz$	1, 2–7 $\exists$E

The following argument is also valid in *PD:*

$(\forall x)(Nx \supset Ox)$
$\sim (\exists y)Oy$

$\sim (\exists x)Nx$

Since the conclusion of this argument is a negation we will use Negation Introduction as our primary strategy and we will try to derive both '$(\exists y)Oy$' and '$\sim (\exists y)Oy$' within a Negation Introduction subderivation:

Derive: $\sim (\exists x)Nx$

1	$(\forall x)(Nx \supset Ox)$	Assumption
2	$\sim (\exists y)Oy$	Assumption
3	$\quad (\exists x)Nx$	A / $\sim$ E
G	$\quad (\exists y)Oy$	
	$\quad \sim (\exists y)Oy$	2 R
G	$\sim (\exists x)Nx$	3–__ $\sim$ I

Since one of the accessible sentences, '$(\exists x)Nx$' is an existentially quantified sentence, we will try to obtain our current goal, '$(\exists y)Oy$', by Existential Elimination:

Derive: $\sim (\exists x)Nx$

1	$(\forall x)(Nx \supset Ox)$	Assumption
2	$\sim (\exists y)Oy$	Assumption
3	$\quad (\exists x)Nx$	A / $\sim$ E
4	$\quad\quad Na$	A / $\exists$E
G	$\quad\quad (\exists y)Oy$	
G	$\quad (\exists y)Oy$	3, 4–__ $\exists$E
G	$\quad \sim (\exists y)Oy$	2 R
G	$\sim (\exists x)Nx$	3–__ $\sim$ I

Looking at the sentences on lines 1 and 4, we see that we will be able to derive 'Oa' by Conditional Elimination after applying Universal Elimination to the sentence on line 1, with 'a' as the instantiating constant. And from 'Oa' we can obtain '(∃y)Oy' by Existential Introduction. So the completed derivation is

Derive: ~ (∃x)Nx

1	(∀x)(Nx ⊃ Ox)	Assumption
2	~ (∃y)Oy	Assumption
3	| (∃x)Nx	A / ~ E
4	| | Na	A / ∃E
5	| | Na ⊃ Oa	1 ∀E
6	| | Oa	4, 5 ⊃E
7	| | (∃y)Oy	6 ∃I
8	| (∃y)Oy	3, 4–7 ∃E
9	| ~ (∃y)Oy	2 R
10	~ (∃x)Nx	3–9 ~ I

We will next consider two arguments, both of which involve relational predicates and quantifiers with overlapping scope. The first is

(∀x)(∀y)(Hxy ⊃ ~ Hyx)
(∀x)(∃y)Hxy

(∀x)(∃y) ~ Hxy

Here our assumptions and our goal sentence are all universally quantified sentences. So we will clearly be using Universal Elimination and Universal Introduction. Using Universal elimination on the second premise will result in an existentially quantified sentence, '(∃y)Hay', which suggests using Existential Elimination:

Derive: (∀x)(∃y) ~ Hxy

1	(∀x)(∀y)(Hxy ⊃ ~ Hyx)	Assumption
2	(∀x)(∃y)Hxy	Assumption
3	(∃y)Hay	2 ∀E
4	| Hab	A / ∃E
	|	
	|	
	|	
G	(∀x)(∃y) ~ Hyx	3, 4–__ ∃E

On line 4 we chose an instantiating constant that does not appear earlier in the derivation, so that the restrictions on the instantiating constant can be met. Clearly at some point we will obtain '(∀x)(∃y) ~ Hyx' by Universal Introduction. The question is whether we will use Universal Introduction before or after ending our Existential Elimination subderivation. We have stressed in earlier examples that it is generally wise to do as much work as possible within Existential Elimination subderivations. This might suggest that we try to obtain '(∀x)(∃y) ~ Hyx' within our Existential Elimination subderivation. But this is, in the present context, a bad idea. The substitution instance of '(∀x)(∃y) ~ Hyx' we will be able to obtain is '(∃y) ~ Hya', in which 'a' is the instantiating constant. The first restriction on Universal Introduction requires that the instantiating constant not occur in any open assumption. But 'a' does occur in 'Hab', the assumption on line 4. So we cannot apply Universal Introduction within the scope of that assumption.

A strategy that will work is to obtain '(∃y) ~ Hya' by Existential Elimination and then, after the assumption 'Hab' is discharged, to apply Universal Introduction. Note that our advice—to do as much work within Existential Elimination subderivations as possible—still holds. The current case is simply a reminder that doing as much work as possible within an Existential Elimination subderivation means, in part, doing as much work as can be done without violating the restrictions on the rules we use.

We have now settled on the following strategy:

Derive: (∀x)(∃y) ~ Hxy

1	(∀x)(∀y)(Hxy ⊃ ~ Hyx)	Assumption
2	(∀x)(∃y)Hxy	Assumption
3	(∃y)Hay	2 ∀E
4	Hab	A / ∃E
G	(∃y) ~ Hya	
G	(∃y) ~ Hya	3, 4–__ ∃E
G	(∀x)(∃y) ~ Hyx	__ ∀I

Our current goal is '(∃y) ~ Hya'. We would like to use Existential Introduction to derive this sentence, which means we first have to derive a substitution instance of this sentence. Looking at our first assumption, '(∀x)(∀y)(Hxy ⊃ ~ Hyx)', we see that with two applications of Universal Elimination we can obtain 'Hab ⊃ ~ Hba', then we can use Conditional Elimination to derive '~ Hba', a substitution instance of our goal, '(∃y) ~ Hya'. Our completed derivation is

Derive: $(\forall x)(\exists y) \sim Hyx$

1	$(\forall x)(\forall y)(Hxy \supset \sim Hyx)$	Assumption
2	$(\forall x)(\exists y)Hxy$	Assumption
3	$(\exists y)Hay$	2 $\forall$E
4	$\quad Hab$	A / $\exists$E
5	$\quad (\forall y)(Hay \supset \sim Hya)$	1 $\forall$E
6	$\quad Hab \supset \sim Hba$	5 $\forall$E
7	$\quad \sim Hba$	4, 6 $\supset$E
8	$\quad (\exists y) \sim Hya$	7 $\exists$I
9	$(\exists y) \sim Hya$	3, 4–9 $\exists$E
10	$(\forall x)(\exists y) \sim Hyx$	9 $\forall$I

We have met all the restrictions for using each of the two rules Existential Elimination and Universal Introduction. The constant we had to worry about in using Existential Elimination is 'b', for it is the instantiating constant used to form a substitution instance of '$(\exists y)Hay$' at line 4. By choosing 'b' as the instantiating constant we were able to meet all the restrictions on Existential Elimination: 'b' does not occur in any assumption that is open at line 9, does not occur in the existentially quantified sentence '$(\exists y)Hay$' at line 3, and does not occur in the sentence '$(\exists y) \sim Hya$' derived by Existential Elimination at line 9.

Our next argument is somewhat more complex, having one premise that contains three quantifiers:

$$\frac{(\forall x)[(\exists z)Fxz \supset (\forall y)Fxy]}{(\exists x)(\exists y)Fxy}$$

$$(\exists x)(\forall w)Fxw$$

The argument is valid in *PD,* and the derivation is not as difficult as may be feared. We will take our first clue from the second assumption, which begins with two existential quantifiers. This suggests we will be using Existential Elimination twice, as follows:

Derive: $(\exists x)(\forall y)Fxy$

1	$(\forall x)[(\exists z)Fxz \supset (\forall y)Fxy]$	Assumption
2	$(\exists x)(\exists y)Fxy$	Assumption
3	$\quad (\exists y)Fay$	A / $\exists$E
4	$\quad\quad Fab$	A / $\exists$E
G	$\quad\quad (\exists x)(\forall w)Fxw$	$\exists$I
G	$\quad (\exists x)(\forall w)Fxw$	3, 4–__ $\exists$E
G	$(\exists x)(\forall w)Fxw$	2, 3–__ $\exists$E

We next use Universal Elimination to produce a conditional to which we can apply Conditional Elimination after applying Existential Introduction to the assumption on line 4, being careful to choose an instantiating constant that will produce a match between the conditional and the existentially quantified sentence we generate. Here the instantiating constant 'a' does the trick:

Derive: $(\exists x)(\forall y)Fxy$

1	$(\forall x)[(\exists z)Fxz \supset (\forall y)Fxy]$	Assumption
2	$(\exists x)(\exists y)Fxy$	Assumption
3	$(\exists y)Fay$	A / $\exists$E
4	Fab	A / $\exists$E
5	$(\exists z)Faz \supset (\forall y)Fay$	1 $\forall$E
6	$(\exists z)Faz$	4 $\exists$I
7	$(\forall y)Fay$	5, 6 $\supset$E
G	$(\exists x)(\forall w)Fxw$	$\exists$I
G	$(\exists x)(\forall w)Fxw$	3, 4–__ $\exists$E
G	$(\exists x)(\forall w)Fxw$	2, 3–__ $\exists$E

Our current goal is '$(\exists x)(\forall w)Fxw$'. To obtain it, by Existential Introduction, we need to first derive a substitution instance of that sentence, say '$(\forall w)Faw$'. We have already derived '$(\forall y)Fay$'. This is not the sentence we need, because it contains the variable 'y' where we want 'w'. But we can easily obtain the substitution instance we want by using Universal Elimination (with a new instantiating constant) followed by Universal Introduction using the variable 'y' instead of the variable 'w'. We do this at lines 8 and 9, completing the derivation:

Derive: $(\exists x)(\forall y)Fxy$

1	$(\forall x)[(\exists z)Fxz \supset (\forall y)Fxy]$	Assumption
2	$(\exists x)(\exists y)Fxy$	Assumption
3	$(\exists y)Fay$	A / $\exists$E
4	Fab	A / $\exists$E
5	$(\exists z)Faz \supset (\forall y)Fay$	1 $\forall$E
6	$(\exists z)Faz$	4 $\exists$I
7	$(\forall y)Fay$	5, 6 $\supset$E
8	Fac	7 $\forall$E
9	$(\forall w)Faw$	8 $\forall$I
10	$(\exists x)(\forall w)Fxw$	9 $\exists$I
11	$(\exists x)(\forall w)Fxw$	3, 4–10 $\exists$E
12	$(\exists x)(\forall w)Fxw$	2, 3–11 $\exists$E

As a final example consider the following argument:

Everyone loves a lover.

Tom loves Alice.

Everyone loves everyone.

Assuming the predicate 'loves' is being used unambiguously, this argument is, perhaps surprisingly, valid. We can reason informally as follows: Because Tom loves Alice, Tom is a lover. And since everyone loves a lover, everyone loves Tom. But then everyone is a lover, and since everyone loves a lover, everyone loves everyone. Here is a symbolization of the argument in *PL:*

$(\forall x)[(\exists y)Lxy \supset (\forall z)Lzx]$

Lta

$(\forall x)(\forall y)Lxy$

As in the last example, it appears that our ultimate goal will be obtained by Universal Introduction, and indeed that our penultimate goal will also be obtained by this rule. Our work would be over if we could proceed as follows:

Derive: $(\forall x)(\forall y)Lxy$

1	$(\forall x)[(\exists y)Lxy \supset (\forall z)Lzx]$	Assumption	
2	Lta	Assumption	
3	$(\forall y)Lty$	2 $\forall$I	**MISTAKE!**
4	$(\forall x)(\forall y)Lxy$	3 $\forall$I	**MISTAKE!**

But of course we cannot do this. Both line 3 and line 4 are in violation of the restrictions on Universal Introduction. In each case the constant we are replacing, first 'a' and then 't', occurs in an open assumption (at line 2). To use Universal Introduction we need to obtain a sentence like 'Lta' but formed from other constants, any other constants. We select 'c' and 'd':

Derive: $(\forall x)(\forall y)Lxy$

1	$(\forall x)[(\exists y)Lxy \supset (\forall z)Lzx]$	Assumption	
2	Lta	Assumption	
G	Lcd		
G	$(\forall y)Lcy$	__ $\forall$I	
G	$(\forall x)(\forall y)Lxy$	__ $\forall$I	

How might we obtain our current goal, 'Lcd'? Recall the reasoning we did in English: from Lta we can infer that Tom is a lover—and we mirror this inference

in *PD* by obtaining '(∃y)Lty' by Existential Introduction. In English we reasoned that if Tom is a lover, then everyone loves Tom. We can mirror this in *PD* by applying Universal Elimination to line 1. And since we have established that Tom is a lover, we can infer that everyone loves him. So we have:

Derive: (∀x)(∀y)Lxy

1	(∀x)[(∃y)Lxy ⊃ (∀z)Lzx]	Assumption
2	Lta	Assumption
3	(∃y)Lty	2 ∃I
4	(∃y)Lty ⊃ (∀z)Lzt	1 ∀E
5	(∀z)Lzt	3, 4 ⊃E
G	Lcd	
G	(∀y)Lcy	— ∀I
G	(∀x)(∀y)Lxy	— ∀I

It is because neither 'c' nor 'd' occur in an open assumption that we will be able to derive our final goal by two uses of Universal Introduction. But how do we get, in *PD,* from line 5 to 'Lcd'? From line 5 we can get 'Ldt' by Universal Elimination. But how does this help us get 'Lcd'? One difference between these two sentences is that 'd' occurs in the first position after 'L' in the first, and in the second position in the second. We also note that line 1, which is our symbolization of 'Everyone loves a lover' contains two occurrences of the two-place predicate 'L', with 'x' occurring in the first position after L in the first occurrence, and in the second position in the second occurrence. So perhaps we can use this sentence to move 'd' from the first position after L to the second position. (Remember that 'Everyone loves a lover' does say that if someone loves then that person gets loved.) Following this clue we proceed as follows:

Derive: (∀x)(∀y)Lxy

1	(∀x)[(∃y)Lxy ⊃ (∀z)Lzx]	Assumption
2	Lta	Assumption
3	(∃y)Lty	2 ∃I
4	(∃y)Lty ⊃ (∀z)Lzt	1 ∀E
5	(∀z)Lzt	3, 4 ⊃E
6	Ldt	5 ∀E
7	(∃y)Ldy	6 ∃I
8	(∃y)Ldy ⊃ (∀z)Lzd	1 ∀E
9	(∀z)Lzd	7, 8 ⊃E
10	Lcd	9 ∀E
11	(∀y)Lcy	10 ∀I
12	(∀x)(∀y)Lxy	11 ∀I

Our derivation is now complete.

'$(\forall z)[Fz \supset (Fz \vee Gz)]$' is a theorem in *PD*. To prove that it is such we need to derive it from the empty set, which means we will need a derivation that has no primary assumptions. The most plausible strategy for obtaining this sentence is Universal Introduction.

Derive: $(\forall z)[Fz \supset (Fz \vee Gz)]$

G	$Fb \supset (Fb \vee Gb)$	
G	$(\forall z)[Fz \supset (Fz \vee Gz)]$	— $\forall$I

Our current goal is a material conditional and can be obtained by Conditional Introduction, using Disjunction Introduction to derive 'Fb $\vee$ Gb' within the Conditional Introduction subderivation:

Derive: $(\forall z)[Fz \supset (Fz \vee Gz)]$

1	Fb	A / $\supset$I
2	$Fb \vee Gb$	2 $\vee$I
3	$Fb \supset (Fb \vee Gb)$	1–2 $\supset$I
4	$(\forall z)[Fz \supset (Fz \vee Gz)]$	4 $\forall$I

We have met both of the restrictions on Universal Introduction. The instantiating constant 'b' does not occur in any assumption that is open at line 4 and does not occur in the sentence derived on line 4 by Universal Introduction.

To prove the theorem '$(\exists x)Fx \supset (\exists x)(Fx \vee Gx)$' we will use Conditional Introduction, Existential Elimination, and Existential Introduction as well as Disjunction Introduction. The proof is straightforward:

Derive: $(\exists x)Fx \supset (\exists x)(Fx \vee Gx)$

1	$(\exists x)Fx$	A / $\supset$I
2	Fa	A / $\exists$E
3	$Fa \vee Ga$	2 $\vee$I
4	$(\exists x)(Fx \vee Gx)$	3 $\exists$I
5	$(\exists x)(Fx \vee Gx)$	1, 2–4 $\exists$E
6	$(\exists x)Fx \supset (\exists x)(Fx \vee Gx)$	1–5 $\supset$I

We used Conditional Introduction as our primary strategy because our ultimate goal is a material conditional. We used Existential Elimination within that strategy because the assumption that begins the Conditional Introduction subderivation is an existentially quantified sentence. And we used Existential Introduction at line 4, within our Existential Elimination subderivation, to generate the consequent of the goal conditional. The consequent does not contain the instantiating constant 'a' and can therefore be pulled out of the Existential Elimination subderivation.

The third theorem we will prove is '$(\exists x)(\forall y)Fxy \supset (\exists x)(\exists y)Fxy$'. This is also a material conditional, and our primary strategy will again be Conditional Introduction. The assumption of our Conditional Introduction subderivation will be an existentially quantified sentence, suggesting that we use Existential Elimination within our Conditional Introduction subderivation. And if we can derive '$(\exists x)(\exists y)Fxy$' within our Existential Elimination subderivation we will be able to end that subderivation and complete our derivation:

Derive: $(\exists x)(\forall y)Fxy \supset (\exists x)(\exists y)Fxy$

1	$(\exists x)(\forall y)Fxy$	Assumption
2	$\quad(\forall y)Fay$	A / $\exists$E
	$\quad\vdots$	
G	$\quad(\exists x)(\exists y)Fxy$	
G	$(\exists x)(\exists y)Fxy$	1, 2–__ $\exists$E
G	$(\exists x)(\forall y)Fxy \supset (\exists x)(\exists y)Fxy$	1–__ $\supset$I

Completing this derivation is now straightforward. We apply Universal Elimination to the sentence on line 2 to produce 'Fab' and then use Existential Introduction twice to derive '$(\exists x)(\exists y)Fxy$'.

Derive: $(\exists x)(\forall y)Fxy \supset (\exists x)(\exists y)Fxy$

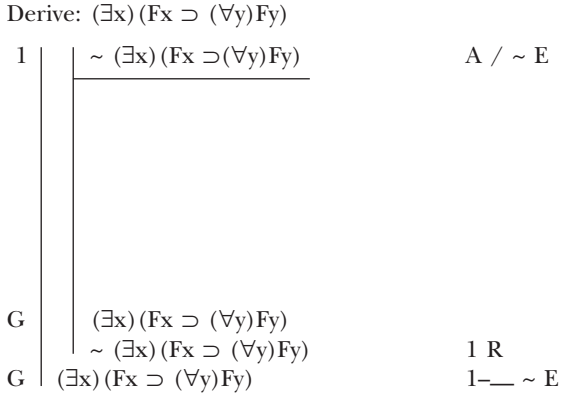
1	$(\exists x)(\forall y)Fxy$	Assumption
2	$\quad(\forall y)Fay$	A / $\exists$E
3	$\quad Fab$	2 $\forall$E
4	$\quad(\exists y)Fay$	3 $\exists$I
5	$\quad(\exists x)(\exists y)Fxy$	4 $\exists$I
6	$(\exists x)(\exists y)Fxy$	1, 2–5 $\exists$E
7	$(\exists x)(\forall y)Fxy \supset (\exists x)(\exists y)Fxy$	1–6 $\supset$I

We have met all the restrictions on Existential Elimination. The instantiating constant 'a' does not occur in any assumption that is open as of line 6. The constant 'a' also does not occur in the existentially quantified sentence to which we are applying Existential Elimination, and it does not occur in the sentence derived at line 6 by Existential Elimination.

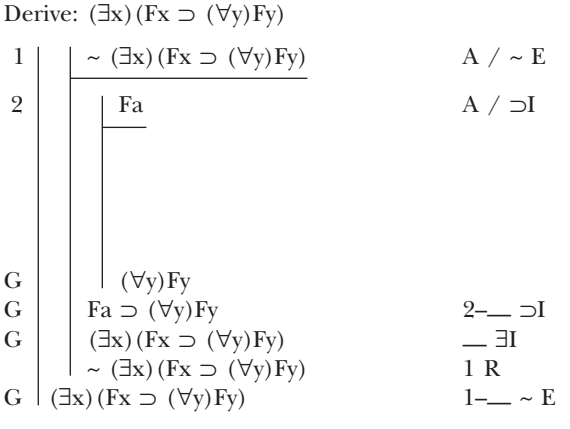
It is worth noting that since there are no restrictions on Existential Introduction, we could have entered 'Faa' rather than 'Fab' at line 3 (there are also no restrictions on Universal Elimination), and then applied Existential Introduction twice.

The last theorem we will consider is the quantified sentence '$(\exists x)(Fx \supset (\forall y)Fy)$'. At first glance it appears that we should use Existential Introduction to derive this sentence from some substitution instance, for example,

'Fa ⊃ (∀y)Fy' and so the latter sentence should be a subgoal. However, this will not work! 'Fa ⊃ (∀y)Fy' is not quantificationally true and therefore cannot be derived in *PD* from no assumptions. So we must choose another strategy. Our primary strategy will be Negation Elimination and the proof will be quite complicated:



We have selected Negation Elimination as our primary strategy because there is no plausible alternative to that strategy. We have selected '(∃x)(Fx ⊃ (∀y)Fy)' and '~ (∃x)(Fx ⊃ (∀y)Fy)' as the contradictory sentences we will derive within that strategy because the latter sentence is our assumption on line 1 and therefore available for use. The question now is how to derive '(∃x)(Fx ⊃ (∀y)Fy)'. Since this is an existentially quantified sentence we will attempt to derive it by Existential Introduction: first deriving the substitution instance 'Fa ⊃ (∀y)Fy' of that sentence (any other instantiating constant could be used). The substitution instance should be derivable using Conditional Introduction:



Our new goal is '(∀y)Fy', a universally quantified sentence. We cannot obtain it by applying Universal Introduction to the sentence on line 2, because 'a'

here occurs in an open assumption. So we will try to obtain a different substitution instance of '(∀y)Fy', 'Fb', and we will try to derive this substitution instance using Negation Elimination:

1	~ (∃x) (Fx ⊃ (∀y)Fy)	A / ~ E
2	Fa	A / ⊃I
3	~ Fb	A / ~ E
G	Fb	
G	(∀y)Fy	__ ∀I
G	Fa ⊃ (∀y)Fy	2–__ ⊃I
G	(∃x) (Fx ⊃ (∀y)Fy)	
G	~ (∃x) (Fx ⊃ (∀y)Fy)	1 R
G	(∃x) (Fx ⊃ (∀y)Fy)	1–__ ~ E

We now have to decide on the sentence and its negation to be derived within the Negation Elimination subderivation. Two negations are accessible at this point: '~ Fb' and '~ (∃x) (Fx ⊃ (∀y)Fy)'. We will make the latter sentence and '(∃x) (Fx ⊃ (∀y)Fy)' our goals as picking 'Fb' and '~ Fb' as goals appears to be unpromising (there is no obvious way to derive 'Fb' from the assumptions on lines 1–3). We plan to derive '(∃x) (Fx ⊃ (∀y)Fy)' using Existential Introduction:

1	~ (∃x) (Fx ⊃ (∀y)Fy)	A / ~ E
2	Fa	A / ⊃I
3	~ Fb	A / ~ E
G	Fb ⊃ (∀y)Fy	
G	(∃x) (Fx ⊃ (∀y)Fy)	__ ∃I
	~ (∃x) (Fx ⊃ (∀y)Fy)	1 R
G	Fb	
G	(∀y)Fy	__ ∀I
G	Fa ⊃ (∀y)Fy	2–__ ⊃I
G	(∃x) (Fx ⊃ (∀y)Fy)	
	~ (∃x) (Fx ⊃ (∀y)Fy)	1 R
G	(∃x) (Fx ⊃ (∀y)Fy)	1–__ ~ E

We have selected 'b' as the instantiating constant in our new goal because we anticipate using Conditional Introduction to derive 'Fb ⊃ (∀y)Fy', and this use

of ‘b’ will give us ‘Fb’ as an assumption, something that is likely to be useful as we already have ‘~ Fb’ at line 3.

1	~ (∃x) (Fx ⊃ (∀y)Fy)	A / ~ E
2	Fa	A / ⊃I
3	~ Fb	A / ~ E
4	Fb	A / ⊃I
G	(∀y)Fy	
G	Fb ⊃ (∀y)Fy	
G	(∃x) (Fx ⊃ (∀y)Fy)	__ ∃I
	~ (∃x) (Fx ⊃ (∀y)Fy)	1 R
G	Fb	
G	(∀y)Fy	__ ∀I
G	Fa ⊃ (∀y)Fy	2–__ ⊃I
G	(∃x) (Fx ⊃ (∀y)Fy)	
	~ (∃x) (Fx ⊃ (∀y)Fy)	1 R
G	(∃x) (Fx ⊃ (∀y)Fy)	1–__ ~ E

Our new goal is ‘(∀y)Fy’ and since ‘Fb’ and ‘~ Fb’ are both accessible, we can easily derive it using Negation Elimination, completing the derivation:

Derive: (∃x) (Fx ⊃ (∀y)Fy)

1	~ (∃x) (Fx ⊃ (∀y)Fy)	A / ~ E
2	Fa	A / ⊃I
3	~ Fb	A / ~ I
4	Fb	A / ⊃I
5	~ (∀y)Fy	A / ~ E
6	Fb	4 R
7	~ Fb	3 R
8	(∀y)Fy	5–7 ~ E
9	Fb ⊃ (∀y)Fy	4–8 ⊃I
10	(∃x) (Fx ⊃ (∀y)Fy)	9 ∃I
11	~ (∃x) (Fx ⊃ (∀y)Fy)	l R
12	Fb	3–11 ~ E
13	(∀y)Fy	12 ∀I
14	Fa ⊃ (∀y)Fy	2–13 ⊃I
15	(∃x) (Fx ⊃ (∀y)Fy)	14 ∃I
16	~ (∃x) (Fx ⊃ (∀y)Fy)	1 R
17	(∃x) (Fx ⊃ (∀y)Fy)	1–16 ~ E

This is a complex derivation, as we warned it would be. In the end we used the same pair of contradictory sentences in two Negation Elimination subderivations. This sometimes happens.

To show that sentences **P** and **Q** of *PL* are equivalent in *PD* we must derive each from the unit set of the other. As our first example we take the sentences '$(\forall x)(Fa \supset Fx)$' and '$Fa \supset (\forall x)Fx$'. We begin by deriving the second of these sentences from the first, and since our goal sentence in this derivation is a material conditional, we will use Conditional Introduction:

Derive: $Fa \supset (\forall x)Fx$

1	$(\forall x)(Fa \supset Fx)$	Assumption
2	$\quad Fa$	A / $\supset$I
G	$\quad (\forall x)Fx$	
G	$Fa \supset (\forall x)Fx$	2–__ $\supset$I

We cannot derive our present goal, '$(\forall x)Fx$', by simply applying Universal Introduction to 'Fa' at line 2, for the sentence on line 2 is an open assumption and 'a' occurs in that sentence. We can rather try to derive a different substitution instance of '$(\forall x)Fx$', say 'Fb', and then apply Universal Introduction. And this is easy to do by applying Universal Elimination to the sentence on line 1 (being careful to use an instantiating constant other than 'a'), and then using Conditional Elimination:

Derive: $Fa \supset (\forall x)Fx$

1	$(\forall x)(Fa \supset Fx)$	Assumption
2	$\quad Fa$	A / $\supset$I
3	$\quad Fa \supset Fb$	1 $\forall$E
4	$\quad Fb$	2, 3 $\supset$E
5	$\quad (\forall x)Fx$	4 $\forall$I
6	$Fa \supset (\forall x)Fx$	2–5 $\supset$I

We have met both restrictions on Universal Introduction at line 5: the instantiating constant 'b' does not occur in any open assumption; nor does it occur in the derived sentence '$(\forall x)Fx$'.

We must now derive '$(\forall x)(Fa \supset Fx)$' from '$Fa \supset (\forall x)Fx$'. A plausible start is

Derive: $(\forall x)(Fa \supset Fx)$

1	$Fa \supset (\forall x)Fx$	Assumption
2	$\quad Fa$	A / $\supset$I
G	$\quad Fb$	
G	$Fa \supset Fb$	2–__ $\supset$I
G	$(\forall x)(Fa \supset Fx)$	__ $\forall$I

We plan to derive the last sentence by Universal Introduction, and the substitution instance on the prior line by Conditional Introduction. And we can now see how to complete the derivation. We can apply Conditional Elimination to the sentences on lines 1 and 2 to derive '(∀x)Fx', from which we can then derive 'Fb':

Derive: (∀x)(Fa ⊃ Fx)

1	Fa ⊃ (∀x)Fx	Assumption
2	Fa	A / ⊃I
3	(∀x)Fx	1, 2 ⊃E
4	Fb	3 ∀E
5	Fa ⊃ Fb	2–4 ⊃I
6	(∀x)(Fa ⊃ Fx)	5 ∀I

Having derived each member of our pair of sentences from the other, we have demonstrated that the sentences '(∀x)(Fa ⊃ Fx)' and 'Fa ⊃ (∀x)Fx' are equivalent in *PD*.

We will next show that '(∀x)Fx ⊃ Ga' and '(∃x)(Fx ⊃ Ga)' are equivalent in *PD*. It is reasonably straightforward to derive '(∀x)Fx ⊃ Ga' from '(∃x)(Fx ⊃ Ga)'. We begin with

Derive: (∀x)Fx ⊃ Ga

1	(∃x)(Fx ⊃ Ga)	Assumption
2	(∀x)Fx	A / ⊃I
G	Ga	
G	(∀x)Fx ⊃ Ga	2–__ ⊃I

We will complete the derivation by using Existential Elimination—being careful to use an instantiating constant other than 'a' (because 'a' occurs in 'Ga', the sentence we plan to derive with Existential Elimination):

Derive: (∀x)Fx ⊃ Ga

1	(∃x)(Fx ⊃ Ga)	Assumption
2	(∀x)Fx	A / ⊃I
3	Fb ⊃ Ga	A / ∃E
4	Fb	2 ∀E
5	Ga	3–4 ⊃E
6	Ga	1, 3–5 ∃E
7	(∀x)Fx ⊃ Ga	2–6 ⊃I

Our use of Existential Elimination at line 6 meets all three restrictions on that rule: the instantiating constant 'b' does not occur in '$(\exists x)(Fx \supset Ga)$', does not occur in any assumption that is open at line 6, and does not occur in the sentence 'Ga' that we derived with Existential Elimination.

Deriving '$(\exists x)(Fx \supset Ga)$' from '$(\forall x)Fx \supset Ga$' is a somewhat more challenging exercise. Since our primary goal is an existentially quantified sentence, both Existential Introduction and Negation Elimination suggest themselves as primary strategies. We have opted to use Negation Elimination, and since the assumption that begins that strategy is a negation, we will make it and the sentence of which it is a negation our goals within the Negation Elimination subderivation:

Derive: $(\exists x)(Fx \supset Ga)$

1	$(\forall x)Fx \supset Ga$	Assumption
2	$\sim (\exists x)(Fx \supset Ga)$	A / ~ E
G	$(\exists x)(Fx \supset Ga)$	
	$\sim (\exists x)(Fx \supset Ga)$	2 R
G	$(\exists x)(Fx \supset Ga)$	1–__ ~ E

When two primary strategies suggest themselves, it is frequently useful to use one as a secondary strategy within the other, primary strategy. Here we will use Existential Introduction as a secondary strategy: We will try to obtain the goal '$(\exists x)(Fx \supset Ga)$' by Existential Introduction, first using Conditional Introduction to derive an appropriate substitution instance of the goal sentence:

Derive: $(\exists x)(Fx \supset (\forall y)Fy)$

1	$(\forall x)Fx \supset Ga$	Assumption
2	$\sim (\exists x)(Fx \supset Ga)$	A / ~ E
3	Fa	A / ⊃I
G	Ga	
G	$Fa \supset Ga$	3–__ ⊃I
G	$(\exists x)(Fx \supset Ga)$	__ ∃I
	$\sim (\exists x)(Fx \supset Ga)$	1 R
G	$(\exists x)(Fx \supset Ga)$	1–__ ~ E

The current goal, 'Ga', can be derived by Conditional Elimination using the sentence on line 1 *if* we can first derive the antecedent '$(\forall x)Fx$' of that sentence. It is not easy to see how the antecedent might be derived, but one strategy is to try to first derive a substitution instance in which the instantiating constant does not occur in an open assumption. This rules out 'Fa'. So we will try to derive 'Fb', and since no more direct strategy suggests itself at this point, we'll try to derive 'Fb' by Negation Elimination:

Derive: $(\exists x)(Fx \supset (\forall y)Fy)$

1	$(\forall x)Fx \supset Ga$	Assumption
2	$\sim (\exists x)(Fx \supset Ga)$	A / ~ E
3	Fa	A / ⊃I
4	$\sim$ Fb	A / ~ E
G	Fb	4–__ ~ E
G	$(\forall x)Fx$	__ ∀I
G	Ga	1, __ ⊃E
G	Fa ⊃ Ga	3–__ ⊃I
G	$(\exists x)(Fx \supset Ga)$	∃I
	$\sim (\exists x)(Fx \supset Ga)$	1 R
G	$(\exists x)(Fx \supset Ga)$	1–__ ~ E

Given '~ Fb' at line 4 we can obtain 'Fb ⊃ Ga'. We know we can do this because we know that given the negation of the antecedent of any conditional we can derive the conditional—as the following schema demonstrates:

n	$\sim$ **P**	
n+1	**P**	A / ⊃I
n+2	$\sim$ **Q**	A / ~ E
n+3	**P**	n+1 R
n+4	$\sim$ **P**	n R
n+5	**Q**	n+2−n+4 ~ E
n+6	**P** ⊃ **Q**	n+1−n+5 ⊃I

Once we derive 'Fb ⊃ Ga' we can obtain '$(\exists x)(Fx \supset Ga)$' by Existential Introduction. Because we already have the negation of that sentence at line 2 we can see our way clear to deriving a sentence and its negation as follows:

Derive: $(\exists x)(Fx \supset Ga)$

1	$(\forall x)Fx \supset Ga$	Assumption
2	$\sim (\exists x)(Fx \supset Ga)$	A / ~ E
3	Fa	A / $\supset$I
4	$\sim Fb$	A / ~ E
5	Fb	A / $\supset$I
6	$\sim Ga$	A / ~ E
7	Fb	5 R
8	$\sim Fb$	4 R
9	Ga	6–8 ~ E
10	$Fb \supset Ga$	5–9 $\supset$I
11	$(\exists x)(Fx \supset Ga)$	10 $\exists$I
12	$\sim (\exists x)(Fx \supset Ga)$	2 R
13	Fb	4–12 ~ E
14	$(\forall x)Fx$	13 $\forall$I
15	Ga	1, 14 $\supset$E
16	$Fa \supset Ga$	3–15 $\supset$I
17	$(\exists x)(Fx \supset Ga)$	16 $\exists$I
18	$\sim (\exists x)(Fx \supset Ga)$	2 R
19	$(\exists x)(Fx \supset Ga)$	1–18 ~ E

We will conclude our discussion of Equivalence in *PD* by deriving each of the following sentences from the unit set of the other:

$$(\forall x)[Fx \supset (\exists y)Gxy] \qquad (\forall x)(\exists y)(Fx \supset Gxy)$$

Establishing that these sentences are equivalent in *PD* is substantially more difficult than was establishing equivalence in our last example, in large part because in these sentences the existentially quantified formulas occur *within the scope of* universal quantifiers. We begin by deriving '$(\forall x)(\exists y)(Fx \supset Gxy)$' from $\{(\forall x)[Fx \supset (\exists y)Gxy]\}$. Since our one primary assumption will be a universally quantified sentence, as will our goal, it is plausible to expect that we will use both Universal Elimination and Universal Introduction:

Derive: $(\forall x)(\exists y)(Fx \supset Gxy)$

1	$(\forall x)[Fx \supset (\exists y)Gxy]$	Assumption
2	$Fa \supset (\exists y)Gay$	1 $\forall$E
G	$(\exists y)(Fa \supset Gay)$	
G	$(\forall x)(\exists y)(Fx \supset Gxy)$	__ $\forall$I

It is now tempting to make 'Fa ⊃ Gab' our next subgoal, to be derived using Conditional Introduction. And if we can obtain 'Fa ⊃ Gab' we can go on to derive '(∃y)(Fa ⊃ Gay)' by Existential Introduction:

Derive: (∀x)(∃y)(Fx ⊃ Gxy)

1	(∀x)[Fx ⊃ (∃y)Gxy]	Assumption
2	Fa ⊃ (∃y)Gay	1 ∀E
3	Fa	A / ⊃I
G	Gab	
G	Fa ⊃ Gab	3–__ ⊃I
G	(∃y)(Fa ⊃ Gay)	__ ∃I
G	(∀x)(∃y)(Fx ⊃ Gxy)	__ ∀I

'(∃y)Gay' can be derived from lines 2 and 3 by Conditional Elimination. We might then plan to use Existential Elimination to get from '(∃y)Gay' to the current goal sentence 'Gab'. But we have to be careful here. If we want to derive 'Gab' by Existential Elimination then the instantiating constant for Existential Elimination has to be a constant other than 'b'.

Derive: (∀x)(∃y)(Fx ⊃ Gxy)

1	(∀x)[Fx ⊃ (∃y)Gxy]	Assumption
2	Fa ⊃ (∃y)Gay	1 ∀E
3	Fa	A / ⊃I
4	(∃y)Gay	2, 3 ⊃E
5	Gac	A / ∃E
G	Gab	
G	Gab	4, 5–__ ∃E
G	Fa ⊃ Gab	3–__ ⊃I
G	(∃y)(Fa ⊃ Gay)	__ ∃I
G	(∀x)(∃y)(Fx ⊃ Gxy)	__ ∀I

But how do we get from 'Gac' to 'Gab'? A negation strategy might work, but it would be complicated as there are no negations among the accessible sentences.

It is time to consider an alternative strategy. We will try to obtain our penultimate goal, '(∃y)(Fa ⊃ Gay)', by Negation Elimination rather than by Existential Introduction:

Derive: (∀x)(∃y)(Fx ⊃ Gxy)

1	(∀x)[Fx ⊃ (∃y)Gxy]	Assumption
2	~ (∃y)(Fa ⊃ Gay)	A / ~ E
G	(∃y)(Fa ⊃ Gay)	15 ∃I
G	~ (∃y)(Fa ⊃ Gay)	2 R
G	(∃y)(Fa ⊃ Gay)	2–17 ~ E
G	(∀x)(∃y)(Fx ⊃ Gxy)	18 ∀I

It may appear that because '(∃y)(Fa ⊃ Gay)' is still our goal we are making no progress. But this is not so, for we now have an additional assumption to work from. We will now proceed much as we did in our first attempt at this derivation:

Derive: (∀x)(∃y)(Fx ⊃ Gxy)

1	(∀x)[Fx ⊃ (∃y)Gxy]	Assumption
2	~ (∃y)(Fa ⊃ Gay)	A / ~ E
3	Fa	A / ⊃I
4	Fa ⊃ (∃y)Gay	1 ∀E
5	(∃y)Gay	3, 4 ⊃E
6	Gac	A / ∃E
G	Gab	
G	Gab	5, 6–__ ∃E
G	Fa ⊃ Gab	3–__ ⊃I
G	(∃y)(Fa ⊃ Gay)	__ ∃I
G	~ (∃y)(Fa ⊃ Gay)	2 R
G	(∃y)(Fa ⊃ Gay)	2–__ ~ E
G	(∀x)(∃y)(Fx ⊃ Gxy)	__ ∀I

Once again we want to get from 'Gac' to 'Gab'. But this time we do have an accessible negation, '~ (∃y)(Fa ⊃ Gay)'. So we will use a negation strategy, assuming '~ Gab' and seeking to derive '(∃y)(Fa ⊃ Gay)' along with reiterating its negation:

Derive: $(\forall x)(\exists y)(Fx \supset Gxy)$

1	$(\forall x)[Fx \supset (\exists y)Gxy]$	Assumption
2	$\sim (\exists y)(Fa \supset Gay)$	A / ~ E
3	Fa	A ⊃I
4	$Fa \supset (\exists y)Gay$	1 ∀E
5	$(\exists y)Gay$	3, 4 ⊃E
6	Gac	A / ∃E
7	$\sim Gab$	A ~ E
G	$(\exists y)(Fa \supset Gay)$	
	$\sim (\exists y)(Fa \supset Gay)$	2 R
G	Gab	7–__ ~ E
G	Gab	5, 6–__ ∃E
G	$Fa \supset Gab$	3–__ ⊃I
G	$(\exists y)(Fa \supset Gay)$	__ ∃I
	$\sim (\exists y)(Fa \supset Gay)$	2 R
G	$(\exists y)(Fa \supset Gay)$	2–__ ~ E
G	$(\forall x)(\exists y)(Fx \supset Gxy)$	__ ∀I

What remains is to derive '$(\exists y)(Fa \supset Gay)$'. This is easily done. We assume 'Fa', derive 'Gac' by Reiteration, derive '$Fa \supset Gac$' by Conditional Introduction, and then '$(\exists y)(Fa \supset Gay)$' by Existential Introduction. The derivation is then complete:

Derive: $(\forall x)(\exists y)(Fx \supset Gxy)$

1	$(\forall x)[Fx \supset (\exists y)Gxy]$	Assumption
2	$\sim (\exists y)(Fa \supset Gay)$	A / ~ E
3	Fa	A ⊃I
4	$Fa \supset (\exists y)Gay$	1 ∀E
5	$(\exists y)Gay$	3, 4 ⊃E
6	Gac	A / ∃E
7	$\sim Gab$	A ~ E
8	Fa	A / ⊃I
9	Gac	6 R
10	$Fa \supset Gac$	8–9 ⊃I
11	$(\exists y)(Fa \supset Gay)$	10 ∃I
12	$\sim (\exists y)(Fa \supset Gay)$	2 R
13	Gab	7–12 ~ E
14	Gab	5, 6–13 ∃E
15	$Fa \supset Gab$	3–14 ⊃I
16	$(\exists y)(Fa \supset Gay)$	15 ∃I
17	$\sim (\exists y)(Fa \supset Gay)$	2 R
18	$(\exists y)(Fa \supset Gay)$	2–17 ~ E
19	$(\forall x)(\exists y)(Fx \supset Gxy)$	18 ∀I

We must now derive '$(\forall x)[Fx \supset (\exists y)Gxy]$' from '$(\forall x)(\exists y)(Fx \supset Gxy)$'. This will be an easier task since we can derive '$(\exists y)(Fa \supset Gay)$' by Universal Elimination and then do the bulk of the derivation within an Existential Elimination subderivation:

Derive: $(\forall x)[Fx \supset (\exists y)Gxy]$

1	$(\forall x)(\exists y)(Fx \supset Gxy)$	Assumption
2	$(\exists y)(Fa \supset Gay)$	1 $\forall$E
3	$Fa \supset Gab$	A / $\exists$E
4	Fa	A $\supset$I
5	Gab	3, 4 $\supset$E
6	$(\exists y)Gay$	5 $\exists$I
7	$Fa \supset (\exists y)Gay$	4–6 $\supset$I
8	$Fa \supset (\exists y)Gay$	3, 4–7 $\exists$E
9	$(\forall x)[Fx \supset (\exists y)Gxy]$	8 $\forall$I

The instantiating constant 'b' for our use of Existential Elimination does not occur in the existentially quantified sentence '$(\exists y)(Fa \supset Gay)$', in any assumption that is open at line 8, or in the sentence '$Fa \supset (\exists y)Gay$' obtained by Existential Elimination. (In this case we could also have applied Universal Introduction *within* the Existential Elimination subderivation and then moved '$(\forall x)[Fx \supset (\exists y)Gxy]$' out of that subderivation.) This completes our demonstration that '$(\forall x)[Fx \supset (\exists y)Gxy]$' and '$(\forall x)(\exists y)(Fx \supset Gxy)$' are equivalent in *PD*.

INCONSISTENCY

We next turn our attention to demonstrating that sets of sentences of *PL* are inconsistent in *PD*. Recall that a set of sentences is inconsistent in *PD* if we can derive both a sentence **Q** and its negation ~ **Q** from the set. As our first example we will show that the set $\{(\forall x)(Fx \equiv Gx), (\exists y)(Fy \,\&\sim Gy)\}$ is inconsistent in *PD*. Because this set does not contain a negation, it is not obvious what our **Q** and ~ **Q** should be. We will use the set member '$(\forall x)(Fx \equiv Gx)$' as **Q**, making ~ **Q** '$\sim (\forall x)(Fx \equiv Gx)$':

Derive: $(\forall x)(Fx \equiv Gx), \sim (\forall x)(Fx \equiv Gx)$

1	$(\forall x)(Fx \equiv Gx)$	Assumption
2	$(\exists y)(Fy \,\&\sim Gy)$	Assumption
G	$\sim (\forall x)(Fx \equiv Gx)$	
G	$(\forall x)(Fx \equiv Gx)$	1 R

The second assumption suggests using Existential Elimination, and we know it is wise to do as much of the work of the derivation as possible within the Existential Elimination subderivation:

Derive: $(\forall x)(Fx \equiv Gx), \sim (\forall x)(Fx \equiv Gx)$

1	$(\forall x)(Fx \equiv Gx)$	Assumption
2	$(\exists y)(Fy \,\&\, \sim Gy)$	Assumption
3	$Fa \,\&\, \sim Ga$	A / $\exists$E
G	$\sim (\forall x)(Fx \equiv Gx)$	
G	$\sim (\forall x)(Fx \equiv Gx)$	2, 3–__ $\exists$E
	$(\forall x)(Fx \equiv Gx)$	1 R

Our current goal is a negation, which we will try to derive using Negation Introduction. We assume '$(\forall x)(Fx \equiv Gx)$' *even though that sentence is one of our primary assumptions and hence already accessible.* We assume it because Negation Introduction requires that we assume the sentence whose negation we wish to derive:

Derive: $(\forall x)(Fx \equiv Gx), \sim (\forall x)(Fx \equiv Gx)$

1	$(\forall x)(Fx \equiv Gx)$	Assumption
2	$(\exists y)(Fy \,\&\, \sim Gy)$	Assumption
3	$Fa \,\&\, \sim Ga$	A / $\exists$E
4	$(\forall x)(Fx \equiv Gx)$	A / $\sim$ I
G	$\sim (\forall x)(Fx \equiv Gx)$	4–__ $\sim$ I
G	$\sim (\forall x)(Fx \equiv Gx)$	2, 3–__ $\exists$E
G	$(\forall x)(Fx \equiv Gx)$	1 R

We are now finally in a position where we can work profitably from the "top down". From line 4 we can derive '$Fa \equiv Ga$' by Biconditional Elimination; from line 3 we can derive 'Fa'; and then it is easy to derive both 'Ga' and '$\sim$ Ga':

Derive: $(\forall x)(Fx \equiv Gx), \sim (\forall x)(Fx \equiv Gx)$

1	$(\forall x)(Fx \equiv Gx)$	Assumption
2	$(\exists y)(Fy \,\&\, \sim Gy)$	Assumption
3	$Fa \,\&\, \sim Ga$	A / $\exists$E
4	$(\forall x)(Fx \equiv Gx)$	A / $\sim$ I
5	$Fa \equiv Ga$	4 $\forall$E
6	Fa	3 &E
7	Ga	5, 6 $\equiv$E
8	$\sim Ga$	3 &E
9	$\sim (\forall x)(Fx \equiv Gx)$	4–8 $\sim$ I
10	$\sim (\forall x)(Fx \equiv Gx)$	2, 3–9 $\exists$E
11	$(\forall x)(Fx \equiv Gx)$	1 R

Had we taken '$(\exists y)(Fy \,\&\, \sim Gy)$' and '$\sim (\exists y)(Fy \,\&\, \sim Gy)$' as our **Q** and $\sim$ **Q** we would have produced the following very similar derivation:

Derive: $(\exists y)(Fy \,\&\, \sim Gy)$, $\sim (\exists y)(Fy \,\&\, \sim Gy)$

1	$(\forall x)(Fx \equiv Gx)$	Assumption
2	$(\exists y)(Fy \,\&\, \sim Gy)$	Assumption
3	$\quad Fa \,\&\, \sim Ga$	A / $\exists$E
4	$\quad\quad (\exists y)(Fy \,\&\, \sim Gy)$	A / $\sim$ I
5	$\quad\quad Fa \equiv Ga$	1 $\forall$E
6	$\quad\quad Fa$	3 &E
7	$\quad\quad Ga$	5, 6 $\equiv$E
8	$\quad\quad \sim Ga$	3 &E
9	$\quad \sim (\exists y)(Fy \,\&\, \sim Gy)$	4–8 $\sim$ I
10	$\sim (\exists y)(Fy \,\&\, \sim Gy)$	2, 3–9 $\exists$E
11	$(\exists y)(Fy \,\&\, \sim Gy)$	2 R

We will next demonstrate that $\{(\forall z)(Hz \supset (\exists y)Gzy)$, $(\exists w)Hw$, $(\forall x) \sim (\exists y)Gxy\}$ is inconsistent in *PD*. Though the set includes no negations, we can immediately derive one, say '$\sim (\exists y)Gay$', by applying Universal Elimination to '$(\forall x) \sim (\exists y)Gxy$'. So we will take '$(\exists y)Gay$' and '$\sim (\exists y)Gay$' as our goals:

Derive: $(\exists y)Gay$, $\sim (\exists y)Gay$

1	$(\forall z)(Hz \supset (\exists y)Gzy)$	Assumption
2	$(\exists w)Hw$	Assumption
3	$(\forall x) \sim (\exists y)Gxy$	Assumption
G	$(\exists y)Gay$	
	$\sim (\exists y)Gay$	3 $\forall$E

Our assumptions include the existentially quantified sentence '$(\exists w)Hw$', so we will try to derive '$(\exists y)Gay$' by Existential Elimination—which means we will have to be careful to pick a constant other than 'a' as the instantiating constant in our Existential Elimination assumption:

Derive: $(\exists y)Gay$, $\sim (\exists y)Gay$

1	$(\forall z)(Hz \supset (\exists y)Gzy)$	Assumption
2	$(\exists w)Hw$	Assumption
3	$(\forall x) \sim (\exists y)Gxy$	Assumption
4	$\quad Hb$	A / $\exists$E
G	$\quad (\exists y)Gay$	
G	$(\exists y)Gay$	2, 4–__ $\exists$E
	$\sim (\exists y)Gay$	3 $\forall$E

There is a problem in the offing here. We used 'b' as the instantiating constant at line 4 because 'a' occurs in the sentence we hope to obtain by Existential Elimination, '(∃y)Gay'. This means that we will be able to obtain '(∃y)Gby', but not '(∃y)Gay' by applying Universal Elimination to line 1 (obtaining 'Hb ⊃ (∃y)Gby' and then doing Conditional Elimination). So we need an alternative strategy for obtaining our current goal, '(∃y)Gay'. We will use Negation Elimination:

Derive: (∃y)Gay, ~ (∃y)Gay

1	(∀z)(Hz ⊃ (∃y)Gzy)	Assumption
2	(∃w)Hw	Assumption
3	(∀x) ~ (∃y)Gxy	Assumption
4	Hb	A / ∃E
5	~ (∃y)Gay	A / ~ E
G	(∃y)Gay	5–__ ~ E
G	(∃y)Gay	2, 4–6 ∃E
	~ (∃y)Gay	3 ∀E

We can now complete the derivation by deriving both '(∃y)Gby' and '~ (∃y)Gby' within the scope of the assumption on line 5, the first by the steps mentioned previously, the second by applying Universal Elimination to the sentence on line 3.

Derive: (∃y)Gay, ~ (∃y)Gay

1	(∀z) (Hz ⊃ (∃y)Gzy)	Assumption
2	(∃w)Hw	Assumption
3	(∀x) ~ (∃y)Gxy	Assumption
4	Hb	A / ∃E
5	~ (∃y)Gay	A / ~ E
6	Hb ⊃ (∃y)Gby	1 ∀E
7	(∃y)Gby	4, 6 ⊃E
8	~ (∃y)Gby	3 ∀E
9	(∃y)Gay	5–8 ~ E
10	(∃y)Gay	2, 4–9 ∃E
11	~ (∃y)Gay	3 ∀E

The technique of using a negation strategy within an Existential Elimination subderivation, as we have just done, is useful as a way of generating a sentence that does not violate any of the restrictions on Existential Elimination. It is useful whenever we can see that some sentence and its negation are derivable within the Existential Elimination subderivation, but those sentences contain a constant that keeps us from moving either out from the Existential Elimination subderivation by Existential Elimination. In such a case we can

always derive a sentence that does not contain the Existential Elimination subderivation's instantiating constant. We can do this by assuming the negation of the desired sentence and deriving the contradictory sentences within the negation elimination subderivation.

10.2**E** EXERCISES

Note: Here, as always, the *Student Solutions Manual* contains answers to all unstarred exercises. In addition, when an exercise is preceded by a number sign (#) the *Solutions Manual* contains a detailed account of how the derivation given in the *Solutions Manual* is constructed.

1. Construct derivations that establish the validity of the following arguments:

a. $(\forall y)[Fy \supset (Gy \,\&\, Hy)]$
$\therefore (\forall x)(Fx \supset Hx)$

*b. $(\forall x)(Fx \equiv Gx)$
$(\exists x)Fx$
$\therefore (\exists x)(Fx \,\&\, Gx)$

#c. $(\forall y)[Gy \supset (Hy \,\&\, Fy)]$
$(\exists x)Gx$
$\therefore (\exists z)Fz$

*d. $(\forall x)[Fx \supset (Gx \,\&\, Hx)]$
$(\exists y)(Fy \,\&\, Dy)$
$\therefore (\exists z)Gz$

e. $(\exists x)Fx \supset (\forall x)Gx$
Fa
$(\forall x)(Gx \supset Hx)$
$\therefore (\forall x)Hx$

*f. $(\forall y)[(Hy \,\&\, Fy) \supset Gy]$
$(\forall z)Fz$
$\therefore (\forall x)(Hx \supset Gx)$

g. $(\forall x)Fx \vee (\forall x)Gx$
$\therefore (\forall x)(Fx \vee Gx)$

*h. $(\forall x)(Dx \equiv \sim Gx)$
$(\forall y)(Gy \supset Hy)$
$(\exists z) \sim Hz$
$\therefore (\exists z)Dz$

#i. $(\forall x)(Fx \supset Hx)$
$(\forall y)(Gy \supset Hy)$
$\therefore (\forall y)[(Fy \vee Gy) \supset Hy]$

*j. $(\exists y)(Fy \vee Gy)$
$(\forall x)(Fx \supset Hx)$
$(\forall x)(Gx \supset Hx)$
$\therefore (\exists z)Hz$

k. $(\exists x)Hx$
$(\forall x)(Hx \supset Rx)$
$(\exists x)Rx \supset (\forall x)Gx$
$\therefore (\forall x)(Fx \supset Gx)$

*l. $\sim (\exists x)Fx \equiv (\forall y)Gy$
$(\forall y) \sim Fy$
$\therefore (\exists y)Gy$

m. $(\forall x)Fx \vee (\forall y) \sim Gy$
$Fa \supset Hb$
$\sim Gb \supset Jb$
$\therefore (\exists y)(Hy \vee Jy)$

*n. $Fa \vee (\forall x) \sim Fx$
$(\exists y)Fy$
$\therefore Fa$

2. Prove that the following sentences of *PL* are theorems of *PD:*

a. $Fa \supset (\exists y)Fy$
*b. $(\forall x)Fx \supset (\exists y)Fy$
c. $(\forall x)[Fx \supset (Gx \supset Fx)]$
*d. $\sim Fa \supset \sim (\forall x)Fx$
e. $\sim (\exists x)Fx \supset (\forall x) \sim Fx$
*f. $(\exists x)(\exists y)Fxy \supset (\exists y)(\exists x)Fyx$
g. $Fa \vee (\exists y) \sim Fy$
*h. $(\forall x)(Hx \supset Ix) \supset [(\exists x)Hx \supset (\exists x)Ix]$
#i. $[(\forall x)Fx \vee (\forall x)Gx] \supset (\forall x)(Fx \vee Gx)$
*j. $[(\forall x)Fx \,\&\, (\exists y)Gy] \supset (\exists x)(Fx \,\&\, Gx)$
k. $(\exists x)(Fx \,\&\, Gx) \supset [(\exists x)Fx \,\&\, (\exists x)Gx]$
*l. $[(\exists x)Fx \vee (\exists x)Gx] \supset (\exists x)(Fx \vee Gx)$
m. $(\forall x)Hx \equiv \sim (\exists x) \sim Hx$

3. Construct derivations that establish that the following pairs of sentences are equivalent in *PD:*

a.	$(\forall x)(Fx \,\&\, Gx)$	$(\forall x)Fx \,\&\, (\forall x)Gx$
*b.	$(\forall x)(Fx \supset Ga)$	$(\exists x)Fx \supset Ga$
c.	$(\forall x)Fx$	$\sim (\exists x) \sim Fx$
*d.	$(\exists y)(Fy \,\&\, (\forall x)Gx)$	$(\exists y)(\forall x)(Fy \,\&\, Gx)$
#e.	$(\exists x)Fx$	$\sim (\forall x) \sim Fx$
*f.	$(\exists x)(Fx \,\&\, \sim Gx)$	$\sim (\forall x)(Fx \supset Gx)$
g.	$(\forall z)(Hz \supset \sim Iz)$	$\sim (\exists y)(Hy \,\&\, Iy)$
*h.	$(\exists x)(Fa \supset Gx)$	$Fa \supset (\exists x)Gx$
i.	$(\forall x)(\exists y)(Fx \supset Gy)$	$(\forall x)(Fx \supset (\exists y)Gy)$

4. Construct derivations that establish that the following sets are inconsistent in *PD:*

a. $\{(\forall x)(Fx \equiv \sim Fx)\}$
*b. $\{(\forall x)Hx, (\forall y) \sim (Hy \vee Gyy)\}$
#c. $\{\sim (\forall x)Fx, \sim (\exists x) \sim Fx\}$
*d. $\{\sim (\forall x) \sim Fx, \sim (\exists x)Fx\}$
e. $\{(\forall x)(Fx \supset Gx), (\exists x)Fx, \sim (\exists x)Gx\}$
*f. $\{(\forall z) \sim Fz, (\exists z)Fz\}$
g. $\{(\forall x)Fx, (\exists y) \sim Fy\}$
*h. $\{(\exists y)(Hy \,\&\, Jy), (\forall x) \sim Jx\}$
i. $\{(\forall x)(Hx \equiv \sim Gx), (\exists x)Hx, (\forall x)Gx\}$
*j. $\{(\forall z)(Hz \supset Iz), (\exists y)(Hy \,\&\, \sim Iy)\}$
k. $\{(\forall z)[Rz \supset (Tz \,\&\, \sim Mz)], (\exists y)(Ry \,\&\, My)\}$
*l. $\{(\forall x)(Fx \supset Gx), (\forall x)(Fx \supset \sim Gx), (\exists x)Fx\}$

5. Construct derivations that establish the following:

a. $\{(\exists y)(\forall x)Fxy\} \vdash (\forall x)(\exists y)Fxy$
*b. $\{(\forall z)(Gz \supset (\exists x)Fxz), (\forall x)Gx\} \vdash (\forall z)(\exists x)Fxz$
c. $\{(\exists x)Fxxx\} \vdash (\exists x)(\exists y)(\exists z)Fxyz$
*d. $\{(\forall x)(\forall y)(Bx \supset Txy\} \vdash (\forall x)(\forall y)[(Bx \,\&\, Ny) \supset Txy]$
e. $\{(\forall x)(Fx \supset (\exists y)Gxy), (\exists x)Fx\} \vdash (\exists x)(\exists y)Gyx$
*f. $\{(\forall x)(\exists y)Gxy, (\forall x)(\forall y)(Hxy \supset \sim Gxy)\} \vdash (\forall x)(\exists z) \sim Hxz$
g. $\{(\forall x)(\forall y)(Hxy \supset \sim Hyx), (\exists x)(\exists y)Hxy\} \vdash (\exists x)(\exists y) \sim Hyx$
*h. $\{(\forall x)(\forall y)Fxy \vee (\forall x)(\forall y)Gxy\} \vdash (\forall x)(\forall y)(Fxy \vee Gxy)$
i. $\{\sim (\exists x)(\exists y)Rxy, (\forall x)(\forall y)(\sim Hxy \equiv Rxy)\} \vdash (\forall x)(\forall y)Hxy$
*j. $\{(\forall x)(\forall y)(Fxy \equiv \sim Gyx), (\exists z)(\exists w)Gzw\} \vdash (\exists x)(\exists y) \sim Fxy$

6. Construct derivations that establish the validity of the following arguments:

a. $(\forall x)(Fx \supset Gba)$
$(\exists x)Fx$
$(\exists y)Gya$

*b. $(\forall x)(Hx \supset (\forall y)Rxyb)$
$(\forall x)(\forall z)(Razx \supset Sxzz)$
$Ha \supset (\exists x)Sxcc$

c. $(\exists x)(\exists y)(Fxy \vee Fyx)$
$(\exists x)(\exists y)Fxy$

*d. $(\forall x)(Fxa \supset Fax)$
$(\exists x)(Hx \,\&\, \sim Fax)$
$\sim (\forall y)(Hy \supset Fya)$

e. $(\forall x)(\forall y)[(\exists z)(Fyz \,\&\, \sim Fzx) \supset Gxy]$
$\sim (\exists x)Gxx$
$(\forall z)(Faz \supset Fza)$

*f. $(\forall x)(\forall y)(Dxy \supset Cxy)$
$(\forall x)(\exists y)Dxy$
$(\forall x)(\forall y)(Cxy \supset Cyx)$
$(\exists x)(\exists y)(Cxy \,\&\, Cyx)$

g. $(\forall x)(Fx \supset (\exists y)Gxy$
$(\forall x)(\forall y) \sim Gxy$
$(\forall x) \sim Fx$

*h. $(\forall x)(Fx \supset (\exists y)Gxy)$
$(\forall x)(\forall y)(Gxy \supset Hxy)$
$\sim (\exists x)(\exists y)Hxy$
$\sim (\exists x)Fx$

7. Prove that the following sentences of *PL* are theorems of *PD:*

a. $(\forall x)(\exists z)(Fxz \supset Fzx)$
*b. $(\forall x)Fxx \supset (\forall x)(\exists y)Fxy$
c. $(\forall x)(\forall y)Gxy \supset (\forall z)Gzz$
*d. $(\exists x)Fxx \supset (\exists x)(\exists y)Fxy$
e. $(\forall x)Lxx \supset (\exists x)(\exists y)(Lxy \,\&\, Lyx)$
*f. $(\exists x)(\forall y)Lxy \supset (\exists x)Lxx$
#g. $(\exists x)(\forall y)Fxy \supset (\exists x)(\exists y)Fxy$
*h. $(\forall x)(Fx \supset (\exists y)Gya) \supset (Fb \supset (\exists y)Gya)$
i. $(\exists x)(\exists y)(Lxy \equiv Lyx)$
*j. $(\exists x)(\forall y)Hxy \supset (\forall y)(\exists x)Hxy$
k. $(\forall x)(\forall y)(\forall z)Gxyz \supset (\forall x)(\forall y)(\forall z)(Gxyz \supset Gzyx)$
*l. $(\forall x)(Fx \supset (\exists y)Gyx) \supset ((\exists x)Fx \supset (\exists x)(\exists y)Gxy)$
m. $(\forall x)(\forall y)(Fxy \equiv Fyx) \supset \sim (\exists x)(\exists y)(Fxy \,\&\, \sim Fyx)$
*n. $(\exists x)(Fx \supset (\forall y)Fy)$

8. Construct derivations that establish that the following pairs of sentences are equivalent in *PD:*

a. $(\forall x)(Fx \supset (\exists y)Gya)$ $\quad$ $(\exists x)Fx \supset (\exists y)Gya$
*b. $(\forall x)(Fx \supset (\forall y)Gy)$ $\quad$ $(\forall x)(\forall y)(Fx \supset Gy)$
#c. $(\exists x)[Fx \supset (\forall y)Hxy]$ $\quad$ $(\exists x)(\forall y)(Fx \supset Hxy)$
*d. $(\forall x)(\forall y)(Fxy \supset Gy)$ $\quad$ $(\forall y)[(\exists x)Fxy \supset Gy]$
e. $(\forall x)(\forall y)(Fxy \equiv \sim Gyx)$ $\quad$ $(\forall x)(\forall y) \sim (Fxy \equiv Gyx)$

9. Construct derivations that establish that the following sets are inconsistent in *PD:*

a. $\{(\forall x)(\forall y)[(Ex \,\&\, Ey) \supset Txy],\ (Ea \,\&\, Eb) \,\&\, \sim Tab\}$
*b. $\{(\forall x)(\exists y)Lyx,\ \sim (\exists x)Lxb\}$

c. {~ (∃x)Fxx, (∃x)(∀y)Fxy}
*d. {(∀x)(∀y)(Fxy ⊃ Fyx), Fab, ~ (∃z)Fza}
e. {(∀x)(∃y)Lxy, (∀y) ~ Lay}
*f. {(∃x)(∀y)Gxy, ~ (∀y)(∃x)Gxy}
g. {(∀x)[Hx ⊃ (∃y)Lyx], (∃x) ~ (∃y)Lyx, (∀x)Hx}
*h. {~ (∃x)Fxx, (∀x)[(∃y)Fxy ⊃ Fxx], (∃x)(∃y)Fxy}
#i. {(∀x)(∃y)Fxy, (∃z) ~ (∃w)Fzw}
*j. {(∀x)(∀y)(Gxy ≡ Gyx), (∃x)(∃y)(Gxy & ~ Gyx)}
k. {(∀x)(∀y)(Fxy ∨ Gxy), (∃x)(∃y)(~ Fxy & ~ Gxy)}
*l. {(∀x)(Fx ⊃ [(∃y)Gy ⊃ (∀y)Gy]), (∃x)(Fx & Gx), (∃y) ~ Gy}

10.3 THE DERIVATION SYSTEM *PD*+

PD+ is a derivation system that includes all the rules of *PD,* the rules that distinguish *SD*+ from *SD,* and one additional rule of replacement. *PD*+ is no stronger than *PD;* however, derivations in *PD*+ are often shorter than the corresponding derivations in *PD*. The rules of replacement in *PD*+ apply to subformulas of sentences as well as to complete sentences. In the following example each of the replacement rules has been applied to a subformula of the sentence on the previous line:

1	(∀x)[(Fx & Hx) ⊃ (∃y)Nxy]	Assumption
2	(∀x)[~ (Fx & Hx) ∨ (∃y)Nxy]	1 Impl
3	(∀x)[~ (Fx & Hx) ∨ ~ ~ (∃y)Nxy]	2 DN
4	(∀x) ~ [(Fx & Hx) & ~ (∃y)Nxy]	3 DeM
5	(∀x) ~ [(Hx & Fx) & ~ (∃y)Nxy]	4 Com

Here Implication was applied to the subformula '(Fx & Hx) ⊃ (∃y)Nxy' of the sentence on line 1 to produce the subformula '~ (Fx & Hx) ∨ (∃y)Nxy' of the sentence on line 2. Double Negation was applied to the subformula '(∃y)Nxy' of the sentence on line 2, to produce the subformula '~ ~ (∃y)Nxy' of the sentence on line 3. De Morgan was applied to the subformula '~ (Fx & Hx) ∨ ~ ~ (∃y)Nxy' of the sentence on line 3 to produce the subformula '~ [(Fx & Hx) & ~ (∃y)Nxy]' of the sentence on line 4. Finally, Commutation was applied to the subformula 'Fx & Hx' of the sentence on line 4 to produce the 'Hx & Fx' of the sentence on line 5.

In applying rules of replacement in *PD*+ it is important to correctly identify subformulas of sentences. Consider the following:

1	(∀x)[Lx ∨ (∃y)(Bxy ∨ Jxy)]	Assumption	
2	(∀x)[(Lx ∨ (∃y)Bxy) ∨ Jxy]	1 Assoc	**MISTAKE!**

Line 2 is a mistake because the immediate subformula of the sentence on line 1 is not of the form **P** ∨ (**Q** ∨ **R**). Rather, it is of the form **P** ∨ (∃**x**)(**Q** ∨ **R**).

In addition to the rules of replacement of *SD*+, *PD*+ contains **Quantifier Negation.** Where **P** is an open sentence of *PL* in which **x** occurs free, the rule is

Quantifier Negation (QN)

$\sim(\forall \mathbf{x})\mathbf{P} \dashv\vdash (\exists \mathbf{x}) \sim \mathbf{P}$
$\sim(\exists \mathbf{x})\mathbf{P} \dashv\vdash (\forall \mathbf{x}) \sim \mathbf{P}$

As with all rules of replacement, Quantifier Negation can be applied to subformulas within a sentence, as well as to an entire sentence. All these are proper uses of Quantifier Negation:

1	$\sim (\exists y) \sim (\forall x)(Fx \supset (\exists z) \sim Gxy)$	Assumption
2	$(\forall y) \sim \sim (\forall x)(Fx \supset (\exists z) \sim Gxy)$	1 QN
3	$(\forall y) \sim (\exists x) \sim (Fx \supset (\exists z) \sim Gxy)$	2 QN
4	$(\forall y) \sim (\exists x) \sim (Fx \supset \sim (\forall z)Gxy)$	3 QN

The definitions of the basic concepts of *PD*+ strictly parallel the definitions of the basic concepts of *PD,* in all cases replacing '*PD*' with '*PD*+'. Consequently the tests for the various syntactic properties are carried out in the same way. The important difference between *PD* and *PD*+ is that *PD,* with fewer rules, provides theoretical elegance and *PD*+, with more rules, provides practical ease.

In Section 10.2 we proved that '$(\exists x)(Fx \supset (\forall y)Fy)$' is a theorem in *PD*. Our derivation was 17 lines long. We repeat it here.

Derive: $(\exists x)(Fx \supset (\forall y)Fy)$

1	$\sim (\exists x)(Fx \supset (\forall y)Fy)$	A / ~ E
2	Fa	A / ⊃I
3	$\sim Fb$	A / ~ I
4	Fb	A / ⊃I
5	$\sim (\forall y)Fy$	A / ~ E
6	Fb	4 R
7	$\sim Fb$	3 R
8	$(\forall y)Fy$	5–7 ~ E
9	$Fb \supset (\forall y)Fy$	4–8 ⊃I
10	$(\exists x)(Fx \supset (\forall y)Fy)$	9 ∃I
11	$\sim (\exists x)(Fx \supset (\forall y)Fy)$	1 R
12	Fb	3–11 ~ E
13	$(\forall y)Fy$	12 ∀I
14	$Fa \supset (\forall y)Fy$	2–13 ⊃I
15	$(\exists x)(Fx \supset (\forall y)Fy)$	14 ∃I
16	$\sim (\exists x)(Fx \supset (\forall y)Fy)$	1 R
17	$(\exists x)(Fx \supset (\forall y)Fy)$	1–16 ~ E

We can show that this sentence is a theorem in *PD*+ in just 10 lines:

Derive: $(\exists x)(Fx \supset (\forall y)Fy)$

1	$\sim (\exists x)(Fx \supset (\forall y)Fy)$	A / ~ E
2	$(\forall x) \sim (Fx \supset (\forall y)Fy)$	1 QN
3	$\sim (Fa \supset (\forall y)Fy)$	2 ∀E
4	$\sim (\sim Fa \vee (\forall y)Fy)$	3 Impl
5	$\sim \sim Fa \,\&\, \sim (\forall y)Fy$	4 DeM
6	$\sim \sim Fa$	5 &E
7	Fa	6 DN
8	$\sim (\forall y)Fy$	5 &E
9	$(\forall y)Fy$	7 ∀I
10	$(\exists x)(Fx \supset (\forall y)Fy)$	1–9 ~ E

In Section 10.2 it took us 19 lines to derive '$(\exists x)(Fx \supset Ga)$' from $\{(\forall x)Fx \supset Ga\}$. We repeat our derivation here:

Derive: $(\exists x)(Fx \supset Ga)$

1	$(\forall x)Fx \supset Ga$	Assumption
2	$\sim (\exists x)(Fx \supset Ga)$	A / ~ E
3	Fa	A / ⊃I
4	$\sim Fb$	A / ~ E
5	Fb	A / ⊃I
6	$\sim Ga$	A / ~ E
7	Fb	5 R
8	$\sim Fb$	4 R
9	Ga	6–8 ~ E
10	$Fb \supset Ga$	5–9 ⊃I
11	$(\exists x)(Fx \supset Ga)$	10 ∃I
12	$\sim (\exists x)(Fx \supset Ga)$	2 R
13	Fb	4–12 ~ E
14	$(\forall x)Fx$	13 ∀I
15	Ga	1, 14 ⊃E
16	$Fa \supset Ga$	3–15 ⊃I
17	$(\exists x)(Fx \supset Ga)$	16 ∃I
18	$\sim (\exists x)(Fx \supset Ga)$	1 R
19	$(\exists x)(Fx \supset Ga)$	1–18 ~ E

We can derive '$(\exists x)(Fx \supset Ga)$' from $\{(\forall x)Fx \supset Ga\}$ in just 12 lines in *PD*+:

Derive: $(\exists x)(Fx \supset Ga)$

1	$(\forall x)Fx \supset Ga$	Assumption
2	$\sim (\exists x)(Fx \supset Ga)$	A / ~ E
3	$(\forall x) \sim (Fx \supset Ga)$	2 QN
4	$\sim (Fb \supset Ga)$	3 ∀E
5	$\sim (\sim Fb \vee Ga)$	4 Impl
6	$\sim \sim Fb \,\&\, \sim Ga$	5 DeM
7	$\sim \sim Fb$	6 &E
8	Fb	7 DN
9	$(\forall x)Fx$	8 ∀I
10	Ga	1, 9 ⊃E
11	$\sim Ga$	6 &E
12	$(\exists x)(Fx \supset Ga)$	2–11 ~ E

10.3**E** EXERCISES

1. Show that each of the following derivability claims holds in *PD*+.

a. $\{\sim (\forall y)(Fy \,\&\, Gy)\} \vdash (\exists y)(\sim Fy \vee \sim Gy)$

*b. $\{(\forall w)(Lw \supset Mw),\ (\forall y)(My \supset Ny)\} \vdash (\forall w)(Lw \supset Nw)$

c. $\{(\exists z)(Gz \,\&\, Az),\ (\forall y)(Cy \supset \sim Gy)\} \vdash (\exists z)(Az \,\&\, \sim Cz)$

*d. $\{\sim (\exists x)(\sim Rx \,\&\, Sxx),\ Sjj\} \vdash Rj$

e. $\{(\forall x)[(\sim Cxb \vee Hx) \supset Lxx],\ (\exists y) \sim Lyy\} \vdash (\exists x)Cxb$

*f. $\{(\forall x)Fx,\ (\forall z)Hz\} \vdash \sim (\exists y)(\sim Fy \vee \sim Hy)$

2. Show that each of the following arguments is valid in *PD*+.

a. $(\forall x) \sim Jx$

$(\exists y)(Hby \vee Ryy) \supset (\exists x)Jx$

$(\forall y) \sim (Hby \vee Ryy)$

*b. $\sim (\exists x)(\forall y)(Pxy \,\&\, \sim Qxy)$

$(\forall x)(\exists y)(Pxy \supset Qxy)$

c. $(\forall x) \sim ((\forall y)Hyx \vee Tx)$

$\sim (\exists y)(Ty \vee (\exists x) \sim Hxy)$

$(\forall x)(\forall y)Hxy \,\&\, (\forall x) \sim Tx$

*d. $(\forall z)(Lz \equiv Hz)$

$(\forall x) \sim (Hx \vee \sim Bx)$

$\sim Lb$

e. $(\forall z)[Kzz \supset (Mz \,\&\, Nz)]$

$(\exists z) \sim Nz$

$(\exists x) \sim Kxx$

*f. $(\exists x)[\sim Bxm \;\&\; (\forall y)(Cy \supset \sim Gxy)]$
$(\forall z)[\sim (\forall y)(Wy \supset Gzy) \supset Bzm]$

$(\forall x)(Cx \supset \sim Wx)$

g. $(\exists z)Qz \supset (\forall w)(Lww \supset \sim Hw)$
$(\exists x)Bx \supset (\forall y)(Ay \supset Hy)$

$(\exists w)(Qw \;\&\; Bw) \supset (\forall y)(Lyy \supset \sim Ay)$

*h. $(\forall y)(Kby \supset \sim Hy)$

$(\forall x)[(\exists y)(Kby \;\&\; Qxy) \supset (\exists z)(\sim Hz \;\&\; Qxz)]$

i. $\sim (\forall x)(\sim Gx \vee \sim Hx) \supset (\forall x)[Cx \;\&\; (\forall y)(Ly \supset Axy)]$
$(\exists x)[Hx \;\&\; (\forall y)(Ly \supset Axy)] \supset (\forall x)(Fx \;\&\; (\forall y)Bxy)$

$\sim (\forall x)(\forall y)Bxy \supset (\forall x)(\sim Gx \vee \sim Hx)$

3. Show that each of the following sentences is a theorem in *PD*+.

a. $(\forall x)(Ax \supset Bx) \supset (\forall x)(Bx \vee \sim Ax)$
*b. $(\forall x)(Ax \supset (Ax \supset Bx)) \supset (\forall x)(Ax \supset Bx)$
c. $\sim (\exists x)(Ax \vee Bx) \supset (\forall x) \sim Ax$
*d. $(\forall x)(Ax \supset Bx) \vee (\exists x)Ax$
e. $((\exists x)Ax \supset (\exists x)Bx) \supset (\exists x)(Ax \supset Bx)$
*f. $(\forall x)(\exists y)(Ax \vee By) \equiv (\exists y)(\forall x)(Ax \vee By)$

4. Show that the members of each of the following pairs of sentences are equivalent in *PD*+.

a. $\sim (\forall x)(Ax \supset Bx)$	$(\exists x)(Ax \;\&\; \sim Bx)$
*b. $(\exists x)(\exists y)Axy \supset Aab$	$(\exists x)(\exists y)Axy \equiv Aab$
c. $\sim (\forall x) \sim [(Ax \;\&\; Bx) \supset Cx]$	$(\exists x)[\sim Ax \vee (\sim Cx \supset \sim Bx)]$
*d. $\sim (\forall x)(\exists y)[(Ax \;\&\; Bx) \vee Cy]$	$(\exists x)(\forall y)[\sim (Cy \vee Ax) \vee \sim (Cy \vee Bx)]$
e. $(\forall x)(Ax \equiv Bx)$	$\sim (\exists x)[(\sim Ax \vee \sim Bx) \;\&\; (Ax \vee Bx)]$
*f. $(\forall x)(Ax \;\&\; (\exists y) \sim Bxy)$	$\sim (\exists x)[\sim Ax \vee (\forall y)(Bxy \;\&\; Bxy)]$

5. Show that each of the following sets of sentences is inconsistent in *PD*+.

a. $\{[(\forall x)(Mx \equiv Jx) \;\&\; \sim Mc] \;\&\; (\forall x)Jx\}$
*b. $\{\sim Fa, \sim (\exists x)(\sim Fx \vee \sim Fx)\}$
c. $\{(\forall x)(\forall y)Lxy \supset \sim (\exists z)Tz,\ (\forall x)(\forall y)Lxy \supset ((\exists w)Cww \vee (\exists z)Tz),$
$(\sim (\forall x)(\forall y)Lxy \vee (\forall z)Bzzk) \;\&\; (\sim (\forall z)Bzzk \vee \sim (\exists w)Cww),\ (\forall x)(\forall y)Lxy\}$
*d. $\{(\exists x)(\forall y)(Hxy \supset (\forall w)Jww),\ (\exists x) \sim Jxx \;\&\; \sim (\exists x) \sim Hxm\}$
e. $\{(\forall x)(\forall y)(Gxy \supset Hc),\ (\exists x)Gix \;\&\; (\forall x)(\forall y)(\forall z)Lxyz,\ \sim Lcib \vee \sim (Hc \vee Hc)\}$
*f. $\{(\forall x)[(Sx \;\&\; Bxx) \supset Kax],\ (\forall x)(Hx \supset Bxx),\ (\exists x)(Sx \;\&\; Hx),$
$(\forall x) \sim (Kax \;\&\; Hx)\}$

6. a. Show that Universal Introduction and Universal Elimination are eliminable in *PD*+ by developing routines that can be used in place of these rules to obtain the same results. (*Hint:* Consider using Quantifier Negation, Existential Introduction, and Existential Elimination.)

*b. Show that Existential Introduction and Existential Elimination are eliminable in *PD*+ by developing routines that can be used in place of these rules to obtain the same results. (*Hint:* Consider using Quantifier Negation, Universal Introduction, and Universal Elimination.)

10.4 THE DERIVATION SYSTEM *PDE*

The symbolic language *PLE* extends *PL* to include sentences that contain functors and the identity predicate. Accordingly we need to extend the derivation system *PD* developed earlier in this chapter to allow for derivations that include these new sentences of *PLE*. We shall do so by adding an introduction rule and an elimination rule for the identity predicate, and then modifying the quantifier rules so as to allow for sentences containing functors. The resulting *extended* predicate derivation system is called *PDE*.

The introduction rule for '=' is

Identity Introduction (=I)

$\triangleright \mid (\forall \mathbf{x})\mathbf{x} = \mathbf{x}$

Identity Introduction is unlike other introduction rules in that it appeals to no previous line or lines of the derivation. Rather, it allows sentences of the specified form to be entered on any line of any derivation, no matter what sentences, if any, occur earlier in the derivation.[1] Identity Introduction is truth-preserving because every sentence that can be introduced by it, that is every sentence of the form $(\forall \mathbf{x})\mathbf{x} = \mathbf{x}$, is quantificationally true. These sentences simply say of each thing that it is identical to itself. Here is a very simple derivation of a theorem using the rule Identity Introduction:

Derive: a = a

1	(∀y)y = y	=I
2	a = a	1 ∀E

Notice that the sentence on line 1 is *not an assumption.*

The elimination rule for "=" is

Identity Elimination (=E)

	$\mathbf{t}_1 = \mathbf{t}_2$			$\mathbf{t}_1 = \mathbf{t}_2$
	$\mathbf{P}$	or		$\mathbf{P}$
$\triangleright$	$\mathbf{P}(\mathbf{t}_1//\mathbf{t}_2)$		$\triangleright$	$\mathbf{P}(\mathbf{t}_2//\mathbf{t}_1)$

where $\mathbf{t}_1$ and $\mathbf{t}_2$ are closed terms.

The notation

$\mathbf{P}(\mathbf{t}_1//\mathbf{t}_2)$

is read '**P** with one or more occurrences of $\mathbf{t}_2$ replaced by $\mathbf{t}_1$'. Similarly $\mathbf{P}(\mathbf{t}_2//\mathbf{t}_1)$ is read '**P** with one or more occurrences of $\mathbf{t}_1$ replaced by $\mathbf{t}_2$'. Recall that the closed terms of *PLE* are the individual constants together with complex terms

[1] Metaformulas (such as '$(\forall \mathbf{x})\mathbf{x}=\mathbf{x}$') that specify sentences that can be introduced without reference to previous sentences occurring in a derivation are usually called **axiom schemas.** An axiom schema is a metaformula such that every formula having its form may be entered in a derivation. Some derivation systems rely primarily on axiom schemas; these are called **axiomatic systems.**

such as 'f(a,b)' and 'f(g(a,b),c)' that contain no variables. Identity Elimination permits the replacement of one closed term with another in a sentence only if those closed terms designate the same thing ($\mathbf{t}_1 = \mathbf{t}_2$ says that $\mathbf{t}_1$ and $\mathbf{t}_2$ do designate the same thing). The following simple examples illustrate the use of this rule:

Derive: Hda

1	c = d	Assumption
2	Hca	Assumption
3	Hda	1, 2 =E

The following three derivations are very similar but not identical:

Derive: $(\forall x)(Fxh \supset Ghx)$

1	h = e	Assumption
2	$(\forall y)(Fye \supset Gey)$	Assumption
3	$(\forall y)(Fyh \supset Ghy)$	1, 2 =E
4	$Fah \supset Gha$	3 $\forall$E
5	$(\forall x)(Fxh \supset Ghx)$	4 $\forall$I

Derive: $(\forall x)(Fxe \supset Ghx)$

1	h = e	Assumption
2	$(\forall y)(Fye \supset Gey)$	Assumption
3	$(\forall y)(Fye \supset Ghy)$	1, 2 =E
4	$Fae \supset Gha$	3 $\forall$E
5	$(\forall x)(Fxe \supset Ghx)$	4 $\forall$I

Derive: $(\forall x)(Fxh \supset Gex)$

1	h = e	Assumption
2	$(\forall y)(Fye \supset Gey)$	Assumption
3	$(\forall y)(Fyh \supset Gey)$	1, 2 =E
4	$Fah \supset Gea$	3 $\forall$E
5	$(\forall x)(Fxh \supset Gex)$	4 $\forall$I

In the first derivation we replaced, at line 3, both occurrences of 'e' in line 2 with 'h'. In the second derivation we replaced, at line 3, only the second occurrence of 'e' in line 2 with 'h'. And in the third derivation we replaced, at line 3, only the first occurrence of 'e' in line 2 with 'h'. All of these are appropriate uses of Identity Elimination, as are the following:

Derive: Hc

1	$(\forall x)H f(a,x)$	Assumption
2	$c = f(a,b)$	Assumption
3	$Hf(a,b)$	1 $\forall$E
4	Hc	2, 3 =E

Derive: Wab

1	Haa ⊃ Waa	Assumption
2	Hab	Assumption
3	a = b	Assumption
4	Hab ⊃ Wab	1, 3 =E
5	Wab	2, 4 ⊃E

Note that from lines 1 through 3 we can obtain, by Identity Elimination, not just 'Hab ⊃ Wab' but a host of additional sentences, including those on lines 5 through 8 below:

1	Haa ⊃ Waa	Assumption
2	Hab	Assumption
3	a = b	Assumption
4	Hab ⊃ Wab	1, 3 =E
5	Hbb ⊃ Wbb	1, 3 =E
6	Haa ⊃ Wbb	1, 3 =E
7	Hbb ⊃ Waa	1, 3 =E
8	Hba ⊃ Wba	1, 3 =E

But these additional sentences do not advance us toward our goal of 'Wab'. There are alternative ways of deriving 'Wab'. Here is one:

Derive: Wab

1	Haa ⊃ Waa	Assumption
2	Hab	Assumption
3	a = b	Assumption
4	Haa	2, 3 =E
5	Waa	1, 4 ⊃E
6	Wab	3, 5 =E

Consider next these derivations:

Derive: Had

1	c = d	Assumption
2	Hac	Assumption
3	Had	1, 2 =E

Derive: (∀x) (Fxh ⊃ Ghx)

1	h = e	Assumption
2	(∀y) (Fye ⊃ Gey)	Assumption
3	(∀y) (Fyh ⊃ Ghy)	1, 2 =E
4	Fah ⊃ Gha	3 ∀E
5	(∀x) (Fxh ⊃ Ghx)	4 ∀I

Derive: Hc

1	(∀x)H*f*(a,x)	Assumption
2	c = *f*(a,b)	Assumption
3	H*f*(a,b)	1 ∀E
4	Hc	2, 3 =E

The sentence '(a = b & b = c) ⊃ a = c' says that if a is identical to b, and b is identical to c, then a is identical to c. As expected, it is a theorem of *PDE*. Here is a proof:

Derive: (a = b & b = c) ⊃ a = c

1	a = b & b = c	A / ⊃I
2	a = b	1 &E
3	b = c	1 &E
4	a = c	2, 3 =E
5	(a = b & b = c) ⊃ a = c	1–4 ⊃I

In considering this example one might well ask whether the justification for line 4 indicates that we have replaced 'b' in line 3 with 'a', based on the identity at line 2, or that we replaced 'b' in line 2 with 'c' based on the identity at line 3. Fortunately, both replacements are allowed so the justification can be understood either way.

As we have already seen, sentences of the form $\mathbf{t}_1 = \mathbf{t}_1$ are normally obtained by Identity Introduction, as in

1	b = b ⊃ Fb	Assumption
2	(∀x)x = x	=I
3	b = b	2 ∀E
4	Fb	1, 3 ⊃E

In special circumstances we can obtain a sentence of the form **a** = **a** by Identity Elimination. This happens when the **a** of **a** = **a** already occurs in an accessible identity sentence. Here is an example:

1	b = b ⊃ Fb	Assumption
2	a = b	Assumption
3	b = b	2, 2 =E
4	Fb	1, 3 ⊃E

Identity Elimination allows us, given a sentence of the form $\mathbf{t}_1 = \mathbf{t}_2$, to replace any occurrence of $\mathbf{t}_1$ with $\mathbf{t}_2$ in any sentence that contains $\mathbf{t}_1$, and vice versa. In our example we have the identity sentence 'a = b' and that very sentence contains 'a', so we can replace the 'a' in 'a = b' with 'b', and we do so at line 3.

As we saw in Chapter 7, the identity predicate is useful in symbolizing sentences containing definite descriptions. Consider the argument:

The Roman general who defeated Pompey conquered Gaul.
Julius Caesar is a Roman general, and he defeated Pompey.

Julius Caesar conquered Gaul.

This argument can be symbolized in *PLE* as:

$(\exists x)[((Rx \ \& \ Dxp) \ \& \ (\forall y)[(Ry \ \& \ Dyp) \supset y = x]) \ \& \ Cxg]$
$Rj \ \& \ Djp$

Cjg

This argument is valid, for if there is one and only one thing that is a Roman general and defeated Pompey, and if Julius Caesar is a Roman general who defeated Pompey, then Caesar is *the* Roman general who defeated Pompey, and is therefore someone who conquered Gaul. We can show this argument is valid in *PDE:*

Derive: Cjg

1	$(\exists x)\ [((Rx \ \& \ Dxp) \ \& \ (\forall y)[(Ry \ \& \ Dyp) \supset y = x]) \ \& \ Cxg]$	Assumption
2	$Rj \ \& \ Djp$	Assumption
3	$((Ra \ \& \ Dap) \ \& \ (\forall y)[(Ry \ \& \ Dyp) \supset y = a]) \ \& \ Cag$	A / $\exists$E
4	$(Ra \ \& \ Dap) \ \& \ (\forall y)[(Ry \ \& \ Dyp) \supset y = a]$	3 &E
5	$(\forall y)[(Ry \ \& \ Dyp) \supset y = a]$	4 &E
6	$(Rj \ \& \ Djp) \supset j = a$	5 $\forall$E
7	$j = a$	2, 6 $\supset$E
8	Cag	3 &E
9	Cjg	7, 8 =E
10	Cjg	1, 3–9 $\exists$E

Here is another argument that involves a definite description.

The primary author of the Declaration of Independence was a slave owner.
Thomas Jefferson was the primary author of the Declaration of Independence.

Thomas Jefferson was a slave owner.

The conclusion of this argument can be symbolized as 'Ot' where 'Ox' is interpreted as 'x owns at least one slave' and 't' designates Thomas Jefferson. To symbolize the premises we need a way of saying there was one and only one primary author of the Declaration of Independence. We can do so as follows:

$(\exists x)[Px \ \& \ (\forall z)(Pz \supset z = x)]$

We are here using 'Px' for 'x is a primary author of the Declaration of Independence'. This sentence of *PL* can be read as 'There is at least one thing x that is a primary author of the Declaration of Independence and each thing z that is a primary author of the Declaration of Independence is identical to x.' The full argument can now be symbolized as:

$$\frac{(\exists x)([Px \ \& \ (\forall z)(Pz \supset z = x)] \ \& \ Ox) \qquad Pt \ \& \ (\forall z)(Pz \supset z = t)}{Ot}$$

We can construct a derivation that establishes that the above argument is valid in *PDE*. Here is a start:

Derive: Ot

1		$(\exists x)([Px \ \& \ (\forall z)(Pz \supset z = x)] \ \& \ Ox)$	Assumption
2		$Pt \ \& \ (\forall z)(Pz \supset z = t)$	Assumption
3	|	$[Pa \ \& \ (\forall z)(Pz \supset z = a)] \ \& \ Oa$	A / ∃E
G	|	Ot	
G		Ot	1, 2–__ ∃E

Our intent is to derive the final goal using Existential Elimination. If we can derive 'Ot' within the Existential Elimination subderivation we will be able to move it out of that subderivation because 't' is not the instantiating constant in our assumption at line 3 (it is for this reason that we picked a constant other than 't' as our instantiating constant at line 3). 'Oa' can be derived immediately from line 3 by Conjunction Elimination. What remains is to get to a point where we can use Identity Elimination to infer 'Ot' from 'Oa' and an appropriate identity sentence, either 'a = t' or 't = a'.

Derive: Ot

1		$(\exists x)([Px \ \& \ (\forall z)(Pz \supset z = x)] \ \& \ Ox)$	Assumption
2		$Pt \ \& \ (\forall z)(Pz \supset z = t)$	Assumption
3	|	$[Pa \ \& \ (\forall z)(Pz \supset z = a)] \ \& \ Oa$	A / ∃E
4	|	Oa	3 &E
G	|	a = t	
G	|	Ot	4, __ =E
G		Ot	1, 2–__ ∃E

Identity sentences are obtainable both from line 2 and from line 3. This suggests two strategies, and both will work. First we will try to obtain 'a = t'. We start by obtaining '(∀z)(Pz ⊃ z = t)' from line 2 by Conjuction Elimination and then 'Pa ⊃ a = t' by Universal Elimination. And 'Pa' is available from line 3 by two uses of Conjunction Elimination. This will allow us to complete the derivation:

Derive: Ot

1	(∃x)([Px & (∀z)(Pz ⊃ z = x) & Ox)	Assumption
2	Pt & (∀z)(Pz ⊃ z = t)	Assumption
3	[Pa & (∀z)(Pz ⊃ z = a)] & Oa	A / ∃E
4	Oa	3 &E
5	(∀z)(Pz ⊃ z = t)	2 &E
6	Pa ⊃ a = t	5 ∀E
7	Pa & (∀z)(Pz ⊃ z = a)	3 &E
8	Pa	7 &E
9	a = t	6, 8 ⊃E
10	Ot	4, 9 =E
11	Ot	1, 3–10 ∃E

We could also have completed our derivation by deriving the identity sentence 't = a' as follows:

Derive: Ot

1	(∃x)([Px & (∀z)(Pz ⊃ z = x)] & Ox	Assumption
2	Pt & (∀z)(Pz ⊃ z = t)	Assumption
3	[Pa & (∀z)(Pz ⊃ z = a)] & Oa	A / ∃E
4	Oa	3 &E
5	Pa & (∀z)(Pz ⊃ z = a)	3 &E
6	(∀z)(Pz ⊃ z = a)	5 &E
7	Pt ⊃ t = a	6 ∀E
8	Pt	2 &E
9	t = a	7, 8 ⊃E
10	Ot	4, 9 =E
11	Ot	1, 3–10 ∃E

When we formulated Identity Elimination we did so in a way that allows for the presence of complex terms in *PDE*. Two of our quantifier rules, Existential Introduction and Universal Elimination, need to be modified so that they too allow for the presence of complex terms. The other rules of *PD* function without modification as part of *PDE*. We recast Existential Introduction and Universal Elimination as follows:

Existential Introduction (∃I)

	P(**t**/**x**)
▷	(∃**x**)**P**

where **t** is any closed term

Universal Elimination (∀E)

	$(\forall \mathbf{x})\mathbf{P}$
▷	$\mathbf{P}(\mathbf{t}/\mathbf{x})$

where **t** is any closed term

Consider the following simple derivations:

Derive: $(\exists z)Fz$

1	$(\forall y)Fy$	Assumption
2	Fa	1 ∀E
3	$(\exists z)Fz$	2 ∃I

Derive: $(\exists z)Fg(z)$

1	$(\forall y)Fy$	Assumption
2	$Fg(a)$	1 ∀E
3	$(\exists z)Fg(z)$	2 ∃I

In the first derivation 'Fa' is the substitution instance associated with both the use of Universal Elimination *and* the use of Existential Introduction. In the terminology of previous sections, 'a' is the instantiating constant for these uses of the two rules. In the second derivation '$Fg(a)$' is the substitution instance associated with both the use of Universal Elimination and the use of Existential Introduction. However, the instantiating term in the use of Universal Elimination is '$g(a)$' (we have replaced 'y' with '$g(a)$') whereas the instantiating term in the use of Existential Introduction is 'a', *not* '$g(a)$' (we replaced the constant 'a' with the variable 'z'). Since the individual term used to form substitution instances associated with the quantifier rules is sometimes an individual constant and sometimes a closed complex term, we will hereafter speak, with reference to substitution instances and uses of Existential Introduction and Universal Elimination, of the **instantiating term** rather than the instantiating constant.

But we will not modify Existential Elimination and Universal Introduction so as to allow substitution instances used in these rules to be formed from complex terms and so we will continue to talk, with reference to these latter rules, only of the **instantiating constant.** To understand why we will not modify Universal Introduction to allow for complex instantiating terms, consider the following attempt at a derivation:

Derive: $(\forall x)Ex$

1	$(\forall x)Ed(x)$	Assumption	
2	$Ed(a)$	1 ∀E	
3	$(\forall x)Ex$	2 ∀I	**MISTAKE!**

If this were a legitimate derivation in *PDE* then the following argument would be valid in *PDE:*

$$\frac{(\forall x)\mathrm{E}d(x)}{(\forall x)\mathrm{E}x}$$

We do not want this argument to be valid in *PDE*. If our UD is the set of positive integers and we interpret 'Ex' as 'x is even' and 'd(x)' as 'x times 2', the premise says that each positive integer is such that 2 times that integer is even, which is true. The conclusion says that each positive integer is even, which is false. The problem is in the attempted inference of line 3 from line 2. The expression 'd(a)' cannot designate an arbitrarily selected member of the UD; rather it can refer only to a member of the UD that is the value of the function d for some member a of the UD. On the interpretation given previously, for example, 'd(a)' can only refer to even numbers.

For similar reasons, we continue to require that in using Existential Elimination the instantiating term must be an individual constant, not a closed complex term. Here is a failed derivation that would be allowed if we dropped this requirement:

Derive: $(\exists x)\mathrm{O}d(x)$

1	$(\exists x)\mathrm{O}x$	Assumption	
2	$\mathrm{O}d(a)$	A / $\exists$E	**MISTAKE!**
3	$(\exists x)\mathrm{O}d(x)$	2 $\exists$I	
4	$(\exists x)\mathrm{O}d(x)$	1, 2–3 $\exists$E	**MISTAKE!**

To see why we do not want this derivation to go through suppose we again use the set of positive integers as our UD and interpret 'Ox' as 'x is odd' and 'd(x)' as 'x times 2'. Then the primary assumption says that there is a positive integer that is odd, which is true. The sentence on line 4 says there is an integer that is 2 times some positive integer and that is odd, and this is false. The problem is that the assumption on line 2 contains information about the individual that is assumed to have property O—namely that it is the value of the function d for some member of the UD, while the existentially quantified sentence on line 1 does not contain this information. The requirement that the assumption for an Existential Elimination subderivation be a substitution instance formed from a constant guarantees that the assumption does not contain information that is absent from the existentially quantified sentence. Hence we continue to require that in using Existential Elimination the assumed substitution instance must be formed using an individual constant.

Having said that, it is important to note that while for Universal Introduction and Existential Elimination the instantiating term must be a constant,

the substitution instances associated with these rules may contain complex terms. For example, the following is a correctly done derivation:

Derive: $(\forall y)\mathrm{E}d(y)$

1	$(\forall x)\mathrm{E}x$	Assumption
2	$\mathrm{E}d(a)$	1 ∀∃
3	$(\forall y)\mathrm{E}d(y)$	2 ∀I

Here 'a' is the instantiating constant for the use of Universal Introduction: In moving from line 2 to line 3 we replaced 'a' with 'y'. But 'd(a)' is the instantiating term associated with Universal Elimination. In moving from line 1 to line 2 we replace 'x' with 'd(a)'. So 'Ed(a)' is a substitution instance of '$(\forall x)$ Ex' because it is the result of replacing every occurrence of 'x' in 'Ex' with 'd(a)' and 'Ed(a)' is a substitution instance of '$(\forall y)\mathrm{E}d(y)$' because it is the result of replacing every occurrence of 'y' in 'Ed(y)' with 'a'.

And the following is an allowed use of Existential Elimination:

Derive:

1	$(\exists x)\mathrm{F}g(x)$	Assumption
2	$\mathrm{F}g(b)$	A / ∃E
3	$(\exists z)\mathrm{F}z$	2 ∃I
4	$(\exists z)\mathrm{F}z$	1, 2–3 ∃E

Here 'Fg(b)' is a substitution instance of '$(\exists x)\mathrm{F}g(x)$' and also a substitution instance of '$(\exists z)\mathrm{F}z$'. In its role as a substitution instance of '$(\exists x)\mathrm{F}g(x)$', the instantiating term is 'b'; in its role as a substitution instance of '$(\exists z)\mathrm{F}z$', 'g(b)' is the instantiating term.

Here are the quantifier rules, modified as appropriate for the system *PDE*.

Universal Elimination (∀E)

$$\begin{array}{l} \\ \triangleright \end{array}\begin{array}{|l} (\forall \mathbf{x})\mathbf{P} \\ \mathbf{P(t/x)} \end{array}$$

Existential Introduction (∃I)

$$\begin{array}{l} \\ \triangleright \end{array}\begin{array}{|l} \mathbf{P(t/x)} \\ (\exists \mathbf{x})\mathbf{P} \end{array}$$

where **t** is a closed term

Universal Introduction (∀I)

$$\begin{array}{l} \\ \triangleright \end{array}\begin{array}{|l} \mathbf{P(a/x)} \\ (\forall \mathbf{x})\mathbf{P} \end{array}$$

provided that:

(i) **a** does not occur in an open assumption.

(ii) **a** does not occur in $(\forall \mathbf{x})\mathbf{P}$.

Existential Elimination (∃E)

$$\begin{array}{l} \\ \\ \\ \triangleright \end{array}\begin{array}{|l} (\exists \mathbf{x})\mathbf{P} \\ \quad\begin{array}{|l} \mathbf{P(a/x)} \\ \hline \mathbf{Q} \end{array} \\ \\ \mathbf{Q} \end{array}$$

provided that:

(i) **a** does not occur in an open assumption.

(ii) **a** does not occur in $(\exists \mathbf{x})\mathbf{P}$.

(iii) **a** does not occur in **Q**.

where **a** is an individual constant.

The definitions of the syntactic properties of sentences and sets of sentences in *PDE* (equivalence, validity, etc.) are all carried over from *PD,* substituting '*PDE*' for '*PD*' in each of the definitions.

In the rest of this section we will illustrate the use of the quantifier rules, as modified for *PDE,* by doing a series of derivations that establish various syntactic properties of sentences and sets of sentences of *PLE*.

ARGUMENTS

We begin by showing that the following argument is valid in *PDE*.

$$\begin{array}{l}(\forall x)(\forall y)(Fx \supset Gxy)\\ (\exists x)F\mathit{f}(x)\\ \hline (\exists x)(\exists y)Gxy\end{array}$$

Since the second premise is an existentially quantified sentence we will use Existential Elimination as our primary strategy:

Derive: $(\exists x)(\exists y)Gxy$

1	$(\forall x)(\forall y)(Fx \supset Gxy)$	Assumption
2	$(\exists x)F\mathit{f}(x)$	Assumption
3	$F\mathit{f}(a)$	A / ∃E
G	$(\exists x)(\exists y)Gxy$	
G	$(\exists x)(\exists y)Gxy$	2, 3–__ ∃E

Two applications of Universal Elimination produce a material conditional that has '$F\mathit{f}(a)$' as its antecedent:

Derive: $(\exists x)(\exists y)Gxy$

1	$(\forall x)(\forall y)(Fx \supset Gxy)$	Assumption
2	$(\exists x)F\mathit{f}(x)$	Assumption
3	$F\mathit{f}(a)$	A / ∃E
4	$(\forall y)(F\mathit{f}(a) \supset G\mathit{f}(a)y)$	1 ∀E
5	$F\mathit{f}(a) \supset G\mathit{f}(a)b$	4 ∀E
G	$(\exists x)(\exists y)Gxy$	
G	$(\exists x)(\exists y)Gxy$	2, 3–__ 3E

We can derive 'Gf(a)b' from lines 3 and 5 by Conditional Elimination, and then we can derive our current goal with two applications of Existential Introduction:

Derive: $(\exists x)(\exists y)Gxy$

1	$(\forall x)(\forall y)(Fx \supset Gxy)$	Assumption
2	$(\exists x)F f(x)$	Assumption
3	$\quad F f(a)$	A / $\exists$E
4	$\quad (\forall y)(F f(a) \supset G f(a)y)$	1 $\forall$E
5	$\quad F f(a) \supset G f(a)b$	4 $\forall$E
6	$\quad G f(a)b$	3, 5 $\supset$E
7	$\quad (\exists y)G f(a)y$	6 $\exists$I
8	$\quad (\exists x)(\exists y)Gxy$	7 $\exists$I
9	$(\exists x)(\exists y)Gxy$	2, 3–8 $\exists$E

Both Universal Elimination and Existential Introduction allow the associated substitution instance to be formed from a closed complex term, as we have done here (the substitution instance on line 4 of the universally quantified sentence on line 1 is formed using the complex term 'f(a)', as is the substitution instance on line 7 of the existentially quantified sentence on line 8).

We next show that the following argument is valid in *PDE:*

$a = g(b)$
$(\forall x)(Fxa \supset (\forall y)Gyx)$
$(\exists y)Fyg(b)$

$(\exists x)(\forall y)Gyx$

We will proceed much as in the previous example, using Existential Elimination as our primary strategy. But this example also requires the use of Identity Elimination:

Derive: $(\exists x)(\forall y)Gyx$

1	$a = g(b)$	Assumption
2	$(\forall x)(Fxa \supset (\forall y)Gyx)$	Assumption
3	$(\exists y)Fyg(b)$	Assumption
4	$\quad Fcg(b)$	A / $\exists$E
5	$\quad Fca \supset (\forall y)Gyc$	2 $\forall$E
6	$\quad Fca$	1, 4 =E
7	$\quad (\forall y)Gya$	5, 6 $\supset$E
8	$\quad (\exists x)(\forall y)Gyx$	7 $\exists$I
9	$(\exists x)(\forall y)Gyx$	3, 4–8 $\exists$E

At line 6 we replaced 'g(b)' in 'Fcg(b)' with 'a'.

The sentence '$(\forall z)(\forall y)(z = y \supset y = z)$' says of each pair of things that if the first member of the pair is identical to the second, then the second is identical to the first. Our derivation will end with two uses of Universal Introduction:

Derive: $(\forall z)(\forall y)(z = y \supset y = z)$

G	$b = c \supset c = b$	
G	$(\forall y)(b = y \supset y = b)$	___ ∀I
G	$(\forall z)(\forall y)(z = y \supset y = z)$	___ ∀I

It is important that we use two different constants to form the goal at the third line from the bottom. If we had picked '$b = b \supset b = b$' as our goal we would not be able to derive '$(\forall y)(b = y \supset y = b)$' by Universal Introduction, as the second restriction on that rule prohibits the instantiating term from occurring in the sentence that is derived by the rule. We will use Conditional Introduction to derive the goal '$b = c \supset c = b$':

Derive: $(\forall z)(\forall y)(z = y \supset y = z)$

1	$b = c$	A / ⊃I
G	$c = b$	
G	$b = c \supset c = b$	1–___ ⊃I
G	$(\forall y)(b = y \supset y = b)$	___ ∀I
G	$(\forall z)(\forall y)(z = y \supset y = z)$	___ ∀I

We can finish the derivation by using Identity Introduction to derive '$(\forall y)y = y$' (or any other sentence of this form), then deriving either '$b = b$' or '$c = c$'—it doesn't matter which—by Universal Elimination and then using Identity Elimination to derive '$c = b$':

Derive: $(\forall z)(\forall y)(z = y \supset y = z)$

1	$b = c$	A / ⊃I
2	$(\forall y)y = y$	=I
3	$c = c$	2 ∀E
4	$c = b$	1, 3 =E
5	$b = c \supset c = b$	1–4 ⊃I
6	$(\forall y)(b = y \supset y = b)$	5 ∀I
7	$(\forall z)(\forall y)(z = y \supset y = z)$	6 ∀I

Once we have '$c = c$' at line 3 we can use Identity Elimination, replacing the second occurrence of 'c' in '$c = c$' with 'b', based on the identity at line 1.

The sentence '$(\forall x)(\forall y)(\forall z)[(x = f(z) \ \& \ y = f(z)) \supset x = y]$' is also a theorem of *PDE*. We will work from the bottom up, anticipating three applications of Universal Introduction:

Derive: $(\forall x)(\forall y)(\forall z)[(x = f(z) \ \& \ y = f(z)) \supset x = y]$

1		
G	$[(a = f(c) \ \& \ b = f(c)) \supset a = b]$	
G	$(\forall z)[(a = f(z) \ \& \ b = f(z)) \supset a = b]$	___ ∀I
G	$(\forall y)(\forall z)[(a = f(z) \ \& \ y = f(z)) \supset a = y]$	___ ∀I
G	$(\forall x)(\forall y)(\forall z)[(x = f(z) \ \& \ y = f(z)) \supset x = y]$	___ ∀I

Our current goal is a material conditional, so we will try to obtain it by Conditional Introduction, assuming '$(a = f(c) \ \& \ b = f(c))$' and deriving '$a = b$'. The latter can be derived using Conjunction Elimination and Identity Elimination:

1	$(a = f(c) \ \& \ b = f(c))$	A/ ⊃ I
2	$a = f(c)$	1 &E
3	$b = f(c)$	1 &E
4	$a = b$	2, 3 =E
5	$[(a = f(c) \ \& \ b = f(c)) \supset a = b]$	1–4 ⊃E
6	$(\forall z)[(a = f(z) \ \& \ b = f(z)) \supset a = b]$	5 ∀I
7	$(\forall y)(\forall z)[(a = f(z) \ \& \ y = f(z)) \supset a = y]$	6 ∀I
8	$(\forall x)(\forall y)(\forall z)[(x = f(z) \ \& \ y = f(z)) \supset x = y]$	7 ∀I

INCONSISTENCY

The set $\{(\forall x)(Fx \vee (\exists y)Gxy), \sim Fg(a,b), g(a,b) = c, \sim (\exists y)Gcy\}$ is inconsistent in *PDE*. To show this we need to derive a sentence **Q** and its negation ~ **Q**. We will use '~ $Fg(a,b)$' as ~ **Q** and we will use Disjunction Elimination as our primary strategy:

Derive: $Fg(a,b), \sim Fg(a,b)$

1	$(\forall x)(Fx \vee (\exists y)Gxy)$	Assumption
2	$\sim Fg(a,b)$	Assumption
3	$g(a,b) = c$	Assumption
4	$\sim (\exists y)Gcy$	Assumption
5	$Fc \vee (\exists y)Gcy$	1 ∀E
6	Fc	A / ∨E
7	$Fg(a,b)$	3, 6 =E
8	$(\exists y)Gcy$	A / ∨E
G	$Fg(a,b)$	
G	$Fg(a,b)$	5, 6–7, 8–___ ∨E
	$\sim Fg(a,b)$	2 R

Our remaining task is to derive '$Fg(a,b)$'. Doing so is not difficult because both '$\sim (\exists y)Gcy$' and '$(\exists y)Gcy$' are available to us, at lines 4 and 8, respectively. So we will use Negation Elimination to complete the derivation:

Derive: $Fg(a,b)$, $\sim Fg(a,b)$

1	$(\forall y)(Fx \vee (\exists y)Gxy)$	Assumption
2	$\sim Fg(a,b)$	Assumption
3	$g(a,b) = c$	Assumption
4	$\sim (\exists y)Gcy$	Assumption
5	$Fc \vee (\exists y)Gcy$	1 ∀E
6	$\quad Fc$	A / ∨E
7	$\quad Fg(a,b)$	3, 6 =E
8	$\quad (\exists y)Gcy$	A / ∨E
9	$\quad\quad \sim Fg(a,b)$	A / ~ E
10	$\quad\quad (\exists y)Gcy$	8 R
11	$\quad\quad \sim (\exists y)Gcy$	4 R
12	$\quad Fg(a,b)$	9–11 ~ E
13	$Fg(a,b)$	5, 6–7, 8–12 ∨E
14	$\sim Fg(a,b)$	2 R

There is an important difference between *PD*+ and our latest system, *PDE*. Although both are extensions of *PD* in the sense that each adds new rules to *PD, PD*+ is not stronger than *PD*. Everything derivable in *PD*+ is derivable in *PD*. However, *PDE,* with two new identity rules and modifications of two of *PD*'s quantifier rules, allows us to derive results in *PDE* that are not derivable in *PD*. The previous examples in this section involving the identity predicate and complex terms illustrate this.

However, it should be clear that we can augment the rules of *PDE* with the additional rules of *PD*+ to form a derivation system *PDE*+ that is equivalent to *PDE*. Here is a short derivation in *PDE*+:

Derive: $\sim (\exists x)f(x) = x$

1	$(\forall x)(\forall y)(f(x) = y \supset \sim f(y) = x)$	Assumption
2	$\quad f(a) = a$	A / ~ I
3	$\quad\quad (\forall y)(f(a) = y \supset \sim f(y) = a)$	1 ∀E
4	$\quad\quad f(a) = a \supset \sim f(a) = a)$	3 ∀E
5	$\quad\quad \sim f(a) = a$	2, 4 ⊃E
6	$\quad\quad f(a) = a$	2 R
7	$\sim f(a) = a$	2–6 ~ I
8	$(\forall x) \sim f(x) = x$	7 ∀I
9	$\sim (\exists x)f(x) = x$	8 QN

10.4**E** EXERCISES

1. Show that each of the following is a theorem in *PDE*.

a. $a = b \supset b = a$

*b. $(a = b \,\&\, b = c) \supset a = c$

c. $(\sim a = b \,\&\, b = c) \supset \sim a = c$

*d. $\sim a = b \equiv \sim b = a$

e. $\sim a = c \supset (\sim a = b \vee \sim b = c)$

2. Show that each of the following is valid in *PDE*.

a. $a = b \,\&\, \sim Bab$

$\sim (\forall x)Bxx$

*b. $Ge \supset d = e$

$Ge \supset He$

$Ge \supset Hd$

c. $(\forall z)[Gz \supset (\forall y)(Ky \supset Hzy)]$

$(Ki \,\&\, Gj) \,\&\, i = j$

Hii

*d. $(\exists x)(Hx \,\&\, Mx)$

$Ms \,\&\, \sim Hs$

$(\exists x)[(Hx \,\&\, Mx) \,\&\, \sim x = s]$

e. $a = b$

$Ka \vee \sim Kb$

3. Show that each of the following is a theorem in *PDE*.

a. $(\forall x)(x = x \vee \sim x = x)$

*b. $(\forall x)(\forall y)(x = x \,\&\, y = y)$

c. $(\forall x)(\forall y)(x = y \equiv y = x)$

*d. $(\forall x)(\forall y)(\forall z)[(x = y \,\&\, y = z) \supset x = z]$

e. $\sim (\exists x) \sim x = x$

4. Symbolize each of the following arguments in *PLE* and show that each argument is valid in *PDE*.

a. The number 2 is not identical to 4. The numbers 2 and 4 are both even numbers. Therefore there are at least two different even numbers.

*b. Hyde killed some innocent person. But Jekyll is Hyde. Jekyll is a doctor. Hence some doctor killed some innocent person.

c. Shakespeare didn't admire himself, but the queen admired Bacon. Thus Shakespeare isn't Bacon since Bacon admired everybody who was admired by somebody.

*d. Rebecca loves those and only those who love her. The brother of Charlie loves Rebecca. Sam is Charlie's brother. So Sam and Rebecca love each other.

e. Somebody robbed Peter and paid Paul. Peter didn't rob himself. Paul didn't pay himself. Therefore the person who robbed Peter and paid Paul was neither Peter nor Paul.

5. Which of the following illustrate mistakes in *PDE?* Explain what each mistake is.

a.
1		$(\exists x)Sx$	Assumption
2	|	$Sg(f)$	A / $\exists$E
3	|	$(\exists x)Sg(x)$	2 $\exists$I
4		$(\exists x)Sg(x)$	1, 2–3 $\exists$E

*b.
1		$(\exists x)Sg(x,x)$	Assumption
2	|	$Sg(i,i)$	A / $\exists$E
3	|	$(\exists x)Sg(i,x)$	2 $\exists$I
4		$(\exists x)Sg(i,x)$	1, 2–3 $\exists$E

c.
1		$(\exists x)Hxg(x)$	Assumption
2	|	$Heg(e)$	A / $\exists$E
3	|	$(\exists y)Hyg(y)$	2 $\exists$I
4		$(\exists y)Hyg(y)$	1, 2–3 $\exists$E

*d.
1		$(\forall x)Rf(x)$	Assumption
2		$Rf(a)$	1 $\forall$E
3		$(\forall z)Rf(z)$	2 $\forall$I

e.
1		$(\forall x)Lxxx$	Assumption
2		$Lf(a,a)a$	1 $\forall$E
3		$(\forall x)Lf(x,x)x$	2 $\forall$I

*f.
1		$(\forall x)Mx$	Assumption
2		$Mf(f(a))$	1 $\forall$E
3		$(\exists x)Mf(x)$	2 $\exists$I

g.
1		$(\forall x)Rf(x,x)$	Assumption
2		$Rf(c,c)$	1 $\forall$E
3		$(\forall y)Ry$	2 $\forall$I

*h.
1		$(\forall x)Jx$	Assumption
2		$Jf(f(a))$	1 $\forall$E
3		$(\exists y)Jf(f(y))$	2 $\exists$I

i. 1 | $(\forall x)Jx$ — Assumption

2 | $Jf(g(a,b))$ — 1 $\forall$E

3 | $(\exists x)Jf(g(x,b))$ — 2 $\exists$I

*j. 1 | $(\forall x)Lx$ — Assumption

2 | $Lf(a,a)$ — 1 $\forall$E

3 | $(\forall x)Lf(a,x)$ — 2 $\forall$I

6. Show that each of the following is a theorem in *PDE*.

a. $(\forall x)(\exists y)f(x) = y$

*b. $(\forall x)(\forall y)(\forall z)[(f(x) = g(x,y) \;\&\; g(x,y) = h(x,y,z)) \supset f(x) = h(x,y,z)]$

c. $(\forall x)Ff(x) \supset (\forall x)Ff(g(x))$

*d. $(\forall x)[\sim f(x) = x \supset (\forall y)(f(x) = y \supset \sim x = y)]$

e. $(\forall x)(f(f(x)) = x \supset f(f(f(f(x)))) = x)$

*f. $(\forall x)(\forall y)(\forall z)[(f(g(x)) = y \;\&\; f(y) = z) \supset f(f(g(x))) = z]$

g. $(\forall x)(\forall y)[(f(x) = y \;\&\; f(y) = x) \supset x = f(f(x))]$

7. Show that each of the following is valid in *PDE*.

a. $(\forall x)(Bx \supset Gxf(x))$

$(\forall x)Bf(x)$

$(\forall x)Gf(x)f(f(x))$

*b. $(\forall x)(Kx \vee Hg(x))$

$(\forall x)(Kg(x) \vee Hg(g(x)))$

c. $(\forall x)(\forall y)(f(x) = y \supset Myxc)$

$\sim Mbac \;\&\; \sim Mabc$

$\sim f(a) = b$

*d. $\sim (\exists x)Rx$

$(\forall x) \sim Rf(x,g(x))$

e. $(\exists x)(\forall y)(\forall z)Lxyz$

$(\exists x)Lxf(x)g(x)$

*f. $(\forall x)[\sim Lxf(x) \vee (\exists y)Ng(y)]$

$(\exists x)Lf(x)f(f(x)) \supset (\exists x)Ng(y)$

g. $(\forall x)[Zx \supset (\forall y)(\sim Dxy \equiv Hf(f(y)))]$

$(\forall x)(Zx \;\&\; \sim Hx)$

$(\forall x)Df(x)f(x)$

*h. $(\forall x)(\forall y)(\exists z)Sf(x)yz$

$(\forall x)(\forall y)(\forall z)(Sxyz \supset \sim (Cxyz \vee Mzyx))$

$(\exists x)(\exists y) \sim (\forall z)Mzg(y)f(g(x))$

GLOSSARY[2]

DERIVABILITY IN *PD:* A sentence **P** of *PL* is *derivable in PD* from a set Γ of sentences of *PL* if and only if there is a derivation in *PD* in which all the primary assumptions are members of Γ and **P** occurs within the scope of only those assumptions.

VALIDITY IN *PD:* An argument of *PL* is *valid in PD* if and only if the conclusion of the argument is derivable in *PD* from the set consisting of the premises. An argument of *PL* is *invalid in PD* if and only if it is not valid in *PD*.

THEOREM IN *PD:* A sentence **P** of *PL* is a *theorem in PD* if and only if **P** is derivable in *PD* from the empty set.

EQUIVALENCE IN *PD:* Sentences **P** and **Q** of *PL* are *equivalent in PD* if and only if **Q** is derivable in *PD* from {**P**} and **P** is derivable in *PD* from {**Q**}.

INCONSISTENCY IN *PD:* A set Γ of sentences of *PL* is *inconsistent in PD* if and only if there is a sentence **P** of *PL* such that both **P** and ~ **P** are derivable in *PD* from Γ. A set Γ of sentences of *PL* is *consistent in PD* if and only if it is not inconsistent in *PD*.

[2]Similar definitions hold for the derivation systems *PD+*, *PDE,* and *PDE+*.

Chapter 11

PREDICATE LOGIC: METATHEORY

Section 11.1 presents a variety of semantic results that will be used to prove important metatheorems for *PL*, while Section 11.2 does the same for *PLE*. Section 11.3 proves that the derivation systems *PD*, *PD*+, and *PDE* are sound for predicate logic, and Section 11.4 proves that these systems are complete. Section 11.5 proves that the tree method is sound for predicate logic, while Section 11.6 proves that it is complete.

11.1 SEMANTIC PRELIMINARIES FOR *PL*

In this chapter, we shall establish four major results: the soundness and completeness of the natural deduction systems *PD*, *PD*+, and *PDE*, and the soundness and completeness of the truth-tree method developed in Chapter 9. The results we establish are part of the **metatheory** of predicate logic.

In our proofs of the adequacy of the natural deduction systems and the tree method, we shall use some fundamental semantic results that may seem obvious but that nevertheless must be proved. The purpose of this section is to establish these results. The reader may skim over this section on the first reading without working through all the proofs but should keep in mind that later metatheoretic proofs depend on the results presented here.

Given any formula **P**, variable **x**, and constant **a**, let $\mathbf{P}(\mathbf{a}/\mathbf{x})$ be the formula that results from replacing every free occurrence of **x** in **P** with **a**. Our first result establishes that every variable assignment $\mathbf{d_I}$ treats $\mathbf{P}(\mathbf{a}/\mathbf{x})$ exactly as

$\mathbf{d_I}[\mathbf{I(a)/x}]$ treats **P**. If $\mathbf{d_I}$ satisfies $\mathbf{P(a/x)}$, then the variable assignment that is just like $\mathbf{d_I}$ except that it assigns the denotation of **a** to **x** will satisfy **P**, and vice versa. This should not be surprising, for if **x** is used to refer to exactly the same thing as **a**, we would expect **P** and $\mathbf{P(a/x)}$ to behave the same way.

> **11.1.1:** Let **P** be a formula of *PL*, let $\mathbf{P(a/x)}$ be the formula that results from replacing every free occurrence of **x** in **P** with an individual constant **a**, let **I** be an interpretation, and let $\mathbf{d_I}$ be a variable assignment for **I**. Then $\mathbf{d_I}$ satisfies $\mathbf{P(a/x)}$ on **I** if and only if $\mathbf{d_I}[\mathbf{I(a)/x}]$ satisfies **P** on **I**.

To prove the result, we shall use mathematical induction on the number of occurrences of logical operators—truth-functional connectives and quantifiers—that occur in **P**.

> *Basis clause:* If **P** is a formula that contains zero occurrences of logical operators, then $\mathbf{d_I}$ satisfies $\mathbf{P(a/x)}$ if and only if $\mathbf{d_I}[\mathbf{I(a)/x}]$ satisfies **P**.
>
> **Proof of basis clause:** If **P** contains zero occurrences of logical operators, then **P** is either a sentence letter or a formula of the form $\mathbf{At_1 \ldots t_n}$, where **A** is a predicate and $\mathbf{t_1}, \ldots, \mathbf{t_n}$ are individual constants or variables. If **P** is a sentence letter, then $\mathbf{P(a/x)}$ is simply **P**—a sentence letter alone does not contain any variables to be replaced. $\mathbf{d_I}$ satisfies $\mathbf{P(a/x)}$, then, if and only if $\mathbf{I(P)} = \mathbf{T}$. And $\mathbf{d_I}[\mathbf{I(a)/x}]$ satisfies **P** if and only if $\mathbf{I(P)} = \mathbf{T}$. So $\mathbf{d_I}$ satisfies $\mathbf{P(a/x)}$ if and only if $\mathbf{d_I}[\mathbf{I(a)/x}]$ satisfies **P**.
>
> If **P** has the form $\mathbf{At_1 \ldots t_n}$, then $\mathbf{P(a/x)}$ is $\mathbf{At_1' \ldots t_n'}$, where $\mathbf{t_i'}$ is **a** if $\mathbf{t_i}$ is **x** and $\mathbf{t_i'}$ is just $\mathbf{t_i}$ otherwise. By the definition of satisfaction,
>
> - $\mathbf{d_I}$ satisfies $\mathbf{At_1' \ldots t_n'}$ if and only if $<\mathrm{den}_{\mathbf{I},\mathbf{d_I}}(\mathbf{t_1'}), \mathrm{den}_{\mathbf{I},\mathbf{d_I}}(\mathbf{t_2'}), \ldots, \mathrm{den}_{\mathbf{I},\mathbf{d_I}}(\mathbf{t_n'})> \in \mathbf{I(A)}$.
>
>   (Recall from Chapter 8 that if $\mathbf{t_i}$ is a variable, $\mathrm{den}_{\mathbf{I},\mathbf{d_I}}(\mathbf{t_i}) = \mathbf{d_I}(\mathbf{t_i})$, and if $\mathbf{t_i}$ is an individual constant, $\mathrm{den}_{\mathbf{I},\mathbf{d_I}}(\mathbf{t_i}) = \mathbf{I}(\mathbf{t_i})$.)
> - $\mathbf{d_I}[\mathbf{I(a)/x}]$ satisfies $\mathbf{At_1 \ldots t_n}$ if and only if $<\mathrm{den}_{\mathbf{I},\mathbf{d_I}[\mathbf{I(a)/x}]}(\mathbf{t_1}), \mathrm{den}_{\mathbf{I},\mathbf{d_I}[\mathbf{I(a)/x}]}(\mathbf{t_2}), \ldots, \mathrm{den}_{\mathbf{I},\mathbf{d_I}[\mathbf{I(a)/x}]}(\mathbf{t_n})> \in \mathbf{I(A)}$.

But now we note that

> - The **n**-tuples $<\mathrm{den}_{\mathbf{I},\mathbf{d_I}}(\mathbf{t_1'}), \mathrm{den}_{\mathbf{I},\mathbf{d_I}}(\mathbf{t_2'}), \ldots, \mathrm{den}_{\mathbf{I},\mathbf{d_I}}(\mathbf{t_n'})>$ and $<\mathrm{den}_{\mathbf{I},\mathbf{d_I}[\mathbf{I(a)/x}]}(\mathbf{t_1}), \mathrm{den}_{\mathbf{I},\mathbf{d_I}[\mathbf{I(a)/x}]}(\mathbf{t_2}), \ldots, \mathrm{den}_{\mathbf{I},\mathbf{d_I}[\mathbf{I(a)/x}]}(\mathbf{t_n})>$ are identical.

Consider: If $\mathbf{t_i}$ is a constant, then $\mathbf{t_i'}$ is $\mathbf{t_i}$ and so $\mathrm{den}_{\mathbf{I},\mathbf{d_I}}(\mathbf{t_i'}) = \mathbf{I}(\mathbf{t_i})$ and $\mathrm{den}_{\mathbf{I},\mathbf{d_I}[\mathbf{I(a)/x}]}(\mathbf{t_i}) = \mathbf{I}(\mathbf{t_i})$. If $\mathbf{t_i}$ is any variable other than **x**, then $\mathbf{t_i'}$ is $\mathbf{t_i}$ and so $\mathrm{den}_{\mathbf{I},\mathbf{d_I}}(\mathbf{t_i'}) = \mathbf{d_I}(\mathbf{t_i}) = \mathbf{d_I}[\mathbf{I(a)/x}](\mathbf{t_i}) = \mathrm{den}_{\mathbf{I},\mathbf{d_I}[\mathbf{I(a)/x}]}(\mathbf{t_i})$—the assignment of $\mathbf{I(a)}$ to **x** in the variable assignment does not affect the value assigned to $\mathbf{t_i}$ in this case. If $\mathbf{t_i}$ is the variable **x**, then the variant ensures that the denotations of **x** and **a** coincide: $\mathbf{t_i'}$ is **a** and $\mathrm{den}_{\mathbf{I},\mathbf{d_I}}(\mathbf{a}) = \mathbf{I(a)} = \mathbf{d_I}[\mathbf{I(a)/x}](\mathbf{x}) = \mathrm{den}_{\mathbf{I},\mathbf{d_I}[\mathbf{I(a)/x}]}(\mathbf{x})$.

We conclude that $\mathbf{d_I}$ satisfies $\mathbf{At}'_1 \ldots \mathbf{t}'_n$ if and only if $\mathbf{d_I}[\mathbf{I}(\mathbf{a})/\mathbf{x}]$ satisfies $\mathbf{At}_1 \ldots \mathbf{t}_n$.

The basis clause—in particular, the case where an atomic formula has the form $\mathbf{At}_1 \ldots \mathbf{t}_n$—is the crux of our proof. It will be straightforward to show that the addition of connectives and quantifiers to build larger formulas does not change matters. The inductive step in the proof of 11.1.1 is

Inductive step: If every formula **P** with **k** or fewer occurrences of logical operators is such that $\mathbf{d_I}$ satisfies $\mathbf{P}(\mathbf{a}/\mathbf{x})$ if and only if $\mathbf{d_I}[\mathbf{I}(\mathbf{a})/\mathbf{x}]$ satisfies **P**, then the same is true of every formula **P** with $\mathbf{k} + 1$ occurrences of logical operators.

Proof of inductive step: Letting **k** be an arbitrary positive integer, we assume that the inductive hypothesis holds—that our claim is true of every formula with **k** or fewer occurrences of logical operators. To show that it follows that the claim is also true of every formula **P** with $\mathbf{k} + 1$ occurrences of logical operators, we consider each form that **P** may have.

Case 1: P has the form $\sim \mathbf{Q}$. Then $\mathbf{P}(\mathbf{a}/\mathbf{x})$ is $\sim \mathbf{Q}(\mathbf{a}/\mathbf{x})$, the negation of $\mathbf{Q}(\mathbf{a}/\mathbf{x})$ (that is, any replacements of **x** that were made had to be made within **Q**).

- By the definition of satisfaction, $\mathbf{d_I}$ satisfies $\sim \mathbf{Q}(\mathbf{a}/\mathbf{x})$ if and only if it does not satisfy $\mathbf{Q}(\mathbf{a}/\mathbf{x})$.
- Because $\mathbf{Q}(\mathbf{a}/\mathbf{x})$ contains fewer than $\mathbf{k} + 1$ occurrences of logical operators, it follows from the inductive hypothesis that **d** fails to satisfy $\mathbf{Q}(\mathbf{a}/\mathbf{x})$ if and only if $\mathbf{d_I}[\mathbf{I}(\mathbf{a})/\mathbf{x}]$ fails to satisfy **Q**.
- By the definition of satisfaction, $\mathbf{d_I}[\mathbf{I}(\mathbf{a})/\mathbf{x}]$ fails to satisfy **Q** if and only if $\mathbf{d_I}[\mathbf{I}(\mathbf{a})/\mathbf{x}]$ does satisfy $\sim \mathbf{Q}$.

Therefore, $\mathbf{d_I}$ satisfies $\sim \mathbf{Q}(\mathbf{a}/\mathbf{x})$ if and only if $\mathbf{d_I}[\mathbf{I}(\mathbf{a})/\mathbf{x}]$ satisfies $\sim \mathbf{Q}$.

Case 2: P has the form $\mathbf{Q} \,\&\, \mathbf{R}$. Then $\mathbf{P}(\mathbf{a}/\mathbf{x})$ is $\mathbf{Q}(\mathbf{a}/\mathbf{x}) \,\&\, \mathbf{R}(\mathbf{a}/\mathbf{x})$,

that is, all replacements of **x** occurred within **Q** and **R**.

- By the definition of satisfaction, $\mathbf{d_I}$ satisfies $\mathbf{Q}(\mathbf{a}/\mathbf{x}) \,\&\, \mathbf{R}(\mathbf{a}/\mathbf{x})$ if and only if $\mathbf{d_I}$ satisfies $\mathbf{Q}(\mathbf{a}/\mathbf{x})$ and $\mathbf{d_I}$ satisfies $\mathbf{R}(\mathbf{a}/\mathbf{x})$.
- Both conjuncts contain fewer than $\mathbf{k} + 1$ occurrences of logical operators, so, by the inductive hypothesis, $\mathbf{d_I}$ satisfies $\mathbf{Q}(\mathbf{a}/\mathbf{x})$ if and only if $\mathbf{d_I}[\mathbf{I}(\mathbf{a})/\mathbf{x}]$ satisfies **Q**, and $\mathbf{d_I}$ satisfies $\mathbf{R}(\mathbf{a}/\mathbf{x})$ if and only if $\mathbf{d_I}[\mathbf{I}(\mathbf{a})/\mathbf{x}]$ satisfies **R**.
- By the definition of satisfaction, $\mathbf{d_I}[\mathbf{I}(\mathbf{a})/\mathbf{x}]$ satisfies both **Q** and **R** if and only if $\mathbf{d_I}[\mathbf{I}(\mathbf{a})/\mathbf{x}]$ satisfies $\mathbf{Q} \,\&\, \mathbf{R}$.

Therefore, $\mathbf{d_I}$ satisfies $\mathbf{Q(a/x)}$ & $\mathbf{R(a/x)}$ if and only if $\mathbf{d_I[I(a)/x]}$ satisfies **Q** & **R**.

Cases 3–5: The proofs for the cases in which **P** has one of the forms $\mathbf{Q \vee R}$, $\mathbf{Q \supset R}$, and $\mathbf{Q \equiv R}$ are similar to that of Case 2 and are left as exercises.

Case 6: P has the form $\mathbf{(\forall y)Q}$. We must consider two possibilities. If **y** is distinct from the variable **x** that **a** is replacing in $\mathbf{(\forall y)Q}$, then $\mathbf{P(a/x)}$ is $\mathbf{(\forall y)Q(a/x)}$—all replacements of **x** are made within **Q**.

- By the definition of satisfaction, $\mathbf{d_I}$ satisfies $\mathbf{(\forall y)Q(a/x)}$ if and only if for every member **u** of the UD, $\mathbf{d_I[u/y]}$ satisfies $\mathbf{Q(a/x)}$.
- Because **Q** contains fewer than $\mathbf{k + 1}$ occurrences of logical operators, it follows from the inductive hypothesis that for every member **u** of the UD, $\mathbf{d_I[u/y]}$ satisfies $\mathbf{Q(a/x)}$ if and only if $\mathbf{d_I[u/y, I(a)/x]}$ satisfies **Q**.
- Each variant $\mathbf{d_I[u/y, I(a)/x]}$ is identical to $\mathbf{d_I[I(a)/x, u/y]}$ because **x** and **y** are distinct variables, and therefore neither of the assignments within the brackets can override the other.
- So every member **u** of the UD is such that $\mathbf{d_I[u/y, I(a)/x]}$ satisfies **Q** if and only if $\mathbf{d_I[I(a)/x, u/y]}$ satisfies **Q**.
- By the definition of satisfaction, every member **u** of the UD is such that $\mathbf{d_I[I(a)/x, u/y]}$satisfies **Q** if and only if $\mathbf{d_I[I(a)/x]}$ satisfies $\mathbf{(\forall y)Q}$.

Therefore, in the case where **y** is distinct from the variable **x** that **a** is replacing, $\mathbf{d_I}$ satisfies $\mathbf{(\forall y)Q(a/x)}$ if and only if $\mathbf{d_I[I(a)/x]}$ satisfies $\mathbf{(\forall y)Q}$.

If **P** is $\mathbf{(\forall x)Q}$, where **x** is the variable that **a** is replacing, then $\mathbf{P(a/x)}$ is also $\mathbf{(\forall x)Q}$. Because **a** replaces only *free* occurrences of **x** in **P** and **x** does not occur free in **P**, no replacements are made within **Q**.

- By the definition of satisfaction, $\mathbf{d_I}$ satisfies $\mathbf{(\forall x)Q}$ (which is our $\mathbf{P(a/x)}$) if and only if every member **u** of the UD is such that $\mathbf{d_I[u/x]}$ satisfies **Q**.
- $\mathbf{d_I[I(a)/x]}$ satisfies $\mathbf{(\forall x)Q}$ (which is our **P**) if and only every member **u** of the UD is such that $\mathbf{d_I[I(a)/x, u/x]}$ satisfies **Q**.
- What is $\mathbf{d_I[I(a)/x, u/x]}$? This variable assignment is just $\mathbf{d[u/x]}$—the first assignment made to **x** within the brackets is overridden by the second.
- So every member **u** of the UD is such that $\mathbf{d_I[I(a)/x, u/x]}$ satisfies **Q** if and only if every member **u** of the UD is such that $\mathbf{d[u/x]}$ satisfies **Q**.

Therefore, $\mathbf{d_I[I(a)/x]}$ satisfies $\mathbf{(\forall x)Q}$ if and only if $\mathbf{d_I}$ satisfies $\mathbf{(\forall x)Q}$.

Case 7: P has the form $(\exists \mathbf{y})\mathbf{Q}$. Again we consider two possibilities. If **y** is distinct from the variable **x** that **a** is replacing, then $\mathbf{P}(\mathbf{a}/\mathbf{x})$ is $(\exists \mathbf{y})\mathbf{Q}(\mathbf{a}/\mathbf{x})$.

- By the definition of satisfaction, $\mathbf{d_I}$ satisfies $(\exists \mathbf{y})\mathbf{Q}(\mathbf{a}/\mathbf{x})$ if and only if at least one member **u** of the UD is such that $\mathbf{d_I}[\mathbf{u}/\mathbf{y}]$ satisfies $\mathbf{Q}(\mathbf{a}/\mathbf{x})$.
- Because $\mathbf{Q}(\mathbf{a}/\mathbf{x})$ contains fewer than $\mathbf{k} + 1$ occurrences of logical operators, it follows from the inductive hypothesis that $\mathbf{d_I}[\mathbf{u}/\mathbf{y}]$ satisfies $\mathbf{Q}(\mathbf{a}/\mathbf{x})$ if and only if $\mathbf{d_I}[\mathbf{u}/\mathbf{y}, \mathbf{I}(\mathbf{a})/\mathbf{x}]$ satisfies **Q**.
- Because **y** and **x** are different variables, $\mathbf{d_I}[\mathbf{u}/\mathbf{y}, \mathbf{I}(\mathbf{a})/\mathbf{x}]$ is the same variable assignment as $\mathbf{d_I}[\mathbf{I}(\mathbf{a})/\mathbf{x}, \mathbf{u}/\mathbf{y}]$.
- So $\mathbf{d_I}[\mathbf{u}/\mathbf{y}, \mathbf{I}(\mathbf{a})/\mathbf{x}]$ satisfies $\mathbf{Q}(\mathbf{a}/\mathbf{x})$ if and only if $\mathbf{d_I}[\mathbf{I}(\mathbf{a})/\mathbf{x}, \mathbf{u}/\mathbf{y}]$ satisfies **Q**.
- By the definition of satisfaction, $\mathbf{d_I}[\mathbf{I}(\mathbf{a})/\mathbf{x}, \mathbf{u}/\mathbf{y}]$ satisfies **Q** if and only if $\mathbf{d_I}[\mathbf{I}(\mathbf{a})/\mathbf{x}]$ satisfies $(\exists \mathbf{y})\mathbf{Q}$.

It follows that in the case where **y** and **x** are different variables, $\mathbf{d_I}$ satisfies $(\exists \mathbf{y})\mathbf{Q}(\mathbf{a}/\mathbf{x})$ if and only if $\mathbf{d_I}[\mathbf{I}(\mathbf{a})/\mathbf{x}]$ satisfies $(\exists \mathbf{y})\mathbf{Q}$.

If **P** is $(\exists \mathbf{x})\mathbf{Q}$, where **x** is the variable that **a** is replacing, then $\mathbf{P}(\mathbf{a}/\mathbf{x})$ is $(\exists \mathbf{x})\mathbf{Q}$—no replacements are made within **Q** because **x** is not free in $(\exists \mathbf{x})\mathbf{Q}$. So we must show in this case that $\mathbf{d_I}[\mathbf{I}(\mathbf{a})/\mathbf{x}]$ satisfies $(\exists \mathbf{x})\mathbf{Q}$ if and only if $\mathbf{d_I}$ satisfies $(\exists \mathbf{x})\mathbf{Q}$.

- By the definition of satisfaction, $\mathbf{d_I}[\mathbf{I}(\mathbf{a})/\mathbf{x}]$ satisfies $(\exists \mathbf{x})\mathbf{Q}$ if and only if at least one member **u** of the UD is such that $\mathbf{d_I}[\mathbf{I}(\mathbf{a})/\mathbf{x}, \mathbf{u}/\mathbf{x}]$ satisfies **Q**.
- $\mathbf{d}[\mathbf{I}(\mathbf{a})/\mathbf{x}, \mathbf{u}/\mathbf{x}]$ is just $\mathbf{d}[\mathbf{u}/\mathbf{x}]$—the second assignment to **x** overrides the first.
- So $\mathbf{d_I}[\mathbf{I}(\mathbf{a})/\mathbf{x}, \mathbf{u}/\mathbf{x}]$ satisfies **Q** if and only if $\mathbf{d_I}[\mathbf{u}/\mathbf{x}]$ satisfies **Q**.
- By the definition of satisfaction, $\mathbf{d_I}[\mathbf{u}/\mathbf{x}]$ satisfies **Q** if and only if $\mathbf{d_I}$ satisfies $(\exists \mathbf{x})\mathbf{Q}$.

Therefore, $\mathbf{d_I}[\mathbf{I}(\mathbf{a})/\mathbf{x}]$ satisfies $(\exists \mathbf{x})\mathbf{Q}$ if and only if $\mathbf{d_I}$ satisfies $(\exists \mathbf{x})\mathbf{Q}$.

That establishes the inductive step, so result 11.1.1 is also established—every formula **P** is such that $\mathbf{d_I}$ satisfies $\mathbf{P}(\mathbf{a}/\mathbf{x})$ on **I** if and only if $\mathbf{d_I}[\mathbf{I}(\mathbf{a})/\mathbf{x}]$ satisfies **P** on **I**.

The next result will enable us to prove a claim that was made in Chapter 8: that for any interpretation and any sentence of *PL*, either all variable assignments satisfy the sentence or none do. We used this claim in defining truth and falsehood for sentences: A sentence is true on an interpretation if it is satisfied by all variable assignments and false if it is satisfied by none. The reason this claim turns out to be true is that there are no free variables in sentences. Result 11.1.2 assures us that only the values that a variable assignment assigns

to the variables that are free in a formula play a role in determining whether the formula is satisfied:

11.1.2: Let **I** be an interpretation, $\mathbf{d_I}$ a variable assignment for **I**, and **P** a formula of *PL*. Then $\mathbf{d_I}$ satisfies **P** on **I** if and only if every variable assignment that assigns the same values to the free variables in **P** as $\mathbf{d_I}$ satisfies **P**.

Proof: Let **I** be an interpretation, $\mathbf{d_I}$ a variable assignment for **I**, and **P** a formula of *PL*. We shall prove 11.1.2 by mathematical induction on the number of occurrences of logical operators in **P**.

Basis clause: If **P** is a formula that contains zero occurrences of logical operators, then $\mathbf{d_I}$ satisfies **P** if and only if every variable assignment that assigns the same values to the free variables in **P** as $\mathbf{d_I}$ satisfies **P**.

Proof of basis clause: If **P** contains zero occurrences of logical operators, then **P** is either a sentence letter or a formula of the form $\mathbf{At}_1 \ldots \mathbf{t_n}$. If **P** is a sentence letter, then any variable assignment satisfies **P** on **I** if and only if $\mathbf{I(P)} = \mathbf{T}$. Therefore $\mathbf{d_I}$ satisfies **P** if and only if every variable assignment that assigns the same values to the free variables in **P** as $\mathbf{d_I}$ (that is, every variable assignment) satisfies **P**.

If **P** has the form $\mathbf{At}_1 \ldots \mathbf{t_n}$, then by the definition of satisfaction,

- $\mathbf{d_I}$ satisfies **P** if and only if $<\text{den}_{\mathbf{I},\mathbf{d_I}}(\mathbf{t}_1), \text{den}_{\mathbf{I},\mathbf{d_I}}(\mathbf{t}_2), \ldots, \text{den}_{\mathbf{I},\mathbf{d_I}}(\mathbf{t_n})> \in \mathbf{I(A)}$.

And where $\mathbf{d_I'}$ is a variable assignment that assigns the same values to the free variables in **P** as $\mathbf{d_I}$,

- $\mathbf{d_I'}$ satisfies **P** if and only if $<\text{den}_{\mathbf{I},\mathbf{d_I'}}(\mathbf{t}_1), \text{den}_{\mathbf{I},\mathbf{d_I'}}(\mathbf{t}_2), \ldots, \text{den}_{\mathbf{I},\mathbf{d_I'}}(\mathbf{t_n})> \in \mathbf{I(A)}$.

But now we note that

- the **n**-tuples $<\text{den}_{\mathbf{I},\mathbf{d_I}}(\mathbf{t}_1), \text{den}_{\mathbf{I},\mathbf{d_I}}(\mathbf{t}_2), \ldots, \text{den}_{\mathbf{I},\mathbf{d_I}}(\mathbf{t_n})>$ and $<\text{den}_{\mathbf{I},\mathbf{d_I'}}(\mathbf{t}_1), \text{den}_{\mathbf{I},\mathbf{d_I'}}(\mathbf{t}_2), \ldots, \text{den}_{\mathbf{I},\mathbf{d_I'}}(\mathbf{t_n})>$ are identical.

For if $\mathbf{t_i}$ is a constant, then $\text{den}_{\mathbf{I},\mathbf{d_I}}(\mathbf{t_i}) = \text{den}_{\mathbf{I},\mathbf{d_I'}}(\mathbf{t_i}) = \mathbf{I}(\mathbf{t_i})$. And if $\mathbf{t_i}$ is a variable, then $\mathbf{t_i}$ is free in $\mathbf{At}_1 \ldots \mathbf{t_n}$ and, by our assumption, $\mathbf{d_I}$ and $\mathbf{d_I'}$ assign the same value to $\mathbf{t_i}$. Hence, we conclude that $\mathbf{d_I}$ satisfies $\mathbf{At}_1 \ldots \mathbf{t_n}$ if and only if every variable assignment $\mathbf{d_I'}$ that assigns the same values to the free variables in $\mathbf{At}_1 \ldots \mathbf{t_n}$ as $\mathbf{d_I}$ satisfies $\mathbf{At}_1 \ldots \mathbf{t_n}$.

Inductive step: If every sentence **P** that has **k** or fewer occurrences of logical operators is such that $\mathbf{d_I}$ satisfies **P** on **I** if and only if every variable assignment that assigns the same values to the free variables in **P** as $\mathbf{d_I}$ satisfies **P**, then the same is true of every sentence **P** that contains $\mathbf{k} + 1$ occurrences of logical operators.

Proof of inductive step: Assume that, for an arbitrary positive integer **k**, the inductive hypothesis is true. We shall show that on this assumption our claim must also be true of every sentence **P** that contains **k** + 1 occurrences of logical operators. Let **I** be an interpretation and **d** a variable assignment for **I**. We consider each form that **P** may have.

Case 1: P has the form ~ **Q**.

- By the definition of satisfaction, $\mathbf{d_I}$ satisfies ~ **Q** if and only if $\mathbf{d_I}$ fails to satisfy **Q**.
- Because **Q** contains fewer than **k** + 1 occurrences of logical operators, it follows from the inductive hypothesis that $\mathbf{d_I}$ fails to satisfy **Q** if and only if every variable assignment that assigns the same values to the free variables in **Q** fails to satisfy **Q**.
- By the definition of satisfaction, every variable assignment that assigns the same values to the free variables in **Q** as **d** fails to satisfy **Q** if and only if every such assignment does satisfy ~ **Q**.
- The variable assignments that assign the same values to the free variables in **Q** as **d** are the variable assignments that assign the same values to the free variables of ~ **Q** as **d**, because **Q** and ~ **Q** contain the same free variables.

Therefore, $\mathbf{d_I}$ satisfies ~ **Q** if and only if every variable assignment that assigns the same values to the free variables in ~ **Q** satisfies ~ **Q**.

Case 2: P has the form **Q** ∨ **R**.

- By the definition of satisfaction, $\mathbf{d_I}$ satisfies **Q** ∨ **R** if and only if either $\mathbf{d_I}$ satisfies **Q** or $\mathbf{d_I}$ satisfies **R**.
- Because **Q** and **R** each contain fewer than **k** + 1 occurrences of logical operators, it follows by the inductive hypothesis that $\mathbf{d_I}$ satisfies **Q** if and only if every variable assignment that assigns the same values to the free variables in **Q** satisfies **Q**, and $\mathbf{d_I}$ satisfies **R** if and only if every variable assignment that assigns the same values to the free variables in **R** satisfies **R**.
- Therefore, $\mathbf{d_I}$ satisfies **Q** ∨ **R** if and only if either every variable assignment that assigns the same values to the free variables in **Q** satisfies **Q** or every variable assignment that assigns the same values to the free variables in **R** satisfies **R**.
- We note that every variable that is free in **Q** is also free in **Q** ∨ **R**, so every variable assignment that assigns the same values as $\mathbf{d_I}$ to the free variables in **Q** ∨ **R** is a variable assignment that assigns the same values as $\mathbf{d_I}$ to the free variables in **Q**; and the same is true of **R.** (The converses do not hold.)

We conclude that $\mathbf{d_I}$ satisfies $\mathbf{Q} \vee \mathbf{R}$ if and only if every variable assignment that assigns the same values as $\mathbf{d_I}$ to the free variables in $\mathbf{Q} \vee \mathbf{R}$ satisfies $\mathbf{Q} \vee \mathbf{R}$.

Cases 3–5: **P** has one of the forms $\mathbf{Q}$ & $\mathbf{R}$, $\mathbf{Q} \supset \mathbf{R}$, or $\mathbf{Q} \equiv \mathbf{R}$. These cases are left as an exercise.

Case 6: **P** has the form $(\forall \mathbf{x})\mathbf{Q}$.

- By the definition of satisfaction, $\mathbf{d_I}$ satisfies $(\forall \mathbf{x})\mathbf{Q}$ if and only if every member **u** of the UD is such that $\mathbf{d_I}[\mathbf{u}/\mathbf{x}]$ satisfies **Q**.
- Because **Q** contains fewer than $\mathbf{k} + 1$ occurrences of connectives, it follows from the inductive hypothesis that every member **u** of the UD is such that $\mathbf{d_I}[\mathbf{u}/\mathbf{x}]$ satisfies **Q** if and only if every variable assignment that assigns the same values to the free variables in **Q** as $\mathbf{d_I}[\mathbf{u}/\mathbf{x}]$ satisfies **Q**.
- It follows that **d** satisfies $(\forall \mathbf{x})\mathbf{Q}$ if and only if every member **u** of the UD is such that every variable assignment that assigns the same values to the free variables in **Q** as $\mathbf{d_I}[\mathbf{u}/\mathbf{x}]$ satisfies **Q**.
- Because the variables other than **x** that are free in **Q** are also free in $(\forall \mathbf{x})\mathbf{Q}$, every variable assignment that assigns the same values to the free variables in **Q** as $\mathbf{d_I}[\mathbf{u}/\mathbf{x}]$ is a variant $\mathbf{d_I'}[\mathbf{u}/\mathbf{x}]$ of a variable assignment $\mathbf{d_I'}$ that assigns the same values to the free variables in $(\forall \mathbf{x})\mathbf{Q}$ as $\mathbf{d_I}$, and vice versa.
- So $\mathbf{d_I}$ satisfies $(\forall \mathbf{x})\mathbf{Q}$ if and only if every member **u** of the UD is such that every variant $\mathbf{d_I'}[\mathbf{u}/\mathbf{x}]$ of any variable assignment $\mathbf{d_I'}$ that assigns the same values to the free variables in $(\forall \mathbf{x})\mathbf{Q}$ as $\mathbf{d_I}$ satisfies **Q**.

It follows by the definition of satisfaction that $\mathbf{d_I}$ satisfies $(\forall \mathbf{x})\mathbf{Q}$ if and only if every variable assignment that assigns the same values to the free variables in $(\forall \mathbf{x})\mathbf{Q}$ satisfies $(\forall \mathbf{x})\mathbf{Q}$.

Case 7: **P** has the form $(\exists \mathbf{x})\mathbf{Q}$. This case is left as an exercise.

It follows immediately from 11.1.2 that

11.1.3: For any interpretation **I** and sentence **P** of *PL*, either every variable assignment for **I** satisfies **P** or no variable assignment for **I** satisfies **P**.

Proof: Let $\mathbf{d_I}$ be any variable assignment. Because **P** is a sentence and hence contains no free variables, every variable assignment for **I** assigns the same values to the free variables in **P** as does $\mathbf{d_I}$. By result 11.1.2,

then, $\mathbf{d_I}$ satisfies **P** if and only if every variable assignment satisfies **P**. Therefore either every variable assignment satisfies **P** or none does.

Each of the following results, which can be established using results 11.1.1–11.1.3, states something that we would hope to be true of quantified sentences of *PL*.

11.1.4: For any universally quantified sentence $(\forall\mathbf{x})\mathbf{P}$ of *PL*, $\{(\forall\mathbf{x})\mathbf{P}\}$ quantificationally entails every substitution instance of $(\forall\mathbf{x})\mathbf{P}$.

Proof: Let $(\forall\mathbf{x})\mathbf{P}$ be any universally quantified sentence, let $\mathbf{P}(\mathbf{a}/\mathbf{x})$ be a substitution instance of $(\forall\mathbf{x})\mathbf{P}$, and let **I** be an interpretation on which $(\forall\mathbf{x})\mathbf{P}$ is true. Then, by 11.1.3, every variable assignment satisfies $(\forall\mathbf{x})\mathbf{P}$, and so, for every variable assignment $\mathbf{d_I}$ and every member **u** of the UD, $\mathbf{d_I}[\mathbf{u}/\mathbf{x}]$ satisfies **P**. In particular, for every variable assignment $\mathbf{d_I}$ the variant $\mathbf{d_I}[\mathbf{I}(\mathbf{a})/\mathbf{x}]$ must satisfy **P**. By 11.1.1, then, every variable assignment $\mathbf{d_I}$ satisfies $\mathbf{P}(\mathbf{a}/\mathbf{x})$, so $\mathbf{P}(\mathbf{a}/\mathbf{x})$ is also true on **I**.

11.1.5: Every substitution instance $\mathbf{P}(\mathbf{a}/\mathbf{x})$ of an existentially quantified sentence $(\exists\mathbf{x})\mathbf{P}$ is such that $\{\mathbf{P}(\mathbf{a}/\mathbf{x})\} \models (\exists\mathbf{x})\mathbf{P}$.

Proof: See Exercise 3.

11.1.4 and 11.1.5 are results that were used to motivate informally two of the quantifier rules in Chapter 10, Universal Elimination and Existential Introduction, and they will play a role in our proof of the soundness of *PD*. We also want to ensure that the motivations for Universal Introduction and Existential Elimination were correct. We'll first establish two further results that we shall need:

11.1.6: Let **I** and **I′** be interpretations that have the same UD and that agree on the assignments made to each individual constant, predicate, and sentence letter in a formula **P** (that is, **I** and **I′** assign the same values to those symbols). Then each variable assignment $\mathbf{d_I}$ satisfies **P** on interpretation **I** if and only if $\mathbf{d_I}$ satisfies **P** on interpretation **I′**.

In stating result 11.1.6, we have made use of the fact that if two interpretations have the same UD, then every variable assignment for one interpretation is a variable assignment for the other. The result should sound obvious: If two interpretations with identical universes of discourse treat the nonlogical symbols of **P** in the same way, and if the free variables are interpreted the same way on the two interpretations, then **P** says the same thing on both interpretations, and the values that **I** and **I′** assign to other symbols of *PL* have no bearing on what **P** says.

Proof of 11.1.6: Let **P** be a formula of *PL* and let **I** and **I′** be interpretations that have the same UD and that agree on the values assigned to

each nonlogical symbol in **P**. We shall prove, by mathematical induction on the number of occurrences of logical operators in **P**, that a variable assignment satisfies **P** on interpretation **I** if and only if it satisfies **P** on interpretation **I′**.

Basis clause: If **P** contains zero occurrences of logical operators, then a variable assignment $\mathbf{d_I}$ satisfies **P** on **I** if and only if it satisfies **P** on **I′**.

Proof of basis clause: If **P** is a sentence letter, then $\mathbf{d_I}$ satisfies **P** on **I** if and only if $\mathbf{I}(\mathbf{P}) = \mathbf{T}$, and it satisfies **P** on **I′** if and only if $\mathbf{I'}(\mathbf{P}) = \mathbf{T}$. By our assumption, $\mathbf{I}(\mathbf{P}) = \mathbf{I'}(\mathbf{P})$, so $\mathbf{d_I}$ satisfies **P** on **I** if and only if $\mathbf{d_I}$ satisfies **P** on **I′**.

If **P** is an atomic formula $\mathbf{At}_1 \ldots \mathbf{t_n}$,

- By the definition of satisfaction, $\mathbf{d_I}$ satisfies **P** on **I** if and only if $<\text{den}_{\mathbf{I},\mathbf{d_I}}(\mathbf{t}_1), \text{den}_{\mathbf{I},\mathbf{d_I}}(\mathbf{t}_2), \ldots, \text{den}_{\mathbf{I},\mathbf{d_I}}(\mathbf{t_n})> \in \mathbf{I}(\mathbf{A})$, and $\mathbf{d_I}$ satisfies **P** on **I′** if and only if $<\text{den}_{\mathbf{I'},\mathbf{d_I}}(\mathbf{t}_1), \text{den}_{\mathbf{I'},\mathbf{d_I}}(\mathbf{t}_2)> \ldots <\text{den}_{\mathbf{I'},\mathbf{d_I}}(\mathbf{t_n})> \in \mathbf{I_}(\mathbf{A})$.
- We note that $<\text{den}_{\mathbf{I},\mathbf{d_I}}(\mathbf{t}_1), \text{den}_{\mathbf{I},\mathbf{d_I}}(\mathbf{t}_2), \ldots, \text{den}_{\mathbf{I},\mathbf{d_I}}(\mathbf{t_n})>$ and $<\text{den}_{\mathbf{I'},\mathbf{d_I}}(\mathbf{t}_1), \text{den}_{\mathbf{I'},\mathbf{d_I}}(\mathbf{t}_2), \ldots, \text{den}_{\mathbf{I'},\mathbf{d_I}}(\mathbf{t_n})>$ are identical. This is because if $\mathbf{t_i}$ is a constant, then $\text{den}_{\mathbf{I},\mathbf{d_I}}(\mathbf{t}_1) = \mathbf{I}(\mathbf{t_i})$, $\text{den}_{\mathbf{I'},\mathbf{d_I}}(\mathbf{t}_1) = \mathbf{I'}(\mathbf{t_i})$, and $\mathbf{I'}(\mathbf{t_i}) = \mathbf{I}(\mathbf{t_i})$ since by assumption, **I** and **I′** assign the same values to the nonlogical symbols in **P**; and if $\mathbf{t_i}$ is a variable, then $\text{den}_{\mathbf{I},\mathbf{d_I}}(\mathbf{t_i}) = \mathbf{d_I}(\mathbf{t_i}) = \text{den}_{\mathbf{I'},\mathbf{d_I}}(\mathbf{t_i})$.
- Moreover, $\mathbf{I}(\mathbf{A}) = \mathbf{I'}(\mathbf{A})$, by stipulation.
- It follows that $<\text{den}_{\mathbf{I},\mathbf{d_I}}(\mathbf{t}_1), \text{den}_{\mathbf{I},\mathbf{d_I}}(\mathbf{t}_2), \ldots, \text{den}_{\mathbf{I},\mathbf{d_I}}(\mathbf{t_n})> \in \mathbf{I}(\mathbf{A})$ if and only if $<\text{den}_{\mathbf{I'},\mathbf{d_I}}(\mathbf{t}_1), \text{den}_{\mathbf{I'},\mathbf{d_I}}(\mathbf{t}_2), \ldots, \text{den}_{\mathbf{I'},\mathbf{d_I}}(\mathbf{t_n})> \in \mathbf{I'}(\mathbf{A})$.

Therefore, $\mathbf{d_I}$ satisfies $\mathbf{At}_1 \ldots \mathbf{t_n}$ on **I** if and only if it does so on **I′**.

Inductive step: If every formula **P** that has **k** or fewer occurrences of logical operators is such that a variable assignment satisfies **P** on **I** if and only if it satisfies **P** on **I′**, then the same is true of every formula **P** that has **k** + 1 occurrences of logical operators.

Proof of inductive step: We consider the forms that **P** may have.

Case 1: P has the form ~ **Q**.

- By the definition of satisfaction, a variable assignment $\mathbf{d_I}$ satisfies ~ **Q** on **I** if and only if it fails to satisfy **Q** on **I**.
- Because **Q** contains fewer than **k** + 1 occurrences of logical operators, it follows from the inductive hypothesis that a variable assignment fails to satisfy **Q** on **I** if and only if it fails to satisfy **Q** on **I′**.
- A variable assignment fails to satisfy **Q** on **I′** if and only if it does satisfy ~ **Q** on **I′**.

Therefore, a variable assignment satisfies ~ **Q** on **I** if and only if it satisfies ~ **Q** on **I′**.

Case 2: P has the form **Q** & **R**.

- By the definition of satisfaction, a variable assignment satisfies **Q** & **R** on **I** if and only if it satisfies both **Q** and **R** on **I**.
- **Q** and **R** each contain **k** or fewer occurrences of logical operators, and so by the inductive hypothesis, a variable assignment satisfies both **Q** and **R** on **I** if and only if it satisfies both **Q** and **R** on **I′**.
- A variable assignment satisfies both **Q** and **R** on **I′** if and only if it satisfies **Q** & **R** on **I′**.

Therefore, a variable assignment satisfies **Q** & **R** on **I** if and only if it satisfies **Q** & **R** on **I′**.

Cases 3–5: P has the form **Q** ∨ **R**, **Q** ⊃ **R**, or **Q** ≡ **R**. We omit proofs for these cases as they are strictly analogous to Case 2.

Case 6: P has the form (∀**x**)**Q**.

- By the definition of satisfaction, a variable assignment $\mathbf{d_I}$ satisfies (∀**x**)**Q** on **I** if and only if every member **u** of **I**'s UD is such that $\mathbf{d_I}[\mathbf{u}/\mathbf{x}]$ satisfies **Q** on **I**.
- $\mathbf{d_I}$ satisfies (∀**x**)**Q** on **I′** if and only if every member **u** of **I′**'s UD (which is the same as **I**'s UD) is such that $\mathbf{d_I}[\mathbf{u}/\mathbf{x}]$ satisfies **Q** on **I′**.
- Because **Q** contains fewer than **k** + 1 occurrences of logical operators, it follows from the inductive hypothesis that every member **u** of the common UD is such that $\mathbf{d_I}[\mathbf{u}/\mathbf{x}]$ satisfies **Q** on **I** if and only if $\mathbf{d_I}[\mathbf{u}/\mathbf{x}]$ satisfies **Q** on **I′**.

Therefore, a variable assignment satisfies (∀**x**)**Q** on **I** if and only if it satisfies (∀**x**)**Q** on **I′**.

Case 7: P has the form (∃**x**)**Q**. This case is similar to Case 6.

That completes the proof of the inductive step, and we may now conclude that 11.1.6 is true.

Result 11.1.7 follows as an immediate consequence of 11.1.6:

11.1.7: Let **I** and **I′** be interpretations that have the same UD and that agree on the assignments made to each individual constant, predicate, and sentence letter in a sentence **P**. Then **P** is true on **I** if and only if **P** is true on **I′**.

Proof: Let $\mathbf{I}$ and $\mathbf{I}'$ be as specified for a sentence $\mathbf{P}$. If $\mathbf{P}$ is true on $\mathbf{I}$, then, by 11.1.2, $\mathbf{P}$ is satisfied by every variable assignment on $\mathbf{I}$. By 11.1.6, this is the case if and only if $\mathbf{P}$ is satisfied by every variable assignment on $\mathbf{I}'$, that is, if and only if $\mathbf{P}$ is true on $\mathbf{I}'$.

With results 11.1.6 and 11.1.7 at hand, we may now show that our motivations for the rules Universal Introduction and Existential Elimination are correct.

11.1.8: Let $\mathbf{a}$ be a constant that does not occur in $(\forall\mathbf{x})\mathbf{P}$ or in any member of the set Γ. Then if $\Gamma \vDash \mathbf{P}(\mathbf{a}/\mathbf{x})$, $\Gamma \vDash (\forall\mathbf{x})\mathbf{P}$.

11.1.9: Let $\mathbf{a}$ be a constant that does not occur in the sentences $(\exists\mathbf{x})\mathbf{P}$ and $\mathbf{Q}$ and that does not occur in any member of the set Γ. If $\Gamma \vDash (\exists\mathbf{x})\mathbf{P}$ and $\Gamma \cup \{\mathbf{P}(\mathbf{a}/\mathbf{x})\} \vDash \mathbf{Q}$, then $\Gamma \vDash \mathbf{Q}$ as well.

We shall prove 11.1.8 here; 11.1.9 is left as an exercise.

Proof of 11.1.8: Assume that $\Gamma \vDash \mathbf{P}(\mathbf{a}/\mathbf{x})$, where $\mathbf{a}$ does not occur in $(\forall\mathbf{x})\mathbf{P}$ or in any member of Γ. We shall assume, contrary to what we want to show, that Γ does not quantificationally entail $(\forall\mathbf{x})\mathbf{P}$—that there is at least one interpretation, call it $\mathbf{I}$, on which every member of Γ is true and $(\forall\mathbf{x})\mathbf{P}$ is false. We shall use $\mathbf{I}$ as the basis for constructing an interpretation $\mathbf{I}'$ on which every member of Γ is true and the substitution instance $\mathbf{P}(\mathbf{a}/\mathbf{x})$ is false, contradicting our original assumption. Having done so, we may conclude that if Γ does quantificationally entail $\mathbf{P}(\mathbf{a}/\mathbf{x})$, it must also quantificationally entail $(\forall\mathbf{x})\mathbf{P}$.

So assume that $\mathbf{I}$ is an interpretation on which every member of Γ is true and on which $(\forall\mathbf{x})\mathbf{P}$ is false. Because $(\forall\mathbf{x})\mathbf{P}$ is false, there is no variable assignment for $\mathbf{I}$ that satisfies $(\forall\mathbf{x})\mathbf{P}$. That is, for every variable assignment $\mathbf{d}_\mathbf{I}$, there is at least one member $\mathbf{u}$ of the UD such that $\mathbf{d}_\mathbf{I}[\mathbf{u}/\mathbf{x}]$ does not satisfy $\mathbf{P}$. Choose one of these members, calling it $\mathbf{u}$, and let $\mathbf{I}'$ be the interpretation that is just like $\mathbf{I}$ except that it assigns $\mathbf{u}$ to $\mathbf{a}$ (all other assignments made by $\mathbf{I}$ remain the same). It is now straightforward to show that every member of Γ is true on $\mathbf{I}'$ and $\mathbf{P}(\mathbf{a}/\mathbf{x})$ is false. That every member of Γ is true on $\mathbf{I}'$ follows from 11.1.7 because $\mathbf{I}$ and $\mathbf{I}'$ assign the same values to all the nonlogical symbols of *PL* other than $\mathbf{a}$, and, by stipulation, $\mathbf{a}$ does not occur in any member of Γ.

On our assumption that $\mathbf{d}[\mathbf{u}/\mathbf{x}]$ does not satisfy $\mathbf{P}$ on $\mathbf{I}$, it follows from 11.1.6 that $\mathbf{d}[\mathbf{u}/\mathbf{x}]$ does not satisfy $\mathbf{P}$ on $\mathbf{I}'$. By the way we have constructed $\mathbf{I}'$, $\mathbf{u}$ is $\mathbf{I}'(\mathbf{a})$ and so $\mathbf{d}_\mathbf{I}[\mathbf{u}/\mathbf{x}]$ is $\mathbf{d}_\mathbf{I}[\mathbf{I}'(\mathbf{a})/\mathbf{x}]$. Result 11.1.1 tells us that $\mathbf{d}_\mathbf{I}[\mathbf{I}'(\mathbf{a})/\mathbf{x}]$ satisfies $\mathbf{P}$ on $\mathbf{I}'$ if and only if $\mathbf{d}_\mathbf{I}$ satisfies $\mathbf{P}(\mathbf{a}/\mathbf{x})$ on $\mathbf{I}'$. So, because $\mathbf{d}_\mathbf{I}[\mathbf{I}'(\mathbf{a})/\mathbf{x}]$ does not satisfy $\mathbf{P}$ on $\mathbf{I}'$, $\mathbf{d}_\mathbf{I}$ does not satisfy $\mathbf{P}(\mathbf{a}/\mathbf{x})$ on $\mathbf{I}'$. By 11.1.3, then, no variable assignment satisfies $\mathbf{P}(\mathbf{a}/\mathbf{x})$ on $\mathbf{I}'$, and it is therefore false on this interpretation. But this contradicts our first assumption, that $\Gamma \vDash \mathbf{P}(\mathbf{a}/\mathbf{x})$, and so we conclude that if $\Gamma \vDash \mathbf{P}(\mathbf{a}/\mathbf{x})$, then $\Gamma \vDash (\forall\mathbf{x})\mathbf{P}$ as well.

Result 11.1.8 tells us that the rule Universal Introduction is indeed truth-preserving.

We shall state four more semantic results that will be needed in the sections that follow and that the reader should now be able to prove. The proofs are left as exercises. The first result relies on 11.1.6 and 11.1.7, much as the proofs of 11.1.8 and 11.1.9 do.

> **11.1.10:** If **a** does not occur in any member of the set $\Gamma \cup \{(\exists \mathbf{x})\mathbf{P}\}$ and if the set is quantificationally consistent, then the set $\Gamma \cup \{(\exists \mathbf{x})\mathbf{P}, \mathbf{P}(\mathbf{a}/\mathbf{x})\}$ is also quantificationally consistent.

Results 11.1.11 and 11.1.12 concern interpretations of a special sort: interpretations on which every member of the UD has a name.

> **11.1.11:** Let **I** be an interpretation on which each member of the UD is assigned to at least one individual constant. Then, if every substitution instance of $(\forall \mathbf{x})\mathbf{P}$ is true on **I**, so is $(\forall \mathbf{x})\mathbf{P}$.
>
> **11.1.12:** Let **I** be an interpretation on which each member of the UD is assigned to at least one individual constant. Then, if every substitution instance of $(\exists \mathbf{x})\mathbf{P}$ is false on **I**, so is $(\exists \mathbf{x})\mathbf{P}$.

Result 11.1.13 says that, if we rename the individual designated by some individual constant in a sentence **P** with a constant that does not already occur in **P**, then, for any interpretation on which **P** is true, there is a closely related interpretation (one that reflects the renaming) on which the new sentence is true:

> **11.1.13:** Let **P** be a sentence of *PL*, let **b** be an individual constant that does not occur in **P**, and let $\mathbf{P}(\mathbf{b}/\mathbf{a})$ be the sentence that results from replacing every occurrence of the individual constant **a** in **P** with **b**. Then if **P** is true on an interpretation **I**, $\mathbf{P}(\mathbf{b}/\mathbf{a})$ is true on the interpretation $\mathbf{I}'$ that is just like **I** except that it assigns $\mathbf{I}(\mathbf{a})$ to **b** ($\mathbf{I}'(\mathbf{b}) = \mathbf{I}(\mathbf{a})$).

11.1**E** EXERCISES

***1.** Prove Cases 3–5 in the proof of result 11.1.1.

***2.** Prove Cases 3–5 and 7 in the proof of result 11.1.2.

***3.** Prove result 11.1.5.

***4.** Prove result 11.1.9.

5. Prove result 11.1.10.

6. Prove result 11.1.11.

***7.** Prove result 11.1.12.

***8.** Prove result 11.1.13.

11.2 SEMANTIC PRELIMINARIES FOR *PLE*

When we turn to the metatheory for *PLE*, we shall need versions of Section 11.1's semantic results that apply to sentences containing the identity operator and complex terms. In this section we discuss the changes that must be made in the statement of the semantic results and in their proofs.

Starting with 11.1.1, we note that we must generalize the result to read:

> Let **P** be a formula of *PLE*, let $\mathbf{P}(\mathbf{t}/\mathbf{x})$ be the formula that results from replacing every free occurrence of **x** in **P** with a closed term **t**, let **I** be an interpretation, let $\mathbf{d_I}$ be a variable assignment for **I**, and let $\mathbf{d_I}' = \mathbf{d_I}[\text{den}_{\mathbf{I},\mathbf{d_I}}(\mathbf{t})/\mathbf{x}]$ (that is, $\mathbf{d_I}'$ is just like $\mathbf{d_I}$ except that it assigns to **x** whatever $\mathbf{d_I}$ and **I** assign to **t**). Then $\mathbf{d_I}$ satisfies $\mathbf{P}(\mathbf{t}/\mathbf{x})$ on **I** if and only if $\mathbf{d_I}'$ satisfies **P** on **I**.

To modify the proof of 11.1.1, we first establish a result concerning complex terms:

> **11.2.1:** Let **t** be a complex term of *PLE*, let $\mathbf{t}(\mathbf{c}/\mathbf{x})$ be the term that results from replacing every occurrence of the variable **x** in **t** with a closed term **c**, let **I** be an interpretation, let $\mathbf{d_I}$ be a variable assignment for **I**, and let $\mathbf{d_I}' = \mathbf{d_I}[\text{den}_{\mathbf{I},\mathbf{d_I}}(\mathbf{c})/\mathbf{x}]$. Then $\text{den}_{\mathbf{I},\mathbf{d_I}'}(\mathbf{t}) = \text{den}_{\mathbf{I},\mathbf{d_I}}(\mathbf{t}(\mathbf{c}/\mathbf{x}))$.

The result states that, if the sole difference between two complex terms $\mathbf{t}_1$ and $\mathbf{t}_2$ is that one contains the closed term **c** where the other contains the variable **x**, then $\mathbf{t}_1$ and $\mathbf{t}_2$ denote the same individual if **x** and **c** do. We shall prove 11.2.1 by mathematical induction on the number of occurrences of functors in the term.

> *Basis clause:* If a complex term **t** contains one functor, then $\text{den}_{\mathbf{I},\mathbf{d_I}'}(\mathbf{t}) = \text{den}_{\mathbf{I},\mathbf{d_I}}(\mathbf{t}(\mathbf{c}/\mathbf{x}))$.
>
> **Proof of basis clause:** If a complex term **t** contains one functor, then **t** is $f(\mathbf{t}_1, \ldots, \mathbf{t}_n)$ where *f* is a functor, each $\mathbf{t_i}$ is either a variable or a constant, and $\mathbf{t}(\mathbf{c}/\mathbf{x})$ is $f(\mathbf{t}_1', \ldots, \mathbf{t}_n')$ where $\mathbf{t_i}'$ is $\mathbf{t_i}$ if $\mathbf{t_i}$ is not **x**, and $\mathbf{t_i}'$ is **c** if $\mathbf{t_i}$ is **x**.
>
> As in the proof of the basis clause of 11.1.1, we note that if $\mathbf{t_i}$ is a constant or variable other than **x,** $\text{den}_{\mathbf{I},\mathbf{d_I}'}(\mathbf{t_i}) = \text{den}_{\mathbf{I},\mathbf{d_I}}(\mathbf{t_i})$—since $\mathbf{d_I}'$ does not differ from $\mathbf{d_I}$ in a way that affects the denotation of these terms. If $\mathbf{t_i}$ is **x**, then $\mathbf{t_i}'$ is **c**, and by the definition of how $\mathbf{d_I}'$ was constructed, $\text{den}_{\mathbf{I},\mathbf{d_I}'}(\mathbf{x}) = \text{den}_{\mathbf{I},\mathbf{d_I}}(\mathbf{c})$. So we know that $<\text{den}_{\mathbf{I},\mathbf{d_I}'}(\mathbf{t}_1), \text{den}_{\mathbf{I},\mathbf{d_I}'}(\mathbf{t}_2), \ldots, \text{den}_{\mathbf{I},\mathbf{d_I}'}(\mathbf{t_n})>$ and $<\text{den}_{\mathbf{I},\mathbf{d_I}}(\mathbf{t}_1'), \text{den}_{\mathbf{I},\mathbf{d_I}}(\mathbf{t}_2'), \ldots, \text{den}_{\mathbf{I},\mathbf{d_I}'}(\mathbf{t_n}')>$ are identical. Therefore the **n**+1-tuple $<\text{den}_{\mathbf{I},\mathbf{d_I}'}(\mathbf{t}_1), \text{den}_{\mathbf{I},\mathbf{d_I}'}(\mathbf{t}_2), \ldots, \text{den}_{\mathbf{I},\mathbf{d_I}'}(\mathbf{t_n}), \mathbf{u}> \in \mathbf{I}(f)$ if and only if $<\text{den}_{\mathbf{I},\mathbf{d_I}}(\mathbf{t}_1'), \text{den}_{\mathbf{I},\mathbf{d_I}}(\mathbf{t}_2'), \ldots, \text{den}_{\mathbf{I},\mathbf{d_I}}(\mathbf{t_n}'), \mathbf{u}> \in \mathbf{I}(f)$, so $\text{den}_{\mathbf{I},\mathbf{d_I}'}(f(\mathbf{t}_1, \ldots, \mathbf{t_n})) = \text{den}_{\mathbf{I},\mathbf{d_I}}(f(\mathbf{t}_1', \ldots, \mathbf{t_n}'))$.

Inductive step: If every complex term **t** that contains **k** or fewer functors is such that $\text{den}_{\mathbf{I},\mathbf{d_I}'}(\mathbf{t}) = \text{den}_{\mathbf{I},\mathbf{d_I}'}(\mathbf{t}(\mathbf{c}/\mathbf{x}))$, then the same is true of every complex term that contains **k** + 1 functors.

Proof of inductive step: We assume that the inductive hypothesis holds—that the claim is true of every complex term that contains **k** or fewer functors, for some arbitrary positive integer **k**. We must show that the claim is also true of every complex term that contains **k** + 1 functors. If **t** contains **k** + 1 functors, then **t** is $f(\mathbf{t_1}, \ldots, \mathbf{t_n})$, where each $\mathbf{t_i}$ has **k** or fewer functors, and $\mathbf{t}(\mathbf{c}/\mathbf{x})$ is $f(\mathbf{t_1}', \ldots, \mathbf{t_n}')$, where each $\mathbf{t_i}'$ is $\mathbf{t_i}(\mathbf{c}/\mathbf{x})$. So each $\mathbf{t_i}$ falls under the inductive hypothesis; that is, $\text{den}_{\mathbf{I},\mathbf{d_I}'}(\mathbf{t_i}) = \text{den}_{\mathbf{I},\mathbf{d_I}}(\mathbf{t_i}(\mathbf{t_c}/\mathbf{x}))$, and so $<\text{den}_{\mathbf{I},\mathbf{d_I}'}(\mathbf{t_1}), \text{den}_{\mathbf{I},\mathbf{d_I}'}(\mathbf{t_2}), \ldots, \text{den}_{\mathbf{I},\mathbf{d_I}'}(\mathbf{t_n})>$ and $<\text{den}_{\mathbf{I},\mathbf{d_I}}(\mathbf{t_1}'), \text{den}_{\mathbf{I},\mathbf{d_I}}(\mathbf{t_2}'), \ldots, \text{den}_{\mathbf{I},\mathbf{d_I}}(\mathbf{t_n}')>$ are identical. Therefore, $<\text{den}_{\mathbf{I},\mathbf{d_I}'}(\mathbf{t_1}), \text{den}_{\mathbf{I},\mathbf{d_I}'}(\mathbf{t_2}), \ldots, \text{den}_{\mathbf{I},\mathbf{d_I}'}(\mathbf{t_n}), \mathbf{u}> \in \mathbf{I}(f)$ if and only if $<\text{den}_{\mathbf{I},\mathbf{d_I}}(\mathbf{t_1}'), \text{den}_{\mathbf{I},\mathbf{d_I}}(\mathbf{t_2}'), \ldots, \text{den}_{\mathbf{I},\mathbf{d_I}}(\mathbf{t_n}'), \mathbf{u}> \in \mathbf{I}(f)$, so $\text{den}_{\mathbf{I},\mathbf{d_I}'}(f(\mathbf{t_1}, \ldots, \mathbf{t_n})) = \text{den}_{\mathbf{I},\mathbf{d_I}}(f(\mathbf{t_1}', \ldots, \mathbf{t_n}'))$.

With result 11.2.1 in hand, we can modify the basis clause of result 11.1.1 as follows:

Basis clause: If **P** is a formula that contains zero occurrences of logical operators, then $\mathbf{d_I}$ satisfies $\mathbf{P}(\mathbf{t}/\mathbf{x})$ if and only if $\mathbf{d_I}'$ satisfies **P**.

Proof of basis clause: The proof is identical to the proof of 11.1.1, except that we must change the second part of the basis clause so that it covers complex terms and we must also consider atomic identity formulas.

If **P** has the form $\mathbf{A}\mathbf{t_1} \ldots \mathbf{t_n}$, then $\mathbf{P}(\mathbf{t}/\mathbf{x})$ is $\mathbf{A}\mathbf{t_1}' \ldots \mathbf{t_n}'$, where $\mathbf{t_i}'$ is **t** if $\mathbf{t_i}$ is **x** and $\mathbf{t_i}'$ is just $\mathbf{t_i}$ otherwise. By the definition of satisfaction,

- $\mathbf{d_I}$ satisfies $\mathbf{A}\mathbf{t_1}' \ldots \mathbf{t_n}'$ if and only if $<\text{den}_{\mathbf{I},\mathbf{d_I}}(\mathbf{t_1}'), \text{den}_{\mathbf{I},\mathbf{d_I}}(\mathbf{t_2}'), \ldots, \text{den}_{\mathbf{I},\mathbf{d_I}}(\mathbf{t_n}')> \in \mathbf{I}(\mathbf{A})$.
- $\mathbf{d_I}'$ satisfies $\mathbf{A}\mathbf{t_1} \ldots \mathbf{t_n}$ if and only if $<\text{den}_{\mathbf{I},\mathbf{d_I}'}(\mathbf{t_1}), \text{den}_{\mathbf{I},\mathbf{d_I}'}(\mathbf{t_2}), \ldots, \text{den}_{\mathbf{I},\mathbf{d_I}'}(\mathbf{t_n})> \in \mathbf{I}(\mathbf{A})$.

But now we note that

- $<\text{den}_{\mathbf{I},\mathbf{d_I}}(\mathbf{t_1}'), \text{den}_{\mathbf{I},\mathbf{d_I}}(\mathbf{t_2}'), \ldots, \text{den}_{\mathbf{I},\mathbf{d_I}}(\mathbf{t_n}')>$ and $<\text{den}_{\mathbf{I},\mathbf{d_I}'}(\mathbf{t_1}), \text{den}_{\mathbf{I},\mathbf{d_I}'}(\mathbf{t_2}), \ldots, \text{den}_{\mathbf{I},\mathbf{d_I}'}(\mathbf{t_n})>$ are identical.

Consider: If $\mathbf{t_i}$ is a constant, then $\mathbf{t_i}'$ is $\mathbf{t_i}$ and so $\text{den}_{\mathbf{I},\mathbf{d_I}}(\mathbf{t_i}') = \mathbf{I}(\mathbf{t_i}) = \text{den}_{\mathbf{I},\mathbf{d_I}'}(\mathbf{t_i})$. If $\mathbf{t_i}$ is a variable other than **x**, then $\mathbf{t_i}'$ is $\mathbf{t_i}$ and so $\text{den}_{\mathbf{I},\mathbf{d_I}}(\mathbf{t_i}') = \mathbf{d_I}(\mathbf{t_i}) = \mathbf{d_I}'(\mathbf{t_i}) = \text{den}_{\mathbf{I},\mathbf{d_I}'}(\mathbf{t_i})$—the variation in the variable assignment does not affect the value assigned to $\mathbf{t_i}$ in this case. If $\mathbf{t_i}$ is the variable **x**, then $\mathbf{t_i}'$ is **t** and $\text{den}_{\mathbf{I},\mathbf{d_I}}(\mathbf{t}) = \text{den}_{\mathbf{I},\mathbf{d_I}'}(\mathbf{x})$. (The variant $\mathbf{d_I}'$ was defined in a way that ensures that the denotations of **x** and of **t** coincide.) And if $\mathbf{t_i}$ is a complex term, it follows from 11.2.1 that $\text{den}_{\mathbf{I},\mathbf{d_I}}(\mathbf{t_i}') = \text{den}_{\mathbf{I},\mathbf{d_I}'}(\mathbf{t_i})$.

Because the **n**-tuples are the same, we conclude that $\mathbf{d_I}$ satisfies $\mathbf{A}\mathbf{t_1}' \ldots \mathbf{t_n}'$ if and only if $\mathbf{d_I}'$ satisfies $\mathbf{A}\mathbf{t_1} \ldots \mathbf{t_n}$.

We must also add a new case to the proof of the basis clause, to cover formulas of the form $\mathbf{t}_1 = \mathbf{t}_2$. We leave this as an exercise. The rest of the proof of 11.1.1 remains the same, except that we replace $\mathbf{d_I}[\mathbf{I}(\mathbf{a})/\mathbf{x}]$ with $\mathbf{d}'_\mathbf{I}$ (which is shorthand for $\mathbf{d_I}[\text{den}_{\mathbf{I},\mathbf{d_I}}(\mathbf{t_c})/\mathbf{x}]$) throughout.

The proof of 11.1.2 is modified in a similar way. First, we need to prove

> **11.2.2:** Let **I** be an interpretation, $\mathbf{d_I}$ a variable assignment for **I**, and **t** a complex term of *PLE*. Then, for any variable assignment $\mathbf{d_I}'$ that assigns the same values to the variables in **t** as $\mathbf{d_I}$, $\text{den}_{\mathbf{I},\mathbf{d_I}'}(\mathbf{t}) = \text{den}_{\mathbf{I},\mathbf{d_I}}(\mathbf{t})$.
>
> **Proof:** See Exercise 2.

With this result at hand, the basis clause in the proof of 11.1.2 can now be modified to include atomic formulas containing complex terms and also to include formulas of the form $\mathbf{t}_1 = \mathbf{t}_2$. Both modifications are left as exercises.

The proof of result 11.1.6 can be similarly modified, once we have established

> **11.2.3:** Let **t** be a complex term of *PLE* and let **I** and **I′** be interpretations that have the same UD and that agree on the values assigned to each individual constant and functor in **t**. Then, for any variable assignment $\mathbf{d_I}$, $\text{den}_{\mathbf{I},\mathbf{d_I}}(\mathbf{t}) = \text{den}_{\mathbf{I}',\mathbf{d_I}}(\mathbf{t})$.
>
> **Proof:** See Exercise 11.2.3.

Result 11.1.6 must itself be changed to say:

> Let **I** and **I′** be interpretations that have the same UD and that agree on the assignments made to each individual constant, functor, predicate, and sentence letter in a formula **P**. Then each variable assignment $\mathbf{d_I}$ satisfies **P** on interpretation **I** if and only if $\mathbf{d_I}$ satisfies **P** on interpretation **I′**.

The basis clause must be modified to cover formulas containing complex terms, as well as formulas of the form $\mathbf{t}_1 = \mathbf{t}_2$. This is left as an exercise.

The proofs of results 11.1.3–11.1.5 and 11.1.7–11.1.13 for *PLE* are the same as for *PL*, except for the following changes:

1. The proofs must use the modified versions of 11.1.1, 11.1.2, and 11.1.6 in order to apply to *PLE*.
2. Where '**a**' and '$\mathbf{P}(\mathbf{a}/\mathbf{x})$' are used in results 11.1.4 and 11.1.5 to refer to substitution instances of **P**, we need to use '**t**' and '$\mathbf{P}(\mathbf{t}/\mathbf{x})$' instead to allow for instantiation with arbitrary closed terms.
3. In 11.1.7, **I** and **I′** must also agree on the assignments made to each functor.

4. Results 1.1.11 and 1.1.12 are true for *PLE* in two senses: We can change 'every substitution instance' to 'every substitution instance in which the instantiating individual term is a constant', or we can leave the phrase as it is, to include substitution instances with instantiation by all closed terms, complex ones as well as constants.

Finally we shall need two additional semantic results for *PLE*:

11.2.4: For any closed terms $\mathbf{t}_1$ and $\mathbf{t}_2$, if $\mathbf{P}$ is a sentence that contains $\mathbf{t}_1$, then $\{\mathbf{t}_1 = \mathbf{t}_2, \mathbf{P}\} \vDash \mathbf{P}(\mathbf{t}_2//\mathbf{t}_1)$, and if $\mathbf{P}$ is a sentence that contains $\mathbf{t}_2$, then $\{\mathbf{t}_1 = \mathbf{t}_2, \mathbf{P}\} \vDash \mathbf{P}(\mathbf{t}_1//\mathbf{t}_2)$.

Proof: See Exercise 11.2.4.

11.2.5: If a quantificationally consistent set Γ contains a sentence with a complex term $f(\mathbf{a}_1, \ldots, \mathbf{a}_n)$, where $\mathbf{a}_1, \ldots, \mathbf{a}_n$ are constants, and the constant $\mathbf{b}$ does not occur in Γ, then the set $\Gamma \cup \{\mathbf{b} = f(\mathbf{a}_1, \ldots, \mathbf{a}_n)\}$ is also quantificationally consistent.

Proof: See Exercise 11.2.5.

11.2E EXERCISES

***1.** Show the changes that must be made in the basis clauses of the proofs of the following results so that they cover formulas containing complex terms and formulas of the form $\mathbf{t}_1 = \mathbf{t}_2$:
a. Result 11.1.1
b. Result 11.1.2
c. Result 11.1.6

***2.** Prove result 11.2.2.

***3.** Prove result 11.2.3.

4. Prove result 11.2.4.

***5.** Prove result 11.2.5.

11.3 THE SOUNDNESS OF *PD*, *PD*+, AND *PDE*

We shall now establish the soundness of our natural deduction systems. A natural deduction system is said to be **sound** for predicate logic if every rule in that system is truth-preserving. The *Soundness Metatheorem* for *PD* is

Metatheorem 11.3.1: For any set Γ of sentences of *PL* and any sentence $\mathbf{P}$ of *PL*, if $\Gamma \vdash \mathbf{P}$ in *PD*, then $\Gamma \vDash \mathrm{P}$.

(As in Chapter 6, we shall drop 'in *PD*' when we use the single turnstile in this chapter, and we shall use the double turnstile to signify *quantificational* entailment.) We shall prove metatheorem 11.3.1 by mathematical induction and in outline, the proof will be like the proof that we presented in Chapter 6 establishing the soundness of *SD* for sentential logic. In fact, much of the proof in Chapter 6 can be used here—for in Chapter 6 we showed that the rules for the truth-functional connectives are all sound for sentential logic, and with a change from talk of truth-value assignments to talk of interpretations, those rules are established to be sound for predicate logic in the same way. The bulk of the proof will therefore concentrate on the rules for quantifier introduction and elimination.

In our proof we shall use several semantic results that were presented in Section 11.1 along with the following result:

11.3.2: If $\Gamma \models \mathbf{P}$ and Γ^* is a superset of Γ, then $\Gamma^* \models \mathbf{P}$.

Proof: If every member of Γ^* is true on an interpretation **I**, then every member of its subset Γ is true on **I**, and if $\Gamma \models \mathbf{P}$, then **P** is also true on **I**. Hence $\Gamma^* \models \mathbf{P}$.

Letting $\mathbf{P_i}$ be the sentence at position **i** in a derivation and letting $\Gamma_\mathbf{i}$ be the set of assumptions that are open at position **i** (and hence within whose scope $\mathbf{P_i}$ lies), the proof of Metatheorem 11.3.1 by mathematical induction is

Basis clause: $\Gamma_1 \models \mathbf{P}_1$.

Inductive step: If $\Gamma_\mathbf{i} \models \mathbf{P_i}$ for every position **i** in a derivation such that $\mathbf{i} \leq \mathbf{k}$, then $\Gamma_{\mathbf{k}+1} \models \mathbf{P}_{\mathbf{k}+1}$.

Conclusion: Every sentence in a derivation is quantificationally entailed by the set of open assumptions in whose scope it lies.

Proof of basis clause: The first sentence in any derivation in *PD* is an assumption, and it lies in its own scope. Γ_1 is just $\{\mathbf{P}_1\}$, and it is trivial that $\{\mathbf{P}_1\} \models \mathbf{P}_1$.

Proof of inductive step: We assume the inductive hypothesis for an arbitrary position **k**, that is, for every position **i** such that $\mathbf{i} \leq \mathbf{k}$, $\Gamma_\mathbf{i} \models \mathbf{P_i}$. We must show that the same holds for position $\mathbf{k} + 1$. We shall show this by considering the justifications that might be used for the sentence at position $\mathbf{k} + 1$.

Cases 1–12: $\mathbf{P}_{\mathbf{k}+1}$ is justified by one of the rules of *SD*. For each of these cases, use the corresponding case from the proof of the soundness of *SD* in Section 6.3, changing talk of truth-value assignments to talk of interpretations, and talk of truth-functional concepts (inconsistency and so on) to talk of quantificational concepts.

Case 13: $\mathbf{P_{k+1}}$ is justified by Universal Elimination. Then $\mathbf{P_{k+1}}$ is a sentence $\mathbf{Q(a/x)}$ derived as follows:

h	$(\forall \mathbf{x})\mathbf{Q}$	
k + 1	$\mathbf{Q(a/x)}$	**h** $\forall$E

where every assumption that is open at position **h** is also open at position **k** + 1, so $\Gamma_{\mathbf{h}}$ is a subset of $\Gamma_{\mathbf{k+1}}$. By the inductive hypothesis, $\Gamma_{\mathbf{h}} \models (\forall \mathbf{x})\mathbf{Q}$. It follows, by 11.3.2, that the superset $\Gamma_{\mathbf{k+1}} \models (\forall \mathbf{x})\mathbf{Q}$. By 11.1.4, which says that a universally quantified sentence quantificationally entails every one of its substitution instances, $\{(\forall \mathbf{x})\mathbf{Q}\} \models \mathbf{Q(a/x)}$. So $\Gamma_{\mathbf{k+1}} \models \mathbf{Q(a/x)}$ as well.

Case 14: $\mathbf{P_{k+1}}$ is justified by Existential Introduction. Then $\mathbf{P_{k+1}}$ is a sentence $(\exists \mathbf{x})\mathbf{Q}$ derived as follows:

h	$\mathbf{Q(a/x)}$	
k + 1	$(\exists \mathbf{x})\mathbf{Q}$	**h** $\exists$I

where every assumption that is open at position **h** is also open at position **k** + 1. So $\Gamma_{\mathbf{h}}$ is a subset of $\Gamma_{\mathbf{k+1}}$. By the inductive hypothesis, $\Gamma_{\mathbf{h}} \models \mathbf{Q(a/x)}$ and so, by 11.3.2, $\Gamma_{\mathbf{k+1}} \models \mathbf{Q(a/x)}$. By 11.1.5, $\{\mathbf{Q(a/x)}\} \models (\exists \mathbf{x})\mathbf{Q}$, so $\Gamma_{\mathbf{k+1}} \models (\exists \mathbf{x})\mathbf{Q}$ as well.

Case 15: $\mathbf{P_{k+1}}$ is justified by Universal Introduction. Then $\mathbf{P_{k+1}}$ is a sentence $(\forall \mathbf{x})\mathbf{Q}$ derived as follows:

h	$\mathbf{Q(a/x)}$	
k + 1	$(\forall \mathbf{x})\mathbf{Q}$	**h** $\forall$I

where every assumption that is open at position **h** is also open at position **k** + 1—so $\Gamma_{\mathbf{h}}$ is a subset of $\Gamma_{\mathbf{k+1}}$—and in addition, **a** does not occur in $(\forall \mathbf{x})\mathbf{Q}$ or in any member of $\Gamma_{\mathbf{k+1}}$ because the rule $\forall$I stipulates this. By the inductive hypothesis, $\Gamma_{\mathbf{h}} \models \mathbf{Q(a/x)}$. Because $\Gamma_{\mathbf{h}}$ is a subset of $\Gamma_{\mathbf{k+1}}$, it follows from 11.3.2 that $\Gamma_{\mathbf{k+1}} \models \mathbf{Q(a/x)}$. And because **a** does not occur in $(\forall \mathbf{x})\mathbf{Q}$ or in any member of $\Gamma_{\mathbf{k+1}}$, it follows from 11.1.8, which we repeat here, that $\Gamma_{\mathbf{k+1}} \models (\forall \mathbf{x})\mathbf{Q}$ as well:

11.1.8: Let **a** be a constant that does not occur in $(\forall \mathbf{x})\mathbf{P}$ or in any member of the set Γ. Then if $\Gamma \models \mathbf{P(a/x)}$, $\Gamma \models (\forall \mathbf{x})\mathbf{P}$.

Case 16: $\mathbf{P_{k+1}}$ is justified by Existential Elimination. Then $\mathbf{P_{k+1}}$ is derived as follows:

h	$(\exists\mathbf{x})\mathbf{Q}$	
j	| $\mathbf{Q(a/x)}$	
m	| $\mathbf{P_{k+1}}$	
k + 1	$\mathbf{P_{k+1}}$	**h, j–m** ∃E

where every member of $\Gamma_{\mathbf{h}}$ is a member of $\Gamma_{\mathbf{k+1}}$ and every member of $\Gamma_{\mathbf{m}}$ except $\mathbf{Q(a/x)}$ is a member of $\Gamma_{\mathbf{k+1}}$ (if any other assumptions in $\Gamma_{\mathbf{m}}$ were closed prior to position $\mathbf{k} + 1$, then the subderivation **j–m** would not be accessible at position $\mathbf{k} + 1$). Because every member of $\Gamma_{\mathbf{m}}$ except $\mathbf{Q(a/x)}$ is a member of $\Gamma_{\mathbf{k+1}}$, $\Gamma_{\mathbf{m}}$ is a subset of $\Gamma_{\mathbf{k+1}} \cup \{\mathbf{Q(a/x)}\}$. Moreover, **a** does not occur in $(\exists\mathbf{x})\mathbf{Q}$, $\mathbf{P_{k+1}}$, or any member of $\Gamma_{\mathbf{k+1}}$ because the rule ∃E stipulates this. By the inductive hypothesis, $\Gamma_{\mathbf{h}} \models (\exists\mathbf{x})\mathbf{Q}$, and so because $\Gamma_{\mathbf{h}}$ is a subset of $\Gamma_{\mathbf{k+1}}$, it follows from 11.3.2 that $\Gamma_{\mathbf{k+1}} \models (\exists\mathbf{x})\mathbf{Q}$. Also by the inductive hypothesis, $\Gamma_{\mathbf{m}} \models \mathbf{P_{k+1}}$, and so, because $\Gamma_{\mathbf{m}}$ is a subset of $\Gamma_{\mathbf{k+1}} \cup \{\mathbf{Q(a/x)}\}$, it follows from 11.3.2 that $\Gamma_{\mathbf{k+1}} \cup \{\mathbf{Q(a/x)}\} \models \mathbf{P_{k+1}}$. Because **a** does not occur in $(\exists\mathbf{x})\mathbf{Q}$, $\mathbf{P_{k+1}}$, or any member of $\Gamma_{\mathbf{k+1}}$, it follows that $\Gamma_{\mathbf{k+1}} \models \mathbf{P_{k+1}}$, by 11.1.9, which we repeat here:

11.1.9: Let **a** be a constant that does not occur in the sentences $(\exists\mathbf{x})\mathbf{P}$ and **Q** and that does not occur in any member of the set Γ. If $\Gamma \models (\exists\mathbf{x})\mathbf{P}$ and $\Gamma \cup \{\mathbf{P(a/x)}\} \models \mathbf{Q}$, then $\Gamma \models \mathbf{Q}$ as well.

This completes the proof of the inductive step; all of the derivation rules of *PD* are truth-preserving. Note that, in establishing that the two quantifier rules ∀I and ∃E are truth-preserving, we made essential use of the restrictions that those rules place on the instantiating constant **a**—the restrictions were included in those rules to ensure that they would be truth-preserving. Having established that the inductive step is true, we may conclude that every sentence in a derivation of *PD* is quantificationally entailed by the set of open assumptions in whose scope it lies. Therefore, we have established Metatheorem 11.3.1: if $\Gamma \vdash \mathbf{P}$ in *PD*, then $\Gamma \models \mathbf{P}$.

The proof that *PD*+ is sound for predicate logic involves the additional steps of showing that the rules of replacement of *SD*+, the three derived rules of *SD*+, and the rule Quantifier Negation are all truth-preserving. The steps in the soundness proof for *SD*+ can be converted into steps showing that the rules are truth-preserving for quantificational logic, by talking of interpretations and variable assignments rather than truth-value assignments. We leave the proof that Quantifier Negation is truth-preserving as an exercise.

Finally, we can prove that *PDE* is sound for predicate logic with identity and functions by extending the inductive step of the proof for *PD* to cover Identity Introduction and Identity Elimination and by making one change in

the basis clause of the soundness proof. We note that, since we have shown in Section 11.2 that all of the semantic results in Section 11.1 can be extended to predicate logic with identity and functions, a soundness proof for *PDE* can refer to all of those results. In particular, even though the rules ∀E and ∃I have been changed for *PDE*, the proof for Cases 13 and 14 in the inductive step of the soundness proof for *PD* will remain the same except that in place of the substitution instance $\mathbf{Q}(\mathbf{a}/\mathbf{x})$ we now have a substitution instance $\mathbf{Q}(\mathbf{t}/\mathbf{x})$, where **t** is any closed term.

In the basis clause for *PD*, we said that the first sentence in a derivation is an assumption. This is not always the case in *PDE*; the first sentence *can* be an assumption, but it can also be a sentence of the form $(\forall\mathbf{x})\mathbf{x} = \mathbf{x}$, introduced by Identity Introduction. So the proof of the basis clause will look like this:

> The first sentence in a derivation in *PDE* is either an assumption or a sentence introduced by Identity Introduction. If the first sentence is an assumption, then it lies in its own scope. In this case Γ_1 is just $\{\mathbf{P}_1\}$, and it is trivial that $\{\mathbf{P}_1\} \vDash \mathbf{P}_1$.
>
> If the first sentence is introduced by Identity Introduction, then Γ_1 is empty—there are no assumptions, and hence no open assumptions, at that point. And ∅ truth-functionally entails every sentence of the form $(\forall\mathbf{x})\mathbf{x} = \mathbf{x}$, because every such sentence is quantificationally true. This was proved in Exercise 8.7.10a.

We add the following two cases to the proof of the inductive step for *PD*:

> **Case 17:** $\mathbf{P}_{\mathbf{k}+1}$ is introduced by Identity Introduction. Then $\mathbf{P}_{\mathbf{k}+1}$ is a sentence of the form $(\forall\mathbf{x})\mathbf{x} = \mathbf{x}$ derived as follows:
>
> $$\begin{array}{rl|ll} \mathbf{k}+1 & & (\forall\mathbf{x})\mathbf{x} = \mathbf{x} & =\text{I} \end{array}$$
>
> Because ∅ quantificationally entails every sentence of the form $(\forall\mathbf{x})\mathbf{x} = \mathbf{x}$ and ∅ is a subset of $\Gamma_{\mathbf{k}+1}$, it follows by 11.3.2 that $\Gamma_{\mathbf{k}+1} \vDash (\forall\mathbf{x})\mathbf{x} = \mathbf{x}$.
>
> **Case 18:** $\mathbf{P}_{\mathbf{k}+1}$ is introduced by Identity Elimination. Then $\mathbf{P}_{\mathbf{k}+1}$ is derived as follows:
>
> $$\begin{array}{rl|ll} \mathbf{h} & & \mathbf{t}_1 = \mathbf{t}_2 & \\ \mathbf{j} & & \mathbf{P} & \\ \mathbf{k}+1 & & \mathbf{P}(\mathbf{t}_1//\mathbf{t}_2) & \mathbf{h}, \mathbf{j}\ =\text{E} \end{array} \quad \text{or} \quad \begin{array}{rl|ll} \mathbf{h} & & \mathbf{t}_1 = \mathbf{t}_2 & \\ \mathbf{j} & & \mathbf{P} & \\ \mathbf{k}+1 & & \mathbf{P}(\mathbf{t}_2//\mathbf{t}_1) & \mathbf{h}, \mathbf{j}\ =\text{E} \end{array}$$
>
> where both $\Gamma_{\mathbf{h}}$ and $\Gamma_{\mathbf{j}}$ are subsets of $\Gamma_{\mathbf{k}+1}$ (because the sentences at positions **h** and **j** are accessible at position **k** + 1). By the inductive hypothesis, $\Gamma_{\mathbf{h}} \vDash \mathbf{t}_1 = \mathbf{t}_2$ and $\Gamma_{\mathbf{j}} \vDash \mathbf{P}$. Because these are both subsets of $\Gamma_{\mathbf{k}+1}$, it follows by 11.3.2 that $\Gamma_{\mathbf{k}+1} \vDash \mathbf{t}_1 = \mathbf{t}_2$ and $\Gamma_{\mathbf{k}+1} \vDash \mathbf{P}$. It follows from 11.2.4, which we repeat here:

11.2.4: For any closed terms $\mathbf{t}_1$ and $\mathbf{t}_2$, if $\mathbf{P}$ is a sentence that contains $\mathbf{t}_1$, then $\{\mathbf{t}_1 = \mathbf{t}_2, \mathbf{P}\} \vDash \mathbf{P}(\mathbf{t}_2//\mathbf{t}_1)$, and if $\mathbf{P}$ is a sentence that contains $\mathbf{t}_2$, then $\{\mathbf{t}_1 = \mathbf{t}_2, \mathbf{P}\} \vDash \mathbf{P}(\mathbf{t}_1//\mathbf{t}_2)$.

that $\Gamma_{\mathbf{k}+1} \vDash \mathbf{P}_{\mathbf{k}+1}$ as well.

These changes establish that *PDE* is sound for predicate logic with identity and functions.

11.3**E** EXERCISES

1. Using Metatheorem 11.3.1, prove the following:

a. Every argument of *PL* that is valid in *PD* is quantificationally valid.

b. Every sentence of *PL* that is a theorem in *PD* is quantificationally true.

*c. Every pair of sentences **P** and **Q** of *PL* that are equivalent in *PD* are quantificationally equivalent.

2. Prove the following (to be used in Exercise 3) by mathematical induction:

11.3.4. Let **P** be a formula of *PL* and **Q** a subformula of **P**. Let $[\mathbf{P}](\mathbf{Q}_1//\mathbf{Q})$ be a sentence that is the result of replacing one or more occurrences of **Q** in **P** with a formula $\mathbf{Q}_1$. If **Q** and $\mathbf{Q}_1$ contain the same nonlogical symbols and variables, and if **Q** and $\mathbf{Q}_1$ are satisfied by exactly the same variable assignments on any interpretation, then the same is true of **P** and $[\mathbf{P}](\mathbf{Q}_1//\mathbf{Q})$.

3. Using 11.3.4, show how we can establish, as a step in an inductive proof of the soundness of *PD*+, that Quantifier Negation is truth-preserving for predicate logic.

***4.**a. Suppose that we changed the rule ∀I by eliminating the restriction that the instantiating constant **a** in the sentence $\mathbf{P}(\mathbf{a}/\mathbf{x})$ to which ∀I applies must not occur in any open assumption. Explain why *PD* would *not* be sound for predicate logic in this case.

b. Suppose that we changed the rule ∃E by eliminating the restriction that the instantiating constant **a** in the assumption $\mathbf{P}(\mathbf{a}/\mathbf{x})$ must not occur in the sentence **Q** that is derived. Explain why *PD* would *not* be sound for predicate logic in this case.

11.4 THE COMPLETENESS OF *PD*, *PD*+, AND *PDE*

In this section we shall prove that our natural deduction systems are **complete** for predicate logic. A natural deduction system is complete for predicate logic if, whenever a sentence is quantificationally entailed by a set of sentences, there is at least one derivation of the sentence from members of that set in the natural deduction system. Metatheorem 11.4.1 is the *Completeness Metatheorem* for *PD*:

Metatheorem 11.4.1: For any set Γ of sentences of *PL* and any sentence **P** of *PL*, if $\Gamma \vDash \mathbf{P}$, then $\Gamma \vdash \mathbf{P}$ in *PD*.

Our proof will be analogous to the proof of the completeness of *SD* for sentential logic in Chapter 6. As in Chapter 6, the Completeness Metatheorem for predicate logic follows almost immediately from the following result:

> **11.4.2:** For any set of sentences of *PL*, if Γ is consistent in *PD* then Γ is quantificationally consistent.

To see how Metatheorem 11.4.1 follows, assume that, for some set Γ and sentence **P**, $\Gamma \models \mathbf{P}$ (this is the antecedent of the metatheorem). Then the set $\Gamma \cup \{\sim \mathbf{P}\}$ is quantificationally inconsistent (see Exercise 11.4.1). It follows, from Lemma 11.4.2, that $\Gamma \cup \{\sim \mathbf{P}\}$ is also inconsistent in *PD*. And from this it follows that $\Gamma \models \mathbf{P}$ in *PD* (see Exercise 11.4.2).

So the bulk of this section is devoted to proving result 11.4.2.

The major steps in the proof of the analogous result in Chapter 6 involved showing that every set of sentences that is consistent in *SD* is a subset of a set of sentences that is maximally consistent in *SD* and showing that every set of sentences that is maximally consistent in *SD* is also truth-functionally consistent. In addition to **maximal consistency in *PD***, which is defined as

> A set Γ of sentences of *PL* is *maximally consistent in PD* if and only if Γ is consistent in *PD* and for every sentence **P** of *PD* that is not a member of Γ, $\Gamma \cup \{\mathbf{P}\}$ is inconsistent in *PD*,

we shall rely on an additional property that sets of sentences of *PL* can have, the property of **$\exists$-completeness** (read as *existential completeness*):

> A set Γ of sentences of *PL* is *$\exists$-complete* if and only if, for each sentence in Γ that has the form $(\exists\mathbf{x})\mathbf{P}$, at least one substitution instance of $(\exists\mathbf{x})\mathbf{P}$ is also a member of Γ.

We will be showing that every set of sentences of *PL* that is both maximally consistent in *PD* and $\exists$-complete is also quantificationally consistent. However, we will not show that every set of sentences that is consistent in *PD* is a subset of a set that is both maximally consistent in *PD* and $\exists$-complete, for there is a complication. To build a set that has these properties, we need to add the substitution instances required by the property of $\exists$-completeness, and for this we need to be sure that infinitely many individual constants that do not already occur in the set we are working with are available for the substitution instances. We shall ensure that infinitely many such constants are available by proving 11.4.2 through the following steps:

> *Step 1 in proof of 11.4.2:* We shall prove in result 11.4.3 that, for any set Γ that is consistent in *PD*, if we double the subscript of every individual constant in Γ (so that every resulting subscript will be even), then the resulting set $\Gamma_{\mathbf{e}}$ is also consistent in *PD*. We call such a set an '**evenly subscripted set**'.

Step 2 in proof of 11.4.2: We shall then show that, because there are infinitely many individual constants (namely, all the oddly subscripted constants) that do not occur in the sentences of any evenly subscripted set, every evenly subscripted set Γ that is consistent in *PD* is a subset of a set that is *maximally consistent in PD* and that is $\exists$-*complete*. This will be established as result 11.4.4.

Step 3 in proof of 11.4.2: We shall next show that there is a straightforward way to construct a model for every set that is maximally consistent in *PD* and that is $\exists$-complete, from which it follows that every such set is quantificationally consistent. This will be established as result 11.4.7. It follows from this result that the evenly subscripted set from which we built the maximally consistent set in Step 2 must be quantificationally consistent.

Step 4 in proof of 11.4.2: Finally we shall show, in result 11.4.8, that the set Γ that we began with must be quantificationally consistent as well.

We begin with 11.4.3, which establishes *Step 1*:

11.4.3: Let Γ be a set of sentences of *PL* and let $\Gamma_{\mathbf{e}}$ be the set that results from doubling the subscript of every individual constant that occurs in any member of Γ. Then if Γ is consistent in *PD*, $\Gamma_{\mathbf{e}}$ is also consistent in *PD*.

Proof: Assume that Γ is consistent in *PD* and that, contrary to what we wish to prove, $\Gamma_{\mathbf{e}}$ is *in*consistent in *PD*. Then there is a derivation of the sort

1	$\mathbf{P}_1$
2	$\mathbf{P}_2$
.	.
n	$\mathbf{P_n}$
.	.
k	**Q**
.	.
p	~ **Q**

where $\mathbf{P}_1, \mathbf{P}_2, \ldots, \mathbf{P_n}$ are members of $\Gamma_{\mathbf{e}}$. We shall convert this derivation into a derivation that shows that Γ is inconsistent in *PD*, contradicting our first assumption. Our strategy, not surprisingly, will be to halve the subscript of every evenly subscripted individual constant occurring in the derivation, thus converting each of $\mathbf{P}_1, \mathbf{P}_2, \ldots, \mathbf{P_n}$ back to a member of the original set Γ. There is a complication, though—in so doing we may end up with a sequence in which either an $\exists$E restriction or an a $\forall$I

restriction is violated. For example, let us suppose that Γ is $\{(\exists x)Px, (\forall x)(Px \supset Qa_1), \sim Qa_1\}$. This set is clearly inconsistent. Γ_e is the set $\{(\exists x)Px, (\forall x)(Px \supset Qa_2), \sim Qa_2\}$. Now suppose that we show that Γ_e is inconsistent in *PD* by producing the following derivation:

1	$(\exists x)Px$	Assumption
2	$(\forall x)(Px \supset Qa_2)$	Assumption
3	$\sim Qa_2$	Assumption
4	$\quad Pa_1$	A / $\exists$E
5	$\quad Pa_1 \supset Qa_2$	2 $\forall$E
6	$\quad Qa_2$	4, 5 $\supset$E
7	Qa_2	1, 4–6 $\exists$E
8	$\sim Qa_2$	3 R

Halving each of the even subscripts in this derivation produces the following, which violates two of the $\exists$E restrictions because 'a_1' now occurs in the primary (thus undischarged) assumption on line 3 and also in the sentence that is justified by E on line 7:

1	$(\exists x)Px$	Assumption	
2	$(\forall x)(Px \supset Qa_1)$	Assumption	
3	$\sim Qa_1$	Assumption	
4	$\quad Pa_1$	A / $\exists$E	
5	$\quad Pa_1 \supset Qa_1$	2 $\forall$E	
6	$\quad Qa_1$	4, 5 $\supset$E	
7	Qa_1	1, 4–6 $\exists$E	**MISTAKE!**
8	$\sim Qa_1$	3 R	

We therefore first take a precaution to ensure that this will not happen.

Let $\mathbf{a}_1, \ldots, \mathbf{a}_\mathbf{m}$ be the distinct constants that are used as instantiating constants for $\exists$E and $\forall$I in the derivation of $\mathbf{Q}$ and $\sim \mathbf{Q}$ from $\Gamma_e = \{P_1, \ldots, P_\mathbf{n}\}$, and let $\mathbf{b}_1, \ldots, \mathbf{b}_\mathbf{m}$ be distinct constants that have odd subscripts that are larger than the subscript of any constant occurring in the derivation. (Because every derivation is a finite sequence, we know that among the constants occurring in our derivation, there is one that has the largest subscript—and, whatever this largest subscript may be, there are infinitely many odd numbers that are larger.) We replace each sentence $\mathbf{R}$ in the original derivation with a sentence $\mathbf{R}^*$ that is the result of first replacing each occurrence of $\mathbf{a_i}$ in $\mathbf{R}$, $1 \leq \mathbf{i} \leq \mathbf{m}$, with $\mathbf{b_i}$, and *then* halving every even subscript in a constant in the resulting sentence.

To continue with our example, we first replace the instantiating constant 'a_1' in the previous (good) derivation with 'a_3':

1	$(\exists x)Px$	Assumption
2	$(\forall x)(Px \supset Qa_2)$	Assumption
3	$\sim Qa_2$	Assumption
4	$\quad Pa_3$	A / $\exists$E
5	$\quad Pa_3 \supset Qa_2$	2 $\forall$E
7	$\quad Qa_2$	4, 5 $\supset$E
8	Qa_2	1, 4–5 $\exists$E
9	$\sim Qa_2$	3 R

Then we halve the even subscripts:

1	$(\exists x)Px$	Assumption
2	$(\forall x)(Px \supset Qa_1)$	Assumption
3	$\sim Qa_1$	Assumption
4	$\quad Pa_3$	A / $\exists$E
5	$\quad Pa_3 \supset Qa_1$	2 $\forall$E
7	$\quad Qa_1$	4, 5 $\supset$E
8	Qa_1	1, 4–5 $\exists$E
9	$\sim Qa_1$	3 R

Here we have eliminated the violation of the $\exists$E restriction.

We claim that the resulting sequence will always be a legitimate derivation in *PD* of $\mathbf{Q}^*$ and $\sim \mathbf{Q}^*$ from members of the set Γ. First note that every new primary assumption $\mathbf{P_i}^*$ is identical to the primary assumption $\mathbf{P_i}$ in the dervation showing that Γ_e is inconsistent in *PD*. This is because none of $\mathbf{a}_1, \ldots, \mathbf{a_m}$ can occur in a primary assumption of that derivation (lest an instantiating constant restriction be violated). So $\mathbf{P}^*$ is just $\mathbf{P_i}$ with all its individual constant subscripts halved—that is, a member of the original set Γ from which $\Gamma_\mathbf{e}$ was constructed. It remains to be shown that the resulting sequence counts as a derivation in *PD*—that every sentence following the primary assumptions in that sequence can be justified. This is left as an exercise.

Step 2 in our proof of Lemma 11.4.2 is to establish the result

11.4.4: If Γ is an evenly subscripted set of sentences that is consistent in *PD*, then Γ is a subset of at least one set of sentences that is both maximally consistent in *PD* and $\exists$-complete.

We shall establish 11.4.4 by showing how, beginning with Γ, to construct a superset that has the two properties. We assume that the sentences of *PL* have been enumerated, that is, that they have been placed in a one-to-one correspondence with the positive integers so that there is a first sentence, a second sentence, a third sentence, and so on. The enumeration can be done analogously to the enumeration of the sentences of *SL* in Section 6.4; we leave proof of this as an exercise (Exercise 11.4.4). We shall now build a sequence of sets $\Gamma_1, \Gamma_2, \Gamma_3, \ldots$ where Γ_1 is an evenly subscripted set Γ that is consistent in *PD*, by considering each sentence in the enumeration, adding the sentence if it can consistently be added, and, if the added sentence is existentially quantified, adding one of its substitution instances as well. The sequence is constructed as follows:

1. Γ_1 is Γ.
2. $\Gamma_{\mathbf{i}+1}$ is
   - (i) $\Gamma_{\mathbf{i}} \cup \{\mathbf{P_i}\}$, if $\Gamma_{\mathbf{i}} \cup \{\mathbf{P_i}\}$ is consistent in *PD* and $\mathbf{P_i}$ does not have the form $(\exists \mathbf{x})\mathbf{P}$, or
   - (ii) $\Gamma_{\mathbf{i}} \cup \{\mathbf{P_i}, \mathbf{P_i}^*\}$, if $\Gamma \cup \{\mathbf{P_i}\}$ is consistent in *PD* and $\mathbf{P_i}$ has the form $(\exists \mathbf{x})\mathbf{Q}$, where $\mathbf{P_i}^*$ is a substitution instance $\mathbf{Q}(\mathbf{a}/\mathbf{x})$ of $(\exists \mathbf{x})\mathbf{Q}$ and $\mathbf{a}$ is the alphabetically earliest constant[1] that does not occur in $\mathbf{P_i}$ or in any sentence in $\Gamma_{\mathbf{i}}$, or
   - (iii) $\Gamma_{\mathbf{i}}$, if $\Gamma_{\mathbf{i}} \cup \{\mathbf{P_i}\}$ is inconsistent in *PD*.

As an example of (ii), if $\Gamma_{\mathbf{i}}$ is the set $\{(\forall x)(Fxa \supset Gx), \sim Hc \vee (\exists y)Jyy\}$ and $\mathbf{P_i}$ is '$(\exists z)(Kz \,\&\, (\forall y)Fzy)$', then $\Gamma_{\mathbf{i}} \cup \{\mathbf{P_i}\}$ is quantificationally consistent, and so $\mathbf{P_i}$ will be added to the set—but we must add a substitution instance of $\mathbf{P_i}$ as well. The alphabetically earliest constant that does not occur in $\mathbf{P_i}$ or in any member of $\Gamma_{\mathbf{i}}$ is 'b', and so this will be the instantiating constant. $\Gamma_{\mathbf{i}+1}$ is therefore

$$\{(\forall x)(Fxa \supset Gx), \sim Hc \vee (\exists y)Jyy, (\exists z)(Kz \,\&\, (\forall y)Fzy), Kb \,\&\, (\forall y)Fby\}$$

The reason for using an instantiating constant that does not already occur in $\Gamma_{\mathbf{i}}$ will become clear shortly when we prove that each set in the sequence is consistent in *PD*. Here it is important to note that we can always meet the requirement in condition (ii)—that $\mathbf{a}$ be a *new* constant—because for any set in the sequence, there is always at least one individual constant that does not already occur in that set. This is because the set that we started with is evenly subscripted, and so we know that infinitely many oddly subscripted individual constants do not occur in Γ.

Because the sequence $\Gamma_1, \Gamma_2, \Gamma_3, \ldots$ is infinitely long, there is no last member in the set. We want a set that contains all the sentences in these sets, so we let Γ^* be the set that contains every sentence that occurs in some set in the infinite sequence $\Gamma_1, \Gamma_2, \Gamma_3, \ldots$ We shall show that Γ^* is both maximally consistent in *PD* and $\exists$-complete. To show that Γ^* is maximally consistent in

[1]This is an extended sense of 'alphabetical order' in which the nonsubscripted constants occur first, in ordinary alphabetical order, followed by the constants subscripted with '1'—in ordinary alphabetical order—and so on.

PD, we first prove that each set $\Gamma_{\mathbf{i}}$ in the sequence is consistent in *PD*, using mathematical induction.

> *Basis clause:* Γ_1 is consistent in *PD*.
>
> **Proof of basis clause:** By definition, Γ_1 is Γ, a set that is consistent in *PD*.
>
> *Inductive step:* If for every $\mathbf{i} \leq \mathbf{k}$, $\Gamma_{\mathbf{i}}$ is consistent in *PD*, then $\Gamma_{\mathbf{k}+1}$ is consistent in *PD*.
>
> **Proof of inductive step:** If $\Gamma_{\mathbf{k}+1}$ is formed in accordance with condition (i), then $\Gamma_{\mathbf{k}+1}$ is obviously consistent in *PD*. If $\Gamma_{\mathbf{k}+1}$ is formed in accordance with condition (ii), then we need to show that it follows that $\Gamma_{\mathbf{i}} \cup \{\mathbf{P_i}, \mathbf{P_i}^*\}$, which is what $\Gamma_{\mathbf{k}+1}$ was defined to be in this case, is also consistent in *PD*. Because the instantiating constant in $\mathbf{P_i}^*$ does not occur in any member of $\Gamma_{\mathbf{i}} \cup \{\mathbf{P_i}\}$, the consistency of $\Gamma_{\mathbf{k}+1}$ follows immediately from result 11.1.10, which we repeat here:
>
> **11.1.10:** If **a** does not occur in any member of the set $\Gamma \cup \{(\exists \mathbf{x})\mathbf{P}\}$ and if the set is quantificationally consistent, then the set $\Gamma \cup \{(\exists \mathbf{x})\mathbf{P}, \mathbf{P}(\mathbf{a}/\mathbf{x})\}$ is also quantificationally consistent.
>
> Finally, if $\Gamma_{\mathbf{k}+1}$ is formed in accordance with condition (iii), then $\Gamma_{\mathbf{k}+1}$ is $\Gamma_{\mathbf{k}}$, and by the inductive hypothesis, $\Gamma_{\mathbf{k}}$ is consistent in *PD*. So, no matter which condition was applied in its construction, $\Gamma_{\mathbf{k}+1}$ is consistent in *PD*.

We conclude that every set in the sequence Γ_1, Γ_2, Γ_3, . . . is consistent in *PD*.

We now need to show that the set Γ^*, which contains all the sentences that occur in any set in the sequence, is itself consistent in *PD*. We shall show this by assuming that it is not consistent in *PD* and deriving a contradiction. So assume that Γ^* is not consistent in *PD*. Then there is a finite nonempty subset Γ' of Γ^* that is inconsistent in *PD* (the proof is analogous to that in the proof of 6.4.6). Because Γ' is finite, some sentence in Γ', say, $\mathbf{P_j}$, occurs later in our enumeration of the sentences of *PL* than any other sentence in Γ'. Every member of Γ' is thus a member of $\Gamma_{\mathbf{j}+1}$, by the way we constructed the sets in the sequence. (If the **i**th sentence is added to one of the sets, it is added by the time that $\Gamma_{\mathbf{i}+1}$ is constructed.) It follows that $\Gamma_{\mathbf{j}+1}$ is also inconsistent in *PD* (the proof is analogous to the proof of 6.4.7). But we have just proved that every set in the sequence is consistent in *PD*, so we conclude that, contrary to our assumption, Γ^* is also consistent in *PD*.

That Γ^* is *maximally* consistent in *PD* is proved in exactly the manner that the parallel result in Section 6.4 was proved—for any sentence $\mathbf{P_k}$, if $\Gamma^* \cup \{\mathbf{P_k}\}$ is consistent in *PD*, then the subset $\Gamma_{\mathbf{k}}$ of Γ^* is such that $\Gamma_{\mathbf{k}} \cup \{\mathbf{P_k}\}$ is consistent in *PD*, and so by the construction of the sequence of sets, $\mathbf{P_k}$ is a member of $\Gamma_{\mathbf{k}+1}$ and hence of Γ^*.

Finally, the proof that Γ^* is $\exists$-complete is left as an exercise. This completes the proof of result 11.4.4—every evenly subscripted set Γ of sentences of *PL* that is consistent in *PD* is a subset of at least one set of sentences that is both maximally consistent in *PD* and $\exists$-complete.

We now turn to *Step 3* in our proof of result 11.4.2. We must prove that every set Γ that is both maximally consistent in *PD* and $\exists$-complete is consistent in *PD*. To do this we shall use the following preliminary results:

11.4.5: If $\Gamma \vdash \mathbf{P}$ and Γ is a subset of a set Γ^* that is maximally consistent in *PD*, then $\mathbf{P} \in \Gamma^*$.

Proof: See Exercise 11.4.9.

11.4.6: Every set Γ^* of sentences that is both maximally consistent in *PD* and $\exists$-complete has the following properties:

a. $\mathbf{P} \in \Gamma^*$ if and only if $\sim \mathbf{P} \notin \Gamma^*$.
b. $\mathbf{P} \,\&\, \mathbf{Q} \in \Gamma^*$ if and only if $\mathbf{P} \in \Gamma^*$ and $\mathbf{Q} \in \Gamma^*$.
c. $\mathbf{P} \vee \mathbf{Q} \in \Gamma^*$ if and only if either $\mathbf{P} \in \Gamma^*$ or $\mathbf{Q} \in \Gamma^*$.
d. $\mathbf{P} \supset \mathbf{Q} \in \Gamma^*$ if and only if either $\mathbf{P} \notin \Gamma^*$ or $\mathbf{Q} \in \Gamma^*$.
e. $\mathbf{P} \equiv \mathbf{Q} \in \Gamma^*$ if and only if either $\mathbf{P} \in \Gamma^*$ and $\mathbf{Q} \in \Gamma^*$ or $\mathbf{P} \notin \Gamma^*$ and $\mathbf{Q} \notin \Gamma^*$.
f. $(\forall \mathbf{x})\mathbf{P} \in \Gamma^*$ if and only if, for every individual constant $\mathbf{a}$, $\mathbf{P}(\mathbf{a}/\mathbf{x}) \in \Gamma^*$.
g. $(\exists \mathbf{x})\mathbf{P} \in \Gamma^*$ if and only if, for at least one individual constant $\mathbf{a}$, $\mathbf{P}(\mathbf{a}/\mathbf{x}) \in \Gamma^*$.

Proof: The proofs that (a)–(e) hold for sets of sentences that are maximally consistent in *PD* and $\exists$-complete parallel exactly the corresponding proofs in Section 6.4, using result 11.4.5 instead of 6.4.5.

Proof of (f): Assume that $(\forall \mathbf{x})\mathbf{P} \in \Gamma^*$. For any substitution instance $\mathbf{P}(\mathbf{a}/\mathbf{x})$ of $(\forall \mathbf{x})\mathbf{P}$, $\{(\forall \mathbf{x})\mathbf{P}\} \vdash \mathbf{P}(\mathbf{a}/\mathbf{x})$ (by $\forall$E); so, by 11.4.5, every substitution instance is a member of Γ^* as well. Now assume that $(\forall \mathbf{x})\mathbf{P} \notin \Gamma^*$. Then $\sim (\forall \mathbf{x})\mathbf{P} \in \Gamma^*$, by (a). The following derivation shows that $\{\sim (\forall \mathbf{x})\mathbf{P}\} \vdash (\exists \mathbf{x}) \sim \mathbf{P}$:

1	$\sim (\forall \mathbf{x})\mathbf{P}$	Assumption
2	$\quad \sim (\exists \mathbf{x}) \sim \mathbf{P}$	A /~ E
3	$\quad\quad \sim \mathbf{P}(\mathbf{a}/\mathbf{x})$	A /~ E
4	$\quad\quad (\exists \mathbf{x}) \sim \mathbf{P}$	3 $\exists$I
5	$\quad\quad \sim (\exists \mathbf{x}) \sim \mathbf{P}$	2 R
6	$\quad \mathbf{P}(\mathbf{a}/\mathbf{x})$	3–5 ~ E
7	$\quad (\forall \mathbf{x})\mathbf{P}$	6 $\forall$I
8	$\quad \sim (\forall \mathbf{x})\mathbf{P}$	1 R
9	$(\exists \mathbf{x}) \sim \mathbf{P}$	2–8 ~ E

(We assume that the constant **a** does not occur in **P**.) Therefore, by 11.4.5, $(\exists \mathbf{x}) \sim \mathbf{P}$ is also a member of Γ*. Because Γ* is ∃-complete, some substitution instance $\sim \mathbf{P}(\mathbf{a}/\mathbf{x})$ of $(\exists \mathbf{x}) \sim \mathbf{P}$ is a member of Γ* as well, and it therefore follows from (a) that $\mathbf{P}(\mathbf{a}/\mathbf{x}) \notin \Gamma^*$. So, if $(\forall \mathbf{x})\mathbf{P} \notin \Gamma^*$, then there is at least one substitution instance of $(\forall \mathbf{x})\mathbf{P}$ that is not a member of Γ*.

Proof of (g): Assume that $(\exists \mathbf{x})\mathbf{P} \in \Gamma^*$. Then, because Γ* is ∃-complete, at least one substitution instance of $(\exists \mathbf{x})\mathbf{P}$ is also a member of Γ*. Now assume that $(\exists \mathbf{x})\mathbf{P} \notin \Gamma^*$. If some substitution instance $\mathbf{P}(\mathbf{a}/\mathbf{x})$ of $(\exists \mathbf{x})\mathbf{P}$ is a member of Γ*, then because $\{\mathbf{P}(\mathbf{a}/\mathbf{x})\} \vdash (\exists \mathbf{x})\mathbf{P}$ (by ∃I), it follows from 11.4.5 that, contrary to our assumption, $(\exists \mathbf{x})\mathbf{P}$ is also a member of Γ*. So, if $(\exists \mathbf{x})\mathbf{P} \notin \Gamma^*$, then none of its substitution instances is a member of Γ*.

We can now complete the proof of *Step 3* by establishing the result

11.4.7: Every set of sentences of *PL* that is both maximally consistent in *PD* and ∃-complete is quantificationally consistent.

We shall prove this by showing how to construct a model for any set Γ* that is both maximally consistent in *PD* and ∃-complete, that is, an interpretation **I*** on which every member of Γ* is true. We begin by associating with each individual constant a distinct positive integer—the positive integer **i** will be associated with the alphabetically **i**th constant. The number 1 will be associated with 'a', 2 with 'b', . . . , 22 with 'v', 23 with 'a_1', and so on. **I*** is then defined as follows:

1. The UD is the set of positive integers.
2. For each sentence letter **P**, $\mathbf{I}^*(\mathbf{P}) = \mathbf{T}$ if and only if $\mathbf{P} \in \Gamma^*$.
3. For each individual constant **a**, $\mathbf{I}^*(\mathbf{a})$ is the positive integer associated with **a**.
4. For each **n**-place predicate **A**, $\mathbf{I}^*(\mathbf{A})$ includes all and only those **n**-tuples $\langle \mathbf{I}^*(\mathbf{a}_1), \ldots, \mathbf{I}^*(\mathbf{a_n}) \rangle$ such that $\mathbf{A}\mathbf{a}_1 \ldots \mathbf{a_n} \in \Gamma^*$.

The major feature of this interpretation is that, for each atomic sentence **P** of *PL*, **P** will be true on **I*** if and only if $\mathbf{P} \in \Gamma^*$. That is why we defined condition 4 (as well as condition 2) as we did. And to be sure that condition 4 can be met, we must have condition 3, which ensures that each individual constant designates a *different* member of the UD. This is necessary because, for example, if 'Fa' is a member of Γ* and 'Fb' is not a member, then if 'a' and 'b' designated the same integer—say, 1—condition 4 would require that the 1-tuple ⟨1⟩ both be and not be a member of $\mathbf{I}^*(\mathrm{F})$. (In addition, condition 3 ensures that every member of the UD is named by a constant, which we shall shortly see is also important when we look at the truth-values that quantified sentences receive on **I***.)

We complete the proof of 11.4.7 by establishing, by mathematical induction on the number of occurrences of logical operators in sentences of *PL*, that each sentence **P** of *PL* is true on **I*** if and only if $\mathbf{P} \in \Gamma^*$.

Basis clause: Each sentence **P** that contains zero occurrences of logical operators is true on **I*** if and only if $\mathbf{P} \in \Gamma^*$.

Proof of basis clause: Either **P** is a sentence letter or **P** has the form $\mathbf{Aa_1 \ldots a_n}$. If **P** is a sentence letter, then, by part 2 of the definition of **I***, it follows that **P** is true on **I*** if and only if $\mathbf{P} \in \Gamma^*$.

If **P** has the form $\mathbf{Aa_1 \ldots a_n}$, then **P** is true on **I*** if and only if $\langle \mathbf{I}^*(\mathbf{a_1}), \ldots, \mathbf{I}^*(\mathbf{a_n}) \rangle \in \mathbf{I}^*(\mathbf{A})$. Part 4 of the definition of **I*** stipulates that $\langle \mathbf{I}^*(\mathbf{a_1}), \ldots, \mathbf{I}^*(\mathbf{a_n}) \rangle \in \mathbf{I}^*(\mathbf{A})$ if and only if $\mathbf{Aa_1 \ldots a_n} \in \Gamma^*$. So in this case as well, **P** is true on **I*** if and only if $\mathbf{P} \in \Gamma^*$.

Inductive step: If each sentence **P** with **k** or fewer occurrences of logical operators is true on **I*** if and only if $\mathbf{P} \in \Gamma^*$, then the same is true of each sentence **P** with $\mathbf{k} + 1$ occurrences of logical operators.

Proof of inductive step: We assume that, for an arbitrary positive integer **k**, the inductive hypothesis is true. We must show that on this assumption it follows that any sentence **P** that has $\mathbf{k} + 1$ occurrences of logical operators is such that **P** is true on **I*** if and only if $\mathbf{P} \in \Gamma^*$. We consider the forms that the sentence **P** may have.

Cases 1–5: P has one of the forms $\sim \mathbf{Q}$, $\mathbf{Q} \,\&\, \mathbf{R}$, $\mathbf{Q} \vee \mathbf{R}$, $\mathbf{Q} \supset \mathbf{R}$, or $\mathbf{Q} \equiv \mathbf{R}$. The proofs for these five cases are analogous to the proofs for the parallel cases for *SL* in Section 6.4, so we omit them here.

Case 6: P has the form $(\forall \mathbf{x})\mathbf{Q}$. Assume that $(\forall \mathbf{x})\mathbf{Q}$ is true on **I***. Then every substitution instance $\mathbf{Q}(\mathbf{a}/\mathbf{x})$ of $(\forall \mathbf{x})\mathbf{Q}$ is true on **I*** because, by 11.1.4, $\{(\forall \mathbf{x})\mathbf{Q}\}$ quantificationally entails every one of its substitution instances. Each substitution instance contains fewer than $\mathbf{k} + 1$ occurrences of connectives, and so, by the inductive hypothesis, each substitution instance is a member of Γ^* since it is true on **I***. It follows from part (f) of 11.4.6 that $(\forall \mathbf{x})\mathbf{Q}$ is also a member of Γ^*.

Now assume that $(\forall \mathbf{x})\mathbf{Q}$ is false on **I***. In this case we shall make use of result 11.1.11, which we repeat here:

11.1.11: Let **I** be an interpretation on which each member of the UD is assigned to at least one individual constant. Then, if every substitution instance of $(\forall \mathbf{x})\mathbf{P}$ is true on **I**, so is $(\forall \mathbf{x})\mathbf{P}$.

I* is an interpretation of the type specified in 11.1.11: Every positive integer in the UD is designated by the individual constant with which we have associated that integer. It follows, then, that if $(\forall \mathbf{x})\mathbf{Q}$ is false on **I***, at least one of its substitution instances $\mathbf{Q}(\mathbf{a}/\mathbf{x})$ must also be false on **I***. Because $\mathbf{Q}(\mathbf{a}/\mathbf{x})$ contains fewer than $\mathbf{k} + 1$ occurrences of logical

operators, it follows from the inductive hypothesis that $\mathbf{Q}(\mathbf{a}/\mathbf{x}) \notin \Gamma^*$. And so, by part (f) of 11.4.6, $(\forall\mathbf{x})\mathbf{Q} \notin \Gamma^*$.

Case 7: **P** has the form $(\exists\mathbf{x})\mathbf{Q}$. Assume that $(\exists\mathbf{x})\mathbf{Q}$ is true on $\mathbf{I}^*$. Then it follows from 11.1.12, which we repeat here, that at least one substitution instance $\mathbf{Q}(\mathbf{a}/\mathbf{x})$ of $(\exists\mathbf{x})\mathbf{Q}$ is true on $\mathbf{I}^*$:

11.1.12: Let **I** be an interpretation on which each member of the UD is assigned to at least one individual constant. Then, if every substitution instance of $(\exists\mathbf{x})\mathbf{P}$ is false on **I**, so is $(\exists\mathbf{x})\mathbf{P}$.

Because the substitution instance $\mathbf{Q}(\mathbf{a}/\mathbf{x})$ contains fewer than $\mathbf{k} + 1$ occurrences of logical operators, it follows from the inductive hypothesis that $\mathbf{Q}(\mathbf{a}/\mathbf{x}) \in \Gamma^*$. So, by part (g) of 11.4.6, $(\exists\mathbf{x})\mathbf{Q} \in \Gamma^*$.

Now assume that $(\exists\mathbf{x})\mathbf{Q}$ is false on $\mathbf{I}^*$. Because each substitution instance $\mathbf{Q}(\mathbf{a}/\mathbf{x})$ is such that $\{\mathbf{Q}(\mathbf{a}/\mathbf{x})\} \vDash (\exists\mathbf{x})\mathbf{Q}$ (this is result 11.1.5), it follows that every substitution instance $\mathbf{Q}(\mathbf{a}/\mathbf{x})$ is also false on $\mathbf{I}^*$. Each of these substitution instances contains fewer than $\mathbf{k} + 1$ occurrences of logical operators, and so it follows from the inductive hypothesis that no substitution instance of $(\exists\mathbf{x})\mathbf{Q}$ is a member of Γ^*. Finally, by part (g) of 11.4.6, it follows that $(\exists\mathbf{x})\mathbf{Q} \notin \Gamma^*$.

That completes the proof of the inductive step, and we may now conclude that each sentence **P** of *PL* is true on $\mathbf{I}^*$ if and only if $\mathbf{P} \in \Gamma^*$. So $\mathbf{I}^*$ is a model of Γ^*, and Γ^* is quantificationally consistent. Result 11.4.7 is therefore true: Every set that is both maximally consistent in *PD* and ∃-complete is quantificationally consistent. Results 11.4.4 and 11.4.7 together establish that every evenly subscripted set of sentences of *PL* that is consistent in *PD* is also quantificationally consistent.

Step 4 of the proof of result 11.4.2 is established by result 11.4.8:

11.4.8: Let Γ be a set of sentences of *PL* and let $\Gamma_\mathbf{e}$ be the set that results from doubling the subscript of every individual constant that occurs in any member of Γ. Then, if $\Gamma_\mathbf{e}$ is quantificationally consistent, Γ is quantificationally consistent as well.

Proof: See Exercise 11.4.8.

We have now completed the four steps in the proof of result 11.4.2, so we may conclude that if a set Γ of sentences of *PL* is quantificationally inconsistent, then Γ is also inconsistent in *PD*. And this establishes the completeness of *PD* for predicate logic. If $\Gamma \vDash \mathbf{P}$, then $\Gamma \cup \{\sim \mathbf{P}\}$ is quantificationally inconsistent. By 11.4.2, $\Gamma \cup \{\sim \mathbf{P}\}$ is also inconsistent in *PD*, and hence $\Gamma \vdash \mathbf{P}$ in *PD*.

Because *PD* is complete for predicate logic, so is *PD*+. Every rule of *PD* is a rule of *PD*+, and so every derivation in *PD* is a derivation in *PD*+. So, if $\Gamma \vDash \mathbf{P}$, then $\Gamma \vdash \mathbf{P}$ in *PD*+ because we know, by Metatheorem 11.4.1, that $\Gamma \vdash \mathbf{P}$ in *PD*.

We also want to be sure that *PDE* is complete for predicate logic with identity and functions. The completeness proof for *PDE* is similar to the completeness proof for *PD*, but there are some important changes. Results 11.4.3 and 11.4.8 must now take into account sentences containing the identity predicate and complex terms; the necessary changes are left as an exercise. Maximal consistency is defined for *PDE* as it was for *PD*, while the definition of ∃-completeness must be modified slightly:

> A set Γ of sentences of *PLE* is ∃-complete if and only if, for each sentence in Γ that has the form $(\exists \mathbf{x})\mathbf{P}$, at least one substitution instance of $(\exists \mathbf{x})\mathbf{P}$ in which the instantiating individual term is a constant is also a member of Γ.

The proof of result 11.4.4 for *PDE*—that every evenly subscripted set of sentences that is consistent in *PDE* is a subset of a set of sentences that is both maximally consistent in *PDE* and ∃-complete—is just like the proof for *PD* except that it speaks of *PLE* and *PDE*, rather than *PL* and *PD*. However, the proof of result 11.4.7 is different because the model **I*** that is constructed for a maximally consistent and ∃-complete set of sentences must be defined differently.

The interpretation **I*** of the maximally consistent and ∃-complete set Γ* that we constructed in the proof of 11.4.7 stipulated that a distinct positive integer be associated with each individual constant and that

> 3. For each individual constant **a**, **I***(**a**) is the positive integer associated with **a**.

This will not do in the case of *PDE*. For suppose that Γ, and consequently its superset Γ*, contains a sentence $\mathbf{a} = \mathbf{b}$ in which **a** and **b** are different constants. If we interpret the constants of *PLE* in accordance with condition 3, **a** and **b** will denote *different* members of the UD, and hence $\mathbf{a} = \mathbf{b}$ will be false. But the interpretation is supposed to make all members of Γ*, including $\mathbf{a} = \mathbf{b}$, true. So we shall have to change condition 3 to take care of the case where a sentence like $\mathbf{a} = \mathbf{b}$ is a member of the set Γ*. We shall also have to interpret the functors in the language, and to do so in a way that makes sentences containing complex terms true if and only if those sentences are members of Γ*.

Before turning to the construction of an interpretation for Γ*, however, we first establish some facts about sets of sentences that are maximally consistent in *PDE* and ∃-complete. As the reader may easily verify, the properties listed in result 11.4.6 remain true for maximally consistent, ∃-complete sets of sentences of *PDE*. We must add three additional properties to the list in result 11.4.6:

> h. For every closed term **t**, $\mathbf{t} = \mathbf{t} \in \Gamma^*$.
>
> **Proof:** Let **t** be any closed term. $\varnothing \vdash \mathbf{t} = \mathbf{t}$, by =I and ∀E. Because the empty set is a subset of Γ*, it follows from 11.4.5 that $\mathbf{t} = \mathbf{t} \in \Gamma^*$.

i. If $\mathbf{t}_1$ and $\mathbf{t}_2$ are closed terms and $\mathbf{t}_1 = \mathbf{t}_2 \in \Gamma^*$, then

a. If $\mathbf{Q}$ is a sentence in which $\mathbf{t}_1$ occurs, $\mathbf{Q} \in \Gamma^*$ if and only if every sentence $\mathbf{Q}(\mathbf{t}_2//\mathbf{t}_1)$ (every sentence obtained by replacing one or more occurrences of $\mathbf{t}_1$ in $\mathbf{Q}$ with $\mathbf{t}_2$) is a member of Γ^*.

b. If $\mathbf{Q}$ is a sentence in which $\mathbf{t}_2$ occurs, $\mathbf{Q} \in \Gamma^*$ if and only if every sentence $\mathbf{Q}(\mathbf{t}_1//\mathbf{t}_2)$ is a member of Γ^*.

Proof: Let $\mathbf{t}_1 = \mathbf{t}_2$ be a sentence that is a member of Γ^* and let $\mathbf{Q}$ be a sentence in which $\mathbf{t}_1$ occurs. Assume that $\mathbf{Q} \in \Gamma^*$. Every sentence $\mathbf{Q}(\mathbf{t}_2//\mathbf{t}_1)$ is derivable from the set $\{\mathbf{t}_1 = \mathbf{t}_2, \mathbf{Q}\}$ by =E. Therefore, by 11.4.5, every sentence $\mathbf{Q}(\mathbf{t}_2//\mathbf{t}_1)$ is a member of Γ^*. Now assume that $\mathbf{Q} \notin \Gamma^*$. Every sentence $\mathbf{Q}(\mathbf{t}_2//\mathbf{t}_1)$ is such that $\{\mathbf{t}_1 = \mathbf{t}_2, \mathbf{Q}(\mathbf{t}_2//\mathbf{t}_1)\} \vdash \mathbf{Q}$, by =E—use $\mathbf{t}_1$ to replace every occurrence of $\mathbf{t}_2$ that replaced $\mathbf{t}_1$ in $\mathbf{Q}(\mathbf{t}_2//\mathbf{t}_1)$, and the result is $\mathbf{Q}$ once again. So, if any sentence $\mathbf{Q}(\mathbf{t}_2//\mathbf{t}_1) \in \Gamma^*$, then, by 11.4.5, $\mathbf{Q} \in \Gamma^*$ as well. Therefore, if $\mathbf{Q} \notin \Gamma^*$, then no sentence $\mathbf{Q}(\mathbf{t}_2//\mathbf{t}_1)$ is a member of Γ^*.

Similar reasoning shows that if $\mathbf{t}_1 = \mathbf{t}_2 \in \Gamma^*$ and $\mathbf{Q}$ is a sentence in which $\mathbf{t}_2$ occurs, then $\mathbf{Q} \in \Gamma^*$ if and only if every sentence $\mathbf{Q}(\mathbf{t}_1//\mathbf{t}_2)$ is a member of Γ^*.

j. For each $\mathbf{n}$-place functor f and $\mathbf{n}$ terms $\mathbf{t}_1, \ldots, \mathbf{t}_\mathbf{n}$, there is at least one constant $\mathbf{b}$ such that $f(\mathbf{t}_1, \ldots, \mathbf{t}_\mathbf{n}) = \mathbf{b} \in \Gamma^*$.

Proof: By property (h), the formula $f(\mathbf{t}_1, \ldots, \mathbf{t}_\mathbf{n}) = f(\mathbf{t}_1, \ldots, \mathbf{t}_\mathbf{n}) \in \Gamma^*$. Since $f(\mathbf{t}_1, \ldots, \mathbf{t}_\mathbf{n}) = f(\mathbf{t}_1, \ldots, \mathbf{t}_\mathbf{n}) \vdash (\exists\mathbf{x})f(\mathbf{t}_1, \ldots, \mathbf{t}_\mathbf{n}) = \mathbf{x}$, the sentence $(\exists\mathbf{x})f(\mathbf{t}_1, \ldots, \mathbf{t}_\mathbf{n}) = \mathbf{x}$ must also be a member of Γ^*, by 11.4.5. And because Γ^* is $\exists$-complete, it follows from our revised definition of $\exists$-completeness that there is at least one constant $\mathbf{b}$ such that the formula $f(\mathbf{t}_1, \ldots, \mathbf{t}_\mathbf{n}) = \mathbf{b}$ is also a member of Γ^*.

We now turn to the proof of result 11.4.7 for *PDE*—that every set of sentences of *PLE* that is both maximally consistent in *PDE* and $\exists$-complete is also quantificationally consistent. Let Γ^* be a set of sentences that is both maximally consistent in *PLE* and $\exists$-complete. In preparation for constructing a model for this set, we associate positive integers with the individual constants of *PLE* as follows:

First associate the positive integer $\mathbf{i}$ with the alphabetically $\mathbf{i}$th individual constant of *PLE*. Let $\mathbf{p}(\mathbf{a})$ stand for the integer that has been associated with the constant $\mathbf{a}$. Thus $\mathbf{p}(\text{'a'})$ is 1, $\mathbf{p}(\text{'b'})$ is 2, and so on.

Now we define a second association $\mathbf{q}$: For each constant $\mathbf{a}$, $\mathbf{q}(\mathbf{a}) = \mathbf{p}(\mathbf{a}')$, where $\mathbf{a}'$ is the alphabetically earliest constant such that $\mathbf{a} = \mathbf{a}'$ is a member of Γ^*.

Note that for each constant $\mathbf{a}$, property (h) of maximally consistent, $\exists$-complete sets assures us that $\mathbf{a} = \mathbf{a} \in \Gamma^*$, and so we can be certain that $\mathbf{q}$ assigns a value to $\mathbf{a}$. According to the definition of $\mathbf{q}$, $\mathbf{q}(\text{'a'})$ is always 1 since property (h)

assures us that 'a = a' is a member of Γ*, and 'a' is the alphabetically earliest constant of *PLE*. But for every other constant, the value that **q** associates with it depends on the identity sentences that the particular set Γ* contains. Suppose that 'b = a', 'b = b', 'b = e', and 'b = m_{22}' are the only identity sentences in Γ* that contain 'b' to the left of the identity predicate. In this case there is an alphabetically earlier constant to the right, namely, 'a', and this is the alphabetically earliest constant so occurring. So **q**('b') = **p**('a') = 1. If 'c = c', 'c = f', and 'c = g_3' are the only identity sentences in Γ* that contain 'c' to the left of the identity predicate, then 'c' is the alphabetically earliest constant occurring to the right, and so **q**('c') = **p**('c') = 3. The definition of **q** will play a role in ensuring that identity sentences come out true on the interpretation that we shall construct if and only if they are members of Γ*, as a consequence of

> **11.4.9:** For any constants **a** and **b**, **q**(**a**) = **q**(**b**) if and only if **a** = **b** ∈ Γ*.

> **Proof:** As preliminaries, let **a**′ be the alphabetically earliest constant such that **a** = **a**′ ∈ Γ* (remember that property (h) guarantees that there is at least one such constant), and let **b**′ be the alphabetically earliest constant such that **b** = **b**′ ∈ Γ*. Then
>
> - **q**(**a**) = **p**(**a**′) by the way **q** is defined, and
> - **q**(**b**) = **q**(**b**′).
> - It follows that **q**(**a**) = **q**(**b**) if and only if **q**(**a**′) = **p**(**b**′)
> - Because **p** associates different values with different constants, **p**(**a**′) = **p**(**b**′) if and only if **a**′ and **b**′ are the same constant.
>
> We conclude that **q**(**a**) = **q**(**b**) if and only if **a**′ and **b**′ are the same constant.
>
> Now assume that **q**(**a**) = **q**(**b**). It follows that **a**′ and **b**′ are the same constant. Therefore, because **b** = **b**′ ∈ Γ*, it follows trivially that **b** = **a**′, which is the same sentence, is a member of Γ*. And because **a** = **a**′ ∈ Γ*, it follows from property (i) of maximally consistent, ∃-complete sets that **a** = **b** ∈ Γ* (because **a** = **b** is a sentence **a** = **a**′ (**b**//**a**′)).
>
> Now assume that **a** = **b** ∈ Γ*.
>
> - Because **a** = **a**′ ∈ Γ*, it follows from property (i) that **b** = **a**′ ∈ Γ*.
> - Because **b** = **b**′ ∈ Γ*, it follows from property (i) that **a** = **b**′ ∈ Γ*.
> - **b**′ was defined to be the alphabetically earliest constant that appears to the right of the identity predicate in an identity sentence containing **b**, and so from the fact that **b** = **a**′ ∈ Γ*, we conclude that **a**′ is not alphabetically earlier than **b**′.
> - **a**′ was defined to be the alphabetically earliest constant that appears to the right of the identity predicate in an identity

sentence containing **a**, and so from the fact that $\mathbf{a} = \mathbf{b}' \in \Gamma^*$, we conclude that $\mathbf{b}'$ is not alphabetically earlier than $\mathbf{a}'$.

- Therefore, $\mathbf{a}'$ and $\mathbf{b}'$ must be the same constant.

So, from (e), we may conclude that $\mathbf{q}(\mathbf{a}) = \mathbf{q}(\mathbf{b})$.

Result 11.4.9 guarantees that if there is an identity sentence in Γ^* that contains the individual constants **a** and **b** then $\mathbf{q}(\mathbf{a}) = \mathbf{q}(\mathbf{b})$, and if there is no identity sentence in Γ^* that contains **a** and **b** then $\mathbf{q}(\mathbf{a}) \neq \mathbf{q}(\mathbf{b})$. And this fact will be crucial in our construction of an interpretation on which every member of a set that is both maximally consistent in *PDE* and ∃-complete is true. We turn now to the construction.

Let Γ^* be a set that is both maximally consistent in *PDE* and ∃-complete, and define the interpretation $\mathbf{I}^*$ as follows:

1. The UD is the set of positive integers that **q** associates with at least one individual constant of *PLE*.
2. For each sentence letter **P**, $\mathbf{I}^*(\mathbf{P}) = \mathbf{T}$ if and only if $\mathbf{P} \in \Gamma^*$.
3. For each individual constant **a**, $\mathbf{I}^*(\mathbf{a}) = \mathbf{q}(\mathbf{a})$.
4. For each **n**-place functor f, $\mathbf{I}^*(f)$ is the set that includes all and only those $\mathbf{n} + 1$-tuples $<\mathbf{I}^*(\mathbf{a}_1), \ldots, \mathbf{I}^*(\mathbf{a}_\mathbf{n}), \mathbf{I}^*(\mathbf{b})>$, where $\mathbf{a}_1, \ldots, \mathbf{a}_\mathbf{n}$ and **b** are individual constants, such that $f(\mathbf{a}_1, \ldots, \mathbf{a}_\mathbf{n}) = \mathbf{b} \in \Gamma^*$.
5. For each **n**-place predicate **A** other than the identity predicate, $\mathbf{I}^*(\mathbf{A})$ is the set that includes all and only those **n**-tuples $\langle \mathbf{I}^*(\mathbf{a}_1), \ldots, \mathbf{I}^*(\mathbf{a}_\mathbf{n}) \rangle$ such that $\mathbf{A}\mathbf{a}_1 \ldots \mathbf{a}_\mathbf{n} \in \Gamma^*$.

We must ensure that conditions 4 and 5 can be met.

For condition 4 we must ensure that the interpretation of f is indeed a function on the UD: that for each **n** members $\mathbf{u}_1, \ldots, \mathbf{u}_\mathbf{n}$ of the UD there is exactly one member $\mathbf{u}_{\mathbf{n}+1}$ of the UD such that $<\mathbf{u}_1, \ldots, \mathbf{u}_\mathbf{n}, \mathbf{u}_{\mathbf{n}+1}> \in \mathbf{I}^*(f)$. That there is *at least one* such member of the UD follows from the fact that every member of the UD is denoted by at least one individual constant (this is guaranteed by condition 1 of our definition of $\mathbf{I}^*$), and property (j) of sets that are maximally consistent in *PDE* and ∃-complete, which we repeat here:

j. For each **n**-place functor f and **n** constants $\mathbf{a}_1, \ldots, \mathbf{a}_\mathbf{n}$, there is at least one constant **b** such that the formula $f(\mathbf{a}_1, \ldots, \mathbf{a}_\mathbf{n}) = \mathbf{b}$ is a member of Γ^*.

Given these two facts, condition 4 ensures that for each **n** members $\mathbf{u}_1, \ldots, \mathbf{u}_\mathbf{n}$ of the UD there is at least one member $\mathbf{u}_{\mathbf{n}+1}$ of the UD such that $<\mathbf{u}_1, \ldots, \mathbf{u}_\mathbf{n}, \mathbf{u}_{\mathbf{n}+1}> \in \mathbf{I}^*(f)$, for any functor f. To show that there is at most one such member $\mathbf{u}_{\mathbf{n}+1}$, let us assume, to the contrary, that there is also a member of the UD $\mathbf{u}'_{\mathbf{n}+1}$, where $\mathbf{u}'_{\mathbf{n}+1} \neq \mathbf{u}_{\mathbf{n}+1}$, such that $<\mathbf{u}_1, \ldots, \mathbf{u}_\mathbf{n}, \mathbf{u}'_{\mathbf{n}+1}> \in \mathbf{I}^*(f)$.

- Then in addition to the sentence $f(\mathbf{a}_1, \ldots, \mathbf{a}_n) = \mathbf{b}$, Γ includes a sentence $f(\mathbf{a}_1, \ldots, \mathbf{a}_n) = \mathbf{c}$, such that $\mathbf{I}^*(\mathbf{c}) = \mathbf{u}'_{n+1} \neq \mathbf{I}^*(\mathbf{b})$.
- By virtue of clause 3 of the definition of $\mathbf{I}^*$, it follows that $\mathbf{q}(\mathbf{a}) \neq \mathbf{q}(\mathbf{b})$.
- But this is impossible, since $\{f(\mathbf{a}_1, \ldots, \mathbf{a}_n) = \mathbf{b}, f(\mathbf{a}_1, \ldots, \mathbf{a}_n) = \mathbf{c}\} \vdash \mathbf{b} = \mathbf{c}$ by =E.
- So $\mathbf{b} = \mathbf{c} \in \Gamma^*$, by 11.4.5, and therefore $\mathbf{q}(\mathbf{c}) = \mathbf{q}(\mathbf{b})$, by 11.4.9,

It follows that $\mathbf{I}^*(\mathbf{c}) = \mathbf{I}^*(\mathbf{b})$, and so there is at most one member $\mathbf{u}_{n+1}$ of the UD such that $<\mathbf{u}_1, \ldots, \mathbf{u}_n, \mathbf{u}_{n+1}> \in \mathbf{I}^*(f)$.

We must also ensure that condition 5 can be met, that is, that there are not two atomic sentences $\mathbf{A}\mathbf{a}_1 \ldots \mathbf{a}_n$ and $\mathbf{A}\mathbf{a}'_1 \ldots \mathbf{a}'_n$ such that one is a member of Γ^* and the other is not, yet $\langle \mathbf{I}^*(\mathbf{a}_1), \ldots, \mathbf{I}^*(\mathbf{a}_n)\rangle = \langle \mathbf{I}^*(\mathbf{a}'_1), \ldots, \mathbf{I}^*(\mathbf{a}'_n)\rangle$. In the case of *PD*, it was simple to show this, for distinct constants were interpreted to designate distinct individuals. However, $\mathbf{q}$ may assign the same positive integer to more than one constant, and as a consequence condition 3 may interpret several constants to designate the same value. Here our previous results will be useful. Suppose that the constants $\mathbf{a}_1, \ldots, \mathbf{a}_n$ and $\mathbf{a}'_1, \ldots, \mathbf{a}'_n$ are such that $\langle \mathbf{I}^*(\mathbf{a}_1), \ldots, \mathbf{I}^*(\mathbf{a}_n)\rangle = \langle \mathbf{I}^*(\mathbf{a}'_1), \ldots, \mathbf{I}^*(\mathbf{a}'_n)\rangle$.

- Then, by clause 3 of the definition of $\mathbf{I}^*$, $\mathbf{q}(\mathbf{a}_i) = \mathbf{q}(\mathbf{a}'_i)$ for each $\mathbf{i}$.
- It follows from 11.4.9 that for each $\mathbf{i}$, $\mathbf{a}_i = \mathbf{a}_i' \in \Gamma^*$.
- Therefore, because $\mathbf{a}_1 = \mathbf{a}'_1$ is a member of Γ^*, property (i) assures us that $\mathbf{A}\mathbf{a}_1 \ldots \mathbf{a}_n \in \Gamma^*$ if and only if $\mathbf{A}\mathbf{a}'_1 \ldots \mathbf{a}_n \in \Gamma^*$, and because $\mathbf{a}_2 = \mathbf{a}'_2$ is a member of Γ^*, property (i) assures us that $\mathbf{A}\mathbf{a}_1\mathbf{a}_2 \ldots \mathbf{a}_n \in \Gamma^*$ if and only if $\mathbf{A}\mathbf{a}'_1\,\mathbf{a}'_2 \ldots \mathbf{a}_n \in \Gamma^*$, and so on until we note that because $\mathbf{a}_n = \mathbf{a}'_n$ is a member of Γ^*, property (i) assures us that $\mathbf{A}\mathbf{a}_1 \ldots \mathbf{a}_n \in \Gamma^*$ if and only if $\mathbf{A}\mathbf{a}'_1 \ldots \mathbf{a}'_n \in \Gamma^*$.

We conclude that if $\langle \mathbf{I}^*(\mathbf{a}_1), \ldots, \mathbf{I}^*(\mathbf{a}_n)\rangle = \langle \mathbf{I}^*(\mathbf{a}'_1), \ldots, \mathbf{I}^*(\mathbf{a}'_n)\rangle$, then $\mathbf{A}\mathbf{a}_1 \ldots \mathbf{a}_n \in \Gamma^*$ if and only if $\mathbf{A}\mathbf{a}'_1 \ldots \mathbf{a}'_n \in \Gamma^*$. So condition 5 can indeed be met.

To establish result 11.4.7 for *PDE*—that every set Γ^* that is both maximally consistent in *PDE* and $\exists$-complete is also quantificationally consistent—we can prove by mathematical induction that a sentence $\mathbf{P}$ of *PDE* is true on $\mathbf{I}^*$ if and only if $\mathbf{P} \in \Gamma^*$. The proof is similar to that for *PD*, except that we must change the basis clause to consider closed complex terms as well as constants, and we must also consider formulas containing the identity operator. We shall use the following result:

11.4.10: For any closed complex term $\mathbf{t}$ and variable assignment $\mathbf{d_I}$, $\text{den}_{\mathbf{I}^*,\mathbf{d_I}}(\mathbf{t}) = \mathbf{I}^*(\mathbf{a})$, where $\mathbf{a}$ is the alphabetically earliest individual constant such that $\mathbf{t} = \mathbf{a} \in \Gamma^*$. (Property (j) of sets that are maximally consistent in *PDE* and $\exists$-complete guarantees that there *is* such a constant $\mathbf{a}$.)

Proof: See Exercise 16.

Here is the revised proof.

Proof of basis clause: Either $\mathbf{P}$ is a sentence letter or $\mathbf{P}$ has the form $\mathbf{At}_1 \ldots \mathbf{t_n}$ or $\mathbf{t}_1 = \mathbf{t}_2$. If $\mathbf{P}$ is a sentence letter, then, by clause 2 of the definition of $\mathbf{I}^*$, it follows that $\mathbf{P}$ is true on $\mathbf{I}^*$ if and only if $\mathbf{P} \in \Gamma^*$.

If $\mathbf{P}$ has the form $\mathbf{At}_1 \ldots \mathbf{t_n}$ then $\mathbf{P}$ is true on $\mathbf{I}^*$ if and only if, for every variable assignment $\mathbf{d}_{\mathbf{I}^*}$, $<\mathrm{den}_{\mathbf{I}^*,\mathbf{d_I}^*}(\mathbf{t}_1), \ldots, \mathrm{den}_{\mathbf{I}^*,\mathbf{d_I}^*}(\mathbf{t_n})> \in \mathbf{I}^*(\mathbf{A})$.

- Property (j) guarantees, for each complex term $\mathbf{t_i}$, that there is an alphabetically earliest constant $\mathbf{a_i}$ such that $\mathbf{t_i} = \mathbf{a_i}$ is a member of Γ^*.
- Moreover, by virtue of the rule =E, $\mathbf{At}_1' \ldots \mathbf{t_n}'$, where $\mathbf{t_i}'$ is $\mathbf{t_i}$ if $\mathbf{t_i}$ is a constant and $\mathbf{t_i}'$ is $\mathbf{a_i}$ otherwise, is derivable from the set consisting of $\mathbf{At}_1 \ldots \mathbf{t_n}$ and each of these identity sentences.
- So, by 11.4.5, $\mathbf{P} \in \Gamma^*$ if and only if the sentence $\mathbf{At}_1' \ldots \mathbf{t_n}' \in \Gamma^*$.
- In addition, $\mathrm{den}_{\mathbf{I}^*,\mathbf{d_I}^*}(\mathbf{t_i}) = \mathbf{I}^*(\mathbf{t_i}')$, trivially if $\mathbf{t_i}$ is a constant and by 11.4.10 if $\mathbf{t_i}$ is a complex term.
- So $<\mathrm{den}_{\mathbf{I}^*,\mathbf{d_I}^*}(\mathbf{t}1), \ldots, \mathrm{den}_{\mathbf{I}^*,\mathbf{d_I}^*}(\mathbf{t_n})> \in \mathbf{I}^*(\mathbf{A})$ if and only if $<\mathbf{I}^*(\mathbf{t_1}'), \ldots, \mathbf{I}^*(\mathbf{t_n}')> \in \mathbf{I}^*(\mathbf{A})$,
- and clause 5 in the definition of $\mathbf{I}^*$ guarantees that $\mathbf{At}_1' \ldots \mathbf{t_n}' \in \Gamma^*$ if and only if $<\mathbf{I}^*(\mathbf{t_1}'), \ldots, \mathbf{I}^*(\mathbf{t_n}')> \in \mathbf{I}^*(\mathbf{A})$.

We conclude that $\mathbf{At}_1 \ldots \mathbf{t_n} \in \Gamma^*$ if and only if $\mathbf{At}_1 \ldots \mathbf{t_n}$ is true on $\mathbf{I}^*$.

If $\mathbf{P}$ has the form $\mathbf{t}_1 = \mathbf{t}_2$, then

- $\mathbf{P}$ is true on $\mathbf{I}^*$ if and only if, for each variable assignment $\mathbf{d}_{\mathbf{I}^*}$, $\mathrm{den}_{\mathbf{I}^*,\mathbf{d_I}^*}(\mathbf{t}_1) = \mathrm{den}_{\mathbf{I}^*,\mathbf{d_I}^*}(\mathbf{t}_2)$.
- Again, for each complex term $\mathbf{t_i}$, property (j) guarantees that there is an alphabetically earliest constant $\mathbf{a_i}$ such that $\mathbf{t_i} = \mathbf{a_i} \in \Gamma^*$.
- So, by virtue of =E and result 11.4.5, $\mathbf{t}_1 = \mathbf{t}_2 \in \Gamma^*$ if and only if $\mathbf{t_1}' = \mathbf{t_2}' \in \Gamma^*$, where $\mathbf{t_i}'$ is $\mathbf{t_i}$ if $\mathbf{t_i}$ is a constant and $\mathbf{t_i}'$ is $\mathbf{a_i}$ otherwise.
- Moreover, $\mathrm{den}_{\mathbf{I}^*,\mathbf{d_I}^*}(\mathbf{t_i}) = \mathbf{I}^*(\mathbf{t_i}')$, trivially if $\mathbf{t_i}$ is a constant and by result 11.4.10 if $\mathbf{t_i}$ is a complex term.
- So $\mathrm{den}_{\mathbf{I}^*,\mathbf{d_I}^*}(\mathbf{t}_1) = \mathrm{den}_{\mathbf{I}^*,\mathbf{d_I}^*}(\mathbf{t}_2)$ if and only if $\mathbf{I}^*(\mathbf{t_1}') = \mathbf{I}^*(\mathbf{t_2}')$.
- By the way in which $\mathbf{I}^*$ was constructed, $\mathbf{I}^*(\mathbf{t_1}') = \mathbf{I}^*(\mathbf{t_2}')$ if and only if $\mathbf{q}(\mathbf{t_1}') = \mathbf{q}(\mathbf{t_2}')$. By result 11.4.9, $\mathbf{q}(\mathbf{t_1}') = \mathbf{q}(\mathbf{t_2}')$ if and only if $\mathbf{t_1}' = \mathbf{t_2}' \in \Gamma^*$.

We may conclude that $\mathbf{t}_1 = \mathbf{t}_2 \in \Gamma^*$ if and only if $\mathbf{t}_1 = \mathbf{t}_2$ is true on $\mathbf{I}^*$.

Because every member of Γ* is true on **I***, Γ* is quantificationally consistent. And, with results 11.4.4 and 11.4.7 established for *PDE*, along with the necessary modifications of 11.4.8 (see Exercise 11.4.15), we know that result 11.4.2 is also true for *PDE*. It follows that *PDE* is complete for predicate logic with identity and functions.

11.4**E** EXERCISES

***1.** Prove that if $\Gamma \models \mathbf{P}$ then $\Gamma \cup \{\sim \mathbf{P}\}$ is quantificationally inconsistent.

2. Prove that if $\Gamma \cup \{\sim \mathbf{P}\}$ is inconsistent in *PD* then $\Gamma \vdash \mathbf{P}$.

3. Using Metatheorem 11.4.1, prove the following:
a. Every argument of *PL* that is quantificationally valid is valid in *PD*.
b. Every sentence of *PL* that is quantificationally true is a theorem in *PD*.
*c. Every pair of sentences **P** and **Q** of *PL* that are quantificationally equivalent are equivalent in *PD*.

4. Prove that the sentences of *PL* can be enumerated. (*Hint:* See Section 6.4.)

5. Prove the following:

If $\Gamma \vdash \mathbf{P}$ and Γ is a subset of Γ′, then $\Gamma' \vdash \mathbf{P}$.

6. Prove that any set Γ* constructed as in our proof of Lemma 11.4.4 is ∃-complete.

7. Prove that the sequence of sentences constructed in the proof of 11.4.3 is a derivation in *PD* by showing (by mathematical induction) that each sentence in the new sequence can be justified with the same rule as the corresponding sentence in the original derivation.

***8.** Prove 11.4.8, using result 11.1.13.

***9.** Prove 11.4.5.

10. Explain why, in results 11.4.4 and 11.4.7, we constructed a set that was both ∃-complete and maximally consistent in *PD*, rather than a set that was just maximally consistent in *PD*.

11. Let system *PD** be just like *PD* except that the rule ∀E is replaced by the following rule:

*Universal Elimination** (∀E*)

$(\forall \mathbf{x})\mathbf{P}$

$\sim (\exists \mathbf{x}) \sim \mathbf{P}$

Prove that the system *PD** is complete for predicate logic.

***12.** Let system *PD** be just like *PD* except that the rules ∃E and ∃I are replaced by the following two rules:

*Existential Elimination** (∃E*)

$$\begin{array}{|l} (\exists \mathbf{x})\mathbf{P} \\ \\ \sim (\forall \mathbf{x}) \sim \mathbf{P} \end{array}$$

*Existential Introduction** (∃I*)

$$\begin{array}{|l} \sim (\forall \mathbf{x}) \sim \mathbf{P} \\ \\ (\exists \mathbf{x})\mathbf{P} \end{array}$$

Prove that system *PD** is complete for predicate logic.

13. Using the results in the proof of Metatheorem 11.4.1, prove the following theorem (known as the *Löwenheim Theorem*):

If a sentence **P** of *PL* is not quantificationally false, then there is an interpretation with the set of positive integers as the UD on which **P** is true.

***14.** Prove the following metatheorem (known as the *Löwenheim-Skolem Theorem*):

If a set Γ of sentences of *PL* is quantificationally consistent, then there is an interpretation with the set of positive integers as the UD on which every member of Γ is true.

***15.** Show the changes that must be made in the proofs of 11.4.3 and 11.4.8 so that these results will hold for *PLE* and *PDE*. (*Hint:* Exercise 8 suggested that you use result 11.1.13 in proving 11.4.8; so you must check whether 11.1.13 needs to be changed.)

16. Prove result 11.4.10.

17. Show that the Löwenheim Theorem (and consequently the more general Löwenheim-Skolem Theorem) does *not* hold for *PLE*.

11.5 THE SOUNDNESS OF THE TREE METHOD

We have presented the tree method as a means of testing for semantic properties of sentences and sets of sentences in both sentential logic and predicate logic. In this section and the next we shall prove that the tree method in Chapter 9 fulfills a claim we have made: A finite set of sentences of *PL* is quantificationally inconsistent if and only if every systematic tree for that set closes. In this section we shall prove that the tree method is **sound** for predicate logic—that if a systematic tree for a set of sentences of *PL* closes, then the set is quantificationally inconsistent. We shall prove the same for predicate logic with identity and functions. In both cases we can then be assured that, if we pronounce a

set of sentences inconsistent because a tree for that set closes, our pronouncement is correct. In the next section we shall prove that the tree method is **complete** for predicate logic—that if a finite set of sentences is quantificationally inconsistent, then every systematic tree for that set is bound to close. Knowing that the method is complete, we shall also know that open systematic trees do establish quantificational consistency. (With a simple adaptation of parts of our proofs, the soundness and completeness of the tree method for sentential logic can also be established. This will be addressed in the exercises.)

Our *Soundness Metatheorem* for the tree method is:

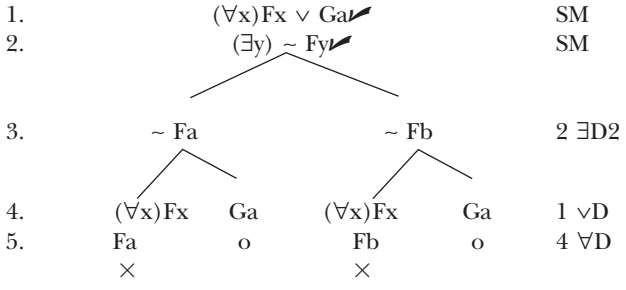
Metatheorem 11.5.1: If a systematic tree for a set Γ of sentences of *PL* closes, then Γ is quantificationally inconsistent.

(As we shall see in the exercises for this section, soundness also holds for nonsystematic trees.) Our proof of Metatheorem 11.5.1 will rely heavily on the following observation about the decomposition rules used in constructing trees: Each rule is consistency-preserving in the sense that, if we have a consistent set of sentences and apply a decomposition rule to one of the sentences in that set, at least one of the sentences that results (there will be only one in the case of a nonbranching rule) can consistently be added to the set. As we build a tree for a set of sentences, we are, in effect, building supersets of the one we started with—the set of sentences occurring on a branch is a superset of the original set. Given the sense in which the decomposition rules are consistency-preserving, at least one of the supersets formed on a branch by repeated application of decomposition rules will be quantificationally consistent if the set being tested is quantificationally consistent. This will be important in establishing Metatheorem 11.5.1, for if the supersets that comprise the branches of a closed tree are all quantificationally inconsistent, each such branch will contain a contradictory pair of literals. Because the decomposition rules are consistency-preserving in the sense described, it follows that the only way we can end up with every superset being quantificationally inconsistent (that is, with a closed tree) is by starting with a set that is quantificationally inconsistent. And that is what Metatheorem 11.5.1 says.

Our observation that the decomposition rules are consistency-preserving must be proved. To facilitate our proof, we introduce the concept of a *level* of a tree. The first (occurrence of a) sentence on any tree is at level 1. For any other sentence **P**, **P** is at level **i** + 1, where **i** is the level of the sentence occurring immediately before **P** on the same branch of the tree. The line numbers used to annotate trees in Chapters 4 and 9 do not always correspond to levels because we adopted the convention in those chapters that only one decomposition rule can be cited on each line. Consider, for example, the tree on page 420. Lines 1–3 do correspond directly to levels 1–3 of that tree. Line 4, however, displays only one of the sentences occurring at level 4. The sentence on line 10 on the left-hand branch is *also* at level 4, for the sentence that occurs immediately before it on the same branch is at level 3. Similar observations hold for sentences further down this branch of the tree.

We shall establish that our decomposition rules are consistency-preserving by showing that at each level **i** of a systematic tree for a quantificationally consistent set of sentences, either there is at least one branch that was completed prior to that level on which the sentences form a quantificationally consistent set or there is at least one branch that extends at least as far as level **i** such that the sentences on that branch up to and including level **i** form a quantificationally consistent set.

As an example of what we want to prove, here is a completed open tree for the set {(∀x)Fx ∨ Ga, (∃y) ~ Fy}:



Our claim holds of each of the levels 1–5 of this tree.

- At level 1 there is at least one branch such that the set of sentences occurring on that branch through and including level 1 form a quantificationally consistent set: {(∀x)Fx ∨ Ga}.
- At level 2 there is at least one branch such that the set of sentences occurring on that branch form a quantificationally consistent set: {(∀x)Fx ∨ Ga, (∃y) ~ Fy}.
- At level 3 there are two (and so at least one) such branches: {(∀x)Fx ∨ Ga, (∃y) ~ Fy, ~ Fa} and {(∀x)Fx ∨ Ga, (∃y) ~ Fy, ~ Fb}.
- At level 4 there are also two such branches: {(∀x)Fx ∨ Ga, (∃y) ~ Fy, ~ Fa, Ga} and {(∀x)Fx ∨ Ga, (∃y) ~ Fy, ~ Fb, Ga}. The branches that include '(∀x)Fx' are not candidates, but we have not claimed that quantificationally consistent sets will be found on *all* branches.
- At level 5 there is no branch that extends to that level that contains a quantificationally consistent set of sentences, but there is at least one branch (in fact, there are two) that was completed at an earlier level, level 4, on which the sentences form a quantificationally consistent set.

So the claim is true of all levels of this tree.

The fact that the claim holds for every level of a systematic tree for a quantificationally consistent set allows us to conclude that if a tree for a set of

sentences closes, then the set must be quantificationally inconsistent. A tree that closes has a finite number of levels, say **n**, and the claim is false for level **n**, because at this point every branch contains a pair of contradictory literals—so the set of sentences on each branch is quantificationally inconsistent. We may therefore conclude that the set of sentences for which the tree was constructed is also quantificationally inconsistent.

To facilitate the proof of our claim about the levels of any systematic tree for a quantificationally consistent set of sentences, we introduce some terminology. For any branch that extends to level **i** or further, we call that part of the branch that extends to level **i** a '**path**' to level **i**, and we say that the path **contains** the set of sentences that occur on it. In the last tree displayed,

- There is exactly one path to level 1, and it contains the set of sentences {(∀x)Fx ∨ Ga}.
- There is exactly one path to level 2, and it contains the set of sentences {(∀x)Fx ∨ Ga, (∃y) ~ Fy}.
- There are two paths to level 3, and they contain the sets
  - {(∀x)Fx ∨ Ga, (∃y) ~ Fy, ~ Fa}
  - {(∀x)Fx ∨ Ga, (∃y) ~ Fy, ~ Fb}.
- There are four paths to level 4, and they contain the sets
  - {(∀x)Fx ∨ Ga, (∃y) ~ Fy, ~ Fa, (∀x)Fx}
  - {(∀x)Fx ∨ Ga, (∃y) ~ Fy, ~ Fa, Ga}
  - {(∀x)Fx ∨ Ga, (∃y) ~ Fy, ~ Fb, (∀x)Fx}
  - {(∀x)Fx ∨ Ga, (∃y) ~ Fy, ~ Fb, Ga}.
- There are two paths to level 5, and they contain the sets
  - {(∀x)Fx ∨ Ga, (∃y) ~ Fy, ~ Fa, (∀x)Fx, Fa}
  - {(∀x)Fx ∨ Ga, (∃y) ~ Fy, ~ Fb, (∀x)Fx, Fb}

Finally, a **completed** path to level **i** of a tree is a completed branch of that tree that ends at level **i**. We state our claim about the levels of a systematic tree for a quantificationally consistent set in terms of paths in the Consistent Branch Lemma:

> **11.5.2** (the *Consistent Branch Lemma*): Each level **i** of a tree for a quantificationally consistent set of sentences of *PL* is such that either (a) there is at least one completed path to a level earlier than **i** that contains a quantificationally consistent set of sentences or (b) there is at least one path to level **i** that contains a quantificationally consistent set of sentences.

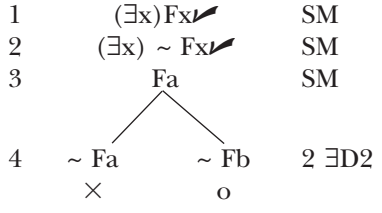
We shall prove 11.5.2 by establishing a more specific claim (which will later be useful in proving that systematic trees for sets of sentences with finite models always have a completed open branch).

Let Γ be a finite set of sentences of *PL* that is quantificationally consistent, and let I be an interpretation.

We call interpretation $\mathbf{I_p}$ a **path-variant** of **I** for path **p** of a tree for the set of sentences Γ if

- $\mathbf{I_p}$ is just like **I** except that, for each constant **a** that occurs in some sentence on the path but not in any member of Γ,
  - there is a member **u** of the UD such that
  - $\mathbf{I_p}(\mathbf{a}) = \mathbf{u}$, and
  - for every other constant **b** occurring on the path but not in any member of Γ, $\mathbf{I_p}(\mathbf{b}) \neq \mathbf{u}$.

Before proceeding, we'll give an example of a path-variant, and explain why we are looking at path-variants, using the following completed open tree as an example:



The rightmost open branch is the only completed open branch in this tree. The sentences on this branch include the constant 'b', which does not occur in the set {(∃x)Fx, (∃x) ~ Fx, Fa} for which the tree was constructed—the rule ∃D2 required the introduction of a new instantiating constant at this point. We want to show that if the set members are all true on an interpretation, then all of the sentences on at least one completed open branch are true on an interpretation. But consider an interpretation **I** that makes the following assignments:

UD: Set of positive integers
F: {<2>}
a: 2
b: 1

All three set members are true on such an interpretation, so this interpretation shows that the set is consistent. But is does not follow that '~ Fb' is true on this interpretation; in fact, it is false. So we can't say that if all of the set members are true on an interpretation, then all of the sentences on a completed open branch are true on *that* interpretation. Rather, this is where the concept of path-variants comes in. There are infinitely many path-variants of **I** for the

completed open branch. Each of these assigns a positive integer other than 1 to the constant 'b', for example:

UD: Set of positive integers
F: {<2>}
a: 2
b: 2

We have intentionally displayed a (in fact, the only) path-variant of **I** on which all sentences on the completed open branch are true. We will be proving that such a path-variant can always be found, no matter what the original set members are and no matter which model of those sentences we begin with.

We shall show that

> **11.5.3:** If a finite set Γ of sentences of *PL* is true on an interpretation **I**, then each level **i** of a systematic tree for Γ is such that either (a) there is at least one completed path **p** to a level earlier than **i** that contains a set of sentences all of which are true on a path-variant $\mathbf{I_p}$ or (b) there is at least one path **p** to level **i** that contains a set of sentences all of which are true on a path-variant $\mathbf{I_p}$.

We establish result 11.5.3 by mathematical induction on the levels of a systematic tree for a quantificationally consistent set of sentences of *PL*. Letting Γ be a finite set of sentences of *PL* and **I** an interpretation on which every member of Γ is true,

> *Basis clause:* Level 1 of a systematic tree for Γ is such that either (a) or (b) holds.
>
> *Inductive step:* If every level less than or equal to level **k** of a systematic tree for Γ is such that either (a) or (b) holds, then the same is true of level **k** + 1 of a systematic tree for Γ.
>
> ---
>
> *Conclusion:* Every level of a systematic tree for Γ is such that either (a) or (b) holds.

Proof of basis clause: There is exactly one path to level 1 of any tree, and that path contains the unit set {**P**}, where **P** is a member of the set Γ for which the tree is being constructed. **P** is true on **I** since every member of Γ is, and **I** is in this case a path-variant of itself (since the path contains no constants that do not occur in Γ). So (b) holds for level 1.

Proof of inductive step: We assume the inductive hypothesis for an arbitrary positive integer **k**: For each level **i** of a tree for Γ that is less than or equal to level **k**, either (a) or (b) holds. We must show that on this assumption either (a) or (b) holds for level **k** + 1 of a tree for Γ as well. We first note that if (a) holds for an earlier level **i**, then (a) holds for level **k** + 1 as well, for a completed path to a level earlier than **i** is also a completed path to a level earlier than **k** + 1.

Now we must consider the case where (a) does not hold for any level prior to $\mathbf{k} + 1$. In this case it follows from the inductive hypothesis that (b) holds for every level prior to $\mathbf{k} + 1$ and, in particular, that (b) is true of level $\mathbf{k}$. If, in addition, one of these paths is *completed* at level $\mathbf{k}$, then (a) is true of level $\mathbf{k} + 1$. If there is not such a path to level $\mathbf{k}$, we still know, on our assumption that (b) is true of level $\mathbf{k}$, that at least one (noncompleted) path $\mathbf{p}$ to level $\mathbf{k}$ contains a set of sentences all of which are true on a path-variant $\mathbf{I_p}$. Call this path $\Gamma_\mathbf{k}$ and the variant $\mathbf{I}_{\Gamma_\mathbf{k}}$. Because the path is not complete at level $\mathbf{k}$, it is extended to level $\mathbf{k} + 1$ by application of some tree rule. We shall now consider the rules that might be used to extend the path to level $\mathbf{k} + 1$ and show that in each case application of the rule results in at least one path $\mathbf{p}$ to level $\mathbf{k} + 1$ that contains a set of sentences all of which are true on a path-variant $\mathbf{I_p}$—thereby establishing that (b) holds for level $\mathbf{k} + 1$ as well.

We divide the rules into six cases.

Case 1: The path $\Gamma_\mathbf{k}$ is extended to level $\mathbf{k} + 1$ by adding a set member at that level, justified by SM. Because decomposition rules apply *after* all set members have been entered, the only sentences on the path $\Gamma_\mathbf{k}$ are members of Γ, and the sentence entered at level $\mathbf{k} + 1$ is also a member of Γ. So there is a path to level $\mathbf{k} + 1$ that contains a subset of Γ, and all sentences on this path are therefore true on $\mathbf{I}$, which in this case is a path-variant of itself.

Case 2: The path $\Gamma_\mathbf{k}$ is extended to level $\mathbf{k} + 1$ as a result of applying one of the nonbranching rules $\sim\sim$ D, &D, $\sim\vee$D, $\sim\supset$D, $\sim\forall$D, or $\sim\exists$D to a sentence $\mathbf{P}$ on $\Gamma_\mathbf{k}$. In each case $\{\mathbf{P}\}$ quantificationally entails the sentence $\mathbf{Q}$ entered on level $\mathbf{k} + 1$ (see Exercise 11.5.3). Therefore all the sentences on $\Gamma_\mathbf{k}$ and the sentence $\mathbf{Q}$ are true on $\mathbf{I}_{\Gamma_\mathbf{k}}$, which is a path-variant of $\mathbf{I}$ for the path that has been extended by one of these nonbranching rules (since none of these rules adds a new individual constant to the tree). Thus, there is a path to level $\mathbf{k} + 1$ that contains a set of sentences all of which are true on a path-variant of $\mathbf{I}$ for this path.

Case 3: The path is extended to form two paths to level $\mathbf{k} + 1$ as a result of applying one of the branching rules $\sim$&D, $\vee$D, or $\supset$D to a sentence $\mathbf{P}$ on $\Gamma_\mathbf{k}$. Letting $\mathbf{Q}$ be one of the sentences that was entered on level $\mathbf{k} + 1$ and $\mathbf{R}$ the other, it can be shown that on any interpretation on which $\mathbf{P}$ is true either $\mathbf{Q}$ is true or $\mathbf{R}$ is true (see Exercise 11.5.4). Therefore either all the sentences on $\Gamma_\mathbf{k}$ plus $\mathbf{Q}$ are true on $\mathbf{I}_{\Gamma_\mathbf{k}}$, which is a path-variant of $\mathbf{I}$ for the new path containing $\mathbf{Q}$, or all the sentences on $\Gamma_\mathbf{k}$ plus $\mathbf{R}$ are true on $\mathbf{I}_{\Gamma_\mathbf{k}}$, which is a path-variant of $\mathbf{I}$ for the new path containing $\mathbf{R}$. Thus there is a path to level $\mathbf{k} + 1$ that contains a set of sentences all of which are true on a path-variant of $\mathbf{I}$ for that path.

Case 4: The path is extended to level $\mathbf{k} + 1$ as a result of applying either $\equiv$D or $\sim\equiv$D (see Exercise 5).

Case 5: The path is extended to level $\mathbf{k} + 1$ as a result of applying $\forall$D. Then $\Gamma_{\mathbf{k}}$ contains a sentence $(\forall \mathbf{x})\mathbf{P}$ such that $\mathbf{P}(\mathbf{a}/\mathbf{x})$ is entered at level $\mathbf{k} + 1$, where $\mathbf{a}$ is either 'a' if no constants occur on $\Gamma_{\mathbf{k}}$ or the alphabetically earliest constant that does occur. Because $(\forall \mathbf{x})\mathbf{P}\mathbf{x} \vDash \mathbf{P}(\mathbf{a}/\mathbf{x})$ (result 11.1.4), $\mathbf{P}(\mathbf{a}/\mathbf{x})$ is true on $\mathbf{I}_{\Gamma_{\mathbf{k}}}$. If no constant occurred on $\Gamma_{\mathbf{k}}$, then $\mathbf{I}_{\Gamma_{\mathbf{k}}}$ is also a path-variant of $\mathbf{I}$ for the new path to level $\mathbf{k} + 1$ because $\mathbf{I}_{\Gamma_{\mathbf{k}}}(\mathbf{a}) \neq \mathbf{I}_{\Gamma_{\mathbf{k}}}(\mathbf{b})$ for any other constant $\mathbf{b}$ occurring on $\Gamma_{\mathbf{k}}$ but not in Γ—there are no such constants $\mathbf{b}$. If, on the other hand, $\mathbf{a}$ is a constant that already occurs on $\Gamma_{\mathbf{k}}$, then we have not added a new constant to the path, and so $\mathbf{I}_{\Gamma_{\mathbf{k}}}$ is in this case also a path-variant of $\mathbf{I}$ for the extended path to level $\mathbf{k} + 1$. Either way, there is a path $\mathbf{p}$ to level $\mathbf{k} + 1$ that contains a set of sentences all of which are true on a path-variant $\mathbf{I}_{\mathbf{p}}$.

Case 6: The path is extended to level $\mathbf{k} + 1$ as a result of applying $\exists$D2. Then $\Gamma_{\mathbf{k}}$ contains a sentence $(\exists \mathbf{x})\mathbf{P}$ such that $\mathbf{P}(\mathbf{a}_1/\mathbf{x}), \ldots, \mathbf{P}(\mathbf{a}_{\mathbf{m}}/\mathbf{x}), \mathbf{P}(\mathbf{a}_{\mathbf{m}+1}/\mathbf{x})$ are entered on distinct paths to level $\mathbf{k} + 1$, where $\mathbf{a}_1, \ldots, \mathbf{a}_{\mathbf{m}}$ are all the individual constants that occur in sentences on $\Gamma_{\mathbf{k}}$ and $\mathbf{a}_{\mathbf{m}+1}$ is the alphabetically earliest constant that does not occur on $\Gamma_{\mathbf{k}}$. We consider two possibilities:

a. If any one (or more) of $\mathbf{P}(\mathbf{a}_1/\mathbf{x}), \ldots, \mathbf{P}(\mathbf{a}_{\mathbf{m}}/\mathbf{x})$ is true on $\mathbf{I}_{\Gamma_{\mathbf{k}}}$, then the path to level $\mathbf{k} + 1$ on which that substitution instance was entered is a path that contains a set of sentences all of which are true on a path-variant of $\mathbf{I}$ for that path ($\mathbf{I}_{\Gamma_{\mathbf{k}}}$ is a path-variant of the newly formed path because the substitution instance does not introduce a new constant).

b. We shall show that if (a) does not hold, then the path that results from the addition of $\mathbf{P}(\mathbf{a}_{\mathbf{m}+1}/\mathbf{x})$ meets the requirement. Because $(\exists \mathbf{x})\mathbf{P}$ is true on $\mathbf{I}_{\Gamma_{\mathbf{k}}}$ and because $\mathbf{a}_{\mathbf{m}+1}$ does not occur in any sentence on $\Gamma_{\mathbf{k}}$, our proof of result 11.1.10 (Exercise 11.1.5) shows that $\mathbf{P}(\mathbf{a}_{\mathbf{m}+1}/\mathbf{x})$ is true on an interpretation $\mathbf{I}_{\Gamma_{\mathbf{k}}'}$ that is just like $\mathbf{I}_{\Gamma_{\mathbf{k}}}$ except that $\mathbf{I}_{\Gamma_{\mathbf{k}}'}(\mathbf{a}_{\mathbf{m}+1}) = \mathbf{u}$, where $\mathbf{u}$ is a member of the UD such that $\mathbf{d}[\mathbf{u}/\mathbf{x}]$ satisfies $\mathbf{P}$ on $\mathbf{I}_{\Gamma_{\mathbf{k}}}$ (we know that there is such a member because $(\exists \mathbf{x})\mathbf{P}$ is true on $\mathbf{I}_{\Gamma_{\mathbf{k}}}$). This member $\mathbf{u}$ is not assigned to any other individual constant $\mathbf{b}$ occurring on $\Gamma_{\mathbf{k}}$ but not in Γ (for, if it were, it would follow from result 11.1.13 that $\mathbf{P}(\mathbf{b}/\mathbf{x})$ is true on $\mathbf{I}_{\Gamma_{\mathbf{k}}}$, which contradicts our assumption that (a) does not hold). Thus $\mathbf{I}_{\Gamma_{\mathbf{k}}'}$ is a path-variant of $\mathbf{I}$, for the path extended to level $\mathbf{k} + 1$ by the addition of $\mathbf{P}(\mathbf{a}_{\mathbf{m}+1}/\mathbf{x})$, on which every sentence in the new path is true.

We have considered each rule that might be used to extend the path $\Gamma_{\mathbf{k}}$ to level $\mathbf{k} + 1$ and have shown that in each case there is at least one path to level $\mathbf{k} + 1$ that contains a set of sentences all of which are true on a path-variant of $\mathbf{I}$ for that path. That completes the proof of the inductive step.

Therefore, result 11.5.3 holds for every level of a tree for a set of sentences all of which are true on interpretation **I**.

The Consistent Branch Lemma 11.5.2 follows immediately from result 11.5.3, for in establishing the existence of paths containing sentences all of which are true on some path-variant of **I**, we have established that the set of sentences on each such path has a model and therefore forms a quantificationally consistent set.

Metatheorem 11.5.1 follows from the Consistent Branch Lemma and the fact that the null tree, which is the single tree for the empty set of sentences of *PL*, is not closed (the null branch does not contain any sentences and therefore does not contain a contradictory pair of literals). If a tree for a set Γ of sentences is closed, then every branch on that tree is closed and hence contains a contradictory pair of literals, which means that the set of sentences on every branch is truth-functionally inconsistent. Therefore, the last level of any closed tree is such that neither (a) nor (b) holds (and there is a "last" level, because such a tree must be finite), and it follows, by the Consistent Branch Lemma, that the set for which the tree was constructed is quantificationally inconsistent. We conclude that the tree method is sound for predicate logic.

To establish that the tree method for predicate logic with identity and functions is also sound, we first note that the cases in the inductive proof of result 11.5.3 carry over to predicate logic with identity and functions. We must add two more cases to the inductive step, one to cover paths that are extended by an application of =D and one to cover paths that are extended by an application of CTD.

> **Case 7:** The path $\Gamma_{\mathbf{k}}$ is extended to level $\mathbf{k} + 1$ as a result of applying =D. Then $\Gamma_{\mathbf{k}}$ contains sentences $\mathbf{t}_1 = \mathbf{t}_2$ and $\mathbf{P}$ such that a sentence $\mathbf{P}(\mathbf{t}_1//\mathbf{t}_2)$ was entered at level $\mathbf{k} + 1$. It follows from 11.2.4, which we repeat here, that $\mathbf{P}(\mathbf{t}_1//\mathbf{t}_2)$ is true on $\mathbf{I}_{\Gamma_k}$:
>
> **11.2.4:** For any closed terms $\mathbf{t}_1$ and $\mathbf{t}_2$, if $\mathbf{P}$ is a sentence that contains $\mathbf{t}_1$, then $\{\mathbf{t}_1 = \mathbf{t}_2, \mathbf{P}\} \vDash \mathbf{P}(\mathbf{t}_2//\mathbf{t}_1)$, and if $\mathbf{P}$ is a sentence that contains $\mathbf{t}_2$, then $\{\mathbf{t}_1 = \mathbf{t}_2, \mathbf{P}\} \vDash \mathbf{P}(\mathbf{t}_1//\mathbf{t}_2)$.
>
> In addition, $\mathbf{I}_{\Gamma_k}$ is a path-variant for the new path to level $\mathbf{k} + 1$ because =D does not introduce new constants.
>
> **Case 8:** The path $\Gamma_{\mathbf{k}}$ is extended to level $\mathbf{k} + 1$ as a result of applying CTD. Then $\Gamma_{\mathbf{k}}$ contains a literal sentence with a closed complex term $f(\mathbf{a}_1, \ldots, \mathbf{a}_n)$, where $\mathbf{a}_1, \ldots, \mathbf{a}_n$ are all constants, such that $\mathbf{b}_1 = f(\mathbf{a}_1, \ldots, \mathbf{a}_n), \ldots, \mathbf{b}_m = f(\mathbf{a}_1, \ldots, \mathbf{a}_n)$, and $\mathbf{b}_{m+1} = f(\mathbf{a}_1, \ldots, \mathbf{a}_n)$ are entered on distinct paths to level $\mathbf{k} + 1$, where $\mathbf{b}_1, \ldots, \mathbf{b}_m$ are all the individual constants that occur in sentences on $\Gamma_{\mathbf{k}}$ and $\mathbf{b}_{m+1}$ is the alphabetically earliest constant that does not occur on $\Gamma_{\mathbf{k}}$. We consider two possibilities:
>
> a. If any one (or more) of $\mathbf{b}_1 = f(\mathbf{a}_1, \ldots, \mathbf{a}_n), \ldots, \mathbf{b}_m = f(\mathbf{a}_1, \ldots, \mathbf{a}_n)$ is true on $\mathbf{I}_{\Gamma_k}$, then the path to level $\mathbf{k} + 1$ on

which that identity sentence was entered is a path that contains a set of sentences all of which are true on a path-variant of $\mathbf{I}_{\Gamma_k}$ for that path because $\mathbf{I}_{\Gamma_k}$ is a path-variant of itself for the newly formed path, since the identity sentence does not introduce a new constant).

b. Now consider the case where none of $\mathbf{b}_1 = f(\mathbf{a}_1, \ldots, \mathbf{a}_n), \ldots, \mathbf{b}_m = f(\mathbf{a}_1, \ldots, \mathbf{a}_n)$ is true on $\mathbf{I}_{\Gamma_k}$. Because $\mathbf{b}_{m+1}$ does not occur in any sentence on Γ_k, our proof of result 11.2.5 (Exercise 11.2.7) shows that $\Gamma \cup \{\mathbf{b}_{m+1} = f(\mathbf{a}_1, \ldots, \mathbf{a}_n)\}$ is true on an interpretation $\mathbf{I'}_{\Gamma_k}$ that is just like $\mathbf{I}_{\Gamma_k}$ except that $\mathbf{I'}_{\Gamma_k}(\mathbf{b}_{m+1}) = \mathbf{u}$, where $\mathbf{u}$ is the member of the UD such that $< \mathbf{I}_{\Gamma_k}(\mathbf{a}_1), \ldots, \mathbf{I}_{\Gamma_k}(\mathbf{a}_n), \mathbf{u}>$ is a member of $\mathbf{I}_{\Gamma_k}(f)$. This member $\mathbf{u}$ is not assigned to any other individual constant $\mathbf{b}_i$ occurring on Γ_k but not in Γ (for, if it were, it would follow that $\mathbf{b}_i = f(\mathbf{a}_1, \ldots, \mathbf{a}_n)$ is true on $\mathbf{I}_{\Gamma_k}$, which contradicts our assumption that (a) does not hold). Thus $\mathbf{I'}_{\Gamma_k}$ is a path-variant of $\mathbf{I}_{\Gamma_k}$, for the path extended to level $\mathbf{k} + 1$ by the addition of $\mathbf{b}_{m+1} = f(\mathbf{a}_1, \ldots, \mathbf{a}_n)$, on which every sentence in the new path is true.

Finally, we must note that a branch of a tree for predicate logic with identity closes in one of *two* cases: Either the branch contains a pair of contradictory literals or the branch contains a sentence of the form $\sim \mathbf{t} = \mathbf{t}$. To show that Metatheorem 11.5.1 for predicate logic with identity follows from the Consistent Branch Lemma 11.5.2, we must therefore additionally show that the set of sentences on a branch that closes because it contains a sentence $\sim \mathbf{t} = \mathbf{t}$ is quantificationally inconsistent. This is not difficult: $\mathbf{t} = \mathbf{t}$ is quantificationally true, so $\sim \mathbf{t} = \mathbf{t}$ is quantificationally false, and therefore any set that contains $\sim \mathbf{t} = \mathbf{t}$ is quantificationally inconsistent. This and the addition of Cases 7 and 8 in the proof of result 11.5.3 suffice to show that the tree method is sound for predicate logic with identity and functions.

Result 11.5.3 also allows us to prove another claim that we made in Chapter 9, namely, that trees constructed in accordance with The System have the **finite model property:**

Metatheorem 11.5.4: If a finite set Γ of sentences of *PL* has a finite model, that is, an interpretation with a finite UD on which every member of Γ is true, then every systematic tree for Γ will contain a *completed* open branch.[2]

In such a case, we shall be able to conclude in a *finite* number of steps that the set is quantificationally consistent.

[2]This metatheorem, along with result 11.5.3, is due to George Boolos, "Trees and Finite Satisfiability: Proof of a Conjecture of Burgess," *Notre Dame Journal of Formal Logic,* 25(3) (1984), 193–197.

Proof of Metatheorem 11.5.4: Let Γ be a finite set of sentences such that there is an interpretation **I** with a finite UD on which every member of Γ is true. By result 11.5.3, every level **i** of a systematic tree for Γ is such that either (a) there is at least one completed path **p** to a level earlier than **i** that contains a set of sentences all of which are true on the path-variant $\mathbf{I_p}$ or (b) there is at least one path **p** to level **i** that contains a set of sentences all of which are true on the path-variant $\mathbf{I_p}$.

Consider, for any level **i**, a path that satisfies either (a) or (b). There is an upper limit to the number of distinct individual constants not already occurring in Γ that can occur on this path (constants that were introduced by an application of ∀D or ∃D2), namely, the size **n** of the finite UD for **I**. For if a path contains more than **n** new individual constants, it cannot meet the condition in the definition of path-variants that each of these constants be assigned a member of the UD that is *different* from the members assigned to other new constants; there would not be enough members of the UD to go around. In addition, because Γ is a finite set it can only contain a finite number of constants, so a path that satisfies either (a) or (b) can contain only a finite number of constants.

But a path of a systematic tree that contains only a finite number of individual constants must be finitely long. Each of the decomposition rules &D, ~ &D, ∨D, ~ ∨D, ~ ⊃D, ~ ≡D, ~ ~ D, ∀D, and ∃D2 produces sentences with fewer occurrences of logical operators than the sentence being decomposed. The rules ⊃D, ≡D, ~ ∃D, and ~ ∀D produce one or two sentences with the same number of occurrences of logical operators as the sentence being decomposed, but the sentences so produced have one of the forms ~ **P** (in the case of ⊃D and ≡D), (∃**x**) ~ **P** (in the case of ~ ∀D), or (∀**x**) ~ **P** (in the case of ~ ∃D). Each of the latter sentences, if not a literal, will be decomposed by a rule that produces only sentences with fewer occurrences of logical operators. Because subsequent applications of decomposition rules produce sentences with fewer and fewer occurrences of logical operators, literals are eventually reached. The only way in which a branch of a systematic tree can continue indefinitely is through repeated instantiation of one or more universally quantified sentences by ∀D, each instantiation containing a different instantiating constant. But this cannot be the case with a branch that contains a finite number of individual constants. Therefore the paths that we are guaranteed by result 11.5.3 can be only finitely long.

In addition, The System was designed to guarantee that if a path *can* be completed (or closed) after a finite number of applications of rules, it *will* be completed. Stages 1 and 2 (and stage 3, in the case of *PLE*) each require that we decompose *all* sentences on the tree of the specified sort before going to the next stage, and at each stage there are only finitely many sentences. The System does not allow one branch to be developed indefinitely while others are ignored, and so

a branch that can be completed after a finite number of steps will be completed.

We conclude that at some finite level **i** of a systematic tree for Γ, there is a path that meets condition (a) of result 11.5.3. In addition, because this path meets (a), it is a completed *open* path. This establishes Metatheorem 11.5.4.

Metatheorem 11.5.4 is also true of *PLE*; this proof is left as an exercise.

11.5**E** EXERCISES

***1.** Show that Metatheorem 11.5.1 holds for nonsystematic trees as well as for systematic ones. (Result 11.5.3 is not generally true of nonsystematic trees, so you should prove Lemma 11.5.2 directly by mathematical induction.)

2. Using Metatheorem 11.5.1, prove the following:

a. If a sentence **P** of *SL* is such that {**P**} has a closed truth-tree, then **P** is quantificationally false.

b. If a sentence **P** of *SL* is such that {~ **P**} has a closed truth-tree, then **P** is quantificationally true.

*c. If a set {~ (**P** ≡ **Q**)} has a closed truth-tree, then **P** and **Q** are quantificationally equivalent.

d. If a set Γ ∪ {~ **P**} has a closed truth-tree, then Γ ⊨ **P**.

*e. If the set consisting of the premises and the negation of the conclusion of an argument has a closed truth-tree, then that argument is quantificationally valid.

3. Prove that if a sentence **Q** is obtained from a sentence **P** by application of one of the following tree rules, then {**P**} ⊨ **Q**.

a. ~ ~ D
*b. &D
*c. ~ ∨D
d. ~ ⊃D
e. ∀D
*f. ~ ∀D
*g. ~ ∃D

4. Prove that if sentences **Q** and **R** are obtained from a sentence **P** by application of one of the following tree rules, then on any interpretation on which **P** is true, either **Q** is true or **R** is true.

a. ~ &D
*b. ∨D
*c. ⊃D

5. Prove Case 4 in the inductive step of the proof of Lemma 11.5.2.

6. If we were to drop the rule ∀D from the tree method, would the method still be sound for predicate logic? Explain.

7. Explain how we can adapt the proof of Metatheorem 11.5.1 to establish that the tree method for *SL* is sound for sentential logic.

***8.** Prove that Metatheorem 11.5.4 is true of *PLE*.

11.6 THE COMPLETENESS OF THE TREE METHOD

In the last section we established that the tree method is sound for predicate logic—if a tree for a set of sentences of *PL* closes, then that set is quantificationally inconsistent. In this section we shall prove that the tree method is also **complete** for sentential logic. The *Completeness Metatheorem* for the tree method is

> **Metatheorem 11.6.1:** If a finite set Γ of sentences of *PL* is quantificationally inconsistent, then every systematic tree for Γ closes.

Whereas soundness ensures that we are correct in pronouncing a set inconsistent if we can construct a closed tree for that set, completeness ensures that we are correct in pronouncing a set consistent if a systematic tree for that set does not close. The requirement that the tree be systematic is important, as we shall see; and the reader should remember that a tree that is constructed in accordance with The System but is abandoned before every branch closes and before at least one branch becomes a completed open branch does not count as a systematic tree.

We shall prove that the tree method is complete by establishing that the contrapositive of Metatheorem 11.6.1 is true—that if a systematic tree for a set of sentences of *PL* does not close, then the set is quantificationally consistent. There are three parts to the proof. First, we shall prove that, if a systematic tree fails to close and does not contain a completed open branch after a finite number of steps in its construction, then it has at least one branch with infinitely many sentences. Second, we shall prove that for any completed open branch or infinite branch of a systematic truth-tree, the set of sentences occurring on that branch is a special sort of set known as a *Hintikka set*.[3] Finally we shall present a method of constructing a model for any Hintikka set. This will establish that every Hintikka set is quantificationally consistent and consequently that the set of sentences occurring on either a completed open branch or an infinite branch of a systematic truth-tree is quantificationally consistent. From these three steps, it follows that, if a systematic tree for Γ fails to close, Γ is a subset of a Hintikka set and is therefore also quantificationally consistent. Therefore, if a finite set Γ is quantificationally *in*consistent, then every systematic tree for Γ will close—and this will establish Metatheorem 11.6.1.

Consider a systematic tree such that at no level **i** does the tree contain a completed open branch and at no level **i** is the tree closed. Our first task is to show that such a tree must contain an infinite branch. Because the tree fails to close or to contain a completed open branch at any level **i**, the tree must contain infinitely many sentences (strictly speaking, infinitely many *occurrences* of sentences—the sentences need not be distinct). The tree contains infinitely

[3]These sets were first studied by J. Hintikka, in "Form and Content in Quantification Theory," *Acta Philosophica Fennica,* 8 (1955), 7–55; and "Notes on Quantification Theory," *Societas Scientiarum Fennica, Commentationes Physico-Mathematicae,* 17(12) (1955).

many sentences because it takes infinitely many steps to construct a systematic tree that neither is closed nor contains a completed open branch at any level, and each step in the construction involves adding at least one new sentence. It remains to be shown that a systematic truth-tree containing infinitely many sentences has at least one branch that is infinitely long—at least one branch that contains an infinite number of sentences. The reason that this needs to be *proven* is that a tree *could* contain infinitely many sentences and yet be such that each of its branches was only finitely long, *if* it contained infinitely many branches. So we need to establish the following lemma, the Infinite Branch Lemma:

> **11.6.2** (the *Infinite Branch Lemma*): Every systematic tree that contains an infinite number of occurrences of sentences has at least one branch that is infinitely long.[4]

> **Proof of Lemma 11.6.2:** Some definitions will be useful for the proof. We shall say that a sentence **P** (throughout, read *occurrence of a sentence* whenever we speak of a sentence) in a tree is **above** sentence **Q** when **P** and **Q** lie on the same branch of the tree and **P** is at an earlier level of the tree. **Q** is an **immediate successor** of **P** if **P** and **Q** lie on the same branch and **P** is one level earlier than **Q**. Every sentence in a tree, except those that occur at the ends of branches, has a finite number of immediate successors—one if a nonbranching rule is applied, two if a branching rule other than ∃D2 is applied, and **m** + 1, where **m** is the number of individual constants already occurring on the sentence's branch, if ∃D2 is applied.
>
> We shall now show that if a systematic tree contains infinitely many sentences then there is at least one infinite branch in the tree, by starting at level 1 and working down through the levels of the tree. The sentence at level 1 of such a tree—call it $\mathbf{P}_1$—is above every other sentence in the tree. Therefore this sentence is above infinitely many sentences (subtracting 1 from an infinite number leaves an infinite number). $\mathbf{P}_1$ has a finite number of successors at level 2. At least one of these immediate successors must therefore be above infinitely many sentences—for if they were all above only a finite number of successors, $\mathbf{P}_1$ would also only be above a finite number of successors. The reasoning that we have just used can be generalized: If a sentence at any level is above infinitely many sentences, then at least one immediate successor of that sentence is above infinitely many sentences. We therefore conclude, by mathematical induction, that at every level of the tree there is at least one sentence that is above infinitely many sentences, which means that at least one branch is infinitely long.

[4]This follows as a special case of a famous lemma known as *König's Lemma* (D. König, *Theorie der endlichen und unendlichen Graphen,* Leipzig, 1936).

We may therefore conclude that a systematic tree that fails to close either has a completed open branch after a finite number of steps or has at least one infinite branch. This will be important in what follows.

Turning to the second step of the proof of Metatheorem 11.6.1, we define a **Hintikka set** to be a set Γ of sentences of *PL* that has the following properties:

a. There is no atomic sentence $\mathbf{P}$ such that both $\mathbf{P}$ and $\sim \mathbf{P}$ are members of Γ.
b. If $\sim \sim \mathbf{P} \in \Gamma$, then $\mathbf{P} \in \Gamma$.
c. If $\mathbf{P} \,\&\, \mathbf{Q} \in \Gamma$, then $\mathbf{P} \in \Gamma$ and $\mathbf{Q} \in \Gamma$.
d. If $\sim (\mathbf{P} \,\&\, \mathbf{Q}) \in \Gamma$, then either $\sim \mathbf{P} \in \Gamma$ or $\sim \mathbf{Q} \in \Gamma$.
e. If $\mathbf{P} \vee \mathbf{Q} \in \Gamma$, then either $\mathbf{P} \in \Gamma$ or $\mathbf{Q} \in \Gamma$.
f. If $\sim (\mathbf{P} \vee \mathbf{Q}) \in \Gamma$, then $\sim \mathbf{P} \in \Gamma$ and $\sim \mathbf{Q} \in \Gamma$.
g. If $\mathbf{P} \supset \mathbf{Q} \in \Gamma$, then either $\sim \mathbf{P} \in \Gamma$ or $\mathbf{Q} \in \Gamma$.
h. If $\sim (\mathbf{P} \supset \mathbf{Q}) \in \Gamma$, then $\mathbf{P} \in \Gamma$ and $\sim \mathbf{Q} \in \Gamma$.
i. If $\mathbf{P} \equiv \mathbf{Q} \in \Gamma$, then either $\mathbf{P} \in \Gamma$ and $\mathbf{Q} \in \Gamma$ or $\sim \mathbf{P} \in \Gamma$ and $\sim \mathbf{Q} \in \Gamma$.
j. If $\sim (\mathbf{P} \equiv \mathbf{Q}) \in \Gamma$, then either $\mathbf{P} \in \Gamma$ and $\sim \mathbf{Q} \in \Gamma$ or $\sim \mathbf{P} \in \Gamma$ and $\mathbf{Q} \in \Gamma$.
k. If $(\forall \mathbf{x})\mathbf{P} \in \Gamma$, then at least one substitution instance of $(\forall \mathbf{x})\mathbf{P}$ is a member of Γ, and for every constant $\mathbf{a}$ that occurs in some sentence of Γ, $\mathbf{P}(\mathbf{a}/\mathbf{x}) \in \Gamma$.
l. If $\sim (\forall \mathbf{x})\mathbf{P} \in \Gamma$, then $(\exists \mathbf{x}) \sim \mathbf{P} \in \Gamma$.
m. If $(\exists \mathbf{x})\mathbf{P} \in \Gamma$, then for at least one constant $\mathbf{a}$, $\mathbf{P}(\mathbf{a}/\mathbf{x}) \in \Gamma$.
n. If $\sim (\exists \mathbf{x})\mathbf{P} \in \Gamma$, then $(\forall \mathbf{x}) \sim \mathbf{P} \in \Gamma$.

We call a branch of a tree a *Hintikka branch* if and only if the sentences on that branch constitute a Hintikka set. We now prove that every completed open branch and every infinite branch of a systematic tree is a Hintikka branch, which will establish the Hintikka Branch Lemma:

> **11.6.3** (the *Hintikka Branch Lemma*): Every systematic tree that is not closed has at least one Hintikka branch.

Afterward we shall show that every Hintikka set is quantificationally consistent.

> **Proof of Lemma 11.6.3:** If a systematic tree fails to close, then either the tree has a completed open branch or, by Lemma 11.6.2, the tree has an infinite branch. We shall show that each of these two types of branches is a Hintikka branch—that is, that the set of sentences occurring on such a branch has properties (a)–(n).
>
> First consider completed open branches. By definition a completed open branch is a finite branch that is open—there is no

contradictory pair of literals **P** and ~ **P** on the branch—and each sentence on that branch is one of the following:

1. A literal (an atomic sentence or the negation of an atomic sentence)
2. A sentence that is not universally quantified and that has been decomposed
3. A universally quantified sentence $(\forall \mathbf{x})\mathbf{P}$ such that at least one substitution instance of $(\forall \mathbf{x})\mathbf{P}$ occurs on the branch and, for each constant **a** occurring on the branch, $\mathbf{P}(\mathbf{a}/\mathbf{x})$ occurs on the branch

The set of sentences on a completed open branch has property (a) because the branch does not include a pair of contradictory literals. Every sentence that has one of the forms described in properties (b)–(j) and (l)–(n) has been decomposed (because this is a completed open branch), and so it is easily verified that the set of sentences on a completed open branch has those properties. (For example, if a sentence ~ ~ **P** occurs on a completed open branch and has been decomposed by an application of ~ ~ D, then **P** also occurs on that branch—which establishes property (b).) Finally the set of sentences on a completed open branch also has property (k), for part 3 of the definition of completed open branches stipulates that property (k) is satisfied. We conclude that a completed open branch is therefore a Hintikka branch.

Now we turn to infinite branches. The System for tree construction was designed to guarantee that every infinite (nonterminating) branch is a Hintikka branch; we shall explain how it does so. First, a nonterminating branch is not closed (a branch that closes contains only finitely many sentences); so the set of sentences on such a branch must have property (a) of Hintikka sets. Second, the alternation of stages 1 and 2 of The System ensures that each nonliteral sentence that does not have the form $(\forall \mathbf{x})\mathbf{P}$ is decomposed a finite number of levels after the level on which it occurs, that for each universally quantified sentence $(\forall \mathbf{x})\mathbf{P}$ and constant **a** on a branch of the tree $\mathbf{P}(\mathbf{a}/\mathbf{x})$ is entered within a finite number of levels, and that at least one substitution instance $\mathbf{P}(\mathbf{a}/\mathbf{x})$ is entered. Each such addition yields only a finite number of levels, so every sentence on a branch must be decomposed if the branch is infinite. Therefore the set of sentences on a nonterminating branch satisfies properties (b)–(n) of Hintikka sets, as well as property (a). Every infinite branch of a systematic tree is therefore a Hintikka branch.

Finally we prove

11.6.4 (the *Hintikka Set Lemma*): Every Hintikka set is quantificationally consistent.

From this it will follow that if a systematic tree for a set Γ of sentences does not close then Γ is quantificationally consistent—for one of the branches that contains the sentences in Γ is a Hintikka branch.

Proof of Lemma 11.6.4: Let Γ be a Hintikka set of sentences of *PL*. We first associate with each individual constant of *PL* a distinct positive integer—**i** is associated with the alphabetically **i**th constant. We shall prove that every sentence of Γ is true on the interpretation **I** defined as follows:

1. The UD is the set consisting of the positive integers that are associated with the individual constants occurring in members of Γ. If no member of Γ contains an individual constant, let the UD be the set {1}.
2. For each sentence letter **P**, $\mathbf{I}(\mathbf{P}) = \mathbf{T}$ if and only if $\mathbf{P} \in \Gamma$.
3. For each individual constant **a** that occurs in some sentence in Γ, $\mathbf{I}(\mathbf{a})$ is the positive integer associated with **a**. For each constant **a** that does not occur in any sentence of Γ, $\mathbf{I}(\mathbf{a})$ is the smallest positive integer in the UD (which, by specification 1, is nonempty).
4. For each **n**-place predicate **A**, $\mathbf{I}(\mathbf{A})$ includes all and only those **n**-tuples $\langle \mathbf{u}_1, \ldots, \mathbf{u}_n \rangle$ such that for some constants $\mathbf{a}_1, \ldots, \mathbf{a}_n$, $\mathbf{A}\mathbf{a}_1 \ldots \mathbf{a}_n \in \Gamma$ and $\langle \mathbf{I}(\mathbf{a}_1), \ldots, \mathbf{I}(\mathbf{a}_n) \rangle = \langle \mathbf{u}_1, \ldots, \mathbf{u}_n \rangle$.

We shall use mathematical induction to prove that every member of Γ is true on **I**. Our induction will not be on the number of occurrences of logical operators in a sentence since some of the clauses of the proof would not work in that case (see Exercise 11.6.5). Instead, we shall appeal to the *length* of a sentence. Where **P** is a formula of *PL*, let the *length* of **P** be the number of occurrences of sentence letters, predicates, and logical operators in **P**. No sentence of *PL* has length 0 since every sentence contains at least one sentence letter or predicate. So the basis clause begins with length 1.

Basis clause: Every sentence **P** of length 1 is such that if $\mathbf{P} \in \Gamma$ then $\mathbf{I}(\mathbf{P}) = \mathbf{T}$.

Inductive step: If every sentence **P** of length less than or equal to **k** is such that if $\mathbf{P} \in \Gamma$ then $\mathbf{I}(\mathbf{P}) = \mathbf{T}$, then the same holds of every sentence **P** of length $\mathbf{k} + 1$.

Conclusion: Every sentence **P** is such that if $\mathbf{P} \in \Gamma$, then $\mathbf{I}(\mathbf{P}) = \mathbf{T}$.

Proof of basis clause: A sentence of length 1 is an atomic sentence. If **P** is a sentence letter, then by part 2 of the definition of **I**, $\mathbf{I}(\mathbf{P}) = \mathbf{T}$ if $\mathbf{P} \in \Gamma$. If **P** is an atomic sentence of the form $\mathbf{A}\mathbf{a}_1 \ldots \mathbf{a}_n$, then by part 4 of the definition of **I**, $<\mathbf{I}(\mathbf{a}_1), \ldots, \mathbf{I}(\mathbf{a}_n)> \in \mathbf{I}(\mathbf{A})$ if $\mathbf{A}\mathbf{a}_1 \ldots \mathbf{a}_n \in \Gamma$, and so $\mathbf{I}(\mathbf{P}) = \mathbf{T}$.

Proof of inductive step: We assume that the inductive hypothesis holds for some arbitrary positive integer **k**—that every sentence of length **k** or less that is a member of Γ is true on **I**. We must show that any sentence **P** of length **k** + 1 that is a member of Γ is also true on **I**. It is easy to verify that **P**, being nonatomic, must have one of the forms specified in properties (a)–(n) of Hintikka sets; and we shall consider each of these forms that **P** may have.

Case 1: P has the form ~ **Q**, where **Q** is an atomic sentence. If ~ **Q** ∈ Γ then, by property (a) of Hintikka sets, **Q** ∉ Γ. If **Q** is a sentence letter then, by part 2 of the definition of **I**, **I**(**Q**) = **F** and so **I**(~ **Q**) = **T**. If **Q** has the form $\mathbf{Aa}_1 \ldots \mathbf{a_n}$ then, by part 4 of the definition of **I**, $\langle \mathbf{I}(\mathbf{a}_1), \ldots, \mathbf{I}(\mathbf{a_n})\rangle \notin \mathbf{I}(\mathbf{A})$. This is because each constant that occurs in some member of Γ designates a positive integer different from that designated by any other constant occurring in Γ (by part 3), and so there is no other set of constants occurring in Γ that also designate the members of the **n**-tuple $\langle \mathbf{I}(\mathbf{a}_1), \ldots, \mathbf{I}(\mathbf{a_n})\rangle$. We may therefore conclude, from the fact that $\mathbf{Aa}_1 \ldots \mathbf{a_n} \notin \Gamma$, that $\mathbf{I}(\mathbf{Aa}_1 \ldots \mathbf{a_n}) = \mathbf{F}$ and $\mathbf{I}(\sim \mathbf{Aa}_1 \ldots \mathbf{a_n}) = \mathbf{T}$.

Case 2: P has the form ~ ~ **Q**. If ~ ~ **Q** ∈ Γ then, by property (b) of Hintikka sets, **Q** ∈ Γ. The length of **Q** is less than **k** + 1, so, by the inductive hypothesis, **I**(**Q**) = **T**. Therefore **I**(~ ~ **Q**) = **T** as well.

Case 3: P has the form **Q** & **R**. If **Q** & **R** ∈ Γ then, by property (c) of Hintikka sets, **Q** ∈ Γ and **R** ∈ Γ. By the inductive hypothesis (**Q** and **R** both having lengths less than **k** + 1), **I**(**Q**) = **T** and **I**(**R**) = **T**. So **I**(**Q** & **R**) = **T**.

Case 4: P has the form ~ (**Q** & **R**). If ~ (**Q** & **R**) ∈ Γ then, by property (d) of Hintikka sets, either ~ **Q** ∈ Γ or ~ **R** ∈ Γ. The lengths of ~ **Q** and of ~ **R** are less than the length of ~ (**Q** & **R**), so, by the inductive hypothesis, either **I**(~ **Q**) = **T** or **I**(~ **R**) = **T**. If **I**(~ **Q**) = **T**, then **I**(**Q**) = **F**, **I**(**Q** & **R**) = **F**, and **I**(~ (**Q** & **R**)) = **T**. Similarly, if **I**(~ **R**) = **T**, then **I**(~ (**Q** & **R**)) = **T**. Either way, **I**(~ (**Q** & **R**)) = **T**.

Cases 5–10: P has one of the forms **Q** ∨ **R**, ~ (**Q** ∨ **R**), **Q** ⊃ **R**, ~ (**Q** ⊃ **R**), **Q** ≡ **R**, or ~ (**Q** ≡ **R**) (see Exercise 11.6.3).

Case 11: P has the form (∀**x**)**Q**. If **P** ∈ Γ then, by property (k) of Hintikka sets, for every constant **a** that occurs in Γ, **Q**(**a**/**x**) ∈ Γ (and there is at least one such constant). Each substitution instance has a length less than **k** + 1, so it follows from the inductive hypothesis that, for each of these sentences, **I**(**Q**(**a**/**x**)) = **T**. Moreover, each member of the UD is designated by some constant occurring in Γ (by part 1 of the definition of the interpretation **I**—because at least one constant occurs in Γ), so for each member of the UD there is a constant **a** such that **Q**(**a**/**x**) ∈ Γ and hence is true on **I**. It therefore follows from 11.6.5 that **I**((∀**x**)**Q**) = **T**:

11.6.5: Let **I** be an interpretation on which for each member **u** of the UD there is at least one constant **a** such that $\mathbf{I}(\mathbf{a}) = \mathbf{u}$ and $\mathbf{I}(\mathbf{P}(\mathbf{a}/\mathbf{x})) = \mathbf{T}$. Then $\mathbf{I}((\forall\mathbf{x})\mathbf{P}) = \mathbf{T}$.

Proof: See Exercise 11.6.4.

Case 12: P has the form $\sim (\forall\mathbf{x})\mathbf{Q}$. If $\sim (\forall\mathbf{x})\mathbf{Q} \in \Gamma$, then, by property (1) of Hintikka sets, $(\exists\mathbf{x}) \sim \mathbf{Q} \in \Gamma$, and by property (m) $\sim \mathbf{Q}(\mathbf{a}/\mathbf{x}) \in \Gamma$ for some constant **a**. The length of $\sim \mathbf{Q}(\mathbf{a}/\mathbf{x})$ is less than $\mathbf{k} + 1$, so, by the inductive hypothesis, $\mathbf{I}(\sim \mathbf{Q}(\mathbf{a}/\mathbf{x})) = \mathbf{T}$ and therefore $\mathbf{I}(\mathbf{Q}(\mathbf{a}/\mathbf{x})) = \mathbf{F}$. Because $\{(\forall\mathbf{x})\mathbf{Q}\} \vDash \mathbf{Q}(\mathbf{a}/\mathbf{x})$ (result 11.1.4), it follows that $\mathbf{I}((\forall\mathbf{x})\mathbf{Q}) = \mathbf{F}$ and $\mathbf{I}(\sim (\forall\mathbf{x})\mathbf{Q}) = \mathbf{T}$.

Cases 13 and 14: P has one of the forms $(\exists\mathbf{x})\mathbf{Q}$ or $\sim (\exists\mathbf{x})\mathbf{Q}$ (see Exercise 11.6.3).

That completes the proof of the inductive step. Therefore every sentence that is a member of the Hintikka set Γ is true on **I**, and this shows that Γ is quantificationally consistent.

The Hintikka Branch Lemma 11.6.3 and the Hintikka Set Lemma 11.6.4 can now be used to establish Metatheorem 11.6.1. If a systematic tree for a set Γ of sentences does not close, then the tree has at least one Hintikka branch (Lemma 11.6.3). The set of sentences on that Hintikka branch is quantificationally consistent (Lemma 11.6.4). Therefore, because every member of Γ lies on that branch (as well as on every other branch), Γ is quantificationally consistent. So if Γ is quantificationally *inconsistent,* then every systematic tree for Γ closes.

We note that the proof that we have just given is a *constructive* completeness proof. We have shown how, given a Hintikka branch of a systematic tree for a set of sentences Γ, to construct a model for Γ. This establishes a claim made in Chapter 9: An interpretation showing the quantificational consistency of Γ can always be constructed from a completed open branch of a tree for Γ.

Finally, the tree method for predicate logic with identity and functions is also complete, and this can be shown by making appropriate changes in the proofs of Lemmas 11.6.3 and 11.6.4. We define a Hintikka set for *PLE* to be a set Γ that has the properties (a)–(n) of our earlier definition and that also has these properties:

o. No sentence of the form $\sim \mathbf{t} = \mathbf{t}$ is a member of Γ.
p. If $\mathbf{a} = \mathbf{t}$, where **a** is a constant, is a member of Γ, and a literal sentence $\mathbf{P} \in \Gamma$ contains **t**, then every sentence $\mathbf{P}(\mathbf{a}//\mathbf{t})$ is also a member of Γ.
q. If a complex term $f(\mathbf{a}_1, \ldots, \mathbf{a}_n)$ in which $\mathbf{a}_1, \ldots, \mathbf{a}_n$ are individual constants occurs in any literal sentence in Γ, then, for at least one constant **b**, $\mathbf{b} = f(\mathbf{a}_1, \ldots, \mathbf{a}_n) \in \Gamma$.

The proof of Lemma 11.6.3—that every systematic tree that does not close *has* a Hintikka branch—runs as before, except that we replace talk of the

constant **a** occurring on a branch with talk of the closed term **t** occurring on a branch. We must also add the following:

> The set of sentences on a completed open branch of a systematic tree must have property (o), because by definition a branch that does not close does not contain a sentence of the form ~ **t** = **t**. Property (p) must hold by virtue of the requirement in clause 4 of the definition of a completed open branch. Property (q) must hold by virtue of the final component of the definition of a completed open branch.
>
> Similar remarks establish that infinite branches in trees for *PLE* are also Hintikka branches: The cycle of stages 1–4 in The System, and the fact that each stage adds only a finite number of sentences, guarantees that every sentence on a branch will be decomposed if the branch is infinite. Note that the requirement in stage 2 of The System, that **P**(**t**/**x**) be entered for a *complex* term **t** only if doing so will close the branch on which it is entered, plays a crucial role here. If The System allowed such substitutions for complex terms that did not close the branch, we could have a branch containing a sequence such as '(∀x)G*f*(x)', 'G*f*(a)', 'a = *f*(a)', 'G*f*(*f*(a))', 'G*f*(*f*(*f*(a)))', 'G*f*(*f*(*f*(*f*(a))))', . . . in which '*f*(a)' has been continuously substituted for 'a' at the expense of decomposing other sentences on the branch. Similarly the restriction (ii) in stage 4 guarantees that we will not have a branch containing a sequence such as 'a = *f*(a)', 'a = *f*(*f*(a))', 'a = *f*(*f*(*f*(a)))', . . . or 'a = *f*(a)', 'Ga', 'G*f*(a)', 'G*f*(*f*(a))', G*f*(*f*(*f*(a))), . . . in which '*f*(a)' has been continuously substituted for 'a' at the expense of decomposing other sentences on the branch.

To show that Lemma 11.6.4 holds for predicate logic with identity and functions, we must define the interpretations of individual constants, and the UD for the interpretation, differently for a *PLE* Hintikka set Γ than these were defined for *PL*. In the case of *PL*, we were able to assign a unique member of the UD to each individual constant. We cannot generally do this for *PLE*, because of the identity predicate: the truth of an identity sentence such as 'a = b' requires that 'a' and 'b' designate the same individual. We associate members of the set of positive integers with individual constants in two steps:

- We begin with an association of unique positive integers with the individual constants of *PLE*, as we did for *PL*: Associate the positive integer **i** with the alphabetically **i**th individual constant of *PLE*. Let '**p**' designate this association and let **p**(**a**) stand for the integer that has been associated with the constant **a**.
- Now we run through the individual constants, in alphabetical order, to determine whether it is necessary to adjust the associated values because of identity sentences in the *PLE* Hintikka set. Our second association, which we shall designate with '**q**', is defined as: **q**(**a**) = **p**(**a**′) if **a**′ is the alphabetically earliest constant such that **a**′ = **a** is a member of Γ, and **q**(**a**) = **p**(**a**) otherwise.

For example, if 'a = b', 'a = e', and 'a = k' are all members of the Hintikka set, then $\mathbf{q}$('a') = 1 (there is no alphabetically earlier constant, so $\mathbf{q}$('a') will never have to be adjusted), $\mathbf{q}$('e') = $\mathbf{q}$('a') = 1, and $\mathbf{q}$('k') = $\mathbf{q}$('a') = 1.

We may now define the interpretation for a *PLE* Hintikka set Γ:

1. The UD is the set consisting of the positive integers that $\mathbf{q}$ assigns to the individual constants occurring in members of Γ. If no member of Γ contains an individual constant, let the UD be the set {1}.

We change clause 3 in the interpretation constructed in Lemma 11.6.4 to:

3. For each individual constant $\mathbf{a}$,
   - If $\mathbf{a}$ occurs in some sentence in Γ, $\mathbf{I}(\mathbf{a}) = \mathbf{q}(\mathbf{a})$.
   - If $\mathbf{a}$ does not occur in any sentence of Γ, $\mathbf{I}(\mathbf{a})$ is the smallest positive integer in the UD.

The reason for not assigning $\mathbf{q}(\mathbf{a})$ to constants that do not occur in Γ is that we want all of the members of the UD to be named by constants that *do* occur in Γ.

We must also add a fifth clause to complete the definition of the interpretation for the Hintikka set Γ:

4. For each $\mathbf{n}$-place functor f, $\mathbf{I}(f)$ consists of all and only the $\mathbf{n} + 1$-tuples $<\mathbf{d}_1, \ldots, \mathbf{d}_n, \mathbf{d}_{n+1}>$ of members of the UD such that either
   (i) there exist constants $\mathbf{a}_1, \ldots, \mathbf{a}_n, \mathbf{a}_{n+1}$ such that $\mathbf{a}_{n+1} = f(\mathbf{a}_1, \ldots, \mathbf{a}_n) \in \Gamma$ and $\mathbf{d}_i = \mathbf{I}(\mathbf{a}_i)$, $1 \leq i \leq n + 1$, or
   (ii) there are no such constants, and $\mathbf{d}_{n+1}$ is the smallest member of the UD.

Part (i) of clause 5 ensures that the relevant identity sentences turn out to be true, while part (ii) ensures that $\mathbf{I}(f)$ is indeed a function defined over all members of the UD, by arbitrarily specifying a value of the function for all other $\mathbf{n}$-tuples.

We must, however, ascertain that clause 5 correctly defines the interpretation of an $\mathbf{n}$-place functor as a function that assigns *exactly one* member of the UD to each $\mathbf{n}$-tuple of members of the UD. It is clear that it assigns at least one member of the UD to each $\mathbf{n}$-tuple. Moreover, if case (i) doesn't apply then case (ii) will assign at most one member. It remains to show that if case (i) applies it will also assign at most one member:

- First we note that if (i) assigns more than one member of the UD to some $\mathbf{n}$-tuple of members of the UD, that will be because
  - there exist constants $\mathbf{a}_1, \ldots, \mathbf{a}_n, \mathbf{a}_{n+1}$ such that $\mathbf{a}_{n+1} = f(\mathbf{a}_1, \ldots, \mathbf{a}_n) \in \Gamma$

- and constants $\mathbf{b}_1, \ldots, \mathbf{b}_n, \mathbf{b}_{n+1}$ such that $\mathbf{b}_{n+1} = f(\mathbf{b}_1, \ldots, \mathbf{b}_n) \in \Gamma$,
- where $\mathbf{I}(\mathbf{a}_i) = \mathbf{I}(\mathbf{b}_i)$ for $1 \leq i \leq n$ and $\mathbf{I}(\mathbf{a}_{n+1}) \neq \mathbf{I}(\mathbf{b}_{n+1})$.

We will show that this is impossible.

- Note that if $\mathbf{I}(\mathbf{a}_i) = \mathbf{I}(\mathbf{b}_i)$ then, by the way we defined the values that **I** assigns to individual constants based on the association **q**, either
  - (a) $\mathbf{a}_i = \mathbf{b}_i \in \Gamma$ (in this case, $\mathbf{a}_i$ is the alphabetically earliest constant that stands on the left-hand side of an identity sentence that has $\mathbf{b}_i$ on the right-hand side), or
  - (b) $\mathbf{b}_i = \mathbf{a}_i \in \Gamma$ (in this case, $\mathbf{b}_i$ is the alphabetically earliest constant that stands on the left-hand side of an identity sentence that has $\mathbf{a}_i$ on the right-hand side), or
  - (c) there is a constant $\mathbf{c}_i$ such that both $\mathbf{c}_i = \mathbf{a}_i$ and $\mathbf{c}_i = \mathbf{b}_i$ are members of Γ ($\mathbf{c}_i$ is distinct from $\mathbf{a}_i$ and $\mathbf{b}_i$ and is the alphabetically earliest constant that stands on the left-hand side of an identity sentence that has $\mathbf{a}_i$ on the right-hand side as well as the alphabetically earliest constant that stands on the left-hand side of an identity sentence that has $\mathbf{b}_i$ on the right-hand side).
- We now perform the following substitutions in the sentences $\mathbf{a}_{n+1} = f(\mathbf{a}_1, \ldots, \mathbf{a}_n)$ and $\mathbf{b}_{n+1} = f(\mathbf{b}_1, \ldots, \mathbf{b}_n)$:
  - For each $\mathbf{a}_i$ and $\mathbf{b}_i$ occurring on the right-hand side of these identity sentences,
  - If (a) holds ($\mathbf{a}_i = \mathbf{b}_i \in \Gamma$), then replace $\mathbf{b}_i$ with $\mathbf{a}_i$ in $\mathbf{b}_{n+1} = f(\mathbf{b}_1, \ldots, \mathbf{b}_n)$.
  - If (b) holds ($\mathbf{b}_i = \mathbf{a}_i \in \Gamma$), then replace $\mathbf{a}_i$ with $\mathbf{b}_i$ in $\mathbf{a}_{n+1} = f(\mathbf{a}_1, \ldots, \mathbf{a}_n)$.
  - If (c) holds ($\mathbf{c}_i = \mathbf{a}_i$ and $\mathbf{c}_i = \mathbf{b}_i$ are members of Γ), then replace $\mathbf{a}_i$ with $\mathbf{c}_i$ in $\mathbf{a}_{n+1} = f(\mathbf{a}_1, \ldots, \mathbf{a}_n)$ and replace $\mathbf{b}_i$ with $\mathbf{c}_i$ in $\mathbf{b}_{n+1} = f(\mathbf{b}_1, \ldots, \mathbf{b}_n)$.
- Note that each replacement generates a sentence that is also a member of the Hintikka set Γ, by property (p) of Hintikka sets.
- Note also that the right-hand sides of the identity sentences that result from these replacements will be identical: either we replaced $\mathbf{b}_i$ with $\mathbf{a}_i$ in $\mathbf{b}_{n+1} = f(\mathbf{b}_1, \ldots, \mathbf{b}_n)$ and left $\mathbf{a}_i$ intact in $\mathbf{a}_{n+1} = f(\mathbf{a}_1, \ldots, \mathbf{a}_n)$, or vice versa, or we replaced both $\mathbf{a}_i$ and $\mathbf{b}_i$ with $\mathbf{c}_i$.

- Let us denote the common right-hand side as $f(\mathbf{t}_1, \ldots, \mathbf{t}_n)$; we have shown that both $\mathbf{a}_{n+1} = f(\mathbf{t}_1, \ldots, \mathbf{t}_n)$ and $\mathbf{b}_{n+1} = f(\mathbf{t}_1, \ldots, \mathbf{t}_n)$ are members of Γ.
- By virtue of property (p) of Hintikka sets for *PLE*, it follows that
  - $\mathbf{a}_{n+1} = \mathbf{b}_{n+1} \in \Gamma$, because $\mathbf{a}_{n+1} = \mathbf{b}_{n+1}$ is the result of replacing $f(\mathbf{t}_1, \ldots, \mathbf{t}_n)$ in $\mathbf{a}_{n+1} = f(\mathbf{t}_1, \ldots, \mathbf{t}_n)$ with $\mathbf{b}_{n+1}$;
  - $\mathbf{b}_{n+1} = \mathbf{a}_{n+1} \in \Gamma$, for a similar reason;
  - $\mathbf{a}_{n+1} = \mathbf{a}_{n+1} \in \Gamma$, because $\mathbf{a}_{n+1} = \mathbf{a}_{n+1}$ is the result of replacing $\mathbf{b}_{n+1}$ in either of the above with $\mathbf{a}_{n+1}$, and
  - $\mathbf{b}_{n+1} = \mathbf{b}_{n+1} \in \Gamma$, for a similar reason.
- But then $\mathbf{I}(\mathbf{a}_{n+1}) = \mathbf{I}(\mathbf{b}_{n+1})$, contrary to our previous assumption. For it follows from the construction of **I** that
  - if no identity sentence with a constant that is alphabetically earlier than $\mathbf{a}_{n+1}$ or $\mathbf{b}_{n+1}$ occurring on the left-hand side is a member of Γ, then both constants will denote either $\mathbf{p}(\mathbf{a}_{n+1})$ or $\mathbf{p}(\mathbf{b}_{n+1})$, depending on which is alphabetically earlier,
  - and if there is a constant **c** that is alphabetically earlier than both $\mathbf{a}_{n+1}$ and $\mathbf{b}_{n+1}$ such that either $\mathbf{c} = \mathbf{a}_{n+1}$ or $\mathbf{c} = \mathbf{b}_{n+1}$ is a member of Γ, then, because the identity sentences $\mathbf{a}_{n+1} = \mathbf{b}_{n+1}$ and $\mathbf{b}_{n+1} = \mathbf{a}_{n+1}$ are both members of Γ, it follows from property (p) of *PLE* Hintikka sets that both $\mathbf{c}_{n+1} = \mathbf{a}_{n+1}$ and $\mathbf{c}_{n+1} = \mathbf{b}_{n+1}$ must be members of Γas well, in which case both $\mathbf{I}(\mathbf{a}_{n+1})$ and $\mathbf{I}(\mathbf{b}_{n+1})$ are defined to be $\mathbf{p}(\mathbf{c})$ for the alphabetically earliest such constant **c**.

We may conclude that case (i) of clause 5 in the definition of an interpretation for the *PLE* Hintikka set Γ will not assign more than one member of the UD to any given **n**-tuple, so that clause 5 does indeed define a function.

We shall use the following result in the proof that every member of a Hintikka set is true on the interpretation **I** we have just defined:

> **11.6.6:** If a Hintikka set Γ contains a literal sentence **P** with a closed complex term **t**, it also contains each sentence $\mathbf{P}(\mathbf{a}//\mathbf{t})$ for some constant **a** such that $\text{den}_{\mathbf{I}}(\mathbf{a}) = \text{den}_{\mathbf{I}}(\mathbf{t})$,[4] where **I** is the interpretation that has just been defined for the Hintikka set.
>
> **Proof:** We shall prove this using mathematical induction on the *complexity* of **t**, which is defined recursively as follows:
>
> - If **t** is $f(\mathbf{a}_1, \ldots, \mathbf{a}_n)$, where each $\mathbf{a}_i$ is a constant, the complexity of **t** is 1.

[4]Because all of the terms mentioned in this proof are closed terms, we omit reference to a variable assignment **d** when referring to the denotation of a term. We do so because the denotation in these cases is independent of any particular variable assignment.

- If **t** is $f(\mathbf{t}_1, \ldots, \mathbf{t}_n)$, where some $\mathbf{t}_i$ is not a constant, the complexity of **t** is 1 greater than the maximum complexity of the terms $\mathbf{t}_1, \ldots, \mathbf{t}_n$.

So 'f(a, b)' has complexity 1, '$f(g$(a), g(b))' has complexity 2, and '$f(g(h$(a)), g(b))' has complexity 3.

Basis clause: 11.6.6 holds for every closed complex term **t** of complexity 1.

Proof of basis clause: In this case the Hintikka set contains an identity sentence $\mathbf{a} = \mathbf{t}$, where **a** is a constant, by property (q) of Hintikka sets, and so, by property (p), it follows that each sentence $\mathbf{P}(\mathbf{a}//\mathbf{t})$ is also a member of Γ. In addition, $\text{den}_{\mathbf{I}}(\mathbf{t}) = \text{den}_{\mathbf{I}}(\mathbf{a})$ by the way that $\mathbf{I}(f)$ is defined in clause 5 of the definition of **I**, since the Hintikka set contains the identity sentence $\mathbf{a} = \mathbf{t}$.

Inductive step: If 11.6.6 holds for every closed complex term **t** of complexity **k** or less, then 11.6.6 holds for every closed complex term **t** of complexity $\mathbf{k} + 1$.

Proof of inductive step: We assume that the inductive hypothesis holds—that is, that 11.6.6 is true of every closed complex term of complexity **k** or less. We must show that it follows that 11.6.6 also holds of every closed complex term of complexity $\mathbf{k} + 1$. Let **t** be a closed complex term of complexity $\mathbf{k} + 1$. Then **t** is $f(\mathbf{t}_1, \ldots, \mathbf{t}_n)$, where each $\mathbf{t}_i$ is a complex term of complexity **k** or less. It follows, by the inductive hypothesis, that for some constants $\mathbf{a}_1, \ldots, \mathbf{a}_n$ such that $\text{den}_{\mathbf{I}}(\mathbf{a}_i) = \text{den}_{\mathbf{I}}(\mathbf{t}_i)$ for each **i**, each formula that results from replacing one or more occurrences of $f(\mathbf{t}_1, \ldots, \mathbf{t}_n)$ in **P** with $f(\mathbf{a}_1, \ldots, \mathbf{a}_n)$ is a member of Γ. By property (q) of Hintikka sets, there is a constant **a** such that $\mathbf{a} = f(\mathbf{a}_1, \ldots, \mathbf{a}_n) \in \Gamma$, and so, by property (p) of *PLE* Hintikka sets, each sentence $\mathbf{P}(\mathbf{a}//\mathbf{t})$ is also a member of Γ. Moreover, because $\mathbf{a} = f(\mathbf{a}_1, \ldots, \mathbf{a}_n) \in \Gamma$, it follows from the definition of $\mathbf{I}(f)$ in clause 5 that $\text{den}_{\mathbf{I}}(\mathbf{a}) = \text{den}_{\mathbf{I}}(f(\mathbf{a}_1, \ldots, \mathbf{a}_n))$, and because $\text{den}_{\mathbf{I}}(\mathbf{a}_i) = \text{den}_{\mathbf{I}}(\mathbf{t}_i)$ for each $\mathbf{a}_i$ and $\mathbf{t}_i$, it follows that $\text{den}_{\mathbf{I}}(f(\mathbf{a}_1, \ldots, \mathbf{a}_n)) = \text{den}_{\mathbf{I}}(f(\mathbf{t}_1, \ldots, \mathbf{t}_n))$; so $\text{den}_{\mathbf{I}}(\mathbf{a}) = \text{den}_{\mathbf{I}}(f(\mathbf{t}_1, \ldots, \mathbf{t}_n))$.

The proof of the basis clause of the inductive proof in 11.6.4 that every member of a Hintikka set Γ is true on the interpretation **I** that we have just defined is changed as follows:

Proof of basis clause: A sentence of length 1 is an atomic sentence. If **P** is a sentence letter, then by part 2 of the definition of **I**, $\mathbf{I}(\mathbf{P}) = \mathbf{T}$ if $\mathbf{P} \in \Gamma$.

If **P** is an atomic sentence of the form $\mathbf{A}\mathbf{t}_1 \ldots \mathbf{t}_n$, where **A** is not the identity predicate, then it follows from 11.6.6 that Γ also contains a sentence $\mathbf{A}\mathbf{a}_1 \ldots \mathbf{a}_n$ such that each $\mathbf{a}_i$ is a constant and $\text{den}_{\mathbf{I}}(\mathbf{t}_i) = \text{den}_{\mathbf{I}}(\mathbf{a}_i)$. By part 4 of the definition of **I**, if $\mathbf{A}\mathbf{a}_1 \ldots \mathbf{a}_n \in \Gamma$, then $\langle \mathbf{I}(\mathbf{a}_1), \ldots, \mathbf{I}(\mathbf{a}_n) \rangle \in \mathbf{I}(\mathbf{A})$, and so $\mathbf{I}(\mathbf{A}\mathbf{a}_1 \ldots \mathbf{a}_n) = \mathbf{I}(\mathbf{A}\mathbf{t}_1 \ldots \mathbf{t}_n) = \mathbf{T}$.

If $\mathbf{P}$ is a sentence of the form $\mathbf{t}_1 = \mathbf{t}_2$, then it follows from 11.6.6 that Γ also contains a sentence $\mathbf{a}_1 = \mathbf{a}_2$ such that $\mathbf{a}_1$ and $\mathbf{a}_2$ are constants and $\text{den}_{\mathbf{I}}(\mathbf{a}_1) = \mathbf{q}(\mathbf{a}_1) = \text{den}_{\mathbf{I}}(\mathbf{t}_1)$ and $\text{den}_{\mathbf{I}}(\mathbf{a}_2) = \mathbf{q}(\mathbf{a}_2) = \text{den}_{\mathbf{I}}(\mathbf{t}_2)$.

- Since $\mathbf{a}_1 = \mathbf{a}_2 \in \Gamma$, it follows, by property (p) of *PLE* Hintikka sets, that $\mathbf{a}_1 = \mathbf{a}_1 \in \Gamma$.
- Now, let $\mathbf{q}(\mathbf{a}_1)$ be $\mathbf{p}(\mathbf{b})$. Because $\mathbf{a}_1 = \mathbf{a}_1 \in \Gamma$, it follows by the way that $\mathbf{q}$ was defined that $\mathbf{b} = \mathbf{a}_1 \in \Gamma$ and that $\mathbf{b}$ must be alphabetically earlier than or identical to $\mathbf{a}_1$.
- It also follows by property (p) that $\mathbf{b} = \mathbf{a}_2 \in \Gamma$.
- Let $\mathbf{q}(\mathbf{a}_2)$ be $\mathbf{p}(\mathbf{c})$. Since $\mathbf{a}_1 = \mathbf{a}_2 \in \Gamma$, it follows that $\mathbf{c}$ is the alphabetically earliest constant such that $\mathbf{c} = \mathbf{a}_2 \in \Gamma$.
- Now, $\mathbf{b}$ cannot be alphabetically earlier than $\mathbf{c}$, since $\mathbf{b} = \mathbf{a}_2 \in \Gamma$.
- Further, by property (p), $\mathbf{c} = \mathbf{b} \in \Gamma$. Therefore $\mathbf{c} = \mathbf{a}_1 \in \Gamma$,
- and so $\mathbf{c}$ cannot be alphabetically earlier than $\mathbf{b}$ since $\mathbf{q}(\mathbf{a}_1)$ is $\mathbf{p}(\mathbf{b})$.
- We conclude that because neither is alphabetically earlier than the other, $\mathbf{b}$ and $\mathbf{c}$ are the same constant, and therefore $\mathbf{q}(\mathbf{a}_1) = \mathbf{q}(\mathbf{a}_2)$.
- Consequently $\text{den}_{\mathbf{I}}(\mathbf{t}_1) = \text{den}_{\mathbf{I}}(\mathbf{a}_1) = \mathbf{q}(\mathbf{a}_1) = \mathbf{q}(\mathbf{a}_2) = \text{den}_{\mathbf{I}}(\mathbf{a}_2) = \text{den}_{\mathbf{I}}(\mathbf{t}_2)$.

We conclude that $\mathbf{I}(\mathbf{t}_1 = \mathbf{t}_2) = \mathbf{T}$.

We must also change the proof of Case 1 in the inductive step:

Case 1: $\mathbf{P}$ has the form $\sim \mathbf{Q}$, where $\mathbf{Q}$ is an atomic sentence.
If $\mathbf{Q}$ is a sentence letter, then if $\sim \mathbf{Q} \in \Gamma$ it follows from part 2 of the definition of $\mathbf{I}$ that $\mathbf{I}(\mathbf{Q}) = \mathbf{F}$ since by property (a) of Hintikka sets $\mathbf{Q} \notin \Gamma$. Therefore $\mathbf{I}(\sim \mathbf{Q}) = \mathbf{T}$.

If $\mathbf{Q}$ has the form $\mathbf{A}\mathbf{t}_1 \ldots \mathbf{t}_n$, where $\mathbf{A}$ is not the identity predicate and $\sim \mathbf{Q} \in \Gamma$, then, by 11.6.6, there is a formula $\sim \mathbf{A}\mathbf{a}_1 \ldots \mathbf{a}_n$ in Γ in which every complex term $\mathbf{t}_i$ occurring in $\sim \mathbf{Q}$ has been replaced by a constant $\mathbf{a}_i$ such that $\text{den}_{\mathbf{I}}(\mathbf{t}_i) = \text{den}_{\mathbf{I}}(\mathbf{a}_i)$. By property (a) of Hintikka sets, $\mathbf{A}\mathbf{a}_1 \ldots \mathbf{a}_n \notin \Gamma$. It follows from part 4 of the definition of $\mathbf{I}$ that $<\mathbf{I}(\mathbf{a}_1), \ldots, \mathbf{I}(\mathbf{a}_n)> \notin \mathbf{I}(\mathbf{A})$. For consider:

- If $<\mathbf{I}(\mathbf{a}_1), \ldots, \mathbf{I}(\mathbf{a}_n)> \in \mathbf{I}(\mathbf{A})$, then there must be constants $\mathbf{a}_1', \ldots, \mathbf{a}_n'$ such that $\mathbf{A}\mathbf{a}_1' \ldots \mathbf{a}_n' \in \Gamma$ and $\mathbf{I}(\mathbf{a}_i) = \mathbf{I}(\mathbf{a}_i')$ for each $\mathbf{a}_i$.
- Because each of these constants occurs in a member of Γ, it follows from clause 3 of the definition of $\mathbf{I}$ that for each of these constants $\mathbf{a}_i$, $\mathbf{I}(\mathbf{a}_i) = \mathbf{q}(\mathbf{a}_i)$ and $\mathbf{I}(\mathbf{a}_i') = \mathbf{q}(\mathbf{a}_i')$ and thus $\mathbf{q}(\mathbf{a}_i) = \mathbf{q}(\mathbf{a}_i')$.

- From the last equation and the way that $\mathbf{q}$ is defined, it follows for each of these pairs of constants that either
  - $\mathbf{a_i} = \mathbf{a_i'} \in \Gamma$, or
  - $\mathbf{a_i'} = \mathbf{a_i} \in \Gamma$, or
  - there is a constant $\mathbf{c_i}$ such that both $\mathbf{c_i} = \mathbf{a_i}$ and $\mathbf{ci} = \mathbf{a_i'}$ are members of Γ.
- We now perform the following substitutions in the sentences $\sim \mathbf{A}\mathbf{a}_1 \ldots \mathbf{a_n}$ and $\mathbf{A}\mathbf{a}_1' \ldots \mathbf{a_n'}$.
  - For each $\mathbf{a_i}$ and $\mathbf{a_i'}$,
  - If (a) holds, then replace $\mathbf{a_i'}$ with $\mathbf{a_i}$ in $\mathbf{A}\mathbf{a}_1' \ldots \mathbf{a_n'}$.
  - If (b) holds, then replace $\mathbf{a_i}$ with $\mathbf{a_i'}$ in $\sim \mathbf{A}\mathbf{a}_1 \ldots \mathbf{a_n}$.
  - If (c) holds, then replace $\mathbf{a_i}$ with $\mathbf{c_i}$ in $\sim \mathbf{A}\mathbf{a}_1 \ldots \mathbf{a_n}$ and replace $\mathbf{a_i'}$ with $\mathbf{c_i}$ in $\mathbf{A}\mathbf{a}_1' \ldots \mathbf{a_i'}$.
- Note that each replacement generates a sentence that is also a member of Γ by property (p) of Hintikka sets, and that at the end of the replacements, we shall have two literal sentences, one of which is the negation of the other and both of which are members of Γ.
- But this is impossible because of property (a) of Hintikka sets.

We may conclude that $<\mathbf{I}(\mathbf{a}_1), \ldots, \mathbf{I}(\mathbf{a_n})> \notin \mathbf{I}(\mathbf{A})$, and it follows that $\mathbf{I}(\sim \mathbf{A}\mathbf{a}_1 \ldots \mathbf{a_n}) = \mathbf{T}$.

Consequently, because $\text{den}_{\mathbf{I}}(\mathbf{t_i}) = \text{den}_{\mathbf{I}}(\mathbf{a_i})$ for each $\mathbf{I}$, it also follows that $\mathbf{I}(\sim \mathbf{A}\mathbf{t}_1 \ldots \mathbf{t_n})$ (that is, $\mathbf{I}(\sim \mathbf{Q})) = \mathbf{T}$.

If $\mathbf{Q}$ has the form $\mathbf{t}_1 = \mathbf{t}_2$ and $\sim \mathbf{Q} \in \Gamma$ then, by 11.6.6, there is a formula $\sim \mathbf{a}_1 = \mathbf{a}_2$ in Γ such that $\text{den}_{\mathbf{I}}(\mathbf{t_i}) = \text{den}_{\mathbf{I}}(\mathbf{a_i})$.

- By property (a) of Hintikka sets, $\mathbf{a}_1 = \mathbf{a}_2 \notin \Gamma$.
- Now, if $\mathbf{q}(\mathbf{a}_1) = \mathbf{q}(\mathbf{a}_2)$, then either

  (a) $\mathbf{a}_1 = \mathbf{a}_2 \in \Gamma$, or
  (b) $\mathbf{a}_2 = \mathbf{a}_1 \in \Gamma$, or
  (c) there is a constant $\mathbf{b}$ such that $\mathbf{b} = \mathbf{a}_1$ and $\mathbf{b} = \mathbf{a}_2$ are both members of Γ.
- We have already shown that (a) does not hold.
- Nor does (b) hold, because if it did then, by property (p) of Hintikka sets, $\sim \mathbf{a}_2 = \mathbf{a}_2$ would be a member of Γ since $\sim \mathbf{a}_1 = \mathbf{a}_2$ is, but that is impossible by property (o) of Hintikka sets.
- Nor does (c) hold, because if did then, by property (p) of Hintikka sets, $\sim \mathbf{b} = \mathbf{b}$ would be a member of Γ since $\sim \mathbf{a}1 = \mathbf{a}2$ is, but that is also impossible by property (o) of Hintikka sets.

- Therefore, $\mathbf{q}(\mathbf{a}_1) \neq \mathbf{q}(\mathbf{a}_2)$, so $\mathbf{I}(\mathbf{a}_1 = \mathbf{a}_2) = \mathbf{F}$ and $\mathbf{I}(\sim \mathbf{a}_1 = \mathbf{a}_2) = \mathbf{T}$.

Because $\text{den}_{\mathbf{I}}(\mathbf{t_i}) = \text{den}_{\mathbf{I}}(\mathbf{a_i})$, we conclude that $\mathbf{I}(\sim \mathbf{t}_1 = \mathbf{t}_2) = \mathbf{T}$.

11.6**E** EXERCISES

1. Using Metatheorem 11.6.1, prove the following:

a. If **P** is quantificationally false, then every systematic tree for {**P**} closes.

b. If **P** is quantificationally true, then every systematic tree for {~ **P**} closes.

*c. If **P** and **Q** are quantificationally equivalent, then every systematic tree for {~ (**P** ≡ **Q**)} closes.

d. If $\Gamma \models \mathbf{P}$, where Γ is finite, then every systematic tree for $\Gamma \cup \{\sim \mathbf{P}\}$ closes.

*e. If an argument of *PL* is quantificationally valid, then every systematic tree for the set consisting of the premises and the negation of the conclusion of that argument closes.

2.a. What is the length of each of the following sentences?

$(\forall y)\text{Wy} \supset \sim (\forall y)\text{Bya}$
$(\exists x)\text{Sxbc}$
$(\forall x)(\text{Mx} \equiv \sim (\exists y)\text{My})$

b. Show that the length of a sentence ~ (**Q** & **R**) is greater than the length of ~ **Q** and greater than the length of ~ **R**.

*c. Show that the length of a sentence **Q** ≡ **R** is greater than the length of ~ **Q** and greater than the length of ~ **R**.

d. Show that the length of a sentence ~ $(\forall \mathbf{x})\mathbf{Q}$ is greater than the length of ~ $\mathbf{Q}(\mathbf{a}/\mathbf{x})$.

3. Complete the following clauses in the inductive proof of the completeness of the tree method.

a. 5
*b. 6
c. 7
*d. 8
*e. 9
f. 10
g. 13
*h. 14

***4.** Prove result 11.6.5.

5. Which clauses in the inductive proof of the completeness of the tree method would have broken down if our induction had been on the number of occurrences of logical operators in the sentences of *PL*?

6. If the rule ∃D were not included in our tree rules, where would the proof of Metatheorem 11.6.1 break down?

7. Suppose that our rule ~ ∀D were replaced by the following rule:

*Negated Universal Decomposition** (~ ∀D*)

~ (∀**x**)**P**

~ **P**(**a**/**x**)

where **a** is a constant foreign to all preceding lines of the tree.

Would the resulting system be complete for predicate logic? Explain.

8. Explain how we can adapt the proof of Metatheorem 11.6.1 to establish that the tree method for *SL* is complete for sentential logic.

9. Prove that every set of *PL* that is both maximally consistent and ∃-complete (as defined in Section 11.4) is a Hintikka set. Prove that every Hintikka set is ∃-complete. Prove that some Hintikka sets are not maximally consistent in *PD*.

Appendix 1

SOME FACTS ABOUT THE POSITIVE INTEGERS

In this appendix, we review some simple facts about the positive integers. Some readers will find it useful to consult this appendix as a refresher prior to symbolizing English sentences about the positive integers (Chapter 7), while others will find the appendix useful as a source of ideas for constructing interpretations using the set of positive integers—or a subset thereof—as the UD (Chapter 8).

The positive integers are the whole numbers 1, 2, 3, ... There are infinitely many positive integers.

- The smallest positive integer is 1.
- Every positive integer has a successor (which is also a positive integer), where the **successor** of an integer **n** is defined to be **n** + 1. As a consequence,
  - There is no greatest positive integer.
  - For any positive integer **n**, there are infinitely many positive integers that are greater than **n**.
- Every positive integer *other than 1* is the successor of a positive integer. *1 is the successor of 0, and 0 is not a positive integer.*

- The greater-than relationship is transitive. That is, for any positive integers **m**, **n**, and **p**: if **m** is greater than **n** and **n** is greater than **p**, then **m** is greater than **p**.
- A positive integer is **even** if it is evenly divisible by 2, that is, if it is divisible by 2 without remainder. Positive integers that are not evenly divisible by 2 are **odd** positive integers.
- The sum of any two positive integers is also a positive integer and is greater than each of those integers.
- The difference between any two positive integers is less than each of those integers but *is not always* a *positive integer*. For example, while $5 - 3$ is 2, a positive integer, *$3 - 3$ is 0 and $3 - 5$ is -2, a negative integer*. More generally, the difference between two positive integers is a positive integer if and only if the integer that is subtracted is less than the integer from which it is subtracted.
- The sum of two positive integers is even if and only if those integers are both even or both odd. Put another way, the sum of two positive integers is odd if and only if one of the two integers is even and the other is odd.
- More generally, the sum of any **n** positive integers is even if and only if those **n** positive integers include either no odd integers or an even number of odd integers (e.g., $2 + 4 + 8 = 14$ and $2 + 3 + 3 + 4 = 12$, while $3 + 5 + 9 = 17$ and $2 + 3 + 4 + 5 + 7 = 21$). Put another way, the sum of any **n** positive integers is odd if and only if those **n** positive integers include an odd number of odd integers.
- The product of any two positive integers (that is, the result of multiplying them) is a positive integer, and the product is even if and only if at least one of the two positive integers is even. More specifically,
  - The product of two odd positive integers is an odd positive integer.
  - The product of an even positive integer and any other positive integer, even or odd, is an even positive integer.
- More generally, the product of **n** positive integers is even if and only if at least one of those **n** integers is even (e.g., $3 \times 2 \times 5 = 30$, while $3 \times 3 \times 5 = 45$). Put another way, the product of any **n** positive integers is odd if and only if all of those **n** integers are odd.
- A **prime** number is a positive integer that is evenly divisible by exactly two positive integers: 1 and itself. As a consequence,
  - 1 is not a prime number.
  - 2, 3, 5, 7, and 11 are all prime numbers (and are the five smallest prime numbers).
- 2 is the only even prime number (since every other even positive integer is evenly divisible by at least 3 positive integers: 1, 2, and itself).

- The product of any two prime numbers is not prime (because the result is evenly divisible by those two primes as well as by 1).
- The sum of prime numbers is sometimes a prime number when one of the primes is 2 (e.g., $2 + 3 = 5$ and $2 + 5 = 7$), but not always (e.g., $2 + 2 = 4$ and $2 + 7 = 9$). The sum of two odd prime numbers is not a prime number since the sum in this case is even and greater than 2.
- There are infinitely many prime numbers, so there is no greatest prime number.

SELECTED BIBLIOGRAPHY

The following books are suggested for further reading.

INFORMAL LOGIC

Fogelin, Robert J., and Walter Sinnott-Armstrong. *Understanding Arguments,* 8th ed. Belmont, Calif.: Wadsworth/Cengage, 2009.

ELEMENTARY LOGIC

Jeffrey, Richard C. *Formal Logic: Its Scope and Limits,* 4th ed. Edited and forward by John C. Burgess. Indianapolis: Hackett, 2006.

Leblanc, Hugues, and William Wisdom. *Deductive Logic,* 3rd ed. Englewood Cliffs, N.J.: Prentice-Hall, 1993.

Quine, W. V. O. *Methods of Logic,* 4th ed. Cambridge, Mass.: Harvard University Press, 1989.

INDUCTIVE LOGIC

Skyrms, Brian. *Choice and Chance,* 4th ed. Belmont, Calif.: Wadsworth/Thomson Learning, 1999.

ADVANCED LOGIC

Hunter, Geoffrey. *Metalogic: An Introduction to the Metatheory of Standard First Order Logic*. Paperback ed. Berkeley: University of California Press, 1996.

Kleene, Stephen Cole. *Introduction to Metamathematics*. Paperback ed. Foreword by Michael Beeson. New York: Ishi Press, 2009.

Mendelson, Elliot. *Introduction to Mathematical Logic,* 5th ed. Boca Raton, Fla.: Taylor & Francis/CRC, 1997.

Quine, W. V. O. *Mathematical Logic,* rev. ed. Cambridge, Mass.: Harvard University Press, 1981.

Smullyan, Raymond M. *First-Order Logic*. New York: Dover, 1995.

Smullyan, Raymond M. *Gödel's Incompleteness Theorems*. New York: Oxford University Press, 1992.

ALTERNATIVE LOGICS

Bergmann, Merrie. *An Introduction to Many-Valued and Modal Logic: Semantics, Algebras, and Derivation Systems*. New York: Cambridge University Press, 2007.

Gottwald, Siegfried. *A Treatise on Many-Valued Logics*. Baldock, Hertfordshire, England: Research Studies Press, 2001.

Hughes, G. E., and M. J. Cresswell. *A New Introduction to Modal Logic*. London: Routledge, Chapman & Hall, 1996.

Rescher, N. *Many-Valued Logic*. Brookfield, Vt.: Ashgate, 1993.

HISTORY OF LOGIC

Bochenski, I. M. *A History of Formal Logic*. Translated and edited by Ivo Thomas. New York: Chelsea, 1970.

Kneale, William, and Martha Kneale. *The Development of Logic*. Oxford: Clarendon, 1985.

PHILOSOPHY OF LOGIC

Gabbay, D., and Guenthner, F., eds. *Handbook of Philosophical Logic*, 2nd ed. Dordrecht, Holland: Kluwer Academic, 2002.

Haack, Susan. *Philosophy of Logics*. New York: Cambridge University Press, 1978.

Quine, W. V. O. *The Philosophy of Logic*. Cambridge, Mass.: Harvard University Press, 1986.

INDEX

accessible sentence, 157
algorithm, 237
ampersand (&), 17, 28
antecedent, 20
argument, 4, 98
Aristotelean logic, 2
Aristotle, 2
Association (Assoc), 218, 222
Assumption, 148
atomic formulas, 269
atomic sentences of *SL*, 20
auxilliary assumptions, 157
axiomatic system, 2

basis clause, 231
Bergmann, Merrie, 459*n*8
Bernays and Schönfinkel, 357*n*2, 379
Biconditional Decomposition (≡D), 117
Biconditional Elimination (≡E), 149, 150, 167
Biconditional Introduction (≡I), 159, 160, 167
binary connective, 18
bound variable, 271
branching rules, 116–117

causal claims, 59, 64
characteristic sentence, 237
characteristic truth-tables, 27
check mark, 114
Church, Alonzo, 356, 357*n*2, 365
closed branch (of a truth-tree), 120, 443
closed subderivation, 157
closed terms, 322
closed truth-tree, 120, 145
Commutation (Com), 218, 222
Compactness Theorem for sentential logic, 260
completed open branch (of a truth-tree), 116, 120, 411, 444
completed truth-tree, 120, 413
completeness, 168
  of deduction systems for predicate logic, 566
  of PD, 566
  of PD+, 576
  of PDE, 583
  of SD/SD+, 252
  of the tree method, 596, 602
  truth-functional, 236
complex term, 322
Complex Term Decomposition (CTD), 459
complex terms, 322
components of sentences of *SL*, 20, 22
compound sentences of *SL*, 20
conclusion, 4
conclusion indicator expressions, 6
Conditional Decomposition (⊃D), 117
Conditional Elimination (⊃E), 149, 150, 167
conjunct, 20
conjunction (&), 27–28
Conjunction Decomposition (&D), 113
Conjunction Elimination (&E), 149, 167
Conjunction Introduction (&I), 149, 150, 167
consequent, 20

consistency
  logical, 10,14
  maximal, in *PD*, 567
  maximal, in *SD*, 254
  quantificational, 361
  truth-functional, 93
Consistent Branch Lemma, 587
contracditory literals, 115
corresponding material biconditional, 106
corresponding material conditional, 100

De Morgan (DeM), 218, 222
decision procedure, 356
deductive logic, 1
definite descriptions
  of English, 264
  symbolizing in *PL*, 315–316
  symbolizing in *PLE*, 316–317
denotation of a term, 340
derivation, 168
  in *PD*, 474–518
  in *PD*+ 521–524
  in *PDE*, 526–540
  in *SD*, 146–209
  in *SD*+, 214–222
discharged assumption, 157
disjunct, 20
disjunction (∨), 27–28
Disjunction Decomposition (∨D), 117
Disjunction Elimination (∨E), 159, 160, 167
Disjunction Introduction (∨I), 149, 150, 167
Disjunctive Syllogism (DS), 216, 221
Distribution (Dist), 218, 222
Double Negation (DN), 217, 218, 222

effective procedure, 237
entailment
  logical, 10
  quantificational, 363
  truth-functional, 95
enumeration, 254
equivalence
  in *PS*, 490
  in *SD*, 175
  logical, 9
  quantificational, 358
  truth-functional, 87
Euclid, 2
Euclidean plane geometry, 2
exclusive 'or', 33, 54
Existential Decomposition (∃D), 404
Existential Decomposition-2 (∃D-2), 427
Existential Elimination (∃E), 480
Existential Introduction (∃I), 476, 532
existential quantifier, 269
expansion, truth-functional, 369–373, 384
Exportation (Exp), 218, 222
expressions of *PL*, 269
expressions of *SL*, 17
extended conjunction, 39
extended disjunction, 39
extension of a predicate, 331, 333

falsity
  logical, 14
  quantificational, 351
  truth-functional, 78
finite model, 426
finite model property, 593
finite truth-tree, 426
formula of *PL*, 269
free variable, 271
Frege, Gottlob, 2
function, 387
functor, 319, 526

Hilbert, David, 2
Hintikka branch, 598
Hintikka Branch Lemma, 598
Hintikka set, 598
Hintikka, J., 596
horseshoe (⊃), 17, 28
Hypothetical Syllogism (HS), 216, 221

Idempotence (Idem), 218, 222
Identity Decomposition (=D), 442
Identity Elimination (=E), 526
Identity Introduction (=I), 526
identity predicate, 311
immediate component of sentence of *SL*, 20
immediate successor, 597
Implication (Imp), 218, 222
inconsistency
  logical, 10, 14
  quantificational, 361
  truth-functional, 93
  in *PD*, 490, 544
  in *SD*, 175, 225
indeterminacy
  logical, 9, 14
  quantificational, 351
  truth-functional, 79
individual constants of *PL*, 268
individual terms of *PL*, 268
individual terms of *PLE*, 322
individual variables of *PL*, 268

inductive hypothesis, 231
inductive step, 231
Infinite Branch Lemma, 597
instantiating constant, 475
instantiating term, 533
interpretation, 330, 332–333
invalidity
  logical, 5
  quantificational, 364
  truth-functional, 98
iterated conjunction, 100
iterated disjunction, 369

justification column in
    truth-trees, 114

König's Lemma, 597

level of a truth-tree, 585
literal, 111
logical
  consistency, 10, 14
  entailment, 10, 14
  equivalence, 9, 14
  falsity, 8, 14
  inconsistency, 10, 14
  indeterminatcy, 9, 14
  invalidity, 5, 14
  operators of *PL*, 270
  possibility, 10
  soundness, 5, 14
  truth, 8, 14
  validity, 5, 14
Löwenheim Theorem, 356
Löwenheim-Skolem Theorem, 366

main connective, 20
main logical operator of *PL*, 270
material biconditional, 20, 27–28
material conditional, 20, 27–28
mathematical induction, 227,
    228–229
maximal consistency in *PD*, 567
maximal consistency in *SD*, 254
Maximal Consistency Lemma, 254
mechanical procedure, 111, 179
metalanguage, 15–16
metavariables, 4, 17
Modus Tollens (MT), 215, 221

Negated Biconditional Decomposition
    (~ ≡D), 117
Negated Conditional Decomposition
    (~ ⊃D), 113
Negated Conjunction Decomposition
    (~ &D), 117
Negated Disjunction Decomposition
    (~ vD), 113
Negated Existential Decomposition
    (~∃D), 403
Negated Negation Decomposition
    (~ ~ D), 113
Negated Universal Decomposition
    (~∀D), 403
negation (~), 27–28
Negation Elimination (~ E), 159,
    160, 167
Negation Introduction (~ I), 159, 167
non-branching rules, 113
non-referring singular terms of
    English, 265
non-subderivation rules of *SD*, 149–150
nonterminating branch (of a
    truth-tree), 426
non-truth-functional compounds
    58–67
non-truth-functional connectives
    58–67
NP-complete problem, 84*n*4
n-place function, 387
n-tuple, 331

object language, 15–16
objectual semantics, 337*n*1
only if, 34
open assumption, 157
open branch (of a truth-tree)
    116, 120, 411
open terms of *PLE*, 322
open truth-tree, 120, 413
outermost parentheses, 23

paraphrasing (sentences of English)
    29–36
path of a truth-tree, 587
path-variant, 588
Peano, Giuseppe, 2
predicates of English, 264–267
predicates of *PL*, 268
premise, 2
premise indicator expresions, 6
primary assumptions, 156–157
pronouns of English, 264
proof of a theorem, 177
proper names of English, 264
properties of relations, 317–318
punctuation marks of *PL*, 268
punctuation marks of *SL*, 18

quantifers of *PL*, 268
quantificational
  consistency, 361, 414
  entailment, 363, 417
  equivalence, 358, 417
  falsity, 351, 416
  inconsistency, 361, 414
  indeterminacy, 351, 416
  invalidity, 364
  truth, 351, 416
  validity, 364, 417
Quantifier Negation (QN), 522
quantifier rules, 535
quantity terms of English, 265–267

recursive definition, 20
referential position, 265
referential semantics, 337*n*1
reflexive relations, 318
Reiteration (R), 148, 149, 167
Russell, Bertrand, 2

satisfaction, 340–341
satisfaction semantics, 337*n*1
scope lines, 156
scope of a quantifier, 271
semantics, 15, 69, 329
sentence letters of *PL*, 268
sentence letters of *SL*, 17
sentence of *PL*, 271
sentential connectives of *SL*, 17
set, 4
simple individual terms of *PLE*, 322
single turnstyle (⊢), 175
singular terms of English, 262–267
SM, 114
soundness, 168
  of deduction systems for predicate logic, 561
  of *PD*, 561
  of *PD*+ 564
  of *PDE*, 564
  of *SD*, 224
  of the tree method, 585,595
square brackets, 23
standard form (of an argument), 4, 37
stronger than/weaker than, 63
subderivation rules of *SD*, 155–160
subderivations, 148
subformula of *PL*, 270
subjunctive conditional, 60, 66–67
subset, 245
substitution instance, 475
substitution semantics, 337*n*1
superset, 245
syllogism, 1
syllogistic logic, 2
symbolization key, 25
symmetric relations, 317
syntax, 15
syntax of *PL*, 268–274
syntax of *SL*, 15–23
System for *PL*, 434
System for *PLE*, 466–467
systematic tree, 434

Tarski, Alfred, 337*n*1
theorem in *PD*, 490, 544
theorem in *SD*, 175, 225
tilde (~), 17, 28
transitive relations, 317
Transposition (Trans), 218, 222
triple bar (≡), 17, 28
truth
  quantificational, 351
  truth-functional, 77
truth-function, 235
truth-function schema, 236
truth-functional completeness, 236
truth-functional connectives
  ampersand (&), 17, 28
  horseshoe (⊃), 17, 28
  tilde (~), 17, 28
  triple bar (≡), 17, 28
  wedge (∨), 17, 28
truth-functional
  consistency, 93, 120, 144
  entailment, 95, 137, 145
  equivalence, 87, 134, 144
  falsity, 78, 129, 144
  inconsistency, 93, 144
  indalidity, 98
  indeterminacy 79, 131, 144
  truth, 77, 131, 144
  validity 98, 138, 145
truth-functional connectives of *PL*, 268
truth-functional expansion, 369–373, 384
truth-functional use of a connective, 26, 68
truth-tree branch
  closed, 120, 145, 443
  completed open, 116, 120
  open, 116, 120, 145, 411
  recovering interpretations from, 406
  recovering truth-value assignments from, 116
truth-value, 3
truth-value assignment, 70

UD, 276, 332
unary connective, 18
unit set, 105

Universal Decomposition (∀D), 403
Universal Elimination (∀E), 475, 533
Universal Introduction (∀I), 477
universal quantifier, 269
universe of discourse, 276
unless, 32, 54
use/mention distinction, 16

validity
truth-functional
  logical, 5
  quantificational, 364
  truth-functional, 98
  in *PD*, 490, 544
  in *SD*, 175, 255
variable, 268
variable assignment, 337
variant of a variable assignment, 338
vocabulary of *SL*, 17–18
wedge (∨), 17, 28

INDEX OF SYMBOLS

~	17	tilde
&	17	ampersand
∨	17	wedge
⊃	17	horseshoe
≡	17	triple bar
|	243	stroke
↓	243	dagger
∀	269	universal quantifier symbol
∃	269	existential quantifier symbol
=	311	identity sign
{ }	4	braces
∅	92	empty set
Γ	92	gamma
∪	107	set union
∈	257	set membership
∉	257	set membership denial
⟨ ⟩	331	angle brackets
⊢	175	single turnstile
⊬	175	single turnstile with slash
⊨	95	double turnstile
⊭	96	double turnstile with slash
▷	148	pointer
◁ ▷	217	double pointer

DERIVATION RULES OF *SD*

Reiteration (R)

	P
▷	**P**

'&' Rules

Conjunction Introduction (&I)

	P
	Q
▷	**P & Q**

Conjunction Elimination (&E)

	P & Q		**P & Q**
▷	**P**	▷	**Q**

or

'⊃' Rules

Conditional Introduction (⊃I)

	P	
	Q	
▷	**P ⊃ Q**	

Conditional Elimination (⊃E)

	P ⊃ Q
	P
▷	**Q**

'~' Rules

Negation Introduction (~ I)

	P	
	Q	
	~ Q	
▷	**~ P**	

Negation Elimination (~ E)

	~ P	
	Q	
	~ Q	
▷	**P**	

'∨' Rules

Disjunction Introduction (∨I)

	P			**P**
▷	**P ∨ Q**	or	▷	**Q ∨ P**

Disjunction Elimination (∨E)

	P ∨ Q		
		P	
		R	
		Q	
		R	
▷	**R**		

'≡' Rules

Biconditional Introduction (≡I)

	P		
	Q		
	Q		
	P		
▷	**P ≡ Q**		

Biconditional Elimination (≡E)

	P ≡ Q			**P ≡ Q**
	P	or		**Q**
▷	**Q**		▷	**P**

DERIVATION RULES OF *SD*+

All the Derivation Rules of *SD* and Rules of Inference

Modus Tollens (MT)

$$\begin{array}{l|l} & \mathbf{P} \supset \mathbf{Q} \\ & \sim \mathbf{Q} \\ \triangleright & \sim \mathbf{P} \end{array}$$

Hypothetical Syllogism (HS)

$$\begin{array}{l|l} & \mathbf{P} \supset \mathbf{Q} \\ & \mathbf{Q} \supset \mathbf{R} \\ \triangleright & \mathbf{P} \supset \mathbf{R} \end{array}$$

Disjunctive Syllogism (DS)

$$\begin{array}{l|l} & \mathbf{P} \vee \mathbf{Q} \\ & \sim \mathbf{P} \\ \triangleright & \mathbf{Q} \end{array} \quad \text{or} \quad \begin{array}{l|l} & \mathbf{P} \vee \mathbf{Q} \\ & \sim \mathbf{Q} \\ \triangleright & \mathbf{P} \end{array}$$

Rules of Replacement

Commutation (Com)

$\mathbf{P} \,\&\, \mathbf{Q} \dashv\vdash \mathbf{Q} \,\&\, \mathbf{P}$
$\mathbf{P} \vee \mathbf{Q} \dashv\vdash \mathbf{Q} \vee \mathbf{P}$

Association (Assoc)

$\mathbf{P} \,\&\, (\mathbf{Q} \,\&\, \mathbf{R}) \dashv\vdash (\mathbf{P} \,\&\, \mathbf{Q}) \,\&\, \mathbf{R}$
$\mathbf{P} \vee (\mathbf{Q} \vee \mathbf{R}) \dashv\vdash (\mathbf{P} \vee \mathbf{Q}) \vee \mathbf{R}$

Implication (Impl)

$\mathbf{P} \supset \mathbf{Q} \dashv\vdash \sim \mathbf{P} \vee \mathbf{Q}$

Double Negation (DN)

$\mathbf{P} \dashv\vdash \sim\sim \mathbf{P}$

De Morgan (DeM)

$\sim (\mathbf{P} \,\&\, \mathbf{Q}) \dashv\vdash \sim \mathbf{P} \vee \sim \mathbf{Q}$
$\sim (\mathbf{P} \vee \mathbf{Q}) \dashv\vdash \sim \mathbf{P} \,\&\, \sim \mathbf{Q}$

Idempotence (Idem)

$\mathbf{P} \dashv\vdash \mathbf{P} \,\&\, \mathbf{P}$
$\mathbf{P} \dashv\vdash \mathbf{P} \vee \mathbf{P}$

Transposition (Trans)

$\mathbf{P} \supset \mathbf{Q} \dashv\vdash \sim \mathbf{Q} \supset \sim \mathbf{P}$

Exportation (Exp)

$\mathbf{P} \supset (\mathbf{Q} \supset \mathbf{R}) \dashv\vdash (\mathbf{P} \,\&\, \mathbf{Q}) \supset \mathbf{R}$

Distribution (Dist)

$\mathbf{P} \,\&\, (\mathbf{Q} \vee \mathbf{R}) \dashv\vdash (\mathbf{P} \,\&\, \mathbf{Q}) \vee (\mathbf{P} \,\&\, \mathbf{R})$
$\mathbf{P} \vee (\mathbf{Q} \,\&\, \mathbf{R}) \dashv\vdash (\mathbf{P} \vee \mathbf{Q}) \,\&\, (\mathbf{P} \vee \mathbf{R})$

Equivalence (Equiv)

$\mathbf{P} \equiv \mathbf{Q} \dashv\vdash (\mathbf{P} \supset \mathbf{Q}) \,\&\, (\mathbf{Q} \supset \mathbf{P})$
$\mathbf{P} \equiv \mathbf{Q} \dashv\vdash (\mathbf{P} \,\&\, \mathbf{Q}) \vee (\sim \mathbf{P} \,\&\, \sim \mathbf{Q})$

DERIVATION RULES OF *PD*

All the Derivation Rules of *SD* and

Universal Introduction (∀I)

$\triangleright$ | $\mathbf{P}(\mathbf{a}/\mathbf{x})$
$\triangleright$ | $(\forall\mathbf{x})\mathbf{P}$

provided that:
(i) **a** does not occur in an open assumption.
(ii) **a** does not occur in $(\forall\mathbf{x})\mathbf{P}$.

Universal Elimination (∀E)

$\triangleright$ | $(\forall\mathbf{x})\mathbf{P}$
$\triangleright$ | $\mathbf{P}(\mathbf{a}/\mathbf{x})$

Existential Introduction (∃I)

$\triangleright$ | $\mathbf{P}(\mathbf{a}/\mathbf{x})$
$\triangleright$ | $(\exists\mathbf{x})\mathbf{P}$

Existential Elimination (∃E)

$\triangleright$ | $(\exists\mathbf{x})\mathbf{P}$
| | $\mathbf{P}(\mathbf{a}/\mathbf{x})$
| | ——
| | $\mathbf{Q}$
$\triangleright$ | $\mathbf{Q}$

provided that:
(i) **a** does not occur in an open assumption.
(ii) **a** does not occur in $(\exists\mathbf{x})\mathbf{P}$.
(iii) **a** does not occur in **Q**.

DERIVATION RULES OF *PD*+

All the Derivation Rules of *SD*+ and of *PD* and

Quantifier Negation (QN)

$\sim(\forall\mathbf{x})\mathbf{P} \dashv\vdash (\exists\mathbf{x}) \sim \mathbf{P}$
$\sim(\exists\mathbf{x})\mathbf{P} \dashv\vdash (\forall\mathbf{x}) \sim \mathbf{P}$

DERIVATION RULES OF *PDE*

All of the Derivation Rules of *PD* and

Universal Elimination (∀E)

$$\begin{array}{l|l} & (\forall \mathbf{x})\mathbf{P} \\ \triangleright & \mathbf{P}(\mathbf{t}/\mathbf{x}) \end{array}$$

where **t** is any closed term

Existential Introduction (∃I)

$$\begin{array}{l|l} & \mathbf{P}(\mathbf{t}/\mathbf{x}) \\ \triangleright & (\exists \mathbf{x})\mathbf{P} \end{array}$$

where **t** is any closed term

Identity Introduction (=I)

$$\triangleright \mid (\forall \mathbf{x})\mathbf{x} = \mathbf{x}$$

Identity Elimination (=E)

$$\begin{array}{l|l} & \mathbf{t}_1 = \mathbf{t}_2 \\ & \mathbf{P} \\ \triangleright & \mathbf{P}(\mathbf{t}_1//\mathbf{t}_2) \end{array} \qquad \text{or} \qquad \begin{array}{l|l} & \mathbf{t}_1 = \mathbf{t}_2 \\ & \mathbf{P} \\ \triangleright & \mathbf{P}(\mathbf{t}_2//\mathbf{t}_1) \end{array}$$

where $\mathbf{t}_1$ and $\mathbf{t}_2$ are closed terms

TRUTH-TREE RULES FOR *SL*

Negated Negation Decomposition (~ ~ D)

$$\sim\sim \mathbf{P}\checkmark$$
$$\mathbf{P}$$

Conjunction Decomposition (&D)

$$\mathbf{P} \,\&\, \mathbf{Q}\checkmark$$
$$\mathbf{P}$$
$$\mathbf{Q}$$

Negated Conjunction Decomposition (~ &D)

$$\sim (\mathbf{P} \,\&\, \mathbf{Q})\checkmark$$
$$\sim \mathbf{P} \qquad \sim \mathbf{Q}$$

Disjunction Decomposition (∨D)

$$\mathbf{P} \vee \mathbf{Q}\checkmark$$
$$\mathbf{P} \qquad \mathbf{Q}$$

Negated Disjunction Decomposition (~ ∨D)

$$\sim (\mathbf{P} \vee \mathbf{Q})\checkmark$$
$$\sim \mathbf{P}$$
$$\sim \mathbf{Q}$$

Conditional Decomposition (⊃D)

$$\mathbf{P} \supset \mathbf{Q}\checkmark$$
$$\sim \mathbf{P} \qquad \mathbf{Q}$$

Negated Conditional Decomposition (~ ⊃D)

$$\sim (\mathbf{P} \supset \mathbf{Q})\checkmark$$
$$\mathbf{P}$$
$$\sim \mathbf{Q}$$

Biconditional Decomposition (≡D)

$$\mathbf{P} \equiv \mathbf{Q}\checkmark$$
$$\mathbf{P} \qquad \sim \mathbf{P}$$

Negated Biconditional Decomposition (~ ≡D)

$$\sim (\mathbf{P} \equiv \mathbf{Q})\checkmark$$
$$\mathbf{P} \qquad \sim \mathbf{P}$$

TRUTH-TREE RULES FOR *PL*

Universal Decomposition (∀D)

$$(\forall \mathbf{x})\mathbf{P}$$
$$\mathbf{P}(\mathbf{a}/\mathbf{x})$$

Negated Universal Decomposition (~ ∀D)

$$\sim (\forall \mathbf{x})\mathbf{P}\checkmark$$
$$(\exists \mathbf{x}) \sim \mathbf{P}$$

Existential Decomposition (∃D)

$$(\exists \mathbf{x})\mathbf{P}\checkmark$$
$$\mathbf{P}(\mathbf{a}/\mathbf{x})$$

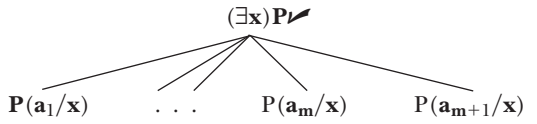
where **a** is a constant foreign to the branch on which $\mathbf{P}(\mathbf{a}/\mathbf{x})$ is entered.

Negated Existential Decomposition (~ ∃D)

$$\sim (\exists \mathbf{x})\mathbf{P}\checkmark$$
$$(\forall \mathbf{x}) \sim \mathbf{P}$$

ALTERNATE EXISTENTIAL DECOMPOSITION RULE

Existential Decomposition-2 (∃D2)



where $\mathbf{a}_1, \ldots, \mathbf{a}_m$ are the constants that already occur on the branch on which Existential Decomposition-2 is being applied to decompose $(\exists \mathbf{x})\mathbf{P}$ and $\mathbf{a}_{m+1}$ is a constant that is foreign to that branch.[1]

TRUTH-TREE RULES FOR *PLE*

All of the Predicate Truth-Tree Rules and

Universal Decomposition (∀D)

$(\forall \mathbf{x})\mathbf{P}$

$\mathbf{P}(\mathbf{t}/\mathbf{x})$

where $\mathbf{t}$ is a closed term

Identity Decomposition (=D)

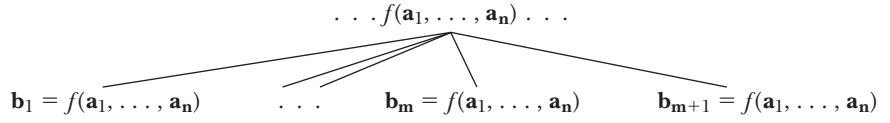
$\mathbf{t}_1 = \mathbf{t}_2$

$\mathbf{P}$

$\mathbf{P}(\mathbf{t}_1//\mathbf{t}_2)$

where $\mathbf{t}_1$ and $\mathbf{t}_2$ are closed individual terms and $\mathbf{P}$ is a literal containing $\mathbf{t}_2$

Complex Term Decomposition (CTD)



where $f(\mathbf{a}_1, \ldots, \mathbf{a}_n)$ is a closed complex term occurring within a literal on some branch, whose arguments $\mathbf{a}_1, \ldots, \mathbf{a}_n$ are individual constants; $\mathbf{b}_1, \ldots, \mathbf{b}_m$ are the constants that already occur on that branch, and $\mathbf{b}_{m+1}$ is a constant that is foreign to that branch.

[1]This Existential Decomposition rule is due to George Boolos, "Trees and Finite Satisfiability: Proof of a Conjecture of Burgess," *Notre Dame Journal of Formal Logic,* 25(3)(1984), 193–197.

The Logic Book is a leading text for symbolic logic courses that presents all concepts and techniques with clear, comprehensive explanations. There is a wealth of carefully constructed examples throughout the text, and its flexible organization places materials within largely self-contained chapters that allows instructors the freedom to cover the topics they want, in the order they choose.

Features of the 6th Edition:

- **New!** There is a fuller and more accessible discussion of formal semantics than in previous editions.
- **New!** A recovery of only extensions of predicates from truth-trees, rather than English readings of those predicates.
- **New!** Many chapters have been reorganized so that technical material is presented at the beginning of the chapter, which gives instructors the flexibility to cover the material quickly before proceeding to the following chapters.

What Instructors are Saying about *The Logic Book:*

"***The Logic Book*** is an ideal choice for an upper-division first or second course in symbolic logic. It contains not only all of the basic material covered in an introductory symbolic logic course, but also a full treatment of mathematical induction and the soundness and completeness of sentential and predicate logic."
-Charles Cross, University of Georgia

"A solid and functional logic textbook that is suitable for a variety of different introductory and intermediate logic courses..."
-Jamin Asay, University of North Carolina at Chapel Hill

"A precise, careful, fully formal approach to first-order logic."
-Arnold Smith, Kent State University

Visit the Online Learning Center for a wealth of student and instructor resources at www.mhhe.com/bergmann6.

The McGraw-Hill Companies

Mc Graw Hill Connect Learn Succeed™

ISBN 978-0-07-803841-9
MHID 0-07-803841-3

EAN 9 780078 038419 90000

www.mhhe.com