

MATHEMATICAL STRUCTURES FOR | COMPUTER SCIENCE

DISCRETE
MATHEMATICS
AND ITS
APPLICATIONS

SEVENTH EDITION

JUDITH L. GERSTING

Mathematical Structures for Computer Science

This page intentionally left blank

Edition

7

Mathematical Structures for Computer Science

DISCRETE MATHEMATICS AND ITS APPLICATIONS

Judith L. Gersting

Indiana University-Purdue University at Indianapolis



W. H. Freeman and Company

A Macmillan Higher Education Company

To my 0110_2 favorite
discrete structures:
(Adam \wedge Francine),
(Jason \wedge Cathryn) \rightarrow
(Sammie \wedge Johnny)

Senior Publisher: Ruth Baruth
Executive Editor: Terri Ward
Senior Editor: Roland Cheyney
Assistant Editor: Liam Ferguson
Marketing Manager: Steve Thomas
Media Editor: Laura Judge
Senior Project Editor: Georgia Lee Hadler
Copy Editor: Penelope Hull
Production Coordinator: Susan Wein
Text Designer: Patrice Sheridan
Cover Designer: Victoria Tomaselli
Illustrations: Network Graphics
Composition: codeMantra
Printing and Binding: RR Donnelley

Library of Congress Control Number: 2013951442

ISBN-13: 978-1-4292-1510-7
ISBN-10: 1-4292-1510-0

© 2014, 2007, 2003, 1999, 1993, 1987, 1982 by W. H. Freeman and Company
All rights reserved

Printed in the United States of America
First printing

W. H. Freeman and Company
41 Madison Avenue, New York, NY 10010
Houndmills, Basingstoke RG21 6XS, England
www.whfreeman.com

Contents in Brief

	Preface	xiii
	Note to the Student	xvi
CHAPTER 1	Formal Logic	001
CHAPTER 2	Proofs, Induction, and Number Theory	097
CHAPTER 3	Recursion, Recurrence Relations, and Analysis of Algorithms	157
CHAPTER 4	Sets, Combinatorics, and Probability	221
CHAPTER 5	Relations, Functions, and Matrices	327
CHAPTER 6	Graphs and Trees	475
CHAPTER 7	Graph Algorithms	553
CHAPTER 8	Boolean Algebra and Computer Logic	617
CHAPTER 9	Modeling Arithmetic, Computation, and Languages	685
APPENDIX A	Derivation Rules for Propositional and Predicate Logic	803
APPENDIX B	Summation and Product Notation	805
APPENDIX C	The Logarithm Function	809
	Answers to Practice Problems	813
	Answers to Odd-Numbered Exercises	851
	Answers to Self-Tests	949
	Index	959

This page intentionally left blank

Contents

CHAPTER 1	Formal Logic	1		
1.1	STATEMENTS, SYMBOLIC REPRESENTATION, AND TAUTOLOGIES	2		
	<i>Connectives and Truth Values</i>	2		
	<i>Tautologies</i>	8		
	<i>Logical Connectives in the Real World</i>	10		
	<i>An Algorithm</i>	12		
<hr/>				
	SPECIAL INTEREST PAGE			
	Can “And” Ever Be “Or”?	15		
<hr/>				
SECTION 1.1	Review	16		
EXERCISES 1.1		16		
1.2	PROPOSITIONAL LOGIC	25		
	<i>Valid Arguments</i>	25		
	<i>Derivation Rules for Propositional Logic</i>	28		
	<i>Deduction Method and Other Rules</i>	32		
	<i>Verbal Arguments</i>	33		
SECTION 1.2	Review	35		
EXERCISES 1.2		35		
1.3	QUANTIFIERS, PREDICATES, AND VALIDITY	39		
	<i>Quantifiers and Predicates</i>	39		
	<i>Translation</i>	42		
	<i>Validity</i>	48		
SECTION 1.3	Review	50		
EXERCISES 1.3		50		
1.4	PREDICATE LOGIC	58		
	<i>Derivation Rules for Predicate Logic</i>	58		
	Universal Instantiation	59		
	Existential Instantiation	60		
	Universal Generalization	61		
	Existential Generalization	62		
	<i>More Work with Rules</i>	62		
	<i>Verbal Arguments</i>	67		
	<i>Conclusion</i>	68		
SECTION 1.4	Review	69		
EXERCISES 1.4		69		
1.5	LOGIC PROGRAMMING	73		
	<i>Prolog</i>	73		
	<i>Horn Clauses and Resolution</i>	75		
	<i>Recursion</i>	79		
	<i>Expert Systems</i>	81		
SECTION 1.5	Review	82		
EXERCISES 1.5		82		
1.6	PROOF OF CORRECTNESS	84		
	<i>Assertions</i>	85		
	<i>Assignment Rule</i>	87		
	<i>Conditional Rule</i>	90		
SECTION 1.6	Review	92		
EXERCISES 1.6		92		
	Chapter 1 Review	95		
	On the Computer	96		
CHAPTER 2	Proofs, Induction, and Number Theory	97		
2.1	PROOF TECHNIQUES	98		
	<i>Theorems and Informal Proofs To Prove or Not to Prove</i>	98		
	<i>Exhaustive Proof</i>	100		
	<i>Direct Proof</i>	101		
	<i>Contraposition</i>	103		
	<i>Contradiction</i>	104		

<i>Serendipity</i>	106	3.2 RECURRENCE RELATIONS	180
<i>Common Definitions</i>	107	<i>Linear First-Order Recurrence Relations</i>	180
SECTION 2.1 Review	107	Expand, Guess, and Verify	180
EXERCISES 2.1	107	A Solution Formula	182
2.2 INDUCTION	110	<i>Linear Second-Order Recurrence Relations</i>	188
<i>First Principle of Induction</i>	110	<i>Divide-and-Conquer Recurrence Relations</i>	193
<i>Proofs by Mathematical Induction</i>	112	SECTION 3.2 Review	197
<i>Second Principle of Induction</i>	118	EXERCISES 3.2	197
SECTION 2.2 Review	122	3.3 ANALYSIS OF ALGORITHMS	203
EXERCISES 2.2	122	<i>The General Idea</i>	203
2.3 MORE ON PROOF OF CORRECTNESS	129	<i>Analysis Using Recurrence Relations</i>	206
<i>Loop Rule</i>	129	<i>Upper Bound (Euclidean Algorithm)</i>	210
<i>Euclidean Algorithm</i>	133	<hr/>	
SPECIAL INTEREST PAGE		SPECIAL INTEREST PAGE	
Making Safer Software	136	Of Trees ... and Pancakes	211
<hr/>		<hr/>	
SECTION 2.3 Review	137	SECTION 3.3 Review	212
EXERCISES 2.3	137	EXERCISES 3.3	212
2.4 NUMBER THEORY	143	Chapter 3 Review	217
<i>The Fundamental Theorem of Arithmetic</i>	144	On the Computer	218
<i>More on Prime Numbers</i>	148	CHAPTER 4 Sets, Combinatorics, and Probability	221
<i>Euler Phi Function</i>	149	4.1 SETS	222
SECTION 2.4 Review	152	<i>Notation</i>	222
EXERCISES 2.4	152	<i>Relationships Between Sets</i>	224
Chapter 2 Review	155	<i>Sets of Sets</i>	227
On the Computer	156	<i>Binary and Unary Operations</i>	228
CHAPTER 3 Recursion, Recurrence Relations, and Analysis of Algorithms	157	<i>Operations on Sets</i>	230
3.1 RECURSIVE DEFINITIONS	158	<i>Set Identities</i>	233
<i>Recursively Defined Sequences</i>	158	<i>Countable and Uncountable Sets</i>	236
<i>Recursively Defined Sets</i>	162	SECTION 4.1 Review	239
<i>Recursively Defined Operations</i>	165	EXERCISES 4.1	239
<i>Recursively Defined Algorithms</i>	166	4.2 COUNTING	252
SECTION 3.1 Review	171	<i>Multiplication Principle</i>	252
EXERCISES 3.1	171	<i>Addition Principle</i>	254
		<i>Using the Principles Together</i>	255
		<i>Decision Trees</i>	257

SECTION 4.2 Review	258	SECTION 4.6 Review	315
EXERCISES 4.2	259	EXERCISES 4.6	315
4.3 PRINCIPLE OF INCLUSION AND EXCLUSION; PIGEONHOLE PRINCIPLE	263	Chapter 4 Review	323
<i>Principle of Inclusion and Exclusion</i>	264	On the Computer	324
<i>Pigeonhole Principle</i>	269	CHAPTER 5 Relations, Functions, and Matrices	327
SECTION 4.3 Review	269	5.1 RELATIONS	328
EXERCISES 4.3	270	<i>Binary Relations</i>	328
4.4 PERMUTATIONS AND COMBINATIONS	272	<i>Properties of Relations</i>	332
<i>Permutations</i>	272	<i>Closures of Relations</i>	334
<i>Combinations</i>	274	<i>Partial Orderings</i>	336
<i>Eliminating Duplicates</i>	277	<i>Equivalence Relations</i>	339
<i>Permutations and Combinations with Repetitions</i>	279	SECTION 5.1 Review	344
<i>Generating Permutations and Combinations</i>	280	EXERCISES 5.1	345
SPECIAL INTEREST PAGE		5.2 TOPOLOGICAL SORTING	356
Archimedes and the Stomachion	286	SECTION 5.2 Review	361
SECTION 4.4 Review	288	EXERCISES 5.2	362
EXERCISES 4.4	288	5.3 RELATIONS AND DATABASES	365
4.5 BINOMIAL THEOREM	294	<i>Entity-Relationship Model</i>	365
<i>Pascal's Triangle</i>	294	<i>Relational Model</i>	366
<i>Binomial Theorem and Its Proof</i>	296	<i>Operations on Relations</i>	369
<i>Applying the Binomial Theorem</i>	298	<i>Null Values and Three-valued Logic</i>	373
SECTION 4.5 Review	299	<i>Database Integrity</i>	375
EXERCISES 4.5	299	SECTION 5.3 Review	376
4.6 PROBABILITY	301	EXERCISES 5.3	376
<i>Introduction to Finite Probability</i>	301	5.4 FUNCTIONS	381
<i>Probability Distributions</i>	304	<i>Definition</i>	381
<i>Conditional Probability</i>	306	<i>Properties of Functions</i>	388
<i>Bayes' Theorem</i>	308	Onto Functions	388
<i>Expected Value</i>	310	One-to-One Functions	389
<i>Binomial Distributions</i>	313	Bijections	390
<i>Average Case Analysis of Algorithms</i>	314	<i>Composition of Functions</i>	390
		<i>Inverse Functions</i>	392
		<i>Permutation Functions</i>	394
		<i>How Many Functions</i>	397
		<i>Equivalent Sets</i>	401
		SECTION 5.4 Review	402
		EXERCISES 5.4	402

5.5	ORDER OF MAGNITUDE	412			
	<i>Function Growth</i>	412			
	<i>More on Analysis of Algorithms</i>	415			
	<i>The Master Theorem</i>	417			
	<i>Proof of the Master Theorem</i>	419			
SECTION 5.5	Review	421			
EXERCISES 5.5		421			
5.6	THE MIGHTY MOD FUNCTION	423			
	<i>Hashing</i>	424			
	<i>Computer Security</i>	427			
	Cryptography	427			
	Hashing for Password				
	Encryption	433			
	<i>Miscellaneous Applications</i>	435			
	Identification Codes	435			
	Generating and Decomposing				
	Integers	437			
	Modular Arithmetic Designs	438			
SECTION 5.6	Review	440			
EXERCISES 5.6		440			
5.7	MATRICES	446			
	<i>Terminology</i>	446			
	<i>Matrix Operations</i>	448			
	<i>Gaussian Elimination</i>	453			
	<i>Boolean Matrices</i>	458			
<hr/>					
SPECIAL INTEREST PAGE					
	Solve Millions of Equations, Faster than Gauss	460			
SECTION 5.7	Review	461			
EXERCISES 5.7		461			
	Chapter 5 Review	470			
	On the Computer	472			
CHAPTER 6	Graphs and Trees	475			
6.1	GRAPHS AND THEIR REPRESENTATIONS	476			
	<i>Definitions of a Graph</i>	476			
	<i>Applications of Graphs</i>	479			
	<i>Graph Terminology</i>	481			
	<i>Isomorphic Graphs</i>	484			
	<i>Planar Graphs</i>	487			
	<i>Computer Representation of Graphs</i>	492			
	Adjacency Matrix	492			
	Adjacency List	494			
<hr/>					
SPECIAL INTEREST PAGE					
	Isomorphic Protein Graphs	497			
SECTION 6.1	Review	498			
EXERCISES 6.1		498			
6.2	TREES AND THEIR REPRESENTATIONS	509			
	<i>Tree Terminology</i>	509			
	<i>Applications of Trees</i>	511			
	<i>Binary Tree Representation</i>	513			
	<i>Tree Traversal Algorithms</i>	514			
	<i>Results about Trees</i>	519			
SECTION 6.2	Review	521			
EXERCISES 6.2		521			
6.3	DECISION TREES	529			
	<i>Searching</i>	529			
	Lower Bounds on Searching	532			
	Binary Tree Search	533			
	<i>Sorting</i>	535			
SECTION 6.3	Review	536			
EXERCISES 6.3		536			
6.4	HUFFMAN CODES	539			
	<i>Problem and Trial Solution</i>	539			
	<i>Huffman Encoding Algorithm</i>	542			
	<i>Justification</i>	544			
	<i>Application of Huffman Codes</i>	546			
SECTION 6.4	Review	547			
EXERCISES 6.4		548			
	Chapter 6 Review	551			
	On the Computer	552			
CHAPTER 7	Graph Algorithms	553			
7.1	DIRECTED GRAPHS AND BINARY RELATIONS; WARSHALL'S ALGORITHM	554			
	<i>Directed Graphs and Binary Relations</i>	555			
	<i>Reachability</i>	557			
	<i>Warshall's Algorithm</i>	562			

SECTION 7.1 Review	566	What is Isomorphism?	626
EXERCISES 7.1	566	Isomorphism as Applied to Boolean Algebra	628
7.2 EULER PATH AND HAMILTONIAN CIRCUIT	571		
<i>Euler Path Problem</i>	571		
<i>Hamiltonian Circuit Problem</i>	576		
SECTION 7.2 Review	577	SECTION 8.1 Review	631
EXERCISES 7.2	577	EXERCISES 8.1	631
7.3 SHORTEST PATH AND MINIMAL SPANNING TREE	581	8.2 LOGIC NETWORKS	638
<i>Shortest-Path Problem</i>	581	<i>Combinational Networks</i>	638
<i>Minimal Spanning Tree Problem</i>	587	Basic Logic Elements	638
		Boolean Expressions	639
		Truth Functions	640
		Networks and Expressions	641
		Canonical Form	642
		Minimization	645
		Programmable Logic Devices	647
		<i>A Useful Network</i>	648
		<i>Other Logic Elements</i>	650
		<i>Constructing Truth Functions</i>	652
SPECIAL INTEREST PAGE			
Pathfinding	589	SPECIAL INTEREST PAGE	
		Pruning Chips and Programs	654
SECTION 7.3 Review	591	SECTION 8.2 Review	655
EXERCISES 7.3	591	EXERCISES 8.2	655
7.4 TRAVERSAL ALGORITHMS	596	8.3 MINIMIZATION	663
<i>Depth-First Search</i>	596	<i>Minimization Process</i>	663
<i>Breadth-First Search</i>	598	<i>Karnaugh Map</i>	665
<i>Analysis</i>	601	Maps for Three and Four Variables	666
<i>Applications</i>	601	Using the Karnaugh Map	668
		<i>Quine–McCluskey Procedure</i>	673
SECTION 7.4 Review	604	SECTION 8.3 Review	677
EXERCISES 7.4	604	EXERCISES 8.3	678
7.5 ARTICULATION POINTS AND COMPUTER NETWORKS	607	Chapter 8 Review	683
<i>The Problem Statement</i>	607	On the Computer	684
<i>The Idea behind the Algorithm</i>	608		
<i>The Algorithm Itself</i>	610	CHAPTER 9 Modeling Arithmetic, Computation, and Languages	685
SECTION 7.5 Review	612	9.1 ALGEBRAIC STRUCTURES	686
EXERCISES 7.5	612	<i>Definitions and Examples</i>	686
Chapter 7 Review	614	<i>Basic Results about Groups</i>	695
On the Computer	615	<i>Subgroups</i>	698
CHAPTER 8 Boolean Algebra and Computer Logic	617	<i>Isomorphic Groups</i>	702
8.1 BOOLEAN ALGEBRA STRUCTURE	618		
<i>Models or Abstractions</i>	619		
<i>Definition and Properties</i>	620		
<i>Isomorphic Boolean Algebras</i>	626		

SECTION 9.1 Review	708	<i>Church–Turing Thesis</i>	769
EXERCISES 9.1	708	<i>Decision Problems and Uncomputability</i>	771
9.2 CODING THEORY	714	Examples of Decision Problems	772
<i>Introduction</i>	714	Halting Problem	773
<i>Background: Homomorphisms and Cosets</i>	715	<i>Computational Complexity</i>	776
<i>Generating Group Codes</i>	717	SECTION 9.4 Review	778
<i>Decoding Group Codes</i>	723	EXERCISES 9.4	779
SECTION 9.2 Review	727	9.5 FORMAL LANGUAGES	782
EXERCISES 9.2	727	<i>Classes of Grammars</i>	789
9.3 FINITE-STATE MACHINES	728	<i>Formal Languages and Computational Devices</i>	792
<i>Definition</i>	729	<i>Context-Free Grammars</i>	793
<i>Examples of Finite-State Machines</i>	729	SECTION 9.5 Review	795
<i>Recognition</i>	733	EXERCISES 9.5	795
<i>Regular Sets and Kleene’s Theorem</i>	735	Chapter 9 Review	799
<i>Machine Minimization</i>	737	On the Computer	800
Unreachable States	737	Appendix A Derivation Rules for Propositional and Predicate Logic	803
Minimization Procedure	739	Appendix B Summation and Product Notation	805
<i>Sequential Networks and Finite-State Machines</i>	744	Appendix C The Logarithm Function	809
SPECIAL INTEREST PAGE		Answers to Practice Problems	813
FSMs Behind the Game	749	Answers to Odd-Numbered Exercises	851
SECTION 9.3 Review	750	Answers to Self-Tests	949
EXERCISES 9.3	750	Index	959
9.4 TURING MACHINES	759		
<i>Definition</i>	760		
<i>Turing Machines as Set Recognizers</i>	764		
<i>Turing Machines as Function Computers</i>	767		

Preface

A course in discrete structures (discrete mathematics) played an important role in Curriculum 68, the very first ACM Computer Science Curriculum Guide: “This course introduces the student to those fundamental algebraic, logical, and combinatoric concepts from mathematics needed in the subsequent computer science courses and shows the applications of these concepts to various areas of computer science.”¹ Fast forward 45 years or so (through mobile computing, wireless networks, robotics, virtual reality, 3-D graphics, the Internet ...) to the joint ACM/IEEE-CS Computer Science Curricula 2013, where—still—discrete structures are of fundamental importance. “The material in discrete structures is pervasive in the areas of data structures and algorithms but appears elsewhere in computer science as well. For example, an ability to create and understand a proof—either a formal symbolic proof or a less formal but still mathematically rigorous argument—is important in virtually every area of computer science, including (to name just a few) formal specification, verification, databases, and cryptography. Graph theory concepts are used in networks, operating systems, and compilers. Set theory concepts are used in software engineering and in databases. Probability theory is used in intelligent systems, networking, and a number of computing applications.”²

This Seventh Edition was guided by Curricula 2013, and virtually all of the Core Tier 1 and Tier 2 topics for discrete structures from that document are included. Covering all those topics can fill a one-semester course, but there is certainly enough material in this edition to make for a very respectable two-semester course.

However much we as instructors may see the value in this foundational course, it is a difficult experience for many students, who often view it as a series of unconnected topics with little or no application to the rest of their chosen field of study. In the big picture, these topics are bound together by themes of

- importance of logical thinking
- power of mathematical notation
- usefulness of abstractions

but such themes are best appreciated in hindsight. Telling students, “You will need ideas from this course in many of your future computer science courses,” is also of little motivation. That’s why it is important to carve out time in your course syllabus (for either a one-semester or two-semester course) for some of the applications of this material. Here are topics in this edition that you may

¹*Communications of the ACM*, Vol. 11, Issue 3 (March 1968), pp. 151–197.

²Computer Science Curricula 2013, Pre-release version, <http://cs2013.com>

choose from, according to your interests and the interests of your students. Yes, students will probably see most of these topics in more detail in later computer science courses, but a quick introduction now can keep their interest and make your claim of relevance more credible.

Section 1.5	Logic programming
Sections 1.6 and 2.3	Proof of correctness
Section 3.3	Analysis of algorithms
Section 5.3	Relations and databases
Section 5.6	The mighty mod function
Section 6.4	Huffman codes
Section 8.2	Logic networks
Section 9.2	Coding theory

In addition, there is a Special Interest Page in each chapter that highlights interesting applications culled from “the real world.”

NEW IN THE SEVENTH EDITION


- The former Chapters 2 and 3 have been reorganized as Chapters 2, 3, and 4 for better clarity and sequencing
- New sections or subsections have been added:
 - Probability**
 - Bayes’ Theorem
 - Binomial Distribution
 - Order of Magnitude (new section)**
 - The Master Theorem
 - Proof of the Master Theorem
 - Matrices**
 - Gaussian Elimination
 - Coding Theory (new section)**
 - Introduction
 - Background: Homomorphisms and Cosets
 - Generating Group Codes
 - Decoding Group Codes
- “Special interest pages”—one per chapter—have been introduced to add relevance and interest to the material being presented.
- Answers to all odd-numbered exercises, as opposed to answers to fewer, selected exercises, appear in the back of the book. When an exercise asks

for a proof, the complete proof is given. Otherwise, the answer is just the answer, not necessarily the solution. A Student Solutions Manual with solutions for odd-numbered exercises from the book is available from the Web site at www.whfreeman.com/gersting. The student manual also includes two sample tests per chapter. A complete Solutions Manual is available to instructors from the publisher.

- Many new exercises have been added, particularly with an eye toward pairing odd-numbered exercises with similar even-numbered exercises.
- Of course, student learning aids such as chapter objectives, practice problems, reminders, section reviews, and chapter reviews remain.

WEB SITE

Online Study Guide

A Web site for the book may be found at www.whfreeman.com/gersting. The Web pages contain representative new example problems (not contained in the text) for many of the end-of-section Techniques. Each Technique that has a corresponding Web page example is marked with the icon .

Each example on the Web first states the problem. Then succeeding pages develop the solution, much as the student would be expected to write it. As the student navigates the pages, the solution unfolds step-by-step. A compressed audio file is also part of each Web page after the initial problem statement. The audio file contains a first-person stream-of-consciousness thought process about that step of the solution—why it occurred to the narrator to try this, why it looked promising, what knowledge was being called on to suggest that this step should come next, and so on. The point is, students see perfect and complete worked-out proofs in the textbook and often see them performed by the instructor. Yet when a student goes home and tries to produce such a solution by himself or herself, he or she is unsure where to start or how to think about the problem or how to see any pattern to enable a guess as to what to do next. Consequently the student gives up in frustration. The purpose of the audio narration is to share the “secret picture” that mathematicians use to solve problems.

To access the problems, after you go to www.whfreeman.com/gersting, select a chapter section, then select a sample problem and follow its step-by-step process with the “Next” button.

PowerPoint Slides

Instructors who visit the web site will also have access to PowerPoint slides accompanying each section of the text.

ACKNOWLEDGMENTS

My thanks to the reviewers of this addition, as well to reviewers of earlier editions, all of whose help is greatly appreciated.

Elizabeth Adams, *James Madison University*
 Kemal Akkaya, *Southern Illinois University*
 Charles Ashbacher, *Mount Mercy College*
 Barnabas Bede, *DigiPen Institute of Technology*
 Terry J. Bridgeman, *Colorado School of Mines*
 David Casperson, *University of Northern British Columbia*
 Adrienne Decker, *SUNY Buffalo*
 Steve Donaldson, *Samford University*
 Mordechai S. Goodman, *Dominican University*
 Michael A. Gray, *American University*
 Jerrold R. Griggs, *University of South Carolina*
 Joseph Hobart, *Okanagan College*
 Mark Jacobson, *University of Northern Iowa*
 Lisa A. Jamba, *University of Northern Florida*

Tim Lin, *Cal Poly*
 David Lugenbuhl, *Western Carolina University*
 Damian Lyons, *Fordham University*
 Mariana Maris, *Arizona State University*
 Mikel D. Petty, *University of Alabama in Huntsville*
 Amar Raheja, *Cal Poly*
 J. Ben Schafer, *University of Northern Iowa*
 Ali Shaykhian, *Florida Institute of Technology*
 Shunichi Toida, *Old Dominion University*
 William J. Weber, *Southeast Missouri State University*
 Eric Westlund, *Luther University*
 Hua Yan, *Borough of Manhattan Community College*
 Yu Zhang, *Texas A&M Corpus Christi*

The folks at W.H. Freeman were very helpful in shepherding this edition to completion, especially Penny Hull (veteran of many previous editions), Terri Ward, Roland Cheyney, Liam Ferguson, Georgia Lee Hadler, and Vicki Tomaselli.

Thanks to Russell Kackley for the audio files on the Web site.

My deepest thanks go to my husband, John, ever my most ardent supporter and dearest friend.

NOTE TO THE STUDENT

As you go through this book, you'll encounter many new terms and new ideas. Try reading with pencil and paper at hand and work the Practice problems as you encounter them. They are intended to reinforce or clarify some new terminology or method just introduced; answers are given at the back of the book. Pay attention also to the Reminders that point out common pitfalls or provide helpful hints.

Be sure to visit the Web site at www.whfreeman.com/gersting for detailed, worked-out solutions to additional example problems tied to the Techniques in each section. The Web site solutions are accompanied by audio files that explain each step. A Student Solutions Manual with solutions for odd-numbered exercises from the book is available from the Web site. The student manual also includes two sample tests per chapter.

You may find at first that the thought processes required to solve the exercises in the book are new and difficult. Your biggest attribute for success will be perseverance. Here's what I tell my students: "If you do not see at first how to solve a problem, don't give up, think about it some more; be sure you understand all the terminology used in the problem, play with some ideas. If no approach presents itself, let it be and think about it again later. Repeat this process for days on end. When you finally wake up in the middle of the night with an idea, you'll know you are putting in the right amount of effort for this course." Mathematical results don't spring fully formed from the foreheads of mathematical geniuses; well, maybe from mathematical geniuses, but for the rest of us, it takes work, patience, false starts, and **perseverance**.

Enjoy the experience!

Formal Logic

CHAPTER OBJECTIVES

After studying this chapter, you will be able to:

- Use the formal symbols of propositional logic.
- Find the truth value of an expression in propositional logic.
- Construct formal proofs in propositional logic, and use such proofs to determine the validity of English language arguments.
- Use the formal symbols of predicate logic.
- Find the truth value in some interpretation of an expression in predicate logic.
- Use predicate logic to represent English language sentences.
- Construct formal proofs in predicate logic, and use such proofs to determine the validity of English language arguments.
- Understand how the programming language Prolog is built on predicate logic.
- Mathematically prove the correctness of programs that use assignment statements and conditional statements.

You have been selected to serve on jury duty for a criminal case. The attorney for the defense argues as follows:

If my client is guilty, then the knife was in the drawer. Either the knife was not in the drawer or Jason Pritchard saw the knife. If the knife was not there on October 10, it follows that Jason Pritchard did not see the knife. Furthermore, if the knife was there on October 10, then the knife was in the drawer and also the hammer was in the barn. But we all know that the hammer was not in the barn. Therefore, ladies and gentlemen of the jury, my client is innocent.

Question: Is the attorney's argument sound? How should you vote?

It's much easier to answer this question if the argument is recast in the notation of formal logic. Formal logic strips away confusing verbiage and allows us to concentrate on the underlying reasoning being applied. In fact, formal logic—the subject of this chapter—provides the foundation for the organized, careful method of thinking that characterizes any reasoned activity—a criminal investigation, a scientific experiment, a sociological study. In addition, formal logic has direct applications in computer science. The last two sections of this chapter explore a programming language based on logic and the use of formal logic to verify the correctness of computer programs. Also, circuit logic (the logic governing

computer circuitry) is a direct analog of the statement logic of this chapter. We will study circuit logic in Chapter 8.

SECTION 1.1 STATEMENTS, SYMBOLIC REPRESENTATION, AND TAUTOLOGIES

Formal logic can represent the statements we use in English to communicate facts or information. A **statement** (or **proposition**) is a sentence that is either true or false.

EXAMPLE 1

Consider the following:

- a. Ten is less than seven.
- b. Cheyenne is the capital of Wyoming.
- c. She is very talented.
- d. There are life forms on other planets in the universe.

Sentence (a) is a statement because it is false. Sentence (b) is a statement because it is true. Sentence (c) is neither true nor false because “she” is not specified; therefore (c) is not a statement. Sentence (d) is a statement because it is either true or false; we do not have to be able to decide which. ●

Connectives and Truth Values

In English, simple statements are combined with connecting words like *and* to make more interesting compound statements. The truth value of a compound statement depends on the truth values of its components and which connecting words are used. If we combine the two true statement, “Elephants are big,” and, “Baseballs are round,” we would consider the resulting statement, “Elephants are big and baseballs are round,” to be true. In this book, as in many logic books, capital letters near the beginning of the alphabet, such as A , B , and C , are used to represent statements and are called **statement letters**; the symbol \wedge is a **logical connective** representing *and*. We agree, then, that if A is true and B is true, $A \wedge B$ (read “ A and B ”) should be considered true.

PRACTICE 1

¹

- a. If A is true and B is false, what truth value would you assign to $A \wedge B$?
- b. If A is false and B is true, what truth value would you assign to $A \wedge B$?
- c. If A and B are both false, what truth value would you assign to $A \wedge B$? ■

The expression $A \wedge B$ is called the **conjunction** of A and B , and A and B are called the **conjuncts** of this expression. Table 1.1 summarizes the truth value of $A \wedge B$ for all possible truth values of the conjuncts A and B . Each row of the table

¹Answers to practice problems are in the back of the text.

A	B	$A \wedge B$
T	T	T
T	F	F
F	T	F
F	F	F

A	B	$A \vee B$
T	T	T
T	F	
F	T	
F	F	

represents a particular truth value assignment to the statement letters, and the resulting truth value for the compound expression is shown.

Another connective is the word *or*, denoted by the symbol \vee . The expression $A \vee B$ (read “ A or B ”) is called the **disjunction** of A and B , and A and B are called the **disjuncts** of this expression. If A and B are both true, then $A \vee B$ would be considered true, giving the first line of the truth table for disjunction (Table 1.2).

PRACTICE 2 Use your understanding of the word *or* to complete the truth table for disjunction, Table 1.2. ■

Statements may be combined in the form “if statement 1, then statement 2.” If A denotes statement 1 and B denotes statement 2, the compound statement would be denoted by $A \rightarrow B$ (read “ A implies B ”). The logical connective here is **implication**, and it conveys the meaning that the truth of A implies or leads to the truth of B . In the implication $A \rightarrow B$, A stands for the **antecedent** statement and B stands for the **consequent** statement.

The truth table for implication is less obvious than that for conjunction or disjunction. To understand its definition, let’s suppose your friend remarks, “If I pass my economics test, then I’ll go to the movie Friday.” If your friend passes the test and goes to the movie, the remark was true. If your friend passes the test but doesn’t go to the movie, the remark was false. If your friend doesn’t pass the test, then—whether he or she goes to the movie or not—you could not claim that the remark was false. You would probably want to give the benefit of the doubt and say that the statement was true. By convention, $A \rightarrow B$ is considered true if A is false, regardless of the truth value of B .

PRACTICE 3 Summarize this discussion by writing the truth table for $A \rightarrow B$. ■

The **equivalence** connective is symbolized by \leftrightarrow . Unlike conjunction, disjunction, and implication, the equivalence connective is not really a fundamental connective but a convenient shortcut. The expression $A \leftrightarrow B$ stands for $(A \rightarrow B) \wedge (B \rightarrow A)$. We can write the truth table for equivalence by constructing, one piece at a time, a table for $(A \rightarrow B) \wedge (B \rightarrow A)$, as in Table 1.3. From this truth table, $A \leftrightarrow B$ is true exactly when A and B have the same truth value.

A	B	$A \rightarrow B$	$B \rightarrow A$	$(A \rightarrow B) \wedge (B \rightarrow A)$
T	T	T	T	T
T	F	F	T	F
F	T	T	F	F
F	F	T	T	T

The connectives we’ve seen so far are called **binary connectives** because they join two expressions together to produce a third expression. Now let’s consider a **unary connective**, a connective acting on one expression to produce a second

expression. **Negation** is a unary connective. The negation of A —symbolized by A' —is read “not A .”

PRACTICE 4 Write the truth table for A' . (It will require only two rows.)

Table 1.4 summarizes the truth values for all of the logical connectives. This information is critical to an understanding of logical reasoning.

A	B	$A \wedge B$	$A \vee B$	$A \rightarrow B$	$A \leftrightarrow B$	A'
T	T	T	T	T	T	F
T	F	F	T	F	F	
F	T	F	T	T	F	T
F	F	F	F	T	T	

Because of the richness of the English language, words that have different shades of meaning are nonetheless represented by the same logical connective. Table 1.5 shows the common English words associated with various logical connectives.

English Word	Logical Connective	Logical Expression
and; but; also; in addition; moreover	Conjunction	$A \wedge B$
or	Disjunction	$A \vee B$
If A , then B . A implies B . A , therefore B . A only if B . B follows from A . A is a sufficient condition for B . B is a necessary condition for A .	Implication	$A \rightarrow B$
A if and only if B . A is necessary and sufficient for B .	Equivalence	$A \leftrightarrow B$
not A It is false that A ... It is not true that A ...	Negation	A'

REMINDER

A only if B means
 $A \rightarrow B$

Suppose that $A \rightarrow B$ is true. Then, according to the truth table for implication, the consequent, B , can be true even though the antecedent, A , is false. So while the truth of A leads to (implies) the truth of B , the truth of B does not imply the truth of A . The phrase “ B is a necessary condition for A ” to describe $A \rightarrow B$ simply

means that if A is true, then B is necessarily true, as well. “ A only if B ” describes the same thing, that A implies B .

EXAMPLE 2

The statement, “Fire is a necessary condition for smoke,” can be restated, “If there is smoke, then there is fire.” The antecedent is “there is smoke,” and the consequent is “there is fire.”

PRACTICE 5

Name the antecedent and consequent in each of the following statements. (*Hint:* Rewrite each statement in if-then form.)

- If the rain continues, then the river will flood.
- A sufficient condition for network failure is that the central switch goes down.
- The avocados are ripe only if they are dark and soft.
- A good diet is a necessary condition for a healthy cat.

EXAMPLE 3

Expressing the negation of a statement must be done with care, especially for a compound statement. Table 1.6 gives some examples.

TABLE 1.6

Statement	Correct Negation	Incorrect Negation
It will rain tomorrow.	It is false that it will rain tomorrow. It will not rain tomorrow.	
Peter is tall and thin.	It is false that Peter is tall and thin. Peter is not tall or he is not thin. Peter is short or fat.	Peter is short and fat. Too strong a statement. Peter fails to have both properties (tallness and thinness) but may still have one property.
The river is shallow or polluted.	It is false that the river is shallow or polluted. The river is neither shallow nor polluted. The river is deep and unpolluted.	The river is not shallow or not polluted. Too weak a statement. The river fails to have either property, not just fails to have one property.

PRACTICE 6

Which of the following represents A' if A is the statement “Julie likes butter but hates cream”?

- Julie hates butter and cream.
- Julie does not like butter or cream.
- Julie dislikes butter but loves cream.
- Julie hates butter or likes cream.

We can string statement letters, connectives, and parentheses (or brackets) together to form new expressions, as in

$$(A \rightarrow B) \wedge (B \rightarrow A)$$

Of course, just as in a computer programming language, certain *syntax rules* (rules on which strings are legitimate) prevail; for example,

$$A)) \wedge \wedge \rightarrow BC$$

would not be considered a legitimate string. An expression that is a legitimate string is called a **well-formed formula**, or **wff**. To reduce the number of parentheses required in a wff, we stipulate an order in which connectives are applied. This *order of precedence* is

1. connectives within parentheses, innermost parentheses first
2. ' (negation)
3. \wedge, \vee (conjunction, disjunction)
4. \rightarrow (implication)
5. \leftrightarrow (biconditional)

This means that the expression $A \vee B'$ stands for $A \vee (B')$, not $(A \vee B)'$. Similarly, $A \vee B \rightarrow C$ means $(A \vee B) \rightarrow C$, not $A \vee (B \rightarrow C)$. However, we often use parentheses anyway, just to be sure that there is no confusion.

In a wff with a number of connectives, the connective to be applied last is the **main connective**. In

$$A \wedge (B \rightarrow C)'$$

the main connective is \wedge . In

$$((A \vee B) \wedge C) \rightarrow (B \vee C')$$

the main connective is \rightarrow . Capital letters near the end of the alphabet, such as P, Q, R , and S , are used to represent wffs. Thus P could represent a single statement letter, which is the simplest kind of wff, or a more complex wff. We might represent

$$((A \vee B) \wedge C) \rightarrow (B \vee C')$$

as

$$P \rightarrow Q$$

if we want to hide some of the details for the moment and only concentrate on the main connective.

Wffs composed of statement letters and connectives have truth values that depend on the truth values assigned to their statement letters. We write the truth table for any wff by building up the component parts, just as we did for $(A \rightarrow B) \wedge (B \rightarrow A)$. The main connective is addressed in the last column of the table.

EXAMPLE 4

The truth table for the wff $A \vee B' \rightarrow (A \vee B)'$ is given in Table 1.7. The main connective, according to the rules of precedence, is implication. ●

A	B	B'	$A \vee B'$	$A \vee B$	$(A \vee B)'$	$A \vee B' \rightarrow (A \vee B)'$
T	T	F	T	T	F	F
T	F	T	T	T	F	F
F	T	F	F	T	F	T
F	F	T	T	F	T	T

If we are making a truth table for a wff that contains n different statement letters, how many rows will the truth table have? From truth tables done so far, we know that a wff with only one statement letter has two rows in its truth table, and a wff with two statement letters has four rows. The number of rows equals the number of true-false combinations possible among the statement letters. The first statement letter has two possibilities, T and F. For each of these possibilities, the second statement letter has two possible values. Figure 1.1a pictures this as a two-level “tree” with four branches showing the four possible combinations of T and F for two statement letters. For n statement letters, we extend the tree to n levels, as in Figure 1.1b. The total number of branches then equals 2^n . The total number of rows in a truth table for n statement letters is also 2^n .

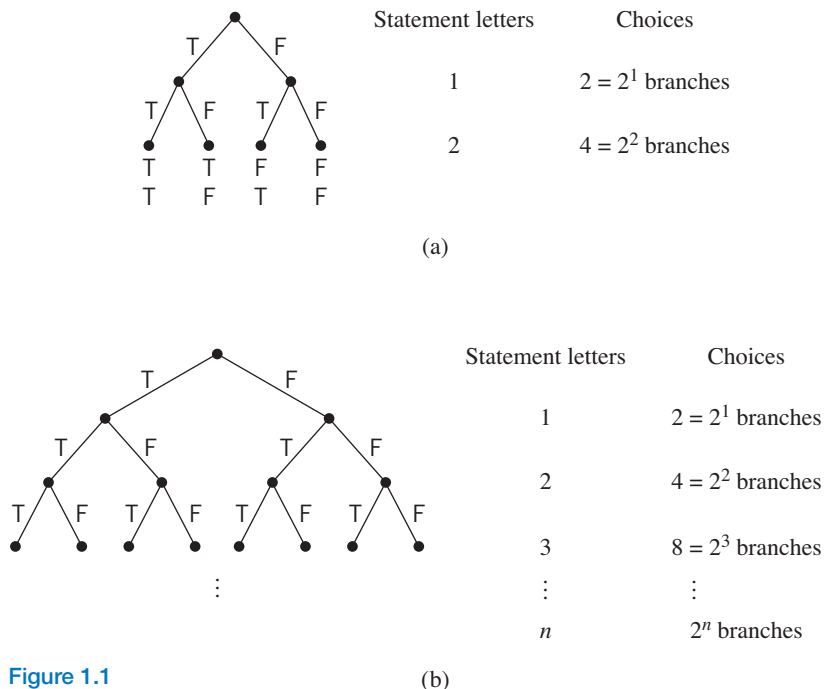


Figure 1.1

A	B	C
T	T	T
T	T	F
T	F	T
T	F	F
F	T	T
F	T	F
F	F	T
F	F	F

This tree structure also tells us how to enumerate all the T–F combinations among the n statement letters when setting up a truth table. If we read each level of the tree from bottom to top, it says that the T–F values for statement letter n (which will compose the last column of the truth table) alternate, those for statement letter $n - 1$ alternate every two values, those for statement letter $n - 2$ alternate every four values, and so forth. Thus a truth table for three statement letters would begin as shown in Table 1.8. The values for statement letter C alternate, those for statement letter B alternate in groups of two, and those for statement letter A alternate in groups of four, resulting in something like a sideways version of the tree. (Reading the rows from the bottom up and using 1 for T and 0 for F shows that we are simply counting up from zero in binary numbers.)

PRACTICE 7

Construct truth tables for the following wffs.

- $(A \rightarrow B) \leftrightarrow (B \rightarrow A)$ (Remember that $C \leftrightarrow D$ is true precisely when C and D have the same truth value.)
- $(A \vee A') \rightarrow (B \wedge B')$
- $[(A \wedge B') \rightarrow C']'$
- $(A \rightarrow B) \leftrightarrow (B' \rightarrow A')$

Tautologies

A wff-like item (d) of Practice 7, whose truth values are always true, is called a **tautology**. A tautology is “intrinsically true” by its very structure; it is true no matter what truth values are assigned to its statement letters. A simpler example of a tautology is $A \vee A'$; consider, for example, the statement “Today the sun will shine or today the sun will not shine,” which must always be true because one or the other of these must happen. A wff like item (b) of Practice 7, whose truth values are always false, is called a **contradiction**. A contradiction is “intrinsically false” by its very structure. A simpler example of a contradiction is $A \wedge A'$; consider “Today is Tuesday and today is not Tuesday,” which is false no matter what day of the week it is.

REMINDER

A, B, C stand for single statement letters; P, Q, R, S stand for wffs.

Suppose that P and Q represent two wffs, and it happens that the wff $P \leftrightarrow Q$ is a tautology. If we did a truth table using the statement letters in P and Q , then the truth values of the wffs P and Q would agree for every row of the truth table. In this case, P and Q are said to be **equivalent wffs**, denoted by $P \Leftrightarrow Q$. Thus $P \Leftrightarrow Q$ states a fact, namely, that the particular wff $P \leftrightarrow Q$ is a tautology. Practice 7(d) has the form $P \leftrightarrow Q$, where P is the wff $(A \rightarrow B)$ and Q is the wff $(B' \rightarrow A')$, and $P \leftrightarrow Q$ was shown to be a tautology. Therefore, $(A \rightarrow B) \Leftrightarrow (B' \rightarrow A')$.

We will list some basic equivalences, prove one or two of them by constructing truth tables, and leave the rest as exercises. We represent any contradiction by 0 and any tautology by 1.

Some Tautological Equivalences

- | | | |
|--|--|---------------------------|
| 1a. $A \vee B \Leftrightarrow B \vee A$ | 1b. $A \wedge B \Leftrightarrow B \wedge A$ | (commutative properties) |
| 2a. $(A \vee B) \vee C \Leftrightarrow A \vee (B \vee C)$ | 2b. $(A \wedge B) \wedge C \Leftrightarrow A \wedge (B \wedge C)$ | (associative properties) |
| 3a. $A \vee (B \wedge C) \Leftrightarrow (A \vee B) \wedge (A \vee C)$ | 3b. $A \wedge (B \vee C) \Leftrightarrow (A \wedge B) \vee (A \wedge C)$ | (distributive properties) |
| 4a. $A \vee 0 \Leftrightarrow A$ | 4b. $A \wedge 1 \Leftrightarrow A$ | (identity properties) |
| 5a. $A \vee A' \Leftrightarrow 1$ | 5b. $A \wedge A' \Leftrightarrow 0$ | (complement properties) |

Note that 2a allows us to write $A \vee B \vee C$ with no need for parentheses because the grouping doesn't matter; similarly, 2b allows us to write $A \wedge B \wedge C$.

EXAMPLE 5

The truth table in Table 1.9a verifies equivalence 1a, the commutative property for disjunction, and that in Table 1.9b verifies 4b, the identity property for conjunction. Note that only two rows are needed for Table 1.9b because 1 (a tautology) cannot take on false truth values.

TABLE 1.9										
(a)	A	B	$A \vee B$	$B \vee A$	$A \vee B \leftrightarrow B \vee A$	(b)	A	1	$A \wedge 1$	$A \wedge 1 \leftrightarrow A$
	T	T	T	T	T		T	T	T	T
	T	F	T	T	T		F	T	F	T
	F	T	T	T	T					
	F	F	F	F	T					

PRACTICE 8 Verify equivalence 5a.

The equivalences in the list are grouped into five pairs. In each pair, one equivalence can be obtained from the other by replacing \wedge with \vee , \vee with \wedge , 0 with 1, or 1 with 0. Each equivalence in a pair is called the **dual** of the other. Thus, 1a and 1b (commutativity of disjunction and commutativity of conjunction) are duals of each other. This list of equivalences appears in a more general setting in Chapter 8.

Two additional equivalences that are very useful are **De Morgan's laws**, named for the nineteenth-century British mathematician Augustus De Morgan, who first stated them. This theorem is easy to prove (see Exercises 26e and 26f).