McGRAW-HILL EDITION

***LIST OF SYMBOLS***

*TOPIC SYMBOL MEANING PAGE*

*LOGIC ""'p* negation of *p* 3 *p 1\ q* conjunction of *p* and *q* 4 *P vq* disjunction of *p* and *q* 4 *P (J) q* exclusive or of *p* and *q* 5 *p-+q* the implication *p* implies *q* 6 *p* ++ *q* biconditional of *p* and *q* 9 *p=.q* equivalence of *p* and *q* 22 T tautology 24 F contradiction 24 *P(xJ, ... , xn)* propositional function 32 *VxP(x)* universal quantification of *P(x)* 34 *3xP(x)* existential quantification of *P(x)* 36 *3!xP(x)* uniqueness quantification of *P(x)* 37 therefore 63 *p{S}q* partial correctness of S 323

*SETS* XES *X* is a member of S 112 *X* fj S *X* is not a member of S 112 *{aJ,* ... , *an}* list of elements of a set 112 *{x* I *P(x)}* set builder notation 112 N set of natural numbers 112 Z set of integers 112 z+ set of positive integers 113 Q set of rational numbers 113 R set of real numbers 113 *S=T* set equality 113 0 the empty (or null) set 114 *Sc.;T* S is a subset of *T* 114 *SeT* S is a proper subset of *T* 115 lSI cardinality of S 116 *P(S)* the power set of S 116 *(aJ, ... , an)* n-tuple 117 *(a, b)* ordered pair 117 *AxB* Cartesian product of *A* and *B* 118 *AUB* union of *A* and *B* 121 *AnB* intersection of *A* and *B* 121 *A-B* the difference of *A* and *B* 123 *A* complement of *A* 123 *n UAi* union of *Ai, i* = 1, 2, ... , *n* 127 i=J *n nAi* intersection of *Ai, i* = 1, 2, ... , *n* 128 i=J *A(J)B* symmetric difference of *A* and *B* 131

*TOPIC SYMBOL MEANING PAGE*

FUNCTIONS *J(a)* value of the function *J* at *a* 133 *J:A* ---+ *B* function from *A* to *B* 133 *JI* + *J2* sum of the functions *JI* and *h* 135 *JI/2* product of the functions *JI* and *h* 135 *J(S)* image of the set Sunder *J* 136 *tA(S)* identity function on *A* 138 *J-I(x)* inverse of *J* 139 *Jog* composition of *J* and g 140 LxJ floor function of *x* 143 fxl ceiling function of *x* 143 *an* term of *{ai}* with subscript *n* 150 *n Lai* sum of *ai, a2,* ... , *an* 153 i=1 *Laa* sum of *aa* over *a* E S 156 aES *n* fl *an* product of *ai, a2,* ... , *an* 162 i=1 *J(x)* is *O(g(x)) J (x)* is big-O of *g(x )* 180 *n! n* factorial 185 *J(x)* is *Q(g(x)) J(x)* is big-Omega of *g(x)* 189 *J(x)* is *8(g(x)) J(x)* is big-Theta of *g(x)* 189 asymptotic 192 *min(x,y)* minimum of *x* and *y* 216 *max(x,y)* maximum of *x* and *y* 217 ~ approximately equal to 395

INTEGERS *alb a* divides *b* 201 *alb a* does not divide *b* 201 *a* **div** *b* quotient when *a* is divided by *b* 202 *a* **mod** *b* remainder when *a* is divided by *b* 202 *a=b(modm) a* is congruent to *b* modulo *m* 203 *a =1= b* (mod *m) a* is not congruent to *b* modulo *m* 203 gcd(a, *b)* greatest common divisor of *a* and *b* 215 lcm *(a, b)* least common multiple of *a* and *b* 217 *(akak-I* ... *alaO)b* base *b* representation 219

MATRICES *[aij ]* matrix with entries *aij* 247 **A+B** matrix sum of A and B 247 **AB** matrix product of A and B 248 In identity matrix of order *n* 251 *At* transpose of A 251 AvB join of A and B 252 AI\B the meet of A and B 252 A0B Boolean product of A and B 253 *A[n] nth* Boolean power of A 254

(List of Symbols continued at back of book)

**Discrete Mathematics and Its Applications**

**Sixth Edition**

**Kenneth H. Rosen**

*AT&T Laboratories*

Boston Burr Ridge, IL Dubuque, IA New York San Francisco St. Louis Bangkok Bogota Caracas Kuala Lumpur Lisbon London Madrid Mexico City Milan Montreal New Delhi Santiago Seoul Singapore Sydney Taipei Toronto

*The McGraw Hilt Compames 4*

DISCRETE MATHEMATICS AND ITS APPLICATIONS, SIXTH EDITION International Edition 2007

Exclusive rights by McGraw-Hill Education (Asia), for manufacture and export. This book cannot be re-exported from the country to which it is sold by McGraw-Hill. The International Edition is not available in North America.

Published by McGraw-Hill, a business unit of The McGraw-Hill Companies, Inc., 1221 Avenue of the Americas, New York, NY 10020. Copyright © 2007 by Kenneth H. Rosen. 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.

10 09 08 07 06 05 04 03 02 01 20 09 08 07 06 CTF BJE

The credits section for this book begins on page C-l and is considered an extension of the copyright page.

When ordering this title, use ISBN-13: 978-007-124474-9 or ISBN-I0: 007-124474-3

Printed in Singapore

www.rnhhe.com

**Contents**

*Preface* vii ***The MathZone Companion Website* xviii *To the Student* xx**

**1 The Foundations: Logic and Proofs .................................. 1**

1.1 Propositional Logic ............................................................. 1 1.2 Propositional Equivalences ..................................................... 21 1.3 Predicates and Quantifiers ...................................................... 30 1.4 Nested Quantifiers ............................................................. 50 1.5 Rules ofInference ............................................................. 63 1.6 Introduction to Proofs .......................................................... 75 1.7 Proof Methods and Strategy .................................................... 86 End-of-Chapter Material ...................................................... 104

**2 Basic Structures: Sets, Functions, Sequences, and Sums ........... 111**

2.1 . Sets ................................................................ : ........ 111 2.2 Set Operations ............................................................... 121 2.3 Functions .................................................................... 133 2.4 Sequences and Summations ................................................... 149 End-of-Chapter Material ...................................................... 163

3 **The Fundamentals: Algorithms, the Integers, and Matrices ........ 167**

3.1 Algorithms .................................................................. 167 3.2 The Growth of Functions ...................................................... 180 3.3 Complexity of Algorithms ..................................................... 193 3.4 The Integers and Division ..................................................... 200 3.5 Primes and Greatest Common Divisors ......................................... 210 3.6 Integers and Algorithms ....................................................... 219 3.7 Applications of Number Theory ............................................... 231 3.8 Matrices ..................................................................... 246 End-of-Chapter Material ...................................................... 257

**4 Induction and Recursion .......................................... 263**

4.1 Mathematical Induction ....................................................... 263 4.2 Strong Induction and Well-Ordering ........................................... 283 4.3 Recursive Definitions and Structural Induction .................................. 294 4.4 Recursive Algorithms ......................................................... 311 iii

Contents

4.5 Program Correctness ......................................................... 322 End-of-Chapter Material ...................................................... 328

**5** Co un **ting .......................................................... 335**

5.1 The Basics of Counting ....................................................... 335 5.2 The Pigeonhole Principle ...................................................... 347 5.3 Permutations and Combinations ............................................... 355 5.4 Binomial Coefficients ......................................................... 363 5.5 Generalized Permutations and Combinations .................................... 370 5.6 Generating Permutations and Combinations ..................................... 382 End-of-Chapter Material ...................................................... 386

6 Discrete Probability ............................................... 393

6.1 An Introduction to Discrete Probability ......................................... 393 6.2 Probability Theory ........................................................... 400 6.3 Bayes' Theorem .............................................................. 417 6.4 Expected Value and Variance .................................................. 426 End-of-Chapter Material ...................................................... 442

7 Advanced Counting Techniques .................................. .449

7.1 Recurrence Relations ......................................................... 449 7.2 Solving Linear Recurrence Relations ........................................... 460 7.3 Divide-and-Conquer Algorithms and Recurrence Relations ....................... 474 7.4 Generating Functions ......................................................... 484 7.5 Inclusion-Exclusion .......................................................... 499 7.6 Applications ofInclusion-Exclusion ........................................... 505 End-of-Chapter Material ...................................................... 513

8 Relations .......................................................... 519

8.1 Relations and Their Properties ................................................. 519 *8.2 n-ary* Relations and Their Applications ......................................... 530 8.3 Representing Relations ....................................................... 537 8.4 Closures of Relations ......................................................... 544 8.5 Equivalence Relations ........................................................ 555 8.6 Partial Orderings ............................................................. 566 End-of-Chapter Material ...................................................... 581

9 Graphs ............................................................ 589

9.1 Graphs and Graph Models ..................................................... 589 9.2 Graph Terminology and Special Types of Graphs ................................ 597 9.3 Representing Graphs and Graph Isomorphism ................................... 611 9.4 Connectivity ................................................................. 621

Contents v

9.5 Euler and Hamilton Paths ..................................................... 633 9.6 Shortest-Path Problems ....................................................... 647 9.7 Planar Graphs ................................................................ 657 9.8 Graph Coloring .............................................................. 666 End-of-Chapter Material ...................................................... 675

**10 Trees ............................................................... 683**

10.1 Introduction to Trees .......................................................... 683 10.2 Applications of Trees ......................................................... 695 10.3 Tree TraversaL ............................................................... 710 10.4 Spanning Trees .............................................................. 724 10.5 Minimum Spanning Trees ..................................................... 737 End-of-Chapter Material ...................................................... 743

**11 Boolean Algebra ................................................... 749**

11.1 Boolean Functions ............................................................ 749 11.2 Representing Boolean Functions ............................................... 757 11.3 Logic Gates .................................................................. 760 11.4 Minimization of Circuits ...................................................... 766 End-of-Chapter Material ...................................................... 781

**12 Modeling Computation ........................................... 785**

12.1 Languages and Grammars ..................................................... 785 12.2 Finite-State Machines with Output ............................................. 796 12.3 Finite-State Machines with No Output .......................................... 804 12.4 Language Recognition ........................................................ 817 12.5 Turing Machines ............................................................. 827 End-of-Chapter Material ...................................................... 838

**Appendixes ....................................................... A-I**

A-I Axioms for the Real Numbers and the Positive Integers .......................... A-I A-2 Exponential and Logarithmic Functions ........................................ A-7 A-3 Pseudocode ................................................................ A-IO

*Suggested Readings* B-1 *Answers to Odd-Numbered Exercises* S-1 *Photo Credits* C-l *Index of Biographies 1-1 Index* 1-2

**About the Author**

K enneth H. Rosen has had a long career as a Distinguished Member ofthe Technical Staff at AT&T Laboratories in Monmouth County, New Jersey. He currently holds the position of visiting research professor at Monmouth University where he is working on the security and privacy aspects of the Rapid Response Database Project and where he is teaching a course on cryptographic applications.

Dr. Rosen received his B.S. in Mathematics from the University of Michigan, Ann Arbor (1972), and his Ph.D. in Mathematics from M.1. T. (1976), where he wrote his thesis in the area of number theory under the direction of Harold Stark. Before joining Bell Laboratories in 1982, he held positions at the University of Colorado, Boulder; The Ohio State University, Columbus; and the University of Maine, Orono, where he was an associate professor of mathematics. While working at AT&T Labs, he taught at Monmouth University, teaching courses in discrete mathematics, coding theory, and data security.

Dr. Rosen has published numerous articles in professional journals in the areas of number theory and mathematical modeling. He is the author of the textbooks *Elementary Number Theory and Its Applications,* published by Addison-Wesley and currently in its fifth edition, and *Discrete Mathematics and Its Applications,* published by McGraw-Hili and currently in its sixth edition. Both of these textbooks have been used extensively at hundreds of universities throughout the world. *Discrete Mathematics and Its Applications* has sold more than 300,000 copies in its lifetime with translations into Spanish, French, Chinese, and Korean. He is also co-author of *UNIX The Complete Reference; UNIX System V Release* 4: *An Introduction;* and *Best UNIX Tips Ever,* each published by Osborne McGraw-Hili. These books have sold more than 150,000 copies with translations into Chinese, German, Spanish, and Italian. Dr. Rosen is also the editor of the *Handbook of Discrete and Combinatorial Mathematics,* published by CRC Press, and he is the advisory editor of the CRC series of books in discrete mathematics, consisting of more than 30 volumes on different aspects of discrete mathematics, most of which are introduced in this book. He is also interested in integrating mathematical software into the educational and professional environments, and has worked on projects with Waterloo Maple Inc.'s Maple software in both these areas.

At Bell Laboratories and AT&T Laboratories, Dr. Rosen worked on a wide range of projects, including operations research studies and product line planning for computers and data commu- nications equipment. He helped plan AT&T's products and services in the area of multimedia, including video communications, speech recognition and synthesis, and image networking. He evaluated new technology for use by AT&T and did standards work in the area of image network- ing. He also invented many new services, and holds or has submitted more than 70 patents. One of his more interesting projects involved helping evaluate technology for the AT&T attraction at EPCOT Center.

**Preface**

I n writing this book, I was guided by my long-standing experience and interest in teaching discrete mathematics. For the student, my purpose was to present material in a precise, readable manner, with the concepts and techniques of discrete mathematics clearly presented and demonstrated. My goal was to show the relevance and practicality of discrete mathematics to students, who are often skeptical. I wanted to give students studying computer science all of the mathematical foundations they need for their future studies. I wanted to give mathematics students an understanding of important mathematical concepts together with a sense of why these concepts are important for applications. And most importantly, I wanted to accomplish these goals without watering down the material.

For the instructor, my purpose was to design a flexible, comprehensive teaching tool using proven pedagogical techniques in mathematics. I wanted to provide instructors with a package of materials that they could use to teach discrete mathematics effectively and efficiently in the most appropriate manner for their particular set of students. I hope that I have achieved these goals.

I have been extremely gratified by the tremendous success of this text. The many improve- ments in the sixth edition have been made possible by the feedback and suggestions of a large number of instructors and students at many of the more than 600 schools where this book has been successfully used. There are many enhancements in this edition. The companion website has been substantially enhanced and more closely integrated with the text, providing helpful material to make it easier for students and instructors to achieve their goals.

This text is designed for a one- or two-term introductory discrete mathematics course taken by students in a wide variety of majors, including mathematics, computer science, and engineering. College algebra is the only explicit prerequisite, although a certain degree of mathematical maturity is needed to study discrete mathematics in a meaningful way.

**Goals of a Discrete Mathematics Course**

A discrete mathematics course has more than one purpose. Students should learn a particular set of mathematical facts and how to apply them; more importantly, such a course should teach students how to think logically and mathematically. To achieve these goals, this text stresses mathematical reasoning and the different ways problems are solved. Five important themes are interwoven in this text: mathematical reasoning, combinatorial analysis, discrete structures, algorithmic thinking, and applications and modeling. A successful discrete mathematics course should carefully blend and balance all five themes.

*1. Mathematical Reasoning:* Students must understand mathematical reasoning in order to read, comprehend, and construct mathematical arguments. This text starts with a discussion of mathematical logic, which serves as the foundation for the subsequent discussions of methods of proof. Both the science and the art of constructing proofs are addressed. The technique of mathematical induction is stressed through many different types of examples of such proofs and a careful explanation of why mathematical induction is a valid proof technique. *2. Combinatorial Analysis:* An important problem-solving skill is the ability to count or enu- merate objects. The discussion of enumeration in this book begins with the basic techniques of counting. The stress is on performing combinatorial analysis to solve counting problems and analyze algorithms, not on applying formulae.

vii

**vJli** Preface

*3. Discrete Structures:* A course in discrete mathematics should teach students how to work with discrete structures, which are the abstract mathematical structures used to represent discrete objects and relationships between these objects. These discrete structures include sets, permutations, relations, graphs, trees, and finite-state machines. *4. Algorithmic Thinking:* Certain classes of problems are solved by the specification of an algorithm. After an algorithm has been described, a computer program can be constructed implementing it. The mathematical portions of this activity, which include the specification of the algorithm, the verification that it works properly, and the analysis of the computer memory and time required to perform it, are all covered in this text. Algorithms are described using both English and an easily understood form of pseudocode. *5. Applications and Modeling:* Discrete mathematics has applications to almost every conceiv- able area of study. There are many applications to computer science and data networking in this text, as well as applications to such diverse areas as chemistry, botany, zoology, linguis- tics, geography, business, and the Internet. These applications are natural and important uses of discrete mathematics and are not contrived. Modeling with discrete mathematics is an extremely important problem-solving skill, which students have the opportunity to develop by constructing their own models in some of the exercises.

**Changes in the Sixth Edition**

The fifth edition of this book has been used successfully at over 600 schools in the United States, dozens of Canadian universities, and at universities throughout Europe, Asia, and Oceania. Although the fifth edition has been an extremely effective text, many instructors, including longtime users, have requested changes designed to make this book more effective. I have devoted a significant amount oftime and energy to satisfy these requests and I have worked hard to find my own ways to make the book better.

The result is a sixth edition that offers both instructors and students much more than the fifth edition did. Most significantly, an improved organization of topics has been implemented in this sixth edition, making the book a more effective teaching tool. Changes have been implemented that make this book more effective for students who need as much help as possible, as well as for those students who want to be challenged to the maximum degree. Substantial enhancements to the material devoted to logic, method of proof, and proof strategies are designed to help students master mathematical reasoning. Additional explanations and examples have been added to clarify material where students often have difficulty. New exercises, both routine and challenging, have been inserted into the exercise sets. Highly relevant applications, including many related to the Internet and computer science, have been added. The MathZone companion website has benefited from extensive development activity and now provides tools students can use to master key concepts and explore the world of discrete mathematics.

**Improved Organization**

• The first part of the book has been restructured to present core topics in a more efficient, more effective, and more flexible way.

• Coverage of mathematical reasoning and proof is con- centrated in Chapter 1, flowing from propositional and predicate logic, to rules of inference, to basic proof techniques, to more advanced proof techniques and proof strategies.

• A separate chapter on discrete structures-Chapter 2 in this new edition---covers sets, functions, sequence, and sums.

• Material on basic number theory, covered in one sec- tion in the fifth edition, is now covered in two sections, the first on divisibility and congruences and the second on primes.

• The new Chapter 4 is entirely devoted to induction and recursion.

Preface **ix**

**Logic**

• Coverage of logic has been amplified with key

• More care is devoted to introducing predicates and ideas explained in greater depth and with more

quantifiers, as well as to explaining how to use and care.

work with them.

• Conditional statements and De Morgan's laws receive expanded coverage.

• The application of logic to system specifications-a topic of interest to system, hardware, and software engineers-has been expanded.

• The construction of truth tables is introduced earlier

• Material on valid arguments and rules of inference is and in more detail.

now presented in a separate section.

**Writing and Understanding Proofs**

• Proof methods and proof strategies are now treated in

conjectures, is illustrated using the easily accessible separate sections of Chapter 1.

topic of tilings of checkerboards.

• An appendix listing basic axioms for real numbers and for the integers, and how these axioms are used to prove new results, has been added. The use of these axioms and basic results that follow from them has been made explicit in many proofs in the text.

• Separate and expanded sections on mathematical in- duction and on strong induction begin the new Chapter 4. These sections include more motivation and a rich collection of examples, providing many examples dif- ferent than those usually seen.

• More proofs are displayed in a way that makes it pos-

• The process of making conjectures, and then using

sible to explicitly list the reason for each step in the different proof methods and strategies to attack these

proof.

**Algorithms and Applications**

• More coverage is devoted to the use of strong induc-

• Examples and exercises from computational geometry tion to prove that recursive algorithms are correct.

have been added, including triangulations of polygons.

• How Bayes' Theorem can be used to construct spam

• The application of bipartite graphs to matching prob- filters is now described.

lems has been introduced.

**Number Theory, Combinatorics, and Probability Theory**

• Coverage of number theory is now more flexible, with four sections covering different aspects ofthe subject and with coverage of the last three of these sections optional.

• The introduction of basic counting techniques, and permutations and combinations, has been enhanced.

**Graphs and Theory of Computation**

• The introduction to graph theory has been streamlined and improved.

• A quicker introduction to terminology and applica- tions is provided, with the stress on making the correct decisions when building a graph model rather than on terminology.

• Material on bipartite graphs and their applications has been expanded.

• Coverage of counting techniques has been expanded; counting the ways in which objects can be distributed in boxes is now covered.

• Coverage of probability theory has been expanded with the introduction of a new section on Bayes' Theorem.

• Examples illustrating the construction of finite-state automata that recognize specified sets have been added.

• Minimization of finite-state machines is now men- tioned and developed in a series of exercises.

• Coverage of Turing machines has been expanded with a brief introduction to how Turing machines arise in the study of computational complexity, decidability, and computability.

x Preface

Exercises and Examples

• Many new routine exercises and examples have been added throughout, especially at spots where key con- cepts are introduced.

• Extra effort has been made to ensure that both odd- numbered and even-numbered exercises are provided for basic concepts and skills.

• A better correspondence has been made between examples introducing key concepts and routine exercises.

• Many new challenging exercises have been added.

• Over 400 new exercises have been added, with more on key concepts, as well as more introducing new topics.

Additional Biographies, Historical Notes, and New Discoveries

• Biographies have been added for Archimedes, Hop- per, Stirling, and Bayes.

• Many biographies found in the previous edition have been enhanced, including the biography of Augusta Ada.

• The historical notes included in the main body of the book and in the footnotes have been enhanced.

• New discoveries made since the publication of the pre- vious edition have been noted.

The MathZone Companion Website *(www.mhhe.comlrosen)*

• MathZone course management and online tutorial sys- tem now provides homework and testing questions tied directly to the text.

• Expanded annotated links to hundreds of Internet resources have been added to the Web Resources Guide.

• Additional Extra Examples are now hosted online, covering all chapters of the book. These Extra Examples have benefited from user review and feedback.

**Special Features**

• Additional Self Assessments of key topics have been added, with 14 core topics now addressed.

• Existing Interactive Demonstration Applets support- ing key algorithms are improved. Additional applets have also been developed and additional explanations are given for integrating them with the text and in the classroom.

• An updated and expanded *Exploring Discrete Mathe- matics with Maple* companion workbook is also hosted online.

ACCESSIBILITY This text has proved to be easily read and understood by beginning students. There are no mathematical prerequisites beyond college algebra for almost all of this text. Students needing extra help will find tools on the MathZone companion website for bringing their mathematical maturity up to the level of the text. The few places in the book where calculus is referred to are explicitly noted. Most students should easily understand the pseudocode used in the text to express algorithms, regardless of whether they have formally studied programming languages. There is no formal computer science prerequisite.

Each chapter begins at an easily understood and accessible level. Once basic mathematical concepts have been carefully developed, more difficult material and applications to other areas of study are presented.

FLEXIBILITY This text has been carefully designed for flexible use. The dependence of chapters on previous material has been minimized. Each chapter is divided into sections of approximately the same length, and each section is divided into subsections that form natural blocks of material for teaching. Instructors can easily pace their lectures using these blocks.

Preface xi

WRITING STYLE The writing style in this book is direct and pragmatic. Precise mathe- mati cal language is used without excessive formalism and abstraction. Care has been taken to balance the mix of notation and words in mathematical statements.

MATHEMATICAL RIGOR AND PRECISION All definitions and theorems in this text are stated extremely carefully so that students will appreciate the precision of language and rigor needed in mathematics. Proofs are motivated and developed slowly; their steps are all carefully justified. The axioms used in proofs and the basic properties that follow from them are explicitly described in an appendix, giving students a clear idea of what they can assume in a proof. Recursive definitions are explained and used extensively.

WORKED EXAMPLES Over 750 examples are used to illustrate concepts, relate different topics, and introduce applications. In most examples, a question is first posed, then its solution is presented with the appropriate amount of detail.

APPLICATIONS The applications included in this text demonstrate the utility of discrete mathematics in the solution of real-world problems. This text includes applications to a wide va- riety of areas, including computer science, data networking, psychology, chemistry, engineering, linguistics, biology, business, and the Internet.

ALGORITHMS Results in discrete mathematics are often expressed in terms of algo- rithms; hence, key algorithms are introduced in each chapter of the book. These algorithms are expressed in words and in an easily understood form of structured pseudocode, which is described and specified in Appendix A.3. The computational complexity of the algorithms in the text is also analyzed at an elementary level.

HISTORICAL INFORMATION The background of many topics is succinctly described in the text. Briefbiographies of more than 65 mathematicians and computer scientists, accom- panied by photos or images, are included as footnotes. These biographies include information about the lives, careers, and accomplishments of these important contributors to discrete mathe- matics and images of these contributors are displayed. In addition, numerous historical footnotes are included that supplement the historical information in the main body of the text. Efforts have been made to keep the book up-to-date by reflecting the latest discoveries.

KEY TERMS AND RESULTS A list of key terms and results follows each chapter. The key terms include only the most important that students should learn, not every term defined in the chapter.

EXERCISES There are over 3800 exercises in the text, with many different types of ques- tions posed. There is an ample supply of straightforward exercises that develop basic skills, a large number of intermediate exercises, and many challenging exercises. Exercises are stated clearly and unambiguously, and all are carefully graded for level of difficulty. Exercise sets contain special discussions that develop new concepts not covered in the text, enabling students to discover new ideas through their own work.

Exercises that are somewhat more difficult than average are marked with a single star \*; those that are much more challenging are marked with two stars \*\*. Exercises whose solutions require calculus are explicitly noted. Exercises that develop results used in the text are clearly identified with the symbol G>. Answers or outlined solutions to all odd-numbered exercises are provided at the back of the text. The solutions include proofs in which most of the steps are clearly spelled out.

REVIEW QUESTIONS A set of review questions is provided at the end of each chapter. These questions are designed to help students focus their study on the most important concepts

Di Preface

and techniques of that chapter. To answer these questions students need to write long answers, rather than just perform calculations or give short replies.

SUPPLEMENTARY EXERCISE SETS Each chapter is followed by a rich and varied set of supplementary exercises. These exercises are generally more difficult than those in the exercise sets following the sections. The supplementary exercises reinforce the concepts of the chapter and integrate different topics more effectively.

COMPUTER PROJECTS Each chapter is followed by a set of computer projects. The approximately 150 computer projects tie together what students may have learned in computing and in discrete mathematics. Computer projects that are more difficult than average, from both a mathematical and a programming point of view, are marked with a star, and those that are extremely challenging are marked with two stars.

COMPUTATIONS AND EXPLORATIONS A set of computations and explorations is included at the conclusion of each chapter. These exercises (approximately 100 in total) are designed to be completed using existing software tools, such as programs that students or instructors have written or mathematical computation packages such as Maple or Mathematica. Many of these exercises give students the opportunity to uncover new facts and ideas through computation. (Some of these exercises are discussed in the *Exploring Discrete Mathematics with Maple* companion workbook available online.)

WRITING PROJECTS Each chapter is followed by a set of writing projects. To do these projects students need to consult the mathematical literature. Some of these projects are historical in nature and may involve looking up original sources. Others are designed to serve as gateways to new topics and ideas. All are designed to expose students to ideas not covered in depth in the text. These projects tie mathematical concepts together with the writing process and help expose students to possible areas for future study. (Suggested references for these projects can be found online or in the printed *Student's Solutions Guide.)*

APPENDIXES There are three appendixes to the text. The first introduces axioms for real numbers and the integers, and illustrates how facts are proved directly from these axioms. The second covers exponential and logarithmic functions, reviewing some basic material used heavily in the course. The third specifies the pseudocode used to describe algorithms in this text.

SUGGESTED READINGS A list of suggested readings for each chapter is provided in a section at the end of the text. These suggested readings include books at or below the level of this text, more difficult books, expository articles, and articles in which discoveries in discrete mathematics were originally published. Some of these publications are classics, published many years ago, while others have been published within the last few years.

**How to Use This Book**

This text has been carefully written and constructed to support discrete mathematics courses at several levels and with differing foci. The following table identifies the core and optional sections. An introductory one-term course in discrete mathematics at the sophomore level can be based on the core sections of the text, with other sections covered at the discretion of the instructor. A two-term introductory course can include all the optional mathematics sections in addition to the core sections. A course with a strong computer science emphasis can be taught by covering some or all of the optional computer science sections.

**Ancillaries**

Preface xiii

*Optional Computer*

*Optional Mathematics Chapter*

*Core Sections*

*Science Sections*

*Sections*

1 1.1-1. 7 (as needed) 2 2.1-2.4 (as needed) 3 3.1-3.5,3.8 (as needed)

3.6

3.7 4 4.1--4.3

4.4,4.5 5 5.1-5.3

5.6

5.4,5.5 6 6.1 6.4

6.2,6.3 7 7.1, 7.5

7.3

7.2, 7.4, 7.6 8 8.1, 8.3, 8.5

8.2

8.4,8.6 9 9.1-9.5

9.6-9.8 10 10.1

10.2, 10.3

10.4, 10.5 11 11.1-11.4 12

12.1-12.5

Instructors using this book can adjust the level of difficulty of their course by choosing either to cover or to omit the more challenging examples at the end of sections, as well as the more challenging exercises. The dependence of chapters on earlier chapters is shown in the following chart.

Chapter I Chapter I 2 Chapter I 3 Chapter I 4 Chapter I 5 Chapler ---~I~ 6 Chapler 7 Chapler 8 Chapler 9 Chapter 11 Chapler 12

Chapler I 10

STUDENT'S SOLUTIONS GUIDE This student manual, available separately, contains full solutions to all odd-numbered problems in the exercise sets. These solutions explain why a particular method is used and why it works. For some exercises, one or two other possible approaches are described to show that a problem can be solved in several different ways. Suggested references for the writing projects found at the end of each chapter are also included in this volume. Also included are a guide to writing proofs and an extensive description of common mistakes students make in discrete mathematics, plus sample tests and a sample crib sheet for each chapter designed to help students prepare for exams.

*(ISBN-1O: 0-07-310779-4) (ISBN-J3: 978-0-07-310779-0)*

INSTRUCTOR'S RESOURCE GUIDE This manual, available by request for instructors, contains full solutions to even-numbered exercises in the text. Suggestions on how to teach the material in each chapter of the book are provided, including the points to stress in each section and how to put the material into perspective. It also offers sample tests for each chapter and a test

xiv Preface

bank containing over 1300 exam questions to choose from. Answers to all sample tests and test bank questions are included. Finally, several sample syllabi are presented for courses with differ- ing emphasis and student ability levels, and a complete section and exercise migration guide is included to help users of the fifth edition update their course materials to match the sixth edition.

*(ISBN-I0: 0-07-310781-6) (ISBN-l3: 978-0-07-310781-3)*

INSTRUCTOR'S TESTING AND RESOURCE CD An extensive test bank of more than 1300 questions using Brownstone Diploma testing software is available by request for use on Windows or Macintosh systems. Instructors can use this software to create their own tests by selecting questions of their choice or by random selection. They can also sort questions by section, difficulty level, and type; edit existing questions or add their own; add their own headings and instructions; print scrambled versions of the same test; export tests to word processors test bank, including or the Web; the and questions create and and their manage answers, course is grade included books. in the A printed *Instructor* version s *Resource* of this

*Guide.*

*(ISBN-1O: 0-07-310782-4) (ISBN-l3: 978-0-07-310782-0)*

**Acknowledgments**

I would like to thank the many instructors and students at a variety of schools who have used this book and provided me with their valuable feedback and helpful suggestions. Their input has made this a much better book than it would have been otherwise. I especially want to thank Jerrold Grossman, John Michaels, and George Bergman for their technical reviews of the sixth edition and their "eagle eyes," which have helped ensure the accuracy of this book. I also appreciate the help provided by all those who have submitted comments via the website.

I thank the reviewers of this sixth and the five previous editions. These reviewers have provided much helpful criticism and encouragement to me. I hope this edition lives up to their high expectations.

Reviewers for the Sixth Edition

Charles Ashbacher,

*Mount Mercy College* Ian Barlami,

*Rice University* George Bergman,

*University of California, Berkeley* David Berman,

*University of North Carolina, Wilmington* Miklos Bona,

*University of Florida* Prosenjit Bose,

*Carleton University* Kirby Brown,

*Polytechnic University* Michael *The Master* Button,

*s College*

Greg Cameron,

*Brigham Young University* John Carter,

*University of Toronto* Greg Chapman,

*Cosumnes River College* Chao-Kun Cheng,

*Virginia Commonwealth University* Thomas Cooper,

*Georgia Perimeter College, Lawrenceville* Barbara Cortzen,

*DePaul University* Daniel Cunningham,

*Buffalo State College* George Davis,

*University of Georgia*

Beverly Diamond,

*College of Charleston* Thomas Dunion,

*Atlantic Union College* Bruce Elenbogen,

*University of Michigan, Dearborn* Herbert Enderton,

*University of California, Los Angeles* Anthony Evans,

*Wright State University* Kim Factor,

*Marquette University* William Farmer,

*McMaster University* Li Feng,

*Albany State University*

Stephanie Fitchett,

*Florida Atlantic University* Jerry Fluharty,

*Eastern Shore Community College* Joel Fowler,

*Southern Polytechnic State University* William Gasarch,

*University of Maryland* Sudpito Ghosh,

*Colorado State University* Jerrold Grossman,

*Oakland University* Ruth Haas,

*Smith College* Patricia Hammer,

*Hollins University* Keith Harrow,

*Brooklyn College* Bert Hartnell,

*St. Mary* s *University* Julia Hassett,

*Oakton College* Kavita Hatwal,

*Portland Community College* John Helm,

*Radford University* Arthur Hobbs,

*Texas A&M University* Fannie Howell,

*Roanoke High School* WeiHu,

*Houghton College* Nan Jiang,

*University of South Dakota* Margaret Johnson,

*Stanford University*

Martin Jones,

*College of Charleston*

Tim Kearns,

*California Polytechnic State University, San Luis Obispo*

**Reviewers of Previous Editions**

Eric Allender,

*Rutgers University*

Stephen Andrilli,

*La Salle University*

Kendall Atkinson,

*University of Iowa, Iowa City*

Jonathan Knappenberger,

*LaSalle University* Ed Korntved,

*Northwest Nazarene University* Przemo Kranz,

*University of Mississippi* Loredana Lanzani,

*University of Arkansas* Yoonjin Lee,

*Smith College* Miguel Lerma,

*Northwestern University* Jason Levy,

*University of Hawaii* Lauren Lilly,

*Tabor College* Ekaterina Lioutikova, *Saint Joseph College* Vladimir Logvinenko,

*De Anza College* Joan Lukas,

*University of Massachusetts, Boston* Lester McCann,

*University of Arizona* Jennifer McNulty,

*University of Montana* John G. Michaels, *SUNY Brockport* Michael Oppedisano,

*Morrisville State College* Michael O'Sullivan,

*San Diego State University* Charles Parry,

*Virginia Polytechnic Institute* Linda Powers,

*Virginia Polytechnic Institute* Dan Pritikin,

*Miami University* Anthony Quas,

*University of Memphis* Eric Rawdon,

*Duquesne University*

Zhaojun Bai,

*University of California, Davis*

Jack R. Barone,

*Baruch College*

Klaus Bichteler,

*University of Texas, Austin*

Preface xv

Henry Ricardo,

*Medgar Evers College* Oskars Rieksts,

*Kutztown University* Stefan Robila,

*Montclair State University* Chris Rodger,

*Auburn University* Robert Rodman,

*North Carolina State University* Shai Simonson,

*Stonehill College* Barbara Smith,

*Cochise College* Wasin So,

*San Jose State University* Diana Staats,

*Dutchess Community College* Lorna Stewart,

*University of Alberta* Bogdan Suceava,

*California State University, Fullerton* Kathleen Sullivan,

*Seattle University* Laszlo Szekely,

*University of South Carolina* Daniel Tauritz,

*University of Missouri, Rolla* Don VanderJagt,

*Grand Valley State University* Fran Vasko,

*Kutztown University* Susan Wallace,

*University of North Florida* Zhenyuan Wang,

*University of Nebraska, Omaha*

Tom Wineinger,

*University of Wisconsin, Eau Claire* Charlotte Young,

*South Plains College*

Alfred E. Borm,

*Southwest Texas State University* Ken W Bosworth,

*University of Maryland* Lois Brady,

*California Polytechnic State University, San Luis Obispo*

**xvi** Preface

Scott Buffett,

*University of New Brunswick*

Russell Campbell,

*University of Northern Iowa* E. Rodney Canfield,

*University of Georgia* Kevin Carolan,

*Marist College*

Tim Carroll,

*Bloomsburg University* Kit C. Chan,

*Bowling Green State University*

Allan C. Cochran,

*University of Arkansas* Peter Collinge,

*Monroe Community College*

Ron Davis,

*Millersville University*

Nachum Dershowitz,

*University of Illinois, Urbana-Champaign*

Thomas Dowling,

*The Ohio State University*

Patrick C. Fischer,

*Vanderbilt University*

Jane Fritz,

*University of New Brunswick* Ladnor Geissinger,

*University of North Carolina*

Jonathan Goldstine,

*Pennsylvania State University*

Paul Gormley,

*Villanova University*

Brian Gray,

*Howard Community College*

Jonathan Gross,

*Columbia University*

Laxmi N. Gupta,

*Rochester Institute of Technology*

Daniel Gusfield,

*University of California, Davis*

David F. Hayes,

*San Jose State University*

XinHe,

*SUNY at Buffalo*

Donald Hutchison,

*Clackamas Community College*

Kenneth Johnson,

*North Dakota State University*

David Jonah,

*Wayne State University*

Akihiro Kanamori, *Boston University* W Thomas Kiley,

*George Mason University* Takashi Kimura,

*Boston University*

Nancy Kinnersley,

*University of Kansas* Gary Klatt,

*University of Wisconsin*

Nicholas Krier,

*Colorado State University* Lawrence S. Kroll,

*San Francisco State University* Shui F. Lam,

*California State University, Long Beach*

Robert Lavelle,

*lona College* Harbir Lamba,

*George Mason University* Sheau Dong Lang,

*University of Central Florida*

Cary Lee,

*Grossmont Community College* Yi-Hsin Liu,

*University of Nebraska, Omaha*

Stephen C. Locke,

*Florida Atlantic University*

George Luger,

*University of New Mexico*

David S. McAllister,

*North Carolina State University*

Robert McGuigan,

*Wesifield State College*

Michael Maller,

*Queens College*

Ernie Manes,

*University of Massachusetts*

Francis Masat,

*Glassboro State College*

J. M. Metzger,

*University of North Dakota*

Thomas D. Morley,

*Georgia Institute of Technology*

D. R. Morrison,

*University of New Mexico*

Ho KuenNg,

Ho KuenNg,

*San Jose State University* Timothy S. Norfolk, Sr. *University of Akron* Truc T. Nguyen,

*San Jose State University* Timothy S. Norfolk, Sr. *University of Akron* Truc T. Nguyen,

*San Jose State University* Timothy S. Norfolk, Sr. *University of Akron* Truc T. Nguyen,

*San Jose State University* Timothy S. Norfolk, Sr. *University of Akron* Truc T. Nguyen,

*San Jose State University* Timothy S. Norfolk, Sr. *University of Akron* Truc T. Nguyen,

*Bowling Green State University* George Novacky,

*Bowling Green State University* George Novacky,

*Bowling Green State University* George Novacky,

*Bowling Green State University* George Novacky,

*University of Pittsburgh* Jeffrey Nunemacher,

*University of Pittsburgh* Jeffrey Nunemacher,

*University of Pittsburgh* Jeffrey Nunemacher,

*University of Pittsburgh* Jeffrey Nunemacher,

*University of Pittsburgh* Jeffrey Nunemacher,

*University of Pittsburgh* Jeffrey Nunemacher,

*Ohio Wesleyan University* Jaroslav Opatrny,

*Ohio Wesleyan University* Jaroslav Opatrny,

*Ohio Wesleyan University* Jaroslav Opatrny,

*Ohio Wesleyan University* Jaroslav Opatrny,

*Concordia University*

*Concordia University*

*Concordia University*

*Concordia University*

Jonathan Pakianathan,

Jonathan Pakianathan,

*University of Rochester* Thomas W Parsons,

*University of Rochester* Thomas W Parsons,

*University of Rochester* Thomas W Parsons,

*University of Rochester* Thomas W Parsons,

*Hofstra University*

*Hofstra University*

*Hofstra University*

*Hofstra University*

Mary K. Prisco,

Mary K. Prisco,

*University of Wisconsin, Green Bay*

*University of Wisconsin, Green Bay*

*University of Wisconsin, Green Bay*

*University of Wisconsin, Green Bay*

Halina Przymusinska,

Halina Przymusinska,

*California Polytechnic State University, Pomona*

*California Polytechnic State University, Pomona*

*California Polytechnic State University, Pomona*

*California Polytechnic State University, Pomona*

Don Reichman,

Don Reichman,

*Mercer County Community College*

*Mercer County Community College*

*Mercer County Community College*

*Mercer County Community College*

Harold Reiter,

Harold Reiter,

Harold Reiter,

Harold Reiter,

*University of North Carolina*

*University of North Carolina*

Adrian Riskin,

Adrian Riskin,

Adrian Riskin,

Adrian Riskin,

*Northern Arizona State University*

*Northern Arizona State University*

Amy L. Rocha,

Amy L. Rocha,

Amy L. Rocha,

Amy L. Rocha,

*San Jose State University*

*San Jose State University*

Janet Roll,

Janet Roll,

Janet Roll,

Janet Roll,

*University of Findlay* Matthew J. Saltzman,

*University of Findlay* Matthew J. Saltzman,

*University of Findlay* Matthew J. Saltzman,

*University of Findlay* Matthew J. Saltzman,

*University of Findlay* Matthew J. Saltzman,

*University of Findlay* Matthew J. Saltzman,

*Clemson University* Alyssa Sankey,

*Clemson University* Alyssa Sankey,

*Clemson University* Alyssa Sankey,

*Clemson University* Alyssa Sankey,

*Clemson University* Alyssa Sankey,

*Clemson University* Alyssa Sankey,

*Slippery Rock University* Dinesh Sarvate,

*Slippery Rock University* Dinesh Sarvate,

*Slippery Rock University* Dinesh Sarvate,

*Slippery Rock University* Dinesh Sarvate,

*Slippery Rock University* Dinesh Sarvate,

*Slippery Rock University* Dinesh Sarvate,

*College of Charleston* Michael J. Schlosser,

*College of Charleston* Michael J. Schlosser,

*College of Charleston* Michael J. Schlosser,

*College of Charleston* Michael J. Schlosser,

*College of Charleston* Michael J. Schlosser,

*College of Charleston* Michael J. Schlosser,

*The Ohio State University*

*The Ohio State University*

Steven R. Seidel,

Steven R. Seidel,

Steven R. Seidel,

Steven R. Seidel,

*Michigan Technological University*

*Michigan Technological University*

Douglas Shier,

Douglas Shier,

Douglas Shier,

*Clemson University* Alistair Sinclair,

*Clemson University* Alistair Sinclair,

*Clemson University* Alistair Sinclair,

*Clemson University* Alistair Sinclair,

*University of California, Berkeley*

*University of California, Berkeley*

Carl H. Smith,

Carl H. Smith,

*University of Maryland*

*University of Maryland*

Hunter S. Snevily,

Hunter S. Snevily,

*University of Idaho*

*University of Idaho*

Preface xvii

Daniel Somerville, Philip D. Tiu, Thomas Upson,

*University 0/ Massachusetts, Boston Oglethorpe University Rochester Institute o/Technology* Patrick Tantalo, Roman Voronka,

*University o/California, Santa Cruz* Lisa Townsley-Kulich,

*Illinois Benedictine College New Jersey Institute o/Technology* Bharti Temkin, George Trapp, James Walker,

*Texas Tech University West Virginia University University 0/ South Carolina* Wallace Terwilligen, David S. Tucker, Anthony S. Wojcik,

*Bowling Green State University Midwestern State University Michigan State University*

I would like to thank the staff of McGraw-Hill for their strong support of this project. In particular, thanks go to Liz Haefele, Publisher, for her backing; to Liz Covello, Senior Sponsoring Editor, for her advocacy and enthusiasm; and to Dan Seibert, Developmental Editor, for his dedication and attention. I would also like to thank the original editor, Wayne Yuhasz, whose insights and skills helped ensure the book's success, as well as all the other previous editors of this book.

To the staff involved in the production of this sixth edition, I offer my thanks to Peggy Selle, Lead Project Manager; Michelle Whitaker, Senior Designer; JeffHuettman, Lead Media Producer; Sandy Schnee, Senior Media Project Manager; Melissa Leick, Supplements Coordi- nator; and Kelly Brown, Executive Marketing Manager. I am also grateful to Jerry Grossman, who checked the entire manuscript for accuracy; Rose Kramer and Gayle Casel, the technical proofreaders; Pat Steele, the manuscript copyeditor; and Georgia Mederer, who checked the accuracy of the solutions in the *Student's Solutions Guide* and *Instructor's Resource Guide.*

*Kenneth H. Rosen*

xviii

**The MathZone Companion Website**

MathZon~ '",

T he extensive companion website accompanying this text has been substantially enhanced lL for the sixth edition and is now integrated with MathZone, McGraw-Hill's customizable Web-based course management system. Instructors teaching from the text can now use Math- Zone to administer online homework, quizzing, testing, and student assessment through sets of questions tied to the text which are automatically graded and accessible in an integrated, exportable grade book. MathZone also provides a wide variety of interactive student tutorials, including practice problems with step-by-step solutions; Interactive Demonstration applets; and NetTutor, a live, personalized tutoring service.

This website is accessible at www.mathzone.com by choosing this text off the pull-down menu, or by navigating to www.mhhe.com/rosen. The homepage shows the *Information Center,* and contains login links for the site's *Student Edition* and *Instructor Edition.* Key features of each area are described below:

THE INFORMATION CENTER The Information Center contains basic information about the book including the Table of Contents, Preface, descriptions of the Ancillaries, and a sample chapter. Instructors can also learn more about MathZone's key features, and learn about custom publishing options for the book.

MATHZONE STUDENT EDITION The Student Edition contains a wealth of resources available for student use, including the following, tied into the text wherever the special icons below are shown in the text: a Extra Examples You can find a large number of additional examples on the site, covering all chapters of the book. These examples are concentrated in areas where students often ask for additional material. Although most of these examples amplify the basic concepts, more-challenging examples can also be found here. a Interactive Demonstration Applets These applets enable you to interactively explore how important algorithms work, and are tied directly to material in the text such as examples and exercises. Additional resources are provided on how to use and apply these applets. a Self Assessments These interactive guides help you assess your understanding of 14 key concepts, providing a question bank where each question includes a brief tutorial followed by a multiple-choice question. If you select an incorrect answer, advice is provided to help you understand your error. Using these Self Assessments, you should be able to diagnose your problems and focus on available remedies. a Web Resources Guide This guide provides annotated links to hundreds of external websites containing relevant material such as historical and biographical information, puzzles and problems, discussions, applets, programs, and more. These links are keyed to the text by page number.

Additional resources in the MathZone Student Edition include:

a *Exploring Discrete Mathematics with Maple* This ancillary provides help for using the Maple computer algebra system to do a wide range of computations in discrete mathematics. Each chapter provides a description of relevant Maple functions and how they are used, Maple programs to carry out computations in discrete mathematics, examples, and exercises that can be worked using Maple.

The MathZone Companion Website xix

*• Applications of Discrete Mathematics* This ancillary contains 24 chapters--each with its own set of exercises-presenting a wide variety of interesting and important applications covering three general areas in discrete mathematics: discrete structures, combinatorics, and graph theory. These applications are ideal for supplementing the text or for independent study.

*• A Guide to Proof-Writing* This guide provides additional help for writing proofs, a skill that many students find difficult to master. By reading this guide at the beginning of the course and periodically thereafter when proof writing is required, you will be rewarded as your proof-writing ability grows. (Also available in the *Student's Solutions Guide.)*

*• Common Mistakes in Discrete Mathematics* This guide includes a detailed list of com- mon misconceptions that students of discrete mathematics often have and the kinds of errors they tend to make. You are encouraged to review this list from time to time to help avoid these common traps. (Also available in the *Student's Solutions Guide.)*

*• Advice on Writing Projects* This guide offers helpful hints and suggestions for the Writ- ing Projects in the text, including an extensive bibliography of helpful books and articles for research; discussion of various resources available in print and online; tips on doing library re- search; and suggestions on how to write well. (Also available in the *Student's Solutions Guide.)*

• NetTutorTM NetTutor is a live, Web-based service that provides online tutorial help. Students can ask questions relating to this text and receive answers in real-time during regular, scheduled hours or ask questions and later receive answers.

MATHZONE INSTRUCTOR EDITION This part of the website provides access to all of the resources in the MathZone Student Edition, as well as the following resources for instructors:

• Course Management Using MathZone's course management capabilities instructors can create and share courses and assignments with colleagues; automatically assign and grade homework, quiz, and test questions from a bank of questions tied directly to the text; create and edit their own questions or import their own course content; manage course announcements and due dates; and track student progress.

• Instructor's Resource Library Collected here are a wide variety of additional instructor teaching resources, created by our wide community of users. These resources include lecture notes, classroom presentation materials such as transparency masters and PowerPoint slides, and supplementary reading and assignments. We will periodically update and add to this library, and encourage users to contact us with their materials and ideas for possible future additions.

*• Suggested Syllabi* Several detailed course outlines are shown, offering suggestions for courses with different emphases and different student backgrounds and ability levels.

*• Migration Guide* This guide provides instructors with a complete correlation of all sections and exercises from the fifth edition to the sixth edition, and is very useful for instructors migrating their syllabi to the sixth edition.

*• Teaching Suggestions* This guide contains detailed teaching suggestions for instructors, including chapter overviews for the entire text, detailed remarks on each section, and com- ments on the exercise sets.

• Printable Tests Printable tests are offered in Word and PDF format for every chapter, and are ideal for in-class exams. These tests were created from the Brownstone Diploma electronic test bank available separately by request on the *Instructor's Testing and Resource CD.*

• Image Bank PowerPoints Downloadable PowerPoints files are offered containing images of all Figures and Tables from the text. These image banks are ideal for preparing instructor class materials customized to the text.

xx

**To the Student**

*W hat is discrete mathematics?* Discrete mathematics is the part of mathematics devoted to the study of discrete objects. (Here *discrete* means consisting of distinct or unconnected elements.) The kinds of problems solved using discrete mathematics include:

a How many ways are there to choose a valid password on a computer system? a What is the probability of winning a lottery? a Is there a link between two computers in a network? a How can I identify spam e-mail messages? a How can I encrypt a message so that no unintended recipient can read it? a What is the shortest path between two cities using a transportation system? a How can a list of integers be sorted so that the integers are in increasing order? a How many steps are required to do such a sorting? a How can it be proved that a sorting algorithm correctly sorts a list? a How can a circuit that adds two integers be designed? a How many valid Internet addresses are there?

You will learn the discrete structures and techniques needed to solve problems such as these.

More generally, discrete mathematics is used whenever objects are counted, when relation- ships between finite (or countable) sets are studied, and when processes involving a finite number of steps are analyzed. A key reason for the growth in the importance of discrete mathematics is that information is stored and manipulated by computing machines in a discrete fashion.

**WHY** STUDY DISCRETE MATHEMATICS? There are several important reasons for studying discrete mathematics. First, through this course you can develop your mathematical maturity: that is, your ability to understand and create mathematical arguments. You will not get very far in your studies in the mathematical sciences without these skills.

Second, discrete mathematics is the gateway to more advanced courses in all parts of the mathematical sciences. Discrete mathematics provides the mathematical foundations for many computer science courses including data structures, algorithms, database theory, automata theory, formal languages, compiler theory, computer security, and operating systems. Students find these courses much more difficult when they have not had the appropriate mathematical foundations from discrete math. One student has sent me an e-mail message saying that she used the contents of this book in every computer science course she took!

Math courses based on the material studied in discrete mathematics include logic, set theory, number theory, linear algebra, abstract algebra, combinatorics, graph theory, and probability theory (the discrete part of the subject).

Also, discrete mathematics contains the necessary mathematical background for solving problems in operations research (including many discrete optimization techniques), chemistry, engineering, biology, and so on. In the text, we will study applications to some of these areas.

Many students find their introductory discrete mathematics course to be significantly more challenging than courses they have previously taken. One reason for this is that one of the primary goals of this course is to teach mathematical reasoning and problem solving, rather than a discrete set of skills. The exercises in this book are designed to reflect this goal. Although there are plenty of exercises in this text similar to those addressed in the examples, a large

To the Student xxi

percentage of the exercises require original thought. This is intentional. The material discussed in the text provides the tools needed to solve these exercises, but your job is to successfully apply these tools using your own creativity. One of the primary goals of this course is to learn how to attack problems that may be somewhat different from any you may have previously seen. Unfortunately, learning how to solve only particular types of exercises is not sufficient for success in developing the problem-solving skills needed in subsequent courses and professional work. This text addresses many different topics, but discrete mathematics is an extremely diverse and large area of study. One of my goals as an author is to help you develop the skills needed to master the additional material you will need in your own future pursuits.

THE EXERCISES I would like to offer some advice about how you can best learn discrete mathematics (and other subjects in the mathematical and computing sciences). You will learn the most by actively working exercises. I suggest that you solve as many as you possibly can. After working the exercises your instructor has assigned, I encourage you to solve additional exercises such as those in the exercise sets following each section of the text and in the supplementary exercises at the end of each chapter. (Note the key explaining the markings preceding exercises. )

Key to the Exercises

No marking \* \*\* *(Requires calculus)*

A routine exercise A difficult exercise An extremely challenging exercise An exercise containing a result used in the book (The Table below shows where each of these exercises are used.) An exercise whose solution requires the use of limits or concepts from differential or integral calculus

**Hand-Icon** Exercises **and** Where They Are Used

*Section Exercise Section Where Used Page Where Used*

1.2 42 11.2 758 1.6 81 1.6 81 2.3 75a 2.4 158 3.1 43 3.1 174 3.2 62 10.2 700 3.7 30 6.2 412 4.1 79 4.1 270 4.2 28 4.2 287 4.3 56 4.3 296 4.3 57 4.3 297 5.4 17 6.2 414 5.4 21 6.4 428 6.2 15 6.2 414 8.1 24 8.4 545 9.4 49 10.1 685 10.1 15 10.1 688 10.1 30 10.1 693 10.1 48 10.2 700 ILl 12 1l.3 763 A.2 4 7.3 477

xxii To the Student

The best approach is to try exercises yourself before you consult the answer section at the end of this book. Note that the odd-numbered exercise answers provided in the text are answers only and not full solutions; in particular, the reasoning required to obtain answers is omitted in these answers. The *Student* s *Solutions Guide,* available separately, provides complete, worked solutions to all odd-numbered exercises in this text. When you hit an impasse trying to solve an odd-numbered exercise, I suggest you consult the *Student* s *Solutions Guide* and look for some guidance as to how to solve the problem. The more work you do yourself rather than passively reading or copying solutions, the more you will learn. The answers and solutions to the even- numbered exercises are intentionally not available from the publisher; ask your instructor if you have trouble with these.

WEB RESOURCES You are *strongly* encouraged to take advantage of additional re- sources available on the Web, especially those on the MathZone companion website for this book found at www.mhhe.com/rosen. You will find many Extra Examples designed to clar- ifY key concepts; Self Assessments for gauging how well you understand core topics; Interactive Demonstration Applets exploring key algorithms and other concepts; a Web Resources Guide containing an extensive selection of links to external sites relevant to the world of discrete mathematics; extra explanations and practice to help you master core concepts; added instruc- tion on writing proofs and on avoiding common mistakes in discrete mathematics; in-depth discussions of important applications; and guidance on utilizing Maple software to explore the computational aspects of discrete mathematics. Places in the text where these additional online resources are available are identified in the margins by special icons. You will also find NetTutor, an online tutorial service that you can use to receive help from tutors either via real-time chat or via messages. For more details on these online resources, see the description of the MathZone companion website immediately preceding this "To the Student" message.

THE VALUE OF THIS BOOK My intention is to make your investment in this text an excellent value. The book, the associated ancillaries, and MathZone companion website have taken many years of effort to develop and refine. I am confident that most of you will find that the text and associated materials will help you master discrete mathematics. Even though it is likely that you will not cover some chapters in your current course, you should find it helpful- as many other students have-to read the relevant sections of the book as you take additional courses. Most of you will return to this book as a useful tool throughout your future studies, especially for those of you who continue in computer science, mathematics, and engineering. I have designed this book to be a gateway for future studies and explorations, and I wish you luck as you begin your journey.

*Kenneth H. Rosen*

CHAPTER

1.1 Propositional

Logic

1.2 Propositional Equivalences

1.3 Predicates and Quantifiers

1.4 Nested

Quantifiers

1.5 Rules of

Inference

1.6 Introduction to

Proofs

1.7 Proof Methods

and Strategy

**The Foundations: Logic and Proofs**

T 11. he help rules us understand of logic specify and reason the meaning with statements of mathematical such as statements. "There exists For an instance, integer these that is rules not the not sum of two exceeding *n* squares" is *n(n* + and 1)/2." "For Logic every is positive integer *n,* the sum of the positive integers the basis of all mathematical reasoning, and of all automated reasoning. It has practical applications to the design of computing machines, to the specification of systems, to artificial intelligence, to computer programming, to programming languages, and to other areas of computer science, as well as to many other fields of study.

To understand mathematics, we must understand what makes up a correct mathematical argument, that is, a proof. Once we prove a mathematical statement is true, we call it a theorem. A collection oftheorems on a topic organize what we know about this topic. To learn a mathematical topic, a person needs to actively construct mathematical arguments on this topic, and not just read exposition. Moreover, because knowing the proof of a theorem often makes it possible to modify the result to fit new situations, proofs play an essential role in the development of new ideas. Students of computer science often find it surprising how important proofs are in computer science. In fact, proofs play essential roles when we verify that computer programs produce the correct output for all possible input values, when we show that algorithms always produce the correct result, when we establish the security of a system, and when we create artificial intelligence. Automated reasoning systems have been constructed that allow computers to construct their own proofs.

In this chapter, we will explain what makes up a correct mathematical argument and intro- duce tools to construct these arguments. We will develop an arsenal of different proof methods that will enable us to prove many different types of results. After introducing many different methods of proof, we will introduce some strategy for constructing proofs. We will introduce the notion of a conjecture and explain the process of developing mathematics by studying conj ectures.

**1.1 Propositional Logic**

I-I

**Introduction**

Our discussion begins with an introduction to the basic building blocks of logic-propositions. A proposition is a declarative sentence (that is, a sentence that declares a fact) that is either true or false, but not both.

The rules of logic give precise meaning to mathematical statements. These rules are used to distinguish between valid and invalid mathematical arguments. Because a major goal of this book is to teach the reader how to understand and how to construct correct mathematical arguments, we begin our study of discrete mathematics with an introduction to logic.

In addition to its importance in understanding mathematical reasoning, logic has numerous applications in computer science. These rules are used in the design of computer circuits, the construction of computer programs, the verification of the correctness of programs, and in many other ways. Furthermore, software systems have been developed for constructing proofs automatically. We will discuss these applications of logic in the upcoming chapters.

**Propositions**

2 1 / The Foundations: Logic and Proofs 1-2

**EXAMPLEl** All the following declarative sentences are propositions.

Exam=~

1. Washington, D.C., is the capital of the United States of America. 2. Toronto is the capital of Canada. 3. 4. 2 1 + + 2 1 = = 2. 3.

Propositions 1 and 3 are true, whereas 2 and 4 are false.

Some sentences that are not propositions are given in Example 2.

**EXAMPLE** 2 Consider the following sentences.

unkS~

1. What time is it? 2. *3. 4. x x* Read + + *y* 1= this *=Z.* 2.

carefully.

Sentences 1 and 2 are not propositions because they are not declarative sentences. Sentences 3 and 4 are not propositions because they are neither true nor false. Note that each of sentences 3 and 4 can be turned into a proposition if we assign values to the variables. We will also discuss other ways to tum sentences such as these into propositions in Section 1.3. ~

We use letters to denote **propositional variables** (or **statement variables),** that is, variables that represent propositions, just as letters are used to denote numerical variables. The conven- tionalletters used for propositional variables are *p, q, r,* S, •••. The **truth value** ofa proposition is true, denoted by T, if it is a true proposition and false, denoted by F, if it is a false proposition. The area oflogic that deals with propositions is called the **propositional calculus or propo- sitional logic.** It was first developed systematically by the Greek philosopher Aristotle more than 2300 years ago.

ARISTOTLE (384 B.C.E.-322 B.C.E.) Aristotle was born in Stagirus (Stagira) in northern Greece. His father was the personal physician of the King of Macedonia. Because his father died when Aristotle was young, Aristotle could not follow the custom of following his father's profession. Aristotle became an orphan at a young age when his mother also died. His guardian who raised him taught him poetry, rhetoric, and Greek. At the age of 17, his guardian sent him to Athens to further his education. Aristotle joined Plato's Academy where for 20 years he attended Plato's lectures, later presenting his own lectures on rhetoric. When Plato died in 347 B.C. E., Aristotle was not chosen to succeed him because his views differed too much from those of Plato. Instead, Aristotle joined the court of King Hermeas where he remained for three years, and married the niece of the King. When the Persians defeated Hermeas, Aristotle moved to Mytilene and, at the invitation of King Philip of Macedonia, he tutored Alexander, Philip's son, who later became Alexander the Great. Aristotle tutored Alexander for five years and after the death of King Philip, he returned to Athens and set up his own school, called the Lyceum.

Aristotle's followers were called the peripatetics, which means "to walk about;' because Aristotle often walked around as he discussed philosophical questions. Aristotle taught at the Lyceum for 13 years where he lectured to his advanced students in the morning and gave popular lectures to a broad audience in the evening. When Alexander the Great died in 323 B.C.E., a backlash against anything related to Alexander led to trumped-up charges of impiety against Aristotle. Aristotle fled to Chalcis to avoid prosecution. He only lived one year in Chalcis, dying of a stomach ailment in 322 B.C.E.

Aristotle wrote three types of works: those written for a popular audience, compilations of scientific facts, and systematic treatises. The systematic treatises included works on logic, philosophy, psychology, physics, and natural history. Aristotle's writings were preserved by a student and were hidden in a vault where a wealthy book collector discovered them about 200 years later. They were taken to Rome, where they were studied by scholars and issued in new editions, preserving them for posterity.

1.1 1-3

Propositional Logic 3

lInkS~

We now tum our attention to methods for producing new propositions from those that we already have. These methods were discussed by the English mathematician George Boole in 1854 in his book *The Laws of Thought.* Many mathematical statements are constructed by combining one or more propositions. New propositions, called **compound propositions,** are formed from existing propositions using logical operators.

**DEFINITION 1**

Let *p* be a proposition. The *negation of p,* denoted by *""'p* (also denoted by *p),* is the statement

"It is not the case that *p."*

The proposition *""'p* is read "not *p."* The truth value ofthe negation of *p, ""'p,* is the opposite of the truth value of *p.*

**EXAMPLE** 3 Find the negation of the proposition

"Today is Friday." **Exam::** ~ and express this in simple English.

*Solution:* The negation is

"It is not the case that today is Friday."

This negation can be more simply expressed by

"Today is not Friday,"

or

"It is not Friday today."

**EXAMPLE** 4 Find the negation of the proposition

**TABLE 1 The Truth Table for the Negation of a Proposition.**

*p ...,p*

T F F T

"At least 10 inches of rain fell today in Miami."

***Remark:*** Strictly speaking, sentences involving variable times such as those in Examples 3 and 4 are not propositions unless a fixed time is assumed. The same holds for variable places unless a fixed place is assumed and for pronouns unless a particular person is assumed. We will always assume fixed times, fixed places, and particular people in such sentences unless otherwise noted . Table 1 displays the **truth table** for the negation of a proposition *p.* This table has a row for each of the two possible truth values of a proposition *p.* Each row shows the truth value of *-'p* corresponding to the truth value of *p* for this row.

and express this in simple English.

*Solution:* The negation is

"It is not the case that at least 10 inches of rain fell today in Miami."

This negation can be more simply expressed by

"Less than 10 inches of rain fell today in Miami."

4 1 / The Foundations: Logic and Proofs /-4

DEFINITION 2

The negation of a proposition can also be considered the result of the operation of the negation operator on a proposition. The negation operator constructs a new proposition from a single existing proposition. We will now introduce the logical operators that are used to form new propositions from two or more existing propositions. These logical operators are also called connectives.

Let *p* and *q* be propositions. The *conjunction* of *p* and *q,* denoted by *p* /\ *q,* is the proposition *"p* and *q."* The conjunction *p* /\ *q* is true when both *p* and *q* are true and is false otherwise.

Table 2 displays the truth table of *p* /\ *q.* This table has a row for each of the four possible combinations of truth values of *p* and *q.* The four rows correspond to the pairs of truth values TT, TF, FT, and FF, where the first truth value in the pair is the truth value of *p* and the second truth value is the truth value of *q.*

Note that in logic the word "but" sometimes is used instead of "and" in a conjunction. For example, the statement "The sun is shining, but it is raining" is another way of saying "The sun is shining and it is raining." (In natural language, there is a subtle difference in meaning between "and" and "but"; we will not be concerned with this nuance here.)

EXAMPLE 5 Find the conjunction of the propositions *p* and *q* where *p* is the proposition "Today is Friday"

and *q* is the proposition "It is raining today."

DEFINITION 3

*Solution:* The conjunction of these propositions, *p* /\ *q,* is the proposition "Today is Friday and it is raining today." This proposition is true on rainy Fridays and is false on any day that is not a Friday and on Fridays when it does not rain. ~

Let *p* and *q* be propositions. The *disjunction* of *p* and *q,* denoted by *p* V *q,* is the proposition *"p* or *q* ." The disjunction *p* V *q* is false when both *p* and *q* are false and is true otherwise.

Table 3 displays the truth table for *p* V *q.*

The use of the connective *or* in a disjunction corresponds to one of the two ways the word *or* is used in English, namely, in an inclusive way. A disjunction is true when at least one of the two propositions is true. For instance, the inclusive or is being used in the statement

"Students who have taken calculus or computer science can take this class."

TABLE 2 The Truth Table for TABLE 3 The Truth Table for the Conjunction of Two the Disjunction of Two Propositions. Propositions.

*p q p/\q p q pVq*

T T T T T T T F F T F T F T F F T T F F F F F F

1.1 1-5

Propositional Logic 5

Here, we mean that students who have taken both calculus and computer science can take the class, as well as the students who have taken only one of the two subjects. On the other hand, we are using the exclusive or when we say

"Students who have taken calculus or computer science, but not both, can enroll in this class."

Here, we mean that students who have taken both calculus and a computer science course cannot take the class. Only those who have taken exactly one of the two courses can take the class. Similarly, when a menu at a restaurant states, "Soup or salad comes with an entree," the restaurant almost always means that customers can have either soup or salad, but not both. Hence, this is an exclusive, rather than an inclusive, or.

EXAMPLE 6

What is the disjunction of the propositions *p* and *q* where *p* and *q* are the same propositions as in Example 5?

*Solution:* The disjunction of *p* and *q, p* v *q,* is the proposition

"Today is Friday or it is raining today."

This proposition is true on any day that is either a Friday or a rainy day (including rainy Fridays). It is only false on days that are not Fridays when it also does not rain. ....

As was previously remarked, the use of the connective *or* in a disjunction corresponds to one of the two ways the word *or* is used in English, namely, in an inclusive way. Thus, a disjunction is true when at least one of the two propositions in it is true. Sometimes, we use *or* in an exclusive sense. When the exclusive or is used to connect the propositions *p* and *q,* the proposition *"p* or *q* (but not both)" is obtained. This proposition is true when *p* is true and *q* is false, and when *p* is false and *q* is true. It is false when both *p* and *q* are false and when both are true.

**DEFINITION 4** Let *p* and *q* be propositions. The *exclusive or* of *p* and *q,* denoted by *p* E9 *q,* is the proposition that is true when exactly one of *p* and *q* is true and is false otherwise.

unkS~

The truth table for the exclusive or of two propositions is displayed in Table 4.

GEORGE BOOLE (1815-1864) George Boole, the son of a cobbler, was born in Lincoln, England, in November 1815. Because of his family's difficult financial situation, Boole had to struggle to educate himself while supporting his family. Nevertheless, he became one of the most important mathematicians of the 1800s. Although he considered a career as a clergyman, he decided instead to go into teaching and soon afterward opened a school of his own. In his preparation for teaching mathematics, Boole-unsatisfied with textbooks of his day-decided to read the works of the great mathematicians. While reading papers of the great French mathematician Lagrange, Boole made discoveries in the calculus of variations, the branch of analysis dealing with finding curves and surfaces optimizing certain parameters.

In 1848 Boole published *The Mathematical Analysis a/Logic,* the first of his contributions to symbolic logic. In 1849 he was appointed professor of mathematics at Queen's College in Cork, Ireland. In 1854 he published *The Laws a/Thought,* his most famous work. In this book, Boole introduced what is now called *Boolean algebra* in his honor. Boole wrote textbooks on differential equations and on difference equations that were used in Great Britain until the end ofthe nineteenth century. Boole married in 1855; his wife was the niece of the professor of Greek at Queen's College. In 1864 Boole died from pneumonia, which he contracted as a result of keeping a lecture engagement even though he was soaking wet from a rainstorm.

6 I / The Foundations: Logic and Proofs 1-6

DEFINITION 5

Assessment i::J

TABLE 4 The Truth Table for TABLE 5 The Truth Table for the Exclusive Or of Two the Conditional Statement Propositions. *p--q.*

Let *p* and *q* be propositions. The *conditional statement p* ~ *q* is the proposition "if *p,* then *q."* The conditional statement *p* ~ *q* is false when *p* is true and *q* is false, and true otherwise. In the conditional statement *p* ~ *q, p* is called the *hypothesis* (or *antecedent* or *premise)* and *q* is called the *conclusion* (or *consequence).*

The statement *p* ~ *q* is called a conditional statement because *p* ~ *q* asserts that *q* is true on the condition that *p* holds. A conditional statement is also called an implication.

*p q pffiq p q p-+q*

T T F T T T T F T T F F F T T F T T F F F F F T

Conditional Statements

We will discuss several other important ways in which propositions can be combined.

The truth table for the conditional statement *p* ~ *q* is shown in Table 5. Note that the statement *p* ~ *q* is true when both *p* and *q* are true and when *p* is false (no matter what truth value *q* has).

Because conditional statements play such an essential role in mathematical reasoning, a variety of terminology is used to express *p* ~ *q.* You will encounter most if not all of the following ways to express this conditional statement:

"if *p,* then *q"* "if *p, q" "p* is sufficient for *q" "q* if *p" "q* when *p"* "a necessary condition for *p* is *q" "q* unless *-'p"*

*"p* implies *q" "p* only if *q"* "a sufficient condition for *q* is *p" "q* whenever *p" "q* is necessary for *p" "q* follows from *p"*

A useful way to understand the truth value of a conditional statement is to think of an obligation or a contract. For example, the pledge many politicians make when running for office is

"If I am elected, then I will lower taxes."

If the politician is elected, voters would expect this politician to lower taxes. Furthermore, if the politician is not elected, then voters will not have any expectation that this person will lower taxes, although the person may have sufficient influence to cause those in power to lower taxes. It is only when the politician is elected but does not lower taxes that voters can say that the politician has broken the campaign pledge. This last scenario corresponds to the case when *p* is true but *q* is false in *p* ~ *q.*

Similarly, consider a statement that a professor might make:

"If you get 100% on the final, then you will get an A."

/-7

EXAMPLE 7

Exlra~ ExampleS~

1.1 Propositional Logic 7

If you manage to get a 100% on the final, then you would expect to receive an A. If you do not get 100% you mayor may not receive an A depending on other factors. However, if you do get 100%, but the professor does not give you an A, you will feel cheated.

Let *p* be the statement "Maria learns discrete mathematics" and *q* the statement "Maria will find a good job." Express the statement *p* -+ *q* as a statement in English.

*Solution:* From the definition of conditional statements, we see that when *p* is the statement "Maria learns discrete mathematics" and *q* is the statement "Maria will find a good job," *p* -+ *q* represents the statement

Many people find it confusing that *"p* only if *q"* expresses the same thing as "if *p* then *q."* To remember this, note that *"p* only if *q"* says that *p* cannot be true when *q* is not true. That is, the statement is false if *p* is true, but *q* is false. When *p* is false, *q* may be either true or false, because the statement says nothing about the truth value of *q.* A common error is for people to think that *"q* only if *p"* is a way of expressing *p* -+ *q.* However, these statements have different truth values when *p* and *q* have different truth values.

The word "unless" is often used to express conditional statements. Observe that *"q* unless *-.p"* means that if *-'p* is false, then *q* must be true. That is, the statement *"q* unless *-.p"* is false when *p* is true and *q* is false, but it is true otherwise. Consequently, *"q* unless *-.p"* and *p* -+ *q* always have the same truth value.

We illustrate the translation between conditional statements and English statements in Example 7.

"If Maria learns discrete mathematics, then she will find a good job."

There are many other ways to express this conditional statement in English. Among the most natural of these are:

"Maria will find a good job when she learns discrete mathematics."

"For Maria to get a good job, it is sufficient for her to learn discrete mathematics."

and

"Maria will find a good job unless she does not learn discrete mathematics."

Note that the way we have defined conditional statements is more general than the meaning attached to such statements in the English language. For instance, the conditional statement in Example 7 and the statement

"If it is sunny today, then we will go to the beach."

are statements used in normal language where there is a relationship between the hypothesis and the conclusion. Further, the first of these statements is true unless Maria learns discrete mathematics, but she does not get a good job, and the second is true unless it is indeed sunny today, but we do not go to the beach. On the other hand, the statement

"If today is Friday, then 2 + 3 = 5."

is true from the definition of a conditional statement, because its conclusion is true. (The truth value of the hypothesis does not matter then.) The conditional statement

"If today is Friday, then 2 + 3 = 6." is true every day except Friday, even though 2 + 3 = 6 is false.

We would not use these last two conditional statements in natural language (except perhaps in sarcasm), because there is no relationship between the hypothesis and the conclusion in either

8 I / The Foundations: Logic and Proofs 1-8

statement. In mathematical reasoning, we consider conditional statements of a more general sort than we use in English. The mathematical concept of a conditional statement is independent of a cause-and-effect relationship between hypothesis and conclusion. Our definition of a conditional statement specifies its truth values; it is not based on English usage. Propositional language is an artificial language; we only parallel English usage to make it easy to use and remember.

The if-then construction used in many programming languages is different from that used in logic. Most programming languages contain statements such as if *p* then S, where *p* is a . proposition and S is a program segment (one or more statements to be executed). When execution of a program encounters such a statement, S is executed if *p* is true, but S is not executed if *p* is false, as illustrated in Example 8.

EXAMPLE 8 What is the value of the variable *x* after the statement

if 2 + 2 = 4 then *x* := *x* + I

if *x* = 0 before this statement is encountered? (The symbol := stands for assignment. The statement *x* : = *x* + 1 means the assignment of the value of *x* + 1 to *x.)*

*Solution:* Because 2 + 2 = 4 is true, the assignment statement *x* := *x* + 1 is executed. Hence, *x* has the value 0 + 1 = 1 after this statement is encountered. ....

CONVERSE, CONTRAPOSITIVE, AND INVERSE We can form some new conditional statements starting with a conditional statement *p* -+ *q.* In particular, there are three related conditional statements that occur so often that they have special names. The proposition *q* -+ *p* is called the converse of *p* -+ *q.* The contrapositive of *p* -+ *q* is the proposition *-.q* -+ *-'p.* The proposition *-'p* -+ *-.q* is called the inverse of *p* -+ *q.* We will see that of these three conditional statements formed from *p* -+ *q,* only the contrapositive always has the same truth value as *p* -+ *q .* .. We first show that the contrapositive, *-.q* -+ *-.p,* of a conditional statement *p* -+ *q* always has the same truth value as *p* -+ *q.* To see this, note that the contrapositive is false only when *-'p* is false and *-.q* is true, that is, only when *p* is true and *q* is false. We now show that neither the converse, *q* -+ *p,* nor the inverse, *-'p* -+ *-'q,* has the same truth value as *p* -+ *q* for all possible truth values of *p* and *q.* Note that when *p* is true and *q* is false, the original conditional statement is false, but the converse and the inverse are both true.

When two compound propositions always have the same truth value we call them equivalent, so that a conditional statement and its contrapositive are equivalent. The converse and the inverse of a conditional statement are also equivalent, as the reader can verify, but neither is equivalent to the original conditional statement. (We will study equivalent propositions in Section 1.2.) Take note that one of the most common logical errors is to assume that the converse or the inverse of a conditional statement is equivalent to this conditional statement.

We illustrate the use of conditional statements in Example 9.

EXAMPLE 9 What are the contrapositive, the converse, and the inverse of the conditional statement

"The home team wins whenever it is raining."?

*Solution:* Because *"q* whenever *p"* is one of the ways to express the conditional statement *p* -+ *q,* the original statement can be rewritten as

"If it is raining, then the home team wins."

Consequently, the contrapositive of this conditional statement is

"If the home team does not win, then it is not raining."

1.1 1-9 Propositional Logic 9

The converse is

"If the home team wins, then it is raining."

The inverse is

"If it is not raining, then the home team does not win."

Only the contrapositive is equivalent to the original statement.

BICONDITIONALS We now introduce another way to combine propositions that expresses that two propositions have the same truth value.

DEFINITION 6

Let *p* and *q* be propositions. The *biconditional statement p* \*+ *q* is the proposition *"p* if and only if *q."* The biconditional statement *p* \*+ *q* is true when *p* and *q* have the same truth values, and is false otherwise. Biconditional statements are also called *bi-implications.*

The truth table for *p* \*+ *q* is shown in Table 6. Note that the statement *p* \*+ *q* is true when both the conditional statements *p* ~ *q* and *q* ~ *p* are true and is false otherwise. That is why we use the words "if and only if" to express this logical connective and why it is symbolically written by combining the symbols ~ and +-. There are some other common ways to express *p* \*+ *q:*

*"p* is necessary and sufficient for *q"* "if *p* then *q,* and conversely" *"p* iff *q."*

The last way of expressing the biconditional statement *p* \*+ *q* uses the abbreviation "iff" for "if and only if." Note that *p* \*+ *q* has exactly the same truth value as *(p* -+ *q) 1\ (q* ~ *p).*

EXAMPLE 10 Let *p* be the statement "You can take the flight" and let *q* be the statement "You buy a ticket."

Then *p* \*+ *q* is the statement

"You can take the flight if and only if you buy a ticket."

This statement is true if *p* and *q* are either both true or both false, that is, if you buy a ticket and can take the flight or if you do not buy a ticket and you cannot take the flight. It is false when *p* and *q* have opposite truth values, that is, when you do not buy a ticket, but you can take the flight (such as when you get a free trip) and when you buy a ticket and cannot take the flight (such as when the airline bumps you). ....

TABLE 6 The Truth Table for the Biconditional *p* +-+ *q.*

*p q p+-+q*

T T T T F F F T F F F T

10 1/ The Foundations: Logic and Proofs *1-10*

Demo~

EXAMPLE 11

IMPLICIT USE OF BICONDITIONALS You should be aware that biconditionals are not always explicit in natural language. In particular, the "if and only if" construction used in biconditionals is rarely used in common language. Instead, biconditionals are often expressed using an "if, then" or an "only if" construction. The other part of the "if and only if" is implicit. That is, the converse is implied, but not stated. For example, consider the statement in English "If you finish your meal, then you can have dessert." What is really meant is "You can have dessert if and only if you finish your meal." This last statement is logically equivalent to the two statements "If you finish your meal, then you can have dessert" and "You can have dessert only if you finish your meal." Because of this imprecision in natural language, we need to make an assumption whether a conditional statement in natural language implicitly includes its converse. Because precision is essential in mathematics and in logic, we will always distinguish between the conditional statement *p* -+ *q* and the biconditional statement *p* ++ *q.*

We have now introduced four important logical connectives--conjunctions, disjunctions, con- ditional statements, and biconditional statements-as well as negations. We can use these con- nectives to build up complicated compound propositions involving any number of propositional variables. We can use truth tables to determine the truth values of these compound propositions, as Example 11 illustrates. We use a separate column to find the truth value of each compound expression that occurs in the compound proposition as it is built up. The truth values of the compound proposition for each combination of truth values of the propositional variables in it is found in the final column of the table.

Construct the truth table of the compound proposition

Truth Tables of Compound Propositions

*(p* v *--.q)* -+ *(p* /\ *q).*

*Solution:* Because this truth table involves two propositional variables *p* and *q,* there are four rows in this truth table, corresponding to the combinations of truth values TT, TF, FT, and FE column The we first find two the columns truth value are used of *--'q,* for the needed truth to values find of the *p* truth and *q,* value respectively. of *p* v *--'q,* In found the third in the value fourth of *(p* column. v *--.q)* -+ The *(p* truth /\ *q)* is value found of in *p* the /\ *q* last is found column. in The the resulting fifth column. truth Finally, table is shown the truth in Table 7. ~

Precedence of Logical Operators

We can construct compound propositions using the negation operator and the logical operators defined so far. We will generally use parentheses to specify the order in which logical operators

TABLE 7 The Truth Table of (p v .., *q)* -+ (p /\ *q).*

*p q ..,q pV"'q p/\q* (p V *-,q)* -+ (p /\ *q)*

T T F T T T T F T T F F F T F F F T F F T T F F

1.1 *1-11* Propositional Logic 11

TABLES Precedence of Logical Operators.

in a compound proposition are to be applied. For instance, *(p* V *q)* /\ *(--.y)* is the conjunction of *p* V *q* and *-.r.* However, to reduce the number of parentheses, we specify that the negation operator is applied before all other logical operators. This means that *-'p* /\ *q* is the conjunc- tion of *-'p* and *q,* namely, *(-'p)* /\ *q,* not the negation of the conjunction of *p* and *q,* namely *-'(p* /\ *q). Operator Precedence*

Another general rule of precedence is that the conjunction operator takes precedence over

~ 1

the disjunction operator, so that *p* /\ *q* V *r* means *(p* /\ *q)* V *r* rather than *p* /\ *(q* V *r).* Because this rule may be difficult to remember, we will continue to use parentheses so that the order of *1\* v 2

the disjunction and conjunction operators is clear. 3

Finally, it is an accepted rule that the conditional and biconditional operators ~ and ++

---+ \*+ 4 5

have lower precedence than the conjunction and disjunction operators, /\ and v. Consequently, *p* V *q* ~ *r* is the same as *(p* V *q)* ~ *r.* We will use parentheses when the order ofthe conditional operator and biconditional operator is at issue, although the conditional operator has precedence over -., /\, the v, biconditional ~, and ++. operator. Table 8 displays the precedence levels of the logical operators, Translating English Sentences

There are many reasons to translate English sentences into expressions involving propositional variables and logical connectives. In particular, English (and every other human language) is often ambiguous. Translating sentences into compound statements (and other types of logical expressions, which we will introduce later in this chapter) removes the ambiguity. Note that this may involve making a set of reasonable assumptions based on the intended meaning of the sentence. Moreover, once we have translated sentences from English into logical expressions we can analyze these logical expressions to determine their truth values, we can manipulate them, and we can use rules of inference (which are discussed in Section 1.5) to reason about them. To illustrate the process of translating an English sentence into a logical expression, consider Examples 12 and 13.

EXAMPLE 12 How can this English sentence be translated into a logical expression?

"You can access the Internet from campus only if you are a computer science major or you are not a freshman."

*Solution:* There are many ways to translate this sentence into a logical expression. Although it is possible to represent the sentence by a single propositional variable, such as *p,* this would not be useful when analyzing its meaning or reasoning with it. Instead, we will use propositional variables to represent each sentence part and determine the appropriate logical connectives between them. In particular, we let *a,* c, and / represent "You can access the Internet from campus," "You are a computer science major," and "You are a freshman," respectively. Noting that "only if" is one way a conditional statement can be expressed, this sentence can be repre- sented as

*a* ~ (c V -./).

EXAMPLE 13 How can this English sentence be translated into a logical expression?

"You cannot ride the roller coaster if you are under 4 feet tall unless you are older than 16 years old."

12 1 / The Foundations: Logic and Proofs 1-12

EXAMPLE 14

*Solution:* Let *q, r,* and s represent "You can ride the roller coaster," "You are under 4 feet tall," and "You are older than 16 years old," respectively. Then the sentence can be translated to

Express the specification "The automated reply cannot be sent when the file system is full" using logical connectives.

Of course, there are other ways to represent the original sentence as a logical expression, but the one we have used should meet our needs. ~

System Specifications

Translating sentences in natural language (such as English) into logical expressions is an essential part of specifying both hardware and software systems. System and software engineers take requirements in natural language and produce precise and unambiguous specifications that can be used as the basis for system development. Example 14 shows how compound propositions can be used in this process. .

*Solution:* One way to translate this is to let *p* denote "The automated reply can be sent" and *q* denote "The file system is full." Then *-,p* represents "It is not the case that the automated reply can be sent," which can also be expressed as "The automated reply cannot be sent." Consequently, our specification can be represented by the conditional statement *q* ~ *-'p.* ~

System specifications should be consistent, that is, they should not contain conflicting requirements that could be used to derive a contradiction. When specifications are not consistent, there would be no way to develop a system that satisfies all specifications.

EXAMPLE 15 Determine whether these system specifications are consistent:

"The diagnostic message is stored in the buffer or it is retransmitted." "The diagnostic message is not stored in the buffer." "If the diagnostic message is stored in the buffer, then it is retransmitted."

*Solution:* To determine whether these specifications are consistent, we first express them using logical expressions. Let *p* denote "The diagnostic message is stored in the buffer" and let *q* denote "The diagnostic message is retransmitted." The specifications can then be written as *p* V *q, -'p,* and *p* ~ *q.* An assignment of truth values that makes all three specifications true must have *p* false to make *-'p* true. Because we want *p* v *q* to be true but *p* must be false, *q* must be true. Because *p* ~ *q* is true when *p* is false and *q* is true, we conclude that these specifications are consistent because they are all true when *p* is false and *q* is true. We could come to the same conclusion by use of a truth table to examine the four possible assignments of truth values to *p* andq. ~

EXAMPLE 16 Do the system specifications in Example 15 remain consistent ifthe specification "The diagnostic

message is not retransmitted" is added?

*Solution:* By the reasoning in Example 15, the three specifications from that example are true only in the case when *p* is false and *q* is true. However, this new specification is *-'q,* which is false when *q* is true. Consequently, these four specifications are inconsistent. ~

*1-13*

unkS~

EXAMPLE 17

Exlra~ ExamPles~

unkS~

EXAMPLE 18 Exam=~

1.1 Propositional Logic 13

Boolean Searches

Logical connectives are used extensively in searches of large collections of information, such as indexes of Web pages. Because these searches employ techniques from propositional logic, they are called Boolean searches.

Web Page Searching Most Web search engines support Boolean searching techniques, which usually can help find Web pages about particular subjects. For instance, using Boolean searching to find Web pages about universities in New Mexico, we can look for pages matching NEW *AND* MEXICO *AND* UNIVERSITIES. The results of this search will include those pages that contain the three words NEW, MEXICO, and UNIVERSITIES. This will include all of the pages of interest, together with others such as a page about new universities in Mexico. (Note that in Google, and many other search engines, the word "AND" is not needed, although it is understood, because all search terms are included by default.) Next, to find pages that deal with universities in New Mexico or Arizona, we can search for pages matching (NEW *AND* MEXICO *OR* ARIZONA) *AND* UNIVERSITIES. *(Note:* Here the *AND* operator takes precedence over the *OR* operator. Also, in Google, the terms used for this search would be NEW MEXICO *OR* ARIZONA.) The results of this search will include all pages that contain the word UNIVERSITIES and either both the words NEW and MEXICO or the word ARIZONA. Again, pages besides those of interest will be listed. Finally, to find Web pages that deal with universities in Mexico (and not New Mexico), we might first look for pages matching MEXICO *AND* UNIVERSITIES, but because the results of this search will include pages about universities in New Mexico, as well as universities in Mexico, it might be better to search for pages matching (MEXICO *AND* UNIVERSITIES) *NOT* NEW. The results of this search include pages that contain both the words MEXICO and UNIVERSITIES but do not contain the word NEW. (In Google, and many other search engines, the word "NOT" is replaced by a minus sign "\_". In Google, the terms used for this last search would be MEXICO UNIVERSITIES - NEW.) ....

Puzzles that can be solved using logical reasoning are known as logic puzzles. Solving logic puzzles is an excellent way to practice working with the rules of logic. Also, computer pro- grams designed to carry out logical reasoning often use well-known logic puzzles to illustrate their capabilities. Many people enjoy solving logic puzzles, which are published in books and periodicals as a recreational activity.

In [Sm78] Smullyan posed many puzzles about an island that has two kinds of inhabitants, knights, who always tell the truth, and their opposites, knaves, who always lie. You encounter two people *A* and *B.* What are *A* and *B* if *A* says *"B* is a knight" and *B* says "The two of us are opposite types"?

In Boolean searches, the connective *AND* is used to match records that contain both of two search terms, the connective *OR* is used to match one or both of two search terms, and the connective *NOT* (sometimes written as *AND NOT)* is used to exclude a particular search term. Careful planning of how logical connectives are used is often required when Boolean searches are used to locate information of potential interest. Example 17 illustrates how Boolean searches are carried out.

Logic Puzzles

We will discuss two logic puzzles here. We begin with a puzzle that was originally posed by Raymond Smullyan, a master of logic puzzles, who has published more than a dozen books containing challenging puzzles that involve logical reasoning.

**14** 1 / The Foundations: Logic and Proofs 1-14

*Solution:* Let *p* and *q* be the statements that *A* is a knight and *B* is a knight, respectively, so that *-'p* and *-.q* are the statements that *A* is a knave and that *B* is a knave, respectively.

We first consider the possibility that *A* is a knight; this is the statement that *p* is true. If *A* is a knight, then he is telling the truth when he says that *B* is a knight, so that *q* is true, and *A* and *B* are the same type. However, if *B* is a knight, then *B's* statement that *A* and *B* are of opposite types, the statement *(p 1\ -.q)* V *(-'p 1\ q),* would have to be true, which it is not, because *A* and *B* are both knights. Consequently, we can conclude that *A* is not a knight, that is, that *p* is false. If *A* is a knave, then because everything a knave says is false, *A's* statement that *B* is a knight, that is, that *q* is true, is a lie, which means that *q* is false and *B* is also a knave. Furthermore, if *B* is a knave, then *B's* statement that *A* and *B* are opposite types is a lie, which is consistent with both *A* and *B* being knaves. We can conclude that both *A* and *B* are knaves. ....

We pose more of Smullyan's puzzles about knights and knaves in Exercises 55-59 at the end of this section. Next, we pose a puzzle known as the muddy children puzzle for the case of two children.

EXAMPLE 19 A father tells his two children, a boy and a girl, to play in their backyard without getting dirty.

unkS~

However, while playing, both children get mud on their foreheads. When the children stop playing, the father says "At least one of you has a muddy forehead," and then asks the children to answer "Yes" or "No" to the question: "Do you know whether you have a muddy forehead?" The father asks this question twice. What will the children answer each time this question is asked, assuming that a child can see whether his or her sibling has a muddy forehead, but cannot see his or her own forehead? Assume that both children are honest and that the children answer each question simultaneously.

*Solution:* Let s be the statement that the son has a muddy forehead and let *d* be the statement that the daughter has a muddy forehead. When the father says that at least one of the two children has a muddy forehead, he is stating that the disjunction s v *d* is true. Both children will answer "No" the first time the question is asked because each sees mud on the other child's forehead. That is, the son knows that *d* is true, but does not know whether s is true, and the daughter knows that s is true, but does not know whether *d* is true.

After the son has answered "No" to the first question, the daughter can determine that *d* must be true. This follows because when the first question is asked, the son knows that s v *d* is

RAYMOND SMULLYAN (BORN 1919) Raymond Smullyan dropped out of high school. He wanted to study what he was really interested in and not standard high school material. After jumping from one university to the next, he earned an undergraduate degree in mathematics at the University of Chicago in 1955. He paid his college expenses by performing magic tricks at parties and clubs. He obtained a Ph.D. in logic in 1959 at Princeton, studying under Alonzo Church. After graduating from Princeton, he taught mathematics and logic at Dartmouth College, Princeton University, Yeshiva University, and the City University of New York. He joined the philosophy department at Indiana University in 1981 where he is now an emeritus professor.

Smullyan has written many books on recreational logic and mathematics, including *Satan, Cantor, and Infinity; What Is the Name of This Book?; The Lady or the Tiger?; Alice* in *Puzzleland; To Mock a Mockingbird; Forever Undecided;* and *The Riddle of Scheherazade: Amazing Logic Puzzles, Ancient and Modern.* Because his logic puzzles are challenging, entertaining, and thought-provoking, he is considered to be a modern-day Lewis Carroll. Smullyan has also written several books about the application of deductive logic to chess, three collections of philosophical essays and aphorisms, and several advanced books on mathematical logic and set theory. He is particularly interested in self-reference and has worked on extending some of Godel's results that show that it is impossible to write a computer program that can solve all mathematical problems. He is also particularly interested in explaining ideas from mathematical logic to the public.

Smullyan is a talented musician and often plays piano with his wife, who is a concert-level pianist. Making telescopes is one of his hobbies. He is also interested in optics and stereo photography. He states "I've never had a conflict between teaching and research as some people do because when I'm teaching, I'm doing research."

1.1 1-15 Propositional Logic 15

TABLE 9 Table for the Bit Operators *OR, AND,* and *XOR.*

*x y xVy x/\y xEBy*

0 0 0 0 0 0 1 1 0 I 1 0 1 0 I 1 1 I 1 0

true, but cannot determine whether s is true. Using this information, the daughter can conclude that *d* must be true, for if *d* were false, the son could have reasoned that because s V *d* is true, then s must be true, and he would have answered "Yes" to the first question. The son can reason in a similar way to determine that s must be true. It follows that both children answer "Yes" the second time the question is asked. ....

Logic and Bit Operations

*Truth Value Bit*

o Computers (zero) and represent 1 (one). information This meaning using of the bits. word A bit bit is comes a symbol from with binary two *digit,* possible because values, zeros namely, and ones are the digits used in binary representations of numbers. The well-known statistician John T 1

Tukey introduced this terminology in 1946. A bit can be used to represent a truth value, because F 0

there are two truth values, namely, *true* and *false.* As is customarily done, we will use a I bit to represent true and a 0 bit to represent false. That is, I represents T (true), 0 represents F (false). A variable is called a Boolean variable if its value is either true or false. Consequently, unllS~

a Boolean variable can be represented using a bit.

Computer bit operations correspond to the logical connectives. By replacing true by a one and false by a zero in the truth tables for the operators /\, V, and EB, the tables shown in Table 9 for the corresponding bit operations are obtained. We will also use the notation *OR, AND,* and *XOR* for the operators v, /\, and EB, as is done in various programming languages.

Information is often represented using bit strings, which are lists of zeros and ones. When this is done, operations on the bit strings can be used to manipulate this information.

DEFINITION 7

A *bit string* is a sequence of zero or more bits. The *length* of this string is the number of bits in the string.

EXAMPLE 20 101010011 is a bit string oflength nine.

We can extend bit operations to bit strings. We define the bitwise *OR,* bitwise *AND,* and bitwise *XOR* of two strings of the same length to be the strings that have as their bits the *OR, AND,* and *XOR* of the corresponding bits in the two strings, respectively. We use the symbols v, /\, and EB to represent the bitwise *OR,* bitwise *AND,* and bitwiseXOR operations, respectively. We illustrate bitwise operations on bit strings with Example 21.

EXAMPLE 21 Find the bitwise *OR,* bitwise *AND,* and bitwise *XOR* of the bit strings 0110110110 and II 0001 1101. (Here, and throughout this book, bit strings will be split into blocks of four bits to make them easier to read.)

16 1 / The Foundations: Logic and Proofs 1-16

*Solution:* The bitwise *OR,* bitwise *AND,* and bitwiseXOR of these strings are obtained by taking the *OR, AND, andXOR* of the corresponding bits, respectively. This gives us

Exercises

01 10110110 11 0001 1101

11 1011 1111 bitwise *OR* 01 0001 0100 bitwise *AND* 10 1010 1011 *bitwiseXOR*

1. Which of these sentences are propositions? What are the

truth values ofthose that are propositions? a) Boston is the capital of Massachusetts. b) Miami is the capital of Florida. c) e) *x* 2 + + 3 2 = = 5. II. d) 5 + 7 = 10.

1) Answer this question. 2. Which ofthese are propositions? What are the truth values

ofthose that are propositions? a) Do not pass go. b) What time is it? c) There are no black flies in Maine. d) 4 *+x* = 5. e) The moon is made of green cheese. 1) *2n ::: 100.*

3. What is the negation of each of these propositions?

a) Today is Thursday.

unkS~

b) c) There 2 + 1 = is no 3.

pollution in New Jersey.

d) The summer in Maine is hot and sunny. 4. Let *p* and *q* be the propositions

*p* : I bought a lottery ticket this week. *q* : I won the million dollar jackpot on Friday. Express each of these propositions as an English sentence. a) *--'p* b) *p* v *q* c) *p* --+ *q* d) *p* /\ *q* e) *p* - *q* 1) *--'p* --+ *--.q* g) *--'p* /\ *--.q* b) *--'p* v *(p* /\ *q)* 5. Let *p* and *q* be the propositions "Swimming at the New Jersey shore is allowed" and "Sharks have been spotted near the shore," respectively. Express each of these com- pound propositions as an English sentence. a) *--.q* b) *p* /\ *q* c) *--'p* v *q* d) *p* --+ *--.q* e) *--.q* --+ *p* 1) *--'p* --+ *--.q* g) *p* \_ *--.q* b) *--'p* /\ *(p* V *--.q)*

JOHN WILDER TUKEY (1915-2000) Tukey, born in New Bedford, Massachusetts, was an only child. His parents, both teachers, decided home schooling would best develop his potential. His formal education began at Brown University, where he studied mathematics and chemistry. He received a master's degree in chemistry from Brown and continued his studies at Princeton University, changing his field of study from chemistry to mathematics. He received his Ph.D. from Princeton in 1939 for work in topology, when he was appointed an instructor in mathematics at Princeton. With the start of World War II, he joined the Fire Control Research Office, where he began working in statistics. Tukey found statistical research to his liking and impressed several leading statisticians with his skills. In 1945, at the conclusion ofthe war, Tukey returned to the mathematics department at Princeton as a professor of statistics, and he also took a position at AT&T Bell Laboratories. Tukey founded the Statistics Department at Princeton in 1966 and was its first chairman. Tukey made significant contributions to many areas of statistics, including the analysis of variance, the estimation of spectra of time series, inferences about the values of a set of parameters from a single experiment, and the philosophy of statistics. However, he is best known for his invention, with 1. W Cooley, ofthe fast Fourier transform. In addition to his contributions to statistics, Tukey was noted as a skilled wordsmith; he is credited with coining the terms *bit* and *software.*

Tukey contributed his insight and expertise by serving on the President's Science Advisory Committee. He chaired several important committees dealing with the environment, education, and chemicals and health. He also served on committees working on nuclear disarmament. Tukey received many awards, including the National Medal of Science.

HISTORICAL NOTE There were several other suggested words for a binary digit, including *binit* and *bigit,* that never were widely accepted. The adoption of the word *bit* may be due to its meaning as a common English word. For an account of Tukey's coining of the word *bit,* see the April 1984 issue of *Annals of the History of Computing.*

1-17

6. Let *p* and *q* be the propositions "The election is de- cided" and "The votes have been counted," respectively. Express each of these compound propositions as an English sentence.

1.1 Propositional Logic 17

Write these propositions using *p, q,* and *r* and logical connectives. a) You get an A in this class, but you do not do every

exercise in this book. b) You get an A on the final, you do every exercise in this

a) *-'p* e) *-'p 1\ q* e) *-.q* --+ *-'p* g) *p++q*

b) *p* vq d) *q* --+ *P* I) *-'p* --+ *-.q* b) *-.q* v *(-.p 1\ q)*

7. Let *p* and *q* be the propositions

*p* : It is below freezing. *q* : It is snowing.

exercise in this book. b) You get an A on the final, you do every exercise in this

exercise in this book. b) You get an A on the final, you do every exercise in this

book, and you get an A in this class. e) To get an A in this class, it is necessary for you to get

book, and you get an A in this class. e) To get an A in this class, it is necessary for you to get

an A on the final. d) You get an A on the final, but you don't do every ex- ercise in this book; nevertheless, you get an A in this class. e) Getting an A on the final and doing every exercise in

an A on the final. d) You get an A on the final, but you don't do every ex- ercise in this book; nevertheless, you get an A in this class. e) Getting an A on the final and doing every exercise in

Write these propositions using *p* and *q* and logical connectives. a) It is below freezing and snowing. b) It is below freezing but not snowing. e) It is not below freezing and it is not snowing. d) It is either snowing or below freezing (or both). e) Ifit is below freezing, it is also snowing. I) It is either below freezing or it is snowing, but it is not

snowing if it is below freezing. g) That it is below freezing is necessary and sufficient

for it to be snowing. 8. Let *p, q,* and *r* be the propositions

*p* : You have the flu. *q* : You miss the final examination. *r* : You pass the course. Express each ofthese propositions as an English sentence. a) *p* --+ *q* b) *-.q* ++ *r* e) *q* --+ *-'r* d) *p* v *q* V *r* e) *(p* --+ *-.r)* v *(q* --+ *-.r)* I) *(p 1\ q)* V *(-.q 1\ r)*

9. Let *p* and *q* be the propositions

*p* : You drive over 65 miles per hour. *q* : You get a speeding ticket. Write these propositions using *p* and *q* and logical connectives. a) You do not drive over 65 miles per hour. b) You drive over 65 miles per hour, but you do not get a

speeding ticket. e) You will get a speeding ticket if you drive over

65 miles per hour. d) If you do not drive over 65 miles per hour, then you

will not get a speeding ticket. e) Driving over 65 miles per hour is sufficient for getting

a speeding ticket. I) You get a speeding ticket, but you do not drive over 65

miles per hour. g) Whenever you get a speeding ticket, you are driving

over 65 miles per hour. 10. Let *p, q,* and *r* be the propositions

*p* : You get an A on the final exam. *q* : You do every exercise in this book. *r* : You get an A in this class.

this book is sufficient for getting an A in this class. I) You will get an A in this class if and only if you either do every exercise in this book or you get an A on the final. 11. Let *p, q,* and *r* be the propositions

*p* : Grizzly bears have been seen in the area. *q* : Hiking is safe on the trail. *r* : Berries are ripe along the trail. Write these propositions using *p, q,* and *r* and logical connectives. a) Berries are ripe along the trail, but grizzly bears have

not been seen in the area. b) Grizzly bears have not been seen in the area and hiking on the trail is safe, but berries are ripe along the trail. e) If berries are ripe along the trail, hiking is safe if and

only if grizzly bears have not been seen in the area. d) It is not safe to hike on the trail, but grizzly bears have not been seen in the area and the berries along the trail are ripe. e) For hiking on the trail to be safe, it is necessary but not sufficient that berries not be ripe along the trail and for grizzly bears not to have been seen in the area. I) Hiking is not safe on the trail whenever grizzly bears have been seen in the area and berries are ripe along the trail. 12. Determine whether these biconditionals are true or false.

a) 2 + 2 = 4 if and only if 1 + 1 = 2. b) 1 + 1 = 2 if and only if2 + 3 = 4. e) 1 + 1 = 3 if and only if monkeys can fly. d) 0> 1 if and only if2 > 1. 13. Determine whether each of these conditional statements

is true or false. a) If 1 + 1 = 2, then 2 + 2 = 5. b) If 1 + 1 = 3, then 2 + 2 = 4. e) If! + 1 = 3, then 2 + 2 = 5. d) If monkeys can fly, then I + 1 = 3. 14. Determine whether each of these conditional statements

is true or false. a) If 1 + 1 = 3, then unicorns exists. b) If 1 + 1 = 3, then dogs can fly. e) If 1 + 1 = 2, then dogs can fly. d) If2 + 2 = 4, then 1 + 2 = 3.

18 I / The Foundations: Logic and Proofs

15. For each of these sentences, determine whether an inclu- sive or or an exclusive or is intended. Explain your answer. a) Coffee or tea comes with dinner. b) A password must have at least three digits or be at least

eight characters long. c) The prerequisite for the course is a course in number

theory or a course in cryptography. d) You can pay using U.S. dollars or euros. 16. For each ofthese sentences, determine whether an inclu- sive or or an exclusive or is intended. Explain your answer. a) Experience with C++ or Java is required. b) Lunch includes soup or salad. c) To enter the country you need a passport or a voter

registration card. d) Publish or perish. 17. For each of these sentences, state what the sentence means ifthe or is an inclusive or (that is, a disjunction) versus an exclusive or. Which ofthese meanings of or do you think is intended? a) To take discrete mathematics, you must have taken

calculus or a course in computer science. b) When you buy a new car from Acme Motor Company,

you get $2000 back in cash or a 2% car loan. c) Dinner for two includes two items from column A or

three items from column B. d) School is closed if more than 2 feet of snow falls or if

the wind chill is below -100. 18. Write each of these statements in the form "if *p,* then *q"* in English. *[Hint:* Refer to the list of common ways to express conditional statements provided in this section.] a) It is necessary to wash the boss's car to get promoted. b) Winds from the south imply a spring thaw. c) A sufficient condition for the warranty to be good is that you bought the computer less than a year ago. d) Willy gets caught whenever he cheats. e) You can access the website only if you pay a subscrip-

tion fee. I) Getting elected follows from knowing the right

people. g) Carol gets seasick whenever she is on a boat. 19. Write each of these statements in the form "if *p,* then *q"* in English. *[Hint:* Refer to the list of common ways to express conditional statements.] a) It snows whenever the wind blows from the northeast. b) The apple trees will bloom ifit stays warm for a week. c) That the Pistons win the championship implies that

they beat the Lakers. d) It is necessary to walk 8 miles to get to the top of

Long's Peak. e) To get tenure as a professor, it is sufficient to be

world-famous. I) If you drive more than 400 miles, you will need to buy

gasoline. g) Your guarantee is good only if you bought your CD

player less than 90 days ago. h) Jan will go swimming unless the water is too cold.

*I-Iii*

20. Write each ofthese statements in the form "if *p,* then *q"* in English. *[Hint:* Refer to the list of common ways to express conditional statements provided in this section.] a) I will remember to send you the address only if you send me an e-mail message. . b) To be a citizen ofthis country, it is sufficient that you

were born in the United States. c) If you keep your textbook, it will be a useful reference

in your future courses. d) The Red Wings will win the Stanley Cup iftheir goalie

plays well. e) That you get the job implies that you had the best

credentials. I) The beach erodes whenever there is a storm. g) It is necessary to have a valid password to log on to

the server. b) You will reach the summit unless you begin your climb

too late. 21. Write each of these propositions in the form *"p* if and

only if *q"* in English. a) If it is hot outside you buy an ice cream cone, and if

you buy an ice cream cone it is hot outside. b) For you to win the contest it is necessary and sufficient

that you have the only winning ticket. c) You get promoted only if you have connections, and

you have connections only if you get promoted. d) If you watch television your mind will decay, and

conversely. e) The trains run late on exactly those days when I

take it. 22. Write each of these propositions in the form *"p* if and

only if *q"* in English. a) For you to get an A in this course, it is necessary and sufficient that you learn how to solve discrete mathe- matics problems. b) If you read the newspaper every day, you will be in-

formed, and conversely. c) It rains if it is a weekend day, and it is a weekend day

if it rains. d) You can see the wizard only if the wizard is not in,

and the wizard is not in only if you can see him. 23. State the converse, contrapositive, and inverse of each of

these conditional statements. a) Ifit snows today, I will ski tomorrow. b) I come to class whenever there is going to be a quiz. c) A positive integer is a prime only if it has no divisors

other than I and itself. 24. State the converse, contrapositive, and inverse of each of

these conditional statements. a) If it snows tonight, then I will stay at home. b) I go to the beach whenever it is a sunny summer day. c) When I stay up late, it is necessary that I sleep until

noon. 25. How many rows appear in a truth table for each of these

compound propositions?

/-/9 a) *p* -+ *-'p*

b) *(p* v *-.r)* /\ *(q* V *-.s)* c) *q* v *p* v -.s V *-'r* V *-.t* V *u* d) *(p* /\ *r* /\ *t)* \_ *(q* /\ *t)* 26. How many rows appear in a truth table for each of these

compound propositions? a) *(q* -+ *-'p)* v *(-'P* -+ *-'q)* b) *(p* v *-.t)* /\ *(p* v *-'s)* c) *(p* -+ *r)* v *(-,s* -+ *-.t)* v *(-,u* -+ v) d) *(p* /\ *r* /\ s) V *(q* /\ *t)* V *(r* /\ *-.t)* 27. Construct a truth table for each of these compound

propositions. a) *p* /\ *-'p* b) *p* v *-'p* c) *(p* v *-.q)* -+ *q* d) *(p* v *q)* -+ *(p* /\ *q)* e) *(p* -+ *q)* \_ *(-.q* -+ *-'p)* f) *(p* -+ *q)* -+ *(q* -+ *p)* 28. Construct a truth table for each of these compound

propositions. a) *p* -+ *-'p* b) *p \_-'p* c) *pffi(pvq)* e) *(q* -+ *-'p)* \_ *(p* \_ *q)* d) *(p* /\ *q)* -+ *(p* v *q)* f) *(p* - *q)* ffi *(p* - *-.q)* 29. Construct a truth table for each of these compound

propositions. a) *(pvq)-+(pffiq)* c) *(pvq)ffi(p/\q)* e) *(p* \_ *q)* ffi *(-'P* \_ *-.r)* f) *(p* ffi *q)* -+ *(p* ffi *-.q)*

b) *(p* ffi *q)* -+ *(p* /\ *q)* d) *(p-q)ffi(-.p- q)*

30. Construct a truth table for each of these compound

propositions. a) *p* ffi *p* c) *p* ffi *-.q* e) *(p* ffi *q)* v *(p* ffi *-.q)*

b) *p* ffi *-'p* d) *-'p* ffi *-.q* f) *(p* ffi *q)* /\ *(p* ffi *-.q)* 31. Construct a truth table for each of these compound

propositions. a) *p* -+ *-.q* b) *-'p* \_ *q* c) *(p* -+ *q)* v *(-'P* -+ *q)* d) *(p* -+ *q)* /\ *(-'P* -+ *q)* e) *(p* \_ *q)* v *(-'P* \_ *q)*

f) *(-'P* - *-'q)* - *(p* - *q)* 32. Construct a truth table for each of these compound

propositions. a) *(p* v *q)* V *r* c) *(p/\q)vr* e) *(pvq)/\-'r*

b) *(pvq)/\r* d) *(p* /\ *q)* /\ *r* f) *(p* /\ *q)* V *-'r* 33. Construct a truth table for each of these compound

propositions. a) *p* -+ *(-.q* V *r)* b) *-'p* -+ *(q* -+ *r)* c) *(p* -+ *q)* v *(-'P* -+ *r)* d) e) *(p* -+ *(p* \_ *q) q)* v /\ *(-.q (-'P* \_ -+ *r) r)* f) *(-'P* - *-.q)* - *(q* - *r)* 34. Construct a truth table for *«p* -+ *q)* -+ *r)* -+ s.

1.1 Propositional Logic 19

35. Construct a truth table for *(p* \_ *q)* \_ *(r* \_ *s).* 36. What is the value of *x* after each of these statements is encountered in a computer program, if *x* = 1 before the statement is reached? a) if 1 + 2 = 3 then *x* := *x* + 1 b) if(1 + 1 = 3) *OR* (2 + 2 = 3) then *x* := *x* + 1 c) if (2 + 3 = 5) *AND* (3 + 4 = 7) then *x* := *x* + 1 d) if(1 + 1 = *2)XOR* (1 + 2 = 3) then *x* := *x* + 1 e) if *x* < 2 then *x* := *x* + 1 37. Find the bitwise *OR,* bitwise *AND,* and bitwise *XOR* of

each ofthese pairs of bit strings. a) 101 1110, 010 0001 b) 1111 0000, 1010 1010 c) 000111 0001, 1001001000 d) 11 1111 1111, 00 0000 0000 38. Evaluate each ofthese expressions. a) I 1000/\(0 1011 v 1 1011) b) (0 1111 /\ 1 0101) v 0 1000 c) (01010 ffi 1 1011) ffi 01000 d) (1 1011 v 0 1010) /\ (1 0001 v 1 1011) Fuzzy logic is used in artificial intelligence. In fuzzy logic, a proposition has a truth value that is a number between 0 and 1, inclusive. A proposition with a truth value of 0 is false and one with a truth value of 1 is true. Truth values that are between 0 and I indicate varying degrees of truth. For instance, the truth value 0.8 can be assigned to the statement "Fred is happy," because Fred is happy most of the time, and the truth value 0.4 can be assigned to the statement "John is happy," because John is happy slightly less than halfthe time. 39. The truth value of the negation of a proposition in fuzzy logic is 1 minus the truth value of the proposition. What are the truth values of the statements "Fred is not happy" and "John is not happy"? 40. The truth value of the conjunction of two propositions in fuzzy logic is the minimum of the truth values of the two propositions. What are the truth values of the statements "Fred and John are happy" and "Neither Fred nor John is happy"? 41. The truth value of the disjunction of two propositions in fuzzy logic is the maximum of the truth values of the two propositions. What are the truth values of the statements "Fred is happy, or John is happy" and "Fred is not happy, or John is not happy"? \*42. Is the assertion "This statement is false" a proposition? \*43. The nth statement in a list of 100 statements is "Exactly

*n* ofthe statements in this list are false." a) What conclusions can you draw from these

statements? b) Answer part (a) if the nth statement is "At least *n* of

the statements in this list are false." c) Answer part (b) assuming that the list contains

99 statements. 44. An ancient Sicilian legend says that the barber in a re- mote town who can be reached only by traveling a dan- gerous mountain road shaves those people, and only those

20 1 / The Foundations: Logic and Proofs

people, who do not shave themselves. Can there be such a barber? 45. Each inhabitant of a remote village always tells the truth or always lies. A villager will only give a "Yes" or a "No" response to a question a tourist asks. Suppose you are a tourist visiting this area and come to a fork in the road. One branch leads to the ruins you want to visit; the other branch leads deep into the jungle. A villager is standing at the fork in the road. What one question can you ask the villager to determine which branch to take? 46. An explorer is captured by a group of cannibals. There are two types of cannibals-those who always tell the truth and those who always lie. The cannibals will barbecue the explorer unless he can determine whether a particular cannibal always lies or always tells the truth. He is allowed to ask the cannibal exactly one question. a) Explain why the question "Are you a liar?" does not

work. b) Find a question that the explorer can use to determine whether the cannibal always lies or always tells the truth. 47. Express these system specifications using the propositions *p* "The message is scanned for viruses" and *q* "The mes- sage was sent from an unknown system" together with logical connectives. a) "The message is scanned for viruses whenever the

message was sent from an unknown system." b) "The message was sent from an unknown system but

it was not scanned for viruses." c) "It is necessary to scan the message for viruses when-

ever it was sent from an unknown system." d) "When a message is not sent from an unknown system

it is not scanned for viruses." 48. Express these system specifications using the proposi- tions *p* "The user enters a valid password," *q* "Access is granted," and *r* "The user has paid the SUbscription fee" and logical connectives. a) "The user has paid the subscription fee, but does not

enter a valid password." b) "Access is granted whenever the user has paid the sub-

scription fee and enters a valid password." c) "Access is denied ifthe user has not paid the subscrip-

tion fee." d) "If the user has not entered a valid password but has paid the subscription fee, then access is granted." 49. Are these system specifications consistent? "The system is in multiuser state if and only if it is operating normally. If the system is operating normally, the kernel is func- tioning. The kernel is not functioning or the system is in interrupt mode. If the system is not in multiuser state, then it is in interrupt mode. The system is not in interrupt mode." 50. Are these system specifications consistent? "Whenever the system software is being upgraded, users cannot ac- cess the file system. If users can access the file system,

*/-20*

then they can save new files. If users cannot save new files, then the system software is not being upgraded." 51. Are these system specifications consistent? "The router can send packets to the edge system only if it supports the new address space. For the router to support the new ad- dress space it is necessary that the latest software release be installed. The router can send packets to the edge sys- tem if the latest software release is installed, The router does not support the new address space." 52. Are these system specifications consistent? "If the file system is not locked, then new messages will be queued. If the file system is not locked, then the system is func- tioning normally, and conversely. If new messages are not queued, then they will be sent to the message buffer. Ifthe file system is not locked, then new messages will be sent to the message buffer. New messages will not be sent to the message buffer." 53. What Boolean search would you use to look for Web pages about beaches in New Jersey? What if you wanted to find Web pages about beaches on the isle of Jersey (in the English Channel)? 54. What Boolean search would you use to look for Web pages about hiking in West Virginia? What if you wanted to find Web pages about hiking in Virginia, but not in West Virginia?

Exercises 55-59 relate to inhabitants of the island of knights and knaves created by Smullyan, where knights always tell the truth and knaves always lie. You encounter two people, *A* and *B.* Determine, if possible, what *A* and *B* are if they address you in the ways described. If you cannot determine what these two people are, can you draw any conclusions?

*55. A* says "At least one of us is a knave" and *B* says nothing. *56. A* says "The two of us are both knights" and *B* says *"A* is

a knave." *57. A* says "I am a knave or *B* is a knight" and *B* says nothing. 58. Both *A* and *B* say "I am a knight." *59. A* says "We are both knaves" and *B* says nothing.

Exercises 60-65 are puzzles that can be solved by translating statements into logical expressions and reasoning from these expressions using truth tables.

60. The police have three suspects for the murder of Mr. Cooper: Mr. Smith, Mr. Jones, and Mr. Williams. Smith, Jones, and Williams each declare that they did not kill Cooper. Smith also states that Cooper was a friend of Jones and that Williams disliked him. Jones also states that he did not know Cooper and that he was out of town the day Cooper was killed. Williams also states that he saw both Smith and Jones with Cooper the day of the killing and that either Smith or Jones must have killed him. Can you determine who the murderer was if a) one of the three men is guilty, the two innocent men are telling the truth, but the statements of the guilty man mayor may not be true? b) innocent men do not lie?

1.2 *1-21*

Propositional Equivalences 21

61. Steve would like to determine the relative salaries of three

"Diana did it." Diana said "Carlos lied when he said that coworkers using two facts. First, he knows that if Fred

I did it." is not the highest paid of the three, then Janice is. Sec-

a) If the authorities also know that exactly one of the four ond, he knows that if Janice is not the lowest paid, then

suspects is telling the truth, who did it? Explain your Maggie is paid the most. Is it possible to determine the

reasoning. relative salaries of Fred, Maggie, and Janice from what

b) If the authorities also know that exactly one is lying, Steve knows? If so, who is paid the most and who the

who did it? Explain your reasoning. least? Explain your reasoning.

\*65. Solve this famous logic puzzle, attributed to Albert 62. Five friends have access to a chat room. Is it possible to determine who is chatting if the following information is known? Either Kevin or Heather, or both, are chatting. Either Randy or Vijay, but not both, are chatting. If Abby is chatting, so is Randy. Vijay and Kevin are either both chatting or neither is. If Heather is chatting, then so are Abby and Kevin. Explain your reasoning. 63. A detective has interviewed four witnesses to a crime. From the stories of the witnesses the detective has con- cluded that if the butler is telling the truth then so is the cook; the cook and the gardener cannot both be telling the truth; the gardener and the handyman are not both lying; and if the handyman is telling the truth then the cook is lying. For each ofthe four witnesses, can the detective de- termine whether that person is telling the truth or lying? Explain your reasoning.

Einstein, and known as the zebra puzzle. Five men with different nationalities and with different jobs live in con- ~ secutive houses on a street. These houses are painted different colors. The men have different pets and have dif- ferent favorite drinks. Determine who owns a zebra and whose favorite drink is mineral water (which is one of the favorite drinks) given these clues: The Englishman lives in the red house. The Spaniard owns a dog. The Japanese man is a painter. The Italian drinks tea. The Norwegian lives in the first house on the left. The green house is immediately to the right of the white one. The photogra- pher breeds snails. The diplomat lives in the yellow house. Milk is drunk in the middle house. The owner of the green house drinks coffee. The Norwegian's house is next to the blue one. The violinist drinks orange juice. The fox is in a house next to that of the physician. The horse is in a house next to that of the diplomat. *[Hint:* Make a table 64. Four friends have been identified as suspects for an unau-

where the rows represent the men and columns represent thorized access into a computer system. They have made

the color of their houses, their jobs, their pets, and their statements to the investigating authorities. Alice said

favorite drinks and use logical reasoning to determine the "Carlos did it." John said "I did not do it." Carlos said

correct entries in the table.]

**1.2 Propositional Equivalences**

**Introduction**

An important type of step used in a mathematical argument is the replacement of a statement with another statement with the same truth value. Because of this, methods that produce propo- sitions with the same truth value as a given compound proposition are used extensively in the construction of mathematical arguments. Note that we will use the term "compound propo- sition" to refer to an expression formed from propositional variables using logical operators, such as *p* /\ *q.*

DEFINITION 1

We begin our discussion with a classification of compound propositions according to their possible truth values.

A compound proposition that is always true, no matter what the truth values of the propositions that occur in it, is called a *tautology.* A compound proposition that is always false is called a *contradiction.* A compound proposition that is neither a tautology nor a contradiction is called a *contingency.*

Tautologies and contradictions are often important in mathematical reasoning. Example 1 illus- trates these types of compound propositions.

22 1 / The Foundations: Logic and Proofs /-22

TABLE 1 Examples of a Tautology and a Contradiction.

*p -,p* pv *-,p P I\-,p*

T F T F F T T F

TABLE 2 De Morgan's Laws.

*-'(p 1\ q)* == *-'p* v *-.q -'(p* v *q)* == *-'p I\-.q*

EXAMPLE 1 We can construct examples of tautologies and contradictions using just one propositional vari- able. Consider the truth tables of *p* v *"""p* and *p 1\ """p,* shown in Table 1. Because *p v"""p* is always true, it is a tautology. Because *p 1\ """p* is always false, it is a contradiction. ~

Demo~

DEFINITION 2

**Examples** Extra~ **.....**

EXAMPLE 2

**Logical Equivalences**

Compound propositions that have the same truth values in all possible cases are called logically equivalent. We can also define this notion as follows.

The compound The notation *p* propositions == *q* denotes that *p* and *p q* are and *q* called *logically equivalent* if *p* ~ *q* is a tautology.

are logically equivalent.

*Remark:* The symbol == is not a logical connective and *p* == *q* is not a compound proposition but of == rather to denote is the logical statement equivalence.

that *p* ~ *q* is a tautology. The symbol {} is sometimes used instead

One way to determine whether two compound propositions are equivalent is to use a truth table. In particular, the compound propositions *p* and *q* are equivalent if and only if the columns giving their truth values agree. Example 2 illustrates this method to establish an extremely important and useful logical equivalence, namely, that *...... (p* v *q)* of *"""p 1\ ...... q.* This logical equivalence is one of the two De Morgan laws, shown in Table 2, named after the English mathematician Augustus De Morgan, of the mid-nineteenth century.

Show that *...... (p* v *q)* and *"""p 1\ ...... q* are logically equivalent.

*Solution:* The truth tables for these compound propositions are the truth values of the compound propositions *...... (p* v *q)* and *......* displayed *p 1\ ...... q* agree in Table for all 3. Because possible combinations ofthe truth values of *p* and *q,* it follows that *...... (p* v *q)* ~ *( ...... p 1\ ...... q)* is a tautology and that these compound propositions are logically equivalent. ~

TABLE 3 Truth Tables for.(p v *q)* and.p *1\ .q.*

*p q pVq -,(p* v *q) -,p -,q -,p 1\ -,q*

T T T F F F F T F T F F T F F T T F T F F F F F T T T T

*I* .J I 1.2 Propositional Equivalences 23

TABLE 4 Truth Tables for *-,p* V *q* and *p-+q.*

*p q -,p -,pVq p-+q*

T T F T T T F F F F F T T T T F F T T T

EXAMPLE 3 Show that *p* -+ *q* and *-'p* V *q* are logically equivalent.

*Solution:* We construct the truth table for these compound propositions in Table 4. Because the truth values of *-'p* V *q* and *p* -+ *q* agree, they are logically equivalent. ~

We will now establish a logical equivalence of two compound propositions involving three different propositional variables *p, q,* and *r.* To use a truth table to establish such a logical equivalence, we need eight rows, one for each possible combination of truth values of these three variables. We symbolically represent these combinations by listing the truth values of *p, q,* and *r,* respectively. These eight combinations of truth values are TTT, TTF, TFT, TFF, FTT, FTF, FFT, and FFF; we use this order when we display the rows of the truth table. Note that we need to double the number of rows in the truth tables we use to show that compound propositions are equivalent for each additional propositional variable, so that 16 rows are needed to establish the logical equivalence of two compound propositions involving four propositional variables, and so on. In general, *2n* rows are required if a compound proposition involves *n* propositional variables.

EXAMPLE 4 Show that *p* V *(q 1\ r)* and *(p* v *q) 1\ (p* V *r)* are logically equivalent. This is the *distributive*

*law* of disjunction over conjunction.

*Solution:* We construct the truth table for these compound propositions in Table 5. Because the truth values of *p* v *(q 1\ r)* and *(p* v *q) 1\ (p* V *r)* agree, these compound propositions are logically equivalent. ~

TABLE 5 A Demonstration That *p* V *(q 1\ r)* and (p V *q) 1\* (p V *r)* Are Logically Equivalent.

*p q r q 1\ r pV(ql\r) pVq pVr (p* V *q) 1\ (p* V *r)*

T T T T T T T T T T F F T T T T T F T F T T T T T F F F T T T T F T T T T T T T F T F F F T F F F F T F F F T F F F F F F F F F

**24** I / The Foundations: Logic and Proofs 1-24

**TABLE 6 Logical Equivalences.**

*Equivalence Name*

*p/\T=p* Identity laws *pvF=p*

pvT=T Domination laws *p/\F=F*

*pvp=p* Idempotent laws *p/\p=p*

~(~p) = *p* Double negation law

*pvq=qvp* Commutative laws *P/\q=q/\P*

*(p* V *q)* V *r* = *p* V *(q* v *r)* Associative laws *(p* /\ *q)* /\ *r* = *p* /\ *(q* /\ *r)*

*p* v *(q* /\ *r)* = *(p* V *q)* /\ *(p* V *r)* Distributive laws *p* /\ *(q* V *r)* = *(p* /\ *q)* V *(p* /\ *r)*

~(p /\ *q)* = ~p v ~q De Morgan's laws ~(p v *q)* = ~p /\ ~q

*P* V *(p* /\ *q)* = *p* Absorption laws *p* /\ *(p* V *q)* = *p*

pV~p =T Negation laws *p* /\ ~p = F

Table 6 contains some important equivalences. \* In these equivalences, T denotes the com- pound proposition that is always true and F denotes the compound proposition that is al- ways false. We also display some useful equivalences for compound propositions involving conditional statements and biconditional statements in Tables 7 and 8, respectively. The reader is asked to verify the equivalences in Tables 6-8 in the exercises at the end of the section.

The associative law for disjunction shows that the expression *P* v *q* V *r* is well defined, in the sense that it does not matter whether we first take the disjunction of *P* with *q* and then the disjunction of *P* v *q* with *r,* or if we first take the disjunction of *q* and *r* and then take the disjunction of *P* with *q* v *r.* Similarly, the expression *P* /\ *q* /\ *r* is well defined. By extending this reasoning, it follows that *PI* v *P2* V ... V *Pn* and *PI* /\ *P2* /\ ... /\ *Pn* are well defined whenever *PI, P2,* ... , *Pn* are propositions. Furthermore, note that De Morgan's laws extend to

and

*-'(PI* /\ *P2* /\ ... /\ *Pn)* == *(-'PI* V *-'P2* V ... v *-'Pn).*

(Methods for proving these identities will be given in Section 4.1.)

'Readers familiar with the concept of a Boolean algebra will notice that these identities are a special case of identities that hold for any Boolean algebra. Compare them with set identities in Table I in Section 2.2 and with Boolean identities in Table 5 in Section Il.l.

1.2 1-25

Propositional Equivalences 25

TABLE 7 Logical Equivalences TABLE 8 Logical Involving Conditional Statements. Equivalences Involving *(p P p p p* ~(p v *1\* ---+ ---+ ---+ *q q* ---+ *q q q)* = = = = *q)* ~(p ~p *1\* ~p ~q = *(p* ---+ *P* ---+ ---+ v ---+ *1\ q*

*q*

~q) ~p

*r)* ~q

= *p* ---+ *(q 1\ r)*

Biconditionals.

*P* \*+ *q* = *(p* ---+ *q) 1\ (q* ---+ *p)*

*P \*+q P* \*+ *q* ~(p \*+ =~p\*+ = *q) (p* = *1\ P* ~q

*q)* V (~p *1\* ~q)

\*+ ~q

*(p* ---+ *r) 1\ (q* ---+ *r)* = *(p* V *q)* ---+ *r (p (p* ---+ ---+ *q) r)* V v *(q (p* ---+ ---+ *r) r)* = = *(p p* ---+ *(q* V *r) 1\ q)* ---+ *r*

Using De Morgan's Laws

The two logical equivalences known as De Morgan's laws are particularly important. They tell us *-'(p* how v *q)* to == negate *-'p 1\* conjunctions *-.q* tells us that and the how negation to negate of junction of the negations of the component propositions. disjunctions. In particular, a disjunction is formed by Similarly, the equivalence the equivalence taking *-'(p* the *1\ q)* con- == *-'p* v *-.q* tells us that the negation of a conjunction is formed by taking the disjunction of the negations of the component propositions. Example 5 illustrates the use of De Morgan's laws.

EXAMPLE 5 Use De Morgan's laws to express the negations of "Miguel has a cellphone and he has a laptop

computer" and "Heather will go to the concert or Steve will go to the concert."

unkS~

*Solution:* Let *p* be "Miguel has a cellphone" and *q* be "Miguel has a laptop computer." Then "Miguel has a cellphone and he has a laptop computer" can be represented by *p 1\ q.* By the first of De Morgan's laws, *-'(p 1\ q)* is equivalent to *-'p* v *-'q.* Consequently, we can express

AUGUSTUS DE MORGAN (1806-1871) Augustus De Morgan was born in India, where his father was a colonel in the Indian army. De Morgan's family moved to England when he was 7 months old. He attended private schools, where he developed a strong interest in mathematics in his early teens. De Morgan studied at Trinity College, Cambridge, graduating in 1827. Although he considered entering medicine or law, he decided on a career in mathematics. He won a position at University College, London, in 1828, but resigned when the college dismissed a fellow professor without giving reasons. However, he resumed this position in 1836 when his successor died, staying there until 1866.

De Morgan was a noted teacher who stressed principles over techniques. His students included many famous mathematicians, including Augusta Ada, Countess of Lovelace, who was Charles Babbage's collaborator in his work on computing machines (see page 27 for biographical notes on Augusta Ada). (De Morgan cautioned the countess against studying too much mathematics, because it might interfere with her childbearing abilities!)

De Morgan was an extremely prolific writer. He wrote more than 1000 articles for more than 15 periodicals. De Morgan also wrote textbooks on many subjects, including logic, probability, calculus, and algebra. In 1838 he presented what was perhaps the first clear explanation of an important proof technique known as *mathematical induction* (discussed in Section 4.1 of this text), a term he coined. In the 1840s De Morgan made fundamental contributions to the development of symbolic logic. He invented notations that helped him prove propositional equivalences, such as the laws that are named after him. In 1842 De Morgan presented what was perhaps the first precise definition of a limit and developed some tests for convergence of infinite series. De Morgan was also interested in the history of mathematics and wrote biographies of Newton and Halley.

In 1837 De Morgan married Sophia Frend, who wrote his biography in 1882. De Morgan's research, writing, and teaching left little time for his family or social life. Nevertheless, he was noted for his kindness, humor, and wide range of knowledge.

**26** 1 / The FOWldations: Logic and Proofs /-26

EXAMPLE 6

Extra~ Examples~

the negation of our original statement as "Miguel does not have a cellphone or he does not have a laptop computer."

Show that *..... (p* -+ *q)* and *p* /\ *""'q* are logically equivalent.

*Solution:* We could use a truth table to show that these compound propositions are equivalent (similar to what we did in Example 4). Indeed, it would not be hard to do so. However, we want to illustrate how to use logical identities that we already know to establish new logical identities, something that is of practical importance for establishing equivalences of compound propositions with a large number of variables. So, we will establish this equivalence by developing a series of logical equivalences, using one of the equivalences in Table 6 at a time, starting with *..... (p* -+ *q)* and ending with *p* /\ *..... q.* We have the following equivalences .

Let *r* be "Heather will go to the concert" and *s* be "Steve will go to the concert." Then "Heather will go to the concert or Steve will go to the concert" can be represented by *r* v *s.* By the second of De Morgan's laws, *..... (r* v *s)* is equivalent to *..... r* /\ *..... s.* Consequently, we can express the negation of our original statement as "Heather will not go to the concert and Steve will not go to the concert." ....

**Constructing New Logical Equivalences**

The logical equivalences in Table 6, as well as any others that have been established (such as those shown in Tables 7 and 8), can be used to construct additional logical equivalences. The reason for this is that a proposition in a compound proposition can be replaced by a compound proposition that is logically equivalent to it without changing the truth value of the original compound proposition. This technique is illustrated in Examples 6-8, where we also use the fact that if *p* and *q* are logically equivalent and *q* and *r* are logically equivalent, then *p* and *r* are logically equivalent (see Exercise 56).

*..... (p* -+ *q)* == == == *p ..... ..... ( (* /\ *..... ..... p p) ..... q* v /\ *q) ..... q* by Example 3 by the second De Morgan law by the double negation law

EXAMPLE 7 Show that *..... (p* v *( ..... p* /\ *q))* and *""'p /\ ..... q* are logically equivalent by developing a series of

logical equivalences.

*Solution:* We will use one of the equivalences in Table 6 at a time, starting with *..... (p* v *( ..... p* /\ *q))* and ending with *""'p* /\ *..... q. (Note:* we could also easily establish this equivalence using a truth table.) We have the following equivalences .

*..... (p* v *( ..... p* /\ *q))* == *""'p* /\ *..... ( ..... p* /\ *q)*

*==""'p* /\ *[ ..... ( ..... p)* v *..... q] ==""'P/\(pv ..... q)* == *( ..... p* /\ *p)* v *( ..... p* /\ *..... q) ==Fv( ..... p/\ ..... q) ==(""'p/\""'q)vF ==""'p /\ ..... q*

by the second De Morgan law

by the first De Morgan law

by the double negation law

by the second distributive law because *-'p* /\ *P* == F by the commutative law for disjunction

by the identity law for F

Consequently *..... (p* v *( ..... p* /\ *q))* and ..... p */\ ..... q* are logically equivalent.

/ :1 7 1.2 Propositional Equivalences **27**

**EXAMPLE** 8 Show that *(p* /\ *q)* -+ *(p* v *q)* is a tautology.

**Ulksitl**

*Solution:* To show that this statement is a tautology, we will use logical equivalences to demon- strate that it is logically equivalent to T. *(Note:* This could also be done using a truth table.)

*(p* /\ *q)* -+ *(p* v *q)* == *....... (p* /\ *q)* V *(p* V *q)*

*==( ....... pv ....... q)v(pvq)* == *( ....... p* v *p)* v *( ....... q* v *q)*

==TvT

==T

by Example 3 by the first De Morgan law by the associative and commutative

laws for disjunction by Example I and the commutative

law for disjunction by the domination law

A truth table can be used to determine whether a compound proposition is a tautology. This can be done by hand for a compound proposition with a small number of variables, but when the number of variables grows, this becomes impractical. For instance, there are 220 = 1,048,576 rows in the truth value table for a compound proposition with 20 variables. Clearly, you need a computer to help you determine, in this way, whether a compound proposition in 20 variables is a tautology. But when there are 1000 variables, can even a computer determine in a reasonable amount of time whether a compound proposition is a tautology? Checking every one of the 21000 (a number with more than 300 decimal digits) possible combinations of truth values simply cannot be done by a computer in even trillions of years. Furthermore, no other procedures are known that a computer can follow to determine in a reasonable amount of time whether a compound proposition in such a large number of variables is a tautology. We will study questions such as this in Chapter 3, when we study the complexity of algorithms.

AUGUSTA ADA, COUNTESS OF LOVELACE (1815-1852) Augusta Ada was the only child from the marriage of the famous poet Lord Byron and Lady Byron, Annabella Millbanke, who separated when Ada was 1 month old, because of Lord Byron's scandalous affair with his half sister. The Lord Byron had quite a reputation, being described by one of his lovers as "mad, bad, and dangerous to know." Lady Byron was noted for her intellect and had a passion for mathematics; she was called by Lord Byron "The Princess of Parallelograms." Augusta was raised by her mother, who encouraged her intellectual talents especially in music and mathematics, to counter what Lady Byron considered dangerous poetic tendencies. At this time, women were not allowed to attend universities and could not join learned societies. Nevertheless, Augusta pursued her mathematical studies mathematician, Mary independently Somerville, and and in with 1834 mathematicians, at a dinner party including hosted William Frend. She was also encouraged by another female by Mary Somerville, she learned about Charles Babbage's ideas for a calculating machine, called the Analytic Engine. In 1838 Augusta Ada married Lord King, later elevated to Earl of Lovelace. Together they had three children.

Augusta Ada continued her mathematical studies after her marriage. Charles Babbage had continued work on his Analytic Engine and lectured on this in Europe. In 1842 Babbage asked Augusta Ada to translate an article in French describing Babbage's invention. When Babbage saw her translation, he suggested she add her own notes, and the resulting work was three times the length of the original. The most complete accounts of the Analytic Machine are found in Augusta Ada's notes. In her notes, she compared the working of the Analytic Engine to that ofthe Jacquard loom, with Babbage's punch cards analogous to the cards used to create patterns on the loom. Furthermore, she recognized the promises ofthe machine as a general purpose computer much better than Babbage did. She stated that the "engine is the material expression of any indefinite fimction of any degree of generality and complexity." Her notes on the Analytic Engine anticipate many future developments, including computer-generated music. Augusta Ada published her writings under her initials A.A.L. concealing her identity as a women as did many women did at a time when women were not considered to be the intellectual equals of men. After 1845 she and Babbage worked toward the development of a system to predict horse races. Unfortunately, their system did not work well, leaving Augusta heavily in debt at the time of her death at an unfortunately young age from uterine cancer.

In 1953 Augusta Ada's notes on the Analytic Engine were republished more than 100 years after they were written, and after they had been long forgotten. In his work in the 1950s on the capacity of computers to think (and his famous Turing Test), Alan Turing responded to Augusta Ada's statement that "The Analytic Engine has no pretensions whatever to originate anything. It can do whatever we know how to order it to perform." This "dialogue" between Turing and Augusta Ada is still the subject of controversy. Because of her fimdamental contributions to computing, the programming language Augusta is named in honor ofthe Countess of Lovelace.

1-28 28 I / The Foundations: Logic and Proofs

**Exercises**

1. Use truth tables to verify these equivalences.

9. Show that each of these conditional statements is a tau- a) tology by using truth tables. c) a) *(P/\q)-,;p* e) c) *--'p* -'; *(p* -'; *q)* e) *--'(p* -'; *q)* -'; *P p P p* /\ /\ v T F *p* == == == F *p p* b) d) t) *p* V F *p* v T *P* /\ *P* == == == *P P* T

2. Show that --.( *--'p)* and *p* are logically equivalent. 3. Use truth tables to verify the commutative

laws a) *p* V *q* == *q* v *p* b) *p* /\ *q* == *q* /\ *P* 4. Use truth tables to verify the associative laws

a) *(p* V *q)* V *r* == *p* V *(q* V *r)* b) *(p* /\ *q)* /\ *r* == *p* /\ *(q* /\ *r)* 5. Use a truth table to verify the distributive law

*p* /\ *(q* V *r)* == *(p* /\ *q)* V *(p* /\ *r).* 6. Use a truth table to verify the first De Morgan law

*--'(p* /\ *q)* == *--'p* v *--'q.*

7. Use De Morgan's laws to find the negation of each of the

following statements. a) Jan is rich and happy. b) Carlos will bicycle or run tomorrow. c) Mei walks or takes the bus to class. d) Ibrahim is smart and hard working. 8. Use De Morgan's laws to find the negation of each of the

following statements. a) Kwame will take ajob in industry or go to graduate

school. b) Yoshiko knows Java and calculus. c) James is young and strong. d) Rita will move to Oregon or Washington.

b) *p* -'; *(p* v *q)* d) *(p* /\ *q)* -'; *(p* -'; *q)* t) *--'(p* -'; *q)* -'; *--.q* 10. Show that each of these conditional statements is a tau-

tology by using truth tables. a) *[--'P* /\ *(p* V *q)]* -'; *q* b) *[(p* -'; *q)* /\ *(q* -'; *r)]* -'; *(p* -'; *r)* c) *[p* /\ *(p* -'; *q)]* -'; *q* d) *[(p vq)/\(p* -'; *r)/\(q* -'; *r)]* -'; *r* 11. Show that each conditional statement in Exercise 9 is a

tautology without using truth tables. 12. Show that each conditional statement in Exercise 10 is a

tautology without using truth tables. 13. Use truth tables to verify the absorption laws.

a) *p* v *(p* /\ *q)* == *p* b) *p* /\ *(p* V *q)* == *P* 14. Determine whether ( --. *p* /\ *(p* -'; *q»* -'; *--.q* is a

tautology. 15. Determine whether *(--.q* /\ *(p* -'; *q»* -'; *--'p* is a

tautology. Each of Exercises 16-28 asks you to show that two compound propositions are logically equivalent. To do this, either show that both sides are true, or that both sides are false, for exactly the same combinations of truth values of the propositional variables in these expressions (whichever is easier). 16. Show that *p* \*+ *q* and *(p* /\ *q)* v *(--'P* /\ *--.q)* are

equivalent. 17. Show that *--'(p* \*+ *q)* and *p* \*+ *--.q* are logically

equivalent.

HENRY MAURICE SHEFFER (1883-1964) Henry Maurice Sheffer, born to Jewish parents in the western Ukraine, emigrated to the United States in 1892 with his parents and six siblings. He studied at the Boston Latin School before entering Harvard, where he completed his undergraduate degree in 1905, his master's in 1907, and his Ph.D. in philosophy in 1908. After holding a postdoctoral position at Harvard, Henry traveled to Europe on a fellowship. Upon returning to the United States, he became an academic nomad, spending one year each at the University of Washington, Cornell, the University of Minnesota, the University of Missouri, and City College in New York. In 1916 he returned to Harvard as a faculty member in the philosophy department. He remained at Harvard until his retirement in 1952.

Sheffer introduced what is now known as the Sheffer stroke in 1913; it became well known only after its use in the 1925 edition of Whitehead and Russell's *Principia Mathematica.* In this same edition Russell wrote that Sheffer had invented a powerful method that could be used to simplify the *Principia.* Because of this comment, Sheffer was something of a mystery man to logicians, especially because Sheffer, who published little in his career, never published the details of this method, only describing it in mimeographed notes and in a brief published abstract.

Sheffer was a dedicated teacher of mathematical logic. He liked his classes to be small and did not like auditors. When strangers appeared in his classroom, Sheffer would order them to leave, even his colleagues or distinguished guests visiting Harvard. Sheffer was barely five feet tall; he was noted for his wit and vigor, as well as for his nervousness and irritability. Although widely liked, he was quite lonely. He is noted for a quip he spoke at his retirement: "Old professors never die, they just become emeriti." Sheffer is also credited with coining the term "Boolean algebra" (the subject of Chapter 11 ofthis text). Sheffer was briefly married and lived most of his later life in small rooms at a hotel packed with his logic books and vast files of slips of paper he used to jot down his ideas. Unfortunately, Sheffer suffered from severe depression during the last two decades of his life.

/-29

18. Show that *p* -+ *q* and ~q -+ ~p are logically equivalent. 19. Show that ~p \_ *q* and *p* \_ ~q are logically equivalent. 20. Show that ~(p EB *q)* and *p* \_ *q* are logically equivalent. 21. Show that ~(p \_ *q)* and ~p \_ *q* are logically

equivalent. 22. Show that *(p* -+ *q)* /\ *(p* -+ *r)* and *p* -+ *(q* /\ *r)* are log-

ically equivalent. 23. Show that *(p* -+ *r)* /\ *(q* -+ *r)* and *(p* v *q)* -+ *r* are log-

ically equivalent. 24. Show that *(p* -+ *q)* v *(p* -+ *r)* and *p* -+ *(q* V *r)* are log-

ically equivalent. 25. Show that *(p* -+ *r)* V *(q* -+ *r)* and *(p* /\ *q)* -+ *r* are log-

ically equivalent. 26. Show that ~ *p* -+ *(q* -+ *r)* and *q* -+ *(p* V *r)* are logically

equivalent. 27. Show that *p* \_ *q* and *(p* -+ *q)* /\ *(q* -+ *p)* are logically

equivalent. 28. Show that *p* \_ *q* and ~p \_ ~q are logically equivalent. 29. Show that *(p* -+ *q)* /\ *(q* -+ *r)* -+ *(p* -+ *r)* is a tautology. 30. Show that *(p* v *q)* /\ (~p V *r)* -+ *(q* V *r)* is a tautology. 31. Show that *(p* -+ *q)* -+ rand *p* -+ *(q* -+ *r)* are not

equivalent. 32. Show that *(p* /\ *q)* -+ rand *(p* -+ *r)* /\ *(q* -+ *r)* are not

equivalent. 33. Showthat(p -+ *q)* -+ *(r* -+ *s) and(p* -+ *r)* -+ *(q* -+ *s)*

are not logically equivalent. The dual of a compound proposition that contains only the logical operators v, /\, and ~ is the compound proposition obtained by replacing each v by /\, each /\ by v, each T by F, and each F by T. The dual of *s* is denoted by *s\*.* 34. Find the dual of each ofthese compound propositions. a) *p* v ~q b) *P* /\ *(q* V *(r* /\ T» c) *(p* /\ ~q) V *(q* /\ F) 35. Find the dual of each of these compound propositions.

a) *p* /\ ~q /\ ~r b) *(p* /\ *q* /\ *r)* V *s* c) *(p* v F) /\ *(q* v T) 36. When does *s\** = *s,* where *s* is a compound proposition? 37. Show that *(s\*)\** = s when *s* is a compound proposition. 38. Show that the logical equivalences in Table 6, except for the double negation law, come in pairs, where each pair contains compound propositions that are duals of each other. \*\*39. Why are the duals of two equivalent compound proposi- tions also equivalent, where these compound propositions contain only the operators /\, v, and~? 40. Find a compound proposition involving the propositional variables *p, q,* and *r* that is true when *p* and *q* are true and *r* is false, but is false otherwise. *[Hint:* Use a conjunction of each propositional variable or its negation.] 41. Find a compound proposition involving the propositional variables *p, q,* andr that is true when exactly two of *p, q,* and *r* are true and is false otherwise. *[Hint:* Form a dis- junction of conjunctions. Include a conjunction for each

1.2 Propositional Equivalences 29

combination of values for which the propositional vari- able is true. Each conjunction should include each of the three propositional variables or their negations.] l? 42. Suppose that a truth table in *n* propositional variables is specified. Show that a compound proposition with this truth table can be formed by taking the disjunction of conjunctions of the variables or their negations, with one conjunction included for each combination of values for which the compound proposition is true. The resulting compound proposition is said to be in disjunctive normal form. A collection of logical operators is called functionally com- plete if every compound proposition is logically equiva- lent to a compound proposition involving only these logical operators. 43. Show that ~, /\, and v form a functionally complete col- lection oflogical operators. *[Hint:* Use the fact that every compound proposition is logically equivalent to one in disjunctive normal form, as shown in Exercise 42.] \*44. Show that ~ and /\ form a functionally complete collec- tion of logical operators. *[Hint:* First use a De Morgan law to show that *p* v *q* is equivalent to ~(~p /\ ~q ).] \*45. Show that ~ and v form a functionally complete collec-

tion of logical operators. The following exercises involve the logical operators *NAND* and *NOR.* The proposition *p NAND q* is true when either *p* or *q,* or both, are false; and it is false when both *p* and *q* are true. The proposition *p NOR q* is true when both *p* and *q* are false, and it is false otherwise. The propositions *p NAND q* and *p NOR q* are denoted by *p* I *q* and *p* + *q,* respectively. (The op- erators I and + are called the Sheffer stroke and the Peirce arrow after H. M. Sheffer and C. S. Peirce, respectively.) 46. Construct a truth table for the logical operator *NAND.* 47. Show that *p* I *q* is logically equivalent to ~(p /\ *q).* 48. Construct a truth table for the logical operator *NOR.* 49. Show that *p* + *q* is logically equivalent to ~(p v *q).* 50. In this exercise we will show that {+} is a functionally

complete collection of logical operators. a) Show that *p* + *p* is logically equivalent to ~p. b) Show that *(p* + *q)* + *(p* + *q)* is logically equivalent to

*pvq.* c) Conclude from parts (a) and (b), and Exercise 49, that {+} is a functionally complete collection of logical operators. \*51. Find a compound proposition logically equivalent to

*p* -+ *q* using only the logical operator +. 52. Show that {I} is a functionally complete collection of log-

ical operators. 53. Show that *p* I *q* and *q* I *p* are equivalent. 54. Show that *p* I *(q* I *r)* and *(p* I *q)* I *r* are not equivalent, so

that the logical operator I is not associative. \*55. How many different truth tables of compound proposi- tions are there that involve the propositional variables *p andq?*

30 1 / The FOWldations: Logic and Proofs

56. Show that if *p, q,* and *r* are compound propositions such that *p* and *q* are logically equivalent and *q* and *r* are logically equivalent, then *p* and *r* are logically equivalent. 57. The following sentence is taken from the specification of a telephone system: "If the directory database is opened, then the monitor is put in a closed state, if the system is not in its initial state." This specification is hard to under- stand because it involves two conditional statements. Find an equivalent, easier-to-understand specification that in- volves disjunctions and negations but not conditional statements. 58. How many of the disjunctions *p* v ~q, ~p v *q, q* V *r, q* V ~r, and ~q v ~r can be made simultaneously true by an assignment of truth values to *p, q,* and *r?* 59. How many of the disjunctions *p* v ~q v s, ~p V ~r V s, ~p V ~r V ~s, ~p v *q* v ~s, *q* V *r* V ~s, *q* V ~r V ~s, ~p v ~q v ~s, *p* v *r* v s, and *p* v *r* v~s can be made

**1.3** Predicate~ **and Quantifiers**

**Introduction**

*/-30*

simultaneously true by an assignment of truth values to *p,q,r,ands?* A compound proposition is satisfiable if there is an assign- ment of truth values to the variables in the compound propo- sition that makes the statement form true. 60. Which of these compound propositions are satisfiable? a) *(p* v *q* V ~r) /\ *(p* V ~q V ~s) /\ *(p* V ~r V ~s) /\

(~p V ~q V ~s) /\ *(p* V *q* V ~s) b) (~p v ~q V *r)* /\ (~p V *q* V ~s) /\ *(p* V ~q V ~s) /\

(~p V ~r V ~s) /\ *(p* V *q* V ~r) /\ *(p* V ~r V ~s) c) *(p* v *q* V *r)* /\ *(p* V ~q V ~s) /\ *(q* V ~r V s) /\

(~p V *r* v s) /\ (~p V *q* V ~s) /\ *(p* V ~q V ~r) /\ (~p V ~q V s) /\ (~p V ~r V ~s) 61. Explain how an algorithm for determining whether a compound proposition is satisfiable can be used to de- termine whether a compound proposition is a tautology. *[Hint:* Look at ~p, where *p* is the compound proposition that is being examined.]

Propositional logic, studied in Sections 1.1 and 1.2, cannot adequately express the meaning of statements in mathematics and in natural language. For example, suppose that we know that

"Every computer connected to the university network is functioning properly."

No rules of propositional logic allow us to conclude the truth of the statement

"MATH3 is functioning properly,"

where MATH3 is one of the computers connected to the university network. Likewise, we cannot use the rules of propositional logic to conclude from the statement

"CS2 is under attack by an intruder,"

where CS2 is a computer on the university network, to conclude the truth of

"There is a computer on the university network that is under attack by an intruder."

In this section we will introduce a more powerful type of logic called predicate logic. We . will see how predicate logic can be used to express the meaning of a wide range of statements in mathematics and computer science in ways that permit us to reason and explore relationships between objects. To understand predicate logic, we first need to introduce the concept of a predicate. Afterward, we will introduce the notion of quantifiers, which enable us reason with statements that assert that a certain property holds for all objects of a certain type and with statements that assert the existence of an object with a particular property.

/-3/ 1.3 Predicates and Quantifiers 31

**Predicates**

Statements involving variables, such as

*"x* > 3," *"x* = *y* + 3," *"x* + *y* = *z,"*

"computer *x* is under attack by an intruder,"

and

"computer *x* is functioning properly,"

are often found in mathematical assertions, in computer programs, and in system specifications. These statements are neither true nor false when the values of the variables are not specified. In this section, we will discuss the ways that propositions can be produced from such statements.

The statement *"x* is greater than 3" has two parts. The first part, the variable *x,* is the subject of the statement. The second part-the predicate, "is greater than 3"-refers to a property that the subject of the statement can have. We can denote the statement *"x* is greater than 3" by *P(x),* where *P* denotes the predicate "is greater than 3" and *x* is the variable. The statement *P(x)* is also said to be the value of the **propositional function** *P* at *x.* Once a value has been assigned to the variable *x,* the statement *P(x)* becomes a proposition and has a truth value. Consider Examples 1 and 2.

EXAMPLE 1 Let *P(x)* denote the statement *"x* > 3." What are the truth values of *P(4)* and *P(2)?*

*Solution:* We obtain the statement *P* (4) by setting *x* = 4 in the statement *"x* > 3." Hence, *P* (4), which is the statement "4 > 3," is true. However, *P(2),* which is the statement "2 > 3," is false. ....

EXAMPLE 2 Let *A(x)* denote the statement "Computer *x* is under attack by an intruder." Suppose that of the computers on campus, only CS2 and MATH 1 are currently under attack by intruders. What are truth values of *A(CSl), A(CS2),* and *A(MATHl)?*

EXAMPLE 3

*Solution:* We obtain the statement *A(CSl)* by setting *x* = CSI in the statement "Computer *x* is under attack by an intruder." Because CS 1 is not on the list of computers currently under attack, we conclude that *A(CSl)* is false. Similarly, because CS2 and MATH 1 are on the list of computers under attack, we know that *A(CS2)* and *A(MATHl)* are true. ....

Let *Q(x, y)* denote the statement *"x* = *y* + 3." What are the truth values of the propositions Q(l, 2) and *Q(3,* O)?

We can also statement *"x* = *y* have + 3." statements We can denote that involve this statement more than by one *Q(x,* variable. *y),* where For instance, *x* consider the and *yare* variables and *Q* is the predicate. When values are assigned to the variables *x* and *y,* the statement *Q(x, y)* has a truth value.

*Solution:* statement To "1 obtain = 2 + Q(l, 3," which 2), set is *x* false. = 1 andy The = statement 2 in the statement *Q(3,* 0) is *Q(x,* the proposition *y).* Hence, Q(l, "3 = 2) 0 + is the 3," which is true. ....

32 I / The Foundations: Logic and Proofs 1-32

EXAMPLE 4 Let *A(c, n)* denote the statement "Computer c is connected to network *n,"* where c is a variable representing a computer and *n* is a variable representing a network. Suppose that the computer MATHI is connected to network CAMPUS2, but not to network CAMPUSl. What are the values of *A(MATH1,* CAMPUS 1) and *A(MATH1,* CAMPUS2)?

*Solution:* Because MATH 1 is not connected to the CAMPUS 1 network, we see that *A* (MATH 1 , CAMPUS 1) is false. However, because MATH 1 is connected to the CAMPUS2 network, we see that *A* (MATH 1 , CAMPUS2) is true. •

Similarly, we can let *R(x, y, z)* denote the statement *"x* + *y* = *z."* When values are assigned to the variables *x, y,* and *z,* this statement has a truth value.

EXAMPLE 5 What are the truth values of the propositions *R(1,* 2, 3) and *R(O,* 0, I)?

unkS~

*Solution:* The proposition *R(l,* 2, 3) statement *R(x, y, z).* We see that *R(1,* note that *R(O,* 0, I), which is the statement is 2, obtained 3) "0 is + the ° by = statement 1," setting is false. "1 *x* = + I, 2 = *y* = 2, and *z* = 3 in the 3," which is true. Also

•

In general, a statement involving the *n* variables *Xl, X2,* ••• *,Xn* can be denoted by

A statement of the form *P(XI, X2,* •.. , *Xn)* is the value of the propositional function *P* at the *n-tuple (Xl, X2,* .•• *,xn),* and *P* is also called a n-place predicate or a n-ary predicate.

Propositional functions occur in computer programs, as Example 6 demonstrates.

CHARLES SANDERS PEIRCE (1839-1914) Many consider Charles Peirce the most original and versatile intellect from the United States; he was born in Cambridge, Massachusetts. He made important contributions to an amazing number of disciplines, including mathematics, astronomy, chemistry, geodesy, metrology, engineer- ing, psychology, philology, the history of science, and economics. He was also an inventor, a lifelong student of medicine, a book reviewer, a dramatist and an actor, a short story writer, a phenomenologist, a logician, and a metaphysician. He is noted as the preeminent system-building philosopher competent and productive in logic, mathematics, and a wide range of sciences. His father, Benjamin Peirce, was a professor of mathematics and natural philosophy at Harvard. Peirce attended Harvard (1855-1859) and received a Harvard master of arts degree (1862) and an advanced degree in chemistry from the Lawrence Scientific School (1863). His father encouraged him to pursue a career in science, but instead he chose to study logic and scientific methodology.

In 1861, Peirce became an aide in the United States Coast Survey, with the goal of better understanding scientific methodology. His service for the Survey exempted him from military service during the Civil War. While working for the Survey, Peirce carried out astronomical and geodesic work. He made fundamental contributions to the design of pendulums and to map projections, applying new mathematical developments in the theory of elliptic functions. He was the first person to use the wavelength oflight as a unit of measurement. Peirce rose to the position of Assistant for the Survey, a position he held until he was forced to resign in 1891 when he disagreed with the direction taken by the Survey's new administration.

Although making his living from work in the physical sciences, Peirce developed a hierarchy of sciences, with mathematics at the top rung, in which the methods of one science could be adapted for use by those sciences under it in the hierarchy. He was also the founder ofthe American philosophical theory of pragmatism.

The only academic position Peirce ever held was as a lecturer in logic at Johns Hopkins University in Baltimore from 1879 to 1884. His mathematical work during this time included contributions to logic, set theory, abstract algebra, and the philosophy of mathematics. His work is still relevant today; some of his work on logic has been recently applied to artificial intelligence. Peirce believed that the study of mathematics could develop the mind's powers of imagination, abstraction, and generalization. His diverse activities after retiring from the Survey included writing for newspapers and journals, contributing to scholarly dictionaries, translating scientific papers, guest lecturing, and textbook writing. Unfortunately, the income from these pursuits was insufficient to protect him and his second wife from abject poverty. He was supported in his later years by a fund created by his many admirers and administered by the philosopher William James, his lifelong friend. Although Peirce wrote and published voluminously in a vast range of subjects, he left more than 100,000 pages of unpublished manuscripts. Because of the difficulty of studying his unpublished writings, scholars have only recently started to understand some of his varied contributions. A group of people is devoted to making his work available over the Internet to bring a better appreciation of Peirce's accomplishments to the world.

1-33 1.3 Predicates and Quantifiers 33

**EXAMPLE** 6 Consider the statement

**if** *x* > 0 **then** *x* := *x* + 1.

When this statement is encountered in a program, the value of the variable *x* at that point in the execution ofthe program is inserted of *x,* the assignment statement *x* := into *x* + *P(x),* 1 is executed, which is so *"x* the > 0." If *P(x)* is true for this value value of *x* is increased by 1. If *P(x)* is false for this value of *x,* the assignment statement is not executed, so the value of *x* is not changed. ~

Predicates are also used in the verification that computer programs always produce the desired output when given valid input. The statements that describe valid input are known as **preconditions** and the conditions that the ouput should satisfy when the program has run are known as **postconditions.** As Example 7 illustrates, we use predicates to describe both preconditions and postconditions. We will study this process in greater detail in Section 4.4.

**EXAMPLE** 7 Consider the following program, designed to interchange the values of two variables *x* and *y.*

**Assessment** ~

temp := x x .- y y := temp

When the variables in a propositional function are assigned values, the resulting statement becomes a proposition with a certain truth value. However, there is another important way, called **quantification,** to create a proposition from a propositional function. Quantification expresses the extent to which a predicate is true over a range of elements. In English, the words *all, some, many, none,* and *few* are used in quantifications. We will focus on two types of quantification here: universal quantification, which tells us that a predicate is true for every element under consideration, and existential quantification, which tells us that there is one or more element under consideration for which the predicate is true. The area of logic that deals with predicates and quantifiers is called the **predicate calculus.**

Find predicates that we can use as the precondition and the postcondition to verify the correctness of this program. Then explain how to use them to verify that for all valid input the program does what is intended.

*Solution:* For the precondition, we need to express that *x* and *y* have particular values before we run the program. So, for this precondition we can use the predicate *P (x* , *y),* where *P (x* , *y)* is the statement *"x* = *a* and *y* = *b,"* where *a* and *b* are the values of *x* and *y* before we run the program. Because we want to verify that the program swaps the values of *x* and *y* for all input values, for the postcondition we can use *Q(x, y),* where *Q(x, y)* is the statement *"x* = band *y* = *a."*

To verify that the program always does what it is supposed to do, suppose that the precon- dition *P (x* , *y)* holds. That is, we suppose that the statement *"x* = *a* and *y* = *boo* is true. This means that *x* = *a* and *y* = *b.* The first step ofthe program, *temp* := *x,* assigns the value of *x* to the variable *temp,* so after this step we know that *x* = *a, temp* = *a,* and *y* = *b.* After the second step of the program, *x* := *y,* we know that *x* = *b, temp* = *a,* and *y* = *b.* Finally, after the third step, we know that *x* = *b, temp* = *a,* and *y* = *a.* Consequently, after this program is run, the postcondition *Q(x, y)* holds, that is, the statement *"x* = band *y* = *a"* is true. ~

**Quantifiers**

34 I / The Foundations: Logic and Proofs 1-34

**Assessmenl** ~

DEFINITION 1

EXAMPLE 8

THE UNIVERSAL QUANTIFIER Many mathematical statements assert that a property is true for all values of a variable in a particular domain, called the domain of discourse (or the universe of discourse), often just referred to as the domain. Such a statement is expressed using universal quantification. The universal quantification of *P (x)* for a particular domain is the proposition that asserts that *P(x)* is true for all values of *x* in this domain. Note that the domain specifies the possible values of the variable *x.* The meaning of the universal quantification of *P(x)* changes when we change the domain. The domain must always be specified when a universal quantifier is used; without it, the universal quantification of a statement in not defined.

The *universal quantification* of *P(x)* is the statement

*" P (x)* for all values of *x* in the domain."

Let *P(x)* be the statement *"x* + 1 > *x."* What is the truth value of the quantification *VxP(x),* where the domain consists of all real numbers?

The notation Vx *P(x)* denotes the universal quantification of *P(x).* Here V is called the universal quantifier. We read Vx *P(x)* as "for all *x P(x)"* or "for every *x P(x)."* An element for which *P(x)* is false is called a counterexample ofVx *P(x).*

The meaning of the universal quantifier is summarized in the first row of Table I. We illustrate the use of the universal quantifier in Examples 8-13.

*Solution:* Because *P(x)* is true for all real numbers *x,* the quantification

*VxP(x)*

is true.

*Remark:* Generally, an implicit assumption is made that all domains of discourse for quantifiers are nonempty. Note that if the domain is empty, then *VxP(x)* is true for any propositional function *P(x)* because there are no elements *x* in the domain for which *P(x)* is false.

Besides "for all" and "for every," universal quantification can be expressed in many other ways, including "all of," "for each," "given any," "for arbitrary," "for each," and "for any."

*Remark:* It is best to avoid using "for any *x"* because it is often ambiguous as to whether "any" means "every" or "some." In some cases, "any" is unambiguous, such as when it is used in negatives, for example, "there is not any reason to avoid studying."

TABLE 1 Quantifiers.

*Statement When True? When False?*

*VxP(x) P(x)* is true for every *x.* There is an *x* for which *P(x)* is false. *3xP(x)* There is an *x* for which *P(x)* is true. *P(x)* is false for every *x.*

1-35 1.3 Predicates and Quantifiers 35

A statement'v'x *P(x)* is false, where *P(x)* is a propositional function, ifandonly if *P(x)* is not always true when *x* is in the domain. One way to show that *P(x)* is not always true when *x* is in the domain is to find a counterexample to the statement *'v'x P(x).* Note that a single counterexample is all we need to establish that *'v'x P(x)* is false. Example 9 illustrates how counterexamples are used.

**EXAMPLE** 9 Let *Q(x)* be the statement *"x* < 2." What is the truth value of the quantification *'v'xQ(x),* where

the domain consists of all real numbers?

*Solution: Q(x)* is not true for every real number *x,* because, for instance, *Q(3)* is false. That is, *x* = 3 is a counterexample for the statement *'v'x Q(x).* Thus

*'v'xQ(x)*

is false.

**EXAMPLE 10** Suppose that *P(x)* is *"x2* > 0." To show that the statement *'v'xP(x)* is false where the uni- verse of discourse consists of all integers, we give a counterexample. We see that *x* = ° is a counterexample because *x2* = ° when *x* = 0, so that *x2* is not greater than ° when *x* = 0. ....

Looking for counterexamples to universally quantified statements is an important activity in the study of mathematics, as we will see in subsequent sections of this book.

When all the elements in the domain can be listed-say, Xl, *X2,* ... , *xn-it* follows that the universal quantification 'v' *x P (x)* is the same as the conjunction

because this conjunction is true if and only if *P(XI), P(X2),* ... , *P(xn )* are all true.

**EXAMPLE 11** What is the truth value of *'v'x P(x),* where *P(x)* is the statement *"x2* < 10" and the domain

consists of the positive integers not exceeding 4?

*Solution:* The statement *'v'x P(x)* is the same as the conjunction

P(l) *1\ P(2) 1\ P(3) 1\ P(4),*

because the domain consists ofthe integers 1,2,3, and 4. Because *P(4),* which is the statement "42 < 10," is false, it follows that *'v'x P(x)* is false. ....

**EXAMPLE 12** What does the statement *'v'xN(x)* mean if *N(x)* is "Computer *x* is connected to the network"

and the domain consists of all computers on campus?

*Solution:* The statement *'v'xN(x)* means that for every computer *x* on campus, that computer *x* is connected to the network. This statement can be expressed in English as "Every computer on campus is connected to the network." ....

As we have pointed out, specifying the domain is mandatory when quantifiers are used. The truth value of a quantified statement often depends on which elements are in this domain, as Example 13 shows.

36 1 / The Foundations: Logic and Proofs 1-36

**EXAMPLE 13** What is the truth value *ofYx(x2* 2: *x)* if the domain consists of all real numbers? What is the

truth value of this statement if the domain consists of all integers?

**DEFINITION 2**

**EXAMPLE 14**

*Solution:* bers, is false. The For universal example, quantification (i)2 *t* i. *Yx(x 2* 2: *x),* where Note that *x 2* 2: *x* the domain consists if and only if *x 2* - of all real num- *x* = *x(x* - 1) 2: O. Consequently, *x 2* 2: *x* if and only if *x* ::'S 0 or *x* ::: 1. It follows that *Yx(x2* 2: *x)* is false if the o domain < *x* consists of all real numbers (because the inequality is false for all real numbers *x* with < 1). However, if the domain consists of the integers, *Yx(x 2* 2: *x)* is true, because there are no integers *x* with 0 < *x* < 1. ....

The *existential quantification* of *P(x)* is the proposition

Let *P(x)* denote the statement *"x* > 3." What is the truth value of the quantification *3xP(x),* where the domain consists of all real numbers?

THE EXISTENTIAL QUANTIFIER Many mathematical statements assert that there is an element with a certain property. Such statements are expressed using existential quantification. With existential quantification, we form a proposition that is true if and only if *P (x)* is true for at least one value of *x* in the domain.

"There exists an element *x* in the domain such that *P (x* )."

We use the notation *3xP(x)* for the existential quantification of *P(x).* Here 3 is called the **existential quantifier.**

A domain must always be specified when a statement *3x P (x)* is used. Furthermore, the meaning of *3xP(x)* changes when the domain changes. Without specifying the domain, the statement *3x P(x)* has no meaning. The existential quantification *3x P(x)* is read as

"There is an *x* such that *P(x),"* "There is at least one *x* such that *P(x),"*

or

"For some *x P(x)."*

Besides the words "there exists," we can also express existential quantification in many other ways, such as by using the words "for some," "for at least one," or "there is."

The meaning of the existential quantifier is summarized in the second row of Table 1. We illustrate the use of the existential quantifier in Examples 14-16.

*Solution:* Because *"x* > 3" is sometimes true-for instance, when *x* = 4-the existential quan- tification of *P (x),* which is *3x P (x),* is true. ....

Observe that the statement *3x P (x)* is false if and only if there is no element *x* in the domain for which *P (x)* is true. That is, *3x P (x)* is false if and only *P (x)* is false for every element of the domain. We illustrate this observation in Example 15.

**EXAMPLE 15** Let *Q(x)* denote the statement *"x* = *x* + I." What is the truth value of the quantification

*3x Q(x),* where the domain consists of all real numbers?

1-37 1.3 Predicates and Quantifiers 37

*Solution:* Because *Q(x)* is false for every real number *x,* the existential quantification of *Q(x),* which is *3x Q(x),* is false. ....

*Remark:* Generally, an implicit assumption is made that all domains of discourse for quantifiers are nonempty. If the domain is empty, then *3x Q(x)* is false whenever *Q(x)* is a propositional function because when the domain is empty, there can be no element in the domain for which *Q(x)* is true.

When all elements in the domain can be listed-say, *XI, X2,* ... , *xn-* the existential quan- tification *3xP(x)* is the same as the disjunction

because this disjunction is true if and only if at least one of *P(xJ), P(X2),* ... , *P(xn)* is true.

EXAMPLE 16 What is the truth value *of3xP(x),* where *P(x)* is the statement *"x2* > 10" and the universe of

discourse consists of the positive integers not exceeding 4?

*Solution:* Because the domain is {l,2,3,4}, the proposition *3xP(x)* is the same as the disjunction

*P(l)* v *P(2)* v *P(3)* v *P(4).*

Because *P(4),* which is the statement "42 > 10," is true, it follows that *3xP(x)* is true.

It is sometimes helpful to think in terms of looping and searching when determining the truth value of a quantification. Suppose that there are *n* objects in the domain for the variable *x.* To determine whether Yx *P(x)* is true, we can loop through all *n* values of *x* to see if *P(x)* is always true. Ifwe encounter a value *x* for which *P(x)* is false, then we have shown that *YxP(x)* is false. Otherwise, Yx *P(x)* is true. To see whether *3x P(x)* is true, we loop through the *n* values of *x* searching for a value for which *P (x)* is true. If we find one, then *3x P (x)* is true. If we never find such an *x,* then we have determined that *3x P (x)* is false. (Note that this searching procedure does not apply ifthere are infinitely many values in the domain. However, it is still a useful way of thinking about the truth values of quantifications.)

**Other Quantifiers**

We have now introduced universal and existential quantifiers. These are the most important quantifiers in mathematics and computer science. However, there is no limitation on the number of different quantifiers we can define, such as "there are exactly two," "there are no more than three," "there are at least 100," and so on. Of these other quantifiers, the one that is most often seen is the uniqueness quantifier, denoted by 3! or 31• The notation *3!xP(x)* [or *31x P(x)]* states "There exists a unique *x* such that *P(x)* is true." Other phrases for uniqueness quantification include "there is exactly one" and "there is one and only one." Observe that we can use quantifiers and propositional logic to express uniqueness (see Exercise 52 in Section 1.4), so the uniqueness quantifier can be avoided. Generally, it is best to stick with existential and universal quantifiers so that rules of inference for these quantifiers can be used.

38 I / The Foundations: Logic and Proofs /-38

**Quantifiers with Restricted Domains**

An abbreviated notation is often used to restrict the domain of a quantifier. In this notation, a condition a variable must satisfY is included after the quantifier. This is illustrated in Example 17. We will also describe other forms of this notation involving set membership in Section 2.1.

EXAMPLE 17 What do the statements *"Ix* < 0 *(x2* > 0), *Vy =I-* 0 *(y3 =I-* 0), and *3z* > 0 *(z2* = 2) mean, where

the domain in each case consists of the real numbers?

*Solution:* The statement *"Ix* < 0 *(x2* > 0) states that for every real number *x* with *x* < *0,x2* > O. That is, it states "The square of a negative real number is positive." This statement is the same as *Vx(x* < 0 -+ *x2* > 0).

The statement *Vy =I-* 0 *(y3 =I-* 0) states that for every real number *y* with *y =I-* 0 and, we have *y3 =I-* O. That is, it states "The cube of every nonzero real number is nonzero." Note that this statement is equivalent to *Vy(y =I-* 0 -+ *y3 =I- 0).*

Finally, the statement *3z* > 0 *(z2* = 2) states that there exists a real number *z* with *z* > 0 such that *z2* = 2. That is, it states "There is a positive square root of 2." This statement is equivalent to *3z(z* > 0 *1\ z2* = 2). <0lIl

Note that the restriction of a universal quantification is the same as the universal quantifi- cation of a conditional statement. For instance, *"Ix* < 0 *(x2* > 0) is another way of expressing *Vx(x* < 0 -+ *x2* > 0). On the other hand, the restriction of an existential quantification is the same as the existential quantification of a conjunction. For instance, *3z* > 0 *(z2* = 2) is another way of expressing *3z(z* > 0 *1\ z2* = 2).

**Precedence of Quantifiers**

The quantifiers "land 3 have higher precedence then all logical operators from propositional calculus. For example, *"Ix P(x)* v *Q(x)* is the disjunction of *"Ix P(x)* and *Q(x).* In other words, it means *(VxP(x))* v *Q(x)* rather than *Vx(P(x)* v *Q(x)).*

**Binding Variables**

When a quantifier is used on the variable *x,* we say that this occurrence of the variable is bound. An occurrence of a variable that is not bound by a quantifier or set equal to a particular value is said to be free. All the variables that occur in a propositional function must be bound or set equal to a particular value to turn it into a proposition. This can be done using a combination of universal quantifiers, existential quantifiers, and value assignments.

The part of a logical expression to which a quantifier is applied is called the scope of this quantifier. Consequently, a variable is free if it is outside the scope of all quantifiers in the formula that specifies this variable.

EXAMPLE 18 In the statement *3x(x* + *y* = 1), the variable *x* is bound by the existential quantification *3x,* but the variable *y* is free because it is not bound by a quantifier and no value is assigned to this variable. This illustrates that in the statement *3x(x* + *y* = I), *x* is bound, but *y* is free.

In the statement *3x(P(x) 1\ Q(x))* v *VxR(x),* all variables are bound. The scope of the first quantifier, *3x,* is the expression *P(x) 1\ Q(x)* because 3x is applied only to *P(x) 1\ Q(x),* and not to the rest ofthe statement. Similarly, the scope of the second quantifier, *"Ix,* is the expression *R(x).* That is, the existential quantifier binds the variable *x* in *P(x) 1\ Q(x)* and the universal quantifier *"Ix* binds the variable *x* in *R(x).* Observe that we could have written our statement

/-39 1.3 Predicates and Quantifiers 39

using two different variables *x* and *y,* as *3x(P(x) 1\ Q(x))* v *YyR(y),* because the scopes of the two quantifiers do not overlap. The reader should be aware that in common usage, the same letter is often used to represent variables bound by different quantifiers with scopes that do not overlap. ....

**Logical Equivalences Involving Quantifiers**

In Section 1.2 we introduced the notion of logical equivalences of compound propositions. We can extend this notion to expressions involving predicates and quantifiers.

DEFINITION 3 Statements involving predicates and quantifiers are *logically equivalent* if and only if they have the same truth value no matter which predicates are substituted into these statements and which domain of discourse is used for the variables in these propositional functions. We use the notation S == *T* to indicate that two statements Sand *T* involving predicates and quantifiers are logically equivalent.

Example 19 illustrates how to show that two statements involving predicates and quantifiers are logically equivalent.

EXAMPLE 19 Show that *Yx(P(x) 1\ Q(x))* and *YxP(x) 1\ YxQ(x)* are logically equivalent (where the same domain is used throughout). This logical equivalence shows that we can distribute a universal quantifier over a conjunction. Furthermore, we can also distribute an existential quantifier over a disjunction. However, we cannot distribute a universal quantifier over a disjunction, nor can we distribute an existential quantifier over a conjunction. (See Exercises 50 and 51.)

*Solution:* To show that these statements are logically equivalent, we must show that they always take the same truth value, no matter what the predicates *P* and *Q* are, and no matter which domain of discourse is used. Suppose we have particular predicates *P* and *Q,* with a common domain. We can show that *Yx(P(x) 1\ Q(x))* and Yx *P(x) 1\ YxQ(x)* are logically equivalent by doing two things. First, we show that if *Yx(P(x) 1\ Q(x))* is true, then *YxP(x) 1\ YxQ(x)* is true. Second, we show that *ifYxP(x) 1\ YxQ(x)* is true, then *Yx(P(x) 1\ Q(x))* is true.

So, suppose that *Yx(P(x) 1\ Q(x))* is true. This means that if *a* is in the domain, then *P(a) 1\ Q(a)* is true. Hence, *P(a)* is true and *Q(a)* is true. Because *P(a)* is true and *Q(a)* is true for every element in the domain, we can conclude that *YxP(x)* and *YxQ(x)* are both true. This means that Yx *P(x) 1\* Yx *Q(x)* is true.

Next, suppose that Yx *P(x) 1\ YxQ(x)* is true. It follows that Yx *P(x)* is true and *YxQ(x)* is true. Hence, if *a* is in the domain, then *P(a)* is true and *Q(a)* is true [because *P(x)* and *Q(x)* are both true for all elements in the domain, there is no conflict using the same value of *a* here]. It follows that for all *a, P(a) 1\ Q(a)* is true. It follows that *Yx(P(x) 1\ Q(x))* is true. We can now conclude that

*Yx(P(x) 1\ Q(x))* == *YxP(x) 1\ YxQ(x).*

**Negating Quantified Expressions**

We will often want to consider the negation of a quantified expression. For instance, consider the negation of the statement

"Every student in your class has taken a course in calculus."

40 1 / The Foundations: Logic and Proofs *1-40*

**ASS8SSlIeo.ii::!**

This statement is a universal quantification, namely,

where *P(x)* is the statement *"x* has taken a course in calculus" and the domain consists of the students in your class. The negation of this statement is "It is not the case that every student in your class has taken a course in calculus." This is equivalent to "There is a student in your class who has not taken a course in calculus." And this is simply the existential quantification of the negation of the original propositional function, namely,

*VxP(x),*

*3x -.P(x).*

This example illustrates the following logical equivalence:

*-'VxP(x)* == 3x *-.P(x).*

To show that *-.Vx P(x)* and *3x (x)* are logically equivalent no matter what the propositional function *P(x)* is and what the domain is, first note that *-.Vx P(x)* is true if and only if "Ix *P(x)* is false. Next, note that "Ix *P(x)* is false if and only there is an element *x* in the domain for which *P(x)* is false. This holds if and only ifthere is an element *x* in the domain for which *-.P(x)* is true. Finally, note that there is an element *x* in the domain for which *-.P(x)* is true if and only if3x *-.P(x)* is true. Putting these steps together, we can conclude that *-.VxP(x)* is true if and only if3x *-.P(x)* is true. It follows that *-.VxP(x)* and *3x -.P(x)* are logically equivalent.

Suppose we wish to negate an existential quantification. For instance, consider the proposi- tion "There is a student in this class who has taken a course in calculus." This is the existential quantification

*3xQ(x),*

where *Q(x)* is the statement *"x* has taken a course in calculus." The negation of this statement is the proposition "It is not the case that there is a student in this class who has taken a course in calculus." This is equivalent to "Every student in this class has not taken calculus," which is just the universal quantification of the negation of the original propositional function, or, phrased in the language of quantifiers,

"Ix *-.Q(x).*

This example illustrates the equivalence

*-'3xQ(x)* == "Ix *-.Q(x).*

To show that *-.3xQ(x)* == "Ix *-.Q(x)* are logically equivalent no matter what *Q(x)* is and what the domain is, first note that *-.3xQ(x)* is true if and only *if3xQ(x)* is false. This is true if and only if no *x* exists in the domain for which *Q(x)* is true. Next, note that no *x* exists in the domain for which *Q(x)* is true if and only if *Q(x)* is false for every *x* in the domain. Finally, note that *Q(x)* is false for every *x* in the domain if and only if *-.Q(x)* is true for all *x* in the domain, which holds if and only ifVx-.Q(x) is true. Putting these steps together, we see that *-.3x Q(x)* is true if and only ifVx-.Q(x) is true. We conclude that *-.3xQ(x)* and "Ix *-.Q(x)* are logically equivalent.

The rules for negations for quantifiers are called De Morgan's laws for quantifiers. These rules are summarized in Table 2.

1.3 1-41 Predicates and Quantifiers 41

TABLE 2 De Morgan's Laws for Quantifiers.

*Negation Equivalent Statement When Is Negation True? When False?*

*-.3xP(x) Yx-.P(x)* For every *x, P(x)* is false. There is an *x* for which

*P(x)* is true. *-.YxP(x) 3x-.P(x)* There is an *x* for which *P(x)* is true for every *x.*

*P(x)* is false.

*Remark:* When the domain of a predicate *P(x)* consists of *n* elements, where *n* is a positive integer, the rules for negating quantified statements are exactly the same as De Morgan's laws discussed in Section 1.2. This is why these rules are called De Morgan's laws for quantifiers. When *-,(P(XI)* the /\ domain *P(X2)* /\ has ... /\ *n P(xn»,* elements which *Xl, X2,* ... , *x n ,* it follows is equivalent to that *-,yx P(x)* is the same as *-,P(xd* v *-,P(X2)* v ... V *-,P(xn )* by De Morgan's laws, and this same as *-,(P(xJ)* v *P(X2)* v ... V is *P(xn»,* the same as *3x-,P(x).* Similarly, *-,3xP(x)* is the which by De Morgan's laws is equivalent to *-,P(xd* /\ *-,P(X2)* /\ ... /\ *-,P(xn ),* and this is the same as *Yx-,P(x).*

We illustrate the negation of quantified statements in Examples 20 and 21.

EXAMPLE 20

What are the negations of the statements "There is an honest politician" and "All Americans eat cheeseburgers"?

*Solution:* Let *H(x)* denote *"x* is honest." Then the statement "There is an honest politician" is represented by *3xH (x),* where the domain consists of all politicians. The negation of this statement is *-'3xH(x),* which is equivalent to *Yx-,H(x).* This negation can be expressed as "Every politician is dishonest." *(Note:* In English, the statement "All politicians are not honest" is ambiguous. In common usage, this statement often means "Not all politicians are honest." Consequently, we do not use this statement to express this negation.)

Let C *(x)* denote *"x* eats cheeseburgers." Then the statement "All Americans eat cheese- burgers" is represented by *YxC (x),* where the domain consists of all Americans. The negation of this statement is *-,YxC(x),* which is equivalent to *3x-,C(x).* This negation can be expressed in several different ways, including "Some American does not eat cheeseburgers" and "There is an American who does not eat cheeseburgers." ....

EXAMPLE 21

What are the negations of the statements *Yx(x2* > *x)* and *3x(x2* = 2)?

*Solution:* The negation of *Yx(x2* > *x)* is the statement *-,Yx(x2* > *x),* which is equivalent to *3x-,(x2* > *x).* This can be rewritten as *3x(x2* ::::: *x).* The negation *of3x(x2* = 2) is the statement *-,3x(x2* = 2), which is equivalent to *Yx-,(x2* = 2). This can be rewritten as *Yx(x2 =f.* 2). The truth values of these statements depend on the domain. ....

We use De Morgan's laws for quantifiers in Example 22.

EXAMPLE 22 Show that *-,Yx(P(x)* --+ *Q(x»* and *3x(P(x)* /\ *-,Q(x»* are logically equivalent.

*Solution:* By De Morgan's law for universal quantifiers, we know that *-'Yx(P(x)* --+ *Q(x»* and *3x(-,(P(x)* --+ *Q(x»)* are logically equivalent. By the fifth logical equivalence in Table 7 in Section 1.2, we know that *-,(P(x)* --+ *Q(x»* and *P(x)* /\ *-,Q(x)* are logically equivalent for every *x.* Because we can substitute one logically equivalent expression for another in a logical equivalence, it follows that *-'Yx(P(x)* --+ *Q(x»* and *3x(P(x)* /\ *-,Q(x»* are logically equivalent. ....

42 1 / The Foundations: Logic and Proofs 1-42

**Translating from English into Logical Expressions**

Translating sentences in English (or other natural languages ) into logical expressions is a crucial task in mathematics, logic programming, artificial intelligence, software engineering, and many other disciplines. We began studying this topic in Section 1.1, where we used propositions to express sentences in logical expressions. In that discussion, we purposely avoided sentences whose translations required predicates and quantifiers. Translating from English to logical ex- pressions becomes even more complex when quantifiers are needed. Furthermore, there can be many ways to translate a particular sentence. (As a consequence, there is no "cookbook" approach that can be followed step by step.) We will use some examples to illustrate how to translate sentences from English into logical expressions. The goal in this translation is to pro- duce simple and useful logical expressions. In this section, we restrict ourselves to sentences that can be translated into logical expressions using a single quantifier; in the next section, we will look at more complicated sentences that require multiple quantifiers.

EXAMPLE 23 Express the statement "Every student in this class has studied calculus" using predicates and

quantifiers.

*Solution:* First, we rewrite the statement so that we can clearly identify the appropriate quantifiers to use. Doing so, we obtain:

"For every student in this class, that student has studied calculus."

Next, we introduce a variable *x* so that our statement becomes

"For every student *x* in this class, *x* has studied calculus."

Continuing, we introduce *C(x),* which is the statement *"x* has studied calculus." Consequently, if the domain for *x* consists ofthe students in the class, we can translate our statement as *YxC(x).* However, there are other correct approaches; different domains of discourse and other predicates can be used. The approach we select depends on the subsequent reasoning we want to carry out. For example, we may be interested in a wider group of people than only those in this class. Ifwe change the domain to consist of all people, we will need to express our statement as

"For every person *x,* if person *x* is a student in this class then *x* has studied calculus."

If *S(x)* represents the statement that person *x* is in this class, we see that our statement can be expressed as *Yx(S(x)* -+ *C(x)). [Caution!* Our statement *cannot* be expressed as *Yx(S(x) 1\ C(x))* because this statement says that all people are students in this class and have studied calculus!]

Finally, when we are interested in the background of people in subjects besides calculus, we may prefer to use the two-variable quantifier *Q(x, y)* for the statement "student *x* has studied subject *y."* Then we would replace C *(x)* by *Q(x,* calculus) in both approaches to obtain *YxQ(x,* calculus) or *Yx(S(x)* -+ *Q(x,* calculus)). ....

In Example 23 we displayed different approaches for expressing the same statement using predicates and quantifiers. However, we should always adopt the simplest approach that is adequate for use in subsequent reasoning.

EXAMPLE 24 Express the statements "Some student in this class has visited Mexico" and "Every student in

this class has visited either Canada or Mexico" using predicates and quantifiers.

1-43 EXAMPLE 25

**Exam:.c**

1.3 Predicates and Quantifiers 43

*Solution:* The statement "Some student in this class has visited Mexico" means that

"There is a student in this class with the property that the student has visited Mexico."

We can introduce a variable *x,* so that our statement becomes

"There is a student *x* in this class having the property that *x* has visited Mexico."

We introduce *M(x),* which is the statement *"x* has visited Mexico." If the domain for *x* consists of the students in this class, we can translate this first statement as *3xM(x).*

However, if we are interested in people other than those in this class, we look at the statement a little differently. Our statement can be expressed as •

"There is a person *x* having the properties that *x* is a student in this class and *x* has visited Mexico."

In this case, the domain for the variable *x* consists of all people. We introduce *S(x)* to represent *"x* is a student in this class." Our solution becomes *3x(S(x) 1\ M(x))* because the statement is that there is a person *x* who is a student in this class and who has visited Mexico. *[Caution!* Our statement cannot be expressed as *3x(S(x)* ~ *M(x)),* which is true when there is someone not in the class because, in that case, for such a person *x, Sex)* ~ *M(x)* becomes either F ~ Tor F ~ F, both of which are true.]

Similarly, the second statement can be expressed as

"For every *x* in this class, *x* has the property that *x* has visited Mexico or *x* has visited Canada."

(Note that we are assuming the inclusive, rather than the exclusive, *or* here.) We let C *(x)* be *"x* has visited Canada." Following our earlier reasoning, we see that if the domain for *x* consists of the students in this class, this second statement can be expressed as *Vx(C(x)* v *M(x)).* However, if the domain for *x* consists of all people, our statement can be expressed as

"For every person *x,* if *x* is a student in this class, then *x* has visited Mexico or *x* has visited Canada."

In this case, the statement can be expressed as *Vx(S(x)* ~ *(C(x)* v *M(x))).*

Instead of using *M (x)* and C *(x)* to represent that *x* has visited Mexico and *x* has visited Canada, respectively, we could use a two-place predicate *V (x* , *y)* to represent *"x* has visited country *y."* In this case, *V (x,* Mexico) and *V (x* , Canada) would have the same meaning as *M (x)* and *C(x)* and could replace them in our answers. If we are working with many statements that involve people visiting different countries, we might prefer to use this two-variable approach. Otherwise, for simplicity, we would stick with the one-variable predicates *M(x)* and *C(x).* ....

**Using Quantifiers in System Specifications**

In Section 1.1 we used propositions to represent system specifications. However, many system specifications involve predicates and quantifications. This is illustrated in Example 25.

Use predicates and quantifiers to express the system specifications "Every mail message larger than one megabyte will be compressed" and "If a user is active, at least one network link will be available."

*Solution:* Let *SCm, y)* be "Mail message *m* is larger than *y* megabytes," where the variable *x* has the domain of all mail messages and the variable *y* is a positive real number, and let C *(m* ) denote

**44** 1 / The Foundations: Logic and Proofs /-44

"Mail message *m* will be compressed." Then the specification "Every mail message larger than one megabyte will be compressed" can be represented as *Ym(S(m,* 1) -+ *C(m)).*

Let *A(u)* represent "User *u* is active," where the variable *u* has the domain of all users, let *S(n, x)* denote "Network link *n* is in state *x* ," where *n* has the domain of all network links and *x* has the domain of all possible states for a network link. Then the specification "If a user is active, at least one network link will be available" can be represented by *3uA(u)* -+ *3nS(n,* available). ....

**Examples from Lewis Carroll**

Lewis Carroll (really C. L. Dodgson writing under a pseudonym), the author of *Alice in Wonder- land,* is also the author of several works on symbolic logic. His books contain many examples of reasoning using quantifiers. Examples 26 and 27 come from his book *Symbolic Logic;* other examples from that book are given in the exercises at the end of this section. These examples illustrate how quantifiers are used to express various types of statements.

EXAMPL E 26 Consider these statements. The first two are called *premises* and the third is called the *conclusion.*

unkS~

The entire set is called an *argument.*

"All lions are fierce." "Some lions do not drink coffee." "Some fierce creatures do not drink coffee."

(In Section 1.5 we will discuss the issue of determining whether the conclusion is a valid conse- quence of the premises. In this example, it is.) Let *P(x), Q(x),* and *R(x)* be the statements *"x* is a lion," *"x* is fierce," and *"x* drinks coffee," respectively. Assuming that the domain consists of all creatures, express the statements in the argument using quantifiers and *P(x), Q(x),* and *R(x).*

*Solution:* We can express these statements as:

*Yx(P(x)* -+ *Q(x)). 3x(P(x)* /\ *...,R(x)).*

*3x(Q(x)* /\ *...,R(x)).*

Notice that the second statement cannot be written as *3x(P(x)* -+ *...,R(x)).* The reason is that *P(x)* -+ *...,R(x)* is true whenever *x* is not a lion, so that *3x(P(x)* -+ *...,R(x))* is true as long as there is at least one creature that is not a lion, even if every lion drinks coffee. Similarly, the third statement cannot be written as

*3x(Q(x)* -+ *...,R(x)).*

CHARLES LUTWIDGE DODGSON (1832-1898) We know Charles Dodgson as Lewis Carroll-the pseudonym he used in his writings on logic. Dodgson, the son of a clergyman, was the third of 11 chil- dren, all of whom stuttered. He was uncomfortable in the company of adults and is said to have spoken without stuttering only to young girls, many of whom he entertained, corresponded with, and photographed (sometimes in poses that today would be considered inappropriate). Although attracted to young girls, he was extremely puritanical and religious. His friendship with the three young daughters of Dean Liddell led to his writing *Alice in Wonderland,* which brought him money and fame.

Dodgson graduated from Oxford in 1854 and obtained his master of arts degree in 1857. He was appointed lecturer in mathematics at Christ Church College, Oxford, in 1855. He was ordained in the Church of England in 1861 but never practiced his ministry. His writings include articles and books on geometry, determinants, and the mathematics of tournaments and elections. (He also used the pseudonym Lewis Carroll for his many works on recreational logic.)

1-45 1.3 Predicates and Quantifiers 45

EXAMPLE 27 Consider these statements, of which the first three are premises and the fourth is a valid

conclusion.

linkS~

EXAMPLE 28

"All hummingbirds are richly colored." "No large birds live on honey."

• "Birds that do not live on honey are dull in color." "Hummingbirds are small."

Let *P(x), Q(x), R(x),* and *S(x)* be the statements *"x* is a hummingbird," *"x* is large," *"x* lives on honey," and *"x* is richly colored," respectively. Assuming that the domain consists of all birds, express the statements in the argument using quantifiers and *P(x), Q(x), R(x),* and *S(x).*

*Solution:* We can express the statements in the argument as

*Vx(P(x)* ---+ *S(x». Vx( -.3x(Q(x) -.R(x) 1\* ---+ *R(x».*

*-.S(x* ». *Vx(P(x)* ---+ *-.Q(x».*

(Note we have assumed that "small" is the same as "not large" and that "dull in color" is the same as "not richly colored." To show that the fourth statement is a valid conclusion ofthe first three, we need to use rules of inference that will be discussed in Section 1.5.) ....

**Logic Programming**

An important type of programming language is designed to reason using the rules of predicate logic. Prolog (from *Programming* in *Logic),* developed in the 1970s by computer scientists working in the area of artificial intelligence, is an example of such a language. Prolog programs include a set of declarations consisting of two types of statements, Prolog facts and Prolog rules. Prolog facts define predicates by specifYing the elements that satisfY these predicates. Prolog rules are used to define new predicates using those already defined by Prolog facts. Example 28 illustrates these notions.

Consider a Prolog program given facts telling it the instructor of each class and in which classes students are enrolled. The program uses these facts to answer queries concerning the professors who teach particular students. Such a program could use the predicates *instructor(p,* c) and *enrolled(s,* c) to represent that professor *p* is the instructor of course c and that student s is enrolled in course c, respectively. For example, the Prolog facts in such a program might include:

instructor (chan,math273) instructor(patel,ee222) instructor(grossman,cs301) enrolled (kevin,math273) enrolled(juana,ee222) enrolled(juana,cs301) enrolled(kiko,math273) enrolled(kiko,cs301)

(Lowercase letters have been used for entries because Prolog considers names beginning with an uppercase letter to be variables.)

46 I / The Foundations: Logic and Proofs 1-46

Exercises

A new predicate *teaches(p, s),* representing that professor *p* teaches student s, can be defined using the Prolog rule

teaches(P,S) :- instructor(P,C), enrolled(S,C)

which means that *teaches(p,* s) is true if there exists a class c such that professor *p* is the instructor of class c and student s is enrolled in class c. (Note that a comma is used to represent a conjunction of predicates in Prolog. Similarly, a semicolon is used to represent a disjunction of predicates.)

Prolog answers queries using the facts and rules it is given. For example, using the facts and rules listed, the query

?enrolled(kevin,math273)

produces the response

yes

because the fact *enrolled* (kevin, math273) was provided as input. The query

?enrolled(X,math273)

produces the response

kevin kiko

To produce this response, Prolog determines all possible values of X for which *enrolled(X,* math273) has been included as a Prolog fact. Similarly, to find all the professors who are instructors in classes being taken by Juana, we use the query

?teaches(X,juana)

This query returns

patel grossman

1. Let *P(x)* denote the statement *"x* :s 4." What are the truth

values?

c) Q(Massachusetts, Boston) d) Q(New York, New York) a) *P(O)* b) *P(4)* c) *P(6)* 2. Let *P(x)* be the statement "the word *x* contains the letter

*a."* What are the truth values? a) P(orange) b) P(lemon) c) *P(true)* d) P(false) 3. Let *Q(x, y)* denote the statement *"x* is the capital of *y."*

What are these truth values? a) Q(Denver, Colorado) b) Q(Detroit, Michigan)

4. State the value of *x* after the statement if *P (x)* then *x* := 1 is executed, where *P(x)* is the statement *"x>* 1," if the value of *x* when this statement is reached is a) *x* = O. c) *x* = 2.

b) *x* = 1.

5. Let *P(x)* be the statement *"x* spends more than five hours every weekday in class," where the domain for *x* consists of all students. Express each of these quantifications in English.

1-47 a) *3xP(x)*

e) *3x ..... P(x)*

b) *VxP(x)* d) *"Ix ..... P(x)* 6. Let *N(x)* be the statement *"x* has visited North Dakota," where the domain consists of the students in your school. Express each of these quantifications in English. a) *3xN(x)* b) *VxN(x)* e) *..... 3xN(x)* d) *3x ..... N(x)* e) *..... VxN(x)* t) *Vx ..... N(x)* 7. Translate these statements into English, where *C(x)* is *"x* is a comedian" and *F(x)* is *"x* is funny" and the domain consists of all people.

a) *Vx(C(x)* -+ *F(x»* e) *3x(C(x)* -+ *F(x»*

b) *Vx(C(x)* /\ *F(x»* d) *3x(C(x)* /\ *F(x»* 8. Translate these statements into English, where *R(x)* is *"x* is a rabbit" and *H (x)* is *"x* hops" and the domain consists of all animals.

a) *Vx(R(x)* -+ *H(x»* e) *3x(R(x)* -+ *H(x»*

b) *Vx(R(x)* /\ *H(x»* d) *3x(R(x)* /\ *H(x»* 9. Let *P(x)* be the statement *"x* can speak Russian" and let *Q(x)* be the statement *"x* knows the computer language C++." Express each ofthese sentences in terms of *P(x), Q(x),* quantifiers, and logical connectives. The domain for quantifiers consists of all students at your school. a) There is a student at your school who can speak Rus-

sian and who knows C++. b) There is a student at your school who can speak Rus-

sian but who doesn't know C++. e) Every student at your school either can speak Russian

or knows C++. d) No student at your school can speak Russian or knows

C++. 10. Let *C(x)* be the statement *"x* has a cat," let *D(x)* be the statement *"x* has a dog," and let *F(x)* be the statement *"x* has a ferret." Express each of these statements in terms *ofC(x), D(x), F(x),* quantifiers, and logical connectives. Let the domain consist of all students in your class. a) A student in your class has a cat, a dog, and a ferret. b) All students in your class have a cat, a dog, or a ferret. e) Some student in your class has a cat and a ferret, but

not a dog. d) No student in your class has a cat, a dog, and a ferret. e) For each of the three animals, cats, dogs, and ferrets, there is a student in your class who has one of these animals as a pet. 11. Let *P(x)* be the statement *"x* = *x2."* If the domain con-

sists of the integers, what are the truth values?

a) *P(O)* d) *P(-I)*

b) *P(l)* e) *3xP(x)*

e) *P(2)* t) *VxP(x)* 12. Let *Q(x)* be the statement *"x* + 1 > *2x."* If the domain

consists of all integers, what are these truth values? a) *Q(O)* b) Q(-I) e) *Q(l)* d) *3xQ(x)* e) *VxQ(x)* t) *3x ..... Q(x)* g) *Vx ..... Q(x)* 13. Determine the truth value of each of these statements if

the domain consists of all integers.

1.3 Predicates and Quantifiers 47

a) *Vn(n* + 1 > *n)* e) *3n(n* = *-n)*

b) *3n(2n* = *3n)* d) *Vn(n2* :::: *n)* 14. Determine the truth value of each of these statements if

the domain consists of all real numbers. a) *3x(x3* = -1) b) *3X(X4* < *x2)* e) *VX«\_X)2* = *x2)* d) *Vx(2x* > *x)* 15. Determine the truth value of each of these statements if

the domain for all variables consists of all integers. a) *Vn(n2* :::: 0) b) *3n(n2* = 2) e) *Vn(n2 :::: n)* d) *3n(n2* < 0) 16. Determine the truth value of each of these statements if

the domain of each variable consists of all real numbers. a) e) *Vx(x3x(x2 2* + = 2 2) :::: 1) b) *3x(x2* = -1)

d) *Vx(x2 =1= x)* 17. Suppose that the domain of the propositional function *P(x)* consists of the integers 0, 1, 2, 3, and 4. Write out each of these propositions using disjunctions, conjunc- tions, and negations. a) *3xP(x)* d) *Vx ..... P(x)*

b) *VxP(x)*

e) *3x ..... P(x)* e) *..... 3xP(x)*

t) *..... VxP(x)* 18. Suppose that the domain of the propositional function *P(x)* consists of the integers -2, -1,0,1, and 2. Write out each of these propositions using disjunctions, con- junctions, and negations. a) *3xP(x)* b) *VxP(x)* d) *Vx ..... P(x)* e) *..... 3xP(x)*

e) *3x ..... P(x)* t) *..... VxP(x)* 19. Suppose that the domain of the propositional function *P(x)* consists of the integers 1, 2, 3, 4, and 5. Express these statements without using quantifiers, instead using only negations, disjunctions, and conjunctions. a) *3xP(x)* b) *VxP(x)* e) *..... 3xP(x)* d) *..... VxP(x)* e) *Vx«x =1=* 3) -+ *P(x»* v *3x ..... P(x)* 20. Suppose that the domain of the propositional function *P(x)* consists of -5, -3, -1,1,3, and 5. Express these statements without using quantifiers, instead using only negations, disjunctions, and conjunctions. a) *3xP(x)* b) *VxP(x)* e) *Vx«x =1=* 1) -+ *P(x»* d) *3x«x* :::: 0) /\ *P(x»* e) *3x( ..... P(x»* /\ *Vx«x* < 0) -+ *P(x»* 21. For each of these statements find a domain for which the statement is true and a domain for which the statement is false. a) Everyone is studying discrete mathematics. b) Everyone is older than 21 years. e) Every two people have the same mother. d) No two different people have the same grandmother. 22. For each of these statements find a domain for which the statement is true and a domain for which the statement is false. a) Everyone speaks Hindi. b) There is someone older than 21 years.

**48** I / The Foundations: Logic and Proofs

c) Every two people have the same first name. d) Someone knows more than two other people. 23. Translate in two ways each of these statements into logi- cal expressions using predicates, quantifiers, and logical connectives. First, let the domain consist of the students in your class and second, let it consist of all people. a) Someone in your class can speak Hindi. b) Everyone in your class is friendly. c) There is a person in your class who was not born in

California. d) A student in your class has been in a movie. e) No student in your class has taken a course in logic

programming. 24. Translate in two ways each of these statements into logi- cal expressions using predicates, quantifiers, and logical connectives. First, let the domain consist of the students in your class and second, let it consist of all people. a) Everyone in your class has a cellular phone. b) Somebody in your class has seen a foreign movie. e) There is a person in your class who cannot swim. d) All students in your class can solve quadratic

equations. e) Some student in your class does not want to be rich. 25. Translate each of these statements into logical expressions using predicates, quantifiers, and logical connectives. a) No one is perfect. b) Not everyone is perfect. c) All your friends are perfect. d) At least one of your friends is perfect. e) Everyone is your friend and is perfect. f) Not everybody is your friend or someone is not

perfect. 26. Translate each ofthese statements into logical expressions in three different ways by varying the domain and by using predicates with one and with two variables. a) Someone in your school has visited Uzbekistan. b) Everyone in your class has studied calculus and C++. c) No one in your school owns both a bicycle and a

motorcycle. d) There is a person in your school who is not happy. e) Everyone in your school was born in the twentieth

century. 27. Translate each ofthese statements into logical expressions in three different ways by varying the domain and by using predicates with one and with two variables. a) A student in your school has lived in Vietnam. b) There is a student in your school who cannot speak

Hindi. c) A student in your school knows Java, Prolog, and

C++. d) Everyone in your class enjoys Thai food. e) Someone in your class does not play hockey. 28. Translate each ofthese statements into logical expressions using predicates, quantifiers, and logical connectives. a) Something is not in the correct place.

1-48

b) All tools are in the correct place and are in excellent

condition. e) Everything is in the correct place and in excellent

condition. d) Nothing is in the correct place and is in excellent

condition. e) One of your tools is not in the correct place, but it is

in excellent condition. 29. Express each of these statements using logical operators,

predicates, and quantifiers. a) Some propositions are tautologies. b) The negation ofa contradiction is a tautology. e) The disjunction of two contingencies can be a

tautology. d) The conjunction of two tautologies is a tautology. 30. Suppose the domain of the propositional function *P (x. y)* consists of pairs *x* and *y,* where *x* is 1, 2, or 3 and *y* is 1,2, or 3. Write out these propositions using disjunctions and conjunctions. a) *3x P(x. 3)* c) *3y-.P(2. y)*

b) *\ly P(I. y)* d) *\Ix -.P(x. 2)* 31. Suppose that the domain of *Q(x, y.* z) consists of triples *x. y.* z, where *x* = O. I. or 2, *y* = 0 or I, and z = 0 or I. Write out these propositions using disjunctions and con- junctions. c) a) *\lyQ(O. 3z-.Q(0. y.* o. *0)*

z)

b) *3xQ(x.* I. I) d) *3x-.Q(x.0, I)* 32. Express each of these statements using quantifiers. Then form the negation of the statement so that no negation is to the left of a quantifier. Next, express the negation in simple English. (Do not simply use the words "It is not the case that.") a) All dogs have fleas. b) There is a horse that can add. e) Every koala can climb. d) No monkey can speak French. e) There exists a pig that can swim and catch fish. 33. Express each of these statements using quantifiers. Then form the negation of the statement, so that no negation is to the left of a quantifier. Next, express the negation in simple English. (Do not simply use the words "It is not the case that.") a) Some old dogs can learn new tricks. b) No rabbit knows calculus. c) Every bird can fly. d) There is no dog that can talk. e) There is no one in this class who knows French and

Russian. 34. Express the negation of these propositions using quanti-

fiers, and then express the negation in English. a) Some drivers do not obey the speed limit. b) All Swedish movies are serious. e) No one can keep a secret. d) There is someone in this class who does not have a

good attitude.

/-49

35. Find a counterexample, if possible, to these universally quantified statements, where the domain for all variables consists of all integers.

a) *'fx(x2* ::: *x)* b) c) *'fx(x 'fx(x* > = 0 1)

v *x* < 0)

36. Find a counterexample, if possible, to these universally quantified statements, where the domain for all variables consists of all real numbers. a) *'fx(x 2* # *x)* c) *'fx(lxl* > 0)

b) *'fx(x2* # 2)

37. Express each of these statements using predicates and

quantifiers. a) A passenger on an airline qualifies as an elite flyer if the passenger flies more than 25,000 miles in a year or takes more than 25 flights during that year. b) A man qualifies for the marathon ifhis best previous time is less than 3 hours and a woman qualifies for the marathon if her best previous time is less than 3.5 hours. c) A student must take at least 60 course hours, or at least 45 course hours and write a master's thesis, and receive a grade no lower than a B in all required courses, to receive a master's degree. d) There is a student who has taken more than 21 credit

hours in a semester and received all A's. Exercises 38-42 deal with the translation between system specification and logical expressions involving quantifiers. 38. Translate these system specifications into English where the predicate *S(x, y)* is *"x* is in state *y"* and where the do- main for *x* and *y* consists of all systems and all possible states, respectively. a) *3xS(x,* open) b) *'fx(S(x,* malfunctioning) v *S(x,* diagnostic» c) *3xS(x,* open) v *3xS(x,* diagnostic) d) *3x..,S(x,* available) e) *'fx..,S(x,* working) 39. Translate these specifications into English where *F(p)* is "Printer *p* is out of service," *B (p)* is "Printer *p* is busy," *L(j)* is "Print job *j* is lost," and *Q(j)* is "Print job *j* is queued."

a) *3p(F(p) 1\ B(p»* -+ *3jL(j)* b) *'fpB(p)* -+ *3jQ(j)* c) *3j(QU) 1\ L(j»* -+ *3pF(p)* d) *('fpB(p) 1\ 'fjQ(j»* -+ *3jL(j)* 40. Express each of these system specifications using predi-

cates, quantifiers, and logical connectives. a) When there is less than 30 megabytes free on the hard

disk, a warning message is sent to all users. b) No directories in the file system can be opened and no files can be closed when system errors have been detected. c) The file system cannot be backed up if there is a user

currently logged on.

1.3 Predicates and Quantifiers 49

d) Video on demand can be delivered when there are at least 8 megabytes of memory available and the con- nection speed is at least 56 kilobits per second. **41.** Express each of these system specifications using predi-

cates, quantifiers, and logical connectives. a) At least one mail message, among the nonempty set of messages, can be saved ifthere is a disk with more than 10 kilobytes of free space. b) Whenever there is an active alert, all queued messages

are transmitted. c) The diagnostic monitor tracks the status of all systems

except the main console. d) Each participant on the conference call whom the host of the call did not put on a special list was billed. 42. Express each of these system specifications using predi-

cates, quantifiers, and logical connectives. a) Every user has access to an electronic mailbox. b) The system mailbox can be accessed by everyone in

the group if the file system is locked. c) The firewall is in a diagnostic state only if the proxy

server is in a diagnostic state. d) At least one router is functioning normally if the throughput is between 100 kbps and 500 kbps and the proxy server is not in diagnostic mode. 43. Determine whether *'fx(P(x)* -+ *Q(x»* and *'fxP(x)* -+ *'Ix Q(x)* are logically equivalent. JustifY your answer. 44. Determine whether *'fx(P(x)* ~ *Q(x»* and *'Ix P(x)* ~ *'Ix Q(x)* are logically equivalent. JustifY your answer. 45. Show that *3x(P(x)* v *Q(x»* and *3xP(x)* v *3xQ(x)* are

logically equivalent. Exercises 46-49 establish rules for **null** quantification that we can use when a quantified variable does not appear in part of a statement. 46. Establish these logical equivalences, where *x* does not occur as a free variable in *A.* Assume that the domain is nonempty. a) *('fxP(x»* v *A* == *'fx(P(x)* v *A)* b) *(3xP(x»* v *A* == *3x(P(x)* v *A)* 47. Establish these logical equivalences, where *x* does not occur as a free variable in *A.* Assume that the domain is nonempty. a) *('fxP(x» 1\ A* == *'fx(P(x) 1\ A)* b) *(3xP(x» 1\ A* == *3x(P(x) 1\ A)* 48. Establish these logical equivalences, where *x* does not occur as a free variable in *A.* Assume that the domain is nonempty. a) *'fx(A* -+ *P(x»* == *A* -+ *'Ix P(x)* b) *3x(A* -+ *P(x»* == *A* -+ *3xP(x)* 49. Establish these logical equivalences, where *x* does not occur as a free variable in *A.* Assume that the domain is nonempty. a) *'fx(P(x)* -+ *A)* == *3xP(x)* -+ *A* b) *3x(P(x)* -+ *A)* == *'fxP(x)* -+ *A*

50 1 *I* The Foundations: Logic and Proofs

50. Show that *VxP(x)* V *VxQ(x)* and *Vx(P(x)* V *Q(x))* are

not logically equivalent. 51. Show that *3xP(x) 1\ 3xQ(x)* and *3x(P(x) 1\ Q(x))* are

not logically equivalent. 52. As mentioned in the text, the notation *3!x P(x)* denotes

"There exists a unique *x* such that *P(x)* is true." If the domain consists of all integers, what are the truth values of these statements? a) c) *3!x(x 3!x(x* + > 3 1) = *2x)* b) d) *3!x(X2 3!x(x* = = *x* 1)

+ 1) 53. What are the truth values of these statements?

a) *3!xP(x)* --+ *3xP(x)* b) *VxP(x)* --+ *3!xP(x)* c) 3!x~P(x) --+ ~VxP(x) 54. Write out *3!x P(x),* where the domain consists ofthe inte- gers 1,2, and 3, in terms of negations, conjunctions, and disjunctions. 55. Given the Prolog facts in Example 28, what would Prolog

return given these queries? a) ?instructor(chan,math273) b) ?instructor(patel,cs301) c) ?enrolled(X,cs301) d) ?enrolled(kiko,Y) e) ?teaches(grossman,Y) 56. Given the Prolog facts in Example 28, what would Prolog

return when given these queries? a) ?enrolled(kevin,ee222) b) ?enrolled(kiko,math273) c) ?instructor(grossman,X) d) ?instructor (X, cs301) e) ?teaches(X,kevin) 57. Suppose that Prolog facts are used to define the predicates *mother(M,* Y) *andfather(F, X),* which represent that *M* is the mother otT and *F* is the father of X, respectively. Give a Prolog rule to define the predicate *sibling(X, Y),* which represents that X and Yare siblings (that is, have the same mother and the same father). 58. Suppose that Prolog facts are used to define the pred- icates *mother(M, Y)* and *father(F, X),* which repre- sent that *M* is the mother of *Y* and *F* is the father of X, respectively. Give a Prolog rule to define the predicate *grandfather(X,* y), which represents that X is the grand- father of *Y. [Hint:* You can write a disjunction in Prolog

**1.4 Nested Quantifiers**

**Introduction**

*1-50*

either by using a semicolon to separate predicates or by putting these predicates on separate lines.] Exercises 59-62 are based on questions found in the book *Symbolic Logic* by Lewis Carroll. 59. Let *P(x), Q(x),* and *R(x)* be the statements *"x* is a profes- sor," *"x* is ignorant," and *"x* is vain," respectively. Express each of these statements using quantifiers; logical con- nectives; and *P(x), Q(x),* and *R(x),* where the domain consists of all people. a) No professors are ignorant. b) All ignorant people are vain. c) No professors are vain. d) Does (c) follow from (a) and (b)? 60. Let *P(x), Q(x),* and *R(x)* be the statements *"x* is a clear explanation," *"x* is satisfactory," and *"x* is an excuse," respectively. Suppose that the domain for *x* consists of all English text. Express each of these statements using quan- tifiers, logical connectives, and *P(x), Q(x),* and *R(x).* a) All clear explanations are satisfactory. b) Some excuses are unsatisfactory. c) Some excuses are not clear explanations. \*d) Does (c) follow from (a) and (b)? 61. Let *P(x), Q(x), R(x),* and *S(x)* be the statements *"x* is a baby," *"x* is logical," *"x* is able to manage a crocodile," and *"x* is despised," respectively. Suppose that the domain consists of all people. Express each of these statements using quantifiers; logical connectives; and *P(x), Q(x), R(x),* and *Sex).* a) Babies are illogical. b) Nobody is despised who can manage a crocodile. c) Illogical persons are despised. d) Babies cannot manage crocodiles. \*e) Does (d) follow from (a), (b), and (c)? If not, is there

a correct conclusion? 62. Let *P(x), Q(x), R(x),* and *Sex)* be the statements *"x* is a duck," *"x* is one of my poultry," *"x* is an officer," and *"x* is willing to waltz," respectively. Express each of these statements using quantifiers; logical connectives; and *P(x), Q(x), R(x),* and *Sex).* a) No ducks are willing to waltz. b) No officers ever decline to waltz. c) All my poultry are ducks. d) My poultry are not officers. \*e) Does (d) follow from (a), (b), and (c)? If not, is there

a correct conclusion?

In Section 1.3 we defined the existential and universal quantifiers and showed how they can be used to represent mathematical statements. We also explained how they can be used to translate

/-5/ 1.4 Nested Quantifiers 51

English sentences into logical expressions. In this section we will study nested quantifiers. Two quantifiers are nested if one is within the scope of the other, such as

*Vx3y(x* + *y* = 0).

Note that everything within the scope of a quantifier can be thought of as a propositional function. For example,

*Vx3y(x* + *y* = 0) is the same thing as *VxQ(x),* where *Q(x)* is *3yP(x,y),* where *P(x,y)* is *x* + *y* = O. Nested quantifiers commonly occur in mathematics and computer science. Although nested quantifiers can sometimes be difficult to understand, the rules we have already studied in Section 1.3 can help us use them.

To understand these statements involving many quantifiers, we need to unravel what the quantifiers and predicates that appear mean. This is illustrated in Examples I and 2.

EXAMPLE 1 Assume that the domain for the variables *x* and *y* consists of all real numbers. The statement

*VxVy(x* + *y* = *y +x)*

says that *x* + *y* = *y* + *x* for all real numbers *x* and *y.* This is the commutative law for addition of real numbers. Likewise, the statement

*Vx3y(x* + *y* = 0)

says that for every real number *x* there is a real number *y* such that *x* + *y* = O. This states that every real number has an additive inverse. Similarly, the statement

*VxVyVz(x* + *(y* + z) = *(x* + *y)* + z)

is the associative law for addition of real numbers.

EXAMPLE 2 Translate into English the statement

*VxVy«x* > 0) /\ *(y* < 0) -+ *(xy* < 0»,

where the domain for both variables consists of all real numbers.

*Solution:* This statement says that for every real number *x* and for every real number *y,* if *x* > 0 andy < 0, then *xy* < O. That is, this statement says that for real numbers *x* andy, if *x* is positive and *y* is negative, then *xy* is negative. This can be stated more succinctly as "The product of a positive real number and a negative real number is always a negative real number." <III

THINKING OF QUANTIFICATION AS LOOPS In working with quantifications of more than one variable, it is sometimes helpful to think in terms of nested loops. (Of course, if there are infinitely many elements in the domain of some variable, we cannot actually I00p through all values. Nevertheless, this way of thinking is helpful in understanding nested quamifiers.) For example, to see whether *VxVyP(x, y)* is true, we loop through the values for *x,* and for each *x* we loop through the values for *y.* If we find that *P(x, y)* is true for all values for *x* and *y,* we have determined that V *x Vy P (x* , *y)* is true. If we ever hit a value *x* for which we hit a value *y* for which *P(x, y)* is false, we have shown that *VxVy P(x, y)* is false.

Similarly, to determine whether *Vx3yP(x, y)* is true, we loop through the values for *x.* For each *x* we loop through the values for *y* until we find a *y* for which *P (x, y)* is true. If for

**52** I / The Foundations: Logic and Proofs I-52

every *x* we hit such a *y,* then *'v'x3yP(x, y)* is true; if for some *x* we never hit such a *y,* then *'v'x3yP(x, y)* is false.

To see whether *3x'v'yP(x, y)* is true, we loop through the values for *x* until we find an *x* for which *P (x, y)* is always true when we loop through all values for *y.* Once we find such an *x,* we know that *3x'v'y P(x, y)* is true. Ifwe never hit such anx, then we know that *3x'v'y P(x, y)* is false. Finally, to see whether *3x3yP(x, y)* is true, we loop through the values for *x,* where for each *x* we loop through the values for *y* until we hit an *x* for which we hit a *y* for which *P (x, y)* is true. The statement *3x3yP(x, y)* is false only if we never hit an *x* for which we hit a *y* such that *P (x* , *y)* is true.

**The Order of Quantifiers**

Many mathematical statements involve multiple quantifications of propositional functions in- volving more than one variable. It is important to note that the order of the quantifiers is important, unless all the quantifiers are universal quantifiers or all are existential quantifiers.

These remarks are illustrated by Examples 3-5.

EXAMPLE 3 Let *P(x, y)* be the statement *"x* + *y* = *y* + *x."* What are the truth values of the quantifications *'v'x'v'yP(x, y)* and *'v'y'v'x P(x, y)* where the domain for all variables consists of all real numbers?

*Solution:* The quantification

*'v'x'v'yP(x, y)* Extra~ **Examples** ~ denotes the proposition

"For all real numbers *x,* for all real numbers *y, x* + *y* = *y* + *x."*

Because *P(x, y)* is true for all real numbers *x* and *y* (it is the commutative law for addition, which is an axiom for the real numbers-see Appendix 1), the proposition *'v'x'v'yP(x, y)* is true. Note that the statement *'v'y'v'x P(x, y)* says "For all real numbers *y,* for all real numbers *x, x* + *y* = *y* + *x."* This has the same meaning as the statement as "For all real numbers *x,* for all *realnumbersy,x* + *y* = *y* + *x."* That is, *'v'x'v'yP(x, y)and'v'y'v'xP(x,* y)havethesamemeaning, and both are true. This illustrates the principle that the order of nested universal quantifiers in a statement without other quantifiers can be changed without changing the meaning of the quantified statement. ...

EXAMPLE 4 Let *Q(x, y)* denote *"x* + *y* = 0." What are the truth values of the quantifications *3y'v'x Q(x, y)*

and *'v'x3yQ(x, y),* where the domain for all variables consists of all real numbers?

*Solution:* The quantification

*3y'v'x Q(x, y)*

denotes the proposition

"There is a real number *y* such that for every real number *x, Q (x* , *y)."*

No matter what value of *y* is chosen, there is only one value of *x* for which *x* + *y* = O. Because there is no real number *y* such that *x* + *y* = 0 for all real numbers *x,* the statement *3y'v'x Q(x, y)* is false.

/-53 1.4 Nested Quantifiers S3

TABLE 1 Quantifications of Two Variables.

*Statement When True? When False?*

*VxVyP(x,y) P(x, y)* is true for every pair *x, y.* There is a pair *x, y* for *VyVxP(x,y)* which *P(x, y)* is false.

*Vx3yP(x,y)* For every *x* there is a *y* for There is an *x* such that

which *P(x, y)* is true. *P(x, y)* is false for every *y.*

*3xVyP(x,y)* There is an *x* for which *P(x, y)* For every *x* there is a *y* for

is true for every *y.* which *P(x, y)* is false.

*3x3yP(x,y)* There is a pair *x* , *y* for which *P(x, y)* is false for every *3y3xP(x,y) P(x, y)* is true. pair *x, y.*

The quantification

*Vx3yQ(x, y)*

denotes the proposition

"For every real number *x* there is a real number *y* such that *Q(x, y)."*

Given a real number *x,* there is a real number *y* such that *x* + *y* = 0; namely, *y* = *-x.* Hence, the statement *Vx3yQ(x, y)* is true. ...

Example 4 illustrates that the order in which quantifiers appear makes a difference. The state- ments *3yVxP(x, y)* and *Vx3yP(x* , y)arenotlogicallyequivalent. The *statement3yVxP(x, y)is* true ifand only ifthere is *ay* that makes *P(x, y)* true for every *x.* So, for this statement to be true, there must be a particular value of *y* for which *P (x, y)* is true regardless of the choice of *x.* On the other hand, Vx*3yP(x ,y)* is true ifand only if for every value *ofx* there is a value of *y* for which *P (x* , *y)* is true. So, for this statement to be true, no matter which *x* you choose, there must be a value of *y* (possibly depending on the *x* you choose) for which *P (x, y)* is true. In other words, in the second case, *y* can depend on *x,* whereas in the first case, *y* is a constant independent of *x .* From these observations, it follows that *if3yVxP(x, y)* is true, then *Vx3yP(x, y)* must also be true. However, *ifVx3yP(x, y)* is true, it is not necessary for *3yVxP(x, y)* to be true. (See Supplementary Exercises 24 and 25 at the end of this chapter.)

Table I summarizes the meanings of the different possible quantifications involving two variables.

Quantifications of more than two variables are also common, as Example 5 illustrates.

EXAMPLE 5 Let *Q(x, y,* z) be the statement *"x* + *y* = *z."* What are the truth values of the statements *VxVy3zQ(x, y,* z) and *3zVxVyQ(x, y, z),* where the domain of all variables consists of all real numbers?

*Solution:* Suppose that *x* and yare assigned values. Then, there exists a real number z such that *x* + *y* = z. Consequently, the quantification

*VxVy3zQ(x, y, z),*

**54** 1 / The Foundations: Logic and Proofs 1-54

which is the statement

"For *x* + *y* all = real *z,"*

numbers *x* and for all real numbers *y* there is a real number z such that

is true. The order of the quantification here is important, because the quantification

*3zVxVyQ(x, y, z),*

which is the statement

"There true that is *x* a + real *y* number = *z,"*

z such that for all real numbers *x* and for all real numbers *y* it is

is false, because there is no value of z that satisfies the equation *x* + *y* = z for all values of *x* andy. ....

**Translating Mathematical Statements into Statements Involving Nested Quantifiers**

Mathematical statements expressed in English can be translated into logical expressions, as Examples 6-8 show.

**EXAMPLE** 6 Translate the statement "The sum of two positive integers is always positive" into a logical

expression.

Exam=~

*Solution:* To translate this statement into a logical expression, we first rewrite it so that the implied quantifiers and a domain are shown: "For every two integers, if these integers are both positive, then all positive the sum integers of these *x* integers and *y, x* is + positive." *y* is positive." Next, we Consequently, introduce the we variables can express *x* and this *y* to statement obtain "For as

*VxVy((x* > 0) *1\ (y* > 0) -+ *(x* + *y* > 0»,

where the domain for both variables consists of all integers. Note that we could also translate this using the positive integers as the domain. Then the statement "The sum of two positive integers is always positive" becomes "For every two positive integers, the sum of these integers is positive. We can express this as

*VxVy(x* + *Y* > 0),

where the domain for both variables consists of all positive integers.

**EXAMPLE** 7 Translate the statement "Every real number except zero has a mUltiplicative inverse." (A **mul-**

**tiplicative inverse** of a real number *x* is a real number *y* such that *xy* = 1.)

*Solution:* We first rewrite this as "For every real number *x* except inverse." We can rewrite this as "For every real number *x,* if *x =I* zero, 0, then *x* has a multiplicative there exists a real number *y* such that *x y* = I." This can be rewritten as

*Vx((x =I* 0) -+ *3y(xy* = 1».

One example that you may be familiar with is the concept of limit, which is important in calculus.