
LOGIC AND DISCRETE MATHEMATICS

*A Computer Science
Perspective*

WINFRIED KARL GRASSMANN
JEAN-PAUL TREMBLAY

LOGIC AND DISCRETE MATHEMATICS

A Computer Science Perspective

WINFRIED KARL GRASSMANN

Department of Computer Science
University of Saskatchewan

JEAN-PAUL TREMBLE

Department of Computer Science
University of Saskatchewan

江苏工业学院图书馆
藏书章



PRENTICE HALL, Upper Saddle River, New Jersey 07458

Library of Congress Cataloging-in-Publication Data

Grassmann, Winfried K.

Logic and discrete mathematics : a computer science perspective /

Winfried Karl Grassmann, Jean-Paul Tremblay.

p. cm.

Includes bibliographical references and index. †

ISBN 0-13-501206-6

I. Computer science — Mathematics. I. Tremblay, Jean-Paul

II. Title.

QA76.9.M35G725 1996

005.1'01'5113—dc20

95-38351

CIP

AC

Acquisitions editor: *Alan Apt*

Editorial/production supervision: *Maes Associates*

Copy editor: *William O. Thomas*

Cover designer: *Bruce Kenselaar*

Manufacturing buyer: *Donna Sullivan*



© 1996 by Prentice-Hall, Inc.

Simon & Schuster/A Viacom Company

Upper Saddle River, New Jersey 07458

All rights reserved. No part of this book may be reproduced, in any form or by any means, without permission in writing from the publisher.

Printed in the United States of America

10 9 8 7 6 5 4 3 2 1

ISBN 0-13-501206-6

PRENTICE-HALL INTERNATIONAL (UK) LIMITED, *London*

PRENTICE-HALL OF AUSTRALIA PTY. LIMITED, *Sidney*

PRENTICE-HALL CANADA INC., *Toronto*

PRENTICE-HALL HISPANOAMERICANA, S.A., *Mexico*

PRENTICE-HALL OF INDIA PRIVATE LIMITED, *New Delhi*

PRENTICE-HALL OF JAPAN, INC., *Tokyo*

SIMON & SCHUSTER ASIA PTE. LTD., *Singapore*

EDITORIA PRENTICE-HALL DO BRASIL, LTDA., *Rio de Janeiro*

To my wife Louise Grassmann and
to my children Stephanie and Bettina Grassmann
—W.K.G.

To my wife Deanna Tremblay and
to my grandchildren
Robert, Leanne, Lisa, and Nicole Tremblay
—J.P.T.

Preface

Most universities require that a course in discrete mathematics be taken by every undergraduate computer science student, and rightly so. Indeed, discrete mathematics gives the appropriate theoretical foundations for computer science, foundations that are not only beneficial for doing theoretical computer science, but also for the practice of computer science as evidenced by the recent proliferation of books on formal methods. The areas covered in a course in discrete mathematics vary, but they traditionally include logic, sets, relations, functions, and graphs. All these topics are included in this book. Moreover, this book reflects several recent trends in computer science. In particular, it gives a more thorough exposure to logical reasoning than most other texts. It also shows how to use discrete mathematics and logic for specifying new computer applications and how to reason about programs in a systematic way. The book contains chapters on languages and grammars, the Z specification language, and relational databases. There is a chapter describing Prolog, a programming language based on logic, and a section discussing Miranda, a language based on functions. In all chapters, numerous examples relate the mathematical concepts to problems in computer science. We found that such examples are essential for keeping the student motivated.

The outline of the book is as follows. Chapter 1 covers propositional calculus. We also define what is to be understood by a formal derivation. In fact, formal derivations are introduced as refinements of the type of logical reasoning we use in daily life. The chapter also contains an extensive discussion on algebraic manipulation of logical expressions.

Chapter 2 discusses predicate calculus. We cover predicates and quantifiers, and we try again to relate this theory to everyday thinking. Chapter 2 also introduces unification, a topic that is needed in order to understand logical languages such as Prolog

(covered in Chapter 4) and resolution theorem proving (described in Chapter 11). Chapter 2 also introduces equational logic. Equational logic is concerned with the manipulation of mathematical equations, and this is obviously of fundamental importance. As it turns out, equational logic is closely related to functions, and indeed, without functions, equational logic would not have attained the dominance in mathematical reasoning which it now has. This motivated us to introduce functions at this point.

Chapter 3 deals with induction and recursion, which is an important part of logic, mathematics, and, above all, computer science. We discuss several proof methods, including mathematical induction, strong induction, proof by recursion, and structural induction. The notions of matrices and sums are also introduced. As an application, recursive programming techniques are explored. This chapter also includes material on recursive functions and decidability. Though recursive functions are very important, they constitute advanced material which may be omitted.

Chapter 4 covers Prolog. The coverage is sufficient to allow the reader to write nontrivial Prolog programs. The connections between logic and Prolog are also explored. A section in Chapter 4 explains what kind of logical statements can be translated into Prolog. Another section shows how to use Prolog to program manipulations involving logical expressions. The study of Prolog also reinforces the concept of recursion, since almost every nontrivial Prolog program makes use of recursion.

Computer scientists deal with many different types of objects, which can in turn be combined in different ways. To understand these compound objects and their manipulation, the student has to understand the concepts of sets and relations, which are discussed in Chapter 5. The important operations on these constructs are described by numerous examples, many of which are directly related to computer applications.

Chapter 6 deals with several topics regarding functions, such as algorithmic analysis and computational complexity. These concepts are illustrated with a number of important examples. In addition to the topics mentioned, the language Miranda is introduced as an example of a functional language.

Chapter 7 gives an overview of graphs and trees, concepts that are fundamental to computer science. In addition to introducing basic terminology dealing with paths, reachability, and connectedness, the emphasis in this chapter is on computing paths, minimum paths, minimum weighted paths, spanning trees, and minimum spanning trees for weighted graphs. This chapter concludes with a discussion of the application of graphs to project planning and management.

Chapter 8 is about Z, a language applying concepts from logic and set theory for specifying and analyzing requirements in software development.

Chapter 9 shows how to use logic, and to a certain extent sets and functions, to reason about programs. As in Chapters 1 and 2, we start with commonsense reasoning, and we introduce formal correctness proofs as a refinement of normal reasoning.

Chapter 10 deals with context-free grammars as a vehicle for defining the syntax of programming languages and their use in syntax analysis. Emphasis is given to an LL(1) parser generator system.

Chapter 11 gives additional information about derivations. In particular, it shows how to use natural deduction and the techniques of resolution theorem proving.

The book concludes with a discussion of relational databases in Chapter 12. This chapter first focuses on the relational data model and its associated relational algebra. An alternative way of describing queries based on predicate calculus is also given. Finally, an overview of a structured query language for specifying queries is given.

This text would appeal to a student in computer science at the freshman (second term), sophomore, or junior level. Preliminary versions of this book were used in a one-semester course given to second-year computer science students. In this course, most of the material contained in Chapters 1 through 6 was covered. To this, we selectively added topics from Chapters 7 through 12. However, the book contains enough material for a full year course, but in discrete mathematics, such courses are not normally offered. Whatever the length of the course, it should lay the mathematical foundations for many classes, including classes in database design, syntactical analysis and parsing, artificial intelligence, programming languages, logic programming, functional languages, and computer hardware.

The application chapters allow the instructors to shift the emphasis according to their interests. In all cases, the material of Chapters 1, 2, and 5 is essential, and most instructors want to cover Section 6.1, which discusses functions. With respect to the other chapters, we would like to suggest the following four scenarios, which can be combined or altered, depending on the preferences of the instructor.

Emphasis on Procedural Programming For procedural programming, recursion is very important, and this is covered in Chapter 3. The same chapter contains a discussion of recursive functions, and it shows what can and what cannot be solved by computers. Section 6.3 gives an introduction to computational complexity which should prove useful to the student. The instructor may also want to cover Chapter 7, which discusses graphs and trees, and Chapter 9, which shows how to prove that a program is correct.

Emphasis on Logic and Logic Programming Since recursion is very important for all types of logic programming, the instructor should cover recursion, which is described in Chapter 3. Chapter 4 then gives an extensive treatment of Prolog. At this point, Chapter 10, grammars and languages, and Chapter 11, which shows resolution theorem proving, provide suitable topics.

Emphasis on Functions and Functional Programming To provide a functional programming emphasis, all of Chapter 3 should be covered. From Chapter 6, Sections 6.1 and 6.3 are important. Of course, Section 6.5, Miranda, is central under this scenario. For the remaining topics, the instructor may select material from Chapter 8, requirement specification, Chapter 9, correctness proofs, and Chapter 10, grammars and languages.

Emphasis on Information Systems The slant to systems analysis requires some knowledge of graphs and trees as it is discussed in Chapter 7, which in turn have the first two sections of Chapter 3 as a prerequisite. Other relevant material can be found in Chapters 8 (specification in Z) and 12 (relational database systems).

The text is geared to the student, and every effort has been made to explain the material as clearly as possible. In general, we motivate all concepts and derivations by means of examples, preferably examples from computer science. This approach allows for an informal introduction to the topic that can be formalized later. This gradual approach helps the students to appreciate the value of formal arguments and prepares them for their later studies in computer science.

As teaching and learning aids, the book includes more than 300 examples and more than 550 problems with detailed solutions provided for all even-numbered problems.

We would like to thank Wayne Mackrell, Christy Kenny, David Haugen, Allan Rempel, Jacob Wickland, and Michael Zaleski for their help in entering the manuscript and suggesting improvements to the text. Wayne and Christy have also assisted us in formulating solutions to the problems and in producing the computer-generated diagrams. A special thank you to Cyril Coupal who suggested the title of the book. Deanna Tremblay proofread many versions of the manuscript during its period of preparation.

We would like to thank our colleagues for their valuable help. Jim Greer read and commented on the first three chapters. Eric Neufeld and Grant Cheston made important suggestions for which we are very grateful. John Cooke read and commented on the relational database chapter.

Finally, we thank the students who used earlier versions of these notes and who made many valuable comments.

Winfried Karl Grassmann

Jean-Paul Tremblay

Contents

Preface xv

1 Propositional Calculus 1

- 1.1 Logical Arguments and Propositions 1
 - 1.1.1 Introduction 1
 - 1.1.2 Some Important Logical Arguments 2
 - 1.1.3 Propositions 4
- 1.2 Logical Connectives 6
 - 1.2.1 Introduction 6
 - 1.2.2 Negation 6
 - 1.2.3 Conjunction 7
 - 1.2.4 Disjunction 8
 - 1.2.5 Conditional 9
 - 1.2.6 Biconditional 11
 - 1.2.7 Further Remarks on Connectives 12
- 1.3 Compound Propositions 13
 - 1.3.1 Introduction 13
 - 1.3.2 Logical Expressions 13
 - 1.3.3 Analysis of Compound Propositions 15
 - 1.3.4 Precedence Rules 18
 - 1.3.5 Evaluation of Expressions and Truth Tables 19
 - 1.3.6 Examples of Compound Propositions 21
- 1.4 Tautologies and Contradictions 23
 - 1.4.1 Introduction 23
 - 1.4.2 Tautologies 24

1.4.3	Tautologies and Sound Reasoning	26
1.4.4	Contradictions	26
1.4.5	Important Types of Tautologies	27
1.5	Logical Equivalences and Their Use	28
1.5.1	Introduction	28
1.5.2	Proving Logical Equivalences by Truth Tables	29
1.5.3	Statement Algebra	30
1.5.4	Removing Conditionals and Biconditionals	32
1.5.5	Essential Laws for Statement Algebra	33
1.5.6	Shortcuts for Manipulating Expressions	34
1.5.7	Normal Forms	36
1.5.8	Truth Tables and Disjunctive Normal Forms	38
1.5.9	Conjunctive Normal Forms and Complementation	40
1.6	Logical Implications and Derivations	42
1.6.1	Introduction	42
1.6.2	Logical Implications	43
1.6.3	Soundness Proofs through Truth Tables	44
1.6.4	Proofs	46
1.6.5	Systems for Derivations	49
1.6.6	The Deduction Theorem	52

2 Predicate Calculus 59

2.1	Syntactic Components of Predicate Calculus	60
2.1.1	Introduction	60
2.1.2	The Universe of Discourse	60
2.1.3	Predicates	61
2.1.4	Variables and Instantiations	63
2.1.5	Quantifiers	65
2.1.6	Restrictions of Quantifiers to Certain Groups	68
2.2	Interpretations and Validity	70
2.2.1	Introduction	70
2.2.2	Interpretations	71
2.2.3	Validity	74
2.2.4	Invalid Expressions	76
2.2.5	Proving Validity	78
2.3	Derivations	79
2.3.1	Introduction	79
2.3.2	Universal Instantiation	80
2.3.3	Universal Generalization	81
2.3.4	Deduction Theorem and Universal Generalization	84
2.3.5	Dropping the Universal Quantifiers	85
2.3.6	Existential Generalization	87
2.3.7	Existential Instantiation	88

- 2.4 Logical Equivalences 92
 - 2.4.1 Introduction 92
 - 2.4.2 Basic Logical Equivalences 92
 - 2.4.3 Other Important Equivalences 94
- 2.5 Equational Logic 96
 - 2.5.1 Introduction 96
 - 2.5.2 Equality 96
 - 2.5.3 Equality and Uniqueness 99
 - 2.5.4 Functions and Equational Logic 100
 - 2.5.5 Function Compositions 103
 - 2.5.6 Properties of Operators 105
 - 2.5.7 Identity and Zero Elements 108
 - 2.5.8 Derivations in Equational Logic 111
 - 2.5.9 Equational Logic in Practice 113
 - 2.5.10 Boolean Algebra 115

3 Induction and Recursion 121

- 3.1 Induction on Natural Numbers 122
 - 3.1.1 Introduction 122
 - 3.1.2 Natural Numbers 123
 - 3.1.3 Mathematical Induction 124
 - 3.1.4 Induction for Proving Properties of Addition 128
 - 3.1.5 Changing the Induction Base 130
 - 3.1.6 Strong Induction 131
- 3.2 Sums and Related Constructs 132
 - 3.2.1 Introduction 132
 - 3.2.2 Recursive Definitions of Sums and Products 133
 - 3.2.3 Identities Involving Sums 135
 - 3.2.4 Double Sums and Matrices 139
- 3.3 Proof by Recursion 141
 - 3.3.1 Introduction 141
 - 3.3.2 Recursive Definitions 143
 - 3.3.3 Descending Sequences 146
 - 3.3.4 The Principle of Proofs by Recursion 147
 - 3.3.5 Structural Induction 149
- 3.4 Applications of Recursion to Programming 154
 - 3.4.1 Introduction 154
 - 3.4.2 Programming as Function Composition 154
 - 3.4.3 Recursion in Programs 158
 - 3.4.4 Programs Involving Trees 163
- 3.5 Recursive Functions 166
 - 3.5.1 Introduction 166
 - 3.5.2 Primitive Recursive Functions 168

- 3.5.3 Programming and Primitive Recursion 172
- 3.5.4 Minimalization 173

4 Prolog 178

- 4.1 Basic Prolog 178
 - 4.1.1 Introduction 178
 - 4.1.2 Facts, Rules, and Queries 179
 - 4.1.3 Derivations Involving Facts 181
 - 4.1.4 Derivations Involving Rules 183
 - 4.1.5 Instantiations and Unification 186
 - 4.1.6 Backtracking 188
 - 4.1.7 Resolution 190
- 4.2 Running and Testing Programs 193
 - 4.2.1 Introduction 193
 - 4.2.2 Prolog Compilers and Interpreters 194
 - 4.2.3 Consulting a Database 194
 - 4.2.4 Debugging and Tracing 196
- 4.3 Additional Features of Prolog 197
 - 4.3.1 Introduction 197
 - 4.3.2 Input and Output 197
 - 4.3.3 Structures 198
 - 4.3.4 Infix Notation 199
 - 4.3.5 Arithmetic 200
 - 4.3.6 Equality Predicates 201
- 4.4 Recursion 203
 - 4.4.1 Introduction 203
 - 4.4.2 Recursive Predicates 204
 - 4.4.3 Termination 205
 - 4.4.4 Loops and Prolog 207
 - 4.4.5 Lists 208
 - 4.4.6 Recursive Predicates Involving Lists 210
 - 4.4.7 Successive Refinement 213
- 4.5 Negation in Prolog 215
 - 4.5.1 Introduction 215
 - 4.5.2 Prolog as a Logic Language 215
 - 4.5.3 Negation as Failure 218
 - 4.5.4 Use of the Clause Order 219
 - 4.5.5 Cuts 220
- 4.6 Application of Prolog to Logic 222
 - 4.6.1 Introduction 222
 - 4.6.2 Lists as Logical Expressions 222
 - 4.6.3 Representing Logical Expressions as Structures 224

5 Sets and Relations 230

- 5.1 Sets and Set Operations 230
 - 5.1.1 Introduction 230
 - 5.1.2 Sets and Their Members 231
 - 5.1.3 Subsets 233
 - 5.1.4 Intersections 235
 - 5.1.5 Unions 236
 - 5.1.6 Differences and Complements 237
 - 5.1.7 Expressions Involving Sets 239
- 5.2 Tuples, Sequences, and Powersets 243
 - 5.2.1 Introduction 243
 - 5.2.2 Tuples and Cartesian Products 244
 - 5.2.3 Sequences and Strings 246
 - 5.2.4 Powersets 247
 - 5.2.5 Types and Signatures 248
- 5.3 Relations 251
 - 5.3.1 Introduction 251
 - 5.3.2 Relations and Their Representation 252
 - 5.3.3 Domains and Ranges 254
 - 5.3.4 Some Operations on Relations 255
 - 5.3.5 Composition of Relations 257
 - 5.3.6 Examples 261
- 5.4 Properties of Relations 263
 - 5.4.1 Introduction 263
 - 5.4.2 Relations on a Set 263
 - 5.4.3 Reflective Relations 264
 - 5.4.4 Symmetric Relations 266
 - 5.4.5 Transitivity 267
 - 5.4.6 Closures 269
 - 5.4.7 Equivalence Relations 270
 - 5.4.8 Partial Orders 272

6 More About Functions 281

- 6.1 Representations and Manipulations Involving Functions 281
 - 6.1.1 Introduction 281
 - 6.1.2 Definitions and Notation 282
 - 6.1.3 Representations of Functions 285
 - 6.1.4 The Lambda Notation 286
 - 6.1.5 Restrictions and Overloading 287
 - 6.1.6 Composition of Functions 289
 - 6.1.7 Injections, Surjections, and Inverses 292
 - 6.1.8 Creating Inverses by Creating Types 296

- 6.2 Enumerations, Isomorphisms, and Homomorphisms 299
 - 6.2.1 Introduction 299[†]
 - 6.2.2 Enumerations 300
 - 6.2.3 Countable and Uncountable Sets 302
 - 6.2.4 Permutations and Combinations 305
 - 6.2.5 Isomorphisms and Homomorphisms 307
- 6.3 Computational Complexity 311
 - 6.3.1 Introduction 311
 - 6.3.2 Polynomials and Polynomial-time Algorithms 312
 - 6.3.3 Functions and Algorithms Related to Exponentials 316
 - 6.3.4 The Limits of Computability 320
 - 6.3.5 Asymptotic Analysis 321
 - 6.3.6 Divide and Conquer 326
 - 6.3.7 Nondeterministic Polynomial 329
- 6.4 Recurrence Relations 332
 - 6.4.1 Introduction 332
 - 6.4.2 Homogeneous Recurrence Relations 333
 - 6.4.3 Nonhomogeneous Recurrence Relations 336
- 6.5 Miranda 341
 - 6.5.1 Introduction 341
 - 6.5.2 Command Level 341
 - 6.5.3 Function Definitions 342
 - 6.5.4 Types, Functions, and Declarations 344
 - 6.5.5 Pattern Matching and Rewriting 346
 - 6.5.6 A Programming Problem 348

7 Graphs and Trees 353

- 7.1 Introduction and Examples of Graph Modeling 354
- 7.2 Basic Definitions of Graph Theory 362
- 7.3 Paths, Reachability, and Connectedness 369
- 7.4 Computing Paths from a Matrix Representation of Graphs 377
- 7.5 Traversing Graphs Represented as Adjacency Lists 392
 - 7.5.1 Introduction 392
 - 7.5.2 Adjacency Lists Representation of Graphs 392
 - 7.5.3 Breadth-first Search 395
 - 7.5.4 Depth-first Search 398
 - 7.5.5 Dijkstra's Algorithm for Finding Minimum Paths 402
- 7.6 Trees and Spanning Trees 409
 - 7.6.1 Introduction 409
 - 7.6.2 Free Trees 409
 - 7.6.3 Spanning Trees 410
 - 7.6.4 Minimum Spanning Trees 416

- 7.7 Scheduling Networks 422
 - 7.7.1 Introduction 422
 - 7.7.2 A Project Management Model 422
 - 7.7.3 Topological Sorting 431

8 Formal Requirement Specification in Z 441

- 8.1 Introduction 441
- 8.2 Software Life Cycle 442
- 8.3 Need for Formal Specifications 446
- 8.4 Introduction to Z 447
 - 8.4.1 Introduction 447
 - 8.4.2 Alphabet and Lexical Elements 448
 - 8.4.3 Types and Declarations 449
 - 8.4.4 Specifying a System with Logic and Sets 450
 - 8.4.5 Schemas 454
 - 8.4.6 Relations 460
 - 8.4.7 Functions 466
 - 8.4.8 Sequences 472

9 Program Correctness Proofs 481

- 9.1 Preliminary Concepts 482
 - 9.1.1 Introduction 482
 - 9.1.2 Programs and Codes 482
 - 9.1.3 Assertions 483
 - 9.1.4 Correctness 485
- 9.2 General Rules Involving Preconditions and Postconditions 486
 - 9.2.1 Introduction 486
 - 9.2.2 Precondition Strengthening 487
 - 9.2.3 Postcondition Weakening 488
 - 9.2.4 Conjunction and Disjunction Rules 490
- 9.3 Correctness Proofs in Loopless Code 493
 - 9.3.1 Introduction 493
 - 9.3.2 Assignment Statements 493
 - 9.3.3 Concatenation of Code 496
 - 9.3.4 The If-Statement 500
- 9.4 Loops and Arrays 503
 - 9.4.1 Introduction 503
 - 9.4.2 A Preliminary While Rule 503
 - 9.4.3 The General While Rule 508
 - 9.4.4 Arrays 510
 - 9.4.5 Program Termination 515

10 Grammars, Languages, and Parsing 519

- 10.1 Languages and Grammars 520
 - 10.1.1 Introduction 520
 - 10.1.2 Discussion of Grammars 521
 - 10.1.3 Formal Definition of a Language 525
 - 10.1.4 Notions of Syntax Analysis 529
 - 10.1.5 Ambiguous Grammars 535
 - 10.1.6 Reduced Grammars 540
- 10.2 Top-down Parsing 545
 - 10.2.1 Introduction 545
 - 10.2.2 General Top-down Parsing Strategy 546
 - 10.2.3 Deterministic Top-down Parsing with LL(1) Grammars 547

11 Derivations 564

- 11.1 Derivations in Propositional Calculus 564
 - 11.1.1 Introduction 564
 - 11.1.2 Basics of Natural Derivation 565
 - 11.1.3 Implementation of the Deduction Theorem 565
 - 11.1.4 Resolution 568
- 11.2 Some Results from Predicate Calculus 574
 - 11.2.1 Introduction 574
 - 11.2.2 Complements 575
 - 11.2.3 Prenex Normal Forms 576
- 11.3 Derivations in Predicate Calculus 577
 - 11.3.1 Introduction 577
 - 11.3.2 Canonical Derivations 578
 - 11.3.3 Quantifiers in Natural Deduction 582
 - 11.3.4 Replacing Quantifiers by Functions and Free Variables 583
 - 11.3.5 Resolution in Predicate Calculus 585

12 An Overview of Relational Database Systems 592

- 12.1 Basic Concepts 593
 - 12.1.1 Introduction 593
 - 12.1.2 Definitions and Concepts 593
 - 12.1.3 Introductory Example of a Relational Database 593
 - 12.1.4 Overview of a Database System 597
- 12.2 Relational Data Model 600
 - 12.2.1 Introduction 600
 - 12.2.2 Overview of the Relational Structure 600
 - 12.2.3 Relations and Their Schemas 602

12.2.4	Representing Relations in the Relational Model	603
12.2.5	Integrity Rules	604
12.3	Relational Algebra	605
12.3.1	Introduction	605
12.3.2	Basic Operations	605
12.3.3	Additional Relational Operations	607
12.3.4	Examples	614
12.4	Relational Calculus	620
12.4.1	Introduction	620
12.4.2	Tuple Calculus	620
12.4.3	Examples	622
12.5	Structured Query Language	624
12.5.1	Introduction	624
12.5.2	Data Definition	625
12.5.3	Data Management	626
12.5.4	Data Queries	627
12.6	Concluding Remarks	637

Bibliography 641

Solutions to Even-numbered Problems 644

Index 736