CONTENTS

Preface xiii
Acknowledgments xvii
Credits xix

PART I: INTRODUCTION 1

1 Why Study the Theory of Computation? 2
  1.1 The Shelf Life of Programming Tools 2
  1.2 Applications of the Theory Are Everywhere 5

2 Languages and Strings 8
  2.1 Strings 8
  2.2 Languages 10
     Exercises 19

3 The Big Picture: A Language Hierarchy 21
  3.1 Defining the Task: Language Recognition 21
  3.2 The Power of Encoding 22
  3.3 A Machine-Based Hierarchy of Language Classes 28
  3.4 A Tractability Hierarchy of Language Classes 34
     Exercises 34

4 Computation 36
  4.1 Decision Procedures 36
  4.2 Determinism and Nondeterminism 41
  4.3 Functions on Languages and Programs 48
     Exercises 52

PART II: FINITE STATE MACHINES AND REGULAR LANGUAGES 53

5 Finite State Machines 54
  5.1 Deterministic Finite State Machines 56
  5.2 The Regular Languages 60
  5.3 Designing Deterministic Finite State Machines 63
5.4 Nondeterministic FSMs 66
5.5 From FSMs to Operational Systems 79
5.6 Simulators for FSMs  80
5.7 Minimizing FSMs  82
5.8 A Canonical Form for Regular Languages 94
5.9 Finite State Transducers  96
5.10 Bidirectional Transducers  98
5.11 Stochastic Finite Automata: Markov Models and HMMs 101
5.12 Finite Automata, Infinite Strings: Büchi Automata  115
Exercises 121

6 Regular Expressions 127
6.1 What is a Regular Expression? 128
6.2 Kleene’s Theorem 133
6.3 Applications of Regular Expressions 147
6.4 Manipulating and Simplifying Regular Expressions 149
Exercises 151

7 Regular Grammars  155
7.1 Definition of a Regular Grammar 155
7.2 Regular Grammars and Regular Languages 157
Exercises 161

8 Regular and Nonregular Languages 162
8.1 How Many Regular Languages Are There? 162
8.2 Showing That a Language Is Regular 163
8.3 Some Important Closure Properties of Regular Languages 165
8.4 Showing That a Language is Not Regular 169
8.5 Exploiting Problem-Specific Knowledge 178
8.6 Functions on Regular Languages 179
Exercises 182

9 Algorithms and Decision Procedures for Regular Languages 187
9.1 Fundamental Decision Procedures 187
9.2 Summary of Algorithms and Decision Procedures for Regular Languages 194
Exercises 196

10 Summary and References 198
References 199
PART III: CONTEXT-FREE LANGUAGES AND PUSHDOWN AUTOMATA 201

11 Context-Free Grammars 203
   11.1 Introduction to Rewrite Systems and Grammars 203
   11.2 Context-Free Grammars and Languages 207
   11.3 Designing Context-Free Grammars 212
   11.4 Simplifying Context-Free Grammars • 212
   11.5 Proving That a Grammar is Correct • 215
   11.6 Derivations and Parse Trees 218
   11.7 Ambiguity 220
   11.8 Normal Forms • 232
   11.9 Island Grammars • 241
   11.10 Stochastic Context-Free Grammars • 243
       Exercises 245

12 Pushdown Automata 249
   12.1 Definition of a (Nondeterministic) PDA 249
   12.2 Deterministic and Nondeterministic PDAs 254
   12.3 Equivalence of Context-Free Grammars and PDAs 260
   12.4 Nondeterminism and Halting 274
   12.5 Alternative Equivalent Definitions of a PDA • 275
   12.6 Alternatives that are Not Equivalent to the PDA • 277
       Exercises 277

13 Context-Free and Noncontext-Free Languages 279
   13.1 Where Do the Context-Free Languages Fit in the Big Picture? 279
   13.2 Showing That a Language is Context-Free 280
   13.3 The Pumping Theorem for Context-Free Languages 281
   13.4 Some Important Closure Properties of Context-Free Languages 288
   13.5 Deterministic Context-Free Languages • 295
   13.6 Ogden’s Lemma • 303
   13.7 Parikh’s Theorem • 306
   13.8 Functions on Context-Free Languages • 308
       Exercises 310

14 Algorithms and Decision Procedures for Context-Free Languages 314
   14.1 The Decidable Questions 314
   14.2 The Undecidable Questions 320
14.3 Summary of Algorithms and Decision Procedures for Context-Free Languages 320
   Exercises 322

15 Context-Free Parsing  ●  323
   15.1 Lexical Analysis 325
   15.2 Top-Down Parsing 327
   15.3 Bottom-Up Parsing 340
   15.4 Parsing Natural Languages 350
      Exercises 358

16 Summary and References 360
   References 360

PART IV: TURING MACHINES AND UNDECIDABILITY 363

17 Turing Machines 364
   17.1 Definition, Notation and Examples 364
   17.2 Computing With Turing Machines 375
   17.3 Adding Multiple Tapes and Nondeterminism 382
   17.4 Simulating a “Real” Computer  ●  393
   17.5 Alternative Turing Machine Definitions  ●  396
   17.6 Encoding Turing Machines as Strings 400
   17.7 The Universal Turing Machine 404
      Exercises 407

18 The Church-Turing Thesis 411
   18.1 The Thesis 411
   18.2 Examples of Equivalent Formalisms  ●  414
      Exercises 424

19 The Unsolvability of the Halting Problem 426
   19.1 The Language H is Semidecidable but Not Decidable 428
   19.2 Some Implications of the Undecidability of H 431
   19.3 Back to Turing, Church, and the Entscheidungsproblem 432
      Exercises 433

20 Decidable and Semidecidable Languages 435
   20.1 D: The Big Picture 435
   20.2 SD: The Big Picture 435
25 Computable Functions  555
  25.1 What is a Computable Function?  555
  25.2 Recursive Function Theory  565
  25.3 The Recursion Theorem and Its Use  573
    Exercises  580

26 Summary and References  581
    References  582

PART V: COMPLEXITY  585

27 Introduction to the Analysis of Complexity  586
  27.1 The Traveling Salesman Problem  586
  27.2 The Complexity Zoo  589
  27.3 Characterizing Problems  590
  27.4 Measuring Time and Space Complexity  593
  27.5 Growth Rates of Functions  597
  27.6 Asymptotic Dominance  598
  27.7 Algorithmic Gaps  604
  27.8 Examples  605
    Exercises  617

28 Time Complexity Classes  621
  28.1 The Language Class P  621
  28.2 The Language Class NP  633
  28.3 Does P = NP?  642
  28.4 Using Reduction in Complexity Proofs  644
  28.5 NP-Completeness and the Cook-Levin Theorem  647
  28.6 Other NP-Complete Problems  656
  28.7 The Relationship between P and NP-Complete  672
  28.8 The Language Class Co-NP  679
  28.9 The Time Hierarchy Theorems, EXPTIME, and Beyond  681
  28.10 The Problem Classes FP and FNP  689
    Exercises  690

29 Space Complexity Classes  695
  29.1 Analyzing Space Complexity  695
  29.2 PSPACE, NPSpace, and Savitch’s Theorem  699
  29.3 PSPACE-Completeness  704
  29.4 Sublinear Space Complexity  713
29.5 The Closure of Space Complexity Classes Under Complement 718
29.6 Space Hierarchy Theorems 719
    Exercises 720

30 Practical Solutions for Hard Problems 721
    30.1 Approaches 721
    30.2 Randomized Algorithms and the Language Classes BPP, RP, Co-RP and ZPP 723
    30.3 Heuristic Search 731
        Exercises 740

31 Summary and References 742
    References 742

APPENDICES 745

A Review of Mathematical Background: Logic, Sets, Relations,
Functions, and Proof Techniques 745
    A.1 Logic 745
    A.2 Sets 753
    A.3 Relations 756
    A.4 Functions 769
    A.5 Closures 776
    A.6 Proof Techniques 779
    A.7 Reasoning about Programs 792
    A.8 A General Definition of Closure 800
        Exercises 804

B The Theory: Working with Logical Formulas 808
    B.1 Working with Boolean Formulas: Normal Forms, Resolution and OBDDs 808
    B.2 Working with First-Order Formulas: Clause Form and Resolution 820
        Exercises 833

C The Theory: Finite State Machines and Regular Languages 835

D The Theory: Context-Free Languages and PDAs 839
    D.1 Proof of the Greibach Normal Form Theorem 839
    D.2 Proof that the Deterministic Context-Free Languages are Closed Under
        Complement 846
    D.3 Proof of Parikh's Theorem 852
E  The Theory: Turing Machines and Undecidability  856
   E.1  Proof that Nondeterminism Does Not Add Power to Turing Machines  856
   E.2  An Analysis of Iterative Deepening  861
   E.3  The Power of Reduction  862
   E.4  The Undecidability of the Post Correspondence Problem  864

F  The Theory: Complexity  869
   F.1  Asymptotic Dominance  869
   F.2  The Linear Speedup Theorem  875

APPENDICES G – Q: APPLICATIONS:  879

G  Applications: Programming Languages and Compilers  880
   G.1  Defining the Syntax of Programming Languages  880
   G.2  Are Programming Languages Context-Free?  883
   G.3  Designing Programming Languages and Their Grammars  885
   G.4  Compilers for Programming Languages  887
   G.5  Functional Programming and the Lambda Calculus  890

H  Applications: Tools for Programming, Databases and Software Engineering  899
   H.1  Proving Correctness Properties of Programs and Hardware  899
   H.2  Statecharts: A Technique for Specifying Complex Systems  910
   H.3  Model-Based Test Case Generation  913
   H.4  Reverse Engineering  914
   H.5  Normal Forms for Data and for Querying Relational Databases  916

I  Applications: Networks  918
   I.1  NetworkProtocols  918
   I.2  Modeling Networks as Graphs  927
   I.3  Exploiting Knowledge: The Semantic Web  929

J  Applications: Security  948
   J.1  Physical Security Systems as FSMs  948
   J.2  Computer System Safety  949
   J.3  Cryptography  955
   J.4  Hackers and Viruses  959

K  Applications: Computational Biology  962
   K.1  A (Very) Short Introduction to Molecular Biology and Genetics  962