

Pearson International Edition

Introduction to
Automata Theory,
Languages,
and Computation

THIRD EDITION

John E. Hopcroft / Rajeev Motwani
Jeffrey D. Ullman

Table of Contents

1 Automata: The Methods and the Madness	1
1.1 Why Study Automata Theory?	2
1.1.1 Introduction to Finite Automata	2
1.1.2 Structural Representations	4
1.1.3 Automata and Complexity	5
1.2 Introduction to Formal Proof	5
1.2.1 Deductive Proofs	6
1.2.2 Reduction to Definitions	8
1.2.3 Other Theorem Forms	10
1.2.4 Theorems That Appear Not to Be If-Then Statements . .	13
1.3 Additional Forms of Proof	13
1.3.1 Proving Equivalences About Sets	14
1.3.2 The Contrapositive	14
1.3.3 Proof by Contradiction	16
1.3.4 Counterexamples	17
1.4 Inductive Proofs	19
1.4.1 Inductions on Integers	19
1.4.2 More General Forms of Integer Inductions	22
1.4.3 Structural Inductions	23
1.4.4 Mutual Inductions	26
1.5 The Central Concepts of Automata Theory	28
1.5.1 Alphabets	28
1.5.2 Strings	29
1.5.3 Languages	30
1.5.4 Problems	31
1.6 Summary of Chapter 1	33
1.7 Gradiance Problems for Chapter 1	35
1.8 References for Chapter 1	36
2 Finite Automata	37
2.1 An Informal Picture of Finite Automata	38
2.1.1 The Ground Rules	38
2.1.2 The Protocol	39

2.1.3	Enabling the Automata to Ignore Actions	41
2.1.4	The Entire System as an Automaton	43
2.1.5	Using the Product Automaton to Validate the Protocol	44
2.2	Deterministic Finite Automata	45
2.2.1	Definition of a Deterministic Finite Automaton	45
2.2.2	How a DFA Processes Strings	46
2.2.3	Simpler Notations for DFA's	47
2.2.4	Extending the Transition Function to Strings	49
2.2.5	The Language of a DFA	52
2.2.6	Exercises for Section 2.2	52
2.3	Nondeterministic Finite Automata	55
2.3.1	An Informal View of Nondeterministic Finite Automata	55
2.3.2	Definition of Nondeterministic Finite Automata	57
2.3.3	The Extended Transition Function	58
2.3.4	The Language of an NFA	59
2.3.5	Equivalence of Deterministic and Nondeterministic Finite Automata	60
2.3.6	A Bad Case for the Subset Construction	64
2.3.7	Exercises for Section 2.3	65
2.4	An Application: Text Search	68
2.4.1	Finding Strings in Text	68
2.4.2	Nondeterministic Finite Automata for Text Search	69
2.4.3	A DFA to Recognize a Set of Keywords	70
2.4.4	Exercises for Section 2.4	71
2.5	Finite Automata With Epsilon-Transitions	72
2.5.1	Uses of ϵ -Transitions	72
2.5.2	The Formal Notation for an ϵ -NFA	73
2.5.3	Epsilon-Closures	74
2.5.4	Extended Transitions and Languages for ϵ -NFA's	75
2.5.5	Eliminating ϵ -Transitions	77
2.5.6	Exercises for Section 2.5	79
2.6	Summary of Chapter 2	80
2.7	Gradience Problems for Chapter 2	80
2.8	References for Chapter 2	83
3	Regular Expressions and Languages	85
3.1	Regular Expressions	85
3.1.1	The Operators of Regular Expressions	86
3.1.2	Building Regular Expressions	87
3.1.3	Precedence of Regular-Expression Operators	90
3.1.4	Exercises for Section 3.1	91
3.2	Finite Automata and Regular Expressions	92
3.2.1	From DFA's to Regular Expressions	93
3.2.2	Converting DFA's to Regular Expressions by Eliminating States	98

3.2.3	Converting Regular Expressions to Automata	102
3.2.4	Exercises for Section 3.2	107
3.3	Applications of Regular Expressions	109
3.3.1	Regular Expressions in UNIX	109
3.3.2	Lexical Analysis	110
3.3.3	Finding Patterns in Text	112
3.3.4	Exercises for Section 3.3	114
3.4	Algebraic Laws for Regular Expressions	115
3.4.1	Associativity and Commutativity	115
3.4.2	Identities and Annihilators	116
3.4.3	Distributive Laws	116
3.4.4	The Idempotent Law	117
3.4.5	Laws Involving Closures	118
3.4.6	Discovering Laws for Regular Expressions	118
3.4.7	The Test for a Regular-Expression Algebraic Law	120
3.4.8	Exercises for Section 3.4	121
3.5	Summary of Chapter 3	123
3.6	Gradience Problems for Chapter 3	123
3.7	References for Chapter 3	125
4	Properties of Regular Languages	127
4.1	Proving Languages Not to Be Regular	128
4.1.1	The Pumping Lemma for Regular Languages	128
4.1.2	Applications of the Pumping Lemma	129
4.1.3	Exercises for Section 4.1	131
4.2	Closure Properties of Regular Languages	133
4.2.1	Closure of Regular Languages Under Boolean Operations	133
4.2.2	Reversal	139
4.2.3	Homomorphisms	140
4.2.4	Inverse Homomorphisms	142
4.2.5	Exercises for Section 4.2	147
4.3	Decision Properties of Regular Languages	150
4.3.1	Converting Among Representations	151
4.3.2	Testing Emptiness of Regular Languages	153
4.3.3	Testing Membership in a Regular Language	154
4.3.4	Exercises for Section 4.3	155
4.4	Equivalence and Minimization of Automata	155
4.4.1	Testing Equivalence of States	155
4.4.2	Testing Equivalence of Regular Languages	159
4.4.3	Minimization of DFA's	160
4.4.4	Why the Minimized DFA Can't Be Beaten	163
4.4.5	Exercises for Section 4.4	165
4.5	Summary of Chapter 4	166
4.6	Gradience Problems for Chapter 4	167
4.7	References for Chapter 4	169

5	Context-Free Grammars and Languages	171
5.1	Context-Free Grammars	171
5.1.1	An Informal Example	172
5.1.2	Definition of Context-Free Grammars	173
5.1.3	Derivations Using a Grammar	175
5.1.4	Leftmost and Rightmost Derivations	177
5.1.5	The Language of a Grammar	179
5.1.6	Sentential Forms	180
5.1.7	Exercises for Section 5.1	181
5.2	Parse Trees	183
5.2.1	Constructing Parse Trees	183
5.2.2	The Yield of a Parse Tree	185
5.2.3	Inference, Derivations, and Parse Trees	185
5.2.4	From Inferences to Trees	187
5.2.5	From Trees to Derivations	188
5.2.6	From Derivations to Recursive Inferences	191
5.2.7	Exercises for Section 5.2	193
5.3	Applications of Context-Free Grammars	193
5.3.1	Parsers	194
5.3.2	The YACC Parser-Generator	196
5.3.3	Markup Languages	197
5.3.4	XML and Document-Type Definitions	200
5.3.5	Exercises for Section 5.3	206
5.4	Ambiguity in Grammars and Languages	207
5.4.1	Ambiguous Grammars	207
5.4.2	Removing Ambiguity From Grammars	209
5.4.3	Leftmost Derivations as a Way to Express Ambiguity	212
5.4.4	Inherent Ambiguity	213
5.4.5	Exercises for Section 5.4	215
5.5	Summary of Chapter 5	216
5.6	Gradience Problems for Chapter 5	218
5.7	References for Chapter 5	224
6	Pushdown Automata	225
6.1	Definition of the Pushdown Automaton	225
6.1.1	Informal Introduction	225
6.1.2	The Formal Definition of Pushdown Automata	227
6.1.3	A Graphical Notation for PDA's	229
6.1.4	Instantaneous Descriptions of a PDA	230
6.1.5	Exercises for Section 6.1	233
6.2	The Languages of a PDA	234
6.2.1	Acceptance by Final State	235
6.2.2	Acceptance by Empty Stack	236
6.2.3	From Empty Stack to Final State	237
6.2.4	From Final State to Empty Stack	240

6.2.5	Exercises for Section 6.2	241
6.3	Equivalence of PDA's and CFG's	243
6.3.1	From Grammars to Pushdown Automata	243
6.3.2	From PDA's to Grammars	247
6.3.3	Exercises for Section 6.3	251
6.4	Deterministic Pushdown Automata	252
6.4.1	Definition of a Deterministic PDA	252
6.4.2	Regular Languages and Deterministic PDA's	253
6.4.3	DPDA's and Context-Free Languages	254
6.4.4	DPDA's and Ambiguous Grammars	255
6.4.5	Exercises for Section 6.4	256
6.5	Summary of Chapter 6	257
6.6	Gradience Problems for Chapter 6	258
6.7	References for Chapter 6	260
7	Properties of Context-Free Languages	261
7.1	Normal Forms for Context-Free Grammars	261
7.1.1	Eliminating Useless Symbols	262
7.1.2	Computing the Generating and Reachable Symbols	264
7.1.3	Eliminating ϵ -Productions	265
7.1.4	Eliminating Unit Productions	268
7.1.5	Chomsky Normal Form	272
7.1.6	Exercises for Section 7.1	275
7.2	The Pumping Lemma for Context-Free Languages	279
7.2.1	The Size of Parse Trees	280
7.2.2	Statement of the Pumping Lemma	280
7.2.3	Applications of the Pumping Lemma for CFL's	283
7.2.4	Exercises for Section 7.2	286
7.3	Closure Properties of Context-Free Languages	287
7.3.1	Substitutions	287
7.3.2	Applications of the Substitution Theorem	289
7.3.3	Reversal	290
7.3.4	Intersection With a Regular Language	291
7.3.5	Inverse Homomorphism	295
7.3.6	Exercises for Section 7.3	297
7.4	Decision Properties of CFL's	299
7.4.1	Complexity of Converting Among CFG's and PDA's	299
7.4.2	Running Time of Conversion to Chomsky Normal Form	301
7.4.3	Testing Emptiness of CFL's	302
7.4.4	Testing Membership in a CFL	303
7.4.5	Preview of Undecidable CFL Problems	307
7.4.6	Exercises for Section 7.4	307
7.5	Summary of Chapter 7	308
7.6	Gradience Problems for Chapter 7	309
7.7	References for Chapter 7	314

8	Introduction to Turing Machines	315
8.1	Problems That Computers Cannot Solve	315
8.1.1	Programs that Print “Hello, World”	316
8.1.2	The Hypothetical “Hello, World” Tester	318
8.1.3	Reducing One Problem to Another	321
8.1.4	Exercises for Section 8.1	324
8.2	The Turing Machine	324
8.2.1	The Quest to Decide All Mathematical Questions	325
8.2.2	Notation for the Turing Machine	326
8.2.3	Instantaneous Descriptions for Turing Machines	327
8.2.4	Transition Diagrams for Turing Machines	331
8.2.5	The Language of a Turing Machine	334
8.2.6	Turing Machines and Halting	334
8.2.7	Exercises for Section 8.2	335
8.3	Programming Techniques for Turing Machines	337
8.3.1	Storage in the State	337
8.3.2	Multiple Tracks	339
8.3.3	Subroutines	341
8.3.4	Exercises for Section 8.3	343
8.4	Extensions to the Basic Turing Machine	343
8.4.1	Multitape Turing Machines	344
8.4.2	Equivalence of One-Tape and Multitape TM’s	345
8.4.3	Running Time and the Many-Tapes-to-One Construction	346
8.4.4	Nondeterministic Turing Machines	347
8.4.5	Exercises for Section 8.4	349
8.5	Restricted Turing Machines	352
8.5.1	Turing Machines With Semi-infinite Tapes	352
8.5.2	Multistack Machines	355
8.5.3	Counter Machines	358
8.5.4	The Power of Counter Machines	359
8.5.5	Exercises for Section 8.5	361
8.6	Turing Machines and Computers	362
8.6.1	Simulating a Turing Machine by Computer	362
8.6.2	Simulating a Computer by a Turing Machine	363
8.6.3	Comparing the Running Times of Computers and Turing Machines	368
8.7	Summary of Chapter 8	370
8.8	Gradience Problems for Chapter 8	372
8.9	References for Chapter 8	374
9	Undecidability	377
9.1	A Language That Is Not Recursively Enumerable	378
9.1.1	Enumerating the Binary Strings	379
9.1.2	Codes for Turing Machines	379
9.1.3	The Diagonalization Language	380

9.1.4	Proof That L_d Is Not Recursively Enumerable	382
9.1.5	Exercises for Section 9.1	382
9.2	An Undecidable Problem That Is RE	383
9.2.1	Recursive Languages	383
9.2.2	Complements of Recursive and RE languages	384
9.2.3	The Universal Language	387
9.2.4	Undecidability of the Universal Language	389
9.2.5	Exercises for Section 9.2	390
9.3	Undecidable Problems About Turing Machines	392
9.3.1	Reductions	392
9.3.2	Turing Machines That Accept the Empty Language	394
9.3.3	Rice's Theorem and Properties of the RE Languages	397
9.3.4	Problems about Turing-Machine Specifications	399
9.3.5	Exercises for Section 9.3	400
9.4	Post's Correspondence Problem	401
9.4.1	Definition of Post's Correspondence Problem	401
9.4.2	The "Modified" PCP	404
9.4.3	Completion of the Proof of PCP Undecidability	407
9.4.4	Exercises for Section 9.4	412
9.5	Other Undecidable Problems	412
9.5.1	Problems About Programs	413
9.5.2	Undecidability of Ambiguity for CFG's	413
9.5.3	The Complement of a List Language	415
9.5.4	Exercises for Section 9.5	418
9.6	Summary of Chapter 9	419
9.7	Gradience Problems for Chapter 9	420
9.8	References for Chapter 9	422
10	Intractable Problems	425
10.1	The Classes $\mathcal{P}$ and $\mathcal{NP}$	426
10.1.1	Problems Solvable in Polynomial Time	426
10.1.2	An Example: Kruskal's Algorithm	426
10.1.3	Nondeterministic Polynomial Time	431
10.1.4	An $\mathcal{NP}$ Example: The Traveling Salesman Problem	431
10.1.5	Polynomial-Time Reductions	433
10.1.6	NP-Complete Problems	434
10.1.7	Exercises for Section 10.1	435
10.2	An NP-Complete Problem	438
10.2.1	The Satisfiability Problem	438
10.2.2	Representing SAT Instances	439
10.2.3	NP-Completeness of the SAT Problem	440
10.2.4	Exercises for Section 10.2	447
10.3	A Restricted Satisfiability Problem	447
10.3.1	Normal Forms for Boolean Expressions	448
10.3.2	Converting Expressions to CNF	449

10.3.3	NP-Completeness of CSAT	452
10.3.4	NP-Completeness of 3SAT	456
10.3.5	Exercises for Section 10.3	458
10.4	Additional NP-Complete Problems	458
10.4.1	Describing NP-complete Problems	459
10.4.2	The Problem of Independent Sets	459
10.4.3	The Node-Cover Problem	463
10.4.4	The Directed Hamilton-Circuit Problem	465
10.4.5	Undirected Hamilton Circuits and the TSP	471
10.4.6	Summary of NP-Complete Problems	473
10.4.7	Exercises for Section 10.4	473
10.5	Summary of Chapter 10	477
10.6	Gradience Problems for Chapter 10	478
10.7	References for Chapter 10	481
11	Additional Classes of Problems	483
11.1	Complements of Languages in $\mathcal{NP}$	484
11.1.1	The Class of Languages $\text{Co-}\mathcal{NP}$	484
11.1.2	NP-Complete Problems and $\text{Co-}\mathcal{NP}$	485
11.1.3	Exercises for Section 11.1	486
11.2	Problems Solvable in Polynomial Space	487
11.2.1	Polynomial-Space Turing Machines	487
11.2.2	Relationship of $\mathcal{PS}$ and $\mathcal{NPS}$ to Previously Defined Classes	488
11.2.3	Deterministic and Nondeterministic Polynomial Space . .	490
11.3	A Problem That Is Complete for $\mathcal{PS}$	492
11.3.1	PS-Completeness	492
11.3.2	Quantified Boolean Formulas	493
11.3.3	Evaluating Quantified Boolean Formulas	494
11.3.4	PS-Completeness of the QBF Problem	496
11.3.5	Exercises for Section 11.3	501
11.4	Language Classes Based on Randomization	501
11.4.1	Quicksort: an Example of a Randomized Algorithm . . .	502
11.4.2	A Turing-Machine Model Using Randomization	503
11.4.3	The Language of a Randomized Turing Machine	504
11.4.4	The Class $\mathcal{RP}$	506
11.4.5	Recognizing Languages in $\mathcal{RP}$	508
11.4.6	The Class $\mathcal{ZPP}$	509
11.4.7	Relationship Between $\mathcal{RP}$ and $\mathcal{ZPP}$	510
11.4.8	Relationships to the Classes $\mathcal{P}$ and $\mathcal{NP}$	511
11.5	The Complexity of Primality Testing	512
11.5.1	The Importance of Testing Primality	512
11.5.2	Introduction to Modular Arithmetic	514
11.5.3	The Complexity of Modular-Arithmetic Computations . .	516
11.5.4	Random-Polynomial Primality Testing	517
11.5.5	Nondeterministic Primality Tests	518

TABLE OF CONTENTS

xvii

11.5.6 Exercises for Section 11.5	521
11.6 Summary of Chapter 11	522
11.7 Gradiance Problems for Chapter 11	523
11.8 References for Chapter 11	524
Index	527