HZ BOOKS

PEARSON
Addison Wesley

经 典 原 版 书 库

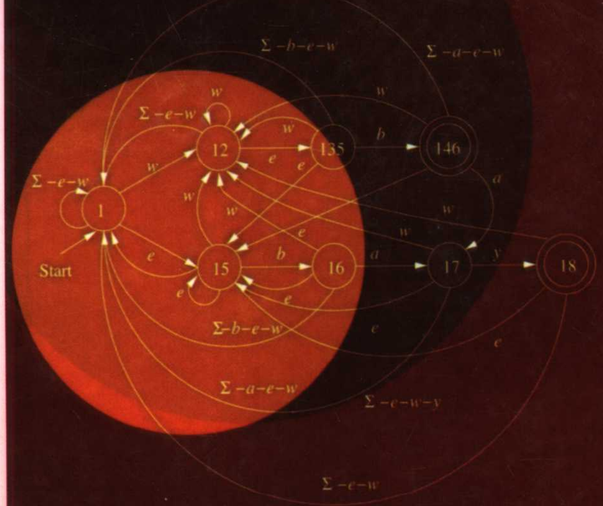
自动机理论、语言和计算导论

（英文版 · 第3版）

INTRODUCTION TO

Automata Theory, Languages, and Computation

3rd Edition



JOHN E. HOPCROFT
RAJEEV MOTW
JEFFREY D. UL

(美) John E. Hopcroft 康奈尔大学
Rajeev Motwani 斯坦福大学
Jeffrey D. Ullman 斯坦福大学 著

机械工业出版社
China Machine Press

TP301/Y11=2
2008

经 典 原 版

自动机理论、语言和计算导论

（英文版 · 第3版）

Introduction to Automata Theory, Languages, and Computation

(Third Edition)

江苏工业学院图书馆
藏书章

（美） John E. Hopcroft 康奈尔大学
Rajeev Motwani 斯坦福大学
Jeffrey D. Ullman 斯坦福大学 著

机械工业出版社
China Machine Press

English reprint edition copyright © 2008 by Pearson Education Asia Limited and China Machine Press.

Original English language title: *Introduction to Automata Theory, Languages, and Computation, Third Edition*(ISBN 0-321-45536-3) by John E. Hopcroft, Rajeev Motwani, and Jeffrey D. Ullman, Copyright © 2007 by Pearson Education, Inc.

All rights reserved.

Published by arrangement with the original publisher, Pearson Education, Inc., publishing as Addison-Wesley.

For sale and distribution in the People's Republic of China exclusively (except Taiwan, Hong Kong SAR and Macau SAR).

本书英文影印版由Pearson Education Asia Ltd. 授权机械工业出版社独家出版。未经出版者书面许可，不得以任何方式复制或抄袭本书内容。

仅限于中华人民共和国境内（不包括中国香港、澳门特别行政区和中国台湾地区）销售发行。

本书封面贴有Pearson Education（培生教育出版集团）激光防伪标签，无标签者不得销售。

版权所有，侵权必究。

本书法律顾问　北京市展达律师事务所

本书版权登记号：图字：01-2007-4209

图书在版编目（CIP）数据

自动机理论、语言和计算导论（英文版·第3版）/（美）霍普克罗夫特（Hopcroft，J. E.）等著. -北京：机械工业出版社，2007.10

（经典原版书库）

书名原文：Introduction to Automata Theory, Languages, and Computation, Third Edition

ISBN 978-7-111-22392-4

Ⅰ. 自…　Ⅱ. 霍…　Ⅲ. 自动机理论-英文　Ⅳ. TP301.1

中国版本图书馆CIP数据核字（2007）第147152号

机械工业出版社（北京市西城区百万庄大街22号　邮政编码　100037）

责任编辑：迟振春

三河市明辉印装有限公司印刷·新华书店北京发行所发行

2008年1月第1版第1次印刷

170mm×242mm·34.75印张

定价：59.00元

凡购本书，如有倒页、脱页、缺页，由本社发行部调换

本社购书热线：(010) 68326294

出版者的话

文艺复兴以降，源远流长的科学精神和逐步形成的学术规范，使西方国家在自然科学的各个领域取得了垄断性的优势；也正是这样的传统，使美国在信息技术发展的六十多年间名家辈出、独领风骚。在商业化的进程中，美国的产业界与教育界越来越紧密地结合，计算机学科中的许多泰山北斗同时身处科研和教学的最前线，由此而产生的经典科学著作，不仅擘划了研究的范畴，还揭橥了学术的源变，既遵循学术规范，又自有学者个性，其价值并不会因年月的流逝而减退。

近年，在全球信息化大潮的推动下，我国的计算机产业发展迅猛，对专业人才的需求日益迫切。这对计算机教育界和出版界都既是机遇，也是挑战；而专业教材的建设在教育战略上显得举足轻重。在我国信息技术发展时间较短、从业人员较少的现状下，美国等发达国家在其计算机科学发展的几十年间积淀的经典教材仍有许多值得借鉴之处。因此，引进一批国外优秀计算机教材将对我国计算机教育事业的发展起积极的推动作用，也是与世界接轨、建设真正的世界一流大学的必由之路。

机械工业出版社华章图文信息有限公司较早意识到“出版要为教育服务”。自1998年开始，华章公司就将工作重点放在了遴选、移译国外优秀教材上。经过几年的不懈努力，我们与Prentice Hall，Addison-Wesley，McGraw-Hill，Morgan Kaufmann等世界著名出版公司建立了良好的合作关系，从它们现有的数百种教材中甄选出Tanenbaum，Stroustrup，Kernighan，Jim Gray等大师名家的一批经典作品，以“计算机科学丛书”为总称出版，供读者学习、研究及庋藏。大理石纹理的封面，也正体现了这套丛书的品位和格调。

“计算机科学丛书”的出版工作得到了国内外学者的鼎力襄助，国内的专家不仅提供了中肯的选题指导，还不辞劳苦地担任了翻译和审校的工作；而原书的作者也相当关注其作品在中国的传播，有的还专程为其书的中译本作序。迄今，“计算机科学丛书”已经出版了近260个品种，这些书籍在读者中树立了良好的口碑，并被许多高校采用为正式教材和参考书籍，为进一步推广与发展打下了坚实的基础。

随着学科建设的初步完善和教材改革的逐渐深化，教育界对国外计算机教材的需求和应用都步入一个新的阶段。为此，华章公司将加大引进教材的力度，除“计算机科学丛书”之外，对影印版的教材，则单独开辟出“经典原版书库”。为了保证这两套丛书的权威性，同时也为了更好地为学校和老师们服务，华章公司聘请了中国科学院、北京大学、清华大学、国防科技大学、复旦大学、上海交通

大学、南京大学、浙江大学、中国科技大学、哈尔滨工业大学、西安交通大学、中国人民大学、北京航空航天大学、北京邮电大学、中山大学、解放军理工大学、郑州大学、湖北工学院、中国国家信息安全测评认证中心等国内重点大学和科研机构在计算机的各个领域的著名学者组成“专家指导委员会”，为我们提供选题意见和出版监督。

这两套丛书是响应教育部提出的使用外版教材的号召，为国内高校的计算机及相关专业的教学度身订造的。其中许多教材均已为M. I. T.，Stanford，U.C. Berkeley，C. M. U. 等世界名牌大学所采用。不仅涵盖了程序设计、数据结构、操作系统、计算机体系结构、数据库、编译原理、软件工程、图形学、通信与网络、离散数学等国内大学计算机专业普遍开设的核心课程，而且各具特色——有的出自语言设计者之手、有的历经三十年而不衰、有的已被全世界的几百所高校采用。在这些圆熟通博的名师大作的指引之下，读者必将在计算机科学的宫殿中由登堂而入室。

权威的作者、经典的教材、一流的译者、严格的审校、精细的编辑，这些因素使我们的图书有了质量的保证，但我们的目标是尽善尽美，而反馈的意见正是我们达到这一终极目标的重要帮助。教材的出版只是我们的后续服务的起点。华章公司欢迎老师和读者对我们的工作提出建议或给予指正，我们的联系方法如下：

电子邮件：hzjsj@hzbook.com
联系电话：(010) 68995264
联系地址：北京市西城区百万庄南街1号
邮政编码：100037

此为试读，需要完整PDF请访问：www.ertongbook.com

专家指导委员会

（按姓氏笔画顺序）

尤晋元	王　珊	冯博琴	史忠植	史美林
石教英	吕　建	孙玉芳	吴世忠	吴时霖
张立昂	李伟琴	李师贤	李建中	杨冬青
邵维忠	陆丽娜	陆鑫达	陈向群	周伯生
周克定	周傲英	孟小峰	岳丽华	范　明
郑国梁	施伯乐	钟玉琢	唐世渭	袁崇义
高传善	梅　宏	程　旭	程时端	谢希仁
裘宗燕	戴　葵			

Preface

In the preface from the 1979 predecessor to this book, Hopcroft and Ullman marveled at the fact that the subject of automata had exploded, compared with its state at the time they wrote their first book, in 1969. Truly, the 1979 book contained many topics not found in the earlier work and was about twice its size. If you compare this book with the 1979 book, you will find that, like the automobiles of the 1970's, this book is "larger on the outside, but smaller on the inside." That sounds like a retrograde step, but we are happy with the changes for several reasons.

First, in 1979, automata and language theory was still an area of active research. A purpose of that book was to encourage mathematically inclined students to make new contributions to the field. Today, there is little direct research in automata theory (as opposed to its applications), and thus little motivation for us to retain the succinct, highly mathematical tone of the 1979 book.

Second, the role of automata and language theory has changed over the past two decades. In 1979, automata was largely a graduate-level subject, and we imagined our reader was an advanced graduate student, especially those using the later chapters of the book. Today, the subject is a staple of the undergraduate curriculum. As such, the content of the book must assume less in the way of prerequisites from the student, and therefore must provide more of the background and details of arguments than did the earlier book.

A third change in the environment is that Computer Science has grown to an almost unimaginable degree in the past three decades. While in 1979 it was often a challenge to fill up a curriculum with material that we felt would survive the next wave of technology, today very many subdisciplines compete for the limited amount of space in the undergraduate curriculum.

Fourthly, CS has become a more vocational subject, and there is a severe pragmatism among many of its students. We continue to believe that aspects of automata theory are essential tools in a variety of new disciplines, and we believe that the theoretical, mind-expanding exercises embodied in the typical automata course retain their value, no matter how much the student prefers to learn only the most immediately monetizable technology. However, to assure a continued place for the subject on the menu of topics available to the computer science student, we believe it is necessary to emphasize the applications along with the mathematics. Thus, we have replaced a number of the more abstruse topics in the earlier book with examples of how the ideas are used today. While applications of automata and language theory to compilers are now so well understood that they are normally covered in a compiler course, there are a variety of more recent uses, including model-checking algorithms to verify protocols and document-description languages that are patterned on context-free grammars.

A final explanation for the simultaneous growth and shrinkage of the book is that we were today able to take advantage of the TEX and LATEX typesetting systems developed by Don Knuth and Les Lamport. The latter, especially, encourages the "open" style of typesetting that makes books larger, but easier to read. We appreciate the efforts of both men.

Use of the Book

This book is suitable for a quarter or semester course at the Junior level or above. At Stanford, we have used the notes in CS154, the course in automata and language theory. It is a one-quarter course, which both Rajeev and Jeff have taught. Because of the limited time available, Chapter 11 is not covered, and some of the later material, such as the more difficult polynomial-time reductions in Section 10.4 are omitted as well. The book's Web site (see below) includes notes and syllabi for several offerings of CS154.

Some years ago, we found that many graduate students came to Stanford with a course in automata theory that did not include the theory of intractability. As the Stanford faculty believes that these ideas are essential for every computer scientist to know at more than the level of "NP-complete means it takes too long," there is another course, CS154N, that students may take to cover only Chapters 8, 9, and 10. They actually participate in roughly the last third of CS154 to fulfill the CS154N requirement. Even today, we find several students each quarter availing themselves of this option. Since it requires little extra effort, we recommend the approach.

Prerequisites

To make best use of this book, students should have taken previously a course covering discrete mathematics, e.g., graphs, trees, logic, and proof techniques. We assume also that they have had several courses in programming, and are familiar with common data structures, recursion, and the role of major system components such as compilers. These prerequisites should be obtained in a typical freshman-sophomore CS program.

Exercises

The book contains extensive exercises, with some for almost every section. We indicate harder exercises or parts of exercises with an exclamation point. The hardest exercises have a double exclamation point.

Some of the exercises or parts are marked with a star. For these exercises, we shall endeavor to maintain solutions accessible through the book's Web page. These solutions are publicly available and should be used for self-testing. Note that in a few cases, one exercise *B* asks for modification or adaptation of your solution to another exercise *A*. If certain parts of *A* have solutions, then you should expect the corresponding parts of *B* to have solutions as well.

Gradiance On-Line Homeworks

A new feature of the third edition is that there is an accompanying set of on-line homeworks using a technology developed by Gradiance Corp. Instructors may assign these homeworks to their class, or students not enrolled in a class may enroll in an "omnibus class" that allows them to do the homeworks as a tutorial (without an instructor-created class). Gradiance questions look like ordinary questions, but your solutions are sampled. If you make an incorrect choice you are given specific advice or feedback to help you correct your solution. If your instructor permits, you are allowed to try again, until you get a perfect score.

A subscription to the Gradiance service is offered with all new copies of this text sold in North America. For more information, visit the Addison-Wesley web site `www.aw.com/gradiance` or send email to `computing@aw.com`.

Support on the World Wide Web

The book's home page is

```
http://www-db.stanford.edu/~ullman/ialc.html
```

Here are solutions to starred exercises, errata as we learn of them, and backup materials. We hope to make available the notes for each offering of CS154 as we teach it, including homeworks, solutions, and exams.

Acknowledgements

A handout on "how to do proofs" by Craig Silverstein influenced some of the material in Chapter 1. Comments and errata on drafts of the second edition (2000) were received from: Zoe Abrams, George Candea, Haowen Chen, Byong-Gun Chun, Jeffrey Shallit, Bret Taylor, Jason Townsend, and Erik Uzureau.

We also received many emails pointing out errata in the second edition of this book, and these were acknowledged on-line in the errata sheets for that

edition. However, we would like to mention here the following people who provided large numbers of significant errata: Zeki Bayram, Sebastian Hick, Kang-Rae Lee, Christian Lemburg, Nezam Mahdavi-Amiri, Dave Maier, A. P. Marathe, Mark Meuleman, Mustafa Sait-Ametov, Alexey Sarytchev, Jukka Suomela, Rod Topor, Po-Lian Tsai, Tom Whaley, Aaron Windsor, and Jacinth H.T. Wu.

The help of all these people is greatefully acknowledged. Remaining errors are ours, of course.

J. E. H.
R. M.
J. D. U.
Ithaca NY and Stanford CA
February, 2006

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

此为试读,需要完整PDF请访问: www.ertongbook.com

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 Gradiance 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 Gradiance 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 Gradiance 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 Gradiance 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 Gradiance 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 Gradiance 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 Gradiance 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 Gradiance 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

此为试读,需要完整PDF请访问: www.ertongbook.com