

PEARSON
Prentice
Hall

大学计算机教育国外著名教材系列



Automata, Computability, and Complexity
Theory and Applications

自动机理论与应用



Elaine Rich 著



清华大学出版社

大学计算机教育国外著名教材系列（影印版）

Automata, Computability, and Complexity

Theory and Applications

自动机理论与应用

Elaine Rich 著

清华大学出版社
北京

English reprint edition copyright © 2009 by PEARSON EDUCATION ASIA LIMITED and TSINGHUA UNIVERSITY PRESS.

Original English language title from Proprietor's edition of the Work.

Original English language title: Automata, Computability, and Complexity: Theory and Applications by Elaine Rich, Copyright © 2009

All Rights Reserved.

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

This edition is authorized for sale and distribution only in the People's Republic of China (excluding the Special Administrative Region of Hong Kong, Macao SAR and Taiwan).

本书影印版由 Pearson Education (培生教育出版集团) 授权给清华大学出版社出版发行。

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

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。

北京市版权局著作权合同登记号 图字 01-2009-4346 号

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。侵权举报电话: 010-62782989 13701121933

图书在版编目(CIP)数据

自动机理论与应用=Automata, Computability, and Complexity: Theory and Applications: 英文 /

(美) 里奇 (Rich, E.) 著. —北京: 清华大学出版社, 2009.11

(大学计算机教育国外著名教材系列)

ISBN 978-7-302-21293-5

I. 自… II. 里… III. 自动机理论—高等学校—教材—英文 IV. TP301.1

中国版本图书馆 CIP 数据核字 (2009) 第 181938 号

责任印制: 李红英

出版发行: 清华大学出版社

<http://www.tup.com.cn>

社 总 机: 010-62770175

投稿与读者服务: 010-62776969, c-service@tup.tsinghua.edu.cn

质 量 反 馈: 010-62772015, zhiliang@tup.tsinghua.edu.cn

印 刷 者: 清华大学印刷厂

装 订 者: 三河市金元印装有限公司

发 行 者: 全国新华书店

开 本: 185×230 印张: 70.5

版 次: 2009 年 11 月第 1 版

印 次: 2009 年 11 月第 1 次印刷

印 数: 1~3000

定 价: 99.00 元

本书如存在文字不清、漏印、缺页、倒页、脱页等印装质量问题, 请与清华大学出版社出版部联系调换。联系电话: 010-62770177 转 3103 产品编号: 023235-01

出版说明

进入 21 世纪, 世界各国的经济、科技以及综合国力的竞争将更加激烈。竞争的中心无疑是对人才的竞争。谁拥有大量高素质的人才, 谁就能在竞争中取得优势。高等教育, 作为培养高素质人才的事业, 必然受到高度重视。目前我国高等教育的教材更新较慢, 为了加快教材的更新频率, 教育部正在大力促进我国高校采用国外原版教材。

清华大学出版社从 1996 年开始, 与国外著名出版公司合作, 影印出版了“大学计算机教育丛书(影印版)”等一系列引进图书, 受到国内读者的欢迎和支持。跨入 21 世纪, 我们本着为我国高等教育教材建设服务的初衷, 在已有的基础上, 进一步扩大选题内容, 改变图书开本尺寸, 一如既往地请有关专家挑选适用于我国高校本科及研究生计算机教育的国外经典教材或著名教材, 组成本套“大学计算机教育国外著名教材系列(影印版)”, 以飨读者。深切期盼读者及时将使用本系列教材的效果和意见反馈给我们。更希望国内专家、教授积极向我们推荐国外计算机教育的优秀教材, 以利我们把“大学计算机教育国外著名教材系列(影印版)”做得更好, 更适合高校师生的需要。

清华大学出版社

PREFACE

This book has three goals:

1. To introduce students to the elegant theory that underlies modern computing.
2. To motivate students by showing them that the theory is alive. While much of it has been known since the early days of digital computers (and some of it even longer), the theory continues to inform many of the most important applications that are considered today.
3. To show students how to start looking for ways to exploit the theory in their own work.

The core of the book, as a standard textbook, is Parts I through V. They address the first of the stated goals. They contain the theory that is being presented. There is more material in them than can be covered in a one-semester course. Sections that are marked with a ● are optional, in the sense that later material does not, for the most part, depend on them. The Course Plans section on page xv suggests ways of selecting sections that are appropriate for some typical computer science courses.

Then there are seventeen appendices:

- Appendix A reviews the mathematical concepts on which the main text relies. Students should be encouraged to review it during the first week of class.
- Appendix B describes techniques for working with logical formulas (both Boolean and first-order).
- Appendices C, D, E and F treat selected theoretical concepts in greater depth. In particular, they contain the details of some proofs that are only sketched in the main text.
- Appendices G through Q address the second and third goals. They describe applications of the techniques that are described in the main body of the book. They also contain some interesting historical material. Although they are long (at least in comparison to the space that is devoted to applications in most other books in this area), they only skim the surface of the applications that they present. But my hope is that that is enough. The World Wide Web has completely changed our ability to access knowledge. What matters now is to know that something exists and thus to look for it. The short discussions that are presented in these appendices will, I hope, give students that understanding.

There is a Web site that accompanies this book: <http://www.theoryandapplications.org/>. It is organized into the same sections as the book, so that it is easy to follow the two in parallel. The symbol □ following a concept in the text means that additional material is available on the Web site.

Throughout the text, you'll find pointers to the material in the appendices, as well as to material on the book's Web site. There are also some standalone application notes. These pointers and notes are enclosed in boxes, and refer you to the appropriate appendix and page number or to the Web. The appendix references look like this:

This technique really is useful. (H. 1. 2.)

Notation

It is common practice to write definitions in the following form:

A something is a *special something* if it possesses property P .

This form is used even though property P is not only a sufficient but also a necessary condition for being a special something. For clarity we will, in those cases, write “if and only if”, abbreviated “iff”, instead of “if”. So we will write:

A something is a *special something* iff it possesses property P .

Throughout the book we will, with a few exceptions, use the following naming conventions:

		Examples
sets	capital letters, early in the alphabet, plus S	A, B, C, D, S
logical formulas	capital letters, middle of the alphabet	P, Q, R
predicates and relations	capital letters, middle of the alphabet	P, Q, R
logical constants	subscripted X 's and specific names	X_1, X_2 , John, Smoky
functions	lower case letters or words	f, g , convert
integers	lower case letters, middle of the alphabet	i, j, k, l, m, n
string-valued variables	lower case letters, late in the alphabet	s, t, u, v, w, x, y
literal strings	written in computer font	abc, aabbbb
language-valued variables	upper case letters starting with L	L, L_1, L_2
specific languages	nonitalicized strings	$A^n B^n$, WW
regular expressions	lower case Greek letters	α, β, γ
states	lower case letters, middle of the alphabet	p, q, r, s, t
nonterminals in grammar rules	upper case letters	A, B, C, S, T
working strings in grammatical derivations	lower case Greek letter	α, β, γ
strings representing a PDA's stack	lower case Greek letter	α, β, γ
other variables	lower case letters, late in the alphabet	x, y, z

Programs and algorithms will appear throughout the book, stated at varying levels of detail. We will use the following formats for describing them:

- Exact code in some particular programming language will be written the same way other strings are written.

- Algorithms that are described in pseudocode will be written as:

Until an even-length string is found do:

Generate the next string in the sequence.

When we want to be able to talk about the steps, they will be numbered, so we will write:

1. Until an even-length string is found do:

1.1. Generate the next string in the sequence.

2. Reverse the string that was found.

When comments are necessary, as for example in code or in grammars, they will be preceded by the string `/*`.

Course Plans

Appendix A summarizes the mathematical concepts on which the rest of the book relies. Depending on the background of the students, it may be appropriate to spend one or more lectures on this material. At the University of Texas, our students have had two prior courses in logic and discrete structures before they arrive in my class, so I have found that it is sufficient just to ask the students to read Appendix A and to work a selection of the exercises that are provided at the end of it.

Part I lays the groundwork for the rest of the book. Chapter 2 is essential, since it defines the fundamental structures: strings and languages. I have found that it is very useful to cover Chapter 3, which presents a roadmap for the rest of the material. It helps students see where we are going and how each piece of the theory fits into the overall picture of a theory of computation. Chapter 4 introduces three ideas that become important later in the book. I have found that it may be better to skip Chapter 4 at the beginning of my class and to return to each of its sections once or twice later, as the concepts are required.

If the optional sections are omitted, Chapters 5, 6, 8, 9, 11–14, 17–21, and, optionally, 23 and/or 24 cover the material in a standard course in Automata Theory. Chapter 15 (Context-Free Parsing) contains material that many computer science students need to see and it fits well into an Automata Theory course. I used to include much of it in my class. But that material is often taught in a course on Programming Languages or Compilers. In that case, it makes sense to omit it from the Automata Theory course. In its place, I now cover the optional material in Chapter 5, particularly the section on stochastic finite automata. I also cover Chapter 22. I've found that students are more motivated to tackle the difficult material (particularly the design of reduction proofs) in Chapter 21 if they can see ways in which the theory of undecidability applies to problems that are, to them, more intriguing than questions about the behavior of Turing machines.

This text is also appropriate for a broader course that includes the core of the classic theory of automata plus the modern theory of complexity. Such a course might

cover Chapters 2–3, 5, 8, 11, 13, 17–21, and 27–30, omitting sections as time pressures require.

This text is unique in the amount of space it devotes to applications of the core theoretical material. In order to make the application discussions coherent, they are separated from the main text and occur in the appendices at the end of the book. But I have found that I can substantially increase student interest in my course by sprinkling application discussions throughout the term. The application references that occur in the main text suggest places where it makes sense to do that.

Resources for Instructors

There is a website, www.prenhall.com/rich, that contains materials that have been designed to make it easy to teach from this book. In particular, it contains:

- a complete set of Powerpoint slides,
- solutions to many of the Exercises, and
- additional problems, many of them with solutions.

I would like to invite instructors who use this book to send me additional problems that can be shared with other users.

ACKNOWLEDGMENTS

This book would not have been possible without the help of many people. When I first began teaching CS 341, Automata Theory, at the University of Texas, I was given a collection of notes that had been written by Bob Wall and Russell Williams. Much of the material in this book has evolved from those notes. I first learned automata theory from [Hopcroft and Ullman 1969]. Over the years that I have taught CS 341, I have used several textbooks, most frequently [Lewis and Papadimitriou 1988] and [Sipser 2006]. Much of what I have written here has been heavily influenced by the treatment of this material in those books.

Several of my friends, colleagues, and students have provided examples, answered numerous questions, and critiqued what I have written. I am particularly indebted to Don Baker, Volker Bandke, Jim Barnett, Jon Bentley, Gary Bland, Jaime Carbonell, Alan Cline, Martin Cohn, Dan Connolly, Ann Daniel, Chris Edmonson-Yurkanan, Scott Fahlman, Warren Gish, Mohamed Gouda, Jim Hendler, Oscar Hernandez, David Jefferson, Ben Kuipers, Greg Lavender, Tim Maxwell, Andy Mills, Jay Misra, Luay Nakhleh, Gordon Novak, Gabriela Ochoa, Dewayne Perry, Brian Reid, Bob Rich, Mike Scott, Cathy Stacy, Peter Stone, Lynda Trader, and David Zuckerman. Luay Nakhleh, Dan Tamir, and Bob Wall have used drafts of this book in their classes. I thank them for their feedback and that of their students.

I would also like to thank all of the students and teaching assistants who have helped me understand both why this material is hard and why it is exciting and useful. A couple of years ago, Tarang Mittal and Mat Crocker finished my class and decided that they should create an organized automata theory tutoring program the following fall. They got the program going and it continues to make a big difference to many students. I'd like to thank Tarang and Mat and the other tutors: Jason Pennington, Alex Menzies, Tim Maxwell, Chris St. Clair, Luis Guimbarda, Peter Olah, Eamon White, Kevin Kwast, Catherine Chu, Siddharth Natarajan, Daniel Galvan, Elton Pinto, and Jack Djeu.

My students have helped in many other ways as well. Oscar Hernandez helped me with several of the application appendices and made the Powerpoint slides that accompany the book. Caspar Lam designed the Web site for the book. David Reaves took pictures. My quilt, Blue Tweed, appears on the book's cover and on the Web site and slides. David took all the pictures that we used.

I would not have been in a position to write this book without the support of my father, who introduced me to the elegance of mathematics, Andy van Dam for my undergraduate experience at Brown, and Raj Reddy for my graduate experience at CMU. I cannot thank them enough.

Special thanks go to my family and friends, particularly my husband, Alan Cline, and my father, Bob Rich, for countless meals taken over by discussions of this material, proofreading more drafts than I can count, and patience while living with someone who is writing a book.

CREDITS

On the Cover:

A quilt, Blue Tweed (1996, 53" x 80", cotton, machine pieced and quilted), made by the author. Notice that your eye fills in the vertical lines, so they appear to run the length of the quilt, even though the colors in the middle of the quilt are all about the same. Quilt photography by David Reaves.


Photo Credits:

- Photograph of a fragment of the Antikythera Mechanism and two photographs of the reconstructed model of it, Figures P.1 and P.2: copyright of the Antikythera Mechanism Research Project.
- Photos of Prague orlog, Figure P.3, page 1056: Ing. Karel Mayr.
- Photo of abacus, Figure P.4, page 1057: David Reaves.
- Photo of Jacquard loom, Figure P.5, page 1058: Stan Sherer.
- Photo of Sony Aibo robot, Figure P.10, page 1062: Alan Cline.

Credits for Exercises:

- Alan Cline: Exercise 27.9.
- [Brachman and Levesque 2004]: Exercise 33.10.
- Jay Misra: Exercise 20.10.
- Luay Nakhleh: Exercises 8.17, 17.5, 17.12, 21.18, 21.21, 21.22.
- Cathy Stacy: Exercise 5.3.
- David Zuckerman: Exercises 22.5, 28.11, 28.16, 28.23(d), 28.26, 29.3, 30.1

Other Credits:

- IBM 7090 example, page 2: Brian Reid.
- IBM 360 JCL, page 3: Volker Bandke, http://www.bsp-gmbh.com/hercules/herc_jcl.html.
- The Java example, page 3: Mike Scott.
- Example 5.10, page 64: from [Misra 2004].
- The poem, "The Pumping Lemma for DFAs", page 198: Martin Cohn .
- The drawings generated by Lindenmayer systems, pages 547–549: Generated by Alan Cline in MATLAB®.
- Graph showing the growth rates of functions, page 598: Generated by Alan Cline in MATLAB®.
- Progression of closures given in Example A.11, pages 777–778: Alan Cline.
- Example A.19, page 784: Alan Cline.
- Analysis of iterative deepening, page 861: Alan Cline.
- The proofs in Section F.1, pages 869–875: Alan Cline.
- The network protocol diagrams and corresponding state machines, pages 919–924: Oscar Hernandez.
- A very long English sentence, page 984: <http://www.plainenglish.co.uk/longsentences.htm>.

xx Credits

- Drawing of girl with cat, page 995: Lynda Trader.
- Drawing of bear with rifle, page 997: Lynda Trader.
- Sound wave for the word “cacophony”, page 1000: Alan Cline.
- Simplified HMM for speech understanding, page 1002: Jim Barnett.
- Drawing of the Towers of Hanoi, page 1058: Alan Cline.
- The schematic diagram and the finite state diagram of a binary multiplier, page 1061: Oscar Hernandez.
- Diagram of the FSM robot controller, page 1063: Peter Stone.

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

- 20.3 Subset Relationships between D and SD 437
 - 20.4 The Classes D and SD Under Complement 438
 - 20.5 Enumerating a Language 440
 - 20.6 Summary 444
 - Exercises 445
- 21** Decidability and Undecidability Proofs 448
- 21.1 Reduction 449
 - 21.2 Using Reduction to Show that a Language is Not Decidable 452
 - 21.3 Are All Questions About Turing Machines Undecidable? 466
 - 21.4 Rice's Theorem • 468
 - 21.5 Undecidable Questions About Real Programs 472
 - 21.6 Showing That a Language is Not Semidecidable 474
 - 21.7 Summary of D, SD/D and \neg SD Languages that Include Turing Machine Descriptions 482
 - Exercises 483
- 22** Decidability of Languages That Do Not (Obviously) Ask Questions about Turing Machines • 487
- 22.1 Diophantine Equations and Hilbert's 10th Problem 488
 - 22.2 Post Correspondence Problem 489
 - 22.3 Tiling Problems 492
 - 22.4 Logical Theories 495
 - 22.5 Undecidable Problems about Context-Free Languages 499
 - Exercises 508
- 23** Unrestricted Grammars • 510
- 23.1 Definition and Examples 510
 - 23.2 Equivalence of Unrestricted Grammars and Turing Machines 516
 - 23.3 Grammars Compute Functions 518
 - 23.4 Undecidable Problems About Unrestricted Grammars 521
 - 23.5 The Word Problem for Semi-Thue Systems 522
 - Exercises 524
- 24** The Chomsky Hierarchy and Beyond • 526
- 24.1 The Context-Sensitive Languages 526
 - 24.2 The Chomsky Hierarchy 539
 - 24.3 Attribute, Feature, and Unification Grammars 540
 - 24.4 Lindenmayer Systems 544
 - Exercises 553

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