

HZ BOOKS
华章教育

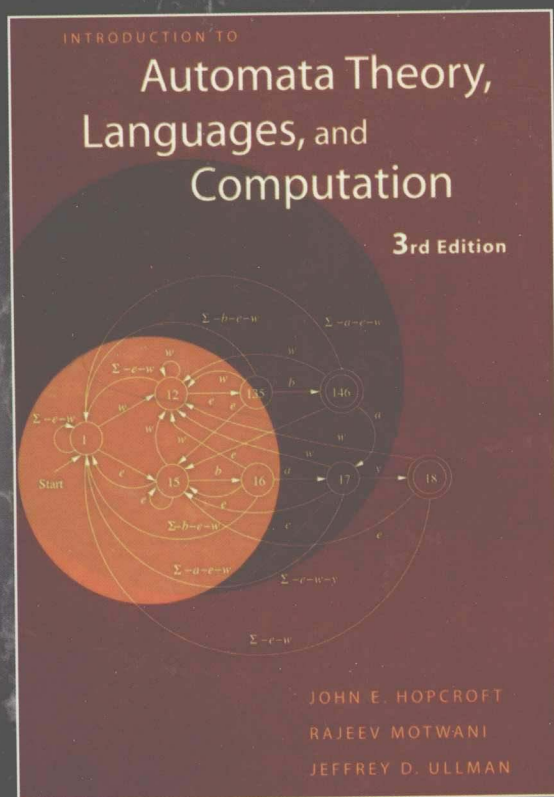
PEARSON
Addison
Wesley

计 算 机 科 学 丛 书

原书第3版

自动机理论、 语言和计算导论

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



Introduction to Automata Theory, Languages, and Computation

Third Edition



机械工业出版社
China Machine Press

计 算 机 科 学 丛 书

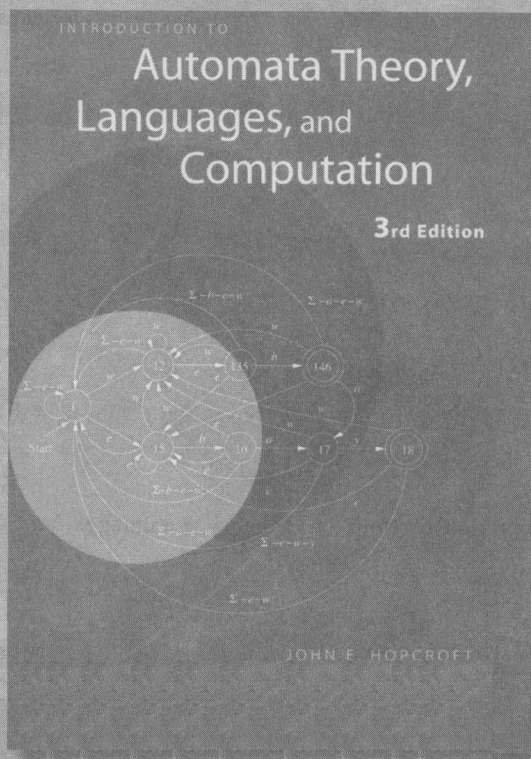
原书第3版

TP301/21=2

2008

自动机理论 语言和计算导论

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



**Introduction to Automata Theory, Languages,
and Computation**
Third Edition



机械工业出版社
China Machine Press

本书是关于形式语言、自动机理论和计算复杂性方面的经典之作，是国际上得到广泛认可的计算机理论和计算机工程专业的优秀教材。书中涵盖了有穷自动机、正则表达式与语言、正则语言的性质、上下文无关文法及上下文无关语言、下推自动机、上下文无关语言的性质、图灵机、不可判定性以及难解问题等内容。本书注重定义、定理的准确性和严格性，注重学生形式化和严格的数学推理能力的培养，同时在定义和证明中运用直观的方法说明抽象概念，借助许多图表帮助传达思想，并包含大量难度各异的示例和习题，便于读者加深对内容的理解。

本书适合作为计算机专业高年级本科生及研究生计算理论课程的教材和教学参考书。

Simplified Chinese 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.

All rights reserved.

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

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：01-2006-5653

图书在版编目（CIP）数据

自动机理论、语言和计算导论（原书第3版）/（美）霍普克罗夫特（Hopcroft, J. E.）等著；孙家骅等译. —北京：机械工业出版社，2008.7

（计算机科学丛书）

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

ISBN 978-7-111-24035-8

I. 自… II. ① 霍… ② 孙… III. ① 自动机理论—教材 ② 形式语言—教材 IV. TP301

中国版本图书馆CIP数据核字（2008）第060185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：刘立卿

北京京北制版厂印刷 · 新华书店北京发行所发行

2008年7月第1版第1次印刷

184mm×260mm · 23.75印张

标准书号：ISBN 978-7-111-24035-8

定价：49.00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中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，还揭示了学术的源变，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和发展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到积极的推动作用，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分社较早意识到“出版要为教育服务”。自1998年开始，华章分社就将工作重点放在了遴选、移译国外优秀教材上。经过多年的不懈努力，我们与Pearson, McGraw-Hill, Elsevier, MIT, John Wiley & Sons, Cengage等世界著名出版公司建立了良好的合作关系，从他们现有的数百种教材中甄选出Andrew S. Tanenbaum, Bjarne Stroustrup, Brian W. Kernighan, Dennis Ritchie, Jim Gray, Alfred V. Aho, John E. Hopcroft, Jeffrey D. Ullman, Abraham Silberschatz, William Stallings, Donald E. Knuth, John L. Hennessy, Larry L. Peterson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、研究及珍藏。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”已经出版了近两百个品种，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。其影印版“经典原版书库”作为姊妹篇也被越来越多实施双语教学的学校所采用。

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。随着计算机科学与技术专业学科建设的不断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将步入一个新的阶段，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。华章分社欢迎老师和读者对我们的工作提出建议或给予指正，我们的联系方式如下：

华章网站：www.hzbook.com

电子邮件：hzedu@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



华章教育

华章科技图书出版中心

译者序

理论计算机科学是推动计算机技术向前发展的强大动力。自动机、形式语言、可计算性和相关方面内容构成的计算理论，是理论计算机科学的基础内容之一。学习、研究这些内容，不仅为进一步学习、研究理论计算机科学所必需，而且对增强形式化能力和推理能力有重要作用，这些能力对从事计算机技术中的软件形式化等研究，是不可缺少的。

本书是由John E. Hopcroft、Rajeev Motwani和Jeffrey D. Ullman三位计算机学者合作编写的，是最著名的理论计算机科学著作之一，是世界各国广泛采用的计算机理论专业和计算机工程专业的优秀教材之一。它主要介绍形式语言、自动机、可计算性和相关方面内容。它特别注意定义、定理的准确性和严格性，这有利于培养学生形式化和严格的数学推理的能力。本书第1版于1979年发行。它的第3版有一个新的特色，那就是增加了一套由Gradiance公司开发的在线作业帮助系统。教师可以利用它给学生安排课后作业，或者给那些没有选课的学生提供一个特殊的综合课程（不是由老师申请开设的课程），使得他们可以利用这些课后作业作为练习和指导。然而遗憾的是，Gradiance在线作业系统只适用于购买原版教材的北美地区学生，中国学生还不能使用这个系统，因此，我们在翻译第3版的过程中，对涉及Gradiance系统的内容进行了删减。

引进国外的优秀计算机教材，无疑会对我国的计算机教育事业的发展起积极推动作用，也是与世界接轨、建立世界一流大学不可缺少的条件。我们把本书介绍给国内从事计算机教育事业的同行们，以供参考。

这个译本是根据原书第3版翻译的。参加本书翻译的有：孙家骅同志，负责各章译稿的详细修改和全书的统稿；刘明浩同志，负责第1~3章及前言、目录的翻译；孙自然同志，负责第4、5章的翻译；王啸吟同志，负责第6、7章的翻译；王华同志，负责第8、9章的翻译；侯姗姗同志，负责第10、11章及索引的翻译。为了与本书第2版中译本用的名词、术语保持统一，在翻译过程中我们参考了由刘田、姜晖和王捍贫同志完成的本书第2版中译本。

由于我们的能力有限，难免有不当之处，敬请读者不吝赐教。

译者

2008年3月



清华大学出版社

北京清华大学学研大厦A座

http://www.tup.com.cn

电子邮箱: tup@tup.com.cn

联系电话: (010) 68022264

地址: 北京市西城区百万庄大街24号

邮编: 100037

前 言

在本书1979年出版的前一版的前言中，Hopcroft和Ullman对于下面的事实感到惊诧不已：与1969年他们写第一本书时的情形相比，自动机这一专题已经有了突破性地发展。的确，1979年版的书中包含许多在以前的著作中找不到的主题，篇幅增加了大约一倍。如果读者有兴趣把本书与1979年版的那本书做个比较就会发现，正如20世纪70年代的汽车那样，本书是“外看大，内看小”。这表面看起来像是退步，但是我们有理由对此变化感到高兴。

首先，在1979年，自动机和语言理论还是一个比较活跃的研究领域。那本书的主要目的是鼓励擅长数学的学生为这个领域做出新的贡献。然而今天，直接针对自动机理论的研究几乎已经销声匿迹（相反，更多的是对其应用的研究），因此也就没有理由继续保持1979年那本书的简洁和高度数学化的风格。

其次，在过去的20年里，自动机和语言理论的角色已经发生了很大的变化。在1979年，自动机还主要是研究生水平的专题，那时我们假定的读者是高年级研究生，特别是使用这本书后几章的那些读者。然而到了今天，它已经成为本科生课程的主要内容。正是基于这个原因，本书在编排时必须假定对于读者只有很少的预备知识的要求，因此必须比前一版本的书提供更多的背景知识介绍和论证的细节。

环境的第三个变化是，计算机科学在过去的30年里已经发展到了几乎无法想像的程度。在1979年，我们觉得可能会经受得起下一轮技术考验的材料太少了，以致于排满课程计划都很难；然而今天，却有非常多的子科目在竞争本科生课程的有限空间。

第四，计算机科学如今已经发展成为更加职业化的科目，在许多学习它的学生中存在着严重的实用主义倾向。但我们仍然相信，自动机理论的方方面面是各种新兴学科的重要工具，并且我们相信，无论学生多么倾向于只去学习那些最能直接赚钱的技术，但在典型的自动机课程中，那些理论的和有利于拓宽思维的习题依然可以保持其价值。然而，为了保证这个科目在计算机专业学生的课程表中继续占有一席之地，我们相信有必要在强调数学理论的同时也强调应用。因此，我们把上一版中许多比较深奥的主题换成了今天如何使用这些思想的例子。虽然自动机和语言理论在众多编译器上的应用如今已经广为人知，以至于这些内容通常已经包含在介绍编译技术的课程中，但是还存在一些较新的用途，包括用来验证协议的模型验证算法以及采用与上下文无关文法的模式的文本描述语言等，本书对此进行了补充说明。

对于本书内容增减的最后一项解释是，如今我们能够充分利用Don Knuth和Les Lamport开发的TEX和LATEX排版系统的优点。特别是后一种系统，它鼓励“开放”式的排版，让书籍篇幅更大，但更容易阅读。感谢这两个人的努力为本书出版发行作出的贡献。

本书用法

本书适合于本科三年级以上学生的一季度或一学期的课程。在斯坦福大学，我们把这份讲

义用在CS154课程中，这是自动机和语言理论的课程。这是门一季度的课程，Rajeev和Jeffrey都曾经教过。由于授课时间有限，第11章不包括在授课范围以内，后面一些材料（比如较难的10.4节中的多项式时间归约等内容）也都省略了。本书的网站（参见下面）包括CS154课程的笔记和教学大纲等材料。

几年前我们发现，许多进入斯坦福大学的研究生都学过一门不包括难解性理论的自动机理论课程。由于斯坦福大学的全体教员相信，对于每一位计算机科学家来说，仅仅停留在知道“NP完全意味着需要花费很长时间”这一层面是远远不够的，充分了解这些思想是十分必要的，所以就有了另外一门课程CS154N，选这个课的学生可以只学习第8、9和10章。他们实际上学习CS154课程大约后三分之一部分的内容，由此来满足CS154N课程的要求。即使在今天，我们仍然发现，每个季度都有一些学生采取这种优化的课程选项。由于这样做几乎不需要额外的努力，所以我们推荐使用这种方法。

预备知识

为了最好地使用本书，读者应当在这以前学过一门关于离散数学（如图、树、逻辑、证明技巧等）的课程。我们还假设读者学过一些有关程序设计的课程，熟悉常见的数据结构、递归和主要系统组件（如编译器）的作用等。这些预备知识一般应当包含在大学一、二年级计算机科学课程计划中。

习题

本书中包含大量的习题，几乎每个章节都有。我们将较难的习题或其中的某一部分用单个叹号标出，最难的则用两个叹号标出。

某些习题或其中的某一部分标有星号。对于这部分习题，其解答方法可以通过本书的网页获得。这些解答可以公开获得，只限于进行自我检查。需要注意的一点是，在少数情况下，习题B要求修改或改编对另一个习题A的解答。如果A的某些部分有解答，那么你就应当预期B的对应部分也应当有解答。

Gradiance 在线作业^①

本书第3版有一个新的特色，即增加了一套由Gradiance公司开发的在线作业伴随系统。教师可以利用它给学生安排课后作业，或者给那些没有选课的学生提供一个特殊的综合课程（不是由老师申请开设的课程），使得他们可以利用这些课后作业作为练习和指导。Gradiance系统所提供的问题和普通问题的形式一样，但会对你提交的解决方案进行检查。如果你提交的答案不正确，系统将会提供一些针对性的建议或反馈意见，以帮助纠正你的解决方案。只要你的老师允许，就可以重复尝试，直到达到一个满意的分数为止。

所有在北美地区购买这本书第3版的读者，都将自动获得这项Gradiance在线作业服务的授权。更多的信息，请访问Addison-Wesley网站 www.aw.com/gradiance 或者发邮件给 computing@aw.com。

^① Gradiance系统对中国学生不适用，因此第3版翻译中删除了有关该系统的内容。——编者注

万维网上的支持

本书的主页是

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

其中包含带星号习题的解答、已知的勘误表、后备材料等。我们希望提供所教CS154课程的所有笔记材料，包括课后作业、习题解答和考试等。

致谢

Craig Silverstein关于“如何进行证明”的讲稿为本书第1章中的一些材料提供了借鉴。对本书第2版手稿（2000）的评论和勘误来自于：Zoe Abrams、George Candea、Haowen Chen、Byong-Gun Chun、Jeffrey Shallit、Bret Taylor、Jason Townsend和Erik Uzureau等。

同时，我们也收到了很多有关本书第2版内容勘误的电子邮件，这些读者的名字已经写进了那一版本在线勘误表的致谢栏中。然而，我们还是希望把那些提出了大量书中出现的重大错误的人员公布如下，他们是：Zeki Bayram、Sebastian Hick、Kang-Rae Lee、Christian Lemburg、Nezam Mahdavi-Amiri、Dave Majer、A.P.Marathe、Mark Meuleman、Mustafa Sait-Ametov、Alexey Sarytchev、Jukka Suomela、Rod Topor、Po-Lian Tsai、Tom Whaley、Aaron Windsor 和 Jacinth H.T.Wu。

非常感谢他们的帮助。当然，剩下的错误都是由我们负责。

J. E. H.

R. M.

J. D. U.

纽约州，艾萨克；加州，斯坦福

2006年2月

目 录

出版者的话	
译者序	
前言	
第1章 自动机：方法与体验	1
1.1 为什么研究自动机理论	1
1.1.1 有穷自动机简介	1
1.1.2 结构表示法	3
1.1.3 自动机与复杂性	3
1.2 形式化证明简介	3
1.2.1 演绎证明	4
1.2.2 求助于定义	6
1.2.3 其他定理形式	7
1.2.4 表面上不是“如果-则”命题的定理	9
1.3 其他的证明形式	9
1.3.1 证明集合等价性	9
1.3.2 逆否命题	10
1.3.3 反证法	12
1.3.4 反例	12
1.4 归纳证明	13
1.4.1 整数上的归纳法	13
1.4.2 更一般形式的整数归纳法	16
1.4.3 结构归纳法	16
1.4.4 互归纳法	18
1.5 自动机理论的中心概念	19
1.5.1 字母表	19
1.5.2 串	20
1.5.3 语言	21
1.5.4 问题	21
1.6 小结	23
1.7 参考文献	24
第2章 有穷自动机	25
2.1 有穷自动机的非形式化描述	25
2.1.1 基本规则	26
2.1.2 协议	26
2.1.3 允许自动机忽略动作	27
2.1.4 整个系统成为一个自动机	29
2.1.5 用乘积自动机验证协议	30
2.2 确定型有穷自动机	30
2.2.1 确定型有穷自动机的定义	31
2.2.2 DFA如何处理串	31
2.2.3 DFA的简化记号	32
2.2.4 把转移函数扩展到串	33
2.2.5 DFA的语言	35
2.2.6 习题	35
2.3 非确定型有穷自动机	37
2.3.1 非确定型有穷自动机的非形式化观点	37
2.3.2 非确定型有穷自动机的定义	38
2.3.3 扩展转移函数	39
2.3.4 NFA的语言	39
2.3.5 确定型有穷自动机与非确定型有穷自动机的等价性	40
2.3.6 子集构造的坏情形	43
2.3.7 习题	45
2.4 应用：文本搜索	46
2.4.1 在文本中查找串	46
2.4.2 文本搜索的非确定型有穷自动机	46
2.4.3 识别关键字集合的DFA	47
2.4.4 习题	49
2.5 带 ϵ 转移的有穷自动机	49
2.5.1 ϵ 转移的用途	49
2.5.2 ϵ -NFA的形式化定义	50
2.5.3 ϵ 闭包	51
2.5.4 ϵ -NFA的扩展转移和语言	52
2.5.5 消除 ϵ 转移	53
2.5.6 习题	54
2.6 小结	55
2.7 参考文献	55

第3章 正则表达式与正则语言	57	4.3.1 在各种表示之间转化	102
3.1 正则表达式	57	4.3.2 测试正则语言的空性	104
3.1.1 正则表达式运算符	57	4.3.3 测试正则语言的成员性	104
3.1.2 构造正则表达式	59	4.3.4 习题	105
3.1.3 正则表达式运算符的优先级	60	4.4 自动机的等价性和最小化	105
3.1.4 习题	61	4.4.1 测试状态的等价性	105
3.2 有穷自动机和正则表达式	61	4.4.2 测试正则语言的等价性	107
3.2.1 从DFA到正则表达式	62	4.4.3 DFA最小化	108
3.2.2 通过消除状态把DFA转化为正则表达式	65	4.4.4 为什么不能比最小DFA更小	110
3.2.3 把正则表达式转化为自动机	69	4.4.5 习题	111
3.2.4 习题	72	4.5 小结	112
3.3 正则表达式的应用	73	4.6 参考文献	112
3.3.1 UNIX中的正则表达式	73	第5章 上下文无关文法及上下文无关语言	115
3.3.2 词法分析	74	5.1 上下文无关文法	115
3.3.3 查找文本中的模式	76	5.1.1 一个非形式化的例子	115
3.3.4 习题	77	5.1.2 上下文无关文法的定义	116
3.4 正则表达式代数定律	77	5.1.3 使用文法来推导	118
3.4.1 结合律与交换律	78	5.1.4 最左推导和最右推导	119
3.4.2 单位元与零元	78	5.1.5 文法的语言	120
3.4.3 分配律	79	5.1.6 句型	121
3.4.4 幂等律	79	5.1.7 习题	122
3.4.5 与闭包有关的定律	79	5.2 语法分析树	124
3.4.6 发现正则表达式定律	80	5.2.1 构造语法分析树	124
3.4.7 检验正则表达式代数定律	81	5.2.2 语法分析树的产生	125
3.4.8 习题	82	5.2.3 推理、推导和语法分析树	125
3.5 小结	83	5.2.4 从推理到树	126
3.6 参考文献	84	5.2.5 从树到推导	127
第4章 正则语言的性质	85	5.2.6 从推导到递归推理	129
4.1 证明语言的非正则性	85	5.2.7 习题	131
4.1.1 正则语言的泵引理	85	5.3 上下文无关文法的应用	131
4.1.2 泵引理的应用	87	5.3.1 语法分析器	131
4.1.3 习题	88	5.3.2 语法分析器生成器 YACC	133
4.2 正则语言的封闭性	89	5.3.3 标记语言	134
4.2.1 正则语言在布尔运算下的封闭性	89	5.3.4 XML和文档类型定义	135
4.2.2 反转	93	5.3.5 习题	140
4.2.3 同态	94	5.4 文法和语言的歧义性	141
4.2.4 逆同态	96	5.4.1 歧义文法	141
4.2.5 习题	99	5.4.2 去除文法的歧义性	143
4.3 正则语言的判定性质	102	5.4.3 最左推导作为表达歧义性的一种	

5.4.1 方式	145	7.2.2 泵引理的陈述	191
5.4.4 固有的歧义性	146	7.2.3 CFL的泵引理的应用	193
5.4.5 习题	147	7.2.4 习题	195
5.5 小结	148	7.3 上下文无关语言的封闭性	196
5.6 参考文献	148	7.3.1 代入	196
第6章 下推自动机	151	7.3.2 代入定理的应用	198
6.1 下推自动机的定义	151	7.3.3 反转	198
6.1.1 非形式化的介绍	151	7.3.4 与正则语言的交	199
6.1.2 下推自动机的形式化定义	152	7.3.5 逆同态	202
6.1.3 PDA的图形表示	154	7.3.6 习题	204
6.1.4 PDA的瞬时描述	154	7.4 CFL的判定性质	205
6.1.5 习题	157	7.4.1 在CFG和PDA之间互相转化的 复杂性	205
6.2 PDA的语言	158	7.4.2 变换到乔姆斯基范式的运行时间	207
6.2.1 以终结状态方式接受	158	7.4.3 测试CFL的空性	207
6.2.2 以空栈方式接受	159	7.4.4 测试CFL的成员性	209
6.2.3 从空栈方式到终结状态方式	159	7.4.5 不可判定的CFL问题一览	211
6.2.4 从终结状态方式到空栈方式	162	7.4.6 习题	211
6.2.5 习题	163	7.5 小结	212
6.3 PDA和CFG的等价性	164	7.6 参考文献	212
6.3.1 从文法到PDA	164	第8章 图灵机导引	215
6.3.2 从PDA到文法	167	8.1 计算机不能解答的问题	215
6.3.3 习题	170	8.1.1 显示“hello, world”的程序	215
6.4 确定型PDA	171	8.1.2 假设中的“hello, world”检验程序	217
6.4.1 确定型PDA的定义	171	8.1.3 把问题归约到另一个问题	219
6.4.2 正则语言与确定型PDA	172	8.1.4 习题	221
6.4.3 DPDA与上下文无关语言	173	8.2 图灵机	221
6.4.4 DPDA与歧义文法	173	8.2.1 寻求判定所有数学问题	222
6.4.5 习题	174	8.2.2 图灵机的记号	222
6.5 小结	175	8.2.3 图灵机的瞬时描述	223
6.6 参考文献	175	8.2.4 图灵机转移图	225
第7章 上下文无关语言的性质	177	8.2.5 图灵机的语言	227
7.1 上下文无关文法的范式	177	8.2.6 图灵机与停机	228
7.1.1 去除无用的符号	177	8.2.7 习题	228
7.1.2 计算产生符号和可达符号	179	8.3 图灵机的程序设计技术	229
7.1.3 去除 ϵ 产生式	180	8.3.1 在状态中存储	230
7.1.4 去除单位产生式	182	8.3.2 多道	231
7.1.5 乔姆斯基范式	185	8.3.3 子程序	232
7.1.6 习题	189	8.3.4 习题	234
7.2 上下文无关语言的泵引理	191	8.4 基本图灵机的扩展	234
7.2.1 语法分析树的大小	191		

8.4.1 多带图灵机	234	9.4.4 习题	281
8.4.2 单带图灵机与多带图灵机的等价性	235	9.5 其他不可判定问题	281
8.4.3 运行时间与多带合一构造	236	9.5.1 与程序有关的问题	281
8.4.4 非确定型图灵机	237	9.5.2 CFG歧义性问题	281
8.4.5 习题	239	9.5.3 表语言的补	283
8.5 受限制的图灵机	240	9.5.4 习题	285
8.5.1 具有半无穷带的图灵机	240	9.6 小结	285
8.5.2 多堆栈机器	242	9.7 参考文献	286
8.5.3 计数器机器	244	第10章 难解问题	289
8.5.4 计数器机器的能力	244	10.1 P 类和 NP 类	289
8.5.5 习题	246	10.1.1 可在多项式时间内解答的问题	290
8.6 图灵机与计算机	247	10.1.2 例子: 克鲁斯卡尔算法	290
8.6.1 用计算机模拟图灵机	247	10.1.3 非确定型多项式时间	293
8.6.2 用图灵机模拟计算机	248	10.1.4 NP 例子: 货郎问题	293
8.6.3 比较计算机与图灵机的运行时间	251	10.1.5 多项式时间归约	294
8.7 小结	252	10.1.6 NP完全问题	295
8.8 参考文献	253	10.1.7 习题	296
第9章 不可判定性	255	10.2 NP完全问题	297
9.1 非递归可枚举语言	255	10.2.1 可满足性问题	297
9.1.1 枚举二进制串	256	10.2.2 表示SAT实例	299
9.1.2 图灵机编码	256	10.2.3 SAT问题的NP完全性	299
9.1.3 对角化语言	257	10.2.4 习题	304
9.1.4 证明 L_d 非递归可枚举	258	10.3 约束可满足性问题	304
9.1.5 习题	258	10.3.1 布尔表达式的范式	304
9.2 是递归可枚举但不可判定的问题	259	10.3.2 把表达式转化成CNF	305
9.2.1 递归语言	259	10.3.3 CSAT的NP完全性	308
9.2.2 递归语言和递归可枚举语言的补	260	10.3.4 3SAT的NP完全性	311
9.2.3 通用语言	262	10.3.5 习题	312
9.2.4 通用语言的不可判定性	263	10.4 其他的NP完全问题	312
9.2.5 习题	264	10.4.1 描述NP完全问题	313
9.3 与图灵机有关的不可判定问题	266	10.4.2 独立集问题	313
9.3.1 归约	266	10.4.3 顶点覆盖问题	316
9.3.2 接受空语言的图灵机	267	10.4.4 有向哈密顿回路问题	317
9.3.3 莱斯定理与递归可枚举语言的性质	269	10.4.5 无向哈密顿回路与TSP	322
9.3.4 与图灵机说明有关的问题	271	10.4.6 NP完全问题小结	323
9.3.5 习题	272	10.4.7 习题	323
9.4 波斯特对应问题	272	10.5 小结	326
9.4.1 波斯特对应问题的定义	273	10.6 参考文献	326
9.4.2 “修改过的”PCP	274	第11章 其他问题类	329
9.4.3 PCP不可判定性证明之完成	276	11.1 NP 中的语言的补	330

11.1.1 $\mathcal{NP}$ 补语言类	330
11.1.2 $\mathcal{NP}$ 完全问题与 $\mathcal{NP}$ 补	330
11.1.3 习题	331
11.2 在多项式空间内可解决的问题	331
11.2.1 多项式空间图灵机	332
11.2.2 $\mathcal{PS}$ 和 $\mathcal{NP}$ 与前面定义的类的关系	332
11.2.3 确定型多项式空间与非确定型	333
多项式空间	333
11.3 对 $\mathcal{PS}$ 完全的问题	335
11.3.1 $\mathcal{PS}$ 完全性	335
11.3.2 带量词的布尔公式	336
11.3.3 带量词的布尔公式的求值	337
11.3.4 QBF问题的 $\mathcal{PS}$ 完全性	338
11.3.5 习题	341
11.4 基于随机化的语言类	342
11.4.1 快速排序: 随机算法举例	342
11.4.2 随机化的图灵机模型	343

11.4.3 随机化图灵机的语言	344
11.4.4 $\mathcal{RP}$ 类	345
11.4.5 识别 $\mathcal{RP}$ 语言	347
11.4.6 $\mathcal{ZPP}$ 类	348
11.4.7 $\mathcal{RP}$ 与 $\mathcal{ZPP}$ 之间的关系	348
11.4.8 与 $\mathcal{P}$ 类和 $\mathcal{NP}$ 类的关系	349
11.5 素数性测试的复杂性	349
11.5.1 素数性测试的重要性	350
11.5.2 同余算术简介	351
11.5.3 同余算术计算的复杂性	352
11.5.4 随机多项式素数性测试	353
11.5.5 非确定型素数性测试	354
11.5.6 习题	356
11.6 小结	356
11.7 参考文献	357
索引	359

第1章 自动机：方法与体验

自动机理论研究抽象计算装置或“机器”。在20世纪30年代计算机出现之前，图灵研究过一种抽象机器，这种机器具备了今天计算机的所有能力，至少在计算能力上是这样的。图灵的目标是精确地描述一条界线，这条界线区分计算机能做什么和不能做什么；图灵的结论不仅适用于抽象的图灵机，也适用于今天的真实机器。

在20世纪40和50年代，许多研究者研究过更简单类型的机器，今天称为“有穷自动机”。起初建议用这些自动机来为人脑功能建立模型，后来发现这些自动机对于1.1节提到的各种其他目的也极为有用。在20世纪50年代后期，语言学家乔姆斯基(N. Chomsky)开始研究形式“文法”。尽管这些文法不是严格意义上的机器，但与抽象自动机有密切关系，而且目前是一些重要软件部件(包括部分编译器在内)的基础。

在1969年，库克(S. Cook)扩展了图灵对什么能被计算和什么不能被计算的研究。库克设法分离出了两类问题：一类是计算机能有效解决的；另一类是计算机理论上能解决，但实际上要花费太长时间，以致除了非常小的问题实例以外，计算机是毫无用处的。后一类问题称为“难解的”或“NP-难的”。计算机硬件一直遵循着计算速度呈指数增长的规律(摩尔定律)，但这也太不太可能显著地影响解决难解问题大实例的能力。

所有这些理论进展都直接影响了计算机科学家今天的工作。有些概念，比如有穷自动机和某些类型的形式文法，用于设计和构造重要类型的软件。另一些概念，比如图灵机，则帮助我们理解能期待软件做什么。特别是，难解问题的理论允许推断是否有可能“正面”处理一个问题，编写解决这个问题的程序(因为这个问题不属于难解的一类)，或者是否需要找到某种方法来迂回处理这个难解的问题：找近似算法、用启发式算法或者用某种其他方法来限制程序解决这个问题所花费的时间。

本章作为入门性的内容，首先从概述高级自动机理论的内容和用途开始。本章许多篇幅用来总结证明技术和发现证明的技巧。这些内容包括：演绎证明、命题变形、反证法、归纳证明以及其他重要概念。最后一节介绍一些贯穿自动机理论的概念：字母表、串以及语言。

1.1 为什么研究自动机理论

有几个理由解释了为什么自动机和复杂性的研究是计算机科学核心的重要部分。本节要向读者介绍主要的动机并概括本书讨论的主要主题。

1.1.1 有穷自动机简介

有穷自动机是许多重要类型的硬件和软件的有用模型。从第2章开始会看到如何使用这些概念的例子。目前只列出一些最重要的类型：

1. 数字电路的设计和性能检查软件。
2. 典型编译器的“词法分析器”，也就是说，把输入文本分解为诸如标识符、关键字和标点符号等逻辑单元的编译器部件。
3. 扫描大量文本（如收集到的网页）来发现单词、短语或其他模式的出现的软件。
4. 所有类型的只有有穷多个不同状态的系统（比如通信协议或安全交换信息的协议）的验证软件。

很快就会遇到各种类型自动机的精确定义，非形式化的介绍从概述有穷自动机是什么和做什么开始。可以认为许多系统或部件（如上面列举的那些）始终处在有穷多个“状态”之一。状态的目的是记住系统历史的有关部分。由于只有有穷多个状态，在一般情况下不可能记住整个历史，所以必须仔细地设计系统，以记住重要的历史部分而忘记不重要的。只有有穷多个状态的好处是用固定的资源就能实现系统。例如，可用硬件将其实现为电路，或者实现为简单形式的程序，这种程序只查看有限的的数据就能做出决定，或者用代码本身中的位置来做出决定。

例1.1 也许最简单的非平凡有穷自动机是两相开关。这个装置记住处在“开”状态或“关”状态，允许用户按按钮，这个按钮根据开关状态起不同作用。也就是说，如果开关处在关状态，按这个按钮就变成开状态；如果开关处在开状态，按这个按钮就变成关状态。

这个开关的有穷自动机模型如图1-1所示。对所有的有穷自动机都一样，状态用圆圈表示。在这个例子中，命名了状态on（开）和off（关）。状态之间的箭头用“输入”来标记，表示作用在系统上的外部影响。这里两个箭头都用Push（按）来标记，表示用户按这个按钮。这两个箭头的含义是：无论系统处在什么状态，当收到输入Push时，就进入另一个状态。

把一个状态指定为“初始状态”，即系统开始时所处的状态。在这个例子中，初始状态是off，习惯上用单词Start（开始）和一个指向这个状态的箭头来说明初始状态。

通常需要说明一个或多个状态是“终止”或“接受”状态。在经历一个输入序列后进入这些状态之一，说明这个输入序列在某种程度上是好的。例如，可以认为图1-1中的on状态是接受状态，因为在这个状态下，这个由开关控制的装置将会运行。习惯上用双圆圈来说明接受状态，但在图1-1中没有做这样的说明。

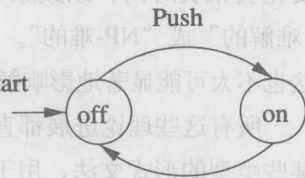


图1-1 为两相开关建立模型的有穷自动机

例1.2 有时候，一个状态记忆的内容比一个开关选择要复杂得多。图1-2显示另一个有穷自动机，它可能是词法分析器的一部分。这个自动机的任务是识别关键字then。因此这个自动机需要5个状态，每个状态表示在单词then中目前已经到达的不同位置。这些位置对应于这个单词的前缀，范围从空串（即一点也没有看见这个单词）到整个单词。

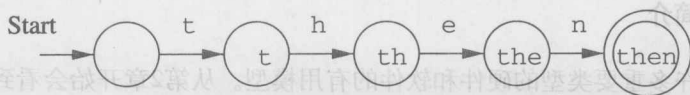


图1-2 为识别then建立模型的有穷自动机

在图1-2中，用目前已经看到的then的前缀来命名5个状态。输入对应字母。可以想像，词法分析器每次检查正在编译的程序的一个字母，要检查的下一个字母就是自动机的输入。初始状态对应空串。每个状态在then的下一个字母上转移到对应于下一个更长前缀的状态上。当输入拼写了单词then时，就进入了名为then的状态。由于这个自动机的任务是识别何时已经看到了then，所以可认为这个状态是惟一的接受状态。

1.1.2 结构表示法

有两种不像自动机但在自动机的研究和应用中起重要作用的重要记号。

1. 文法在设计某种软件时是有用的模型，这种软件处理带递归结构的数据。最著名的例子是“语法分析器”，即处理典型程序设计语言中递归嵌套特征的编译器部件，这些特征比如是表达式（如算术表达式、条件表达式等）。例如，像 $E \Rightarrow E + E$ 这样的文法规则提出，把两个表达式用加号连起来就形成一个表达式；对于真正的程序设计语言如何构造表达式来说，这条规则是典型的。第5章将介绍通常所说的上下文无关文法。
2. 正则表达式也表示数据的结构，特别是文本串。在第3章将会看到，正则表达式描述的串模式恰好与有穷自动机描述的一样。这些表达式的风格与文法的风格有显著不同。这里只举一个简单的例子。UNIX风格的正则表达式 `'[A-Z][a-z]*[ ][A-Z][A-Z]'` 表示首字母大写的单词后面跟着一个空格和两个大写字母。这个表达式表示文本中城市 and 州的模式，如Ithaca NY。这个表达式不能表示有多个单词的城市名，如Palo Alto CA，这可以用更复杂的表达式

`'[A-Z][a-z]*([ ][A-Z][a-z]*)*[ ][A-Z][A-Z]'`

来表示。在解释这种表达式时，只需要知道 `[A-Z]` 表示从大写字母A到大写字母Z的字符范围（即任意的大写字母），用 `[ ]` 表示单个的空格字符。而且，符号 `*` 表示“任意多个”前面的表达式。括号用来对表达式各部分进行分组，而不表示所描述的文本中的字符。

1.1.3 自动机与复杂性

对计算局限性的研究来说，自动机是必不可少的。正如本章前言中所说，有两个重要问题：

1. 计算机到底能做什么？这种研究称为“可判定性”，计算机能解决的问题称为“可判定的”。第9章讨论这个主题。
2. 计算机能有效地做什么？这种研究称为“难解性”，计算机用不超过输入规模的某个缓慢增长函数的时间所能解决的问题，称为“易解的”。通常，把所有的多项式函数当作“缓慢增长的”，而把比任何多项式都增长得快的函数看作增长得太快了。第10章讨论这个主题。

1.2 形式化证明简介

如果在20世纪90年代以前的任何时候，在高中学习平面几何，就很可能不得不做一些详细的“演绎证明”，即用详细的步骤和理由序列来证明命题的正确性。对于为什么在高中学习这个数学分支来说，几何有实用的一面（例如，要为房间买到大小合适的地毯，就要知道计算矩形面积的公式），但学习形式化证明方法也是同样重要的理由。

20世纪90年代，美国的流行趋势是把证明当做对命题的个人感觉。感觉要使用的命题是正

确的，这是好事，但在高中不再掌握重要的证明技术了。但证明还是每个计算机科学家需要理解的东西。某些计算机科学家采取极端的观点：程序正确性的形式化证明应当与编写程序本身同步进行。作者怀疑这样做是否有成效。另一方面，有人说在程序设计的原理中不包括证明。这个阵营经常提出这样的口号：“如果不能肯定程序是正确的，就运行这个程序来测试一下”。

作者的立场处在这两种极端之间。程序测试肯定是必要的。但是，测试只能走这么远，因为不可能在每个输入上测试程序。更重要的是，如果程序是复杂的，比如说棘手的递归或迭代，那么在进行循环或递归调用函数时，若不能理解正在干什么，就不太可能写出正确的程序代码。当测试说明代码不正确时，仍然需要修改代码。

要形成正确的迭代或递归，就要建立归纳假设；形式化或非形式化地推导出这个假设与迭代或递归是相容的，这样做是有帮助的。理解正确程序的工作过程，这个过程与用归纳法证明定理的过程实质上是一样的。因此，在提供某种类型软件的有用模型之外，学习形式化证明的方法学，这成了自动机理论课程的传统。也许比起计算机科学的其他核心课题来说，自动机理论本身有着更多自然的和有趣的证明，既有演绎类型（有根据的步骤的序列），也有归纳类型（带参数命题的递归证明，证明用到带“较小”参数值的命题本身）。

1.2.1 演绎证明

前面说过，演绎证明包含命题序列，其正确性导致从某个初始命题（称为前提或已知命题）得出“结论”命题。证明中每一步都必须根据某条公认的逻辑原理，利用已知的事实或演绎证明中前面的一些命题，或利用这两者的组合。

前提可能为真也可能为假，这通常依赖于参数值。前提常常包含几个独立命题，用逻辑AND（与）联结。在这些情况下，就说每个命题是一个前提或已知命题。

当从前提 H 到结论 C 时，所证明的定理是命题“如果 H ，则 C ”。说 C 是从 H 演绎出来的。形如“如果 H ，则 C ”的一个示范定理会解释这些要点。

定理1.3 如果 $x \geq 4$ ，则 $2^x \geq x^2$ 。

非形式化地说明定理1.3为真，这是不难的，但形式化证明要求用归纳法，将其留给例1.17。首先注意，前提 H 是“ $x \geq 4$ ”。这个前提有参数 x ，因此前提既不为真也不为假。实际上，前提的正确性依赖于参数 x 的值，例如， H 对于 $x = 6$ 为真，对于 $x = 2$ 为假。

同样，结论 C 是“ $2^x \geq x^2$ ”。这个命题也用参数 x ，对于特定的 x 值为真，对于其他值为假。例如， C 对于 $x = 3$ 为假，因为 $2^3 = 8$ ，不大于 $3^2 = 9$ 。另外， C 对于 $x = 4$ 为真，因为 $2^4 = 4^2 = 16$ 。对于 $x = 5$ ，命题也为真，因为 $2^5 = 32$ ，不小于 $5^2 = 25$ 。

读者也许看出这个直观论证，这个论证表明：只要 $x \geq 4$ ，结论 $2^x \geq x^2$ 就为真。我们已经看到，对 $x = 4$ ，结论为真。当 x 增长到大于4时，每次 x 增加1，左边的 2^x 就翻一倍。但右边的 x^2 按照比值 $\left(\frac{x+1}{x}\right)^2$ 来增长。如果 $x \geq 4$ ，则 $(x+1)/x$ 不可能大于1.25，因此 $\left(\frac{x+1}{x}\right)^2$ 不可能大于1.5625。由于 $1.5625 < 2$ ，每次 x 增长到大于4，左边的 2^x 就比右边的 x^2 增长得多。因此，只要从类似 $x = 4$ 这样的值开始（这个值已经满足不等式 $2^x \geq x^2$ ），就可以让 x 任意增长，仍然满足这个不等式。

6