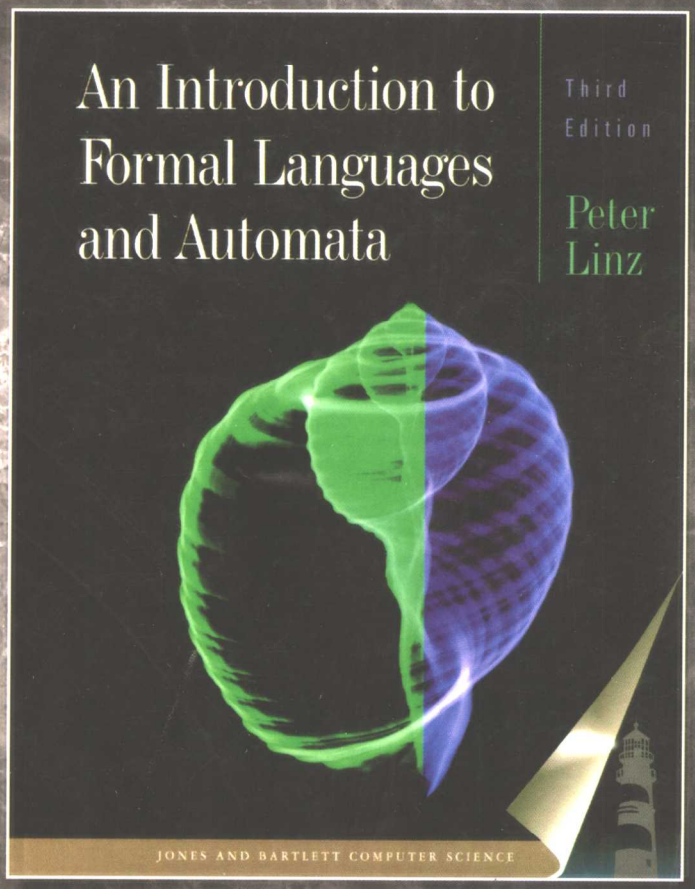
Jones and Bartlett

计 算 机 科 学 丛 书

原书第3版

形式语言与自动机导论

（美） Peter Linz 著　孙家骕 等译



An Introduction to Formal Languages and Automata, Third Edition

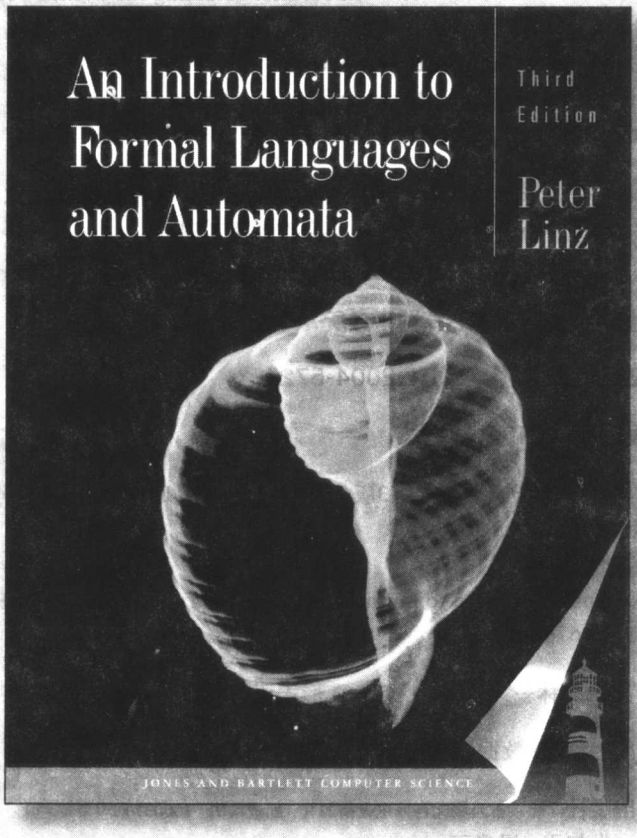
机械工业出版社
China Machine Press

计 算 机 科 学 丛 书

原书第3版

形式语言与自动机导论

(美) Peter Linz 著　孙家骕 等译



An Introduction to Formal Languages and Automata, Third Edition

机械工业出版社
China Machine Press

本书主要介绍形式语言、自动机、可计算性和相关内容。主要内容包括：计算理论导引、有穷自动机、正则语言与正则文法、上下文无关语言及文法、下推自动机、图灵机、形式语言和自动机的层次结构、计算复杂性等。每节后面都给出了习题，并包含部分习题的解答，方便教学。

本书是理论计算机科学方面的优秀教材之一，可作为高等院校计算机专业的教材，也可作为计算机系统研发人员的参考书。

Peter Linz: An Introduction to Formal Languages and Automata, Third Edition (ISBN 0-7637-1422-4).

本书版权登记号：图字：01-2004-5721

图书在版编目（CIP）数据

形式语言与自动机导论（原书第3版）/（美）林兹（Linz, P.）著；孙家骕等译. －北京：机械工业出版社，2005.9

（计算机科学丛书）

书名原文：An Introduction to Formal Languages and Automata, Third Edition

ISBN 7-111-16788-0

Ⅰ.形…　Ⅱ.①林…　②孙…　Ⅲ.①形式语言　②自动机理论　Ⅳ.TP301

中国版本图书馆CIP数据核字（2005）第070358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：朱起飞　冯春丽

北京京北制版厂印刷 · 新华书店北京发行所发行

2005年9月第1版第1次印刷

787mm × 1092mm 1/16 · 18.75印张

印数：0 001 - 4 000册

定价：36.00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，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了一个新的阶段。为此，华章公司将加大引进教材的力度，在“华章教育”的总规划之下出版三个系列的计算机教材：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”；同时，引进全美通行的教学辅导书“Schaum's Outlines”系列组成“全美经典学习指导系列”。为了保证这三套丛书的权威性，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。

这三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. Berkley，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

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			

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。形式语言、自动机、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。

本书是由美国加利福尼亚大学戴维斯分校的Peter Linz教授编写的高等学校教材，主要介绍形式语言、自动机、可计算性和相关内容。本书特别注意定义、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；本书强调讲述问题、定理时的直觉性，这有利于学生对问题的理解；本书在定理的证明中往往只给出框架，而略去细节，这有利于学生进一步思考，加强对问题的理解；本书在每节后面都给出了习题，对部分习题还给出了解答，这有利于学生通过做习题加深对基本原理的理解并增强应用能力。本书是理论计算机科学的优秀教材之一。

本书可以用作计算机理论专业和计算机工程专业的教材，也可以作为计算机系统研发人员的参考书。

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的推动作用，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。

本书第1版于1990年发行，这个译本是我们根据它的第3版翻译的。参加本书翻译的有：孙家骕同志负责各章译稿的详细修改和全书的统稿；郝丹同志负责第1章到第5章的翻译；罗景同志负责第6章到第9章的翻译；李炎同志负责第11章到第14章的翻译；孙宏涛同志负责第10章、习题解答及索引的翻译。

由于我们的能力有限，译文中难免有不当之处，敬请读者不吝赐教。

译 者

2005年6月

前　言

本书主要介绍形式语言、自动机和可计算性等相关内容。这些内容形成计算理论的主体。这方面的课程现在是计算机科学课程体系中必有的一部分，并且这门课程通常开设在计算机课程的早期。因此，这本书面向的读者主要包括计算机科学系或计算机工程系大学二年级和三年级的学生。

学习本书的预备知识包括了解某种高级程序设计语言（通常包括C、C++或Java），熟悉数据结构和算法的基础知识。另一个预备课程是离散数学，包括集合论、函数、关系、逻辑和数学推理的基础。这门课程也是标准的计算机科学导论课程的一部分。

计算理论的研究有几个目的，其中最为重要的目的包括：（1）使学生熟悉计算机科学的基础和原则，（2）为后续课程做准备，（3）增强学生进行规范和严格数学推理的能力。尽管我在这本书中采用的表达方式主要是为了达到前两个目的，但是我认为这本书也达到了第三个目的。为了表达得更加清晰，使得读者能够认识事物的本质，本书强调直觉上的动机，并通过例子阐述这些想法。只要可以选择，我就会选择那些容易掌握的论据，而不是那些简洁、优雅，可是在概念上却难以理解的论据。我在这本书中准确地声明定义和定理，给出证明的动机，不过省略了程序和冗长的细节。这样做是出于教育学的考虑。许多证明是归纳或反证的一般应用，它们对于不同的特殊问题而言也不同。这样的详细论证不仅是不必要的，而且会影响到整体的流畅性。因此，我们略过很多论证的细节，可能会有人出于完整性的考虑而质疑这种做法，我本人并不认为这是一个缺陷。数学技巧无法通过阅读别人的论证过程而培养出来，而应该是考虑问题的本质，发现能够证明论点的想法，然后通过精确的细节实现它。这后一种技巧是必学的。我认为这本书的论证框架提供了一个这样实践的正确起点。

从事计算机科学研究的学生有时把计算理论方面的课程视为非必需的，且认为它缺乏实践意义。为了说服他们，我们需要吸引他们的兴趣和精力，比如处理难解问题的坚韧性和创造性。为此，我的方法强调通过解决问题来学习。

使用这种解决问题的学习方法，我想让学生主要通过按问题分类的示例来学习知识。正如它们同定理和定义之间的关联性一样，这种例子通过隐藏其后的概念可以显现其动机。同时，这些例子还有其不平凡的一面，它们有助于学生发现解决方案。在这种方法中，课后习题更有助于学习过程。每个章节后面的习题阐明和解释了本章的内容，在不同的层次上调动学生解决问题的能力。其中一些习题相当简单，挑选了一些书中没有完成讨论的内容让学生继续进行思考。另一些习题非常难，即使对比较聪明的学生也是一种挑战。这两种习题的适当结合会成为非常有效的教学手段。学生不必解决所有的问题，但是应该根据课程和教师的要求完成部分习题。不同的学校开设的计算机科学课程不同，有的强调理论方面，有的几乎完全面向实际应用。倘若课程选择考虑到了学生的背景和兴趣，那么我相信这本书可以用于上述任何学校。同时，教师需要告诉学生他们期望学生能够做到的抽象程度。对于面向论证的习题更是如此。当我说“证明”或“表明”时，我认为学生可以构造出一个论证过程，然后产生一个清晰的结论。一个论证过程要形式化到什么程度由教师决定，而且这种指导应该

在课程的早期给出。

这本书的内容适合在一学期内完成。教师可以讲授这本书的大部分内容，不过，重点就由教师自己决定了。尽管这本书中证明不多，不过我在我的班级里通常都是简略地讲述这些证明。我通常只给出足够的讲解，使结论可信，然后让学生独自去阅读剩下的部分。总的来说，如果不想以后碰到困难，还是尽可能不要跳过某些章节。标有星号的章节可以略过，略过这部分内容不会影响你读后面的内容。尽管如此，大部分的内容还是必需的，而且不能够跳过。

这本书的第1版出版于1990年，第2版出版于1996年。对新版本的需求是令人高兴的，这表明我的这种通过语言而不是计算的教学方法是可行的。第2版相对于第1版的变化不是彻底的变化，而是在原有基础上的改进。它修改了第1版中晦涩和错误的内容。无论如何，第2版看起来已经相对稳定，需要进行修改的地方也很少。所以，第3版的大部分内容和第2版差不多。第3版主要的新特点是增加了一些带解答的习题。

最初，我并不愿意给出习题的解答，因为这样就会减少可以用来作为课后练习的习题数量。然而，几年来，我收到来自世界各地学生的求助请求，因此我决定在这一版中给出一部分习题的解答。我又增加了一些新的习题从而保证没有解答的习题的数量。我只挑选那些有重要指导意义的习题来给出解答。因此，我没有仅仅给出最终的答案，而是给出最终结果的推理依据。许多习题使用了同样的原理，我通常从中选择具有代表性的习题给出解答，希望学生能够举一反三。我相信挑选出来的这部分习题的解答可以帮助学生提高他们解决问题的能力，并且保留一部分比较好的习题给教师作为课后作业使用。在本书中，有解答和提示的习题都标有●。

此外，为响应读者的建议，我把一些比较难的习题做了标注。这样做不太容易，毕竟习题跨越的难度范围很大，而且同样一道习题对于某个学生而言简单，另一个学生可能就会认为很难。但是仍然存在一部分的习题，它们对于我的大多数学生而言都是有挑战性的。这部分的习题我用一个星号（★）标明。当然还存在一些特殊的习题，它们没有明确的答案。要想解决它们，可能需要思索，阅读一些附加内容，或者需要一些计算机编程。虽然它们不适合作为常规的作业来布置，但是它们可以作为进一步学习的切入点。这种习题我标了双星号（★★）。

在过去的10年里，我得到了很多评论家、教师和学生的有益建议。这样的人太多，我在这里无法一一列举他们的名字。我衷心感谢他们对我的帮助。他们的反馈对于我改进这本书而言是极其宝贵的。

Peter Linz

目　录

第1章 计算理论导引

计算机科学是一门实践学科。从事这一领域的人经常是那些偏爱解决理论中有用而切实的问题的人。同样地，计算机科学方向的学生主要对解决现实世界中的困难感兴趣。只有当理论问题有助于他们找到好的解决方案的时候，他们才对理论问题感兴趣。这种态度是正确的，因为没有应用，谁还会对计算机感兴趣呢。但是，既然认为计算机科学是面向实践的，那么人们不禁会问“为什么要研究理论？”

第一，理论提供有助于理解学科一般本质的概念和原则。计算机科学领域包含了广泛的特殊课题，从机器设计到编程。在现实世界中成功地应用计算机需要学习大量特殊的细节。这使得计算机科学成为一门涉及知识宽泛的学科。尽管计算机科学涉及多方面的知识，可是仍然存在一些通用的基本原则。为了研究这些基本原则，我们构造了计算机和计算的抽象模型。这些模型体现了硬件和软件的一些共同的重要特点，并且和我们在使用计算机工作的过程中碰到的许多特殊和复杂的构造是本质相关的。即使当这些模型太简单而不能立即应用到现实世界中时，我们仍然可以通过研究它们获得一些知识，为特殊的应用提供基础。这种方法当然不是仅仅应用在计算机科学中。模型构造是任何科学学科的本质之一，并且一门学科的有效性往往取决于其简单而强大的理论和规律。

第二，我们讨论的想法有即时和重要的应用，这点或许不是很明显。但是数字设计、程序设计语言和编译器都是最好的例子，当然还存在着很多其他例子。我们这里研究的概念像线一样贯穿整个计算机科学，从操作系统到模式识别。

第三，是我们想要说服读者。这个学科在智力上充满刺激和乐趣。它提出了许多有挑战性的、让人迷惑不解的题目，这些题目能让人废寝忘食。本质上，这就是解决问题。

在这本书中，我们关注那些能够表现所有计算机和它们的应用特点的模型。为了能够给计算机的硬件建模，我们引入了*自动机*（automaton）（复数：automata）的表示。自动机的构造包含了数字计算机所有必不可少的特点。它接受输入，产生输出，可能还有临时存储空间，并在把输入转化成输出的过程中做出决定。*形式语言*（formal language）是对程序设计语言的一般特点的抽象。形式语言包括符号集和把符号组成称为句子的实体的构成规则。一种形式的语言是所有符合构成规则的符号串的集合。尽管我们这里研究的某些形式语言，它们要比程序设计语言简单，但是它们有着相同的本质特点。通过形式语言，我们可以获得很多关于程序设计语言的知识。最后，我们通过*给算法*（algorithm）一个精确的定义来把机械计算的概念形式化，研究哪些问题适合用机械方式解决，哪些不适合。学习的过程中，我们将展示这些抽象表示之间的密切关系，研究我们从中可以获得的结论。

在第1章中，我们从一个宽广的视角去看这些基本想法，从而为后面的内容做准备。在1.1节中，我们复习必需的数学基本知识。因为直觉经常成为我们探究问题的指南，所以我们得出的结论应该以严格的推理为基础。这就要涉及很多数学工具，但是并不会涉及得过多。读者需要掌握集合论、函数和关系的术语及基本结论。书中还会经常用到树结构和图结构，不

2 过仅仅需要知道带标记图和有向图的定义。或许最重要的要求是能够理解证明和形成一个合理的数学推理。这包括熟悉演绎、归纳和反证法的基本证明技巧。我们假设读者已经具备了这些必需的背景知识。1.1节用来复习一些在后面的章节中使用的主要结论，并建立符号表示基础。

在1.2节中，我们首先关注语言、文法和自动机的核心概念。这些概念以各种不同的形态出现在书中。在1.3节中，我们给出了这些基本概念的一些简单应用，从而解释这些概念在计算机科学中的广泛应用。这两节的讨论是直观的，并不严格。在后面，我们会使这些概念更准确。但是暂时这样做的目的是为了更加清晰地描述我们要用到的概念。

1.1 数学预备知识和表示

1.1.1 集合

集合（set）是元素的组合，除去成员资格关系，无其他结构。我们用$x \in S$表示x是集合S的元素，$x \notin S$表示x不是集合S中的元素。集合在大括号内列出它的元素，例如，整数0，1，2的集合表示成

$$S = \{0, 1, 2\}$$

当集合含义清楚时，我们可以用省略号和其内的元素来表示集合。因此，$\{a, b, \cdots, z\}$代表所有的英文小写字母，而$\{2, 4, 6, \cdots\}$代表所有的正偶数。如果需要的话，我们会更多地使用直接表示，对于上面最后一个例子，我们可以写成

$$S = \{i : i > 0，\ i\text{是偶数}\} \tag{1-1}$$

通过这个式子我们可以知道“S是所有i的集合，其中，i大于零且为偶数”。当然这里暗示i是整数。

通常的集合运算包括并（union，表示成$\cup$）、交（intersection，表示成$\cap$）和差（difference，表示成$-$），它们的定义分别是

$$S_1 \cup S_2 = \{x : x \in S_1 \text{或} x \in S_2\}$$
$$S_1 \cap S_2 = \{x : x \in S_1 \text{且} x \in S_2\}$$
$$S_1 - S_2 = \{x : x \in S_1 \text{且} x \notin S_2\}$$

3 另一个基本运算是补（complementation）。集合S的补表示成$\overline{S}$，包含所有不在集合S中的元素。为了使这个概念有意义，我们首先需要知道全集（universal set）U包含哪些元素。如果给定U，那么

$$\overline{S} = \{x : x \in U, x \notin S\}$$

没有元素的集合称为空集（empty set或null set），表示成$\emptyset$。根据集合的定义，显然有

$$S \cup \emptyset = S - \emptyset = S$$
$$S \cap \emptyset = \emptyset$$
$$\overline{\emptyset} = U$$
$$\overline{\overline{S}} = S$$

下面这些有用的恒等式被称为**德摩根定律**（DeMorgan's laws），在有些情况下需要使用它们:

$$\overline{S_1 \cup S_2} = \overline{S}_1 \cap \overline{S}_2 \tag{1-2}$$

$$\overline{S_1 \cap S_2} = \overline{S}_1 \cup \overline{S}_2 \tag{1-3}$$

如果集合S_1的元素都是集合S的元素，那么集合S_1是集合S的**子集**（subset），记作

$$S_1 \subseteq S$$

如果$S_1 \subseteq S$，但是S包含一个不是S_1中的元素，那么S_1是S的**真子集**（proper subset），记作

$$S_1 \subset S$$

如果S_1和S_2没有共同的元素，即$S_1 \cap S_2 = \emptyset$，那么，这两个集合称为**不相交**（disjoint）。

如果集合包含有限个元素，那么这个集合是有限的；否则，就是无限的。一个有限集合的大小指的是它包含的元素个数，记作$|S|$。

一个给定的集合通常有很多子集。集合S的所有子集的集合称作集合S的**幂集**（powerset），记成2^S。注意2^S是集合的集合。

例1.1 如果集合$S = \{a, b, c\}$，那么它的幂集就是

$$2^S = \{\emptyset, \{a\}, \{b\}, \{c\}, \{a, b\}, \{a, c\}, \{b, c\}, \{a, b, c\}\}$$

这里$|S| = 3$，$|2^S| = 8$。这是一个例子的一般结果。如果S是有限的，那么

$$|2^S| = 2^{|S|}$$ □ 4

在我们的很多例子中，一个集合的元素是其他集合元素的有序排列。这个集合称为其他集合的**笛卡儿积**（Cartesian product）。两个集合的笛卡儿积是有序对的集合，表示成

$$S = S_1 \times S_2 = \{(x, y) : x \in S_1, y \in S_2\}$$

例1.2 设$S_1 = \{2, 4\}$，$S_2 = \{2, 3, 5, 6\}$。那么

$$S_1 \times S_2 = \{(2,2),(2,3),(2,5),(2,6),(4,2),(4,3),(4,5),(4,6)\}$$

注意上述书写的元素对的顺序。有序对（4, 2）属于集合$S_1 \times S_2$，但是（2, 4）就不属于。这种表示显然可以扩展到多个（大于两个）集合的笛卡儿积。通用的定义为

$$S_1 \times S_2 \times \cdots \times S_n = \{(x_1, x_2, \cdots, x_n) : x_i \in S_i\}$$ □

1.1.2 函数和关系

函数（function）是建立一个集合的元素和另一个集合的唯一的一个元素对应关系的规则。如果f表示一个函数，那么第一个集合称为函数f的**定义域**（domain），第二个集合称为它的**值域**（range）。我们用

$$f : S_1 \to S_2$$

表示函数f的定义域是集合S_1的子集，值域是集合S_2的子集。如果函数f的定义域就是集合S_1本

身，我们就说函数f是集合S_1上的全函数（total function）；否则就是部分函数（partial function）。

在许多应用中，函数的定义域和值域是正整数集合。而且，我们经常只对函数的自变量变大时所表现出来的函数行为感兴趣。在这种情况下，理解变化率就足够了，我们只需要使用普通的数量级符号表示。假设函数$f(n)$和$g(n)$的定义域是正整数的子集。如果存在一个正整数c满足：对于所有的n，

$$f(n) \leqslant cg(n)$$

那么我们就说g是f的最大阶（order at most），记作

5

$$f(n) = O(g(n))$$

如果

$$|f(n)| \geqslant c|g(n)|$$

那么g是f的最小阶（order at least），记作

$$f(n) = \Omega(g(n))$$

最后，如果存在常数c_1和c_2，并且

$$c_1|g(n)| \leqslant |f(n)| \leqslant c_2|g(n)|$$

则称f和g等价（same order of magnitude），表示成

$$f(n) = \Theta(g(n))$$

在数量级表示中，我们忽略乘法的常数，以及随着n增大可以忽略的低阶项。

例1.3 设

$$f(n) = 2n^2 + 3n$$
$$g(n) = n^3$$
$$h(n) = 10n^2 + 100$$

那么

$$f(n) = O(g(n))$$
$$g(n) = \Omega(h(n))$$
$$f(n) = \Theta(h(n))$$

在数量级表示中，=号不应该被解释成相等，数量级表达式不能像普通的表达式那样对待。诸如这样的写法

$$O(n) + O(n) = 2O(n)$$

就不是很明智，很可能会导致错误的结论。然而如果使用正确的话，数量级的命题是有效的，这一点我们可以在后面关于算法分析的章节中看到。 □

一些函数被表示成有序对的集合

$$\{(x_1, y_1), (x_2, y_2), \cdots\}$$

这里，x_i是函数定义域的元素，y_i是函数值域中对应的元素。使用这样的集合来定义函数，任

意x_i至多出现在有序对的第一个元素处一次。如果不满足这一点，那么这个集合称为**关系**（relation）。关系比函数的概念更宽泛：在函数中，定义域中的每一个元素在值域中都恰好有一个关联元素；在关系中，可能有多个值域中的元素与之相关联。 6

另一种特殊的关系是**等价**（equivalence），它就是通常的恒等关系。表示有序对(x,y)是等价关系可以写成

$$x \equiv y$$

如果一个标有≡的关系满足下面三个规则，那么这个关系是等价关系：
自反性规则

$$\text{对于所有的}x\text{，都有}x \equiv x$$

对称性规则

$$\text{如果}x \equiv y\text{，那么}y \equiv x$$

传递性规则

$$\text{如果}x \equiv y\text{并且}y \equiv z\text{，那么}x \equiv z$$

例1.4 定义在非负整数集合上的关系

$$x \equiv y$$

成立，当且仅当

$$x \bmod 3 = y \bmod 3$$

根据上面的定义可以得到$2 \equiv 5$，$12 \equiv 0$和$0 \equiv 36$。很明显，这是一个等价关系，因为它满足自反性、对称性和传递性。 □

1.1.3 图和树

一个**图**（graph）包括两个有限集合，**顶点**（vertex）集合$V = \{v_1, v_2, \cdots, v_n\}$和**边**（edge）集合$E = \{e_1, e_2, \cdots, e_m\}$。每条边由顶点集$V$中的一对顶点构成，例如

$$e_i = (v_j, v_k)$$

是一条从顶点v_j到顶点v_k的边。我们认为边e_i对于v_j而言是输出边，对于v_k而言是输入边。这种构造实际上是**有向图**（digraph），因为我们把每条边都关联了一个方向（从v_j到v_k）。图也可以被标记，这个标记可以是名字或和图的部分相关联的其他信息。顶点和边都可以被标记。 7

图的这种结构通常以图表的方式来表现。在图表中，顶点表示成圆圈，边表示成连接顶点的带箭头的直线。图1-1中描绘的图包含顶点集合$\{v_1, v_2, v_3\}$和边集合$\{(v_1, v_3), (v_3, v_1), (v_3, v_2), (v_3, v_3)\}$。

一个边的序列$(v_i, v_j), (v_j, v_k), \cdots, (v_m, v_n)$称为从$v_i$到$v_n$的**通道**（walk）。通道的长度指的是从起点到终点经过的边的数目。没有重复边的通道称为**路径**（path）。没有重复顶点的路径成为**简单**（simple）路径。一个从顶点v_i出发又回到该顶点的无重复边的通道称为以v_i为**始点**（base）的**回路**（cycle）。如果回路中除了作为始点的顶点以外没有重复的顶点，那么这个回路是简单的。在图1-1中，$(v_1, v_3), (v_3, v_2)$是从v_1到v_2的简单路径。边序列$(v_1, v_3), (v_3, v_3), (v_3, v_1)$是回路，

但不是简单回路。如果图中的边被标记，那么我们就可以谈论通道的那个标记了。这个标记就是遍历路径时经过的边的顺序标号。最后，一条从某个顶点到它自身的边称作环（loop）。在图1-1中，顶点v_3有个环。

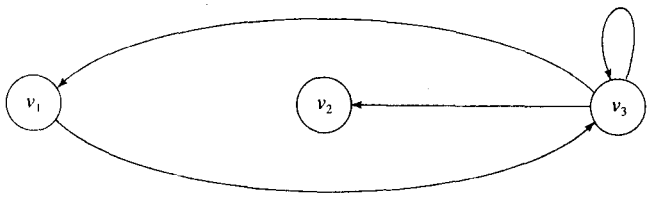


图 1-1

在几种情况下，我们会用到一个寻找两个给定顶点间所有简单路径（或以某一个顶点为始点的所有简单回路）的算法。如果不考虑效率的话，我们可以使用下面这个方法。从某个给定的顶点（假设是v_i）出发，列出所有的输出边 (v_i, v_k), (v_i, v_l), …这样，我们就获得了所有起点为v_i，长度为1的路径。对于所有这样到达的顶点v_k, v_l, …，只要输出边的另一个顶点不是已经构造出的路径中的某个顶点，我们就可以列出所有的输出边。之后，我们就获得了所有始点为v_i，长度为2的简单路径。继续上述过程直到不可能增加新的顶点为止。因为顶点的数目是有限的，所以我们最终会列出所有起点是v_i的简单路径。从这些路径中，我们选出那些以给定的另一顶点为终点的路径。

树是一种特殊的图。树是一个没有回路的有向图。在这个图中，有一个唯一的顶点，称为*根结点*（root），从根结点到任何一个其他顶点只有唯一的一条路径。这个定义告诉我们，根结点是没有输入边的，而且树中存在一些没有输出边的顶点。这些没有输出边的顶点就是树的*叶结点*（leave）。如果从顶点v_i到顶点v_j存在边，那么v_i就是v_j的*父结点*（parent），v_j是v_i的*子结点*（child）。每个结点的*层数*（level）指的是从根结点到该结点的路径的边数。树的*高度*（height）指的是所有结点的最大层数。这些术语在图1-2中以图例的方式进行了解释。

8

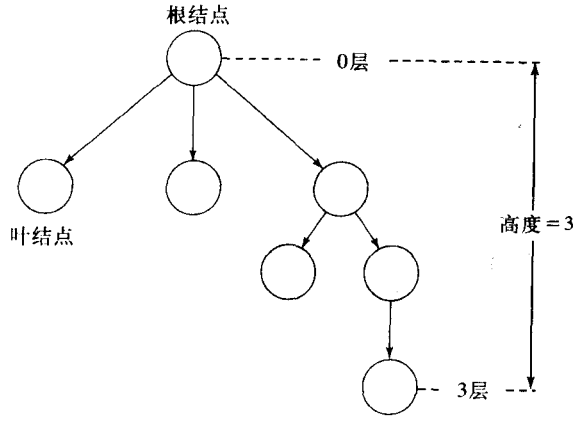


图 1-2

有时，我们想要把每层的结点和一定的顺序相联系，这种情况的树，称为*顺序树*（ordered tree）。

关于图和树的更多知识可以在离散数学书中获得。