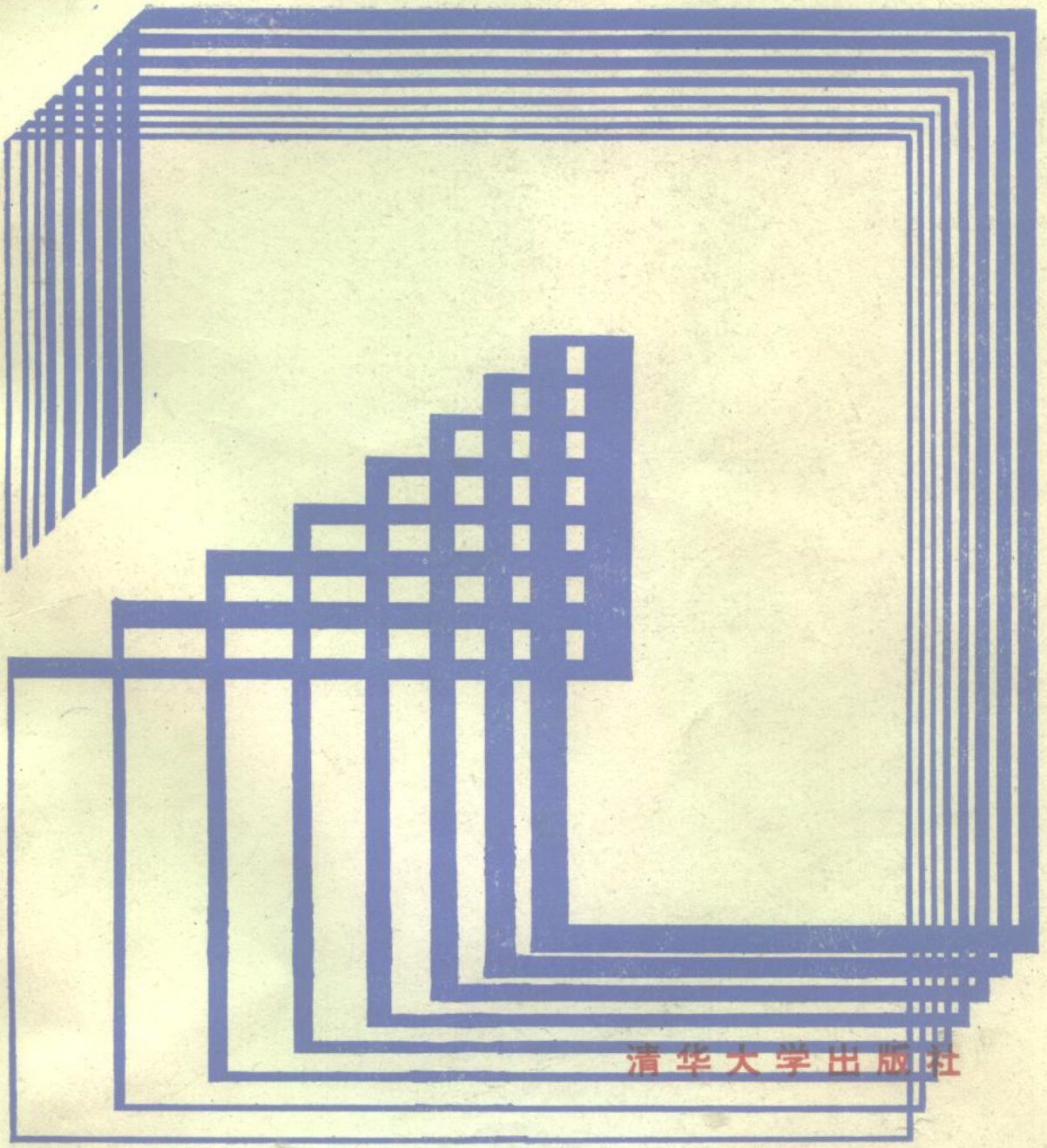
[美] M. D. 戴维斯
E. J. 威尤克 著

张立昂 陈进元 耿素云 译

可计算性 复杂性 语言

理论计算机科学基础



清华大学出版社

可计算性 复杂性 语言 理论计算机科学基础

社

内 容 简 介

本书的作者 M. 戴维斯是世界著名的数理逻辑学家，这部著作是他近年来的新著。书中全面、系统地阐述了理论计算机科学的主要分支，包括可计算性理论、文法和自动机理论、命题演算和量词理论，计算复杂性和不可解性等。本书结构严谨，论述精辟透彻，内容丰富且有不少新颖、独到之处，很适宜作为高等院校计算机专业本科生或研究生的教科书；对从事计算机科学工作，尤其是理论计算机科学研究者，具有较高的参考价值。

Computability, Complexity, and Languages
Fundamentals of Theoretical Computer Science
Martin D. Davis Elaine J. Weyuker
ACADEMIC PRESS, INC. 1983

可计算性 复杂性 语言
理论计算机科学基础
张立昂等 译

☆

清华大学出版社出版
北京 清华园
中国科学院印刷厂印刷
新华书店总店科技发行所发行

☆

开本：787×1092 1/16 印张：19.75 字数：468 千字
1989年11月第1版 1989年11月第1次印刷
印数：0001—4000
ISBN 7-302-00426-9/TP·141
定价：3.95元

译 者 序

本书是清华大学计算机科学技术系卢开澄教授推荐并鼓励我们把它翻译出来，介绍给国内读者的。本书的主要作者 M.戴维斯现任美国纽约州立大学计算机科学系教授，是世界著名的数理逻辑学家，他曾和 H. Putnan、J. Robinso、Yu. Matijasevic 等人一起解决了著名的希尔伯特第十问题——刁番图方程问题。

本书是 M. 戴维斯近年来的一部新著，全面而系统地阐述了理论计算机科学的主要分支，包括可计算性理论，文法和自动机理论，命题演算和量词理论，计算复杂性和不可解性等。本书结构严谨，论述透彻，内容丰富且有不少新颖、独到之处，很适宜作为计算机专业高年级本科生或研究生的教科书，对于从事计算机科学工作尤其是理论计算机科学研究者，亦有较好的参考价值。

1985 年夏当我们正着手翻译此书时，恰逢 M. 戴维斯来华讲学。他对此书能以中文出版甚感欣幸，并亲自来函指出原书中一些错误之处，我们在翻译时据此一一予以纠正。本书的第一章至第六章由耿素云翻译，第七章至第十二章由陈进元翻译，第十三章至第十七章由张立昂翻译。北京大学计算机科学技术系的吴允曾教授生前对本书的初稿曾作了统校工作。

在本书出版之际，我们要向卢开澄教授表示诚挚的谢意，感谢他向我们及国内广大读者推荐了这样一部极有价值的书。我们还深深地感谢和怀念吴允曾教授，他对本书的翻译曾作了不少具体的指导，并花费了很多心血。遗憾的是，他未能见到本书的出版就因心脏病发作而匆匆地离开了我们。这部书，将寄托着我们对他的永远的敬意和哀思！

译者谨致

1988 年 5 月于燕园

前　言

理论计算机科学是研究计算模型的数学学科。就这一点而言，它始于本世纪30年代逻辑学家 Church、Gödel（哥德尔）、Kleene（克林）、Post（波斯特）以及 Turing（图灵）的工作，那时还没有出现现代计算机。这些早期的工作对计算科学的实践和理论的发展有着深远的影响。不仅图灵机模型被证明是理论的基础，而且这些先驱者的工作预示了计算实践的许多方面。这些方面现在看来是很平常的，但它们的思想渊源并不为使用者知道。其中包括在原则上存在通用数字计算机、把程序作为形式语言中的指令表的概念、解释程序的可能性、软件和硬件之间的对偶性以及用产生式为基础的形式结构表示语言。计算机科学的注意中心在于已经取得真正激动人心的技术进展方面，而它的基础方面的重要研究工作也在继续进行。我们写这本书的目的是向大学生和研究生，介绍关于理论计算机科学的各个方面。这个介绍是足够全面的，通过介绍将使这方面的专业文献（论文及研究文章）易于为读者所理解。

我们所研究的是一个非常年青的、还正在自我发现的领域。关于这个学科的哪一部分具有持久的意义，计算机科学家们的观点是不一致的。在这种形势下，我们冒着风险，力图选择那些已十分成熟且相信在今后的研究中将起重要作用的题目。

我们假定读者中许多人没有什么数学证明的经验，但差不多都具有丰富的程序设计经验。因而，第一章介绍在数学中的证明方法以及对术语和记号的通常解释。然后，我们通过研究一个极简单的抽象的程序设计语言中的可计算性理论进入读者熟悉的内容。通过系统地运用宏展开技术来证实这个语言的惊人能力，并以一个通用程序为结束。该程序详细地写在一页纸上。然后，通过一系列模拟，我们得到可计算性的各种不同阐述的等价性，其中包括图灵的阐述。关于这些模拟，我们认为在这个阶段读者不必去填写那些概略叙述的模糊不清的论证的全部细节，而恰当地安排材料使模拟能够简单、清楚、完全地表示出来则是作者的责任。

本书的素材曾在纽约大学、Brooklyn 学院、秘鲁国际大学数学院、贝克莱加州大学、圣巴巴拉加州大学以及 Worcester 工学院给大学生和研究生使用过。

虽然我们把关于形式语言的材料作为本书的第二部分，但是可以紧接着第一章的后面阅读关于"正则语言"和"上下文无关语言"两章。我们相信，用关于上下文无关语言的乔姆斯基-许曾贝格表示定理来研究它们和下推自动机的关系更清楚一些。第三部分讲解逻辑中我们认为对计算机科学来说是重要的方面，它也可以紧接着第一章后面读。第四部分的三章中，每一章介绍一个重要的计算复杂性理论，包括 NP 完全性理论在内。第五章部分介绍了高等递归论，其中某些课题在多项式时间可计算性理论中有类似的具有丰富成果的课题。最后介绍了关于递归可枚举图灵级的优先结构。在努力了解算法的基本性质时必须考虑这些结构所呈现的反常现象，尽管没有理由相信所生成的具体算法将在实践中被证明是有用的。

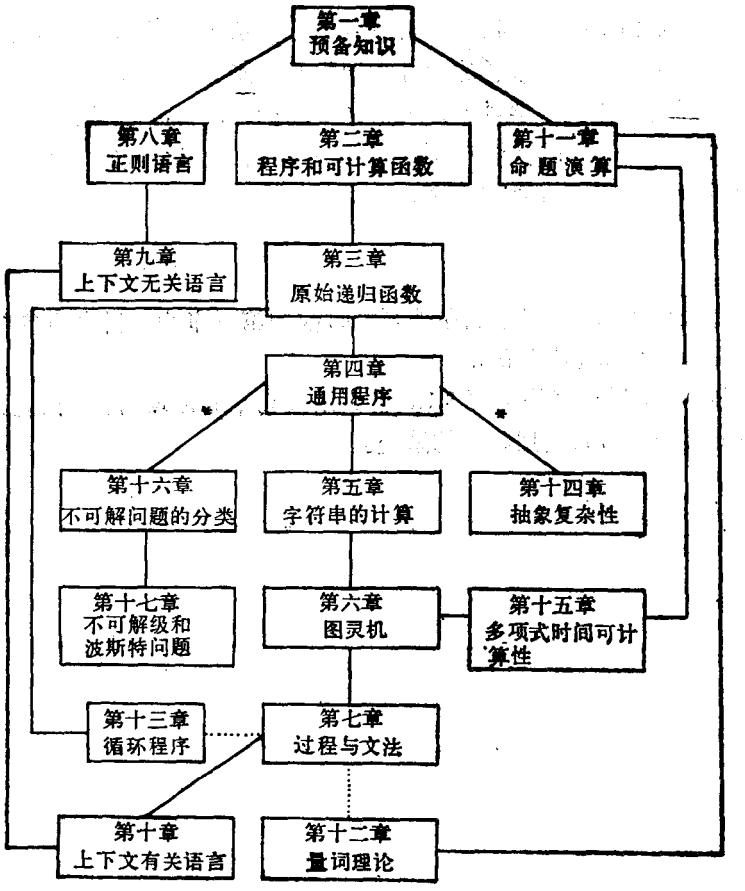
由于许多章是相互独立的，所以可以采取各种各样的方式使用本书。本书的内容比研究生全学年的计算理论课程的材料丰富。我们在纽约大学成功地在一学期的低年级的课程——计算理论引论——中使用了第一章到第六章以及第八章的所有不带星号的内容。关于有限自动机和形式语言的课程可以第一、八和九章为基础。一学期或者半学期的计算机科学系的学生的逻辑课程可以从第一部分和第三部分中挑选一些内容作为基础。由后面的关系图显而易见它的安排和课程是可能的。但是，我们的愿望是这本书将帮助读者看到理论计算机科学并不是一份许多离散课题拼凑而成的清单，而是一门统一的学科，它吸取了强有力的数学方法和计算技术丰富的实践经验，是对人类知识的一个生机勃勃的新领域进行了有价值的研究。

致读者

很多读者想从第二章开始，而在需要的时候把第一章的材料作为参考。喜欢跳着看书的读者将发现关系图是有用的。

引用定理 8.1 是指该章的定理 8.1。当引用其它章的定理时，将指明是哪一章。在引用带标号的公式和练习时使用同样的系统。

关 系 图



两章之间的实线表示高编号章的不带星号的节与低编号章的不带星号的节有关。实线旁的星号表示低编号章的带星号的节的知识也是需要的。虚线说明高编号章的带星号的节需要低编号章的不带星号的节的知识。

目　录

第三部分 逻辑

第四部分 复杂性

第五部分 不可解性

第一章 预备知识

1. 集合与 n 元组

我们将经常处理由某些确定类型的对象构成的**集合**。将实物的汇集视为一个集合，相当于将整个汇集视为一个对象。我们将使用**类**这个词作为集合的同义词。特别地，我们用 N 表示**自然数** 0,1,2,…的集合。在本书中除特殊声明外，数这个词总是指自然数。

我们用

$$a \in S$$

表示 a **属于** S，或等价地说 a **是** S **的元素**，而

$$a \notin S$$

表示 a 不属于 S。谈及空集是有用的，记作 $\varnothing$，它没有任何元素。设 R 和 S 是集合，等式 $R=S$ 表示 R 和 S 作为集合是等同的，即它们恰好有同样的元素。我们写出 $R \subseteq S$ 并称 R **是** S **的子集**，这是指 R 的每个元素也是 S 的元素。于是，$R=S$ 当且仅当 $R \subseteq S$ 和 $S \subseteq R$。还注意到，对任意集合 R，有 $\varnothing \subseteq R$ 和 $R \subseteq R$。我们用 $R \subset S$ 表示 $R \subseteq S$ 但 $R \neq S$。在这种情况下，称 R 是 S **的真子集**。如果 R 和 S 是集合，我们把 R 和 S 的**并集**，记作 $R \cup S$，它是 R 的元素或 S 的元素或 R 和 S 的元素的全体的汇集。把 R 和 S 的**交集**，记作 $R \cap S$，它是同时属于 R 和 S 的所有对象的集合。$R-S$ 是 R 与 S 的**差集**，它是属于 R、但不属于 S 的所有对象的集合。S 可能包含不在 R 中的对象。因此，$R-S=R-(R \cap S)$。通常我们将所考虑的全部集合看作某个确定的集合 D（有时称 D 为**域**或**全集**）的子集。在这种情况下，将 $D-S$ 写作 $\bar{S}$，称 $\bar{S}$ 是 S **的补集**。最常见的是把 $N-S$ 写作 $\bar{S}$。德·摩尔根恒等式

$$\overline{R \cup S} = \bar{R} \cap \bar{S}$$

$$\overline{R \cap S} = \bar{R} \cup \bar{S}$$

是很有用的；这两个等式容易验证，不熟悉它们的读者应该验证它们。记

$$\{a_1, a_2, \cdots, a_n\}$$

为由 n 个对象 $a_1, a_2, \cdots, a_n$ 组成的集合。能写成这种形式的集合以及空集称为**有穷的**。不是有穷的集合，例如 N，称作是**无穷的**集合。必须仔细地注意 a 和 $\{a\}$ 不是相同的。特别是，$a \in S$ 为真当且仅当 $\{a\} \subseteq S$。因为两集合相等，当且仅当它们有相同的元素，因此 $\{a,b,c\}=\{a,c,b\}=\{b,a,c\}$，也就是说，我们可以选取任意的顺序写出集合中的元素。当顺序是重要的时候，把它说成 n **元组**或**表**。n 元组用圆括号而不是用花括号来表示：

$$(a_1, a_2, \cdots, a_n)$$

当然，组成 n 元组的元素不必是不同的，例如，$(4,1,4,2)$ 是一个 4 元组。2 元组又叫做**有序对**，3 元组叫做**有序 3 元组**。对于只含一个对象的集合，我们不用区分对象 a 和 1 元组

(a)。n 元组的关键性质是

$$(a_1,a_2,\cdots,a_n)=(b_1,b_2,\cdots,b_n)$$

当且仅当

$$a_1=b_1,\ a_2=b_2,\cdots,\ a_n=b_n$$

如果 $S_1,S_2,\cdots,S_n$ 是给定的集合，则用 $S_1\times S_2\times\cdots\times S_n$ 表示由所有 n 元组$(a_1,a_2,\cdots,a_n)$ 组成的集合，其中 $a_1\in S_1,a_2\in S_2,\cdots,a_n\in S_n$。有时称 $S_1\times S_2\times\cdots\times S_n$ 为 $S_1,S_2,\cdots,S_n$ 的**笛卡儿积**。在 $S_1=S_2=\cdots=S_n=S$ 的情况下，将笛卡儿积

$$S_1\times S_2\times\cdots\times S_n$$

记成 S^n。

2. 函数

实际上，在纯数学和应用数学的每个分支中函数都是很重要的。我们可以将函数简单地定义为一个集合 f，它的所有元素都是有序对而且有特殊性质：

$$(a,b)\in f \text{ 且 } (a,c)\in f \text{ 蕴涵 } b=c。$$

然而，直观上将列出的有序对看作一张表格的两行更有用些。对于函数 f，人们用 $f(a)=b$ 表示 $(a,b)\in f$；按照函数的定义，对于每一个 a 至多存在一个这样的 b。对于某个 b，$(a,b)\in f$ 的所有 a 组成的集合称作 f 的**定义域**。对于 f 的定义域中的 a，所有 $f(a)$ 组成的集合称作 f 的**值域**。

作为一个例子，令 f 是有序对 (n,n^2) 组成的集合，其中 $n\in N$，则对于每个 $n\in N$，$f(n)=n^2$。f 的定义域是 N。f 的值域是完全平方数的集合。

函数 f 常由从 a 得到 $f(a)$ 的过程的算法来确定。确定函数的方法在计算机科学中是特别重要的。然而，就象在第四章将看到的那样，完全可能有一种算法，它规定了一个函数，但是不能够指出哪些元素属于它的定义域。这就使得所谓**部分函数**的概念在可计算性理论中起核心作用。集合 S 上的部分函数是以 S 的子集为其定义域的函数。$g(n)=\sqrt{n}$ 是集合 N 上的部分函数，在这里 g 的定义域是完全平方数集合。如果 f 是集合 S 上的部分函数，而且 $a\in S$，那么我们用 $f(a)\downarrow$ 表示 a 在 f 的定义域中，并且说 $f(a)$ 是有定义的；如果 a 不在 f 的定义域中，则记作 $f(a)\uparrow$，并且说 $f(a)$ 没有定义。如果 S 上的部分函数的定义域是 S，则称它是**全函数**。最后，我们应该指出空集 $\varnothing$ 本身是一个函数，当把它看作某个集合 S 上的部分函数时，它是**处处无定义**的。

对于笛卡儿积 $S_1\times S_2\times\cdots\times S_n$ 上的部分函数 f，我们写作 $f(a_1,a_2,\cdots,a_n)$ 而不写成 $f((a_1,a_2,\cdots,a_n))$。集合 S^n 上的部分函数 f 称作 S 上的 n 元部分函数，或 S 上的 n 个变量的函数。对于 n 元部分函数，我们常用 $f(x_1,\cdots,x_n)$ 代替 f，以明显地表示 f 是 n 元的。

3. 字母表和字符串

字母表是称作**符号**的对象的有穷非空集合 A。A 中的符号的 n 元组称作 A 上的**字**或**字符串**。我们把字$(a_1,a_2,\cdots,a_n)$简单地写成 $a_1a_2\cdots a_n$。如果 $u=a_1a_2\cdots a_n$，则称 u 的长度为 n 且记作 $|u|=n$。把唯一的零字写作 0，它的长度是 0（数字零与零字使用相同的

符号的理由将在第五章加以说明)。把字母表 A 上的所有字的集合记作 A^*。A^* 的任何子集称为 **A 上的语言**或**字母表为 A 的语言**。我们不区分符号 $a \in A$ 和该符号组成的长度为 1 的字。如果 $u,v \in A^*$，则用 $\widehat{uv}$ 表示将字符串 v 放在字符串 u 之后所得到的字。例如，如果 $A=\{a,b,c\}$，$u=bab$，$v=caa$，则

$$\widehat{uv}=babcaa,\quad \widehat{vu}=caabab$$

当不会引起混淆时，我们把 uv 代替 $\widehat{uv}$。对所有的 u，有

$$u0=0u=u$$

而对所有的 u,v,w，

$$u(vw)=(uv)w$$

另外，如果 $uv=uw$，或 $vu=wu$，则 $v=w$。

如果 u 是一个字符串，且 $n \in N$，$n>0$，则记

$$u^{[n]}=\underbrace{uu\cdots u}_{n}$$

我们也记作 $u^{[0]}=0$。为了避免同数值指数混淆，我们使用方括号。

如果 $u \in A^*$，用 u^R 表示倒过来写的 u；即，如果 $u=a_1a_2\cdots a_n$，$a_1,a_2,\cdots,a_n \in A$，则 $u^R=a_n\cdots a_2a_1$。显然，$0^R=0$，而对于 $u,v \in A^*$，$(uv)^R=v^Ru^R$。

4. 谓词

集合 S 上的**谓词**或**布尔值函数**，指的是 S 上的全函数 P，对于每个 $a \in S$

$$P(a)=\text{TRUE} \text{ 或 } P(a)=\text{FALSE}$$

其中 TRUE 和 FALSE 是一对称作真值的不同对象。当 $P(a)=$ TRUE 时，我们常说 $P(a)$ 为**真**；而当 $P(a)=$ FALSE 时，则说 $P(a)$ 为**假**。用特定的数表示真值对于我们来说是有益的，因而我们令

$$\text{TRUE}=1 \text{ 和 } \text{FALSE}=0$$

于是，谓词是在 N 中取值的一种特殊类型的函数。集合 S 上的谓词常用表达式来规定，当表达式中的变量用指定的 S 中的固定元素代替时，表达式就变成真或假的命题。于是，表达式

$$x<5$$

规定 N 上的一个谓词，即

$$P(x)=\begin{cases}1 & \text{当 } x=0,1,2,3,4\\ 0 & \text{否则}\end{cases}$$

表 4.1

p	$\sim p$
0	1
1	0

p	q	$p\&q$	$p \vee q$
1	1	1	1
0	1	0	1
1	0	0	1
0	0	0	0

表 4.1 定义了有关真值的三种基本运算：

如果 P 和 Q 是集合 S 上的谓词，那么 $\sim P$，$P\&Q$，$P\vee Q$ 也是 S 上的谓词。恰好当 P 为假时，$\sim P$ 为真。当 P 与 Q 同时为真时，$P\&Q$ 为真；否则 $P\&Q$ 为假。当 P 为真或 Q 为真或 P 与 Q 同时为真时，$P\vee Q$ 为真；否则 $P\vee Q$ 为假。给定集合 S 上的谓词 P，存在 S 的相应子集 R，R 是使 $P(a)=1$ 的所有元素 $a\in S$ 组成的集合。记

$$R=\{a\in S|P(a)\}$$

相反地，给定集合 S 的子集 R，表达式

$$x\in R$$

定义 S 上的一个谓词，它规定如下

$$P(x)=\begin{cases}1 & \text{若 } x\in R\\ 0 & \text{若 } x\notin R\end{cases}$$

当然，在这种情况下，

$$R=\{x\in S|P(x)\}$$

谓词 P 称为集合 R 的特征函数。集合和谓词之间的关系如此密切，以至于人们能把有关其中之一的论述迅速地转化为有关另一个的论述。于是，我们有

$$\{x\in S|P(x)\&Q(x)\}=\{x\in S|P(x)\}\cap\{x\in S|Q(x)\}$$

$$\{x\in S|P(x)\vee Q(x)\}=\{x\in S|P(x)\}\cup\{x\in S|Q(x)\}$$

$$\{x\in S|\sim P(x)\}=S-\{x\in S|P(x)\}$$

为了表示两个含变量的表达式定义相同的谓词，在它们之间置符号“$\Leftrightarrow$”。例如，

$$x<5\Leftrightarrow x=0\vee x=1\vee x=2\vee x=3\vee x=4$$

第一节中的德·摩尔根恒等式可以用集合 S 上的谓词表示如下：

$$P(x)\&Q(x)\Leftrightarrow\sim(\sim P(x)\vee\sim Q(x))$$

$$P(x)\vee Q(x)\Leftrightarrow\sim(\sim P(x)\&\sim Q(x))$$

5. 量词

在本节，我们将只涉及不同 m 值的 N^m 上的谓词（或 N 上的 m 元谓词）。当意思清楚时，今后将省去短词“在 N 上”。这样，设 $P(t,x_1,\cdots,x_n)$ 是 $(n+1)$ 元谓词。考虑用

$$Q(y,x_1,\cdots,x_n)\Leftrightarrow P(0,x_1,\cdots,x_n)\vee P(1,x_1,\cdots,x_n)\vee\cdots\vee P(y,x_1,\cdots,x_n)$$

定义的谓词 $Q(y,x_1,\cdots,x_n)$。于是，恰好当存在一个值 $t\leqslant y$ 使得 $P(t,x_1,\cdots,x_n)$ 为真时，谓词 $Q(y,x_1,\cdots,x_n)$ 为真。记这个谓词 Q 为

$$(\exists t)_{<y}P(t,x_1,\cdots,x_n)$$

表达式“$(\exists t)_{<y}$”称为**有界存在量词**。类似地，将谓词

$$P(0,x_1,\cdots,x_n)\&P(1,x_1,\cdots,x_n)\&\cdots\&P(y,x_1,\cdots,x_n)$$

记作

$$(\forall t)_{<y}P(t,x_1,\cdots,x_n)$$

恰好当所有 $t\leqslant y$，$P(t,x_1,\cdots,x_n)$ 为真时，这个谓词为真。表达式“$(\forall t)_{<y}$”称为**有界全称量词**。我们还用 $(\exists t)_{<y}P(t,x_1,\cdots,x_n)$ 表示恰好在至少有一个 $t<y$ 使

$$P(t,x_1,\cdots,x_n)$$

为真时为真的谓词，用 $(\forall t)_{<y}P(t,x_1,\cdots,x_n)$ 表示恰好在 $P(t,x_1,\cdots,x_n)$对所有 $t<y$ 都为真时为真的谓词。

我们用

$$Q(x_1,\cdots,x_n)\Longleftrightarrow(\exists t)P(t,x_1,\cdots,x_n)$$

表示这样一个谓词，如果存在 $t\in N$ 使 $P(t,x_1,\cdots,x_n)$ 为真，则这个谓词为真。类似地，如果 $P(t,x_1,\cdots,x_n)$ 对所有 $t\in N$ 都为真，则 $(\forall t)P(t,x_1,\cdots,x_n)$ 为真。

下述广义德．摩尔根恒等式有时是有用的：

$$\sim(\exists t)_{<y}P(t,x_1,\cdots,x_n)\Longleftrightarrow(\forall t)_{<y}\sim P(t,x_1,\cdots,x_n)$$

$$\sim(\exists t)P(t,x_1,\cdots,x_n)\Longleftrightarrow(\forall t)\sim P(t,x_1,\cdots,x_n)$$

读者可容易地验证下面的例子：

$$(\exists y)(x+y=4)\Longleftrightarrow x\leqslant 4$$

$$(\exists y)(x+y=4)\Longleftrightarrow(\exists y)_{<4}(x+y=4)$$

$$(\forall y)(xy=0)\Longleftrightarrow x=0$$

$$(\exists y)_{<z}(x+y=4)\Longleftrightarrow(x+z\geqslant 4\ \&\ x\leqslant 4)$$

6. 反证法

在本书中将许多我们所做的断言叫做“定理”（“推论”或“引理”）并“证明”它们是正确的。证明为什么是必须的？下面的例子将有助于回答这个问题：

回忆一下如果一个数恰好有两个不同的因子：本身和1，则把这个数叫做素数。2，17和41是素数，而0，1，4和15不是素数。考虑下述断言：

对所有的 $n\in N$，n^2-n+41 是素数。

事实上这个断言是错误的。例如，当 $n=41$ 时，表达式变成

$$41^2-41+41=41^2$$

它当然不是素数。然而，该断言对所有 $n\leqslant 40$ 均成立（有机会使用计算机的读者很容易验证它！）。这个例子说明，从有限个实例（即使是大量的）推断出关于无穷集合（例如 N）的全体元素的结论可能是很危险的。证明就是要去克服这个障碍。

证明以某些初始命题开始并且使用逻辑推理去推断附加的命题（在第十一章和十二章中，将看到如何使逻辑推理的概念精确化；但实际上，我们将以非形式的直观方式使用逻辑推理）。如果认为证明开始时的初始命题是正确的，则可以认为推断出来的命题也是正确的。但是常常不能用这种简单的思想方式来实现证明。在这一节和下一节中，我们将讨论更复杂的证明模型。

在反证法中，人们首先假设要证明的断言是假的，然后可以自由地用我们要证明的结论的否定作为证明过程中的一个初始命题。在用反证法时，我们寻找在证明中推导出的一对彼此矛盾的命题。由于它们两者不可能同时为真，所以可得出初始假设是错误的结论，因而我们所希望的结论就是正确的。

这里给出两个反证法的例子。在本书中将有许多反证法的例子。第一个例子是相当著名的。我们知道每个数或是偶数（即，对某个 $n\in N$，等于 $2n$）或者是奇数（即，对某个 $n\in N$，等于 $2n+1$）。因而，如果 m 是偶数，$m=2n$，则 $m^2=4n^2=2\times 2n^2$ 是偶数，

而如果 m 是奇数，$m = 2n + 1$，则 $m^2 = 4n^2 + 4n + 1 = 2(2n^2 + 2n) + 1$ 是奇数。我们要证明对于 $m, n \in N$，方程

$$2 = (m/n)^2 \tag{6.1}$$

没有解(即，$\sqrt{2}$ 不是"有理"数)。我们假设方程有解并且导出矛盾。假设方程 (6.1) 有解，则它必有 m 和 n 不都是偶数的解。这是正确的，因为如果 m 和 n 都是偶数，我们可以重复地从分子和分母中"约去" 2，直到至少其中之一为奇数。另一方面，可以证明方程 (6.1) 的每一个解，m 和 n 一定都是偶数。这个矛盾表明我们的假设是错误的，即方程 (6.1) 没有解。

剩下的问题是证明在方程(6.1)的每个解中，m 和 n 都是偶数。我们可将(6.1)式写成

$$m^2 = 2n^2$$

这说明 m^2 是偶数。如上所述，这意味着 m 是偶数，譬如说 $m = 2k$。这样，$m^2 = 4k^2 = 2n^2$，或 $n^2 = 2k^2$，于是，n^2 是偶数，从而 n 是偶数。 ■

注意符号"■"表示"这个证明到此完成了"。

我们的第二个例子涉及到第 3 节讨论的字符串。

定理 6.1 设 $x \in \{a,b\}^*$ 使得 $xa = ax$，则对于某个 $n \in N$，$x = a^{[n]}$。

证明：假设 $xa = ax$，而 x 含字母 b。我们可将 x 写成 $x = a^{[n]}bu$，这里明显地表示出在 x 中首次(即，最左边的)出现的 b。则

$$a^{[n]}bua = aa^{[n]}bu = a^{[n+1]}bu$$

于是，

$$bua = abu$$

但这是不可能的，因为同一个字符串的第一个符号不可能既是 a 又是 b，这个矛盾证明该定理成立。 ■

练习

1. 证明对于 $p, q \in N$，方程 $(p/q)^2 = 3$ 没有解。
2. 证明如果 $x \in \{a,b\}^*$ 且 $abx = xab$，则对于某个 $n \in N$，$x = (ab)^{[n]}$。

7. 数学归纳法

数学归纳法给证明形如 $(\forall n)P(n)$ 的命题提供了一个重要的方法，其中 P 是 N 上的谓词。人们通过证明一对辅助命题来进行证明，即证明

$$P(0)$$

和

$$(\forall n)(\text{If } P(n) \text{ then } P(n+1)) \tag{7.1}$$

我们一旦成功地证明了这一对辅助命题，就可以认为 $(\forall n)P(n)$ 也被证明了。其理由如下：

从第二个辅助命题可以推断下述无穷多个命题：

If $P(0)$ then $P(1)$,

If $P(1)$ then $P(2)$,

$$\text{If } P(2) \text{ then } P(3), \cdots。$$

因为我们已证明了 $P(0)$，就可以推断出 $P(1)$。现在已证明了 $P(1)$，可以得到 $P(2)$，等等。于是，对所有的 n，$P(n)$ 为真。因而，$(\forall n)P(n)$ 为真。

这为什么有用呢？因为有时证明(7.1)式比用其它方法证明 $(\forall n)P(n)$ 容易得多。在证明第二个辅助命题时，人们可以有代表性地考虑 n 的某个固定的，但又是任意的值 k，并且证明如果假设 $P(k)$ 就可以证明 $P(k+1)$。$P(k)$ 称之为**归纳假设**。这个方法论使我们能够把 $P(k)$ 用做证明过程中的一个初始命题。

有关用数学归纳法的证明有一些似非而是的东西。表面上看这似乎是一个循环推理的例子，它好像是对任意的 k 假设 $P(k)$，而 $P(k)$ 恰好是要它证明的东西。当然，数学归纳法实际上没有假设 $(\forall n)P(n)$。对于某个具体的 k 假设 $P(k)$ 为的是证明 $P(k+1)$ 成立。

在使用归纳法(我们常省去"数学"这个词)时，另一个似非而是的地方是，为了证明命题先"加强"这些命题，有时这样做更容易些。我们可以概略地说明如下：我们希望证明 $(\forall n)P(n)$，而决定去证明较强的断言 $(\forall n)(P(n) \& Q(n))$（当然它蕴涵原来的命题）。用归纳法证明较强的命题要求证明

$$P(0) \& Q(0)$$

和

$$(\forall n)[\text{If } P(n) \& Q(n) \text{ then } P(n+1) \& Q(n+1)]$$

在证明第二个辅助命题时，可以取 $P(k) \& Q(k)$ 作为归纳假设。这样，虽然加强要证明的命题使我们要证明更多的内容，但是也给予我们更强的归纳假设，因而给我们提供了更多的用来工作的条件。这种为了使归纳法证明变得更容易些而故意加强要证明的内容的方法叫做**加重归纳法**。

现在用归纳法证明例题。下述定理对做第六章的一道练习题是有用的。

定理 7.1 对所有的 $n \in N$，有 $\sum_{i=0}^{n} (2i+1) = (n+1)^2$。

证明。对于 $n = 0$，该定理只是说 $1 = 1^2$，成立。

假设对于 $n = k$，结论成立。即，我们的归纳假设是

$$\sum_{i=0}^{k} (2i+1) = (k+1)^2。$$

那么

$$\begin{aligned}\sum_{i=0}^{k+1} (2i+1) &= \sum_{i=0}^{k} (2i+1) + 2(k+1) + 1 \\ &= (k+1)^2 + 2(k+1) + 1 \\ &= (k+2)^2\end{aligned}$$

这正是我们所希望的对于 $n = k+1$ 的结果。 ■

数学归纳法另一种有用的形式称作**串值归纳法**，或者有时叫做**完全归纳法**。在用串值归纳法时，我们证明一个辅助命题

$$(\forall n)[\text{If } (\forall m)_{m<n}P(m) \text{ then } P(n)] \tag{7.2}$$

然后得出 $(\forall n)P(n)$ 成立的结论。串值归纳法可能引起混乱的地方是没有初始命题 $P(0)$，而实际上并不缺少 $P(0)$。(7.2) 式 $n=0$ 的情况是

$$\text{If } (\forall m)_{m<0}P(m) \text{ then } P(0)$$

而"归纳假设" $(\forall m)_{m<0}P(m)$ 恒假，因为不存在 $m \in N$ 使得 $m<0$。所以对于 $n=0$，(7.2) 式提供给我们的正是 $P(0)$。实际上，有时可能给出(7.2)式对包括 $n=0$ 在内的所有 n 的一个证明。但是，$n=0$ 的情况常常必须个别处理。

为了领会串值归纳法工作的原理，根据我们所说的关于 $n=0$ 的情况，(7.2)式导出下面无穷多个命题：

$$\begin{aligned}
&P(0),\\
&\text{If } P(0) \text{ then } P(1),\\
&\text{If } P(0)\,\&\,P(1) \text{ then } P(2),\\
&\text{If } P(0)\,\&\,P(1)\,\&\,(2) \text{ then } P(3),\\
&\quad\vdots
\end{aligned}$$

下面是用串值归纳法证明定理的一个例子。

定理 7.2 不存在字符串 $x \in \{a, b\}^*$，使得 $ax = xb$。

证明：考虑下述谓词：如果 $x \in \{a, b\}^*$ 且 $|x| = n$，则 $ax \neq xb$。我们要证明对于所有的 $n \in N$ 都是成立的。于是，我们假设对于所有的 $m < k$（k 是给定的）它都成立，并证明对于 k 它成立，用反证法证明。假设 $|x| = k$ 且 $ax = xb$，该等式蕴涵 a 是 x 中的第一个符号，b 是 x 中的最后一个符号，所以，可以记成 $x = aub$。于是

$$aaub = aubb$$

即

$$au = ub$$

但 $|u| < |x|$，由归纳假设，$au \neq ub$。这个矛盾证明了定理。 ■

总可以改写使用串值归纳法的证明以便引用下述原理：如果某谓词对于 N 中的某个元素为真，则 N 中一定有使该谓词为真的最小元素。下面以这种方式给出定理 7.2 的证明：

假设存在一个字符串 $x \in \{a, b\}^*$ 使得 $ax = xb$，则必存在满足这个方程的长度最小的字符串。设 x 是这样的一个字符串，那么，$ax = xb$。但是，如果 $|u| < |x|$，则 $au \neq ub$。然而，$ax = xb$ 蕴涵 $x = aub$，所以 $au = ub$ 且 $|u| < |x|$，这个矛盾证明了这个定理。 ■

练习

1. 用数学归纳法证明 $\sum_{i=1}^{n} i = n(n+1)/2$。

2. 这里有一个用数学归纳法的"证明"，它"证明"如果 $x, y \in N$，则 $x = y$，错在哪

里？

对于 $x,y\in N$，令

$$\max(x,y)=\begin{cases}x & \text{若 } x\geqslant y\\ y & \text{否则}\end{cases}$$

考虑谓词

$$(\forall x)(\forall y)\ [\text{If } \max(x,y)=n,\ \text{ then } x=y]$$

对于 $n=0$，这显然是正确的。假设这个结论对于 $n=k$ 成立，且设 $\max(x,y)=k+1$，令 $x_1=x-1$，$y_1=y-1$，则 $\max(x_1,y_1)=k$。由归纳假设，$x_1=y_1$，因而，

$$x=x_1+1=y_1+1=y$$

3. 下面有另一个错误的证明，它声称用数学归纳法证明了所有的花有相同的颜色！错在哪里？

考虑谓词：如果 S 是由 n 朵花组成的集合，则 S 中的所有花有相同的颜色。如果 $n=1$，这个谓词显然为真。我们假设对于 $n=k$ 它为真，且证明该结论对于 $n=k+1$ 也成立，于是，设 S 是 $k+1$ 朵花的集合。如果从 S 中拿走一朵花，则得到 k 朵花的集合。因而，由归纳假设它们有相同的颜色。现在将拿走的花取回且拿走另一朵花。再由归纳假设，剩下的花都有相同的颜色。而现在两次拿走的花已被证明与其余的花有相同的颜色，于是，S 中所有的花有相同颜色。

4. 证明不存在字符串 $x,y\in\{a,b\}^*$，使得 $xay=ybx$。

5. 给出不使用数学归纳法的定理 7.2 的“单线”证明。