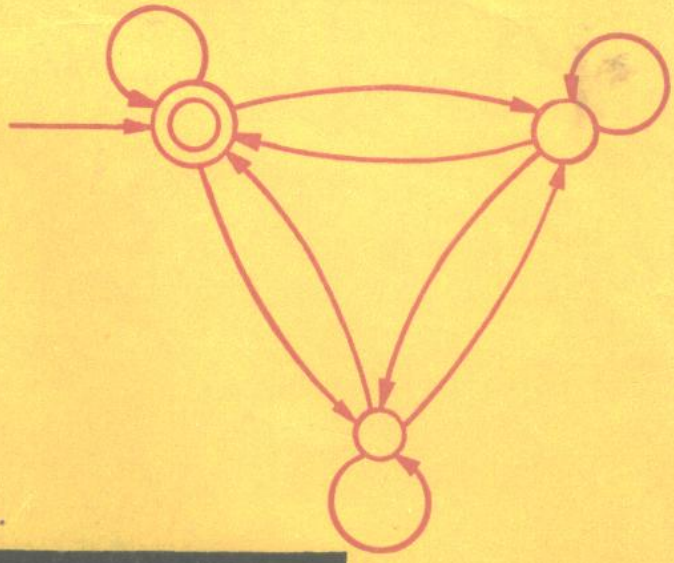
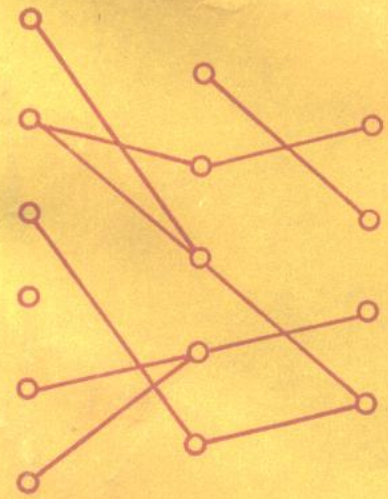
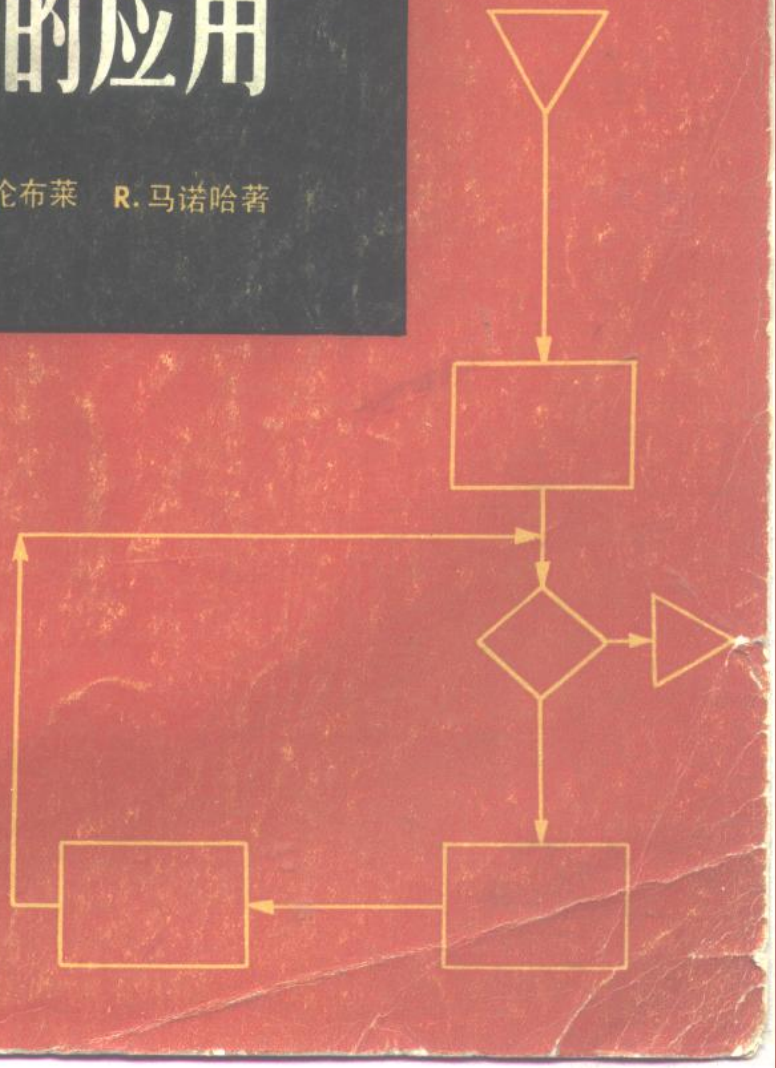


离散数学结构及其在计算机科学中的应用

J. P. 特伦布莱　R. 马诺哈著





离散数学结构及其在计算机科学中的应用

J. P. 特伦布莱　R. 马诺哈　著

罗远诠　李盘林

迟忠先　张清绵　裘春航　译

上海科学技术出版社

Discrete Mathematical Structures
with Applications to Computer Science

J. P. TREMBLAY R. MANOHAR
McGraw-Hill Inc. 1975

离散数学结构及其在计算机科学中的应用

J. P. 特伦布莱 R. 马诺哈 著

罗远诠 李盘林

迟忠先 张清绵 裘春航 译

上海科学技术出版社出版
（上海瑞金二路 450 号）
新华书店上海发行所发行 上海中华印刷厂印刷
开本 787×1092 1/16 印张 28 字数 668,000
1982 年 4 月第 1 版 1983 年 10 月第 2 次印刷
印数 10,201—18,200

统一书号：13119·966 定价：（科五）3.15 元

译 者 的 话

本书译者分工如下：第一章罗远诠，第二章李盘林，第三、六章及附录迟忠先，第四章张清绵，第五章裘春航.

徐润炎同志帮助校阅了全部译稿，裘春航同志进行了部分校阅工作.

本书译出后，在使用中发现原文有若干错误，这些已根据译者的理解予以改正.

限于译者水平，一定有不少错误. 恳切希望读者指正.

译者 1981年1月

序　言

一些计算机科学的高级书籍，都是用一章假定读者已知的数学专题的选编作为开始的．这些专题的阐述常常是简明的，但是其所概括的主要结果对于书中以后部分来说，却是必备的知识．然而读者不可能从这样简要的处理来学习这些专题．对于攻读计算机科学的大学生，通过听数学系所开的每一门专题课程去学习他们所需要的全部专题也是不可能的．一般来说，趋向是选择若干计算机科学研究领域中是主要的数学专题，然后让学生通过其它的途径去接受这方面的训练．类似的情况也出现在大多数工程课程中．按照这种精神，本书讨论经过选择的一些专题，他们在数学中被称之为"离散数学"．除了中学中所教的那些数学之外，不要求读者具备其它的预备知识．即使许多听离散数学课的大学生在一年级已经上过微积分课程，但这门课对学习本书来讲，也绝对不是必须的预备知识．但是，一些附加的数学课程，有助于使学生的数学修养更成熟．

我们的目的不是使本书包含离散数学中所有的专题．某些读者可能将会感到书中省略了计算技术、排列和概率等内容，我们认为许多中学生是已经掌握这些内容了．

我们所选择的一些专题，其意图是尽可能向读者介绍大多数基本的术语，而这些术语是在许多高级计算机科学课程中要用到的．为了起到引导学生的作用，我们感到在引进术语的同时介绍有关的应用，这种做法是很重要的．采用这种作法有几个优点，学生首先会看到一些抽象思想的关联，因而得到启发，而且他们会获得应用这些思想来解决实际问题的信念．

我们希望强调的是，概念和术语在使用之前要介绍好．否则，学生必须一面去掌握基本工具，另一面还要去熟悉使用这些工具的题材．这本书的大部分材料对于许多计算机科学来说是必备知识．它们应当在大学三年级之前就讲授完．在编写本书时注意到了这一事实．

我们要讨论的数学专题有逻辑、集合论、代数结构、布尔代数、图论和可计算性理论的基本知识．虽然在这些领域中都有相应著名和杰出的书籍，但我们在介绍这些专题时仍然保持一种特色，即注意到读者在一定实际应用中，特别和计算机科学有关的应用中最终将使用它们．只要有可能，我们就力求以适当明确的数学形式介绍理论材料，而避免冗长的原理的讨论，避免悖论问题和对某些理论的任何公理化的探讨．我们这本书中所选择的那些题目也有助于计算机科学教学计划中更高级的课程，例如自动机、可计算性、人工智能、形式语言和语法分析、信息结构和检索、开关理论、计算机离散结构表示法以及程序设计语言这样一些方面．我们希望读者掌握了本书中的理论材料之后，将使学生在阅读计算机科学的文章和书籍时，能对在一开始所提及的那些数学准备知识有充分的理解．

由于读者可能不清楚怎样用数学或者在何处用到数学，所以我们讨论了某些数学结构的计算机表示法．我们从许多不同的领域中选择各种应用，以说明计算机科学中引入离散结构是非常必须的．书中对大多数应用写出了算法，并且对其中一部分给出了计算机程序．但对离散结构的计算机表示和处理，如串、树、群以及丛都没有进行详尽的讨论，而只是讨论到对某个特定的实际应用形成解答为止．

第一章讨论数理逻辑. 对于在教育、商业、经济和社会科学方面的学生，常常在一门名称为"有限数学"的课程之下，接受关于逻辑方面若干专题的初等教育. 但是，这样一些讨论常常以构造真值表而结束. 而且在某些例题中，简单地介绍了语句演算的推断理论. 为了使学生能阅读计算机科学的技术文章和书籍，有必要使他们知道一些谓词演算的知识. 所以我们在关于逻辑的讨论中比通常的有限数学书中更进一步了. 我们还避免了那些供数学家和哲学家所看的数理逻辑书中开展的哲学讨论和艰涩的细节. 本章还简要地介绍了怎样把逻辑应用于二值器件.

第二章讨论集合论. 在讨论中保持了某种程度的数学严密性并且有时给出了证明. 但是我们不提出集合论的悖论问题和公理化方法. 讨论了集合、关系、次序和递归函数. 这一章还介绍了某些结构的计算机表示和使用. 也给出了一个反映集合论和逻辑内在关系的例子. 此外还相当详尽地讨论了递归(及其实现)这个专题，因为许多程序设计语言允许使用这种结构. 而且递归这个概念就其自身来说也是很重要的. 因为计算机科学家在整个一生工作中很难说是会不碰到递归的问题. 这一章以提出命题演算中用作证明定理用的一个算法作为结束.

第三章讨论代数结构. 大多数近世代数书中几乎都把注意力集中在群论上，而对半群和独异点却很少说到. 后者是本书要强调的，因为半群和独异点理论在计算机科学的某些领域，例如形式语言理论、语法分析及自动机理论中都是非常重要的. 本章还包括许多应用，诸如波兰表达式及其编译、语言和文法、快速加法器、错误检测和校正码.

第四章讨论布尔代数以及它在开关理论和时序机上的应用. 这一章介绍了布尔函数的最小化及其在数字计算机系统逻辑设计上的应用，也讨论了时序机以及它们的等价物.

第五章简单介绍了图论. 图论的原理在所有计算机领域几乎是不可缺少的. 这里给出了图论在一些领域中的应用，例如语法分析、计算机故障检查和诊断以及最小路径问题. 这一章也讨论了图的计算机表示和使用，还包括某些重要的算法.

最后，第六章对可计算性理论作了非常简单的介绍. 指出了有限态接收机和正则文法之间的等价性. 最后介绍了有效过程的概念，同时也证明了图灵机可以计算任意的部分递归函数.

为了进一步理解计算机科学各个领域中概念的应用，本书的习题有的偏向理论方面，而有的是程序设计方面的. 本书所编的材料，除了关于逻辑那部分以外，大部分是和 ACM(美国计算机协会)课程委员会推荐作为"离散结构引论"那门课程的内容相吻合的.

我们希望这本书将对计算机科学家、工程师、期待对离散数学的内容有一个中等程度的掌握的非数学专业的学生，以及想要熟悉计算机科学理论的应用的数学家们都是有用的. 具有现代逻辑和代数的初步基础的学生，用一个学期就可以掌握这些材料. 对于其他不具有这些基础的学生，那末这本书可以用作两学期的课程. 也可以选择一些专题作为一个学期的课程. 如果略去书中关于应用的那些部分，那末也不致会有损于材料的连贯性. 本书是根据笔者在萨斯卡通(Saskatoon)的萨斯卡乞旺大学 (The University of Saskatchewan) 四年讲授离散结构这门课程的经验而写成的.

我们假定读者基本上熟悉标准的 FORTRAN 和 PL/1，PL/1 在包括递归或者表结构的应用中是顶用的. (以下为致谢部分，译略.)

J. P. TREMBLAY　R. MANOHAR

目　录

第一章 数理逻辑

引 言

逻辑学的一个主要目标，是提供一组能够决定任何一个论证或推理是否是有效(正确)的规则.

一切推理，无论是法律的论证，还是数学证明，或是在科学理论中从一组假设得出的结论都与逻辑有关. 由于逻辑在应用上的多样性，这些规则(被称为推理规则)必须以一般的术语来表述，而且必须与任一特定的论证或所涉及的学科无关. 这些规则也应该与在论证中所使用的任一特殊的语言无关. 更确切地说，在逻辑学中我们关心的是论证的形式，而不是论证本身. 象任何其他科学理论一样，我们把推理理论以这样的方式形式化，使得能够按照上述规则机械地，并且与我们自己对论证的感觉无关地判定一个论证的有效性. 当然，用这种方式处理，要求把规则表述得没有二义性.

表述任何一组规则或任一理论都需要使用语言. 自然语言并非总是足够精确的. 他们还有二义性，因而使用它们是不合适的. 所以，必需首先制定一种被称为对象语言的形式语言. 形式语言是一种语法被完善地定义了的语言. 事实上，科学中的每一个学科都在发展它自己的对象语言，它由某些已完善地定义了的术语以及这些术语的规定妥当的用法所组成. 逻辑学与其他学科仅有的差别是：在其他学科中所关心的是对象语言的使用，而在逻辑学中对于分析对象语言如同使用对象语言一样地感兴趣. 事实上，在本章的前半部分我们将着眼于制订与分析对象语言，而不考虑它在推理理论中的应用. 这些研究在计算机设计以及某些二值器件中有重要的应用(见§1.2.15). 我们着重逻辑学的这一部分，是因为形式语言的研究已成为研制人与计算机的通讯手段的重要部分. 这一研究的继续是§1.4的关于推理理论的研究. 我们很快就可以明白，至今所研究的对象语言有很大的局限性，这个推理理论甚至不能概括某些简单的论证形式. 所以，在§1.5中我们把对象语言扩展到包含谓词，然后在§1.6中讨论谓词逻辑的推理理论.

为了避免二义性，我们使用对象语言中已明确地定义了的符号. 使用符号的另一个原因是它们便于书写和使用. 由于使用了符号，因此我们所研究的逻辑也称为符号逻辑. 在我们研究对象语言时必须使用另外的语言. 为此，我们可以选择任一种自然语言. 这里我们选择英语. 因此关于对象语言的语句将用英语来写. 而把这一自然语言(英语)称为元语言[注]. 可以预料，在这个过程中将有某些固有的困难. 因为我们希望研究一种精确的语言，而使用的却是不那么精确的另一种语言.

§1.1 语句和表示法

这一节介绍对象语言的某些基本单位，我们把它们称为原始(本原、原子)语句. 首先假

[注] 因原书是用英语写的，故作者如此说. 对于读者现在使用的中译本来说，元语言是汉语. ——译者注

定：对象语言包含一组陈述句子，它们不能剖开或分解为更加简单的句子，这些就是原始语句．在对象语言中，我们只采用这样的陈述句子，它们有且仅有两个可能的真假值中的一个值．这两个真假值是"真"和"假"，我们分别用符号 T 和 F 表示，有时也用符号"1"和"0"表示．真假值与我们对于这些所采用的句子的正确或错误的感觉毫无关系，因为这些感觉是主观的而且依赖于上下文．对我们来说，只要假定总可以赋给一个陈述句子以两个可能值中之一就足够了．在我们的研究中，只关心将特定的真假值赋给陈述句子的效果，而不关心这些语句实在的真假值．因为我们采用的只是两个可能的真假值，所以有时把这种逻辑称为*二值逻辑*．我们要研究一种手段，按照它我们可以在对象语言中构造具有两种可能真假值之一的另外的陈述句子．注意，在对象语言中不采用任何其他类型的句子，例如：感叹句、疑问句等等．

在对象语言中陈述句有两种类型．第一种类型是对象语言中的原始语句，用大写字母 $A, B, C, \cdots, P, Q, \cdots$ 表示；第二种类型的陈述句是由原始语句用某些称为*联结词*的符号，以及某些标点符号（例如括号）联结起来而得到的句子．总之，可以而且仅可以赋给它以两种可能的真假值之一的陈述句子称为*语句*．不包含任何联结词的语句称为*原子（原始、本原）语句*．

现在，我们给出句子的例子，并且说明为什么其中有些没有在对象语言中采用，因此也不打算给以符号化．

(1) 加拿大是一个国家．

(2) 莫斯科是西班牙的首都．

(3) 这个语句是假的．

(4) 1+101=110．

(5) 把门关上．

(6) 多伦多是一座古老的城市．

(7) 1980年人将到达火星．

显然，语句(1)和(2)的真假值分别是真和假．按照我们的定义，句子(3)不是语句，因为我们不能适当地指派它一个确定的真假值．如果我们指派它的值为真，那末句子(3)说明语句(3)的值是假．另一方面，如果我们指派它的值为假，那末句子(3)蕴含着语句(3)取真值．这个例子说明了一种语义上的自相矛盾．(4)是一个真假值取决于上下文的语句．即，如果我们所说的数是十进制的，那末这是一个假的语句；另一方面，对二进制数来说它是一个真的语句．语句的真假值常常取决于它的上下文，这通常不加说明但还是清楚的．我们马上就会看到：我们不是非知道语句的真假值不可，我们关心的只是它具有真假值这个事实．在这种意义上，(4)、(6)以及(7)都是语句．可以看出，语句(6)在世界的某些地方被认为是真的，而在另一些地方则是假的．语句(7)的真假值只能到1980年或者在这个日期之前有人到达火星时才能确定．但我们对此是不感兴趣的．注意，(5)不是语句，因为它是一句命令句．

一旦我们知道了在对象语言中可采纳的那些原子语句，就可以用符号来代表它们．构造和分析由一个或几个原子语句构成的语句的方法在§1.2中讨论．在这里，先讨论使用和提述语句中名字的若干惯例，然后再描述原子语句符号化的方法．

在构造关于某对象的语句时，通常使用对象的名字而不是对象本身．例如，考虑语句

(8) 这个桌子是大的.

措辞"这个桌子"被用作对象的名字. 而在语句中并不使用真实的对象,即一个特定的桌子. 把真实的桌子放到措辞"这个桌子"的地方是不合适的. 即使在对象很小,而能够把具体对象嵌入它的名字的地方时, 这样做法也不会使我们能够对同一对象做出两个同时存在的语句,而不在这句或那句中使用它的名字. 因此,可以约定,关于某对象的语句不包含对象本身而只包含它的名字. 实际上,这个理所当然的习惯我们是很熟悉的.

现在,我们考虑如下情况,即希望讨论有关某名字的一些事情,以致于在一个要造的语句中,这个名字成为这个语句的一个组成部分. 按照刚才所说的规则,我们不应当在语句中使用这个名字本身,而使用这个名字的某种名字. 如何给一个名字命名呢? 常用的方法是用引号把名字括起来,并把它作为这个名字的名字来对待. 例如让我们观察下面二个语句:

(9) Clara 很聪明.

(10) "Clara" 包含 5 个字母.

在(9)中,说的是关于名叫 Clara 的人的某件事. 但是,语句(10)不是关于一个人而是关于一个名字的. 于是,"Clara" 被当做这个名字的名字. 把人名放入括号内清楚地表明,语句(10)是关于名字而不是关于人的.

这个约定也可以这样说明:当某词用作被考虑对象的名字时,我们称为在句子中使用这个词;另一方面,当某词不是作为对象的名字而是作为这个词本身的名字时,我们称为在句子中提述这个词. "提述"一词意味着:该词本身已经成为被考虑的对象.

在整个这本书中,不仅要构造通常所考虑的对象的语句,也构造关于其他语句的语句. 于是,就有必要给所考虑的语句命名. 给名字命名的方法也可用来给语句命名. 用引号括起的语句将作为该语句的名字. 更一般地,任一括在引号中的辞句可以作为该辞句的名字. 换句话说,所谈论的辞句是放在引号中的. 上述讨论可用如下语句来说明.

(11) "Clara 很聪明"包含"Clara".

语句(11)是关于语句(9)以及单词"Clara"的语句. 这里,语句(9)首先被引号括起作为它的名字,然后这名字与名字"Clara"一起用在(11)中.

上述讨论中,我们已经使用了给语句命名的另外一些方法. 这些方法之一是把语句用与全文隔开一行的方法排印. 我们假定这种排印法与使用引号分界语句的方法在本书中具有同样的效果. 往后,我们有时在这些语句的左边插入一个数字作为此语句的编号. 以后引证该语句时,这个数字就作为它的名字. 本书内写这些数都不用加引号. 这种排印法以及给语句编号的方法可以减少引号的数目. 在全书中,给语句命名时将混合使用各种方法,例如语句

(12) "Clara 很聪明"是真的.

可以被写成"(9)是真的",或等价地写成

(12a) (9)是真的.

特定的人或对象可以有不止一个名字. 一个肯定的原则是:在给定的语句中,某对象的名字可用同一对象的其他名字代替而不改变此语句的含义. 这个原则已在语句(12)和(12a)中用过.

我们将使用刚才所讨论的构造名字的方法来构造语句的名字. 通常,从上下文看来名字和对象之间的差别是清楚的,而数学著作中却常常不加区别. 然而实际上有时会导致混

乱.

在许多程序设计语言中,也存在与上面所讨论的名字、对象这些概念相类似的情况. 特别是当调用一个过程(函数或子程序)时,经常需要把变量名和它的值加以区别. 过程语句的自变量(也称为实在参数)或者通过名字或者通过值,与过程的(形式)参数发生联系. 如果通过值来联系,那末只是将自变量的值传送给相应的参数. 这种处理意味着不能从函数内部改变这个自变量的值,因为不知道这个自变量存放在计算机内存贮器的何处. 另一方面,换名调用使自变量的名字或地址与过程发生联系. 这种联系允许在过程中用指令改变自变量的值. 下面讨论一些程序设计语言是如何实现换名调用和赋值调用的.

在某些 FORTRAN 编译程序中,例如 IBM 的 FORTRAN H 和 G 中,将变量的名而不是它的值传递给函数或子程序. 这个约定也适用于自变量是常数的情况. 常数的地址(它存储在编译程序的某个符号表中)被传送给函数的相应的参数. 这种处理方法可能导致灾难性的结果. 例如,考虑由下面一组语句描述的简单函数 FUN:

```
      INTEGER FUNCTION FUN(I)
      I=5
      FUN=I
      RETURN
      END
```

假定调用 FUN 的主程序由下列普通的语句组成:

```
      K=3
      J=FUN(K)*3
      L=FUN(3)*3
      PRINT 10, J, L
10    FORMAT(1H, I3, I3)
      STOP
      END
```

这个程序所产生的变量 J 和 L 的值分别是 15 和 25. 计算 J 时,在函数内部 K 的地址是已知的,在函数中,当执行语句 I=5 时 K 的值被改变成 5. 从这个函数返回时的函数值是 5,同时得到 J 的值是 15. 然而计算 L 时情况很不相同. 3 的地址被传递给函数. 由于相应的参数 I 已变成 5,故主程序符号表中 3 的值也被改变成 5. 注意,对符号表中常数 3 的表值的所有引用都是在编译时就完成了的,在主程序中余下部分一直是引用 3 的表值,但现在这个值是 5 而不是 3. 更明确地说,算式 L 中乘法的右操作数的名字 3 的值已经是 5.

其他的 FORTRAN 编译程序通过对每一个是常数的自变量设立一个哑变量的方法来防止这种结局. 这些内部(哑)变量程序员是取不到的. 相应于哑变量的参数被改变时只改变哑变量的值,而不改变构造出这个哑变量的原始自变量的值.

WATFIV 允许变量以值来传递,只要将这种变量放在两个斜杠之间即可. 例如,在函数调用

```
TEST(I, /K/, 5)
```

中,K 的值被传送给函数 TEST.

在 PL/1 中,自变量既可以用值也可以用名字来传递. 如果自变量被括在圆括号中,则

它由值来传递；否则它由名字来传递．在函数调用

TEST(I, (K), 5)

中，自变量 I 和 K 分别用名字和值来传递．

如前所述，在符号逻辑中用大写字母 A, B, …, P, Q, …（T 和 F 例外[注]）以及带下标的大写字母来表示语句．例如，我们写

(13) P: It is raining today.

在语句(13)中，包含着这样的信息："P"是符号逻辑中的一个相应于英语语句"It is raining today"的语句．这种情形类似于将同一语句翻译成法语"Aujourd'hui il pleut"．因此，语句(13)中的"P"(即"It is raining today")和"Aujourd'hui il pleut"是同一语句的名字．注意，作为语句的名字用的是"P"而不是用 P．

§1.2 联 结 词

我们已经介绍了语句及其真假值的概念．简单语句的真假值是很显然的．然而，可以用某些起联结作用的字或短语，从简单语句构造更为复杂的语句，这些字或短语被称为"*句子联结词*"．英语中已有一些这样的联结词．由于它们可在各种各样的含义下使用，所以必须定义一组有确定含义的联结词．用符号表示这些新的联结词是适宜的．在这一节中，我们先定义这些联结词，然后建立一种确定由它所形成的语句的真假值的方法；并讨论这些语句的各种性质以及它们之间的某些关系．此外，我们还指出：语句和这些联结词一块儿定义一种满足一组性质的代数．根据这些性质可以把语句当作对象来进行某些运算．这里所研究的代数在开关理论和计算机逻辑设计中有有趣和重要的应用(见§1.2.15)．§1 至§4 所讨论的推理理论也用到它的某些结果．

我们先考虑简单语句，即*原子或原始语句*．如上面所述，应用句子联结词可以从原子语句形成新的语句．后者被称为*分子或合成语句*．于是，原子语句是不包含任何联结词的那些语句．

在日常语言中，我们使用"与"、"若非……即"、"或"等联结词，把两个或更多的语句结合成另外的语句．但是，它们的使用并非总是精确而且无二义性的．所以，在我们的对象语言中不预备把这些联结词符号化，然而，将定义与英语中的联结词有某些类似之处的联结词．

在§1.1 中已经介绍了用大写字母 P, Q, …, P_1, P_2, … 来表示语句的想法．现在，还打算用同样的符号(即带或不带下标的大写字母)来表示任意的语句．在这种意义下，语句"P"或者代表特定的语句，或者替任意一个语句占有一个位置．前者称为*常量*，后者称为*变量*．同一符号有这样两种不同用法不会引起任何混淆，因为可从上下文来作出判断．"P"的真假值是它所表示的实际语句的真假值．应当着重指出：当"P"被用作语句变量时，它没有真假值，因而在符号逻辑中它就不代表语句．我们明白，如果要替换掉它，它只能被一个语句所代替．然后才能确定它的真假值．把这种"P"叫做"*语句公式*"是适宜的．"语句公式"的概念在§1.2.4 讨论．然而，在其后面几节中我们时常把名词"语句公式"简称为"语句"．这种简化使得我们的讨论简单，并且突出了所介绍的联结词的意义．

例如，令

[注] T 代表 true，即真；F 代表 faulse，即假．——译者注

P: 今天下雨.

Q: 正在下雪.

又令 R 是一个可以被 P 和 Q 代替的语句变量. 如果没有指派什么来代替 R, 它仍然是一个语句变量而没有真假值. 另一方面, P 和 Q 的真假值是可以确定的, 因为它们是语句.

§1.2.1 否定

语句的否定通常是这样构成的: 在语句的适当地方插入"不"字, 或者把短语"这件事不成立"放在语句的后面. 如果"P"表示一个语句, 那末"P"的否定写为"$\neg P$"而且读为"非P". 如果"P"的真假值是 T, 那末"$\neg P$"的真假值是 F; 同样, 如果"P"的真假值是 F, 那末"$\neg P$"的真假值是 T. 否定的这种定义总结在表 1.2-1 中.

表 1.2-1 否定的真假值表

P	$\neg P$
T	F
F	T

注意, 我们没有用引号来表示该表中语句的名字. 这个做法与我们以前用过的, 把语句与全文分开而且隔开一行来写的方法是一致的. 从现在起, 只要不引起混淆, 甚至在本文内也不用带引号的符号来给语句命名. 现在举例说明语句的否定的构成.

考虑语句

P: 伦敦是个城市.

那末 $\neg P$ 是语句

$\neg P$: 伦敦是个城市这件事不成立.

通常, $\neg P$ 也可以写为

$\neg P$: 伦敦不是个城市.

虽然, "伦敦是个城市这件事不成立"和"伦敦不是个城市"这两个语句是不完全相同的, 但它们都是由 $\neg P$ 翻译出来的. 其理由是在英语中这两个语句有相同的含义. 在对象语言中, 一个给定的语句用一个符号表示, 而它可以对应于英语中多个语句. 产生这种多样性的原因是: 在自然语言中, 一个对象语句有各种不同的表达方法.

例如, 如果某语句是

P: 昨天我去上课.

那末 $\neg P$ 是下述语句中的任一个:

(1) 昨天我没有去上课.

(2) 昨天我是课堂的缺席者.

(3) 昨天我去上课这件事不成立.

在这里符号"$\neg$"被用来表示否定. 文献中用到的其他符号有"$\sim$", 上面加一横, 或者"非", 于是 $\neg P$ 被写为 $\sim P$, $\bar{P}$, 或者非 P. 注意, 否定虽然仅仅修饰一个语句, 但仍把它叫做联结词. 在这个意义上, 否定是一个一元运算, 它对单独的语句或语句变量进行运算. "运算"一词在第二章中再作解释. 现在只要注意到, 对语句进行运算产生另外的语句这一点就

足够了. 我们选择"¬"表示否定, 因为这个符号在逻辑学的书籍中以及各种程序设计语言中是通用的. 这里, 我们将使用这些程序语言中的一种.

§1.2.2 合取

两个语句 P 和 Q 的合取是语句 $P \wedge Q$, 它被读为"P 与 Q". 当 P 和 Q 的真假值都是 T 时, 语句 $P \wedge Q$ 的真假值为 T; 否则, 它的真假值总为 F. 合取用表 1.2-2 来定义.

表 1.2-2 合取的真假值表

P	Q	$P \wedge Q$
T	T	T
T	F	F
F	T	F
F	F	F

例 1 构造下列语句的合取.

P: 今天下雨.

Q: 这个房间里有 20 个桌子.

解 今天下雨而且这个房间里有 20 个桌子.

在日常语言中, 通常两个语句之间有某种关系才使用合取"与". 这样一来, "今天下雨与 2+2=4"这样的语句听起来是可笑的. 但是, 由语句"今天下雨"与"2+2=4"构成的上述语句在逻辑学中是完全可以接受的.

例 2 把下面语句翻译为符号.

雅克和吉尔走上山坡.

解 为了把它写成两个语句的合取, 首先要把这个语句意译为

雅克走上山坡与吉尔走上山坡.

现在, 如果写

P: 雅克走上山坡.

Q: 吉尔走上山坡.

那末它可写成符号的形式: $P \wedge Q$.

至今, 我们已经看到: 符号 $\wedge$ 被用来翻译出现在英语中的联结词"与". 然而, 联结词"与"有时候有另外的含义, 在这种情况下, 它不能翻译为上面所定义的符号 $\wedge$. 为了看清这种区别, 考察下述语句:

(1) 玫瑰是红的与紫罗兰是蓝的.

(2) 他打开书本与开始读.

(3) 雅克与吉尔是堂兄弟.

语句(1)中联结词"与"的含义和符号 $\wedge$ 的含义是相同的. (2)中单词"与"作"然后"理解, 因为"他开始读"所说的动作发生在"他打开书本"所说的动作之后. (3) 中的单词"与"不是联结词. 注意, 合取的定义关于 P 和 Q 是对称的; 这就是说, $P \wedge Q$ 以及 $Q \wedge P$ 对确定的 P 和 Q 的值都有相同的真假值. 显然, 如果我们把(1)写为

紫罗兰是蓝的与玫瑰是红的.