智能模拟

复旦大学　谢铭培

国防工业出版社

智 能 模 拟

复旦大学　谢 铭 培

国 防 工 业 出 版 社

智 能 模 拟

复旦大学　谢铭培

国防工业出版社出版

北京市书刊出版业营业许可证出字第 074 号

解放军第七二二六工厂印刷　内部发行

*

787×1092 1/16　印张 11 6/8　272 千字

1981 年 8 月第一版　1981 年 8 月第一次印刷　印数 1-4,000 册

统一书号：N15034（四教 72）定价 1.25 元

内 容 简 介

本书介绍了智能模拟方面的一些基础知识及基本原理。

全书共分五章。第一章介绍了智能模拟的概况及必备的数学知识。第二章详述了问题求解的各种方法。第三章系统地介绍了自动定理证明的基本知识。第四章简要介绍了模式识别的统计决策方法及模式文法。并介绍了自动机在模式识别中的应用。第五章概要地介绍了智能机器人。如机器视觉、对答系统等。

本书可作为高等学校有关自动控制、计算机科学与工程等专业的教科书。也可供从事这方面研究工作的科技人员参考。

前　言

本书系高等学校工科电子类自动控制专业统编教材之一。但它也可作为其他理工科计算机科学、计算机工程、心理学等有关专业的选修课教材。也可作为有关专业研究生的参考书。

智能模拟（即人工智能）这门学科是一门较新的学科，也是一门较艰难的学科。随着我国科学技术的迅猛发展及计算机科学技术的日益普及，掌握这门学科知识的欲望也在提高。所以有必要编写这门教材。这就是我们编写这门教材的宗旨。但考虑到我国的现状，我们只介绍其基本原理，而对如何用语言（如LISP，FORTRAN）来实现它，则没有加以介绍。

在第一章中简要地介绍了本课程必备的集合论和图论的知识。这是为了照顾到有些学生的数学知识欠缺之故。在第二章中较详细地介绍了问题求解的几种方法。在第三章中较系统地介绍了自动定理证明的基本知识。例如，命题演算、谓词演算及分解原理。在第四章中扼要地介绍了模式识别方法。例如，决策论方法及模式文法。还介绍了自动机的基本知识及其与形式语言的关系。第五章在智能机器人的标题下，介绍了多种技术，且较多地介绍了对答系统。

本课程的学时数为54学时。第一章为6学时；第二章为12学时；第三章为12学时；第四章为12学时；第五章为12学时。这样安排是很紧凑的。仅供讲课教师参考。他们可以根据各自的具体情况灵活安排。例如：(1) 已学过《离散数学》课程，则第一章可以不讲，让学生自学，熟悉一下基本符号即可。(2) 从知识的连贯性角度看，各章几乎是独立的。甚至于在一章中（除了第三章而外）有些节也是独立的。删去某些章节并不影响知识的系统性。所以可以只选择其中几个章节详细讲解。

在各章之后附有习题。书后还附有词汇索引。读者还可从书后所附的参考文献中找到其他参考文献。

清华大学计算机工程与科学系林尧瑞、石纯一、刘植桢、黄昌宁等同志审阅了本书并提出许多宝贵意见。特此深表感谢。

对本书进行审阅工作的还有：复旦大学计算机科学系朱洪（第一章）、吴立德（第二章）、廖有为（第三章和第四章）和倪重匡（第四章）等同志。他们提了许多宝贵意见。另外，复旦大学计算机科学系自动化教研室的李宗葛、陆道政、许时明、王松年、王宗彩等教师为编写该书协助搜集了许多资料。特表谢意。

由于作者水平有限，必有许多错误之处。望请读者多多批评指正。

编　者

1980.5.

目　录

第一章　概论及数学知识准备*

1.1　概　　论

智能模拟，又名**人工智能**或**机器智能**，是研究使机器能够做一些需要人类的智慧才能完成的工作的一门学科。“需要人类的智慧才能完成”这点相当重要。人类原先幻想孙悟空的“千里眼”和“一觔斛翻十万八千里”，现在的雷达技术、飞机、火箭等业已实现。但是，雷达、飞机，火箭等等都是人的眼、腿等等五官和四肢的延伸。它们并不是大脑的智慧劳动才能完成的。因此，这方面的工作不属于智能模拟这门学科的范畴。但是，什么工作才算是智慧劳动？这个问题至今仍未明确。四则运算难道不是智慧劳动吗？但是，一般的计算器都能做这项工作，却算不上智能机。因而，确切地说，什么是智能模拟，至今都没有一个很明确的定义。随着科学技术的突飞猛进，对于“智能模拟”的定义要求也愈来愈高了。在现阶段，问题求解，定理证明，博奕、模式识别、判决、自然语言理解，对答系统以及景物分析等等都是属于智能模拟这门学科的范畴。

直到目前为止，智能活动的奥秘仍未揭开。但是，却引起科学家的重视。人（包括某些动物在内）如何用其大脑（物质基础）来进行智能活动的？甚至连早期人类都还不知道“思考问题是通过大脑”的情况下，人类也已经在进行智能活动了。这些奥秘引起了心理学家和生理学家的重视。在 1943 年，W.S.McCulloch 和W.H.Pitts 就提出了神经网络的数学模型。他们从神经网络的生物原型出发，最早最明确地探索了脑功能。有人把这一尝试看作是智能模拟的开端。

不过，从 Leibnitz 开始，就考虑用机械性的办法来证明定理了。后来，人们做了一番努力，试图寻找能够从一个公理系统中得出全部永真公式的算法。但以失败而告终。到了三十年代，Herbrand 找到了一个算法。如果某个公式是永真公式，则利用这个算法可以验证它。但若它不是永真公式，则该算法可能永不停止。由于当时的计算技术条件限制，都无法得到较好的效果。后来人们又做了一番努力，也因效率太低而未能实用。1957 年心理学家 Allen Newell、Herbert Simon 和 J.C.Shaw 合作编制了名为“逻辑理论家”（LT）的程序，当时就成功地证明了 Whitehead 和 Russell 的一部数学名著《数学原理》第二章 52 个定理中的 38 个。另 14 个定理由于当时计算机的限制未能证出。直到 1963 年才在一部较大的计算机中最终全部证明了这 52 个定理。LT 模拟了数理逻辑在证明定理时的思维规律，把一个问题分解为若干个子问题，而后根据记忆中的公理或已被证明了的定理，用代入法（如把常量代入变量）、替换法（以一个逻辑符号替换另一个逻辑符号）来求解这些子问题，从而最终解决这个问题。

1959 年 H.Gelernter 发表了证明平面几何问题的程序。J.Slagle 于 1963 年发表了求不

---

* 本章主要参考文献见“参考文献”6、11、12、15、16 和 17。

不定积分的搜索程序，编进了人们求不定积分的许多技巧。结果，机器的解题行为和人很相似。

到了 1965 年，Robinson 提出了分解原理，使定理的自动证明又有了重大突破。对 这个方法，人们也提出了许多改进办法，使之达到相当实用程度。

1976 年美国伊利诺斯大学数学家 K.Appel 和 W.Haken 用计算机证明了“四色定理”，轰动了世界数学界。1878 年 A.Cayley 在伦顿的数学会上提出“在任意地图上着色，只需四种颜色就够了”的猜想。1879 年 A.B.Kempe 发表文章加以 证 明了。可 是 在 1880 年 P.J.Heawood 指出“该证明有误，只能说明‘五色就够了’，并不能充分证明‘四色 已 够了’的结论”。从此，“四色猜想”仍是一个猜想。但是 Kempe 的证明方法仍然是可取的。许多数学家做了不懈的努力。这么过了近百年，K.Appel 和 W.Haken 才应用新算法，在计算机上花了一千二百小时的计算时间才得到了证明。这就大大推动了以计算机为基础的智能模拟的发展。即使这样，人们仍然提出这只是一个计算机辅助的定理证明，并非机器有智能，智能的主要角色还是人。因此，人们提出了计算机与人如何合理分工的问题。把一些繁琐的细微工作让给计算机做，它是一丝不拘的。而人则做一些创造性的工作。即让机器协助人工作。

IBM 的 A.Samnel 让计算机向人学习下跳棋，结果机器于 1962 年击败了美国某州的跳棋冠军。

在下五子棋方面，计算机已达到相当高的水平。没有经过深造的人是很难胜过它的。

在下国际象棋方面，让计算机记住开局和残局的棋谱，并给予一定训练，也有相当的下棋水平。国际上有时还举行计算机象棋比赛。

在以上所述的问题求解方法中，绝大部分属于演绎法思维范畴。而如何让机器像人那样从观察到的许多事实中概括出一般性规律来，这种归纳法思维过程，至今仍是很大的难题。

让机器像人的视觉、听觉那样来识别文字、图形、脸谱、声音等等这类模式识别方面的研究，已经有了可观的成就。已经可以使机器辨认英文印刷体和某些手写体文字，并应用于文件、资料检索、自动分类、自动排版，自动信件分检等等。汉字的识别也有很大的进展。在日本已有相当的研究。用机器来识别癌细胞、染色体，已经是现实了。它的广泛应用将造福于人类。不过，在模式识别方面虽然应用较广，但是还需要人们花很大的努力。例如，机器对虎视眈眈及温情脉脉的眼神至今还未能分辨。

模式识别的研究方法很多，主要有三个类型。一类是从数学角度出发，称为统计决策方法，有人称之为“数学式的模式识别”。它解决了许多实际问题，而且对图形分类的统计理论也做出了贡献。但是其对特征的抽取没有建立统一的理论，而且当特征矢量的维数高时，往往就遇到困难。另一类是从人识别图形的角度考虑问题的。不仅要辨认出某些特征是否存在，而且要看特征之间的结构关系。正象要理解一句话必须有文法规则一样，识 别 一 个 模式，就是用一些简单的模式元和一定的“文法规则”来描述大量复杂的模式。这种方法称为句法结构方法。第三类是所谓机器视觉。让机器象人一样，根据二维图片识别其所描述的三维物体，例如长方体以及景物。

让机器理解自然语言（包括书写文字的语言及有声语言）也是科学工作者的努力目标。五十年代初的机器翻译，简单化地采用“巨型字典”。但一字多义、一词多义，常无法理解其义而失败。Chomsky 的“短语结构文法”和“转换文法”，对理解自然语言的 研 究 起了

很大作用。但是着重语法分析，往往出现机器在答话时语法完全正确，但却毫无意义而十分可笑。因而，人们就转至语义方面的分析。在自然语言中，有时是不符合语法规则的，但对方却能理解其语义而不苛求。1972 年 T.Winograd 的理解自然语言的程序是把语法、语义和推理规则以互相交织的方式运用的。使机器的理解力更强了。目前，仍是处于试验阶段。对有声语言方面，还有语气问题。而对答系统中，还有上下文关系问题。总之是相当难的课题。总的说来，这样的机器还属于低能儿。

在神经元研究方面，目前也仅是一些设想的模型。有些生理学家还对人的记忆机理进行探讨。但是，有许多奥秘——回忆、联想记忆等都未探明。可以说，在神经元的模拟方面，人类还是门外汉。

智能模拟的研究方法大约从三方面着手。一方面是从数学的角度来模拟人的某些智能。有时用了高等的深奥的数学。另一方面是研究人类在智能活动过程中的心理活动，从而模拟它。再一方面是研究人的神经的生理机能，为智能模拟提供重要的物质依据。这三方面研究，往往是互相结合的，当然也有所侧重。

智能模拟的研究甚至于引起哲学方面的争议。关于机器能否模拟人的智能？机器的智能可否超过人？这些问题至今都还在争论不休。甚至有人认为人也是机器。也有人认为机器如能超过人的智能，这是对人类尊严的污辱。有人主张智能活动也是物质运动的一种形式，因此是可认识，可模拟、可超过的。说不定这场争论会进行到人类灭亡才能停止。

智能模拟的研究确实有很大的难度。进展也是缓慢的。但是在研究的过程中与工程技术，工农业等各方面的实际应用课题结合起来，则会取得不可估量的成效。

1.2 集合论的基本概念

1.2.1 集合与成员

集合 把一些可区别的客体按某些共同特性加以汇集，而具有共同特性的这些客体的全体就称为**集合**，或简称为**集**。

成员 组成一个集合的客体称为该集合的**成员**，或称为**元**。

集合表示法 集合的表示法有二种：

(1) 把所有的成员加以列举，或者列举出其中的某些成员而其他成员可一目了然地类推，再用一对｛ ｝括起来。

例 1.1 $\{a,b,c\}$表示有三个成员 a,b,c 的集合。

$\{1,2,3\cdots\}$表示成员为自然数的集合。

(2) 用公式的形式 $\{x|P(x)\}$ 表示。读为“使得 $P(x)$为真的所有客体 x 的集合”或“满足特性或条件 $P(x)$的所有客体 x 的集合”。

例 1.2 $\{x|x=2^n, n=1,2,3,\cdots\}$表示成员为 2,4,8,… 的集合。

用大写英文字母表示集合，用小写英文字母表示成员。用符号$\in$表示成员属于集合，用符号$\notin$表示成员不属于集合。

例 1.3 $a\in A$ 表示 a 是集合 A 的成员。而 $a\notin B$ 表示 a 不是 B 的成员。

用|A|表示集合 A 的成员个数。成员个数是有限个的集合称为**有限集合**，用$|A|<\infty$表

示。成员个数是无限个的集合称为**无限集合**，用$|A|=\infty$表示。

例 1.4　$A=\{0,1,2,\cdots,9\}$　则$|A|=10$

$B=\{x|x=2^n, n=1,2,3,\cdots\}$　则$|B|=\infty$。

集合相等　两个集合相等的充要条件是它们有共同的成员。

例 1.5　$\{0,1,\cdots,9\}=\{9,8,\cdots,1,0\}$

$\{0,1\}=\{x|x^2-x=0\}=\{x|x$为一位二进制数字$\}$

$\{1,2,3\}=\{3,2,1\}=\{1,3,2,2\}\neq\{1,2,4\}$

注意：(1) 集合相等不是指集合的成员个数相等，也不要求其顺序相同，也不要求分类特性相同。

(2) 用第二种表示法$\{x|P(x)\}$比用第一种表示法的信息量多一些。

空集　没有成员的集合称为**空集合**，或**空集**。用ϕ或$\{\}$表示

例 1.6　$\{x|x<1$ 且 $x>2\}=\phi$　$\{x|x\neq x\}=\phi$

注意：(1) $\{0\}\neq\phi$　因有成员为 0 的集合不是空集合。

(2) $\phi\neq 0$　因空集是一个集合，不是数值 0。

(3) $\{\phi\}\neq\phi$　集合中有成员（即空集合）存在，故不是空集。（此处的成员是一个集合，已有类的概念了。）

类　集合的类是一个其成员本身也是集合的集合。即集合的集合也是集合，称为**集合的类**，简称为**类**。用斜体字母表示类。由于类也是集合，所以有时也不特称为类，而统称为集合。

例 1.7　$A=\{\{0\},\{1,2\},\{3,4,5\},\cdots\}$

$B=\{A|A=\{x|x$ 是某个英文单词中的字母$\}$

注意，此处$B\neq\{A,B,C,\cdots,X,Y,Z\}$。而且$\{x,y,w\}\notin B$，因为$x,y,w$组合不成一个英文单词。不过，单词 *act*，*cat*，*tact* 都给出了同一集合$\{a,c,t\}\in B$。

1.2.2　蕴含

蕴含　集合 A **蕴含于**集合 B 的充要条件是集合 A 的每个成员都是集合 B 的成员。记为$A\subseteq B$。

如果集合 A 蕴含于集合 B 的话，也可称为集合 B **蕴含着**集合 A。记为$B\supseteq A$。因而$A\subseteq B$与$B\supseteq A$是等价的。

子集　若$A\subseteq B$，则称 A 是 B 的**子集合**（或称**子集**）。称 B 是 A 的**上界集合**（或**超集合**）。

若$A\subseteq B$且$A\neq B$，则称 A 是 B 的**真子集**。记为$A\subset B$。

若$A\subseteq B$不成立，则记为$A\not\subseteq B$。

全称　为了简化叙述，常使用全称符号$\forall$。$\forall$表示“全体”、“任一个”、“所有的”或“每一个”。

例如，“对于集合 A 的每一个成员，也都是 B 的成员”就可简述为“$\forall x\in A$，有$x\in B$”。

例 1.8　定理“对于所有的集合 A 而言，空集都是 A 的子集”就可简述为“$\forall A$，则$\phi\subseteq A$”。

存在　为了简化叙述，常使用存在符号$\exists$。$\exists$表示“存在着”、“有一个”、“存在着一个”或“可找到一个”。例如，“对于任何非空集合 A，总可找到一个成员属于 A”就可简述为“$\forall A\neq\phi$，则$\exists x\in A$”。

例 1.9　“A 是 B 的真子集的必要条件是在 B 中至少存在着一个成员，该成员不属于 A”就可简述为“A⊂B 的必要条件是 $\exists x \in B$，使得 $x \notin A$。”

定理 1.1　A = B 的充要条件是 A⊆B 及 B⊆A。

证明：必要性是显然的。从略。

证充分性。设 A≠B，则 $\exists x \in A$，使得 $x \notin B$。或者 $\exists x \in B$，使得 $x \notin A$。前者与 A⊆B 有矛盾，后者与 B⊆A 有矛盾。

1.2.3　幂集

集合 A 的所有子集所组成的类称为 A 的**幂集**。记为 2^A 或 $P(A)$。

即　$P(A) \triangleq \{B \mid B \subseteq A\}$

其中，≜表示“根据定义等于”。

例 1.10　若 A = {1,2,3}，则 2^A = {ϕ, {1}, {2}, {3}, {1,2}, {1,3}, {2,3}, {1,2,3}}

定理 1.2　如 $|A| = n < \infty$，则 $|P(A)| = 2^{|A|} = 2^n$

将这个证明留给读者。

1.2.4　集合的运算

并集　也称为**和集**，记为 A∪B。定义如下：

$$A \cup B \triangleq \{x \mid x \in A \text{ 或者 } x \in B\} \tag{1.1}$$

交集　**交集**记为 A∩B，定义如下：

$$A \cap B \triangleq \{x \mid x \in A \text{ 且 } x \in B\} \tag{1.2}$$

若 A∩B≠ϕ，则称 A、B 相交。

若 A∩B = ϕ，则称 A、B 不相交。

直和　**直和**记为 A+B，定义如下：

$$\text{若}\quad A \cap B = \phi,\ \text{则}\ A + B \triangleq A \cup B。 \tag{1.3}$$

差集　记为 A−B，**差集**定义如下：

$$A - B \triangleq \{x \mid x \in A \text{ 且 } x \notin B\} \tag{1.4}$$

对称差　集合 A 与 B 的**对称差**也是一个集合，记为 A⊖B，定义如下：

$$A \ominus B \triangleq (A \cup B) - (A \cap B) = (A - B) \cup (B - A) \tag{1.5}$$

补集　若 $A_i \subseteq A$，则 $A - A_i \subseteq A$。置

$$A_i^c \triangleq A - A_i \tag{1.6}$$

称 A_i^c 为 A_i 的关于 A 的**补集**。有时也用 $\overline{A}_i$ 记之。

显然　$(A_i^c)^c = A_i$

为了更易理解以上运算的定义，在图 1.1 中用图形表示之，其中的阴影部分是运算结果。

例 1.11　设 A = {a,b,c}，B = {c,d,e}，C = {f}，D = {a}，则 A∪B = {a,b,c,d,e}，A∩B = {c}，A+C = {a,b,c,f}，　A−B = {a,b}，

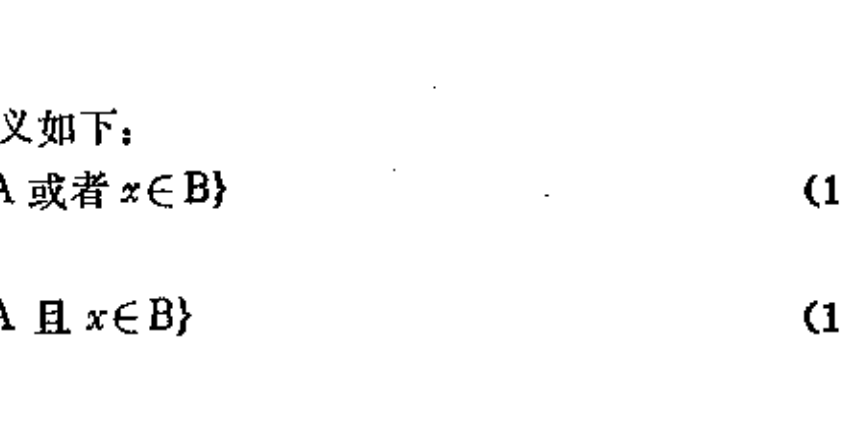


图 1.1　集合运算的图解

$A \ominus B = \{a, b, d, e\}$，$D^c = \{b, c\}$（在此，$D^c$表示D关于A的补集）

1.2.5 全集

如果在所研究的场合中，由所考察的客体而构成的集合全都是某一个固定集合U的子集合，则称该固定集合U为所考察集合的**全集**。

例1.12 在研究的范围中有男、女、老年、青年、壮年、少年、儿童、婴儿、学生、教师、大学生、职员、工人等几类，则全集为“人类”。

定理1.3 令A为集合，U为全集，则 $A \cup A^c = U$，$A \cap A^c = \phi$。

1.2.6 n值运算

对某集合M的任意 n 个成员施以被定义过的任何运算操作，所得的“结果”仍然是原来M的唯一的一个成员，称该运算操作为在M中的**n值运算**。

特别地，$n=2$ 时的运算，称为**2值运算**。

例1.13 $A, B \in P(U)$，A，B对以上的所有运算所得的新的集合也是 $P(U)$ 的唯一的某一成员。因而，以上所定义的运算操作为在 $P(U)$ 中的 n 值运算。

1.2.7 加索引集合

加索引集合 集合A的子集用加索引的下脚标来表示，再将这些下脚标汇集成一个集合I。则称I是**索引集合**，I的成员是**索引**。称集合A的子集所组成的类为**加索引集合**。

用数学形式定义如下：

令 A 是集合 $A_{s_1}, A_{s_2}, A_{s_3}, \cdots$ 的类，其中，如 $s_i = s_j$ 时则 $A_{s_i} = A_{s_j}$。则 A 的成员就可由集合 $I = \{s_1, s_2, s_3, \cdots\}$ 的成员来等价。可写为 $A = \{A_i \mid i \in I\}$。

引入索引集可提供方便。索引集比加索引集要简单些。正象查阅资料一样，查索引号总比查原名称要简单得多。甚至于有时可将某些集合的运算等价于索引集的运算。这就提供更大的方便了。

例1.14 集合 $A = \{a, b\}$ 的四个子集为 $\{\ \}$，$\{a\}$，$\{b\}$，$\{a, b\}$，将它们写为 B_{00}，B_{01}，B_{10}，B_{11}。则 $I = \{00, 01, 10, 11\}$。加索引集写成如下形式：

$$P(A) = \{B_i \mid i \in I\}$$

其中 $I = \{i \mid i$ 是二进制数整数，且 $00 \leqslant i \leqslant 11\}$。

从例中可看出，如果从索引集中找到索引10，则就等价于找到了A的子集 $\{b\}$。

例1.15 已知 $I = \{a, b, c\}$，$B = \{A_i \mid i \in I\}$。将类 B 具体地列出其所有的成员，则为 $B = \{A_a, A_b, A_c\}$。

∪拓广 有了索引集I，可将并运算加以拓广为

$$\bigcup_{i \in I} A_i = \{a \mid a \in A_i, \exists i \in I\} \tag{1.7}$$

∩拓广 有了索引集I，也可将交运算拓广为

$$\bigcap_{i \in I} A_i = \{a \mid a \in A_i, \forall i \in I\} \tag{1.8}$$

例1.16 若 $I = \{1, 2, 3, 4\}$，且 $A_1 = \{a, b\}$，$A_2 = \{a, c\}$，$A_3 = \{a, d\}$，$A_4 = \{a, b, c, e\}$，则

$$\bigcup_{i \in I} A_i = \{a, b, c, d, e\}$$

$$\bigcap_{i \in I} A_i = \{a\}$$

1.2.8 划分

若$A=\{A_i \mid i\in I,\ A_i\neq\phi,\ A_i\subseteq A\}$而且 $\bigcup_{i\in I}A_i=A$，以及$A_i\cap A_j=\phi$ $(\forall\ i,j\in I,\ i\neq j)$，则称$A$是集合 A 的**划分**。

例 1.17　A = {1,2,3,4}，A 的划分有如下几种：

$A_1=\{\{1,2\},\{3,4\}\}$；$A_2=\{\{1\},\{2,4\},\{3\}\}$；$A_3=\{\{1,2,3,4\}\}$；$A_4=\{\{1,4\},\{2,3\}\}$；$A_5=\{\{1,2,3\},\{4\}\}$等等。它们各个都可成为 A 的划分。

1.2.9 集合的代数式

集合运算的一些基本代数式列于后（证明留给读者）：

$$A\cup B=B\cup A \qquad A\cap B=B\cap A \tag{1.9}$$

$$(A\cup B)\cup C=A\cup(B\cup C) \qquad (A\cap B)\cap C=A\cap(B\cap C) \tag{1.10}$$

$$A\cup(B\cap C)=(A\cup B)\cap(A\cup C) \qquad A\cap(B\cup C)=(A\cap B)\cup(A\cap C) \tag{1.11}$$

$$A\cup A=A \qquad A\cap A=A \tag{1.12}$$

$$A\cup\phi=A \qquad A\cap\phi=\phi \tag{1.13}$$

$$A\cup U=U \qquad A\cap U=A \tag{1.14}$$

$$A\cup A^c=U \qquad A\cap A^c=\phi \tag{1.15}$$

$$\phi^c=U \tag{1.16}$$

$$U^c=\phi \tag{1.17}$$

$$(A^c)^c=A \tag{1.18}$$

$$A\cup(A\cap B)=A \qquad A\cap(A\cup B)=A \tag{1.19}$$

$$(A\cup B)^c=A^c\cap B^c \qquad (A\cap B)^c=A^c\cup B^c \tag{1.20}$$

1.2.10 有序偶

集合的成员是不考虑其顺序的，如{2,3}和{3,2}是等价的。但有些场合却需要考虑顺序的。如解析几何中表示点的坐标就是考虑顺序的，(2,3)与(3,2)所定义的点是不同的。

用 (a,b) 来表示**有序偶**。有序偶的成员之前后顺序不可调。a 和 b 的有序偶 (a,b) 可用集合$\{\{a\},\{a,b\}\}$来表示。称 a 是(a,b)的第一坐标，b 是 (a,b) 的第二坐标。

可见，$(a,b)=\{\{a\},\{a,b\}\}=\{\{a,b\},\{a\}\}=\{\{b,a\},\{a\}\}$。但是 $(a,b)\neq\{\{b\},\{a,b\}\}$（除 $a=b$ 而外）。

有序偶相等可如此定义。$(a_1,b_1)=(a_2,b_2)$的充要条件是 $a_1=a_2$，$b_1=b_2$。

例 1.18　有序偶$\{\{a\},\{a,b\}\}$与$\{\{c\},\{d,c\}\}$相等，则 $a=c$，$b=d$。

例 1.19　$(a,a)=\{\{a\},\{a,a\}\}=\{\{a\},\{a\}\}=\{\{a\}\}$。注意，$\{\{a\}\}\neq\{a\}$。

无序偶则不考虑其顺序，记为$\langle a,b\rangle$。$\langle a,b\rangle=\langle b,a\rangle$。

1.2.11 笛卡尔积

笛卡尔积又称**直积**，记为 A×B。集合 A 和 B 的笛卡尔积定义如下：

$$A\times B\triangleq\{(a,b)\mid a\in A,b\in B\} \tag{1.21}$$

式 (1.21) 是有序偶的笛卡尔积。

无序偶的笛卡尔积，记为 A△B，定义如下：

$$A\triangle B\triangleq\{\langle a,b\rangle\mid a\in A,b\in B\} \tag{1.22}$$

例 1.20　若 A = {1,2}，B = {a,b,c}，则

$$A\times B=\{(1,a),(1,b),(1,c),(2,a),(2,b),(2,c)\}$$

$$A\triangle B=\{\langle 1,a\rangle,\langle 1,b\rangle,\langle 1,c\rangle,\langle 2,a\rangle,\langle 2,b\rangle,\langle 2,c\rangle\}$$

例 1.21 若 $A=\{x,y,z\}$，则

$$A\times A=\{(x,x),(x,y),(x,z),(y,x),(y,y),(y,z),(z,x),(z,y),(z,z)\}$$

$$A\triangle A=\{\langle x,x\rangle,\langle x,y\rangle,\langle x,z\rangle,\langle y,y\rangle,\langle y,z\rangle,\langle z,z\rangle\}$$

可见，如$|A|=n$，则$|A\times A|=n^2$，$|A\triangle A|=n(n+1)/2$。

n 个集合的直积 n 个集合的笛卡尔积，也称为 **n 个集合的直积**，是一个 n 元组。n 个集合 A_1，A_2，A_3，…，A_n 之间的直积定义如下：

$$A_1\times A_2\times\cdots\times A_n\triangleq\{(a_1,a_2,\cdots,a_n)\mid a_1\in A_1,a_2\in A_2,\cdots,a_n\in A_n\} \tag{1.23}$$

例 1.22 若$A=\{a,b\}$，则 $A\times A\times A=\{(a,a,a),(a,a,b),(a,b,a),(a,b,b),(b,a,a),(b,a,b),(b,b,a),(b,b,b)\}$

其中 $A\times A\times A$ 可简写为 A^3。若 $A_1=A_2=\cdots=A_n=A$，则 $A_1\times A_2\times\cdots\times A_n$ 可简写为 A^n。

1.2.12 对应与映照

对应与映照的概念，在数学方面仍是与集合论相并列的最基本的概念。不管是图论，还是智能模拟，都必须具备集合论、对应与映照的概念。

对应 对于二个集合 A 和 B，根据某规则 Γ，由 A 的各成员 $(\forall a\in A)$ 分别确定 B 的子集 $\Gamma(a)$时，则称这个规则 Γ 为从 A 至 B 的**对应**。称对于 $a\in A$ 所确定的 $\Gamma(a)$ $(\subseteq B)$ 为 a 关于 Γ 的**象**。这时 A、B 分别称为对应 Γ 的**始集合**和**终集合**。图 1.2 可以形象化地表现对应的概念。

对应的符号表示如下：

$$\Gamma:A\to B \tag{1.24}$$

注意，对于对应 $\Gamma:A\to B$ 而言，即使 a，$b(a\neq b)\in A$，$\Gamma(a)=\Gamma(b)$ 却可能存在，而且使得 $\Gamma(a)=\phi$ 的 $a(\in A)$也是存在的。

若从 A 至 B 的两个对应 Γ_1 及 Γ_2，对于任一 $a\in A$，$\Gamma_1(a)=\Gamma_2(a)$ 都成立，称 Γ_1 与 Γ_2 **相等**。记为 $\Gamma_1=\Gamma_2$。

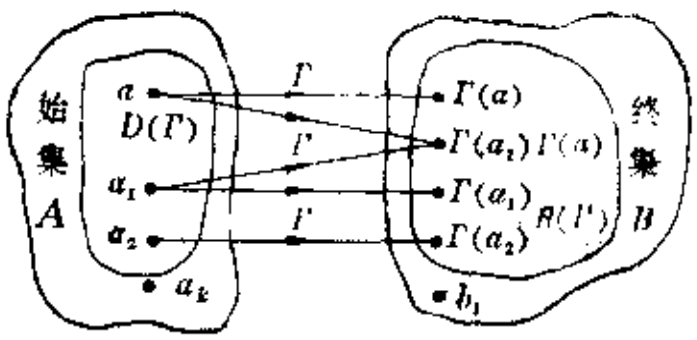


图 1.2 集合之间的对应

对于对应 $\Gamma:A\to B$ 而言，由

$$D(\Gamma)\triangleq\{a\mid a\in A,\ \Gamma(a)\neq\phi\} \tag{1.25}$$

$$R(\Gamma)\triangleq\{b\mid b\in\Gamma(a),\ a\in D(\Gamma)\} \tag{1.26}$$

所定义的集合 $D(\Gamma)(\subseteq A)$及 $R(\Gamma)(\subseteq B)$分别称为 Γ 的**定义域**和**值域**。可参看图 1.2。

显然，如果 $\Gamma(a)=\phi$ 的 $a\in A$ 不存在的话，即对任一 $a\in A$，在 B 中都有其象存在（即 $\Gamma(a)\subseteq B$），则 Γ 的始集 A 就是其定义域 $D(\Gamma)$。即 A 与 $D(\Gamma)$是一致的。

逆对应 在对应 $\Gamma:A\to B$ 中，如果关于每个 $b\in B$ 的集合 $\Gamma'(b)$由式

$$\Gamma'(b)\triangleq\{a\mid b\in\Gamma(a)\} \tag{1.27}$$

所定义的话，则确定了对应 $\Gamma':B\to A$。这个对应 Γ' 称为 Γ 的**逆对应**。记为 Γ^{-1}。还有，

$b\in B$ 关于 Γ^{-1}的象 $\Gamma^{-1}(b)$，称为 b 关于 Γ 的**原象或逆象**。

从该定义可见，由于有 $b\in\Gamma(a)$的限制（当然 $b\in R(\Gamma)$），所以 $\Gamma^{-1}(b)\neq\phi$。

映照 在对应 $\Gamma:A\to B$ 中，如果对于任一 $a\in A$，$\Gamma(a)$仅是由 B 的唯一的成员组成的话，则称这个对应 Γ 为从 A 至 B 的**映照**。

映照通常用小写拉丁字母 f，g 等表示。即 $f:A\to B$。

从定义可知，在 $f:A\to B$ 中，不仅对任一 $a\in A$，$|f(a)|=1$，而且 $A=D(f)$。但是 $B=R(f)$不一定成立。

由于$|f(a)|=1$，所以简写为 $f(a)=b$。

例 1.23 对于实数域 R，若有一个对应 $\exp(x)$ $(x\in R)$ 的话，则这个对应是从 R 至其自身的映照。如用 f 表示，则对于任一 $x\in R$，写为 $f(x)=\exp(x)$。

满照 在 $f:A\to B$ 中，如果 $B=R(f)$，即值域与终集一致的话，称这个 f 为**满照**。也称 f 为**从 A 至 B 上的映照**。

$R(f)\subseteq B$ 且 $B\neq R(f)$时，就称 $f:A\to B$ 为**从 A 至 B 中的映照**。

一对一映照 在映照 $f:A\to B$ 中，对于任意的 a,a'，如果

$$a\neq a'\Rightarrow f(a)\neq f(a') \tag{1.28}$$

时（其中记号⇒表示“可导出”之意），则称 f 为从 A 至 B 的**一对一映照**。

一一映照 既是满照又是一对一映照的映照称为**一一映照**。

例 1.24 从实数域R至自身的映照 f_1，f_2，f_3 由如下定义的话，

$$f_1(x)\triangleq x-1,\quad f_2(x)\triangleq\sin x,\quad f_3(x)\triangleq x^3-x^2$$

则 f_1 是从 R 至 R_x^-的一一映照；f_2 既不是满照又不是一对一映照；f_3 是满照但不是一对一映照。

逆映照 一一映照 f 的逆对应 f^{-1} 称为**逆映照**

一一映照与逆映照的关系可用图 1.3 形象地表达。

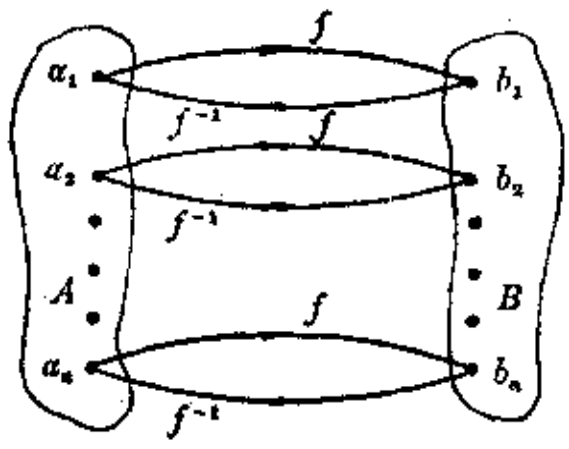


图 1.3 一一映照与逆映照

例 1.25 例 1.24 中的一一映照 $f_1(x)\triangleq x-1$ 之逆对应 $f_1^{-1}(y)\triangleq y+1$ 就是 f_1 的逆映照。

从定义可知，逆映照也符合“既是满照又是一对一映照”的条件的。因而，逆映照本身也是个一一映照。

其逆映照的逆映照就是一一映照它本身。即

$$(f^{-1})^{-1}=f \tag{1.29}$$

复合 设一一映照 $f:A\to B$，一一映照 $g:B\to C$。如果对任一 $a\in A$，先关于 f 映照至 $B(f(a)\in B)$，再关于 g 映照至$C(g(f(a))\in C)$，则可找到一个从A 至 C 的一一映照 $h:A\to C$，使得 $h(a)=g(f(a))\in C$。就称 h 是 f 与 g 的**复合**。用下式来表达这个复合映照 h。（参阅图 1.4，即可理解之）。

$$h=g\circ f \tag{1.30}$$

映照的复合是符合结合律的。即

$$f\circ(g\circ h)=(f\circ g)\circ h \tag{1.31}$$

通常情况下，一一映照的复合是不可交换的。即 $g\circ f\neq f\circ g$

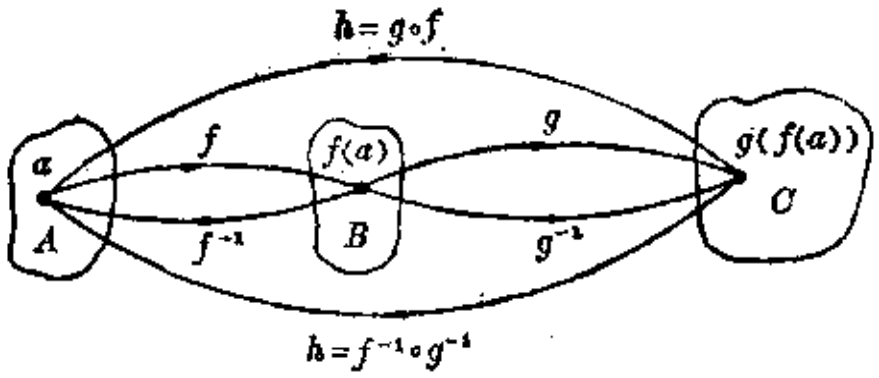


图 1.4 一一映照的复合及复合映照的逆映照

对于 $f:A\to B$，$f^{-1}\circ f=I_A$；对于 $f^{-1}:B\to A$，$f\circ f^{-1}=I_B$。称 I_A，I_B 为**恒等映照**。特别地，如果 $f:A\to A$ 的话，则 $I_A=I_B$。即

$$f^{-1}\circ f=f\circ f^{-1}=I \tag{1.32}$$

恒等映照有如下特性。

$$I\circ f=f\circ I=f \tag{1.33}$$

$$I_B\circ f=f \tag{1.34}$$

$$f\circ I_A=f \tag{1.35}$$

复合映照的逆映照可由各映照的逆映照按下式求得：

$$(g\circ f)^{-1}=f^{-1}\circ g^{-1} \tag{1.36}$$

参阅图 1.4 就可理解式（1.36）了。

1.2.13 可列集

如果集合 A 能与自然数集合 N 成一一映照关系，则该集合 A 称为**可列集**。

例 1.26 正偶数的集合 $\{2n\}$ 是可列集。因为它可与自然数集合 $N=\{n\}$ 成一一映照关系。即可找到一个映照 $f:\{2n\}\to\{n\}$。$f(2n)=\dfrac{2n}{2}=n$。

1.3 图论的基本概念

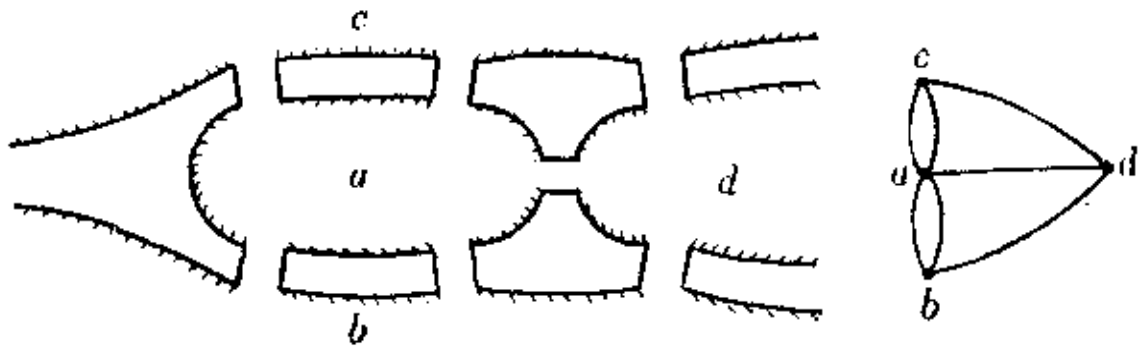
1.3.1 图

图论是一门拓朴几何学。它把所研究的某些客体加以抽象而用几何的点来表示。而某两客体之间的关系则用连结着对应于该两客体的两点之曲线来表示。

例 1.27 数学家 L.Euler 在 1736 年提出著名的“七桥问题”，从而成为图论的发明者。“七桥问题”如下：当时东普鲁士的 Pregel 河中有两个岛屿（如图 1.5 (*a*) 所示）。*a*、*d* 岛有七个桥架着。能否从 *a*,*b*,*c*,*d* 的某地出发，走过所有的桥，但每桥只能走过一次。Euler 就将陆地抽象为几何学上的点，而连通两陆地的桥抽象为点之间的连线。参见图 1.5 (*b*)。从而归结为“一笔划”问题。

在欧几里得空间中，连接着某两点的线段称为**曲线**。若曲线的两端点相同，称该曲线为**闭曲线**。若两端点不相同的曲线，称为**开曲线**。利用这些几何学概念，就可以定义图了。

图 所谓图 G 是由满足以下条件的欧几里得空间的点的集合 $V(\neq\phi)$ 和曲线的集合 E 所组成：



(a) (b)

图 1.5 七桥问题

(1) 属于 E 的任何闭曲线含有一个且仅仅一个属于 V 的点。

(2) 属于 E 的任何开曲线恰恰含有二个属于 V 的点，并且这二个点就是开曲线的端点。

(3) 属于 E 的任何曲线自身不相交。

(4) 属于 E 的任何不同的二曲线，若有公共的点，则它必属于 V。

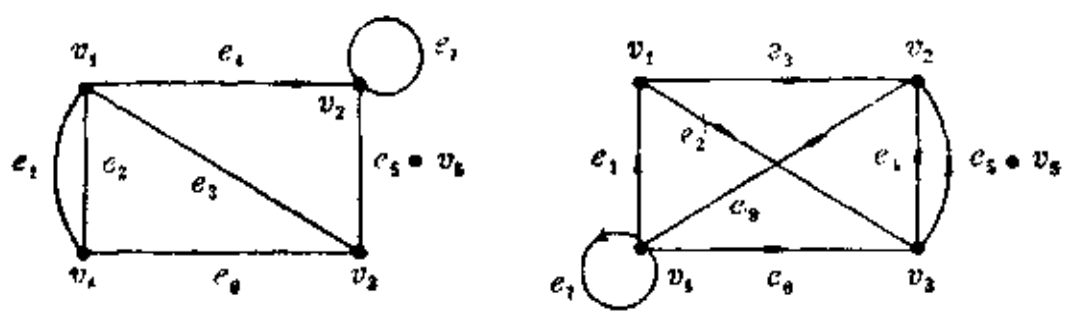
这时，用二元组来表示图 G，即 G = (V, E)。称属于 V 的各个点为 G 的**顶点**或**节点**。称属于 E 的各曲线为 G 的**边**或**棱**。没有任何边连结的节点，称为**孤立节点**。G 的开曲线和闭曲线分别称为**开边**和**闭边**。

如果 G 中的边有指定连结方向，即边上付与由它的某节点至它的另一节点的连结方向时，称该边为**有向边**。用箭头付与边上来表示连结方向。不指定连结方向的边，称为**无向边**。

有向图 图 G 的边全都是有向边，则称该图为**有向图**。

无向图 图 G 的边全都是无向边，则称该图为**无向图**。

例 1.28 图 1.6 (*a*)是一个无向图，其中 v_5 是孤立节点，e_7 是闭边。$e_1 \sim e_6$ 都是开边。图 1.6 (*b*)是一个有向图。其中 v_5 是孤立节点，e_7 是闭边，$e_1 \sim e_6$ 及 e_8 是开边。注意，



(a) (b)

图 1.6 无向图与有向图

e_2 和 e_6 并不意味着相交，因为没有用节点表示出它们相交。

无向图和有向图也可以给予形式定义。

所谓无向图，就是由非空的节点集合 V、和 V 有关的边的集合 E 以及映照 $\Phi: E \to V \triangle V$ 等组成的三元组 $G = (V, E, \Phi)$。其中 $V \triangle V$ 的意义可参阅式 (1.22)，由于它表示了边是无序偶，所以就表达了无向图的意义。

所谓有向图，就是由非空的节点集合 V、和 V 有关的边的集合 E 以及映照 $\Psi: E \to V \times V$ 等组成的三元组 $G = (V, E, \Psi)$。其中 $V \times V$ 的意义可参阅式 (1.21)，由于它表示了边

是有序偶，所以就表达了有向图的意义。即对于 $e_k \in E$，若 $\Psi(e_k) = (v_i, v_j)$ 时，有向边 e_k 就把节点 v_i 作为始点，把节点 v_j 作为终点。或者称 e_k 为从 v_i 出发而进入 v_j。也可简单地称 v_i，v_j 为 e_k 的**端点**。

路 如果从某节点出发，通过若干条边而到达另一节点时，则称该两节点之间有**路**。若路由 l 条边组成，则称该路的**路长**为 l。

例 1.29 图 1.6 (a)中 v_1 至 v_4 有路(e_1)，(e_2)，(e_3, e_6)及(e_4, e_5, e_6)等四条路。路长分别是 1，1，2，3。而 v_1 至 v_5 则没有路。

回路 如果从某节点出发，通过若干条边又回到该节点时，称由这些边组成的路为**回路**，或**闭路**，，或**环路**。否则，称为**开路**。

闭边则称为**自回路**。

例 1.30 图 1.6 (a)中(e_1, e_2)，(e_2, e_6, e_3)，(e_3, e_5, e_4)，(e_2, e_6, e_5, e_4)等都是回路。而 e_7 为自回路。

伴随无向图 将有向图的边去除方向之后而得到的无向图，称为该有向图的**伴随无向图**。

如果有向图的某两节点在其伴随无向图中有路存在的话，则称该两节点在有向图中也有路存在，称该路为**有向路**。

如果其伴随无向图有回路存在的话，则在有向图中对应的回路，称为**有向回路**。

如果节点 v_i 至节点 v_k 的有向路中各边的方向都一致，而且 v_i 是该路中第一条边的始点，而 v_k 是该路中最后一条边的终点的话，称该路为**顺向路**。

例 1.31 图 1.7(b)是图 1.7(a)的伴随无向图。由于图 1.7(b)中 v_5 至各节点都没有路存在，故图 1.7 (a)的有向图中 v_5 至各节点也没有有向路存在。而且图 1.7(a)中 v_1 至 v_4 有顺向路 (e_1, e_2, e_3) 存在。还有有向回路 (e_1, e_2, e_4) 存在。

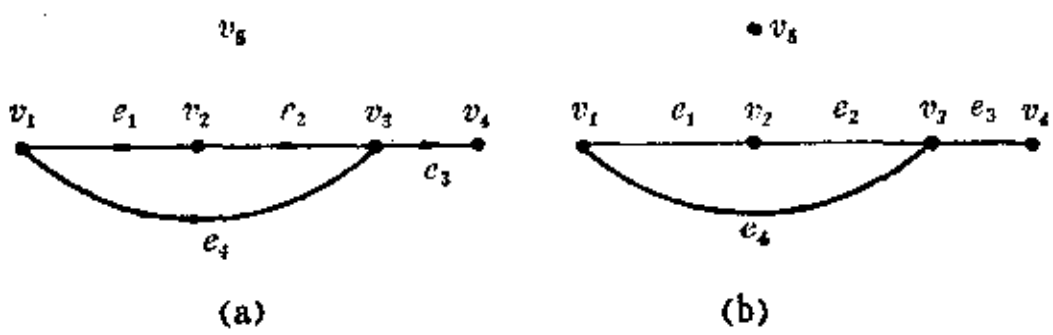


图 1.7 有向图及其伴随无向图

先驱节点与后裔节点 在有向图中如果从某节点 v_i 至另一节点 v_j 有顺向路存在，而 v_j 至 v_i 却没有顺向路存在的话，则称 v_i 为 v_j 的**先驱节点**，称 v_j 为 v_i 的**后裔节点**。如果其路长为 1 的话，则称 v_i 为 v_j 的**父节点**，v_j 为 v_i 的**子节点**，等等。

子图 如果某一图的所有边和节点都是图 G 的边和节点的话，则称该图为图 G 的**子图**。

真子图 如果在图 G 中至少存在着一个节点或一条边，它不属于图 G 的子图的话，则称该子图是图 G 的**真子图**。

1.3.2 树

树 在某无向图的某一子图中，如果各节点之间都有路存在，而且不存在任何回路的

话，则称该子图是该无向图的**树**。

可见，树中的任两节点之间必有且仅有一条路存在。

有向图的树是其伴随无向图的树所对应的有向子图。

跨树 如果某图的所有节点都在该图的某树上，则称该树为关于该图的**跨树**。可见，含有弧立节点的图没有跨树存在。

有向图的跨树是其伴随无向图的跨树所对应的有向子图。

树上的边称为**树枝**或**枝**。显然，同一图的树不是唯一的。

例 1.32 图 1.8(*a*)中粗线边所示的是树，但没有跨树存在。图 1.8(*b*) 中粗线边所示的是跨树。

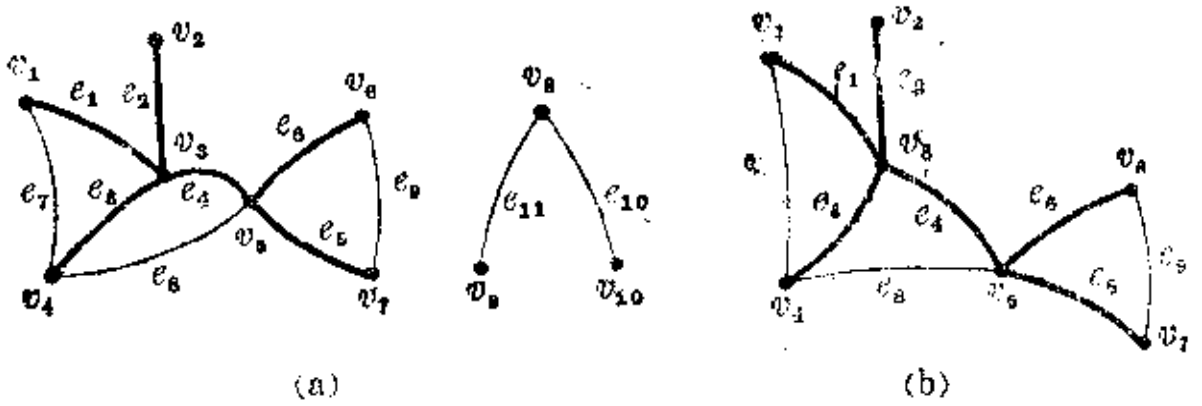


图 1.8 树与跨树

例 1.33 图 1.9 (*a*)中的跨树就有如图 1.9 (*b*)中实线边所示的八种。

从图 1.9 中可见，同一图中的任何跨树，其树枝数目都是相同的。即如果有跨树存在的话，则跨树的树枝数目等于节点数减 1 。

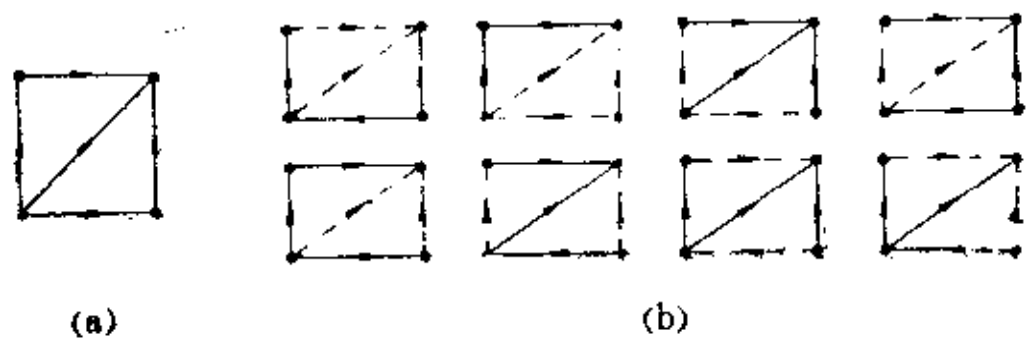


图 1.9 跨树不是唯一的例子

图中除了某一跨树的树枝之外所有的边组成的子图，称为该图关于这跨树的**补图**。如图 1.8(*b*)的细线边及图 1.9(*b*)中的虚线边都分别组成补图。补图的边称为**补边**或**弦**。

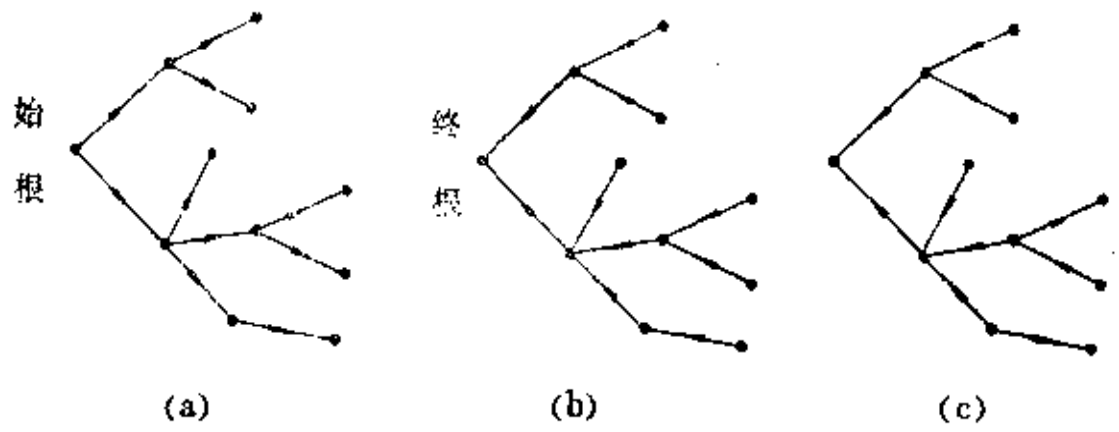
显然，在跨树上加上任一补边，就会形成一个回路。

外向树 在有向树中，如果有某一个节点存在，使得所有其他的节点都是该节点的后裔节点的话，则称该有向树为**外向树**；称该节点为该外向树的**始根**。

内向树 在有向树中，如果有某一个节点存在，使得所有其他的节点都是该节点的先辈节点的话，则称该有向树为**内向树**；称该节点为该内向树的**终根**。

始根与终根统称为**树根**。内向树与外向树统称为**根树**。

图 1.10(*a*)是一棵外向树，图 1.10(*b*)是一棵内向树。图 1.10(*c*)既不是外向树 也 不 是内向树。



(a) (b) (c)

图 1.10 外向树、内向树和有向树

外向树中，没有后裔节点的节点，称为外向树的端节点。
内向树中没有先辈节点的节点，称为内向树的端节点。
有时把端节点称为**树叶**，而把端节点及其连着的枝称为**树梢**。

1.3.3 耗散图

在有些实际问题中，单用边来表示客体间的关系，有时显得不足。常在边上加注有**耗散值（或称权、流量**等）来表达。甚至于在边上加注某些条件来表示也可。

例 1.34 为了表示城市之间的交通关系及里程数，除了用节点表示城市，用边表示两城市之间的交通线外，还在边上加上里程数字，这些数值就称为其对应边的**耗散**。如图 1.11 所示。

同理，如果不以里程数而以票价来考虑的话，则耗散值就相应改为票价。

可见，“在若干城市中从某城出发不重复地游览各个城市又回到原城，而旅费又要最少”这样一个导游问题，就可归结为另一个图论问题：“寻找一个子图，其每节点都有而且仅有二条边。并且各边的耗散之总和最小”。

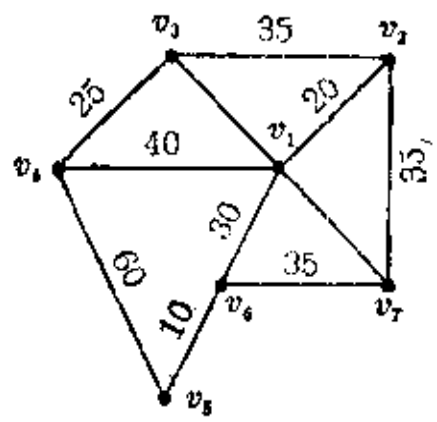


图 1.11 有耗散的图

对于树、有向图、有向树等等也可加注耗散值。

1.3.4 图的数据结构

如何把图的结构整个地存入计算机中，很关键的问题就是如何把图中的节点以及边用数据方式存入计算机，而且存入的数据与图的整个结构是一一对应的。现在就简要介绍一下有关图的数据结构的一些最基本的概念。

图的节点集合中各个节点都具有该集合中规定的某些特征。各项特征由特征的名加以区分。不同的成员，其特征不尽相同，由此而区别各成员。例如在学生的集合中，图的各节点就是学生。其成员的特征可规定为学号、姓名，性别、年龄、籍贯、身高、体重、学业成绩等等。所有这些特征都用数据形式来表示。每个成员都有具体的特征值表示它。一个特征值就是一个**信息组**。各成员的数据就是由这些规定的信息组组成的，称为**记录**。记录的组合就称为**文件**。可见，文件是和集合相对应的。

多个文件的组合称为**文件库**。文件库中部分文件组成的新文件库称为**子文件库**。由一个

文件中部分记录组成的新文件称为**子文件**。类似地可定义**子记录**和**子信息组**。

记录内的信息组当中，常常要有一个信息组，用它来识别这个成员的，称这个信息组为这记录的**主信息**，或称**主信息组**，或称为**键**。例如，上述的学生集合中，记录内的信息组当中，不仅姓别、年龄、籍贯…等不能区别出学生成员，而且姓名这信息组也不能唯一区别各学生成员，因为同名同姓的学生还是不少的。但学号这个信息组却是可以唯一地区别出各学生的。因而，学号是主信息组。

文件存放在计算机的内存或外存里。存放记录的单位称为**单元**。每个单元都有一个地址。如果单元是由若干机器字组成，则其第一个机器字的地址就是**单元的地址**。如果单元是由若干字节组成，则其第一个字节的地址就是单元的地址。

例 1.35 学生集合中的某一记录如下：

学号　　姓名　　姓别　　年龄　　籍贯

790532#CHEN MIN# NAN # 19 #SHANG HAI #……

从例 1.35 可看出，学号，姓别，年龄等信息组的数据长度是固定的，而姓名、籍贯等信息组的数据长度是不固定的。数据长度固定的信息组称为**定长信息组**；数据长度不固定的信息组称为**不定长信息组**。

信息组之间可用分隔符#把它们分隔开。如果记录中都是定长信息组，则分隔符可以省略。记录也有定长与不定长之分。

内存存放文件的存区，称为**表**。表中不存记录的单元称为**空单元**。没有空单元的表称为**稠密表**；反之称为**松散表**。全是空单元的表称为**空表**。表的上半是稠密部分、下半是空区，这样的表称为**半稠密表**。

在记录的信息可排序的情况下，如表内单元的位置顺序和主信息组的数值（称为**键值**）大小顺序相一致时，称该表为**有序表**。

例 1.36 除了具有且仅具有一个始根（或终根）和一个树叶外，其它节点都具有且仅具有二个树枝，这样的树称为**单值树**或**单叉树**。以此类推，可定义**二值树**、**三值树**，**多值树**。图 1.12(*a*)所示的单值树中节点的先辈、后裔关系，就可定义为节点间的顺序关系。因而可存入如图 1.12(*b*)所示的有序表中。表 L 中第 $i+1$ 个单元所对应的节点恰恰是第 i 个单元所对应的节点的子节点。

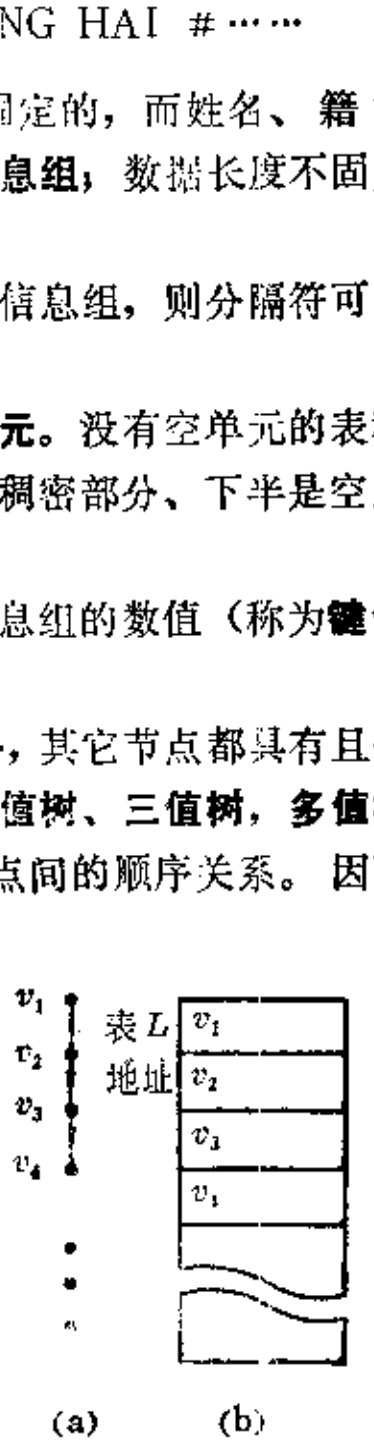


图 1.12　单值树及其有序表

按记录出现的顺序，依次地存放的表，称为**顺序表**。顺序表是随机表的一种。

如果欲将新出现的记录存入顺序表中，那是很方便的，如图 1.13(*a*)所示。如果欲将它存入有序表中，则先要根据记录的顺序关系找出其表的所在位置，而后下推其后的记录，再嵌入之，如图 1.13(*b*)所示。图 1.13(*c*)所示的就是删除某记录的过程。一般情况下，每嵌入或删除某一记录，有序表中其后部分都要下推或上移一次。为了克服这一缺点，就引进一种勾连表。

勾连表　如果在表的每个记录内增设**指针信息组**(简称为**指针**)的话，就可用指针来表示

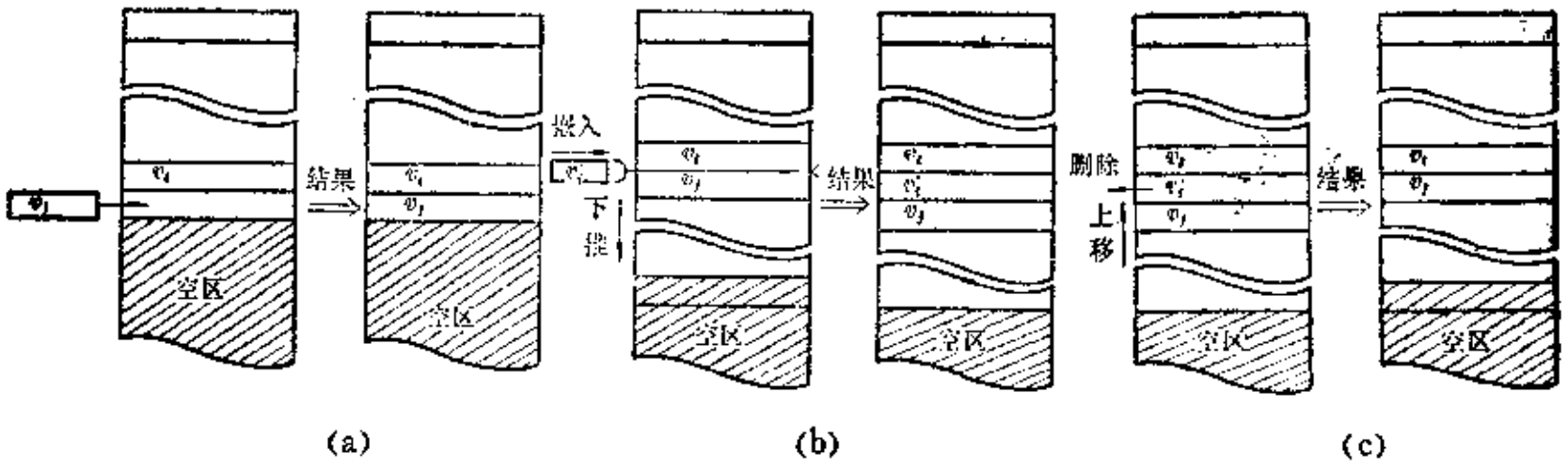


图 1.13　顺序表的存入及有序表的嵌入和删除

该记录的下一个记录的所在地址。在表内存放这些记录的单元由指针按序勾连起来，代表原来的记录顺序，这种表称为**勾连表**。

勾连表中的单元如图 1.14(*a*)所示，简化为图 1.14(*b*)。

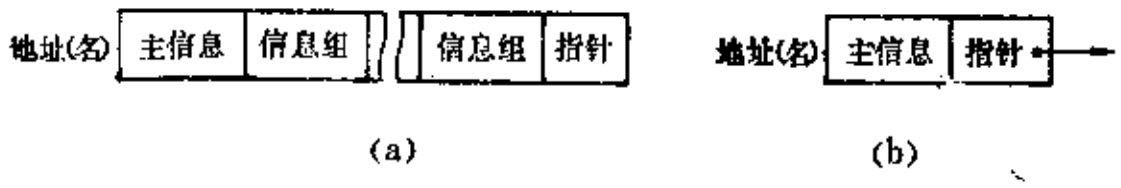


图 1.14　勾连表的单元

每个勾连表都有一个表头（不属于表区内），指示表区上存放第一个记录的单元地址。表上最后一个记录的单元之指针部分为空，今后用星号 * 表示，它表示终止而不指向任何单元，称为**终止指针**。如图 1.15 所示。

将空单元勾连起来而成一个空区，称为**空区勾连表**。如图 1.15 所示。其中 *f* 为文件表，*s* 为空区。

在勾连表中要嵌入新出现的记录或删除某一记录就较方便了。这个操作过程留给读者练习。

以上所述的勾连表都是单勾连表，因为只有一个指针。查到某记录时，只知道其后续记录，而不知其前接的记录，有时带来不便。为此，在每个单元中再增加一个指针。它指向该记录的前接记录。称它为**向后指针**或**返回指针**（原来的指针称为**向前指针**）。这种勾连表称为**双勾连表**。

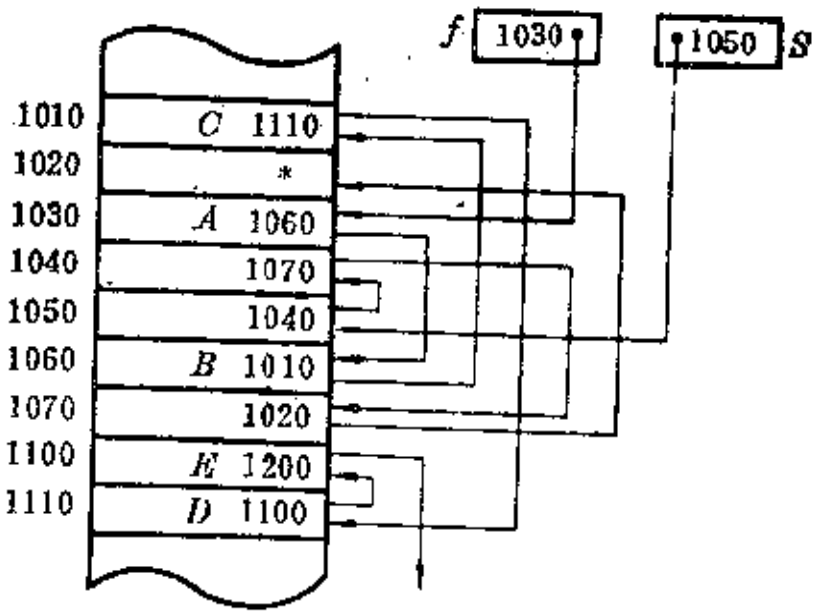


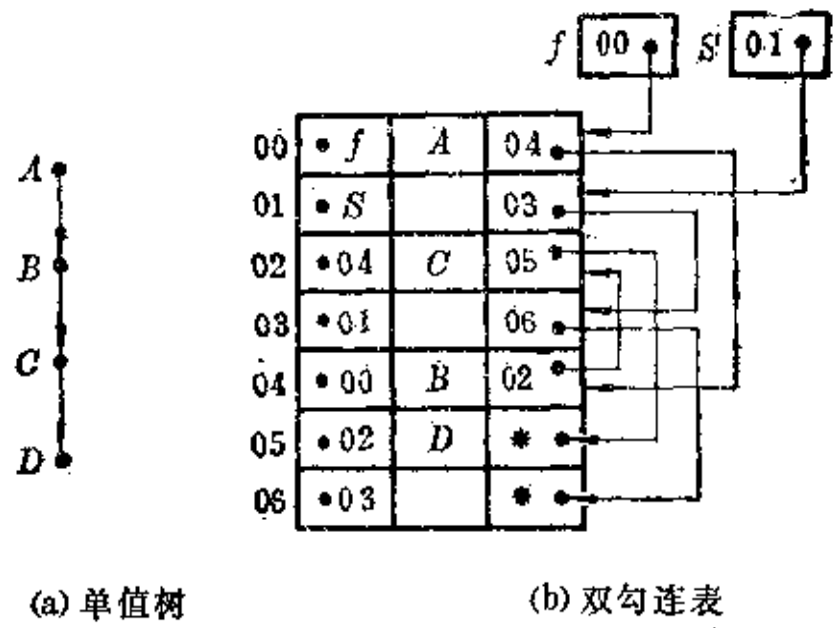
图 1.15　勾连表的一例

有了双勾连表，欲查找单值树中任一节点及其父节点和子节点，就很方便了。

图 1.16 示出单值树在内存里双勾连表的文件 *f*。

指针场 可以根据需要增设向前指针或向后指针，也可以取消向前指针或向后指针。记录中有多个指针的话，有时称它为**指针场**。

例 1.37 二值外向树的数据结构，可以将向后指针改为向前指针。有了二个向前指针就很方便地表示图 1.17(*a*)的二值树了，如图 1.17(*b*)所示。其中，为了形象化起见，将表的单元不按地址的顺序排列，而用图的形式排列。。



(a) 单值树　　(b) 双勾连表

图 1.16 单值树在双勾连表的文件

对于无向树而言，其指针没有向前、向后之意。

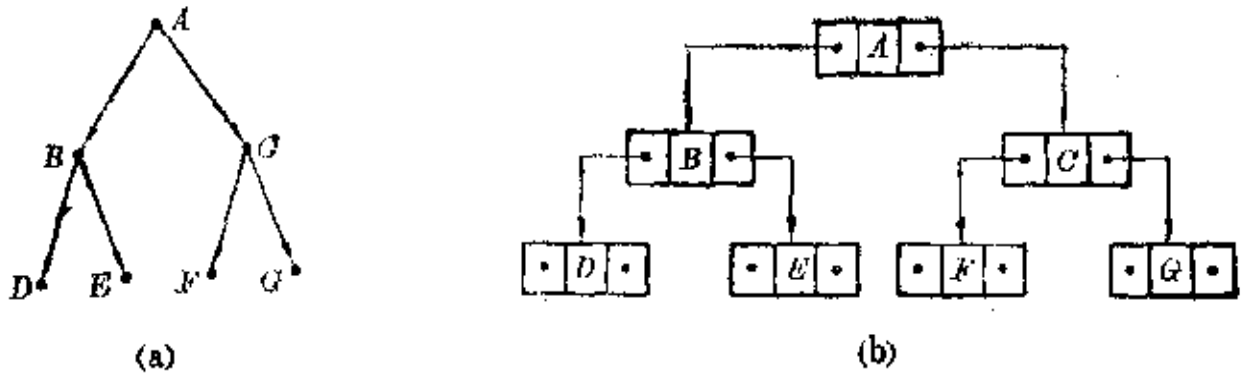
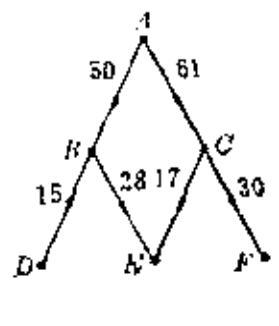


图 1.17 二值树及其勾连表

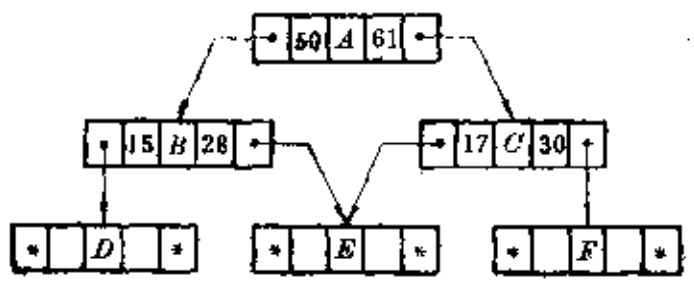
对于一般的图而言，也是类似于有向树的结构，读者是容易推得的。

耗散图 对于耗散图（或耗散树）而言，除了需要指针外，还需在每个单元中增加信息组来表示耗散值。每个指针增加一个耗散信息组，如图 1.18(*a*)所示。图 1.18(*b*)所示的耗



(a)　　(b)

指针	耗散信息组	主信息	耗散信息组	指针



(c)

图 1.18 耗散记录以及耗散图及其数据结构

散图之数据结构如图 1.18 (c)所示。

如果在耗散图中需增加节点或删除节点或改变节点之间的关系，则除了改变相应的指针外，还需改变其相应的耗散信息组。

数据结构还有许多方法，如目录法和映照法等。就是上述的勾连表法，也有许多内容，如装入、删除、维护等等。此外，还有表如果溢出应如何处理等等许多问题。在此不一一介绍。

第一章 习 题

一、下列集合中有哪些集合是空集合？

$A = \{x | x > 2 \text{ 且 } x < 2\}$

$B = \{x | x + 4 = 4\}$

$C = \{\phi\}$

二、下列结论是否对？试举例说明之。

(1) 如果 $A \cup B = A \cup C$，则 $B = C$。

(2) 如果 $A \cap B = A \cap C$，则 $B = C$。

(3) 如果 $A - B = A - C$，则 $B = C$。

三、在下列集合中，试找出相等的集合。试找出有蕴含关系的集合，并指出其中哪个集合是另一个集合的真子集。

$A = \{x | x \text{ 为小于 } 4 \text{ 的正整数}\}$

$B = \{x | x \text{ 为小于 } 5 \text{ 之质数}\}$

$C = \{1, 3\}$

$D = \{x | x^2 = 4\}$

$E = \{1, 2, 3, 1\}$

四、分别满足下列条件的集合 P 和 Q 有什么关系？

(1) $P \cap Q = P$　　(2) $P \cup Q = P$

(3) $P \ominus Q = P$　　(4) $P \cap Q = P \cup Q$

(5) $P - Q = Q - P$

五、设 A、B、C 是集合。试问，分别在什么情况下下式成立？

(1) $(A - B) \cup (A - C) = A$

(2) $(A - B) \cup (A - C) = \phi$

六、在四年级的 100 个学生中，懂英语的有 69 人，懂日语的有 55 人，懂德语的有 27 人，懂英日语的有 31 人，懂英德语的有 18 人，懂日德语的有 14 人，英日德语都懂的有 10 人。试问，仅懂其中一门外语的有多少人？这三门外语都不懂的有多少人？

七、在学校的学生登记卡中，每个在校学生都有且仅有一张登记卡片。卡片上登录有学号、姓名、性别等。如果将卡片组成一个集合 A，将在校学生组成一个集合 B，将各在校学生的姓名组成一个集合 C，将其性别组成一个集合 D。试从 ABCD 四个集合中找出分别具有对应、映照、满照、一对一映照和一一映照等关系的某二个集合。

八、试证明定理 1.2 。

九、试对图 1.19 的二个图，分别回答如下几个问题：

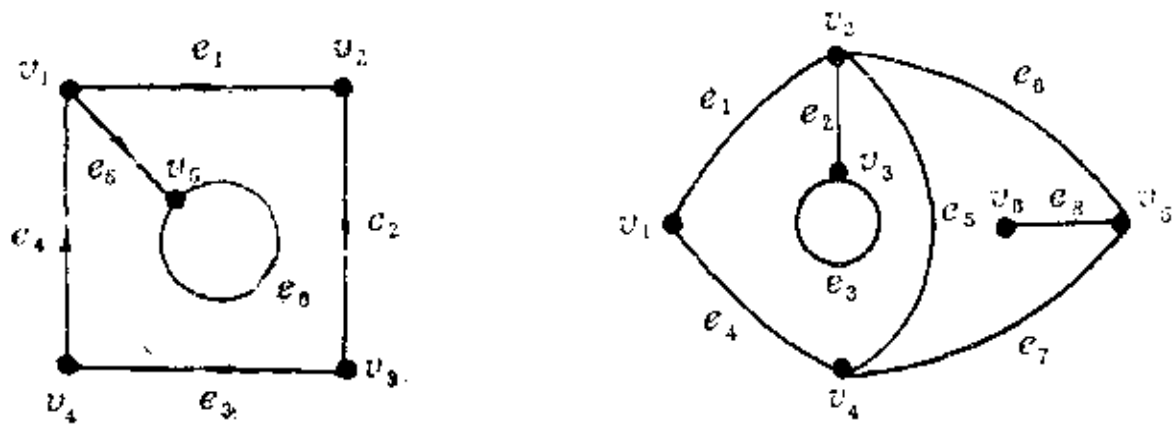


图 1.19　二个图

(1) 它们是有向图还是无向图？

(2) 它们有否孤立节点？

(3) 两图中的 v_1 至 v_5 有路存在吗？如有，有几条路？

(4) 两图中分别有哪几条自回路？

(5) 两图中分别有哪几条回路？

(6) 它们各自有跨树存在吗？如有，试找出各种跨树。

(7) 它们的跨树是有向树、内向树或外向树？如是根树则指出其始根或终根之所在。

十、叙述一下在勾连表中嵌入一个新出现的记录或删除表中某一记录的操作过程。

十一、一个农民带有一只狼、一只羊和一棵菜的渡河问题，这是众所皆知的问题。试用有向图来描述其各种可能渡河法。

第二章　问题求解*

问题求解是智能模拟的重要组成部分。从广泛意义上讲，问题求解包括了所有的计算机科学，因为任何可计算的课题都可以看作为一个被求解的问题。在这章中我们只从狭隘意义的角度叙述其中的状态空间、问题归约、与-或图、博奕树及非确定程序等方法。这不是它的分类，而是为了方便而如此叙述。实际上，问题归约法也有与-或图的方法。博奕树也属于与-或树之类；但也有将它另分一类的。

至于**通用问题求解**（GPS）方法，在问题归约法中加以介绍。

2.1　状态空间搜索法

2.1.1　状态空间的表示法及穷搜索法

为了表达问题的方便，我们将通常所用的n维空间（a_1，…，a_n）的n维变量a_i用状态来表示，以表达问题的状态结构。称此空间为**状态空间**。

例2.1　与世界末日有关的“梵塔”问题。传说古代在印度北部的贝拿勒斯的圣庙里，安放着三根插在黄铜板上的宝石针。印度的主神梵天做了梵塔，即在其中一根针上从下到上地按由大到小的顺序串上了六十四片金片。而后要僧侣轮流值班把这些金片在三根针上移来移去。一次只能移动一片，而且不管在哪一根针上，小片永远在大片的上面。并说如果把这六十四片金片全部移至另一根针上时，则世界就将在一声霹雳中毁灭。这个传说在世界上相当有名。要达到这个要求，需要移动的次数为$2^{64}-1=18,446,744,073,709,511,615$。如果每秒钟移动一次，则需将近5800亿年。但据科学家从能源角度推算，太阳系的寿命也只有150亿年。连梵天也万万没有想到世界的生命已毁灭时他的任务还远远没有完成。为了把问题简化，如图2.1所示，假设金片只有三片，要求从1号针移至3号针。用状态空间表示法解之。

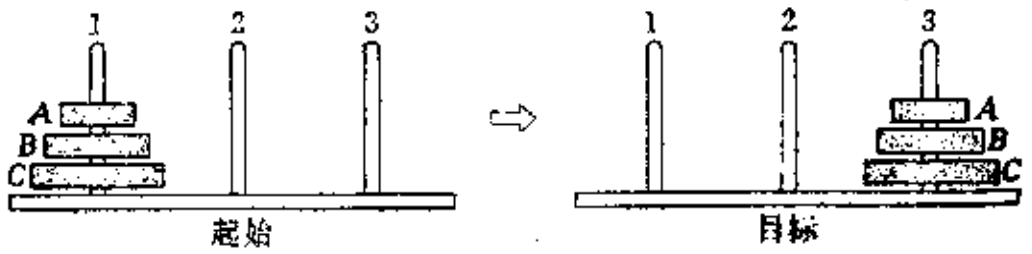


图2.1　梵　　塔

用状态(i,j,k)表示最大金片C在第i根针上，金片B在第j根针上，最小金片A在第k根针上。如果同一根针上有二片以上金片的话，则假设依次地较大者在下面。如（1，2，2）表示C在第1根针上，B和A在第2根针上，而且B在A下。那么初始状态表示为

---

*本章主要参考文献见“参考文献”3、6、11、12和14。

（1 1 1），目标状态表示为（3 3 3）。另外，用M(D,*i*,*j*)表示进行了一次金片D从第*i*根针移至第*j*根针上的操作（也可称为运算）。例如从初始状态把金片A移至第2根针上，则称为做了一次运算M(A,1,2)，使状态（1 1 1）变为状态（1 1 2）。注意，不允许从状态（1 1 2）做一次运算M(B,1,2)而变成状态（1 2 2），因为违反了小片在大片上的规则。即M(B,1,2)不能作用于状态（1 1 2）上。

为了实现图2.1的目标，若把各种可能的操作都搜索穷尽（称这种方法为**穷搜索法**）的话，就如图2.2所示。可见，最好的操作如图中的粗线所示，需要2^3-1次。

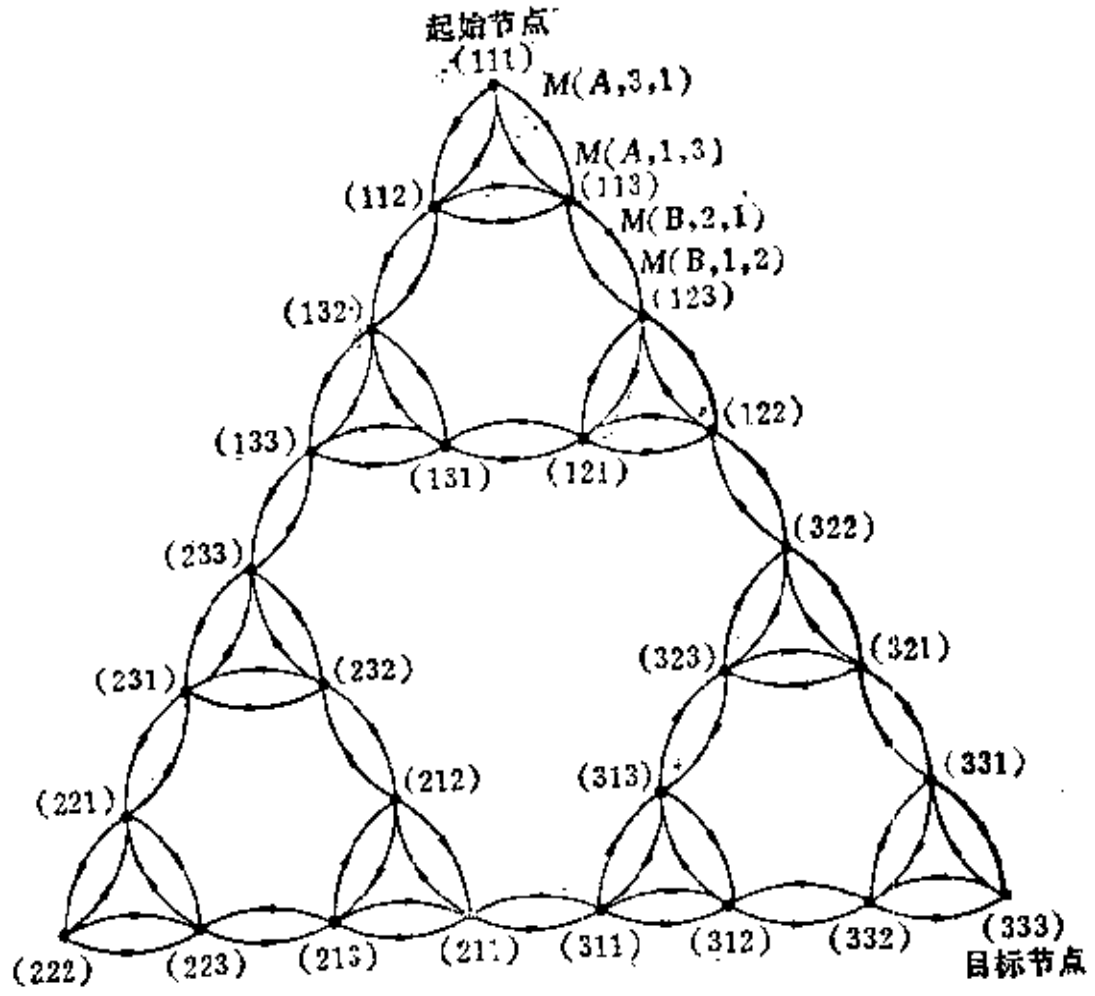


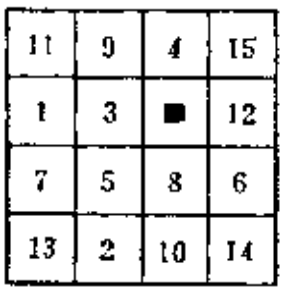
图2.2　三金片的梵塔搜索图

2.1.2　广度优先搜索法

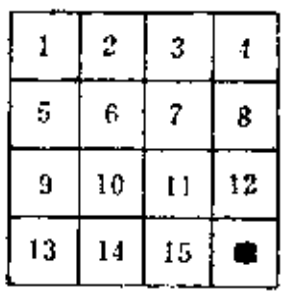
所谓**广度优先搜索法**就是在求解问题过程中，从初始状态开始按照规则进行第一次各种可能的运算之后，产生了第一级各种新的状态。而后再依序地分别对这些第一级新状态都进行第二次各种可能的运算再产生第二级各种更新的状态。以此类推，一直到产生目标状态的某一级为止。

例2.2　我国古代传入欧洲之后、曾经风靡一时的“移动十五”。在4 × 4格中放有十五个分别带有1～15号码的将牌，有一格是空的。移动的规则是每次只能移动一个将牌至空格里。要求把预先给定的一种初始结构，经过一系列移动成为所希望的结构，如图2.3所示。

现把“移动十五”简化成“重排九宫”。其初始状态及目标状态如图2.4所示。其中的空格用黑将牌表示。

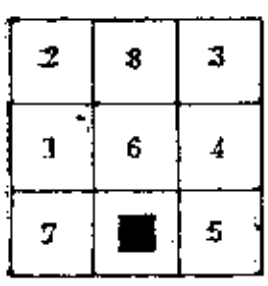


起始状态

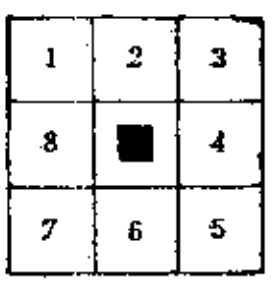


目标状态

图2.3　移动十五



起始状态



目标状态

图2.4　重排九宫

将黑将牌与邻近的号码将牌按左、上、右、下的顺序调换。其搜索的结果如图2.5所示。图中每种结构就是一个节点。搜索过程就是一棵外向树。其始根为初始状态，某一树梢

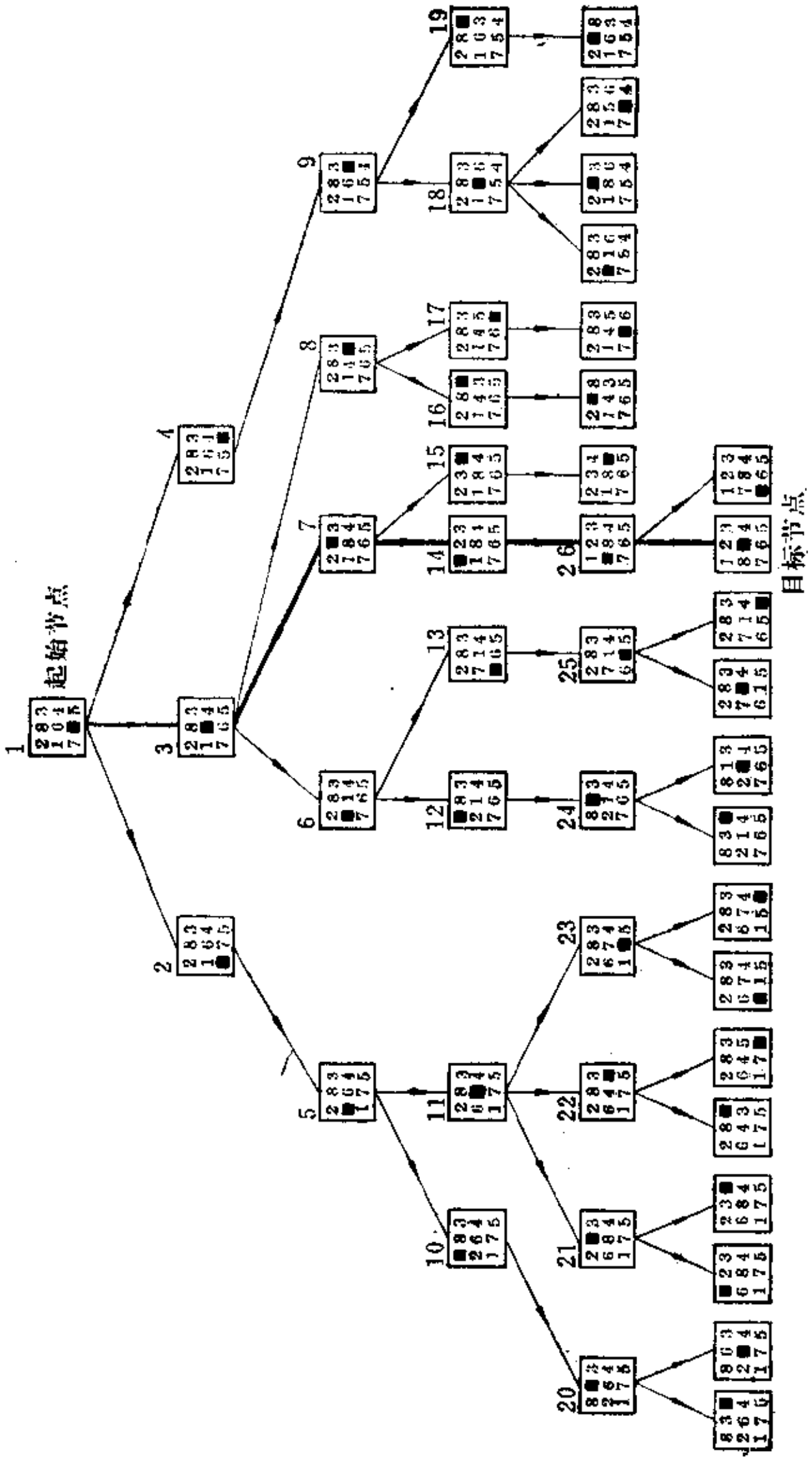


图 2.5 广度优先搜索法产生的外向树

为目标状态的节点。外向树的节点扩展过程是从起始节点开始扩展而产生子节点的，而后依子节点产生的先后次序再扩展各子节点的子节点。以此类推。图中各节点结构图的左（或右）角上的数字表示节点扩展的先后次序。从图 2.5 中可见，在其解找到之前总共产生了46个节点以及扩展了 26 个节点。而且，最简单的移动方法应是图中粗线枝所示的树。只需移动五次就可达到目标。这种表示从起始节点到达目标节点的树，称为**解树**。

从例 2.2 可以看出，广度优先搜索法是依照“先产生的节点优先扩展”的规则来扩展节点的。计算机产生这种广度优先搜索树时，要用一个 OPEN 表和一个 CLOSED 表。OPEN 表是登录产生的节点的，在此表中的节点都是等待扩展的。由于是先产生的节点优先扩展，所以该 OPEN 表是一个先进先出的顺序表。CLOSED 表是登录已被扩展的节点的。参看图 2.6。

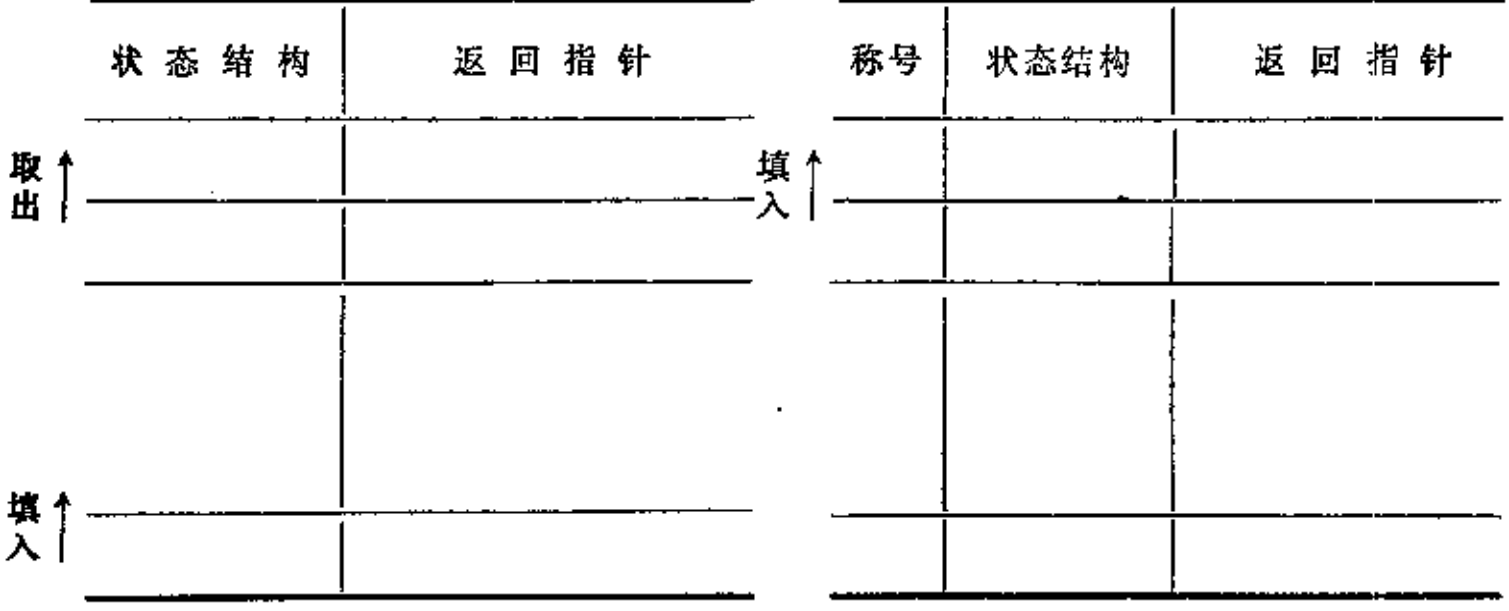


图 2.6 OPEN 表与 CLOSED 表

该方法的算法步骤如下：

(1) 把起始节点放置在 OPEN 表里。

(2) 如果 OPEN 表是空的，则失败；否则继续下一步。

(3) 把 OPEN 表中最前面的节点取出来，放置在 CLOSED 表中；冠这个节点予称号 n。

(4) 把节点 n 加以扩展。如果它没有子节点的话，就立即转到(2)；否则就产生它的所有子节点，并把这些子节点依序放在 OPEN 表的末尾，而且给出由这些子节点返回到节点n的指针。

(5) 如果刚产生的这些子节点中的某一个是目标节点的话，这就说明已求解出来了。只要从目标节点根据指针而往回追溯，一直到起始节点为止，就得了其解树；否则，转到(2)。

以上的流程图示于图 2.7 中。注意，在此不失一般性，假设起始节点本身不是目标节点。否则只需在一开头就加以测试一下它是否目标节点即可。另外，今后也同样省略 n←n+1 操作及置初态 n←0 操作。

显然，如果问题有解存在的话，则广度优先搜索法保证能在这树中找到一条由起始节点至目标节点而长度最短的路。如果没有解存在的话，则该方法就产生了一个有限图，并宣告

失败而退出，或者将永不终止地产生一个无限图。

在有些问题上，着眼点并不在于寻找至目标节点最少的操作次数（即路长最短的路），而是希望有一个具有某些特性的解。这些特性，就采用在图中的枝上加有耗散值来体现。这么一来，求解问题就变成为寻找一条由起始状态至目标状态而具有最小耗散的路。这种更一般的广度优先搜索法，称之为**均匀耗散方法**。上述的求解搜索树方法对均匀耗散方法有很大参考价值。只是需要计算耗散函数值罢了。

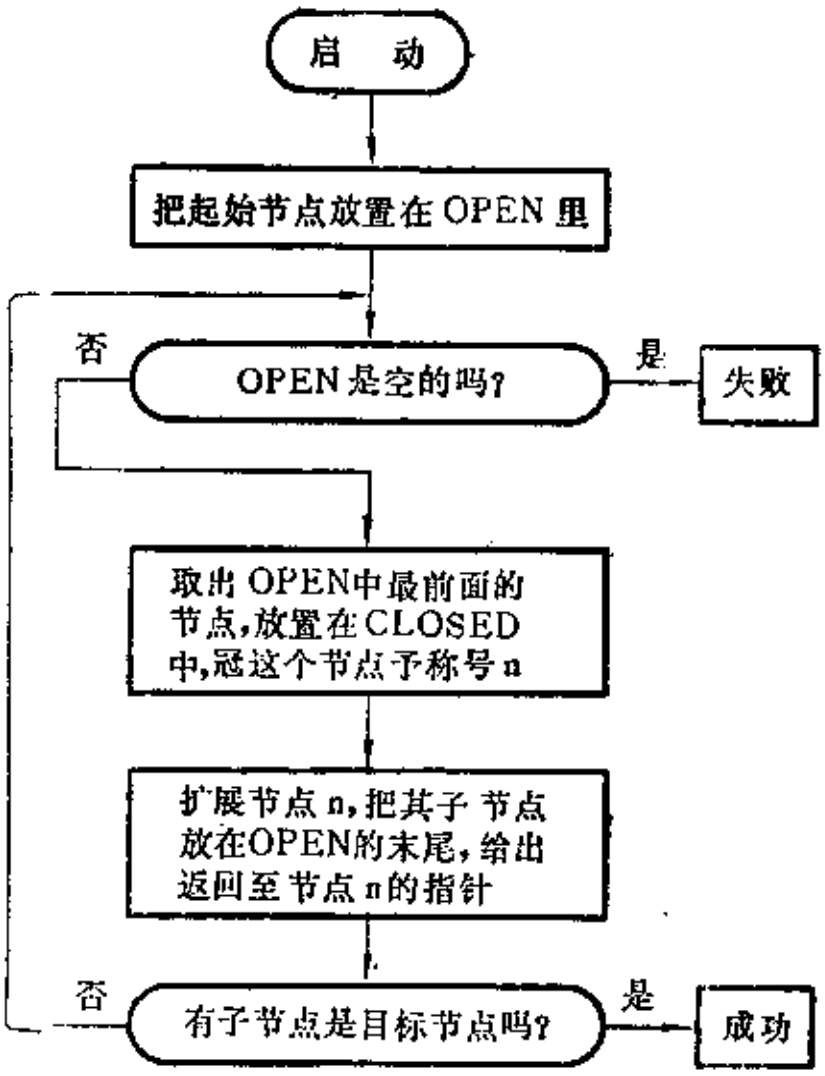


图 2.7 广度优先搜索法的流程图

例设耗散函数 $c(n,n_i)$ 是表示从节点 n 达到其后裔节点 n_i 的路程耗散。$\hat{g}(n)$ 表示在搜索树中从起始节点 s 至节点 n 的路程耗散。在搜索树的过程中，假设 $\hat{g}(n)$ 也是从 s 至 n 的最小耗散路的耗散，因为它是唯一的路。那么，均匀耗散方法在扩展节点时不是以产生节点的先后次序作为节点扩展的先后次序。而是把 $\hat{g}$ 值最小的节点先扩展。因此，在 OPEN 和CLOSED 表中各单元还需有一个表示 $\hat{g}$ 值的信息组。其算法步骤如下：

(1) 把起始节点 s 放置在 OPEN 表里，设置 $\hat{g}(s)=0$。

(2) 如果 OPEN 表是空的，则失败；否则，继续下步。

(3) 把 OPEN 表中其 $\hat{g}$ 值最小的节点取出来，放置在 CLOSED 表中，冠这个节点予称号 n。（因有了“$\hat{g}$ 值最小”的限制，一旦找到目标节点，其路程耗散就保证是最小了。）

(4) 如果 n 是目标节点的话，通过指针往回一直追溯，就得到了一条所希望的解的路；否则，继续下一步。

(5) 扩展节点 n。如果它没有子节点的话，立即转到 (2)；否则，就产生它的所有子节点。对于每个子节点 n_i，用 $\hat{g}(n_i)=\hat{g}(n)+c(n,n_i)$ 来计算 $\hat{g}(n_i)$。使这些子节点带上刚计算过的对应 $\hat{g}$ 值及返回至 n 的指针，而后放置在 OPEN 表上。

(6) 转到(2)。

采用这种算法所找到的目标节点，保证是一条最小耗散路。该算法的流程图示于图2.8中。

只要假设节点和其子节点之间的耗散为 1，这个均匀耗散算法也就能用来寻找最小长度的路了。

如果有若干个起始状态的话，这两个算法在步骤(1)中稍加修改为“把所有的起始节点都放在 OPEN 表中”。

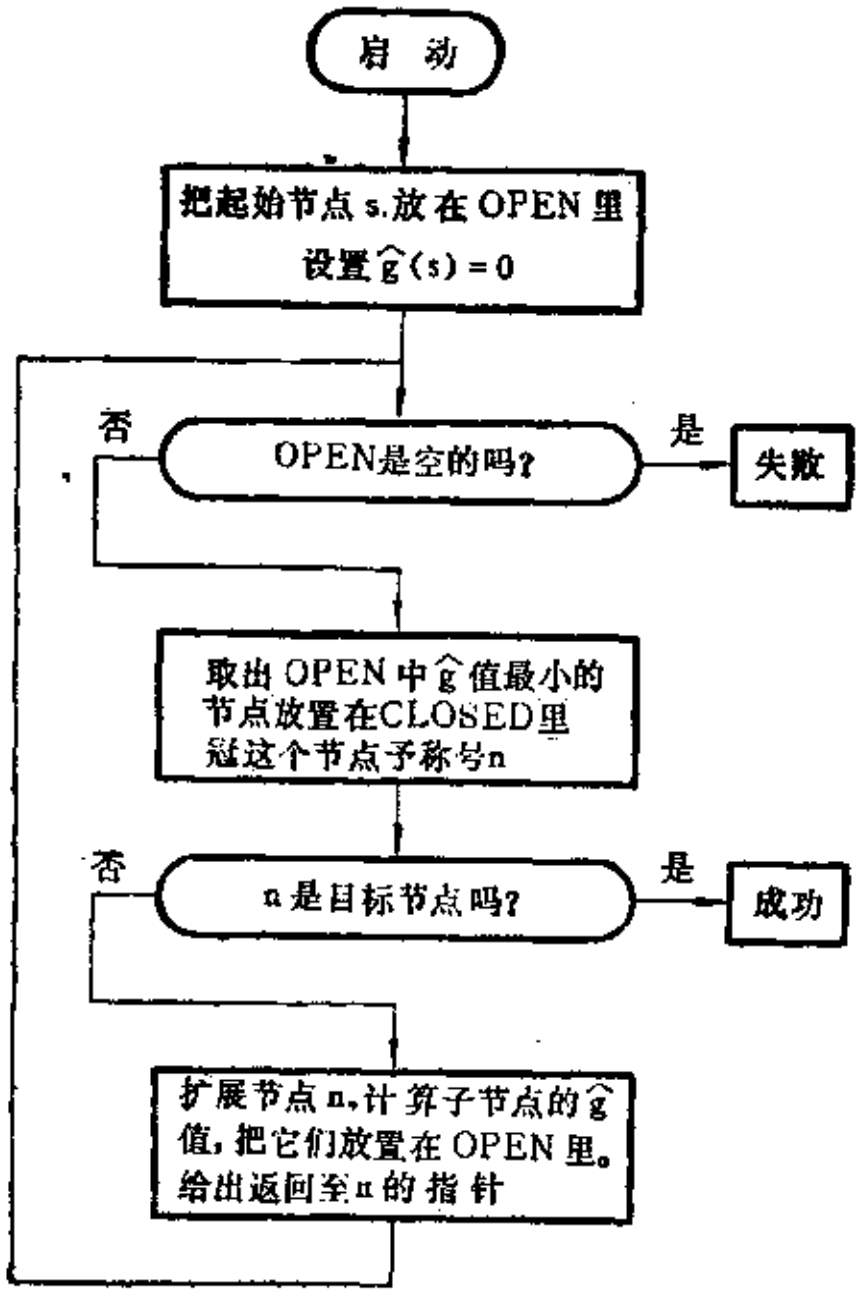


图 2.8 均匀耗散方法的算法。

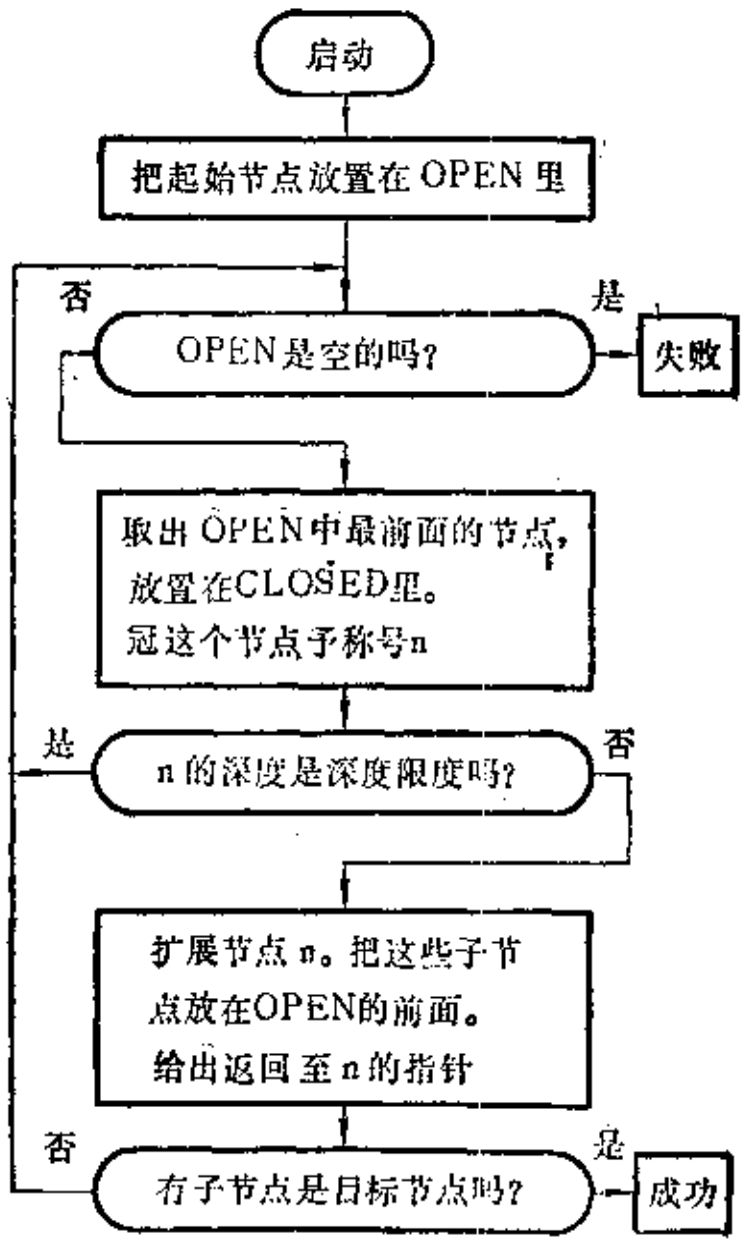


图 2.9 深度优先搜索法的流程图

如果目标状态事先已能描述出来的话，也可以倒过来令目标节点为起始节点，而起始节点为目标节点，再进行搜索。最后把新求出的解的过程翻转一下即可。这种搜索方法称为**反向搜索**。而原来的搜索法称为**顺向搜索**。

在搜索过程中，顺向搜索和反向搜索方法也可同时进行。称该搜索法为**双向搜索**。在双向搜索过程中，所谓“目的已达到了”是指：在顺向搜索过程中所产生的某一个中间状态与在反向搜索过程中所产生的另一个中间状态完全相同。

2.1.3 深度优先搜索法

深度优先搜索法是把最近产生的节点（或者说，最晚产生的节点）优先扩展。在上述的广度优先搜索法所得到的图 2.5 中节点扩展的次序是 1,2,3,…。但深度优先搜索法在扩展起始节点而产生节点 2 时，因为节点 2 比节点 3 更新近产生，故扩展节点 2 而产生节点 5 。因此，节点扩展的顺序是图 2.5 中的序号 1,2,5,10,20,…。从中可理解该方法的名称中“深度”之意。

为了描述各节点在图中的“深度”是多少。我们给在树中节点的**深度**下定义如下：

(1) 根节点的深度为 0 。

(2) 根节点的任何后裔节点的深度就是它的父辈节点的深度加上 1 。

在这样的搜索法中，如果往某一路一直搜索到底而该路可能达不到目标节点，或者能达到，但不是最佳，那是很不上算的。应有一定的深度限制。因此，要注意两点：首先，要给

定“**深度限度**”。扩展的最深节点不能超过这个限度（参见下述的步骤(4)）。其次，当某路搜索到一定深度时，还应沿原路返回再继续搜另一路，所以要给出返回途径（参见下述的步骤(5)）。下述步骤(5)的“放置在 OPEN 表的前面”及步骤(3)的“把 OPEN 表中最前面的节点取出来…”，就可保证“最新近产生的节点优先扩展”这点。此时 OPEN 表就是一个后进先出的**堆栈**（或称为**下推表**）。

深度优先搜索法同样需要二个表，OPEN 表 与CLOSED 表。其算法如下：

(1) 把起始节点放置在 OPEN 表中。定义起始节点的深度为 0。

(2) 如果 OPEN 表是空的，则失败；否则，继续下步。

(3) 把 OPEN 表中最前面的节点取出来并放置在 CLOSED 表中。冠这个节点予称号n。

(4) 如果节点 n 的深度等于深度限度的话，就转到(2)；否则，继续下一步。

(5) 扩展节点 n。如果 n 没有子节点，则转到(2)；否则，就产生 n 的所有子节点。计算子节点的深度。并把这些子节点配上返回 n 的指针。且按任意次序放置在 OPEN 表的前面。

(6) 如果刚产生的这些子节点中的某一个是目标节点的话，通过指针往回一直追溯，就能得到解；否则，转到(2)。

以上的流程图示于图 2.9 中。

例 2.3　以图 2.4 的重排九宫为例，应用深度优先搜索法。假设深度限度取为 5 。就可得到如图 2.10 所示的搜索树。

要特别注意各节点放进 OPEN 表及从 OPEN 表中取出节点的详细过程。在即将扩展起始节点 1 之际，OPEN 表中原来只存放的节点 1 被取出而放置到 CLOSED 表中去了。OPEN 表是空的。而扩展节点 1 之后，其子节点 A，18,2（参阅图 2,10 中的节点序号），又放置在 OPEN 表的前面。虽然子节点放进的顺序任意，但一旦放好，其顺序自然也就形成了。设放进的顺序为 A，18，2。那么 OPEN 表中节点的前后顺序恰恰依序为 2，18，A。再要扩展节点时，就应首先扩展节点 2 。在即将扩展节点 2 之际，OPEN 表中只有 18，A 了。而 CLOSED 表中有节点 1，2 。扩展了节点 2 之后，OPEN 表中有节点 3，18，A。这样一直下去。当扩展了节点 9 之后，OPEN 表中的节点（依前后顺序）有 10,11,12,15,18,A。以此类推。当找到目标节点 E 之后，OPEN 表中的节点(依前后顺序)有 E,D,C,B,A。

深度限度取得过小，就会找不到目标而失败。一旦失败，则再增大深度限度，重新搜索。从图中可见，至目标节点的路长总是小于或等于深度限度。但如果深度限度取得过大，则会造成既多费时又多费存贮空间。故深度限度要取适当的值。

如果问题复杂而枝上的耗散不全为 1 时，即具有耗散的深度优先搜索，也可以有类似的算法。就留给读者自己练习。

比较一下图 2.5 和图2.10，可以看出，如果深度限度取得恰当，深度优先搜索法比广度优先搜索法要少产生 12 个节点。如果目标节点恰好在最后一个分枝上，则产生的节点就一样多。但如果目标节点恰好在最前面的分枝上，则产生的节点更少了。总的说来，前者比后者优越。

2.1.4　评价函数和启发式程序

广度优先和深度优先的搜索法虽然没有像前述的穷搜索法那样把各种解法都求出来。但是在寻找目标节点时，也是搜索得相当彻底的。因而，它还是属于穷搜索法之类。所有这些搜索法都属于**盲目搜索法**。原理上讲，这些方法能给出解来，但是用这些方法常常失败，因为

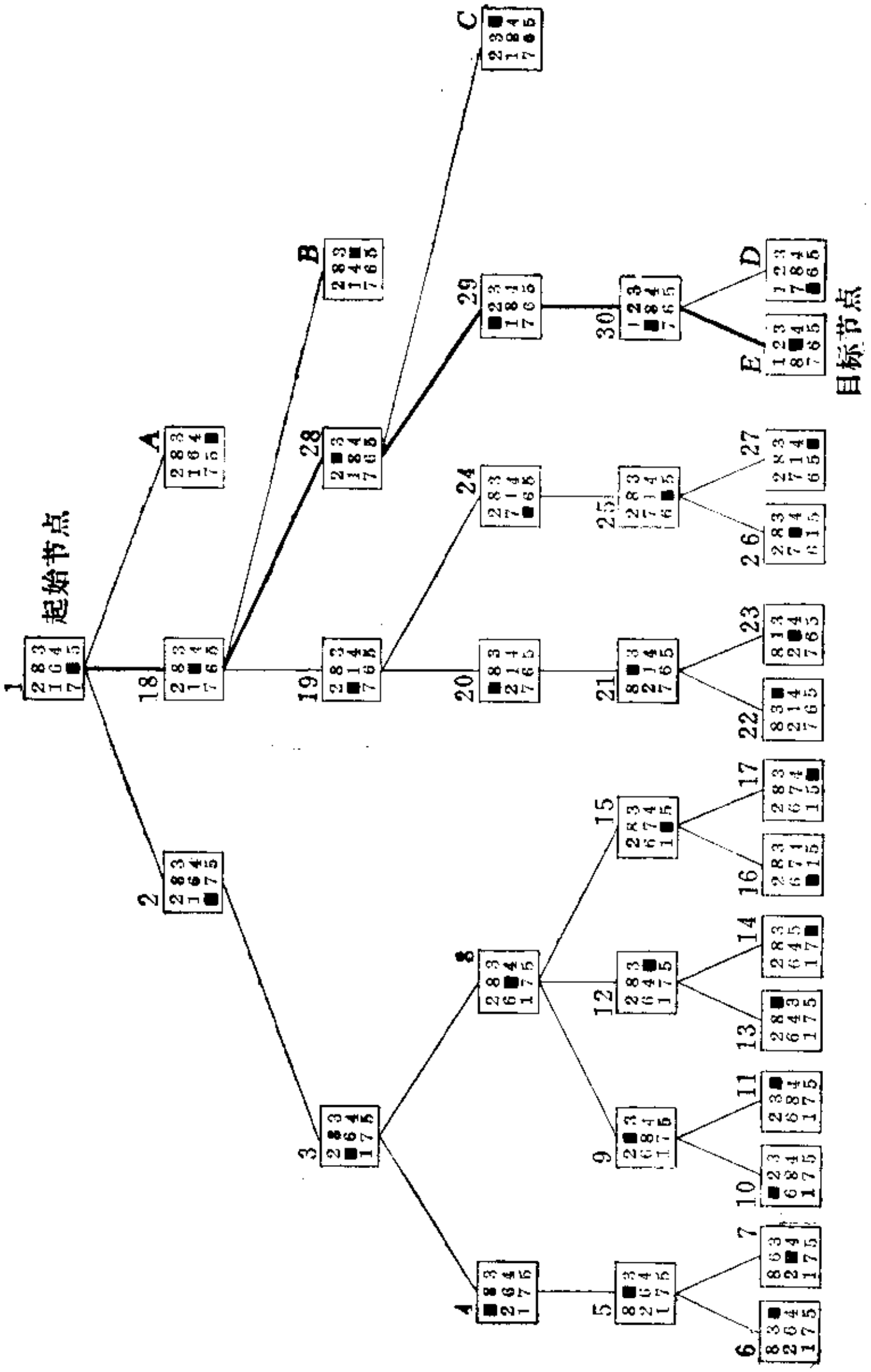


图 2.10　重排九宫的深度优先搜索树

在稍为复杂的问题的搜索过程中在未找到目标节点之前，还需要扩展的节点数实在太多了。总要受到计算机的时间和存贮空间的限制。所以有必要再研究另一种更有效的方法，称之为**启发式搜索法或启发式程序。**

启发式搜索法与盲目搜索法的根本区别在于：前者在搜索过程中选择最有“希望”早达到目标节点的节点来优先扩展。为了描述这个“希望”的程度，定义它为**“评价函数”**。根据各种各样想法来描述评价函数。有的定义它是节点处于最佳路上的概率。也有提出，用在任意节点和目标节点之间的距离来描述。在滑块游戏、重排九宫之类中常常是根据当时格局的得分（即要达到目标的希望）来描述。

有序搜索算法 用$\hat{f}(n)$表示在节点 n 上评价函数的数值，今后我们将知道 $\hat{f}$ 是从起始节点通过节点 n 到达目标节点的最小耗散路的耗散估计。照惯例，把要扩展的节点按照它们的$\hat{f}$值由小到大的次序加以整理。并采用一种像具有均匀耗散算法那样的算法，选择在 OPEN 上$\hat{f}$值最小的节点来扩展。这样的过程称之为**有序搜索算法。**

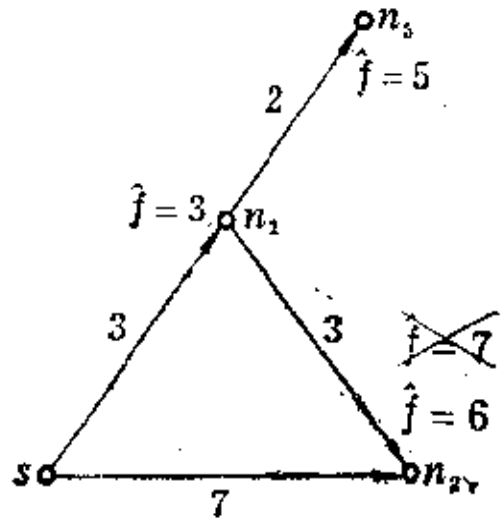


图 2.11 计算$\hat{f}$的例子

为了使这个有序搜索算法适用于搜索不一定是树的一般图，必须考虑到新产生的节点有时可能在 OPEN 表中或在 CLOSED 表中。如果是在 OPEN 表中的话，则说明可能找到了另一条从起始节点至该节点耗散更小的路。如图 2.11 所示的图中，有节点 s 和 n_1，n_2，n_3。其耗散如图所示。当扩展节点 s 之后，产生了节点 n_1 和 n_2，其$\hat{f}$值分别为 3，7。而后再扩展节点 n_1。产生了节点 n_2，n_3，其$\hat{f}$值分为 6，5。可见又有一条 s 至 n_2 的路。因此必须选择耗散较小的路。因为由 s 通过 n_1 再到 n_2，其耗散为 6，而原来由 s 至 n_2 的路的耗散为 7，故选择 s —n_1—n_2 的路为宜。在算法中，如果在扩展节点时所产生的子节点出现在 OPEN 表中的话，则将该节点仍然保留在 OPEN 表中，但其$\hat{f}$值及指针应加以修改，取较小的$\hat{f}$值。如果新产生的子节点出现在 CLOSED 表中，则原来的路的耗散必然较小，该节点没有再扩展的必要，就不放置在 OPEN 表中了。

有序搜索算法的步骤如下：

(1) 将起始节点 s 放置在 OPEN 表中，并计算$\hat{f}(s)$值。

(2) 如果 OPEN 表是空的，则失败而退出；否则，继续下一步。

(3) 从 OPEN 表中取出$\hat{f}$值最小的节点，并把它放置在 CLOSED 表中。冠这个节点予称号 n。

(4) 如果 n 是目标节的话，通过指针返回追溯就得到了一条所希望的解的路；否则，就继续下一步。

(5) 扩展节点 n。如果它没有子节点的话，立即转到(2)；否则，就产生它的所有子节点。对于每个子节点 n_i，计算$\hat{f}(n_i)$值。

(6) 如果子节点既不在 OPEN 中也不在 CLOSED 表中，就将它配上(5)中刚计算过的$\hat{f}$值和返回到 n 的指针。把这些节点放置在 OPEN 表中。

(7) 如果子节点已在CLOSED表中，则该子节点不放置在OPEN表中；如果子节点已在OPEN表中，就把其原先的$\hat{f}$值与在(5)中刚计算过的$\hat{f}$值加以比较：若前者不大于后者，则不作任何更改；若前者大于后者，则将OPEN表中该子节点的$\hat{f}$值更改为刚计算的$\hat{f}$值，返回指针更改为n。

(8) 转到(2)

该算法的流程图与图2.8的均匀耗散方法类似，故在此从略。注意，由这个算法所产生的指针和节点的集合就形成了一棵搜索树。在OPEN表中的节点全在该树叶上。反之，在该树叶上的节点不一定全在OPEN表中。因为若节点没有子节点的话，它是树叶，但已放在CLOSED表中。

例2.4　再以图2.4的重排九宫为例来说明该算法。在这个例子中暂时$\hat{f}(n)$值采用如式(2.1)的简单评价函数。

$$\hat{f}(n) = \hat{g}(n) + w(n) \tag{2.1}$$

其中，$\hat{g}(n)$是在搜索树中从起始节点s到节点n的路长。而$w(n)$是在节点n所描述的状态中放错地方的将牌数目。例如起始节点的评价函数值等于$\hat{f}(0) = \hat{g}(0) + w(0) = 0 + 4 = 4$。

应用有序搜索算法之结果示于图2.12。图中各节点的$\hat{f}$值，用数字加圆圈表示，不加圆圈的数字表示节点扩展的顺序。与图2.10比较，两者找到同样解的路。但是应用了评价函数，需扩展的节点就少得多了。

选用了评价函数可以对搜索的结果给予一定的评价。但是如果所用的评价函数掌握得过严而对确实有希望的某些节点也拒绝的话，则其结果可能在非最小耗散路上。另一方面，如果所用评价函数掌握得过宽，而对可能有希望的节点全都承诺（如均匀耗散算法）的话，其结果将会出现节点扩展得太多了。因此有必要研究一种既保证能找出最小的耗散路又能使扩展的节点数目最少。这就是**最佳搜索算法**。

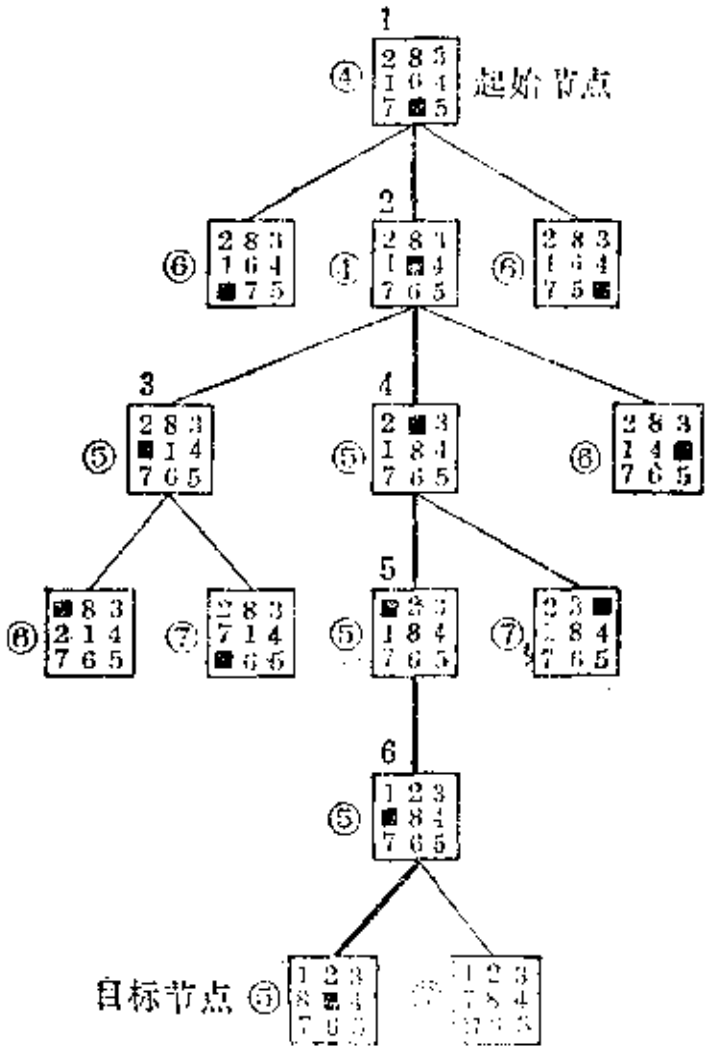


图 2.12　重排九宫的有序搜索法

最佳搜索算法　为了讨论问题方便，函数$g(n)$定义为从起始节点s到节点n的最小耗散路的实际耗散。函数$h(n)$定义为节点n到目标节点的最小耗散路的耗散（如果从该节点不能达到目标节点，则函数$h(n)$无定义）。如果从节点n到目标节点取得$h(n)$值的任一条路，则称之为从n到目标的**最佳路**。因此

$$f(n) = g(n) + h(n) \tag{2.2}$$

是从起始节点s到达目标节点（约束它应通过节点n）的最佳路之耗散。注意，$f(s) = h(s)$

是从 s 到目标的无约束（即不一定通过节点 n）的最佳路的实际耗散。

但是，既然实际的最佳路仍未找到，$g(n)$、$h(n)$值是无法得出精确值来的，只能“估计”而已。不过我们总是希望所得的估计值应当是最佳路（即耗散最小）的“估计”。因此，将 $\hat{g}(n)$定义为 $g(n)$的估计，$\hat{h}(n)$为$h(n)$的估计。则$\hat{f}(n)$由下式定义：

$$\hat{f}(n)=\hat{g}(n)+\hat{h}(n) \tag{2.3}$$

由式（2.3）可知，$\hat{f}(n)$就是约束它应通过节点 n 的最小耗散路的耗散估计，即 $\hat{f}(n)$为 $f(n)$的估计。

原先关于 $\hat{g}(n)$的选择仍是在从 s 到 n 的搜索树中从 n 根据指针返回到 s，对所遇到的枝的耗散求和而给出的路的耗散。而这条路是在至今所采用的算法中找到的从 s 到 n 的最低耗散路。注意，这个定义包含有 $\hat{g}(n)\geqslant g(n)$之意。

称应用式（2.3）的有序搜索算法为**算法 A***。注意，当 $\hat{h}\equiv 0$ 时，算法 A^*与原来所述的均匀耗散算法相同。

由上述讨论可知，$\hat{f}(n)$如何求得是很重要的。如果 $\hat{f}(n)$值求得不恰当，不仅会使问题复杂化，甚至于使机器由于存贮空间的限制而无法求解问题。因此，就详细讨论 $\hat{f}(n)$值中 $\hat{h}(n)$和 $\hat{g}(n)$值。

$\hat{h}$ 的启发能力 $\hat{h}$ 的选择是牵涉到有序搜索算法的效率问题的。如果某种算法的效率高，就称该算法的启发能力强。因而，$\hat{h}$ 的选择是提高算法的启发能力的关键所在。就以重排九宫为例来说明 $\hat{h}$ 的重要性。式（2.1）中函数 $\hat{h}(n)=w(n)$（在此，$w(n)$ 是放错地方的将牌数目）是 $h(n)$ 上的低限。但是，对于将牌结构的难点（如在到达目标的步数方面），并没有提供很好的估计。如图 2.13 中所示的两种将牌结构图。按照 $\hat{h}(n)=w(n)$计算，图 2.13(a)的 $\hat{h}(n)=w(n)=7$，而图 2.13(b)的 $\hat{h}(n)=w(n)=5$。似乎前者到达目标的步数比后者多。其实不然。较好的估计还需考虑函数 $\hat{h}(n)=p(n)$。在此 $p(n)$是每个将牌离“家”（该将牌在目标结构中的位置）的距离（即现在将牌所在的格子离“家”有多少格子，不管其间有否别的将牌插入）之总和。然而，这样的估计仍是太粗糙了。它实际上并没有对两个相邻将牌调换位置的困难程度加以评价。

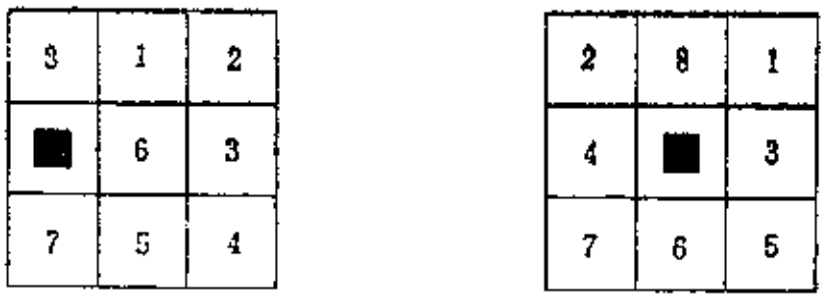


图 2.13　两种将牌结构

对于重排九宫而言，相当好的估计是

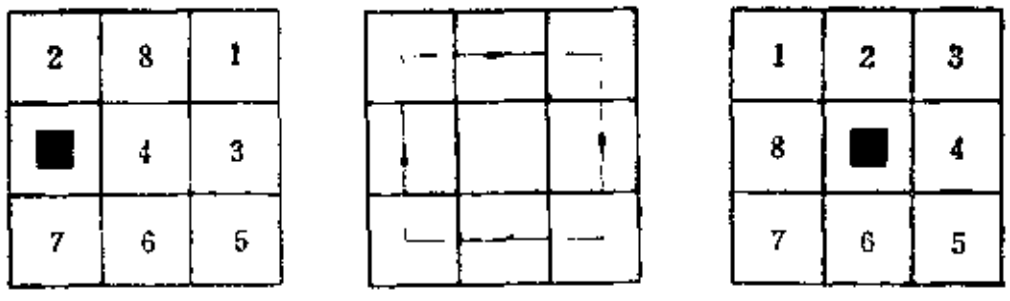
$$\hat{h}(n)=p(n)+3s(n) \tag{2.4}$$

其中，$p(n)$ 是每个将牌离“家”的距离之总和。$s(n)$ 是这样来评分的：沿着周围那些非中心方格上依序（如图 2.14(b) 所示）检查每个将牌。只要某个将牌后面紧跟着的另一个将牌恰恰也是在目标结构中该将牌的后续者的话，对于到达目标是有利的，故该将牌得 0 分；否则，得 2 分。在正中心位置上有将牌得 1 分；否则，得 0 分。

而对搜索树中每个将牌的评分标准如下式所示。

$$\hat{f}(n) = \hat{g}(n) + p(n) + 3s(n) \tag{2.5}$$

在此，$\hat{g}(n)$如式（2.1）所定义，$p(n)$和$s(n)$如式（2.4）所定义。



(a) (b) (c)

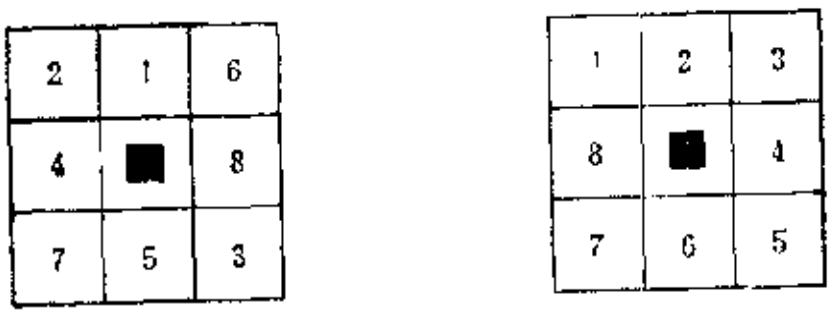
图 2.14 重排九宫的$\hat{h}(n)$计算法

例2.5 以图 2.14 为例来说明如何评分。图 2.14(*a*)是现行将牌结构图，图 2.14(*c*) 是目标的结构图。因而沿着周围非中心格上的将牌应有的后续者顺序，应是 1，2，3，4，5，6，7，8，1（或黑将牌）。对图 2.14(*a*) 的结构图进行检查。由于将牌 2，1，3，7 之后应分别是将牌 3，2，4，8 紧跟着，但它们现在却都没有应有的后续者紧跟着。故该四个将牌各得 2 分。此时这四个将牌累计分为 8；而将牌 5，6，8，之后都有应有的后续者紧跟着，故得分都为 0。而且正中心方格有将牌 4，得 1 分。这样，$s(n)$的累计分为 9。

$p(n)$ 值是这样求的。因将牌 5，6，7 都恰好已在其“家”，故得 0 分。而将牌 2，3，4 离家的距离都是 1，故各得 1 分。此时 $p(n)$的累计分为 3。将牌 1，8 离家的距离都是 2，故各得 2 分。此时 $p(n)$的累计分为 7。因此 $\hat{h}(n) = p(n) + 3s(n) = 7 + 3 \times 9 = 34$。显然，目标结构图的 $\hat{h}(n) = 0$。

$\hat{g}(n)$值在此无法具体计算。只好在整个搜索树中具体地从它离起始节点的路长来计算。可见，$\hat{f}(n)$值愈小，该节点愈有希望是处于到达目标节点的最佳路。极而言之，同样是两个目标节点（$\hat{h}(n) = 0$），但其 $\hat{g}(n)$值较小者，说明从起始节点出发其移动的步数也是较少的。

应用式（2.5）的有序搜索算法，对于比以前已求解的更难的重排九宫结构，也就能够容易地解出了。例如图 2.15 所要求的重排九宫，其解的搜索树如图 2.16 所示。在图中，节点外面加有圆圈的数字表示该节点的 $\hat{f}$ 值，没有加圆圈的数字表示扩展的次序。



起始结构 目标结构

图 2.15 重排九宫的另一例

从图 2.16 中可见，虽然由于 $\hat{h}$ 函数不是 h 的低限，我们不能保证找到一条最佳路。但是所找到的路却是最短(只有 18 步)。

从上例可见，$\hat{h}$ 函数的确定是很重要的。首先，如上例所示，$\hat{h}$ 值是在注视着如何搜索而产生的。更确切地说，是在注视着如何径直地通向目标而产生的。这样，在起始节点附近只出现一些很少量的枝叶。确定有序算法的搜索效率的另一个因素是花在计算 $\hat{h}$ 上的琐碎工作量，最佳的 $\hat{h}$（扩展的节点数的确最少）将是一个等于 h 的函数。例如，若有这样的 $\hat{h}$，

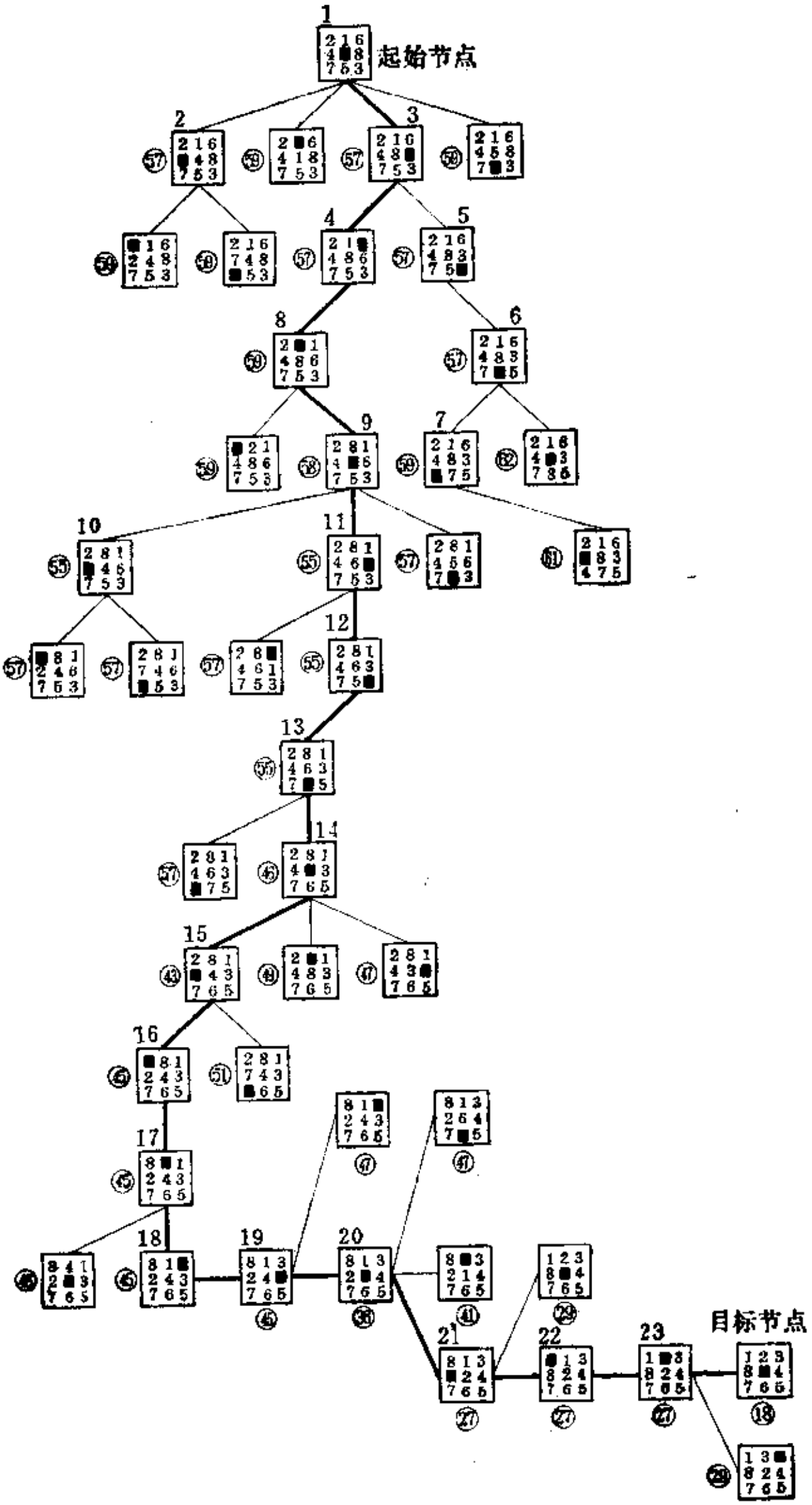


图 2.16　有序搜索法产生的树

在每个节点都有一个独立的完整的搜索，但是它的总的琐碎计算工作量却可能是相当之庞大。这就不一定是理想的。

有时 $\hat{h}$ 函数虽然并不是 h 的低限，但却比较容易计算。在这种情况下，启发能力可以成倍增长。因为扩展的节点数可以降低，而且琐碎的计算工作量也降低。在某些情况下，可以采用把它乘上一些比 1 大的正系数的简便方法，来提高 $\hat{h}$ 的启发能力。

扼要讲来，有三个重要因素影响有序搜索算法的启发能力：

(1) 路的耗散；

(2) 在寻找路的过程中所扩展的节点数；

(3) 计算 $\hat{h}$ 时所需要的计算工作量。

对于每个问题，就要权衡一下这些因素，以使启发能力最大。

$\hat{g}$ 的重要性 在许多问题中，只希望找到达到目标节点的路而不在乎路的耗散。当然也关心在搜索过程中所需要的工作量。在这样场合下，在评价函数 f 中应包含有 g 呢还是可忽略它？都有些直观的理由。

当只需求达到目标的任一路时，$\hat{g}$ 可忽略，因为在搜索期间的任一级上，不关心发展至今的路耗散。关心的是找到目标时还需的搜索工作。这个搜索工作（可能与 OPEN 中节点的 $\hat{h}$ 值有关）确实与 OPEN 中节点的 $\hat{g}$ 值无关。因此，对于这样问题，就采用评价函数 $\hat{f}=\hat{h}$。另外，$\hat{h}$ 值相当大时，$\hat{g}$ 值也可忽略。

有时 $\hat{g}$ 却不要忽略为好。例如，两个不同节点经过搜索达到相同的第三个节点时，但从起始节点分别经过这两个节点而达到第三个节点的路长（或耗散）$\hat{g}$ 值，可能是不同的。其 $\hat{g}$ 值最小者当然要好一些。因为相同的第三个节点到目标节点的 $\hat{h}$ 值应是同一值，这样两者的 $\hat{f}$ 值就不同了。

上述两者理由各执已见。大概还是后者合理些，能综合各方优点。

2.1.5 性能的测量

搜索技术的启发能力并不是通用的。根据具体问题而定。如何确定它，则要根据试验结果，而不是理论计算来评定。评定各种算法的好坏也是如此。然而对各种算法的性能测量也需计算一下。虽然他们不能完整地确定启发能力，但是它们仍然用来对各种搜索技术进行比较。在此介绍两种方法。

一种测量方法称为**外显率** P。这是一个生物学词汇。用式子表示如下：

$$P=\frac{L}{T} \tag{2.6}$$

在此，L 是从起始节点至目标节点的路长，T 是在搜索期间所产生的节点总数（包括目标节点在内，但不包括起始节点）。

从式 (2.6) 可知，P 值愈大的搜索技术愈优越。如果有由许多搜索技术所产生的许多搜索树的话，用 P 来测量一下，则必然淘汰那些树叶繁茂的灌木树，而选留短而细长枯枝落叶的苗条树。就以重排九宫为例，前述的方法有广度优先搜索法，深度优先搜索法，$\hat{f}(n)=\hat{g}(n)+\omega(n)$ 以及 $\hat{f}(n)=\hat{g}(n)+p(n)+3s(n)$ 等四种。它们的 L 值分别为 5，5，5，18；而 T 值分别为 46，34，13，44。其 P 值列于表 2.1 中。

表 2.1　重排九宫各例的外显率及实际分枝系数

搜索方法	广度优先	深度优先	$\hat{f}=\hat{g}+w(n)$	$\hat{f}=\hat{g}+P(n)+3s(n)$
外显率 P	5/46 0.108	5/34 0.147	5/13 0.385	18/44 0.409
实际分枝系数 B	1.86	1.75	1.34	1.08

因为随着 L 的增加，T 增加得更快些，所以对同样的问题要大量地试验而后才评定那种搜索技术较好。

另一种测量称为**实际的分枝系数** B。它设想一棵树具有深度等于路长且节点总数等于在搜索期间所产生的节点数。在这树中，被扩展的每个节点，其子节点数目都是常量 B。因此所产生的节点总数 T 与路长 L 及 B 有关。如下式所示。

$$B+B^2+\cdots+B^L=T$$

即

$$\frac{B}{B-1}(B^L-1)=T \tag{2.7}$$

虽然式 (2.7) 中 B 无法写成 L 和 T 的显函数，但是对于不同的 L 值，B 和 T 的关系曲线可由图 2.17 示出。

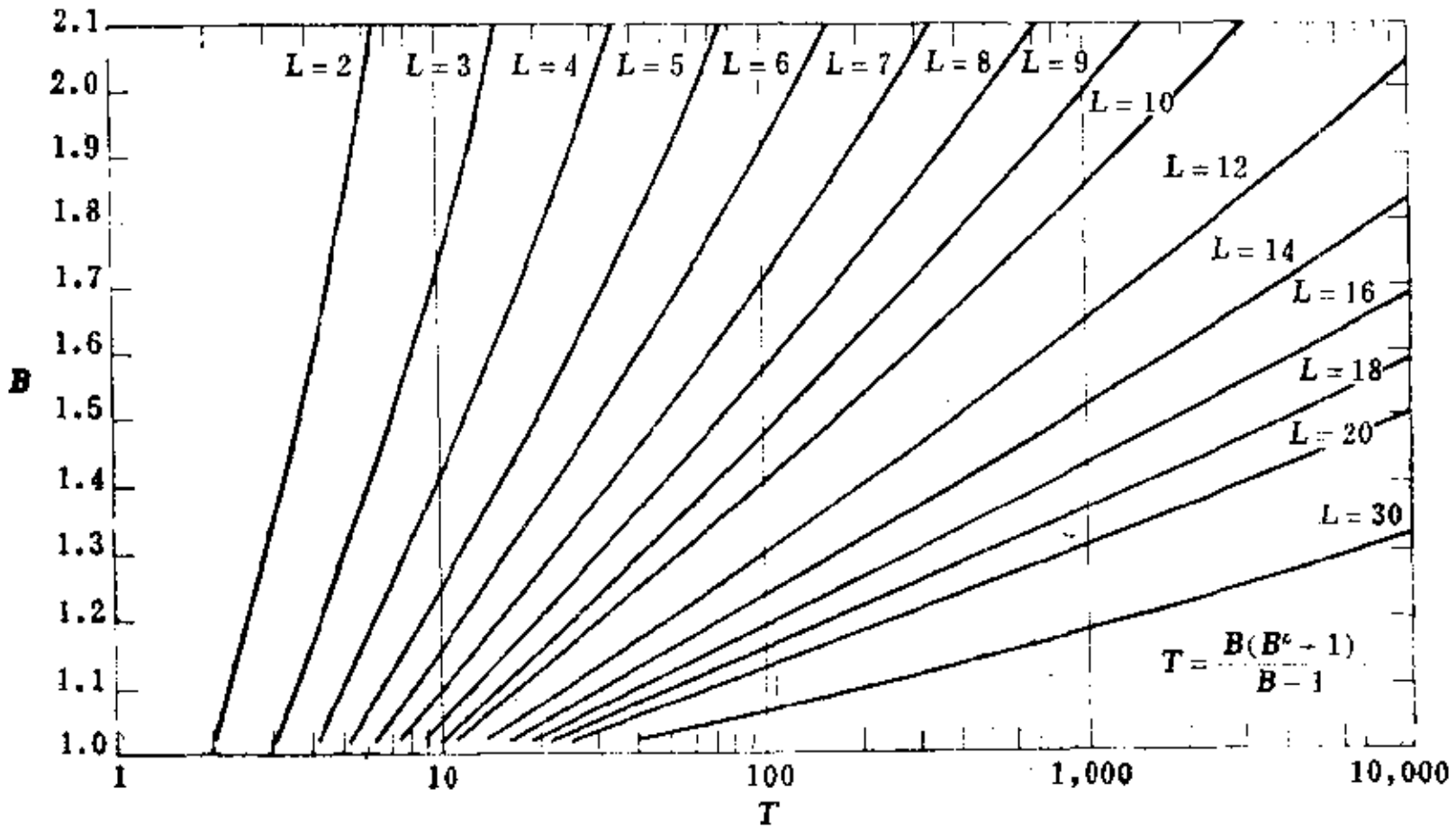


图 2.17　对于不同 L 值的 B 与 T 的关系曲线

对于前述的例子，它们的 B 值也一起列于表 2.1 中。

P 对于 B 和 L 的关系可由下式表示：

$$P=L(B-1)/B(B^L-1) \tag{2.8}$$

对于不同的 B 值，P 和 L 的关系曲线示于图 2.18 中。

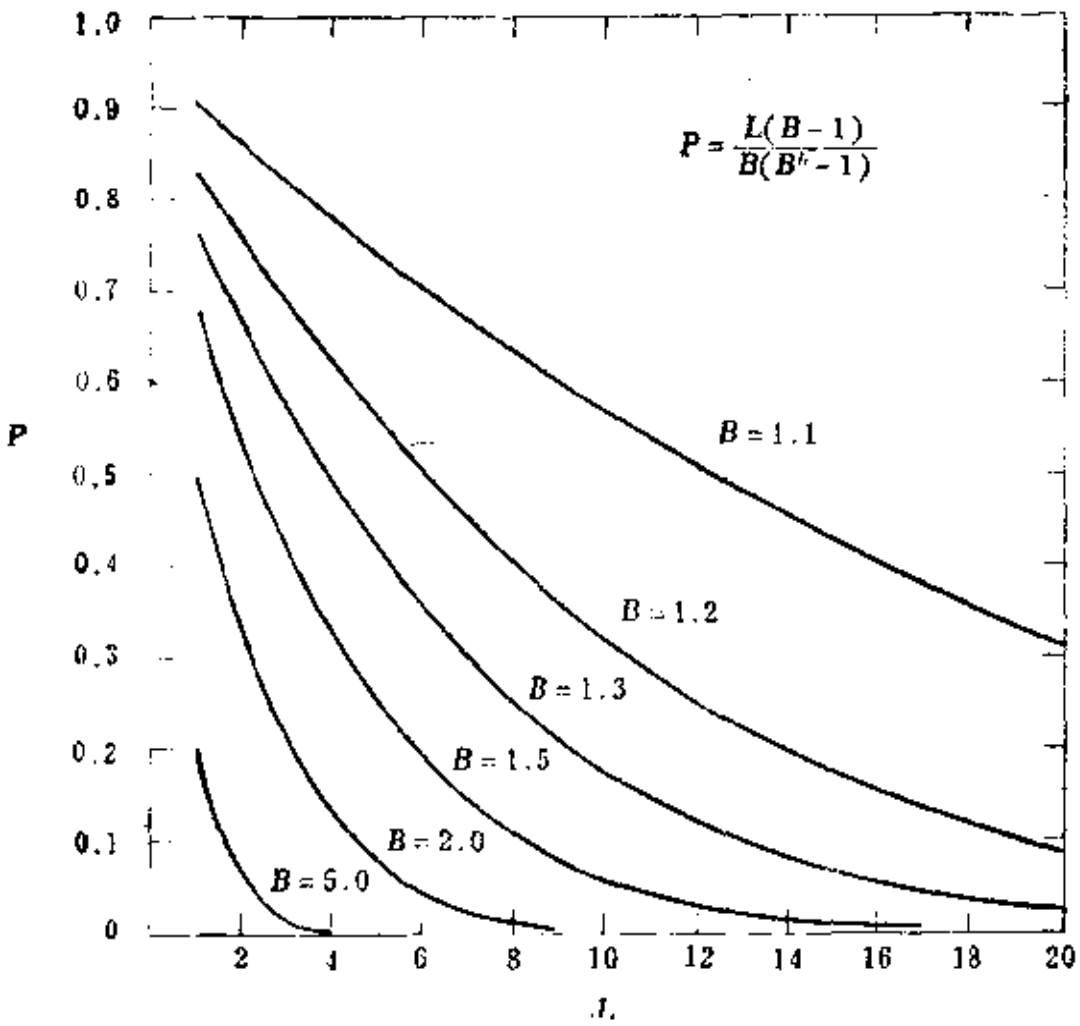


图 2.18　对于不同 B 值的 P 与 L 的关系曲线

2.2　问题归约法

问题归约法的基本想想是倒推论。即从欲解的问题出发，确立若干子问题。这些子问题一旦能求解，原先的问题也就可求解出来了。如果这些子问题中的某些问题仍不能解出，则再从它们出发，再确定若干子-子问题（或略称为子问题）。类推之。直到最终的子问题都是已知的真命题或已被证明（或求解）过的命题。原先的问题也就最终求解出来了。

2.2.1　算符法

问题归约算符法是把一个问题描述变换成归约过（或后裔）的问题描述的一个集合。注意，此处的“变换”必须是：所有的后裔问题能求解就意味着其先辈问题也能求解。换句话说，其先辈问题的解可以由其后裔问题的解按照一些简单运算规则而得出。如果其后裔问题的集合的成员仅有一个，那么该先辈问题就可由其后裔问题所等价或代替。

例 2.6　欲证平面几何的定理：角平分线上的一点至该角两边的距离相等。如图 2.19 所示。改写如下：

已知：∠DBA = ∠DBC　　AD⊥BA　　CD⊥BC

　　DB 是线段　△BCD 是三角形　△BAD 是三角形

求证：AD = CD

对这样问题可以如此归约。根据已知的定理“恒等三角形的对应边相等”对它进行变换。变成另一个命题，已知条件不变，但求证部分改为“△BAD≅△BCD”。可见，后者能

被证明为真的话，就意味着原来的定理也为真。而且后者是原先定理的唯一后裔问题，故取代原先欲证的定理，而着眼点移至这样的命题“△BAD≅△BCD”上。

例 2.7　如图 2.20 所示。欲证如下定理。

已知：ABCD 为四边形　线段 BC 平行线段 AD　BC = AD　AB、CD 都是线段

求证：AB = CD

这样的定理，引入补助线 BD，就可以把它变换成“△ABD≅△CDB”、“△ABD 成三角形”、“△CDB 成三角形”（简写为“△ABD”，“△CDB）这样三个后裔问题。

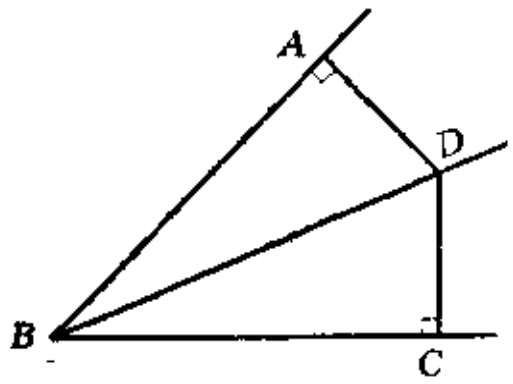


图 2.19　平面几何的定理

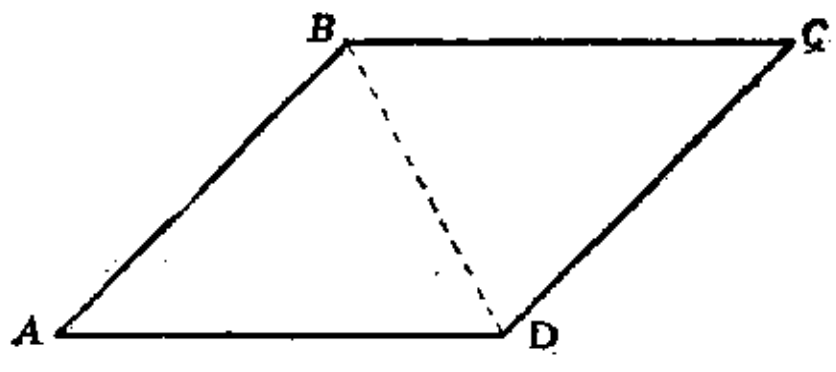


图 2.20　引补助线的平面几何定理

把这类方法加以系统化。令 S 代表欲证其为真的陈述（命题、定理），T 代表那些假定为真的前提陈述（命题、定理）的一个集合。令 S|T 表示“由 T 中给定的前提来证明 S”这样的问题。读成“由 T 给出 S”。

图 2.19 所示的例 2.6 中，T 就是所有已知条件“∠DBA = ∠DBC, …”所组成的集合，S 就是欲求证的“AD = CD”。而 S|T 就是 AD = CD|∠DBA = ∠DBC，AD⊥BA，…。这就是原欲证定理的陈述。

这类归约问题的共同格式就是引入新的前提至原始的问题中去，而后建立求解新前提的附加问题。

例 2.6 中，引入新前提“△ABD≅△CDB”，原始问题变为“AD = CD|∠DBA = ∠DBC，…，△ABD≅△CDB”，而后建立另外欲证的附加问题为“△ABD≅△CDB|∠DBA = ∠DBC, …”。

用算符来描述则是：原始问题是“S|T”，引入新前提 X 之后变为“S|T，X”，而后建立欲证的附加问题为“X|T”。

注意，“算符“S|T，X”必须是公理或被证明过的定理之类的**本原问题**。否则，还必须再进行归约。在平面几何定理的证明中，本原问题有很多。公理肯定是本原问题。从公理证出的定理，从公理或已证过的定理再证出的定理都可归入本原问题。在此，仅举几例。

$\angle X_1 = \angle X_2 | \angle X_1 =$ 直角，$\angle X_2 =$ 直角。　$X_1X_3 = X_1X_2 | \triangle X_1X_2X_3$，$\angle X_2 = \angle X_3$。

$\angle X_1X_2X_3 = \angle X_2X_3X_4 | X_1X_2 // X_3X_4$。这些都可是本原问题。

引入的新前提也可以是若干个（需有限个），设为 N 个，记为 X_i。则可得到归约问题的集合：

$$S|T, X_1, X_2 \cdots, X_i, X_{i+1}, \cdots, X_N,$$

$$X_1|T, X_2, \cdots, X_i, X_{i+1}, \cdots, X_N,$$

……

$X_i|T, X_{i+1}, \cdots, X_N$,

……

$X_{N-1}|T, X_N$,

$X_N|T$

例 2.8　例 2.7 的算符是："AB = CD|ABCD 为四边形, BC//AD, BC = AD, AB, CD,"。为方便起见，简写为"AB = CD|T"。由于引入补助线 BD，故引入了新前提"△ABD"，"△CDB"，"∠CBD = ∠BDA"，"△ABD≅△CDB"。则可得到其归约问题的集合如下：

AB = CD|T, △ABD≅△CDB, ∠CBD = ∠BDA, △ABD, △CDB

△ABD≅△CDB|T, ∠CBD = ∠BDA, △ABD, △CDB,

∠CBD = ∠BDA|T, △ABD, △CDB,

△ABD|T, △CDB,

△CDB|T

然而，对于任一给定的问题，都会有许多能用得上的归约算符，而每个归约算符都可能会产生一个子问题的集合。这些子问题有的可能是无解的。

例 2.9　图 2.21 所示的算符为

AB = BC|BD = AB, ∠BDC = ∠DCB, BC, CD, AD, △DAB, △DBC

在此，根据本原问题"$X_1X_3 = X_1X_2|\triangle X_1X_2X_3$, $\angle X_2 = \angle X_3$"，可引入新前提"BC = BD"，即可解出该题。但也可能引入"△DAB≅△DBC"。而△DAB≅△DBC|BD = AB, ∠BDC = ∠DCB, …… 是无解的。因此，为了产生其成员都是有解的集合，必须试试几种引入的新前提及其算符。这么一来，问题归约算符法中也有搜索的问题了。

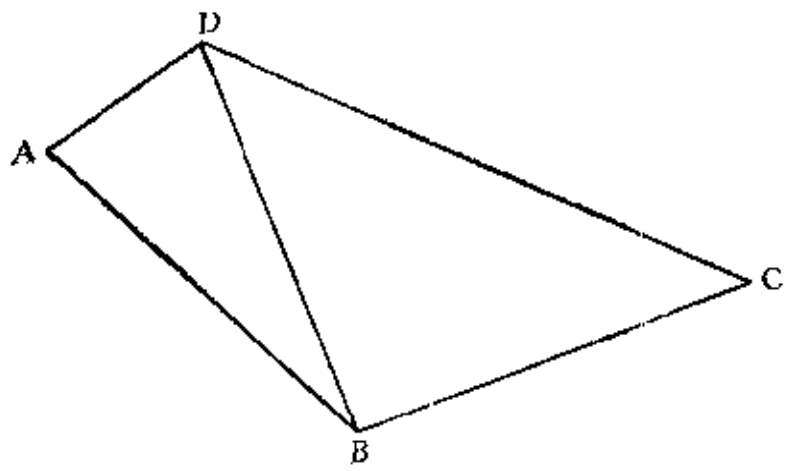


图 2.21　平面几何问题

2.2.2　关键法

问题的描述方法很多，有列表法，树形法，矢量法，阵列法，…等。通常用状态空间表示法还是方便的。现在将问题用三元组 (S,F,G) 来表示。其中

(1) S 是起始状态的集合。

(2) F 是把某些状态描述映照到另一些状态描述的若干操作的集合。

(3) G 是目标状态的集合。

可见，如果在具体问题中对于 S、F、G 的具体内容加以给定的话，那么三元组 (S,F,G) 就用作为一个问题描述了。它定义了一个问题：从起始状态 ($\in$S) 出发，经过有限次的某些操作 ($\in$F) 之后，要达到目标状态 ($\in$G)。

例 2.10　例 2.1 的梵塔问题。S 只有一个成员(111)；F 集合的成员则包括凡是符合金片移动规则的所有操作，即图 2.2 中的所有有向边都是；G 集合中有二个成员：(222) 和

(333)。

问题归约关键法就是在分析和求解问题过程中，寻找出并紧紧地抓住从问题的起始状态过渡到目标状态之间必须经过的某些**关键状态** g（也称之为**里程碑状态**）或者某些**关键操作** f，从而将原先的问题归约为若干子问题，对其子问题再采用同样方法进行归约，一直到最终的子问题是本原问题为止。这就是**通用问题求解器**（GPS）的方法。

关键状态 如果找到某个里程碑状态 g 的话，由于 g 是关键，在求解过程中为了得到最佳解，g 状态是必须出现的。因此，要得到问题的最佳解就必须在从起始状态开始寻找时，其首要的目标应是 g。一旦能找到至 g 的最佳解之后，再把 g 作为起始状态，开始寻找其到最终目标 G 的最佳解。这样，问题 (S,F,G) 就归约为两个子问题：(S,F,{g}) 和 ({g},F,G)。若后两个子问题能求解出来，原先的问题也就能求解出来。其归约图如图 2.22(a)所示。如果能找到的里程碑状态是一个序列 $g_1, g_2, \cdots, g_N$，则 (S,F,G) 可归约为子问题序列 (S,F,{g_1})，({g_1},{F,{g_2}),…，({g_N}, F,G)。如图 2.22(b)所示。解出

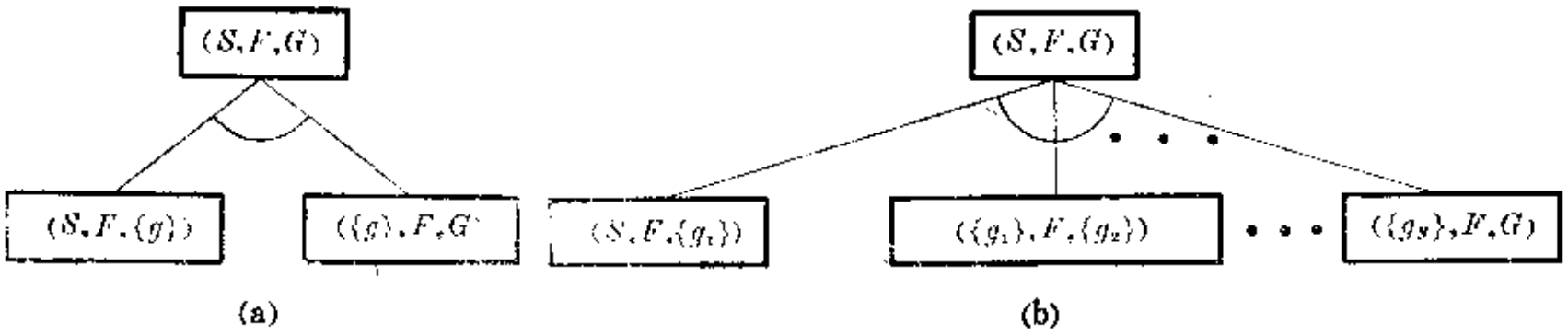


图 2·22 问题归约的关键状态法

了所有这些子问题，也就等效于解出了原先问题。图中的有向枝用圆弧连接起来，表示这些有向枝指向的子节点之间的关系是与关系，称这些子节点为**与节点**。即只有在其子节点“全部都能求解”的情况下，该节点才能最终得到求解。

例 2.11 图 2.1 的梵塔问题。从图 2.2 中可见，其里程碑状态是 (122) 和 (322)。则原先问题 ({(111)},F,{(333)}) 可归约为三个子问题：({(111)},F,{(122)})，({(122)},F,{(322)})及({(322)}, F,{(333)})。

关键操作 如果能找到某个关键操作 f 的话，说明 f 操作必须使用。把允许 f 操作的状态全部找出来而成一个集合，记为 G_f。再把 f 对 G_f 中任一成员 g 进行操作之后而产生的状态（记为 $f(g)$）全部收集来而成集合，记为{$f(g)$}。这样一来，原先的问题 (S,F,G) 就归约为三个子问题：(S,F,G_f)，({g},{f},{$f(g)$}) 及 ({$f(g)$}, F,G)。如图 2.23(a)所示。其中第二个子问题 ({g},{f},{$f(g)$}) 是**本原问题**，因为只经过一次关键操作 f 就

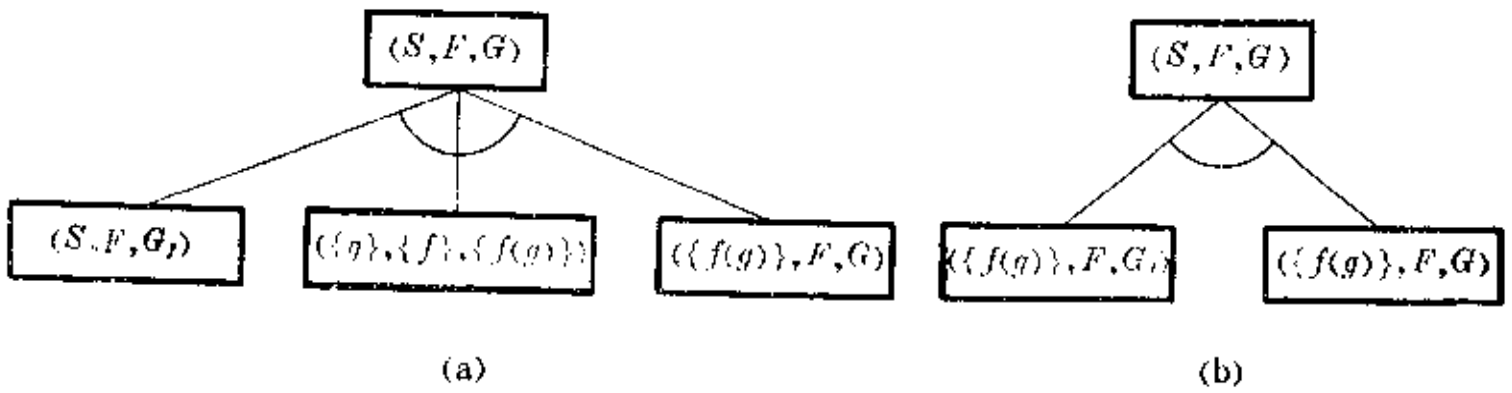


图 2.23 问题归约的关键操作法

可解决了。为了节省篇幅，今后在图中略去不画。如图 2.23(*b*) 所示。甚至于在叙述中有时也略去这个子问题不讲出来。但不要误会为“凡是本原问题都一律可略去”。

例 2.12 图 2.1 梵塔问题的关键操作 f 就是$M(C,1,3)$。而该操作能作用得上的状态 g 只有状态 (122) 。而 $f(g)$为状态 (322) 。因此原先的问题 ({(111)},F,{(333)}) 归约为三个子问题：({(111)},F,{(122)}),({(122)}, {M(C,1,3)}, {(322)}) 及({(322)},F,{(333)})。或简写为二个子问题：({(111)},F,{(122)})及({(322)},F,{(333)})。实际上，问题仍未最终解决，因为这二个子问题不是本原问题。还必须再对它们进行归约。归约后所得结果如图 2.24 所示。从图中可看出．最终的子问题都是本原问题。因此原先问题被解出。

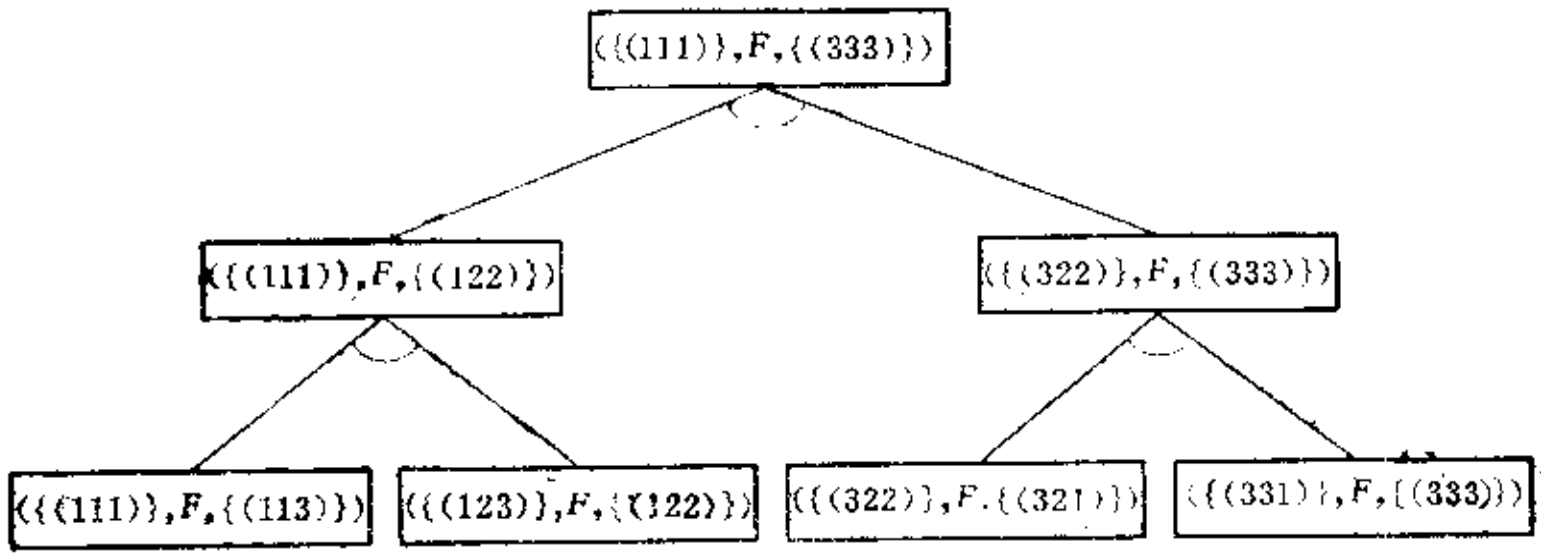


图 2.24 梵塔的一种归约图

例 2.13 例 2.10 的 G 有二个：(222)和(333)。则其归约图就如图 2.25 所示。图中起始节点的两条有向枝没有用圆弧连接起来。这种分枝表示其指向的子节点之间的关系是“**或**”关系。称其子节点为**或节点**。即只要其子节点中的某一个能求解的话，该节点就能最终得到求解。图 2.25 也可称为与-或图。

在寻找关键操作的过程中，不见得找到的关键操作都是能解得出的或是最佳解的。所以还是有个搜索问题。因此说得更确切一些，它们是**候选的关键操作**。

寻找候选的关键操作 寻找候选的关键操作的一种方法就是预测出关于问题 (S,F,G) 的差异。放宽地讲，关于 (S,F,G) 的差异是这样的部分清单，即“S 的成员为什么不能满足 G”这样推论的部分清单。如果 S 的某些成员是在 G 中，问题已解出了，也就没有差异了。例如，如果目标集合 G 由状态中的某些条件的集合来定义，而且如果∈S的某些条件满足这些条件的某几条但不是全部条件，那么，差异可以就是 s 不满足的条件的部分清单。如果这些条件的重要性能够分得出主次的话，就可以用最重要的不能满足的条件作为差异。

而后就寻找和每个可能的差异有关联的某些状态空间操作或操作的集合。所谓“有关联的”是指只要用了某个操作之后就能消除那个差异，就称这个操作和那个差异有关联了。这些操作或操作的集合就是**候选的关键操作**。

例 2.14 猴子与香蕉问题。在房里地板上有一个箱子和一串香蕉吊在天花板。猴子只能登在箱子上才能取到香蕉。但箱子不在香蕉的下方。猴子只有将箱子推到香蕉的下方而后登上箱子再抓香蕉才行。现用归约法求解之。

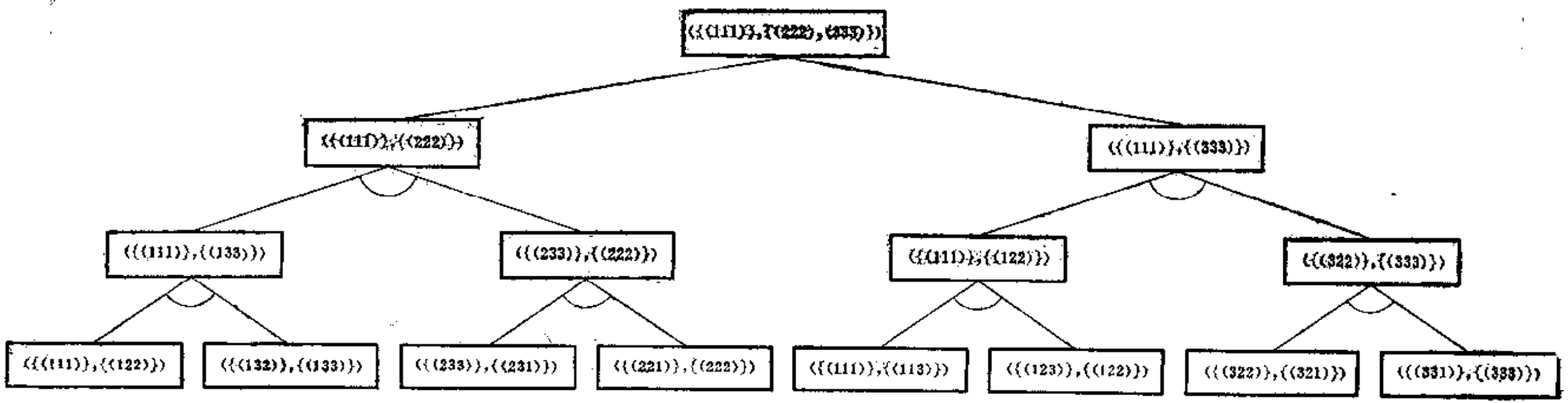


图 2.25　梵塔的另一种归约约图

用四元组（w,x,y,z）表示猴子和箱子在房里的位置以及是否抓到香蕉等状态。其中

w：是二维矢量，表示猴子在房里的水平位置；

x：其值为 1,0。分别表示猴子已在或不在箱子上；

y：是二维矢量，表示箱子在房里的水平位置；

z：其值为 1,0。分别表示猴子已抓到或未抓到香蕉。

为了表示水平位置，应将房里地板划分成适当密的有限的网格，根据这些网格来定坐标位置。如果设 c 是香蕉对应的水平位置的话，则（w,x,y,z）的目标状态为（c,1,c,1），即 $G=\{(c,1,c,1)\}$。

猴子可能执行如下四种操作：

(1)走到 u　　猴子走到水平位置 u

(2)推到 v　　猴子将箱子推到水平位置 v

(3)登箱　　猴子登到箱顶上

(4)抓到　　猴子抓到香蕉

把以上操作表示为状态空间变化形式，则 F 集就有以下四个成员（即 $F=\{f_1,f_2,f_3,f_4\}$）：

f_1　(w,0,y,z) $\xrightarrow{走到u}$ (u,0,y,z)

f_2　(w,0,w,z) $\xrightarrow{推到v}$ (v,0,v,z)

f_3　(w,0,w,z) $\xrightarrow{登箱}$ (w,1,w,z)

f_4　(c,1,c,0) $\xrightarrow{抓到}$ (c,0,c,1)

假设初态是：猴子和箱子的水平位置各为 a、b。则 S 集合只有一个成员（a,0,b,0）。即 $S=\{(a,0,b,0)\}$。这么一来，这个问题可表示成$(S,F,G)=(\{(a,0,b,0)\},\{f_1,f_2,f_3,f_4\},\{(c,1,c,1)\})$。

现在就可这样来归约这个问题（参阅图 2.26）：

首先，对起始状态和目标状态测出差异来。在(a,0,b,0)不能满足(c,1,c,1)的条件中，最重要的是该四元组中的最后一元 z 不为 1，这就是差异。与这差异有关联的操作就是 f_4。用 f_4 就可对原先问题归约为三个子问题：$(\{(a,0,b,0)\},F,G_{f_4})$，$(G_{f_4},\{f_4\},\{f(s_1)\})$和$(\{f_4(s_1)\},F,G)$。其中 G_{f_4}是允许 f_4 操作的状态集合。具体说来，它只有一个成员，$G_{f_4}=\{(c,1,c,0)\}$；而且$s_1\in G_{f_4}$。在此 s_1 就是(c,1,c,0)了。因此，$f_4(s_1)$实际上就是(c,1,c,1)，即 $\in G$。因而 $(\{f_4(s_1)\},F,G)$ 中的起始状态就是目标状态，没有差异，故是本原问题。为简便起见，在这个问题上，我们不仅将第二个子问题$(G_{f_4},\{f_4\},\{f(s_1)\})$(本原问题）省略掉，而且将三元组中的 F 元组也省略不写。那么以上就简述为：把$(\{(a,0,b,0)\},G)$归约为$(\{(a,0,b,0)\},G_{f_4})$和$(\{f_4(s_1)\},G)$。

其次，对$(\{(a,0,b,0)\},G_{f_4})$预测差异。(a,0,b,0)不能满足 G_{f_4}(即(c,1,c,0)) 的条件及对应的关键操作为：

箱子不在水平位置 c 上　　f_2　　(2.9)

猴子不在水平位置 c 上　　f_1　　(2.10)

猴子没有登在箱子上　　f_3　　(2.11)

抓住其中一个关键操作进行归约都可以。为说明问题方便，就利用这些关键操作依次地都产

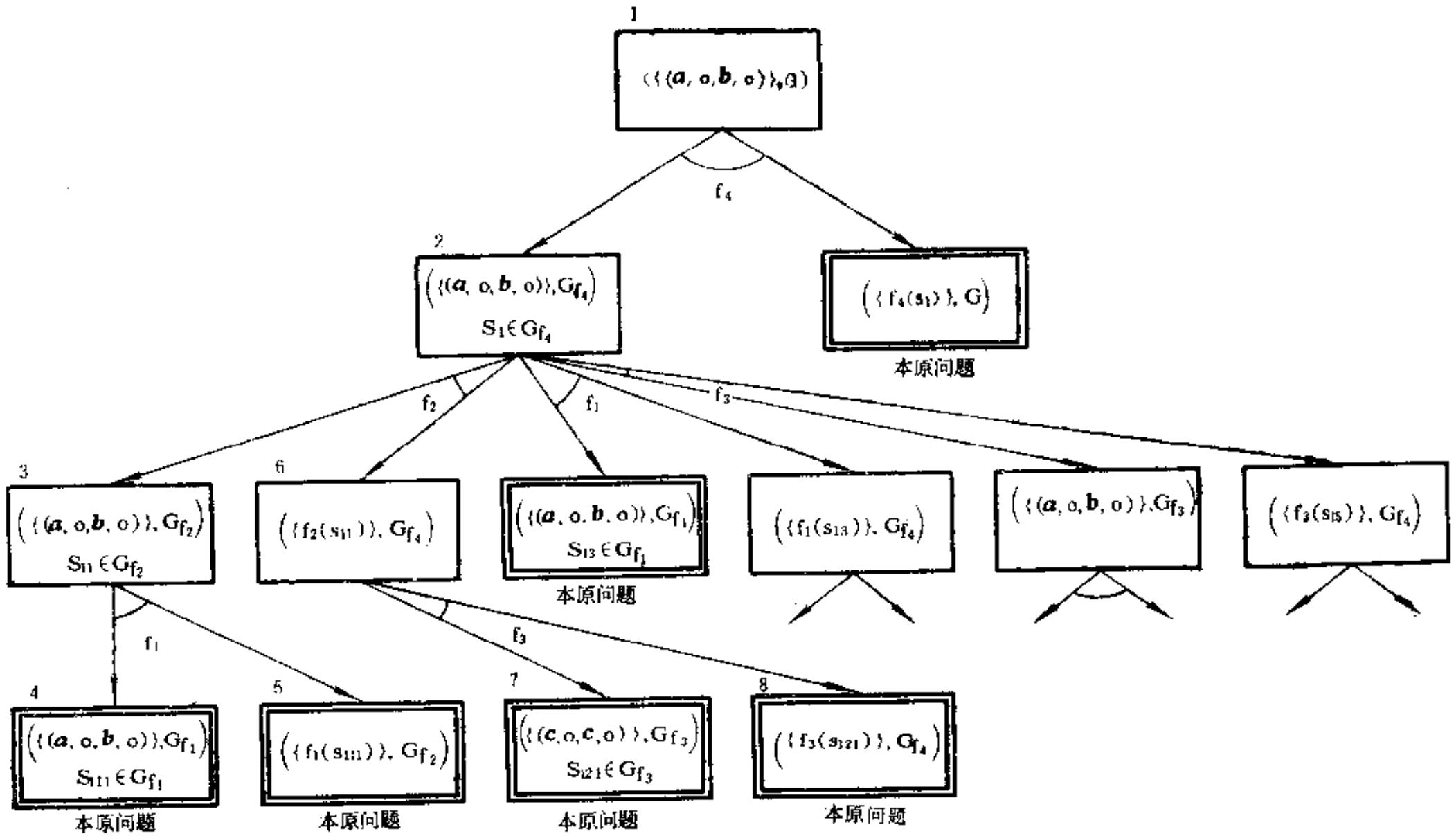


图 2.26　猴子与香蕉问题的与-或图

生供选择的归约子问题。用其第一个关键操作(式(2.9))来归约问题 $(\{(a,0,b,0)\}, Gf_4)$ 就得到

$$(\{(a,0,b,0)\}, Gf_2) \tag{2.12}$$

$$(\{f_2(s_{11})\}, Gf_4) \tag{2.13}$$

其中 $s_{11} \in Gf_2$。具体说来，Gf_2只有一个成员 $s_{11}=(b,0,b,0)$；而 $f_2(s_{11})$就是 $(c,0,c,0)$。

再对式 (2.12) 子问题预测。其差异是“猴子不在水平位置 b 上”，因此其关键操作为

f_1　　走到 b　　猴子走到水平位置 b

用这个关键操作将 $(\{(a,0,b,0)\}, Gf_2)$ 归约为子问题：

$$(\{(a,0,b,0)\}, Gf_1) \tag{2.14}$$

$$(\{f_1(s_{111})\}, Gf_2) \tag{2.15}$$

其中 $s_{111} \in Gf_1$。$Gf_1=\{s_{111}\}=\{(a,0,b,0)\}$；而 $f(s_{111})$就是$(b,0,b,0)$。可见式 (2.14) 及式 (2.15) 两式的子问题都没有差异，是本原问题。

同理，对式 (2.13) 所示的另一子问题也可归约，直至其子问题为本原问题而得到求解为止。

从图 2.26 中可以分析得出：其解的操作序列应是

（走到b，推到c，登箱，抓）

也可以对式 (2.10) 和式 (2.11) 的关键操作进行归约。如图 2.26 所示。其详细归约图，留给读者练习。

从图中也可看出节点之间的与-或关系。

2.3 与-或图法

从算符法和关键法中都可以看出，其归约过程的流程图都涉及到与-或图这个工具。有这个工具，对问题的归约过程能一目了然。因而有必要对与-或图作更进一步研究。

图 2.24、图 2.25 和图 2.26 所示的图都是与-或图。将它们简化之后就分别如图 2.27(a)(b)(d)的形式。特别是图 2.27(d)中节点 D、E 是与关系，F、G 是与关系，H、I 是与关系。但 D、E 跟 F、G，跟 H、I 的关系是或关系。即 B 的后裔节点中既有与节点又有或节点。为了把与、或关系归一化，可把它们改画为图 2.27(c)的形式。图中多增加了或节点 B_1、B_2 和 B_3。这么一来，同一级的节点都是与节点，或者都是或节点。因此，可做到使任何与-或图中的与节点及或节点一级级地交替出现。

与-或图不仅是上述几种方法的工具，也是归约的一种方法。甚至于在不知目标状态的情况下也可根据已知的操作进行归约，得出与-或图而求得解来。

例2.15　求解不定积分。归约与-或图中的起始节点就是原先问题的描述。操作（在此恰恰是运算规则）集合的成员有很多，有简单的积分公式，也有分部积分法，和的积分分解法则，代数与三角代换之类的变换规则等等。即使是简单的积分公式也有很多，略举几例如下：

$$\int du = u$$

$$\int u\,du = \frac{u^2}{2}$$

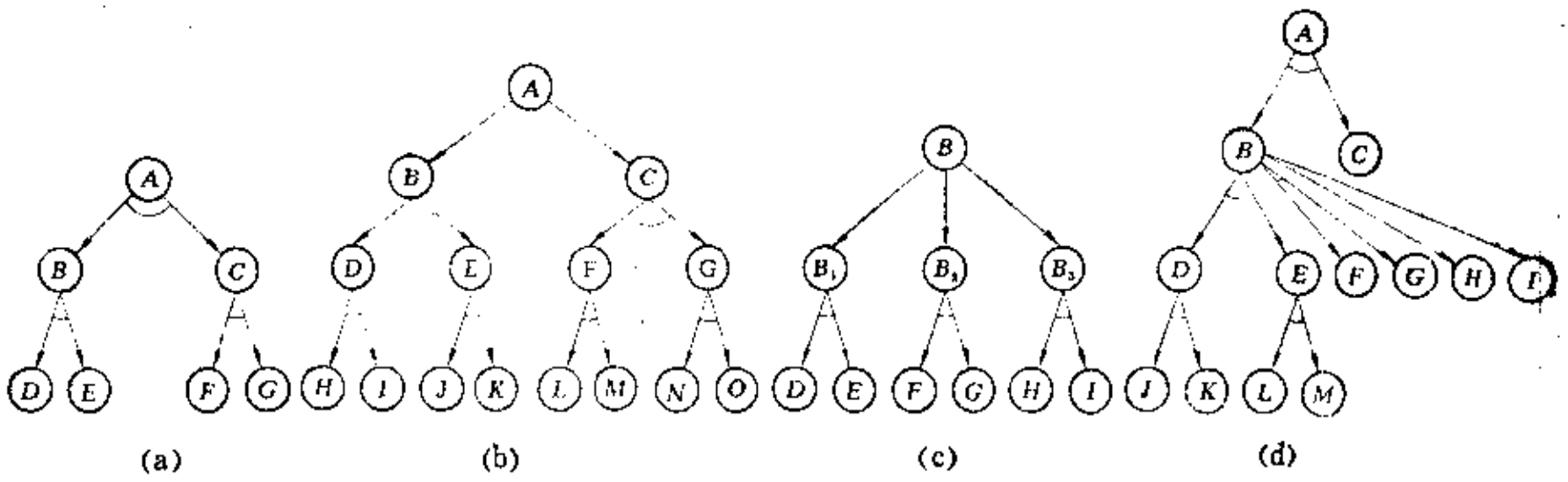


图 2.27　与-或图的几个例子

$$\int \sin u du = -\cos u$$

$$\int a^u du = a^u \log_a e$$

$$\int u^n du = \frac{1}{n+1} u^{n+1}$$

这些简单公式都可作为一种操作，被求解过的积分公式(例如积分表)都可作为操作集合之成员。也可以把等式左边的算式作为**本原问题**。

分部积分

$$\int u dv = u\int dv - \int v du$$

和的积分分解

$$\int \left(\sum_{i=1}^{n} f_i(x) \right) dx = \sum_{i=1}^{n} \int f_i(x) dx$$

常系数因式分解

$$\int k f(x) dx = k \int f(x) dx$$

……

类似于以上种种方法都不能作为本原问题，只能作为一种操作规则，可属于操作集合中的成员。这些方法用与-或图表示分别如图 2.28 所示。

对于欲求解的积分，可以选择对它可能进行操作的各种操作来试试，直到问题得到求解为止。从而得到一个与-或搜索图。

图 2.29 示出了积分

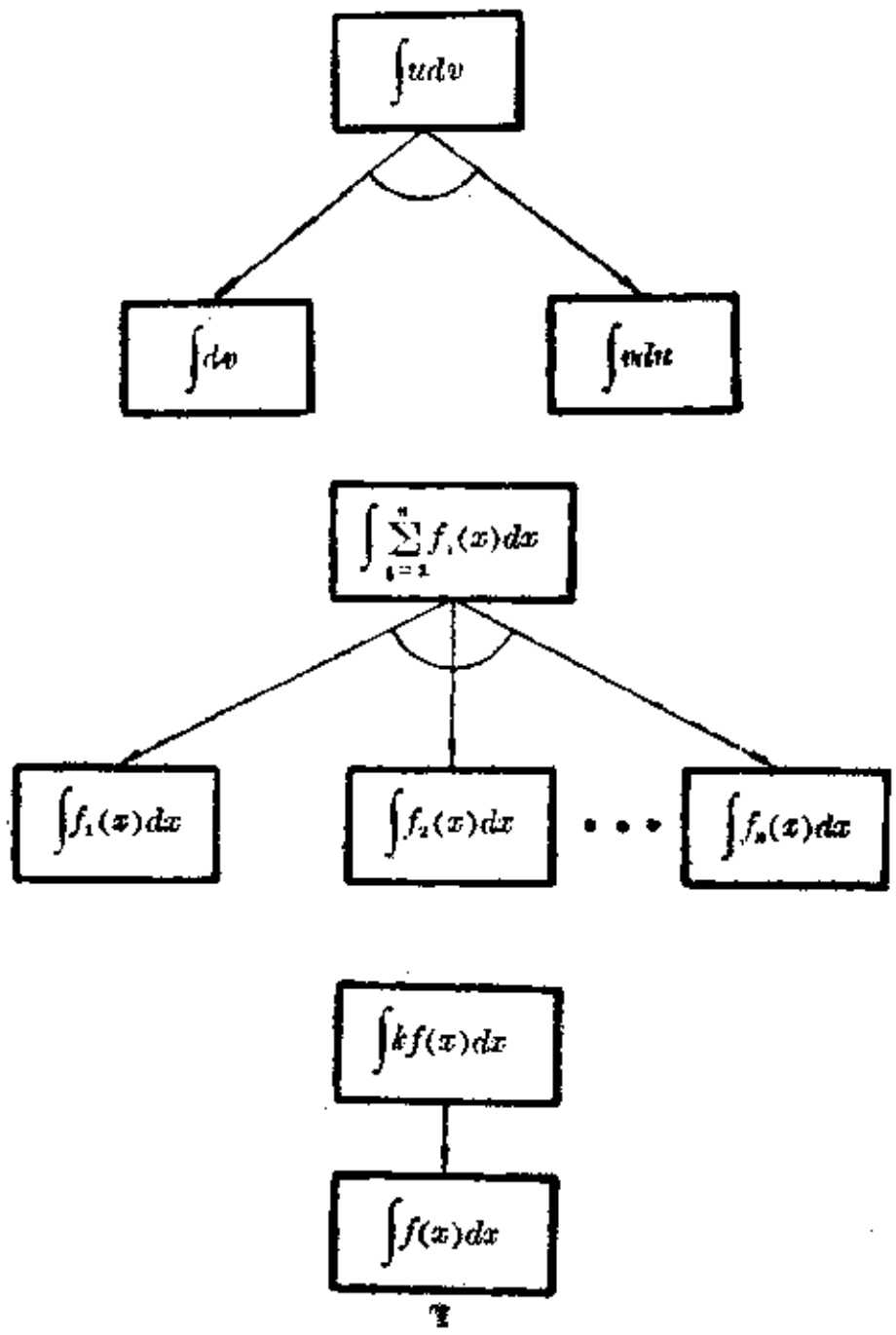


图 2.28　不定积分的某些操作的与-或图

$$\int \frac{x^4}{(1-x^2)^{5/2}} dx$$

的归约与-或搜索图。

图 2.29 中的节点表示问题描述。有双线框的节点表示终节点（对应于积分表中的积分）。粗线的枝示出了该问题的解图。从这个解图以及由积分表得到的积分中，可以确定：

$$\int \frac{x^4}{(1-x^2)^{5/2}} dx = -\tan(\arcsin x) + \frac{1}{3}\tan^3(\arcsin x) + \arcsin x$$

注意，此处所述的**终节点**是指对应于本原问题描述的那些节点。并不是所有的端节点都是终节点，因为有些端节点是不能解的，因而没有子节点。终节点是可求解的。

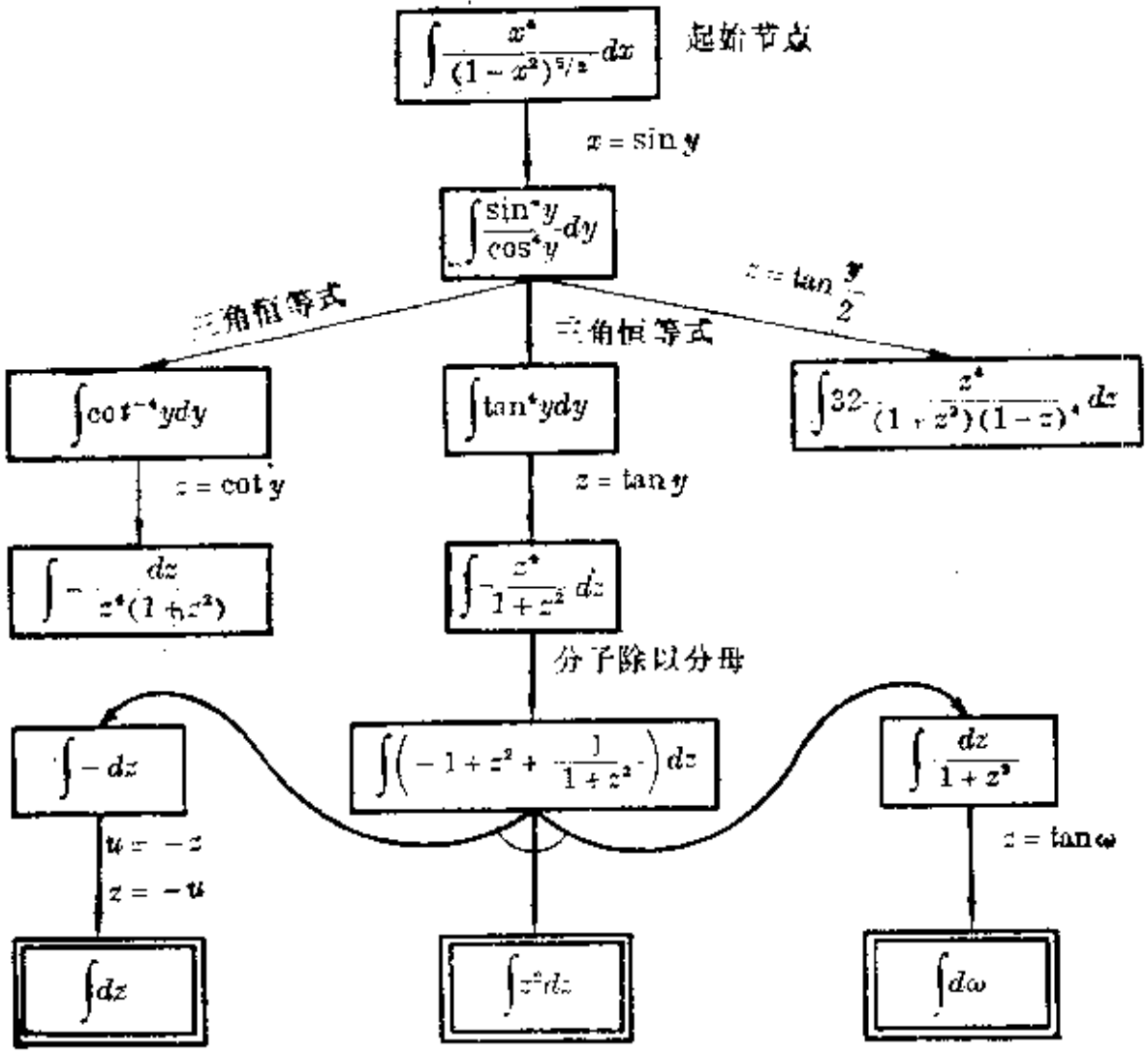


图 2.29 积分 $\int \frac{x^4}{(1-x^2)^{5/2}} dx$ 的与-或搜索图

能解节点与不能解节点 在与-或图中怎样才算起始节点是能解的节点呢？或是不能解节点呢？又应当怎样来判定呢？这是与-或图方法的重要内容。

与-或图中**能解节点**的一般定义可递归定义如下：

终节点都是能解节点（因为它们都和本原问题相关联）。

如果非终节点有“或”后接节点的话，当且仅当其后接节点中至少有一个是能解节点时，该非终节点就是能解节点。

如果非终节点有“与”后接节点的话，当且仅当其后接节点全部都是能解节点时，该非

终节点才是能解节点。

在所得的与-或图中就根据以上定义，从能解节点开始，而推导出“起始节点是能解的”。把这个过程中所涉及到的能解节点及有关的边组成一个子图。这个子图就是所需的解图。

在图 2.30 中就是与-或图的一些例子。图中终节点用字母 t 表示，能解节点用实心圆点表示，其解图用粗线边表示。

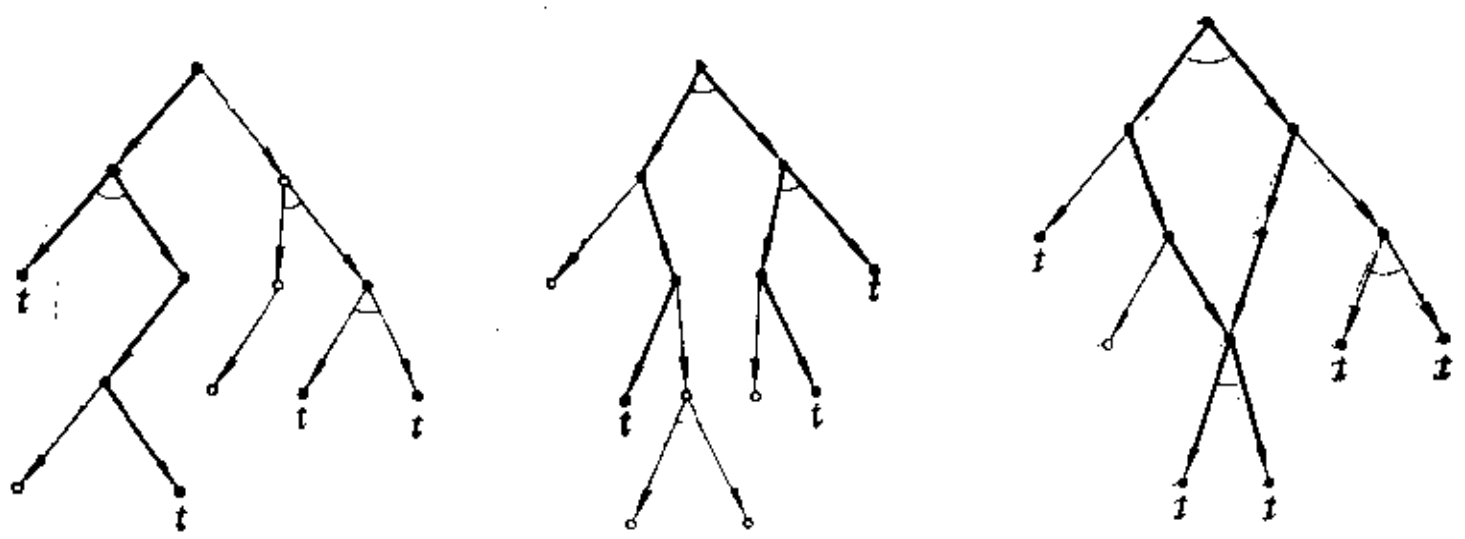


图 2.30　与-或图及其解图的一些例子

当与-或图中非终节点完全没有后接节点时，就说它是不能解节点。这样一些不能解节点的出现，可以意味着图中的其它节点（甚至于起始节点）也可能是不能解的。

不能解节点的一般定义可以递归定义如下：

没有后接节点的非终节点是不能解节点。

如果非终节点有“或”后接节点的话，当且仅当所有的后接节点都是不能解节点时，该非终节点才是不能解节点。

如果非终节点有“与”后接节点的话，当且仅当其后接节点中至少有一个是不能解节点时，该非终节点就是不能解节点。

可以简单递归或反复应用能解节点或不能解节点二定义，在与-或图中标记所有的能解或不能解节点。这样的操作过程分别称为**能解标记过程**或**不能解标记过程**。

一旦起始节点标记为“能解”，就胜利告终。其解图就是一个表示起始节点有解的仅仅含有能解节点的子图。

一旦起始节点标记为“不能解”，就以失败为告终。更确切地说，如果企图求解不能解的问题，或者并不确知原先问题是否能解时，起始节点被标记为“不能解”，应当说是成功了，而不是失败。不过，今后仍然称之为“失败”。

产生与-或图及解图的一般性步骤如下：

(1) 把原先的问题描述作为起始节点。

(2) 应用能适用的问题归约操作，预测出起始节点的后接节点集合。令 Γ 是预测节点的所有后接节点的组合操作。Γ 作用到某节点的过程，称为“**扩展**”**某节点**。(重申一下，如果产生“与”后接节点的多个集合，则应将每个非单一集合按类地分在居中的或节点下面。如图2.27(d)中的 B 节点就是这类节点，应按类地分成为图 2.27(c)。)

(3)设置表示从每个后接节点返回至该节点的指针。当企图标记能解和不能解节点时，

要用到这些指针。它们也指示出由终节点返至起始节点的解图。

(4)应用能解或不能解标记过程。

(5)扩展节点和设置指针的过程一直继续到起始节点能够标记为"能解"或"不能解"为止。

从上面的讨论中可知，与-或图的产生过程也有一个搜索过程。所形成的与-或图称为**搜索图**。为方便起见，我们仅讨论其特殊情况——与-或树。在搜索一棵与-或的过程中树就产生一棵子树，称为**与-或搜索树**。

后面介绍几种与-或树搜索法，与上节相似，故不多加说明。

2.3.1 与-或树的广度优先搜索法

与-或树的广度优先搜索法就是按节点产生的先后次序来扩展节点的，所以是顺序搜索法的一种。其过程的结构是十分简单的。可参阅图 2.31 所示的流程图。

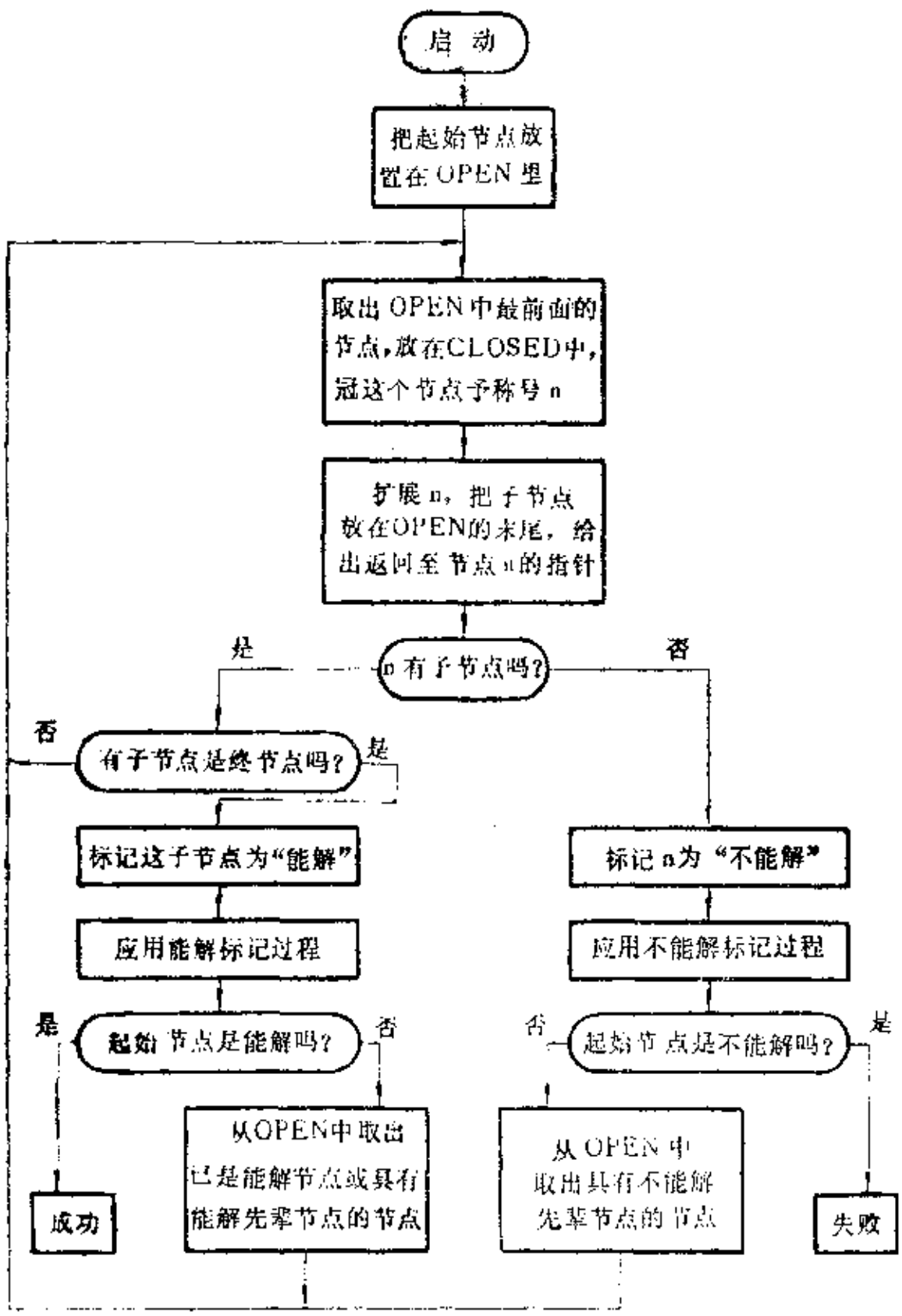


图 2.31 与-或树的广度优先搜索法的流程图

关于与-或树的广度优先搜索法的步骤如下。

(1) 将起始节点 s 放在 OPEN 表中。

(2) 取出 OPEN 表中最前面的节点，并放进 CLOSED 表中，冠这个节点予称号 n。

(3) 扩展节点 n，即产生所有的子节点。将这些子节点放在 OPEN 表的末尾。并且设置由这些子节点返回至节点 n 的指针。如果没有子节点，就标记节点 n 为“不能解”。且继续下步；否则，转至(8)。

(4) 对搜索树应用不能解标记过程。

(5) 如果起始节点标记上“不能解”，则失败退出；否则，继续下步。

(6) 从 OPEN 表中取出其先辈节点是不能解节点的所有节点。但不把它们放在 CLOSED 表中。（这个步骤就避免了一些不必要的气力。否则还会再花气力去求解这些节点，但它们对原先问题是否能解的结论并没有新贡献。）

(7) 转到(2)。

(8) 如果任一子节点是终节点的话，就标记这个子节点为“能解”，且继续下步；否则，转至(2)。

(9) 对搜索树应用能解标记过程。

(10) 如果起始节点标记了“能解”，则就能得到证明起始节点有解的解树；否则，继续下去。

(11) 从 OPEN 表中取出已是能解节点或其先辈是能解节点的所有节点。但不把它们放在 CLOSED 表中。（这一步骤就避免了一些不必要的气力，否则还会再费气力去求解这些节点，但它们对原先问题是否能解的结论并没有新贡献。只有在企图求解出多种方法的问题时才需花这些气力。）

(12) 转至(2)。

与-或树中节点的**深度**定义如下：

(1) 起始节点的深度为 0；

(2) 任何其它节点的深度是 1 加上它的父辈节点的深度。（该定义与 2.1.3 节中在树中节点的深度定义是一致的。）

可以从理论上加以证明，只要解树确实存在的话，刚刚所述的广度优先搜索过程保证能找到其最深节点是终节点的那种解树，而且深度最小。由于篇幅所限，在此从略。

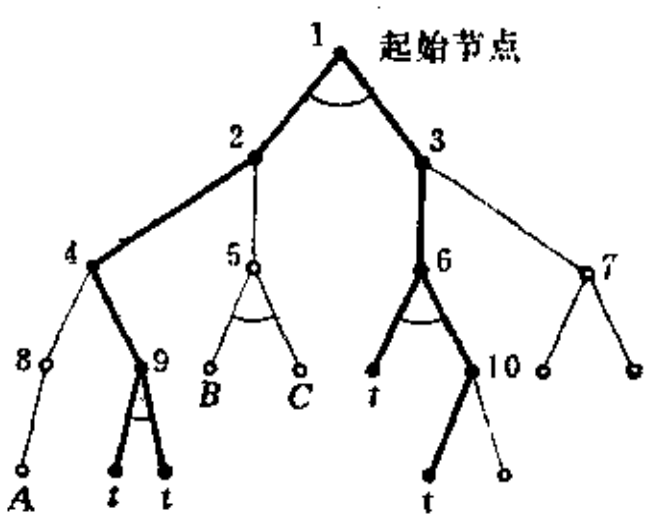


图 2.32 采用广度优先搜索法的节点扩展过程

在图 2.32 中示出了该法扩展节点过程的例子。图中节点标上的数字，表示节点扩展的顺序；能解节点用实心圆点标记；所找到的解树由粗线枝表示。注意，要理解：当扩展节点 9 而其子节点被识别出是终节点时，不仅节点 9、4和 2 被标记为能解节点，而且节点 A，B和 C 也从OPEN表中取走。不理解这点就不能理解上述步骤。

2.3.2 与-或树的深度优先搜索法

与-或树的深度优先搜索法是优先扩展最新近产生的节点，来寻找在某个深度限度内的

解树，比深度限度更深的节点就不再扩展了。因而恰好在深度限度上的节点是非终节点时，就标记上“不能解”。正如状态空间搜索法一样，深度限度有碍于寻找解，但是该过程却能得到在深度限度范围内的任一解。

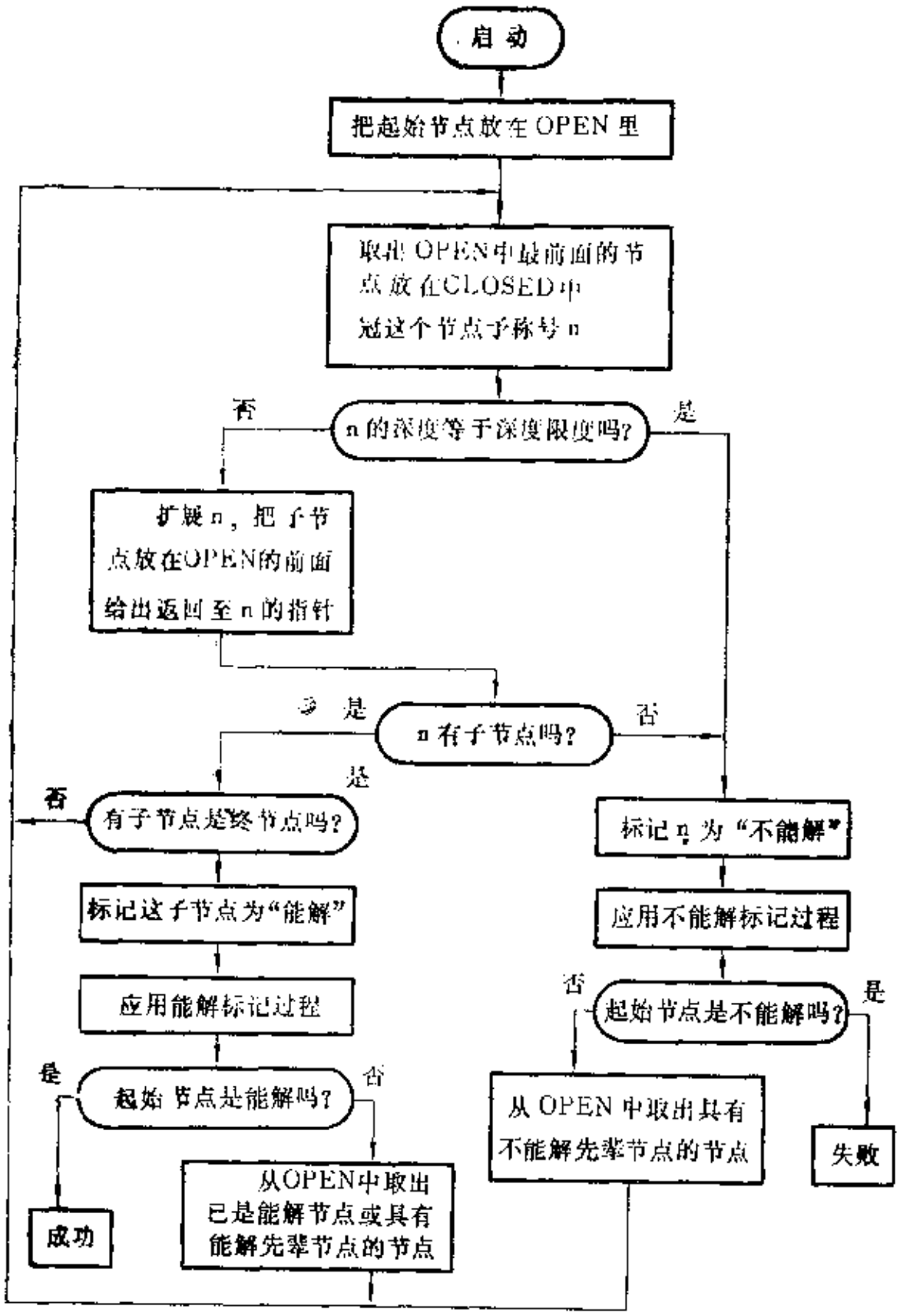


图 2.33　与-或树的深度优先搜索法的流程图

深度优先搜索法的流程图示于图 2.33 中。其步骤如下：

(1) 将起始节点 s 放在 OPEN 表中。

(2) 取出 OPEN 表中最前面的节点，并将它放在 CLOSED 表中，冠这个节点予称号 n。

(3) 如果 n 的深度等于深度限度的话，就标记节点 n 为“不能解”且转至 (5)；否则，继续下步。

(4) 扩展节点 n，即产生其所有的子节点。将这些子节点放在 OPEN 表中的前面，并

且设置由这些子节点返回至节点 n 的指针。如果没有子节点，就标记节点 n 为“不能解”，且继续下一步；否则，转至 (9)。

(5) 对搜索树应用不能解标记过程。

(6) 如果起始节点标上了“不能解”，则失败退出；否则，继续下一步。

(7) 从 OPEN 表中取出其先辈节点是不能解节点的所有节点。

(8) 转至 (2)。

(9) 如果任一子节点是终节点的话，就标记这个子节点为“能解”，且继续下步；否则，转至 (2)。

(10) 对搜索树应用能解标记过程。

(11) 如果起始节点标上了“能解”，则就能得到证明起始节点为有解的解树，成功而退出；否则，继续下去。

(12) 从 OPEN 表中取出已是能解节点或其先辈节点是能解节点的所有节点。

(13) 转至 (2)。

在图 2.34 中示出了该法扩展节点过程的例子。在此例中，深度限度设为 4 。图中能解节点用实心圆点表示；不能解节点用空心圆点表示；解树用粗线枝表示。节点标上的数字表示节点扩展的顺序。

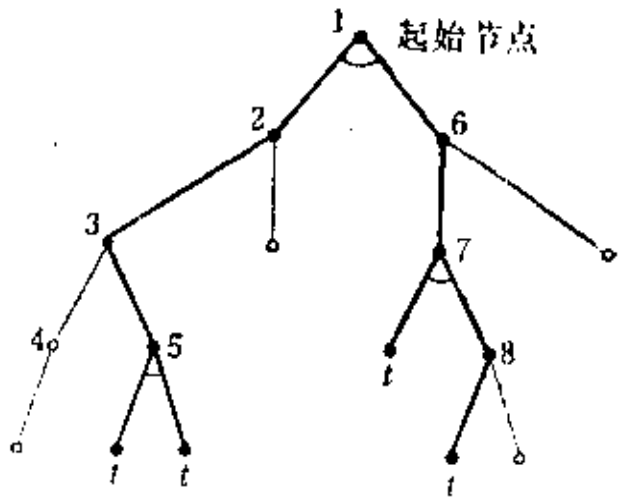


图 2.34 采用深度优先搜索法的节点扩展过程

从上两种搜索法都仅讨论与-或树的情况。然而在许多情况下产生的与-或图不是树，而是有回路的与-或图，即有的节点具有二个以上的前接节点。

如果在图结构中可以把实际上相同的问题描述看作为不同的树表示法的话，则可以把可能出现但不必选上它的回路消除掉，同样可得到解树。如图 2.35(a)所示，其粗线边为一种解树。当然，节点 1 和节点 3 …，这一支路也可以是另一种解树。

有些回路是不可消除的。如图 2.35(b)(c) 所示。特别是图(c)，它无法消除回路。一旦消除，问题也无法解出。故无解。可见，由于与节点而产生的回路是无法消除的。

图2.35(a)中回路没有消除，但解图却是存在的。如图中的粗线边所示。

但是有时也可用简单的方法来消除那些或节点的回路。即扩展节点时，不产生已是自己的前接节点的那些后接节点。从而得到解图。如图 2.35(b) 中的节点 5，不必产生节点 3 作为自己的后接节点，同样可得到解图。

当试图描述关于不是树的与-或图的搜索过程时，就出现了若干复杂问题。首先，深度的定义比起树来就复杂一些。统一的定义如下：

(1) 起始节点的深度为 0 。

(2) 任一节点的深度是 1 加上它的最接近的前接节点的深度。

其次，设置指针也复杂。如果从一个节点需要返回指向它的多个前接节点的话，一根指针不够用，需要有多个指针的指针场。如图 2.35(b) 中从节点 2 返至节点 1 和 3 的指针在解图中都是需要的。另一方面，指针场的某些指针在解图中仍是不需要的。如图 3.35(b) 中从节点

3 返回至节点 5 的指针确实不需要。因此在搜索与-或图过程中必须具有**分析搜索图**的**指针**的能力，把不相干的指针抛弃而且保留那些确定解图所需要的指针。

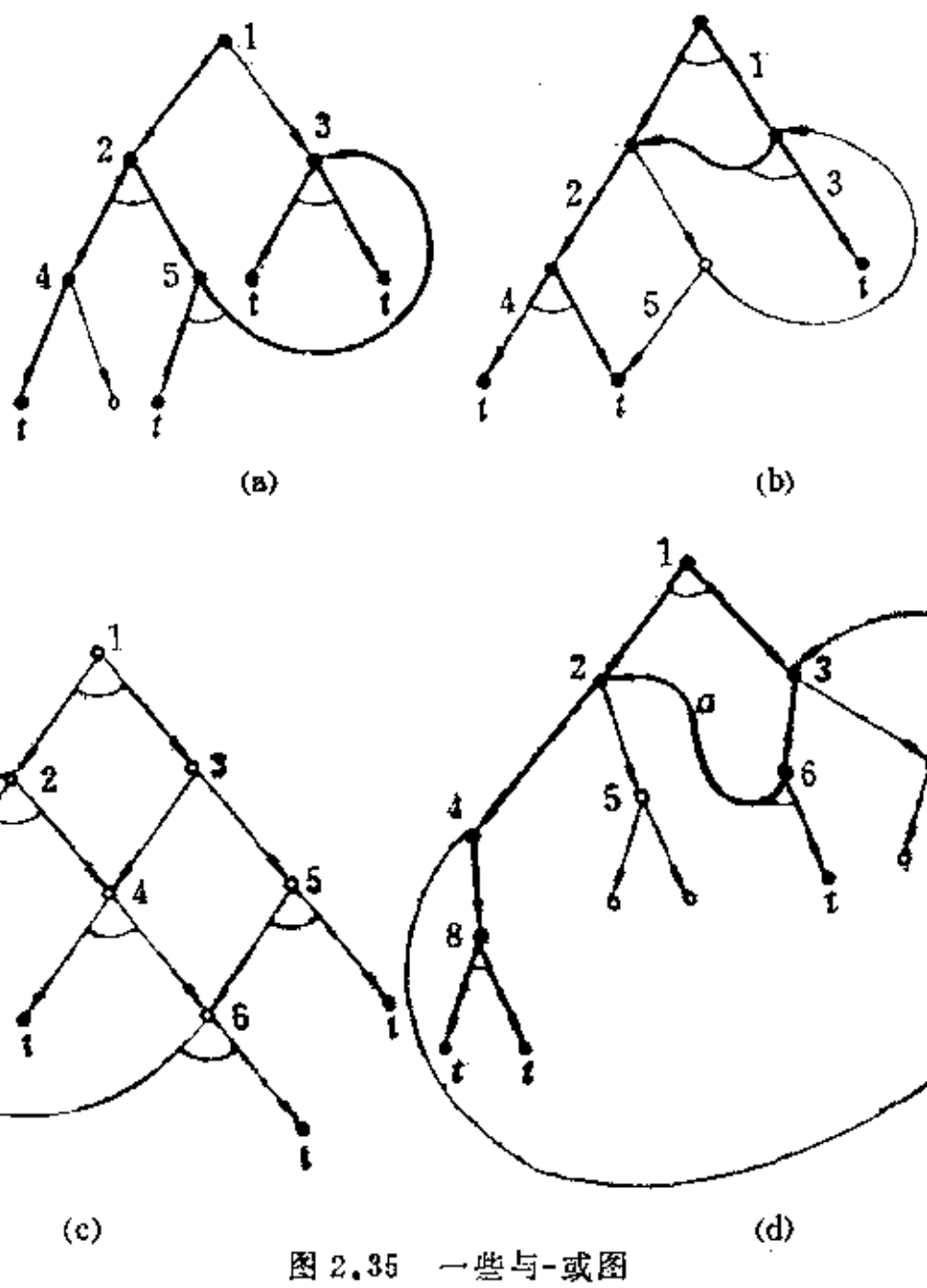


图 2.35　一些与-或图

再者，当产生了其后接节点时，必须要检查一下，其后接节点中是否有已在 CLOSED 表中或以前已标记了“能解”或“不能解”的节点；如果某个已经能解（或不能解）的后接节点又由另一条路被涉及到的话，那么就要应用一下能解（或不能解）标记过程，查一查是否起始节点为能解（或不能解）的。

搜索一般与-或图的这种方法，留给读者练习。

2.3.3　与一或图的有序搜索法

与-或图的广度优先搜索法和深度优先搜索法同样也都是盲目搜索法。与状态空间有序搜索法相似，与-或图也有有序搜索法。在状态空间有序搜索法中，采用了评价函数来确定节点扩展的先后顺序。而确定这个评价函数的关键办法是抓住启发函数。**启发函数**是对从某节点到目标节点的最佳路的耗散进行估计的。在与-或图中也有相应的办法。但是这些函数的具体定义根据各自不同的特点而有所不同。在本小节中先介绍一下具体定义，而后再介绍算法。

与-或树的耗散　两节点 n_i、n_j 之间路的耗散总和称为该两节点间的路耗散，记为 $C(n_i,n_j)$。

由于与-或图中节点之间有“与”、“或”关系，而在解树中的节点也有“与”关系。所

以其解树的耗散也有两种定义。

第一种，将解树中所有的枝耗散的总和定义为**"和耗散"**。如图2.36 所示。图中示出的两棵解树，其和耗散分别为 20，18。

第二种，将在解树中从起始节点至各个节点的各个路耗散之中最大者定义为该解树的**最大路耗散**。如图 2.36 的解树 A 中从起始节点至终节点的路耗散有 9 和 15。其最大路耗散为 15。同理，解树 B 的最大路耗散为 17。

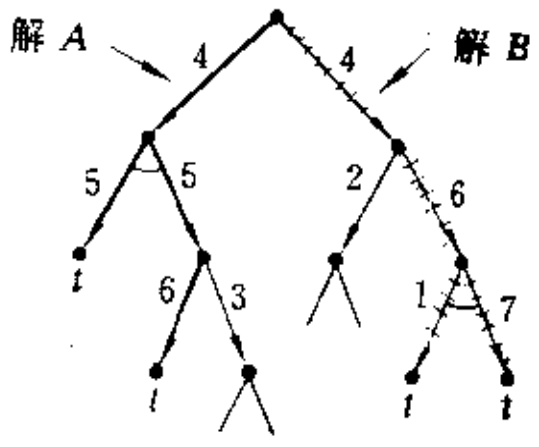


图 2.36　两种解树及其耗散

我们希望能找到这样的解树，它在整棵完整的与-或树范围内具有最小的耗散。称这种解树为**最佳解树**。用 $h(s)$表示以起始节点 s 为根的最佳解树的耗散。因此，推广至以任一节点 n 为根的解树的最小耗散，则可用 $h(n)$表示。此处的 $h(n)$ 之具体含意如下：

(1) 如果 n 是终节点（对应于本原问题），则

$$h(n) = 0$$

(2) 如果 n 是一个具有 k 个"或"子节点 $n_1, \cdots, n_k$ 的非终节点，则

$$h(n) = \min_{1 \leqslant i \leqslant k} [c(n, n_i) + h(n_i)]$$

(3) 如果 n 是一个具有 k 个"与"子节点 $n_1, \cdots, n_k$ 的非终节点，则

$$h(n) = \sum_{i=1}^{k} [c(n, n_i) + h(n_i)] \qquad \text{（和耗散）}$$

$$h(n) = \max_{i} [c(n, n_i) + h(n_i)] \qquad \text{（最大耗散）}$$

当然，如果 n 是一个不能解节点，则 $h(n)$ 没有定义。

我们较关心的是要搜索出一棵最小耗散的解树，是其和耗散最小，还是其最大耗散最小，则可根据具体问题的要求而定。

启发函数　在未求出解树时，$h(n)$是无法准确求出的，只能估计而已。在搜索解树过程中，只要节点 n 不是不能解节点，就可对 $h(n)$ 进行估计。令 $\hat{h}(n)$是 $h(n)$的估计。称 $\hat{h}$ 为**启发函数**。利用它可以确定节点的扩展次序。

整个搜索过程就是一个将搜索树逐步扩展的过程，即扩展搜索树的端节点的过程。

搜索树的端节点不外乎下列三种情况：

(1) 搜索过程中发现端节点已是终节点。

(2) 搜索过程中发现端节点是非终节点，而且它不可能有子节点。

(3) 搜索过程中发现端节点是非终节点，但其子节点还未产生。

对于搜索树中的任一节点 n，启发函数 $\hat{h}(n)$如下确定：

(1) n 是一个端节点：

(a) 如果 n 是终节点，则 $\hat{h}(n) = 0$。

(b) 如果 n 是非终节点且没有子节点，则 $\hat{h}(n)$ 没有定义。

(c) 如果 n 是非终节点，但它的子节点还未产生，则具体地根据与-或树所表示的问题领域的启发信息来确定 $h(n)$ 的这个估计 $\hat{h}(n)$。

(2) 如果 n 是一个具有“或”子节点 $n_1,\cdots,n_k$ 的非端节点，则

$$\hat{h}(n)=\min_i[c(n,n_i)+\hat{h}(n_i)] \tag{2.16}$$

(3) 如果 n 是一个具有“与”子节点 $n_1,\cdots,n_k$ 的非端节点，则

$$\hat{h}(n)=\sum_{i=1}^{k}[c(n,n_i)+\hat{h}(n_i)] \quad \text{(和耗散)} \tag{2.17}$$

$$\hat{h}(n)=\max_i[c(n,n_i)+\hat{h}(n_i)] \quad \text{(最大耗散)} \tag{2.18}$$

期望解树 在扩展节点的过程中，只要该节点具有“或”子节点，那么根据将来的扩展结果，每个或子节点都有希望长成以 s 为根的解树。就暂称它们为“希望”解树。如果有“与”子节点，当然也有希望长成“希望”解树。因此，除了已经标记为“不能解”的节点以外，搜索树的任何节点 n 都有希望长成以 s 为根的解树。而从根 s 开始至节点 n（包括其子节点）的所有搜索子树，都分别是“希望”解树的近根部分。还可以根据 $\hat{h}(n)$ 值，再从这些近根部分挑选出估计它是以 s 为根的最佳解树的近根部分(称之为**期望解树**，记为 τ_0)。因而 τ_0 是最有期望的“希望”解树。

τ_0 的具体定义除了需应用 $\hat{h}$ 值外，还有如下几条：

(1) 起始节点是在 τ_0 中。

(2) 如果节点 n 是在 τ_0 中，则

(a) 如果在搜索树中节点 n 有“或”子节点 $n_1,\cdots,n_k$，则具有 $[c(n,n_i)+\hat{h}(n_i)]$ 的最小值的那个子节点，才是在 τ_0 中（如果比较不出大小，则任选一个）。

(b) 如果在搜索树中节点 n 有“与”子节点，则所有的子节点都是在 τ_0 中。

τ_0 是该启发搜索过程的关键概念。在搜索过程中必须记录搜索树中的每个节点上的 $\hat{h}$ 值，才有可能挑选出 τ_0 来。而将 τ_0 中还未扩展的端节点进行扩展（扩展深度根据具体问题而定），而后再重新计算一下这个成长过的搜索树的 $\hat{h}$ 值。又再重新挑选 τ_0。反复进行至得到解树为止。

例 2.16 求图 2.37(d)所示的解树。

在该例中，假设我们所希望的是要得到和耗散最小的解树，而且每相邻接的两点 n_i，n_j 之间的耗散都定义为 1 [$c(n_i,n_j)=1$]，扩展深度为 2。

一开始就从起始节点 s 出发来扩展节点，如图 2.37(a) 所示，得到了“或”节点及端节点 B，C 和 E，F。对端节点的 $\hat{h}$ 值进行计算，分别得到 $\hat{h}(B)=3$，$\hat{h}(C)=3$，$\hat{h}(E)=3$ 及 $\hat{h}(F)=2$，如图中节点旁边的数字所示。而节点 A 和 D 的 $\hat{h}$ 值根据式 (2.17) 计算而分别得到 $\hat{h}(A)=8$，$\hat{h}(D)=7$。节点 A、D 是关于 s 的或节点，根据式 (2.16) 计算而得 $\hat{h}(s)=8$。此时，期望解树 τ_0 应当挑选由节点 s，D，E，F 所组成的子树。注意，E 和 F 是与节点，必须同时选上。如图 2.37(a) 的粗线枝所示。此时 τ_0 的端节点为节点 E、F。

再对节点 E 进行扩展。参看图 2.37(b)。重新计算一下 τ_0 中节点的 $\hat{h}$ 值。可以发现，

原来对于节点E的估计有出入了。此时，期望解树 τ_0 应挑选节点 S，A，B，C 所组成的子树较好。应该修正。故 τ_0 立即修改为图 2.37(*b*) 中的粗线枝所示的子树。

再先后扩展 τ_0 的端节点 B，C。设先扩展节点 B。发现节点 H、I 是终节点。在进行能解标记过程中，节点 G、B 都标上“能解”（图中用实心点表示）。如图 2.37(*c*) 所示。重新计算 τ_0 的 $\hat{h}$ 值。τ_0 没有必要重新大改动。这样一直进行到如图 2.37(*d*) 中所示为止。其解树用粗线枝表示。图中节点旁边加圆圈的数字，表示标记“能解”的顺序。

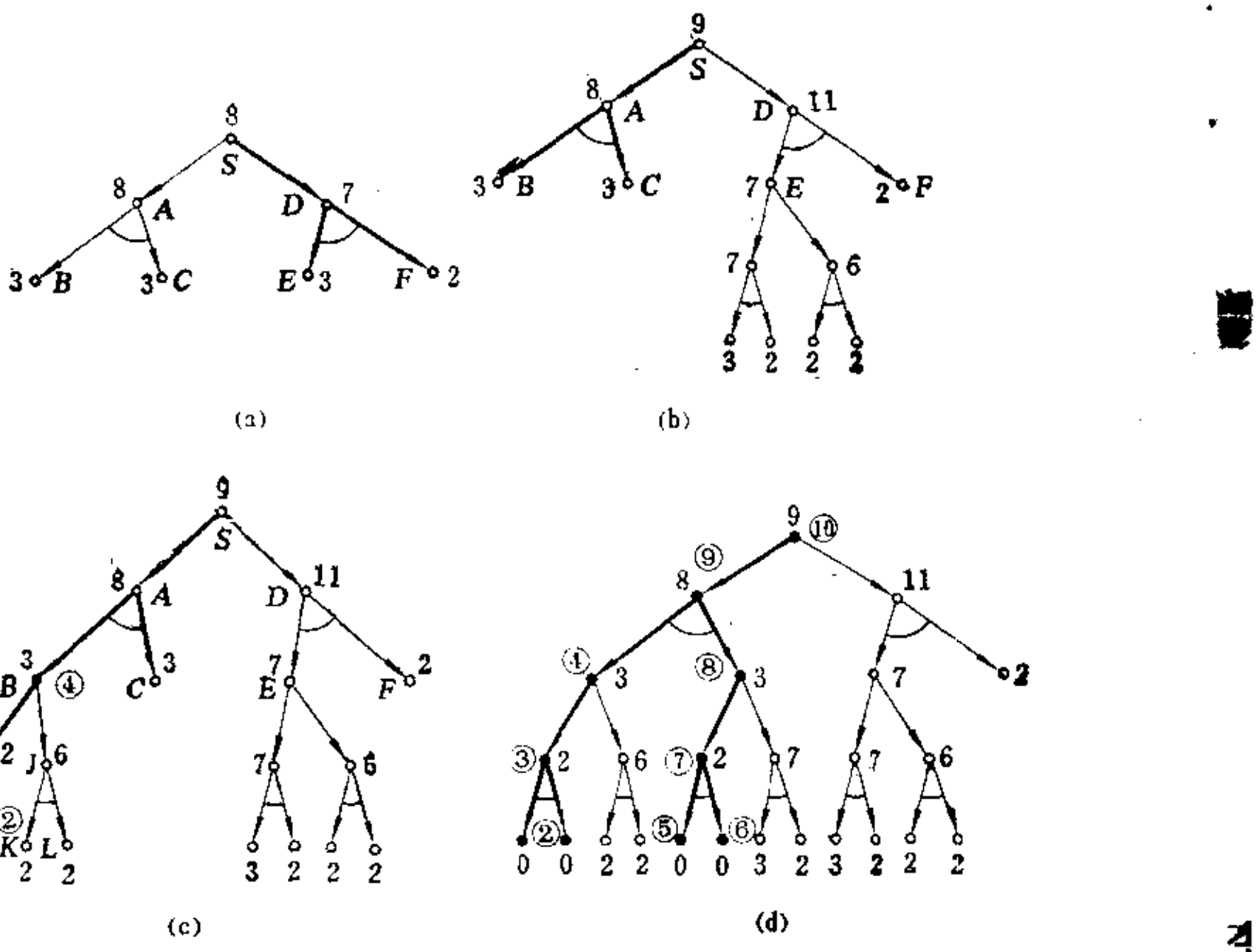


图 2.37　用有序搜索算法产生搜索树的过程

关于 $\hat{h}$ 值的计算问题可根据具体情况而定。甚至于可以如状态空间法中采用 $\hat{f}=\hat{g}+\hat{h}$ 来求。此处不赘述了。

与-或树的有序搜索算法　它与广度优先搜索法略有差别。其步骤如下（其流程图见图 2.38）：

(1) 将起始节点 *s* 放在 OPEN 表中，且计算 $\hat{h}(s)$ 值。

(2) 计算期望解树 τ_0，即用搜索树中节点的 $\hat{h}$ 值来估计一下以 *s* 为根的最佳解树的近根部分是哪个。并选进 τ_0 中。

(3) 把已放置在 OPEN 表中的 τ_0 的端节点选出来且放在 CLOSED 表中，冠这个节点予称号 *n*。

(4) 如果 *n* 是终节点，则标记 *n* 为“能解”，且继续下步；否则，转至 (9)。

(5) 对 τ_0 应用能解标记过程。

(6) 如果起始节点标上了“能解”，则解树就存在于 τ_0 之中；否则继续下步。

(7) 从 OPEN 表中取出其先辈节点为“能解”的所有节点。

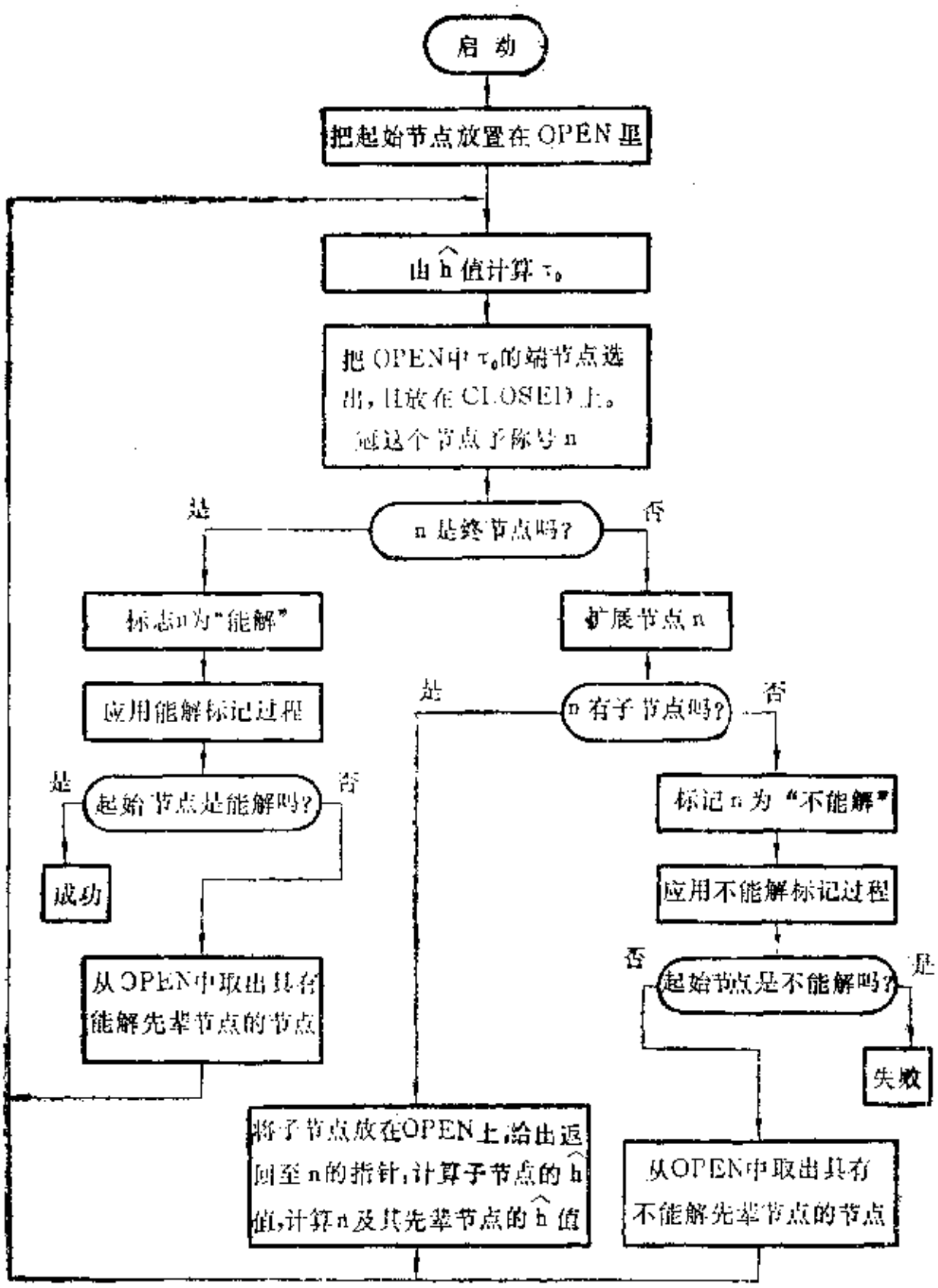


图 2.38　与-或树的有序搜索算法的流程图

(8) 转至 (2)。

(9) 对节点 n 进行扩展操作，产生 n 的所有的子节点。在此设扩展深度只有 1。如果没有子节点，则标记 n 为“不能解”，且继续下步；否则转至 (14)。

(10) 对 τ_0 应用不能解标记过程。

(11) 如果起始节点标上了“不能解”，则失败；否则，继续下步。

(12) 从 OPEN 表中取出其先辈为“不能解”的所有节点。

(13) 转至 (2)。

(14) 将这些子节点放在 OPEN 表中且设置由这些子节点返回到 n 的指针。计算这些子节点的 $\hat{h}$ 值。重新计算 n 和 n 的先辈节点的 $\hat{h}$ 值。

(15) 转至 (2)。

2.4 博 奕 树

博奕是“赌博”与奕棋之类具有对策性的课题。主要是研究在博奕过程中对方做过的操作都知道，而且自己想做什么操作、主观上可以选择的博奕。而对于那些随机性的“靠运气”的博奕（如掷骰子之类）我们并不感兴趣。因为这不是智能工作。因而本课题是根据已知的格局，研究自己应采用符合所定规则（制约条件）的何种操作（即策略）才能使自己不被对方所击败，甚至取胜对方。

2.4.1 归约博奕图

问题归约方法也能用于某些博奕中。该方法是通过求证自己博奕必胜的过程中，寻找出获胜的策略。

假设郑某与傅某二人在博奕，我们要找出使郑某获胜的博奕策略。用 X^t 表示轮到 t 博奕时的格局。在此 X^+ 表示轮到郑某博奕的格局，而 X^- 表示轮到傅某博奕的格局。用 $W(X^t)$ 表示欲求证郑某从 X^t 格局中能取胜的问题。

博奕的合法的若干着法提供了归约博奕问题为子问题的格式。假设现在轮到郑某奕的格局是 X^+，而且此时郑某有 N 个合法的着法 X_1^{+1}，X_2^{+1}，…，X_N^{+N}。这几个着法中有的对郑某自己有利，有的对自己不利。此时主动权完全掌握在郑某人手中。他只要能挑选一种对自己最有利的着法就行。所以为了证明 $W(X^+)$，必须能够而且只需证明 $W(X_i^{+i})$的某个即可。可见 $W(X_i^{+i})$之间是“或”关系。另一方面，如果轮到傅某奕的格局是 X^-，而且此时傅某有M个合法的着法 Y_1^{-1}，Y_2^{-2}，…，Y_m^{-m}。这M个着法的选择权不掌握在郑某手中。从郑某角度考虑问题，应估计到傅某总是会选择对郑某最不利的一着。可见，这M个着法中只要有一着能让郑某失利，郑某都可能失败。因此为了证明$W(X^-)$，必须能够证明所有的$W(Y_i^{-i})$才行。故 $W(Y_i^{-i})$之间是“与”关系。当然，当我们试图仅仅证明可能和局时，类似的归约也可用。

用这些问题归约操作就产生与-或图之类，称之为**博奕图**。另外，所谓**本原问题**就是由自己获胜的博奕终局规则来给定的。证明的过程可以继续到终结于本原问题而找到解图为止。

例 2.17　郑某和傅某在玩分火柴棒游戏。有几堆火柴棒，轮到谁玩，他就必须把其中的某一堆分成数量不等的两堆。两人依次轮流地玩。谁首先玩不下去，谁就输。

显然，玩到最后必定是每一堆恰好只有二根或一根火柴棒，因为在这种情况下不是分成数量都是一根的两堆，就是无法分成两堆，都不符合规则。

假设开始时只有一堆共七根火柴，而且由傅某先玩。这个游戏的结构就是表示各堆中的火柴棒数目的一序列数字，再加上接着轮到谁玩的正负号。因此，(7^-)就是表示开始的结构。而终局（无法再玩下去了）只有三种结构:(2,1,1,1,1,1)，或(2,2,1,1,1) 或 (2,2,2,

1)。其中每个数字表示一堆的火柴棒数目。

从（7^-）中，傅某有三种可选择的玩法，即（$6,1^+$）、（$5,2^+$）或者（$4,3^+$）。而后轮到郑某玩。应用问题归约过程，就可得到如图 2.39 所示的与-或图。图中的粗线边表示解图，本原问题用双框节点表示。从图中可看出，不管开始傅某选择何种玩法，郑某只要抓住关键格局（$4,2,1^-$）就必定获胜。

如果改为让郑某先玩，则无法证明W（7^+）。后玩者必胜。

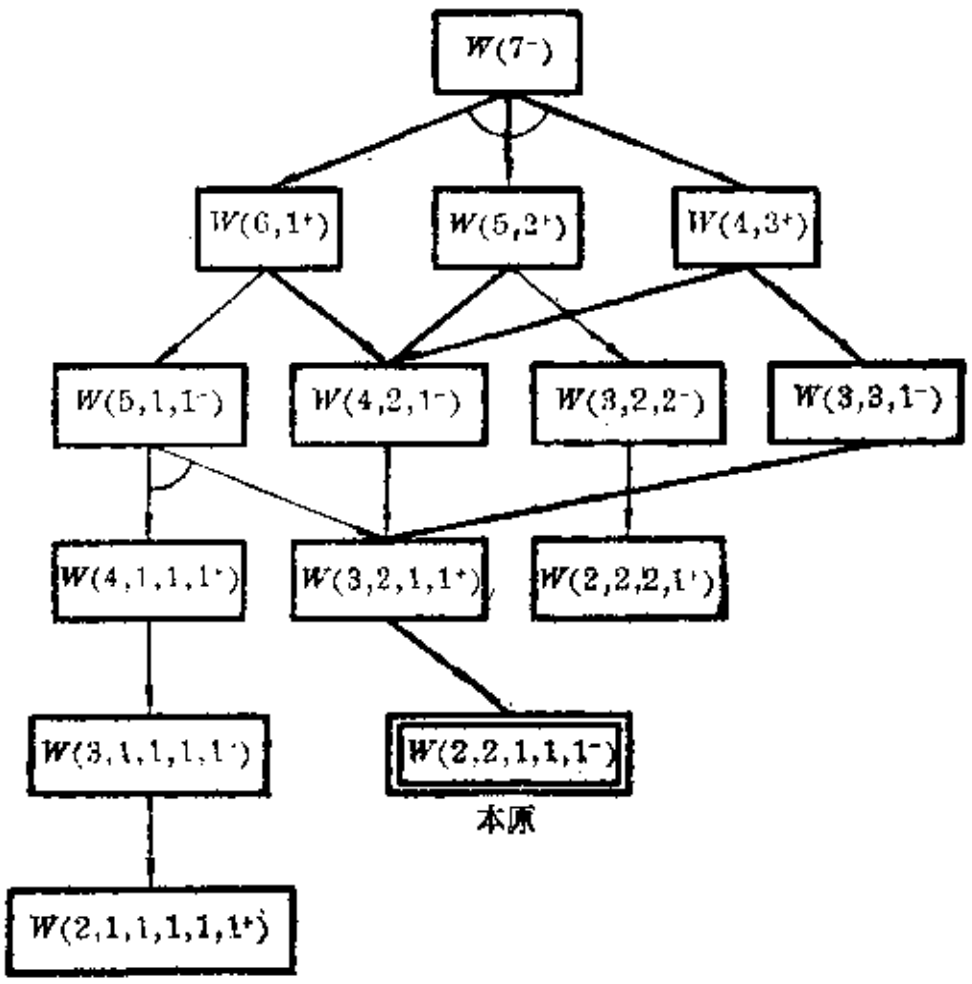


图 2.39　分火柴棒游戏的与-或图

2.4.2　最小最大值搜索法

按照上述归约方法，对于一般的博奕是不可能做到的。就看一个简单的井字游戏。如图 2.40(a) 所示的九空格图，由郑、傅二人对奕。轮到谁奕，谁就可在任一空格上放上自己的一只棋子。谁先使自己的棋子能够组成三子一直线（如图 2.40(b) 所示）就得胜。如果把所

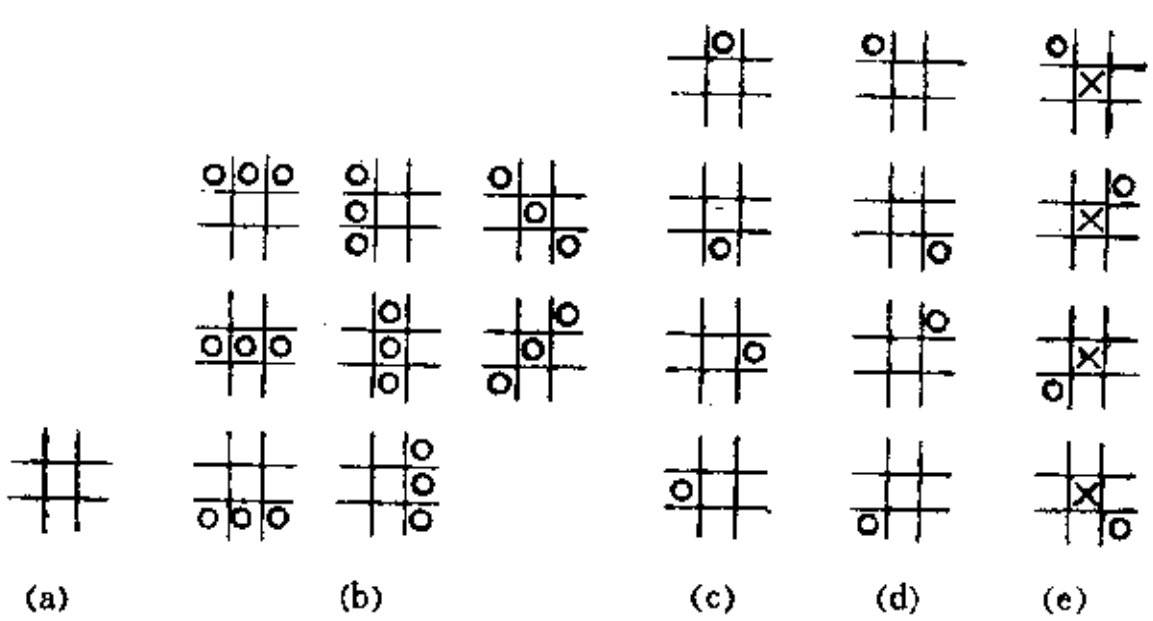


图 2.40　井字游戏

有可能的着法都归约出来的话，有 $9! = 362880$ 种着法。当然在某一方早已取胜时，后面的着法就无效了。另外，有些情况是对称的，如图 2.40(c)(d)(e)，各可看成是同一着法。但就算它的着法的三分之一，也是惊人的。还有人计算过，跳棋和国际象棋分别有 10^{40} 和 10^{120} 种着法。如果计算机产生某格局的一个后裔格局只需三分之一毫微秒的话，也需要花费 10^{21} 世纪的时间才能完成完整的跳棋与-或树。而太阳系的寿命只有 15×10^{7} 世纪。因此，穷搜索是不行的，要有能肯定胜或负的结论也是办不到的，只能是估计而已。

但上述的博奕图也给我们有益的启示。我们欲求的博奕图总是从郑某能取胜的角度出发的。但是不能只考虑眼前的一着棋。高明的棋手总是深思熟虑的。他在奕一着棋时，至少要考虑到：自己有几种着法，而每种着法之后对方可能采用几种对策，而且对方一定采取对自己最不利的对策，在这种对自己最不利格局下，自己还有几种可能着法，……这么反复思考之后，才举棋下了一着。未思虑成熟就会举棋不定，而草率从事则必败。

这样的思虑过程就是求解一棵“与”“或”相间的搜索与-或树的过程。愈是深思熟虑，在搜索树中最深的端节点的深度就愈深。但是有许多因素限制了这种智能。如运算次数及存贮空间限制，以及评价函数如何求得等等。

可以采用前述的广度优先、深度优先及有序搜索等方法来生成这种**搜索博奕与-或树**（或称**博奕树**）。在博奕树搜索结束之后，还必须从博奕树中选出估计是“最好”的一着作为对策。这种估计可以应用对博奕树的端节点求评价函数的方法来求得。评价函数必须考虑到现在格局的价值以及对将来的影响。如象棋，不仅要考虑现行格局中各棋子的价值以及各棋子所在位置的价值，还要考虑对将来局势有何影响等等。例如，一般说来车比马、炮、兵价值高，但是一旦车被困必被捕获时其价值也就接近于 0 了。

通常在计算评价函数时，对自己一方（如郑某）较有利者，得分为正，而对对方（如傅某）较有利者，得分为负。自己必胜，则得分为 $+\infty$，自己必败，则得分为 $-\infty$。整个格局估计下来，正分愈大，对郑某愈有利；负分愈大，对郑某愈不利；接近于 0 时，则对双方都差不多有利。要选择最好的一着，可以采用**最小最大值搜索方法**。现在以井字游戏来具体说明这个过程。

例 2.18 由郑某先奕的井字游戏的最小最大值搜索法。郑某为了奕一着棋，起码要得到如图 2.41 所示的博奕树。此处，搜索深度只有二级。×表示郑某放的棋子，○表示傅某放的棋子。还忽略了一些对称情况。第一级（子节点）是郑某可能选择的几种着法，所以是或节点；第二级（孙节点）是傅某可能选择的几种着法，所以是与节点。现在必须着手考虑傅某可能选择的格局中对郑某的利弊如何，即对该博奕树的端节点进行估计，求其评价函数。

在此游戏中某格局 P 的评价函数 $e(P)$ 定义如下：

(1) 如果 P 不是一种胜负可定的格局，则

$$e(P) = e(+P) - e(-P)$$

其中 $e(+P)$ 表示在格局 P 中所有空格都放上郑某的棋子之后，郑某的棋子可能组成三子一直线的数目。同理，$e(-P)$ 与上述类似，则表示傅某的得分了。

(2) 如果 P 对于郑某取胜，则

$$e(P) = +\infty \qquad (\infty\text{表示一个足够大的数})$$

(3) 如果 P 对于傅某取胜，则

$$e(P) = -\infty$$

依上述定义，对图 2.41 中的端节点分别求出 $e(P)$ 值。如图中端节点下方的算式所示。

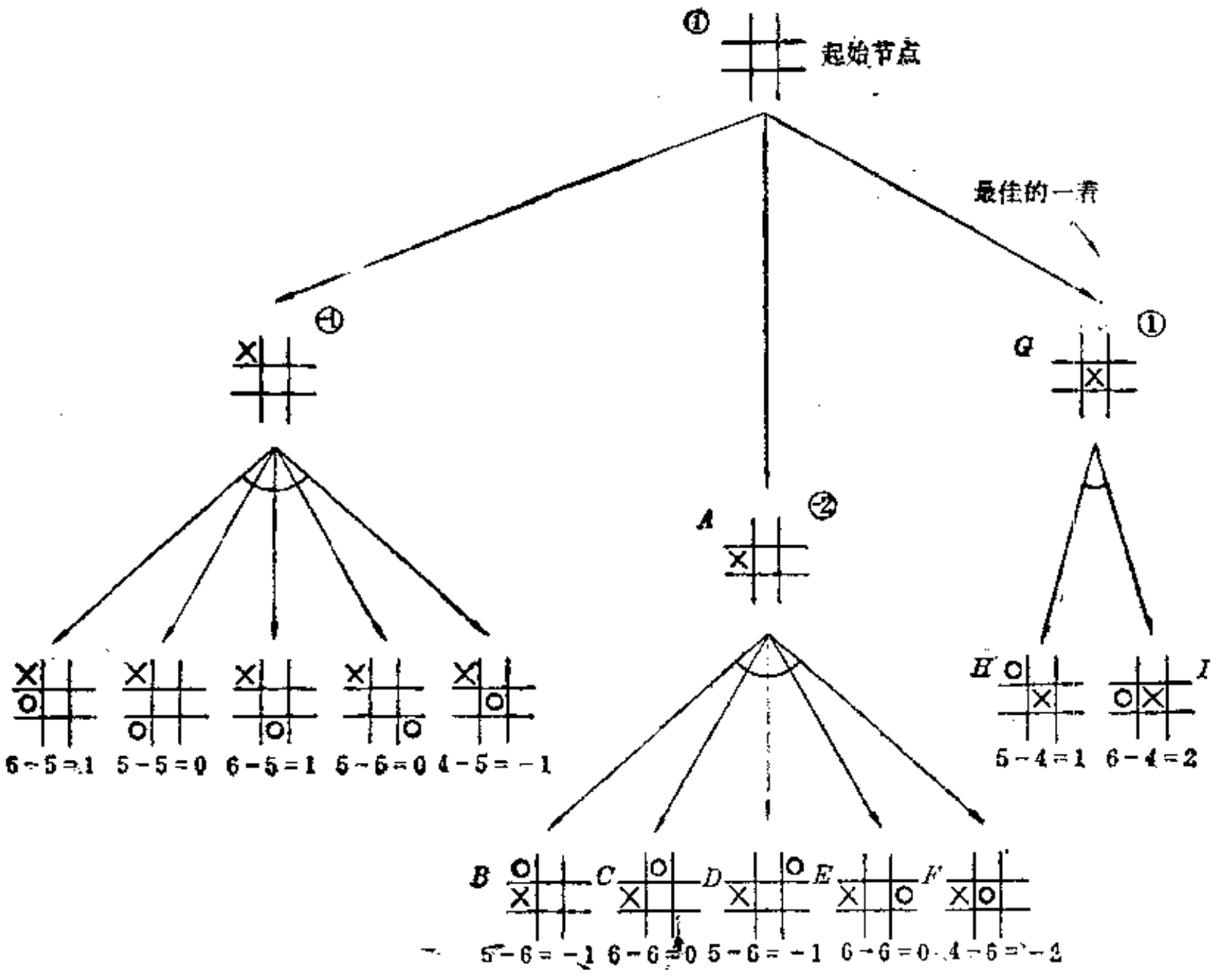


图 2.41　井字游戏的最小最大值搜索（第一着）

而后返回向上计算评价函数。由于在此图中端节点之间是“与”关系，是傅某可能选择的格局。对郑某而言，应考虑对自己最不利的情况。故由这些端节点（如图中的端节点 B、C、D、E、F）“返回”至其父节点（如图中的节点 A）的评价函数值，应取这些节点(B、C、D、E、F）中最小的评价函数值。在此，其最小值为 e(F) 。所以节点 A 的评价函数值为－2。这值称为**返上值**。图中用加圆圈的数字表示各节点的返上值。而称端节点的评价函数值为**静态评价函数值**。

由第一级节点再返回向上至其父节点(即起始节点)的返上值应这样求得：此处，第一级节点之间是“或”关系，是郑某自己可以选择对自己最有利的一着而奕。所以应取第一级节点中评价函数值最大者作为起始节点的返上值。图中起始节点的评价函数值取为 1 。

总之，凡是其子节点是或节点的节点，其返上值应取其子节点返上值中的最大者。称这一级为**取最大值级**。凡是其子节点是与节点的节点，其返上值应取其子节点返上值中的最小者。称这一级为**取最小值级**。博奕树中“与”“或”节点常是一级一级相间出现的。

经过对图 2.41 的分析，郑某开始奕的最好一着应是节点 G。

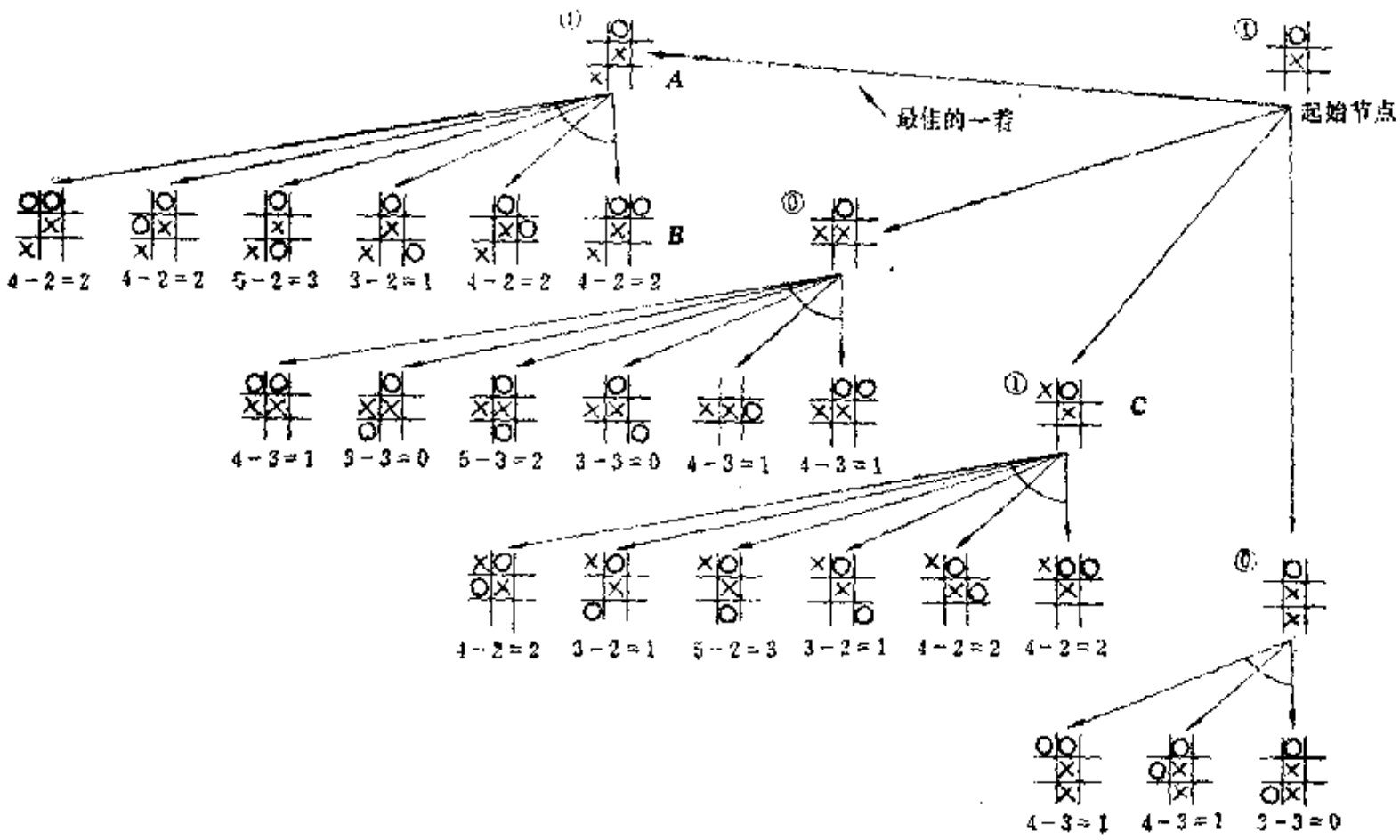


图 2.42 井字游戏的最小最大值搜索（第二着）

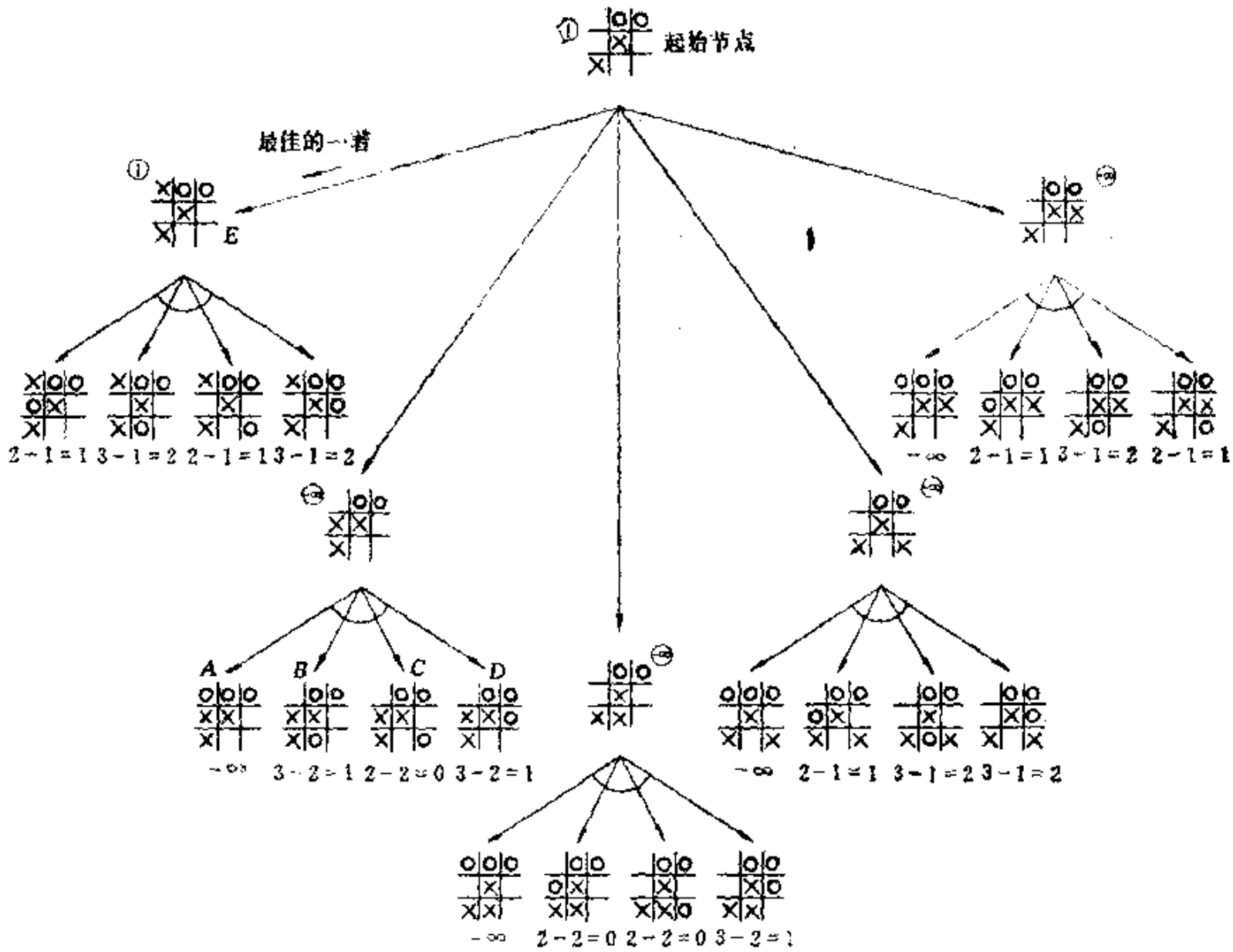


图 2.43 井字游戏的最小最大值搜索（第三着）

接着，轮到傅某奕棋。如果傅某也是聪明人，他必会选择图 2.41 中的节点 H 那一着作为对策。假设他不甚聪明，而是选用图中的节点 I 作为他的一着来回敬郑某。

此时又轮到郑某奕棋。郑某又采用博奕树搜索技术，立即以图 2.41 中的节点 I 作为起始节点，产生深度为 2 的博奕树，并计算出各端节点的静态评价函数值及其他节点的返上值，如图 2.42 所示。可见郑某应选择图中节点 A 或 C 的任一着来奕（因两节点的返上值相同）。设郑某选择 A。

傅某接着选择图 2.42 中节点 B 的那一着作为对策。

随后郑某又产生第三着的博奕树。如图 2.43 所示。并选择最佳的一着（该图中的节点 E）。其余四着都必败。

因此，郑某选择节点E这着之后已成必胜定局了。无论傅某采用何种对策都无法挽回他的局败了。

2.4.3 α-β 搜索法

刚刚所述的搜索法把搜索树生成的过程与格局评价函数的求解过程完全脱离开来。仅在树生成之后才开始求格局的评价函数值。这样，各节点都要生成。但是值得注意的是：如果求端节点的评价函数值以及计算返上值与树的生成同时进行的话，有可能简化搜索过程中的许多工作，而且同样找到最佳的一着。

从图 2.43 可看出，如果端节点一产生就立即求其评价函数值的话，那么产生节点 A 并求其评价函数值（为 $-\infty$）之后，也就没有必要再去产生节点 B、C 和 D 了，更不必求它们的评价函数值了。因为节点 A、B、C、D 是与节点，其父节点的返上值是取最小值的，而 $e(A)$ 为 $-\infty$。$e(B)$、$e(C)$ 和 $e(D)$ 值再怎么小，最多也是 $e(A)$ 值而已。其父节点的返上值总是 $-\infty$。因此可将节点 B、C、D 修剪掉，不必再去产生这些节点了。现用一具体例子来说明这种修剪技术，而后再加以归结。

例 2.19　博奕树的修剪技术。设图 2.44 所示的是一棵完整的博奕树。图中的或节点用圆圈表示，与节点用双圆圈表示。如果采用类似深度优先搜索法，一开始就从起始节点出发，每节点只扩展一个子节点，一直搜索到深度限度（此处为 4）的端节点，并且计算其评价函数值为 0。如图中所示的①（图中节点附近的加圆圈数字表示求其估价函数值及返上值的步骤）。由于该端节点是与节点，可以推测，其父节点的返上值不可能超过该端节点的评价函数值 0。所以可肯定该父节点的返上值一定是 $\leqslant 0$。但不能肯定它的返上值一定为 0。故该值 0 称为**候选返上值**（PBV）。

再搜索和端节点有“与”关系的其他端节点，并求出其评价函数值为 + 5，如图中③。由于没有其他有“与”关系的端节点，故可肯定其父节点的返上值为等于 0 了（如图中④）。

接着，将返上值 0 推测为祖父节点的候选返上值（PBV）。由于这个祖父节点是取最大值级的节点，所以将来的返上值必定是不小于 0（如图中⑤所示）。可见给 PBV 加上不等号是很恰当的。这可以直观地知道其发展趋势。

这个祖父节点还有其他后裔。因此需要再往其后裔节点搜索，直至某个端节点为止。如图中⑥所示，其评价函数值为 - 3。因此其父节点的 PBV $\leqslant -3$（如图中⑦所示）。可以看出，祖父节点的返上值只能取 0，不可能再大了（如图中⑧所示）。注意，节点⑦的其它子节点就不必产生了。博奕树的这一分枝就被修剪掉了。图中的细线树枝表示被修剪的树枝，

而节点上加记号×表示修剪的部位。

这种修剪技术可以再继续使用下去。例如到了第⑫步之后，在推测第⑬步时就可在节点⑫处加以修剪。

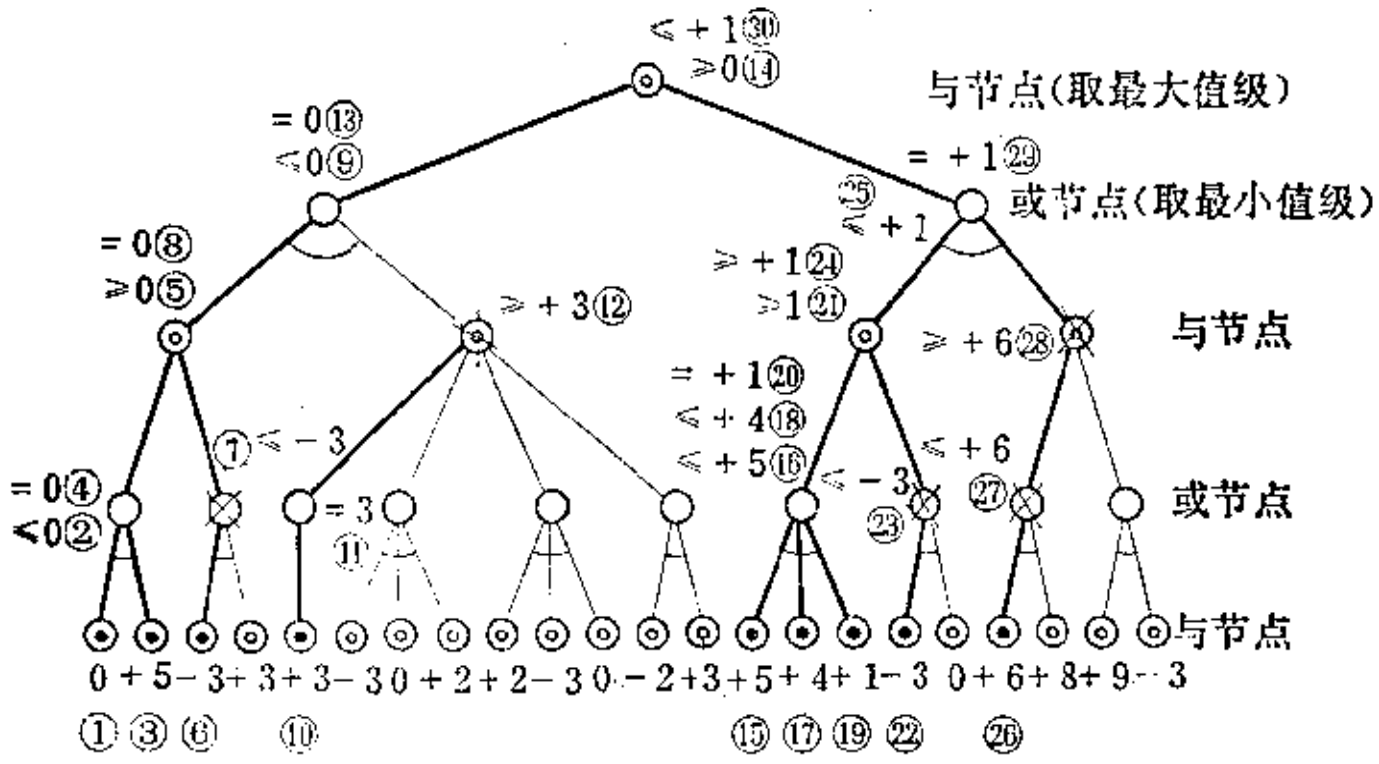


图 2.44 博奕树的修剪过程

但是不能乱修剪。例如第⑯步之后，不仅没有在节点⑯处修剪，而且还要继续搜索到节点⑲，还将其评价函数值 + 1 返上到始节起点，并最终定为起始节点的返上值。这是由于在第⑭步时确定了起始节点的 PBV ≥ 0 。而后搜索到节点⑮⑰甚至于⑲时，它们的评价函数值分别为 + 5 ， + 4 ， + 1 ，都是符合条件（≥ 0 ）的，故不可修剪。

从例 2.19 可知：

(1) 与节点的 PBV（通常称为 **α 值**）在搜索过程中也可能变动，但是永不下降。所以其 PBV加上符号≥来表示。

(2) 或节点的 PBV（通常称为 **β 值**）在搜索过程中也可能变动，但永不上升。所以其 PBV 加上符号≤来表示。

采用上述这种保留 PBV 踪迹的修剪技术的过程，称为 **α − β 过程**，或称 **α − β 搜索**。

α-β 搜索的修剪规则叙述如下：

(1) 在某个与节点中，如果它的 PBV 大于或等于它的先辈节点中任何一个或节点的 PBV(即 β 值)的话，则在该与节点可以修剪掉未搜索的分枝，而不再搜索其后裔节点。此修剪称为 **β 修剪**。因为它受到其先辈节点的 β 值的约束而修剪。此与节点的 PBV 也就可最终地确定下来了。

(2) 在某个或节点中，如果它的 PBV 小于或等于它的先辈节点中任何一个与节点（包括起始节点）的 PBV（即α值）的话，则 在该或节点处可以修剪掉未搜索的分枝，而不再搜索其后裔节点。此修剪称为 **α 修剪**。因为它受到其先辈节点的 α 值的约束而修剪。此或节点的 PBV 也就可最终地确定下来了。

从图 2.44 可以看出，α-β 搜索比最小最大值搜索要好。原来需要搜索 22 个端节点，现在只需搜索 9 个端节点。并且对于起始节点而言，得到的返上值也是相同的，而且等于同

一个端节点⑲的评价函数值。如果这个端节点在深度优先搜索中最先搜到，那么修剪掉的节点数就将是最大。

如果修剪掉的节点数目达到最大时，应产生及应求评价函数值的端节点有多少呢？

假设树的深度为 D，除端节点外，每个节点都恰恰有 B 个子节点。这样的树有 B^D 个端节点。设 α-β 搜索所产生的子节点的顺序很巧：对于或节点而言，评价函数值（或 PBV）最低的子节点先产生；对于与节点而言，评价函数值（或 PBV）最高的子节点先产生。这么一来，修剪掉的节点数达到最大，而端节点产生的数目达到最小。用 N_D 表示端节点的这个最小值，则

$$\begin{cases} N_D = 2B^{D/2} - 1 & (D \text{ 为偶数时}) \\ N_D = {}^{(D+1)/2} + B^{(D-1)/2} - 1 & (D \text{ 为奇数时}) \end{cases}$$

这就是说，由最佳 α-β 搜索法而产生的而且深度为 D 的端节点数，与不是 α-β 搜索法而产生的且深度为 D/2 的端节点数大致相同。因此对于同样的存贮空间而言，α-β 搜索法能够把搜索深度加深至二倍。如图 2.45 所示。

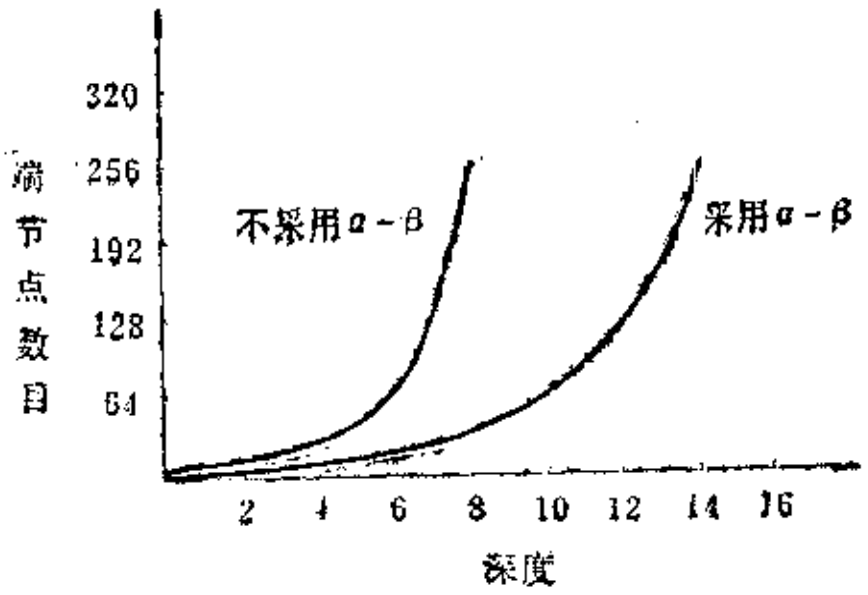


图 2.45　采用与不采用α-β搜索法的比较

2.4.4　A-B 型 α-β 搜索法

前述的搜索法中，节点的搜索顺序是固定不变的，称为**固定顺序搜索法**。改变节点的搜索顺序，可以修剪得多些。这种方法称为**变动顺序搜索法**。**A-B 型 α-β 搜索法**就是其中之一。

以图 2.46 为例来说明 A-B 型 α-β 搜索法。图中节点旁边的数字分别是对于各个格局进行静态评价而得的评价函数值。

开始，根据第一级节点的评价函数值的大小，理成顺序为 $N_1(9)$、$N_2(7)$、$N_3(5)$。如图 2.46(a)所示。这时把第二大的评价函数值 $N_2(7)$ 称为 **A 值**。A 取为 7。接着，顺着评价函数值最大的节点

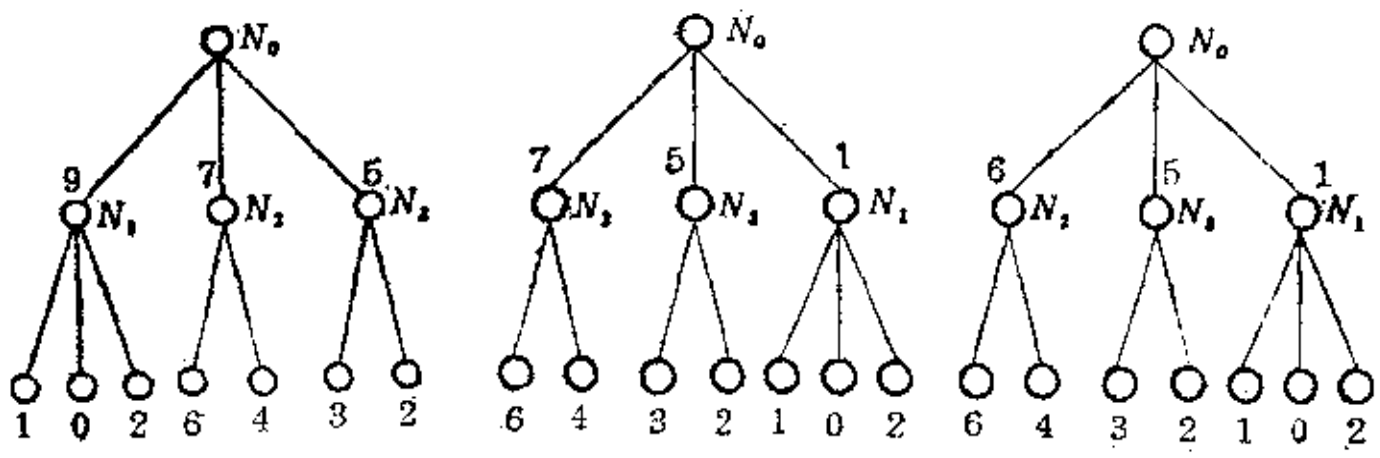


图 2.46　A-B 型 α-β 搜索法

N_1进行搜索。其子节点的评价函数值为 1，比 A 值小。则对 N_1 的搜索就停止，立即变更第一级的顺序为 $N_2(7)$、$N_3(5)$、$N_1(1)$。如图 2.46(b) 所示。新的 A 值取为 5。根据这个顺序对

N_2 进行搜索。其子节点的评价函数值为 6，比 A 值大。故不改变其顺序。此时顺序为 N_2(6)、N_3(5)、N_1(1)。A 值仍为 5。如图 2.46(c) 所示。依此类推，搜索下去。

以上例子是在取最大值级上评价的，则取第二大的值作为 A 值。而在取最小值级上评价时，则取第二小的值作为 **B值**。这样来改变节点顺序。

根据新得到的评价函数值而变更 α-β 搜索顺序的这种方法，称为 **A-B 型α-β搜索法**。

搜索顺序的变更取决于第二级的评价函数值与 A 值之差。为了控制这个顺序的变更，引入阻抗 R。如果新得到的评价函数值比 A－R 小（在取最大值级上评价时），或比 B＋R 大（在取最小值级上评价时），才变更顺序。否则不变更。这么一来，如果 R 值很大很大它就很近似于固定顺序搜索法；如果 R 值很小很小，则顺序的变更是相当频繁的。可取适当的R值。

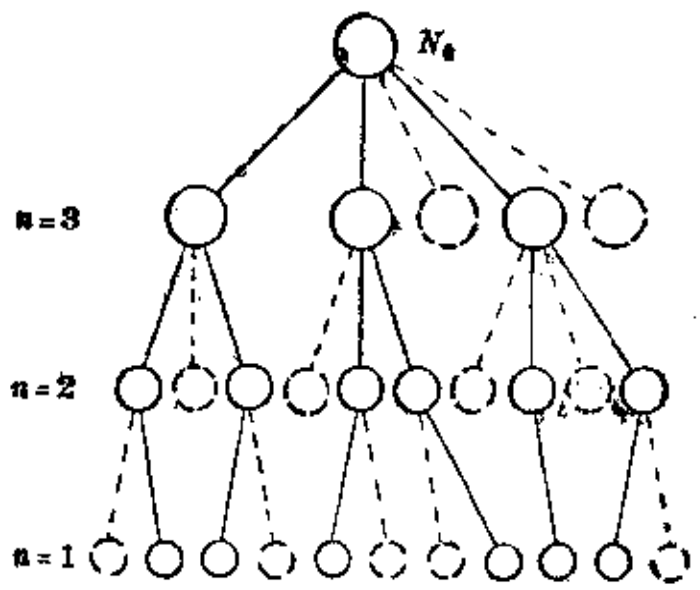


图 2.47　向前修剪搜索的一例

2.4.5　向前修剪的搜索法

向前修剪的搜索法仍是选择有效地接近解中“希望”较大的某些节点而进行搜索的方法。在为了得到这个“希望”的过程中，并不去调查其后裔的所有的部分搜索树。当然，在作为无希望而被修剪掉的树枝中，可能也有一些树枝是很有效的。在这些场合，就作出了一些牺牲。这种修剪就可能是**灾难性修剪**。

这种搜索法有若干种。例如，在各个搜索级中只选择一定数量的有希望的节点；又如，随着搜索深度的增加而减少选择的节点数，等等。后者如图 2.47 所示。其中 n 值为在各个级选择的节点数，虚线表示修剪掉的枝和节点。

2.5　非确定程序

以上讨论的方法有一个共同的特点：在搜索过程中，无论遇到什么情况，根据操作规则及现行条件，总是只有一种可能而继续搜索下去。因而只要现行条件不变，让机器多次重复搜索的话，它的搜索过程（步骤及操作）总是可以重复出现的。所以它们都属于确定程序这类方法。

如果在搜索过程中，根据操作规则可以有多种可能操作供选择，不过只任选其中一种而搜索下去，则称之为**非确定程序**。

非确定程序也可以有状态空间法及与-或图法等。

2.5.1　状态空间法

用 x 表示输入的数据结构，作为程序变量；y 表示任意的数据结构，是程序变量；F(x，y) 表示把 x 和 y 的正交定义域映照到 y 域的一个非空子集合之一个全函数，用符号 {F(x、y)} 表示这个子集合。那么状态空间法的非确定程序中较简单的流程图如图 2.48 所示。

图 2.48(a) 中置初态框图内的“y←x”，表示将输入的数据结构设定为 y。

第一个“或分枝”表示根据 x，y 有 n 种判定 $P_1(x, y)$，$P_2(x, y)$，…，$P_n(x, y)$。这些判定在 x，y 的正交域上真值有真或假，但至少有一个判定值是真。每个判定对应于一个分枝。选上的某一分枝为真，而常常是可以有几个分枝都是真。程序在执行的当时，允许不确定地选定具有真的任一种。这里所用的“不确定”，并不全和“随机”同义，不包含任何偶然事件的期望计算。

“设定”框中的操作“y←SELECT{F(x, y)}”表示选择子集合中的对应于选上的 P_i(x, y)的成员，设定为 y。

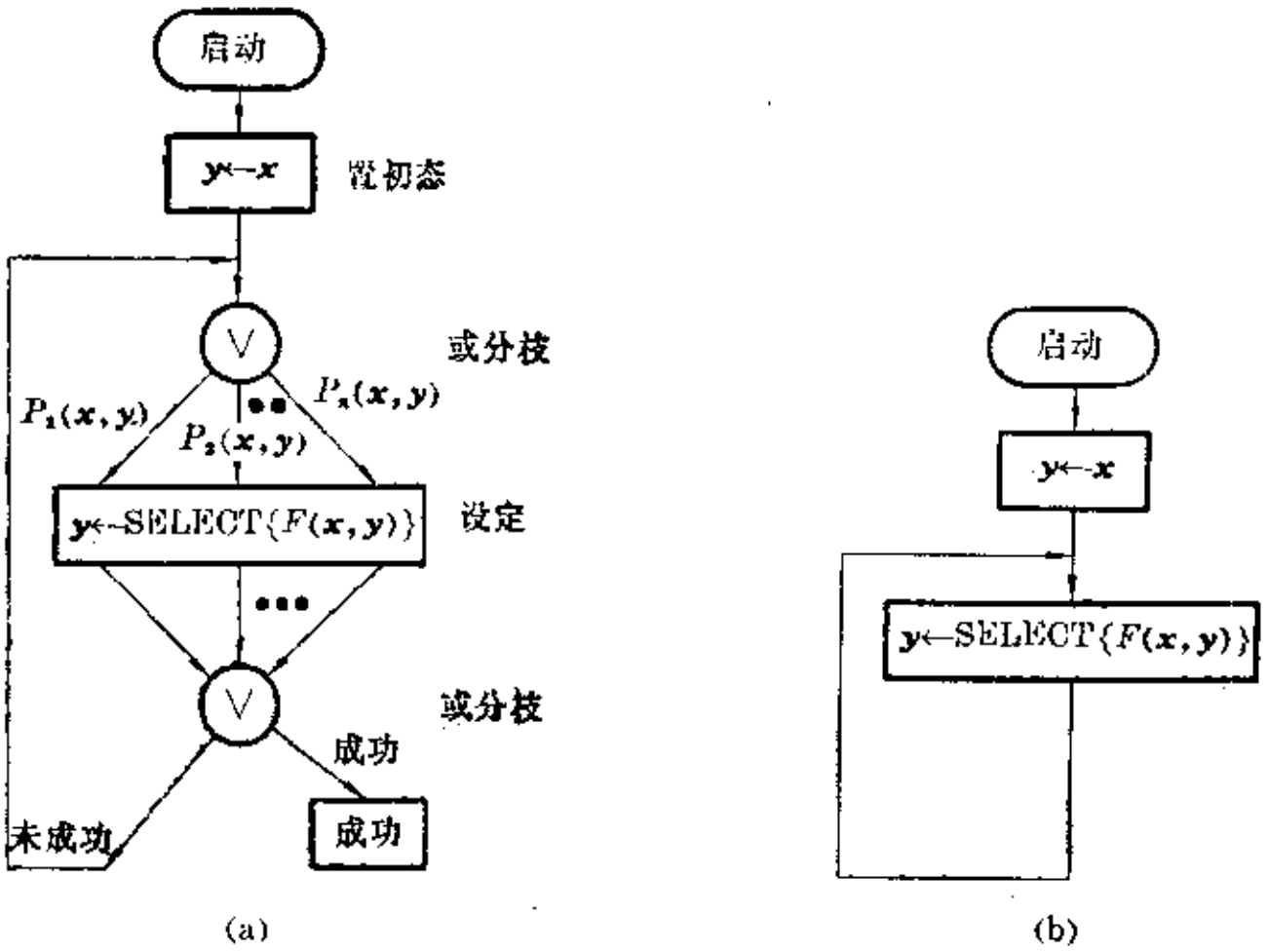


图 2.48　状态空间法的非确定程序

图 2.48(a) 也可简化为图 2.48(b)。其中 y→SELECT{F(x, y)} 已包含有或分枝之意。而且忽略了程序结束的条件。

如果{F(x, y)}中只与 y 有关，则 F(x, y)就退化为前述的求 y 的后裔的一般操作的集合 Γ(y)。y←SELECT{F(x, y)}就退化成y←SELECT[Γ(y)]。

例2.20　“移动十五”的非确定程序。如图 2.49 所示，图中函数 L，U，R 及 D 分别对应黑块左移，上移，右移及下移所得到的格局。

从图中可以看出，如果黑块在第 2 行（或第 3 行）的第 2 列（或第 3 列）的四个方格子里，则各有四种可能的移动法；如果黑块在第 1 行的第 1 列（或第 4 列）以及第 4 行的第 1 列（或第 4 列）等四个方格子里，则各有二种可能的移动法。在执行时，只要操作规则允许，就可随意地选择其中对应的某一种。

2.5.2　与-或图法

用非确定程序也可以产生与-或图。类似于 SELECT 的设定语句，可以定义 ALL 的设定语句。其简化的流程图如图 2.50 (a)所示。

图中同样也忽略了程序结束的条件。操作 y←ALL{F(x,y)}表示设置程序变量 y 的新

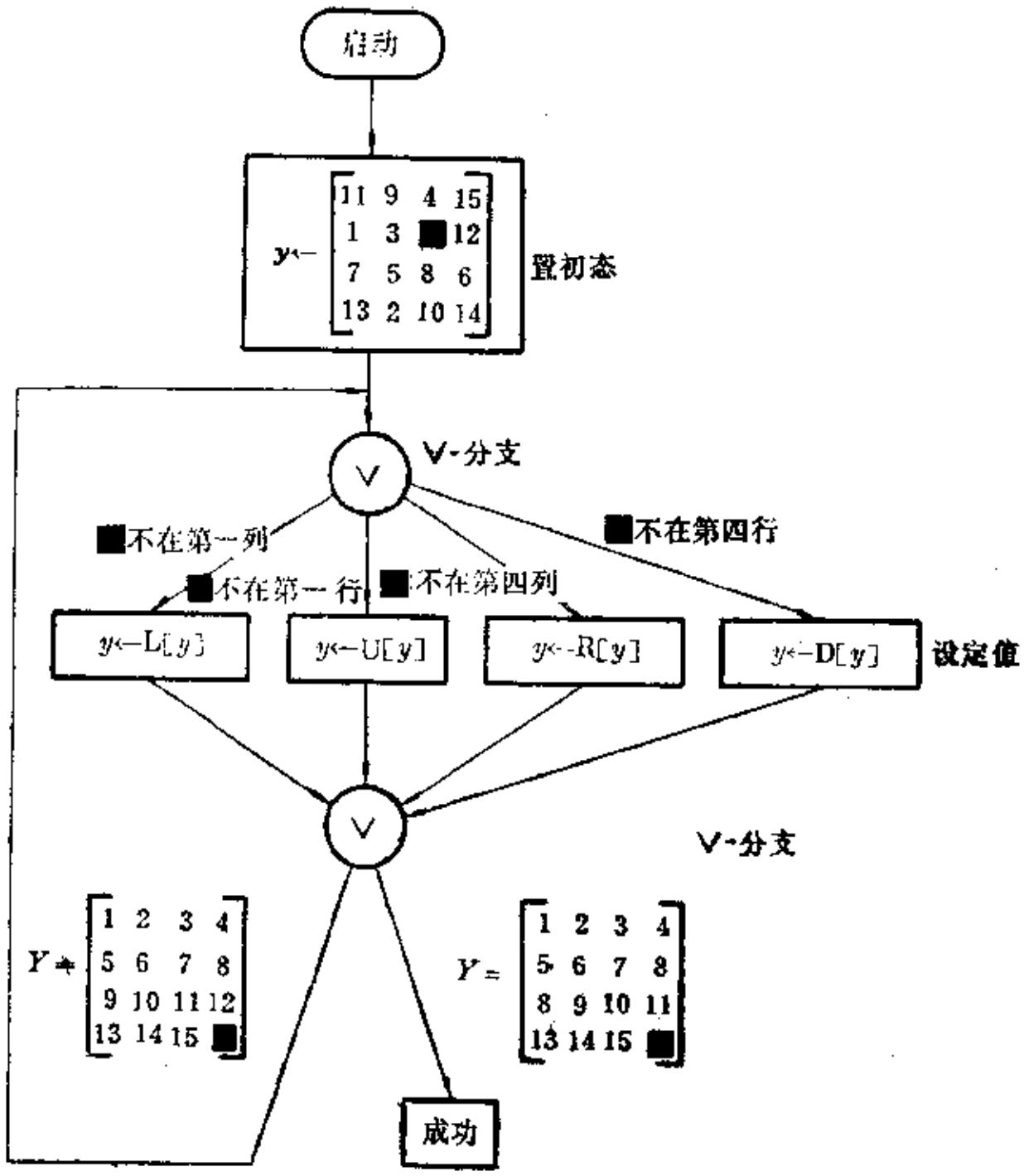


图 2.49 “移动十五”的非确定程序

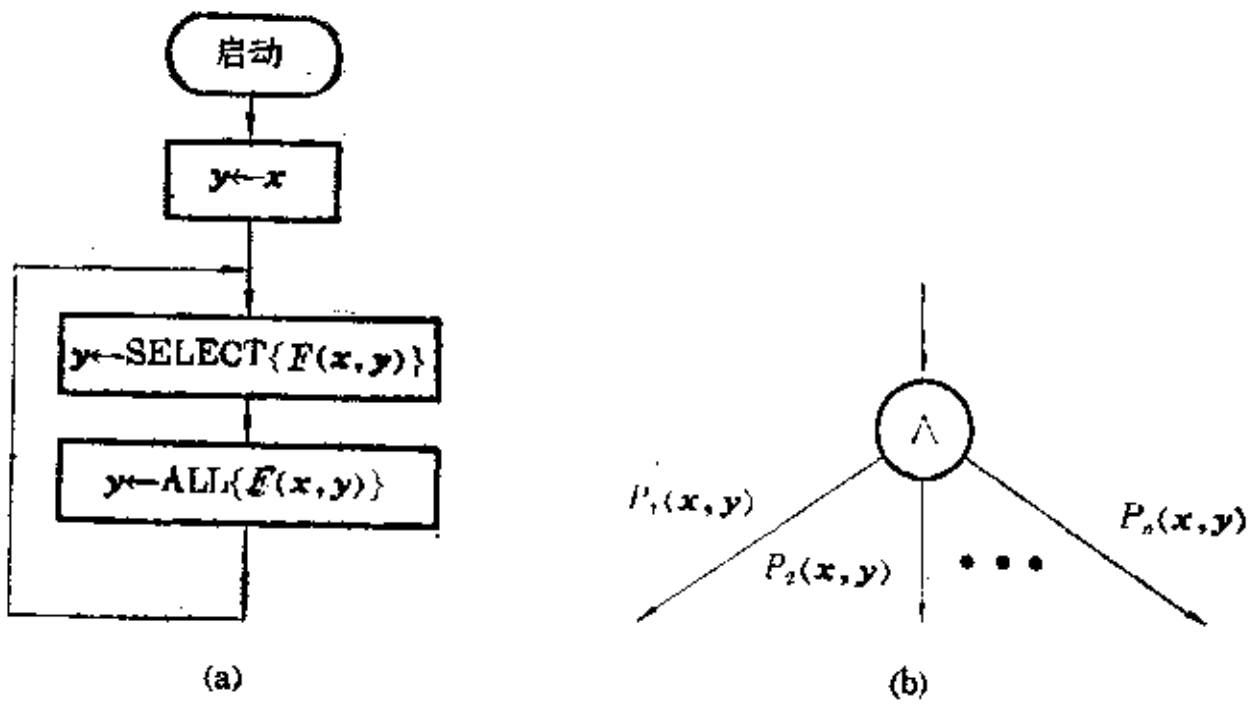


图 2.50 非确定程序的与-或图法

值等于{F(x,y)}的所有成员。注意，此处用“所有成员”就使之与 SELECT {F(x,y)}根本区别开来。

对于 y←ALL{F(x,y)}，同样也可引入一个如图 2.50(*b*)所示的用 n 个判定 $P_1(x,y)$，$P_2(x,y)$，…$P_n(x,y)$表示 n 路分枝的“与”分枝记号。这些判定在 x 和 y 正交域上真值有真和假。至少有一个判定的值为真。对应于判定值具有真的所有（这就是 ALL 及“与”的意义所在！）分枝都应该选上。

原先的确定程序的分枝仍是“与”分枝的特例。而且，当 F(x,y)的值是一个单值的话，那么 SELECT 语句，ALL 语句和原先的确定程序的“设定”都是相同的了。

从图 2.50 (*a*)中可以看出，由于在“或”分枝和 SELECT 语句中只选取其中一种，而在“与”分枝和 ALL 语句中要取所有可能的选择。这就是一个与-或图。在执行过程中，如果程序一进入某个“与”分枝时，只有在它的每个分枝都终结的情况下，它才能算作终结。

同样，此处的 F(x,y)也可退化为 Γ(y)。

例 2.21 八个皇后问题。在国际象棋棋盘上，要放上八个皇后，但是不能使某个皇后和另一个皇后互相捕获到。其非确定程序的流程图如图 2.51 所示。

在这个流程图中，r_i 和 c_i 分别表示第 i 个皇后所在的行和列的位置。(r_i, c_i) 表示第 i 个皇后的位置。判定 cap $((r_i, c_i), (r_j, c_j))$ 表示第 i 个皇后与第 j 个皇后能否互相捕获到。只有在这两个皇后能互相捕获时，cap $((r_i,c_i), (r_j, c_j))$ 才为真；否则，取值为假（图中用～cap$((r_i,c_i), (r_j,c_j))$表示）。原来问题的要求就变成：求 (r_i, c_i)，使之对于所有的 i，j，$(i,j=1,2,\cdots,8, i\neq j)$，cap$((r_i, c_i), (r_j,c_j))$都为假。

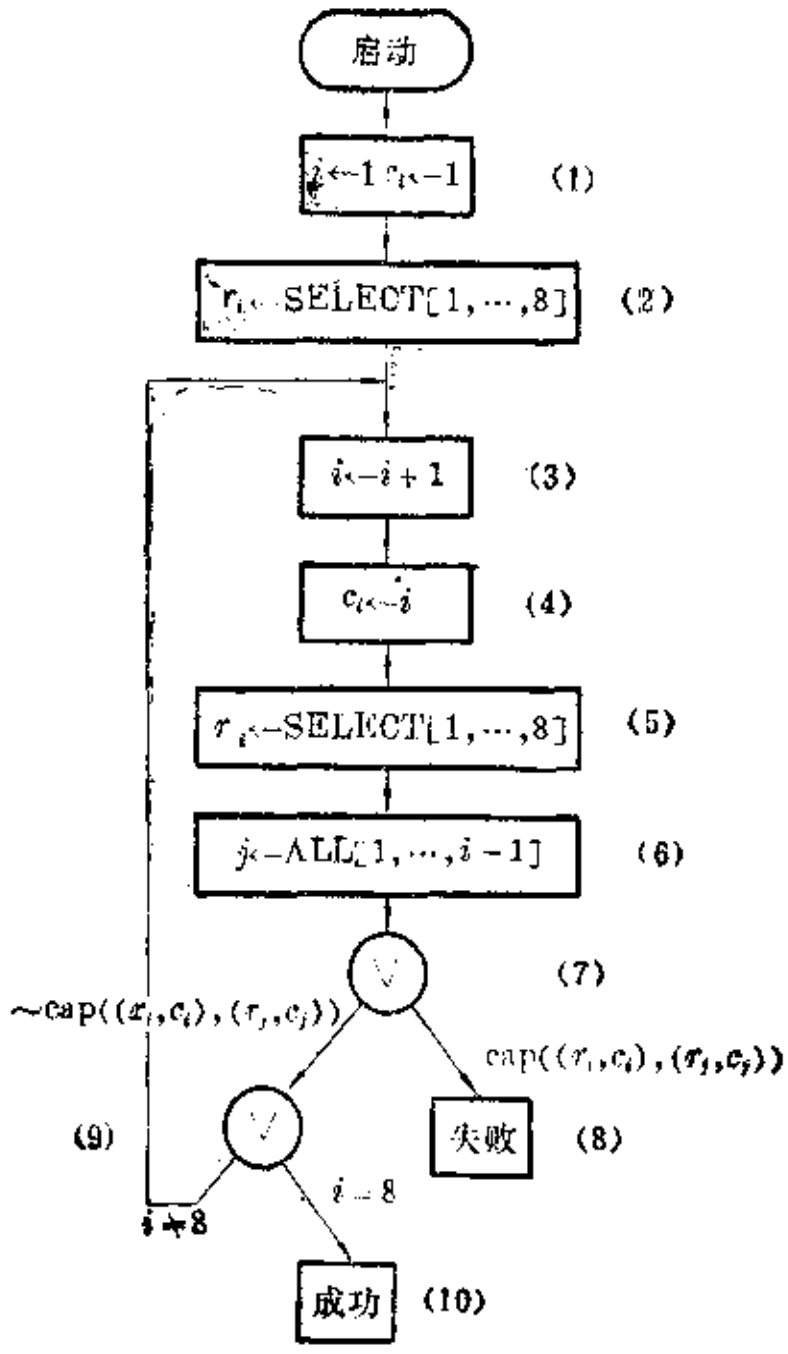
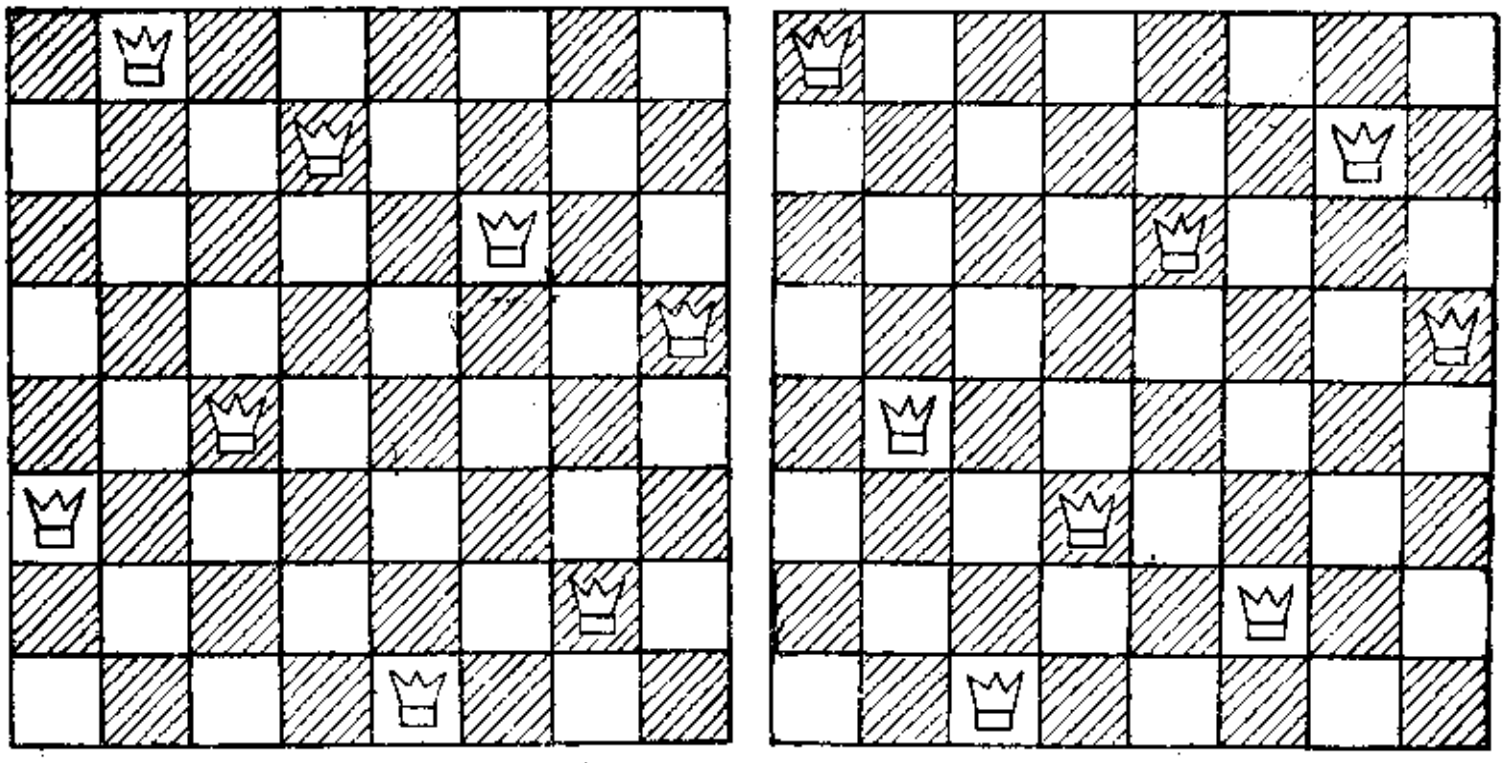


图 2.51 八个皇后问题的非确定程序

图中的置初态程序框(1)和(2)中，其意义为取第 1 个皇后放在第 1 列和第 1 至第 8 行中的任一行上。第(3)框中表示准备放置下一个皇后的位置。第(4)框中表示将第 i 个皇后放在第 i 列上，但放在第几行上呢？如第(5)框所示那样，就任选取 1 至 8 行中的某一行位置来试试看。这样放置能否与先前放置的皇后互相捕获到吗？在第(6)(7)框中进行检验。

第(6)框表示先前放置的皇后（1至 $i-1$）要一个不漏地都取出来检验。如果检验的结果发现已互相捕获，则失败（转至第(8)框）而退出。否则，说明这个皇后放置的位置是允许的，可继续下去。在第(9)框中再检验一下八个皇后都放置完了吗？如果已放置完，则宣告成功（第10)框）；否则，再转至第(3)框准备继续放置下一个皇后。

图 2.51 的程序流程图是可行的。可得出如图 2.52 所示的其中二个正确答案。



(a)　　(b)

图 2.52　八个皇后问题的其中二个答案

第二章　习　　题

一、有五个城市 A、B、C、D 和 E。各城市之间的交通费用如图 2.53 所示。某游客欲从城市 A 出发，游遍其他四个城市，而后回到城市 A。请你导游，如何使交通费用最省？？

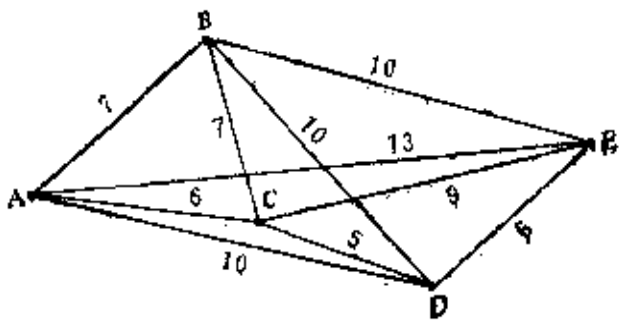


图 2.53　游览图

二、有三只猩猩和三个人欲从河的西岸渡船到河的东岸。河西只有一条可载二物的空船。除了人及一只猩猩会划船之外，其余二只猩猩不会划船。在他（它）们可能在一起的任何情况中，都不允许猩猩的数目大于人的数目，否则会发生伤人事故。试用状态空间搜索法求解出其各种可能渡法。

三、试画出有四个金片的“梵塔”问题之状态空间搜索圈。

四、有十二只乒乓球，其中有一只仅是重量不同的次品。试用一架没有法码的天平，只称三次，把次品挑出来，并说明该次品是过重或过轻？

五、试写出各枝耗散不全相同的深度优先搜索法之步骤，并画出其流程图。

六、深度优先搜索法区别于广度优先搜索法的主要不同点在于（选择其中一个答案）：

(1) 在把某节点加以扩展而产生的子节点放入 OPEN 表时，要考虑这些子节点的先后顺序。

(2) 从 OPEN 表中取出节点来扩展时，不管顺序而任意取其中某一节点即可。

(3) 从 OPEN 表中取出节点来扩展时，后放进的节点优先扩展。

(4) 从 OPEN 表中取出节点来扩展时，先放进的节点优先扩展。

七、启发式程序法区别于广度或深度优先搜索法的主要不同点在于（选择其中一个答案）：

(1) 根据节点产生的先后顺序来扩展节点。

(2) 根据 OPEN 表中的评价函数值的大小顺序来扩展节点。

(3) 根据评价函数值求算的先后顺序来扩展节点。

(4) 因为先产生的节点先得到启发，所以先扩展。

八、用启发式程序法解出如图 2.54 所要求的重排九宫问题。

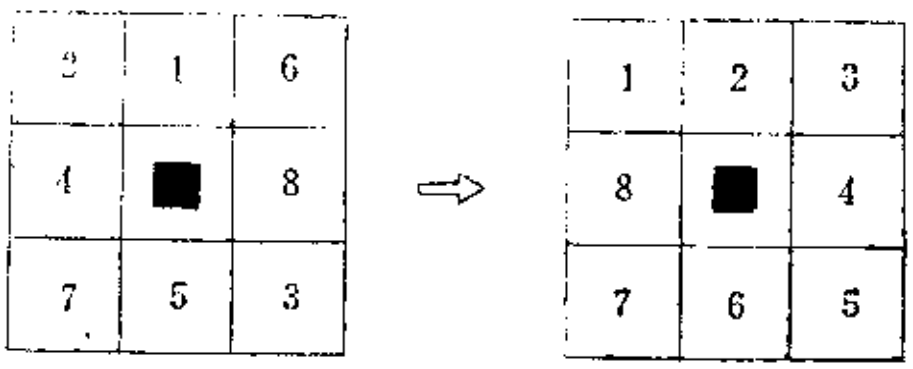


图 2.54　另一个重排九宫问题

九、用算符法证明下列的平面几何问题：

(1) 平行于第三条直线的二条直线也互相平行。

(2) 矩形的对角线相等。

十、在如图 2.26 的猴子与香蕉问题中，试分别对式(2.10)和式（2.11）的关键操作进行归约，求出其归约图。

十一、用二只各可盛五斤和七斤水的空瓶，到河边提水，要提六斤水回来。试画其约图。

十二、试写出搜索一般与-或图的方法。

十三、求解如下不定积分：

(1) $\int(3x^2-2x+1)dx$

(2) $\int\frac{x+1}{3\sqrt{3x+1}}dx$

(3) $\int te^t dt$

十四、甲乙丙三人学习雷锋做好事又不留名。有次，厂里接到群众的表扬信，并要求找到该人。根据分析，肯定是他们三人中的一人做的，而且他们都知道是谁做的。后来要他们对该事各讲一句真话和一句假话。他们都严格遵守这个约定。

甲说："不是我做的。也不是乙做的。"

乙说："不是我做的。是丙做的。"

丙说："不是我做的。也不是甲做的。"

试问，到底是谁做的？（用与-或图法）

十五、甲说："乙在说谎。"乙说："丙在说谎。"丙说："甲和乙都在说谎。"到底

甲乙丙说的是真话，还是假话。试用与-或图求出解来。

十六、将图 2.55 所示的电子线路图用归约法解出其阻抗 Z_i 来。设 L_i、R_i、C_i 为已知。

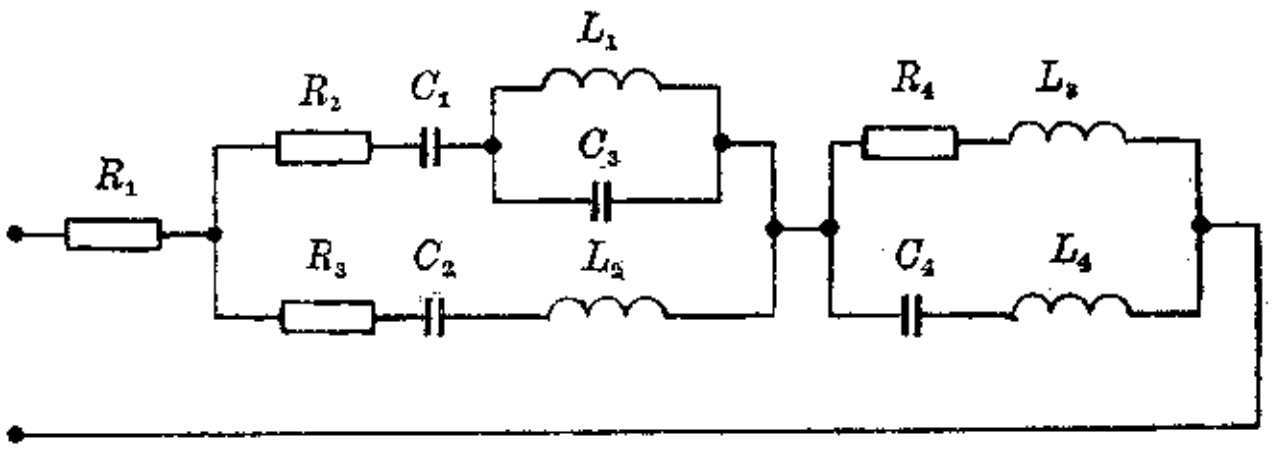


图 2.55　电子线路图

十七、设图 2.56 是一个与-或图。图中节点的数值表示其耗散值。试分别在节点旁边用数字表示出广度优先搜索法，深度优先搜索法和有序搜索法。（在做前两种方法时将图中的耗散值取消。）

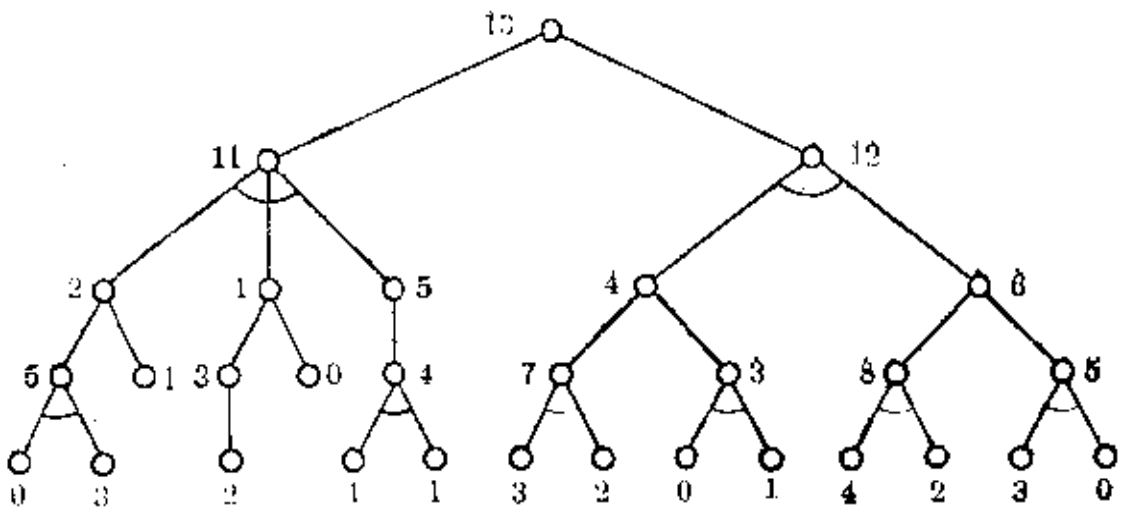


图 2.56　一个与-或图

十八、现在有五顶帽子，其中红色二顶，绿色、白色、兰色各一顶。先让依序排成一列纵队的甲乙丙丁四人都看一下。而后他们都闭上眼睛。接着分别给每人戴上一顶帽子，余下一顶就藏起来。让他们都张开眼睛之后，就问排在最后的丁他头上戴什么帽子。丁看了看前面三人头上的帽子而后回答说不知道。接着问丙。丙根据丁的回答及前面二人所戴的帽子，也猜不出。再问乙，乙根据丁丙的回答及甲所戴帽子，也猜不出。最后问甲，甲根据三人的回答就猜出自己头上戴的是红色帽子。试将甲的推理过程用与-或图表达出来。

十九、有一堆火柴棒，共九根。甲乙二人轮流玩。每人必须而且只能从中取走 1 根、或 2 根、或 3 根。取走最后一根者输。试用与-或树法来证明后玩者必胜。

二十、上题中，如果有十根火柴的话，后玩者必胜或必输？如果火柴的根数是随意给定的话，那么后玩比先玩者的必胜概率是大呢还是小？

二十一、图 2.57 是一棵博奕树。试分别标出最小最大值搜索过程和 α-β 修剪过程。

二十二、甲乙二人玩五子棋。即在 19×19 格的棋盘上每人轮流放自己的棋子一只，谁先使自己的棋子达到五子成一直线（横、竖、斜都行），谁获胜。试写出五子棋的评价函数。

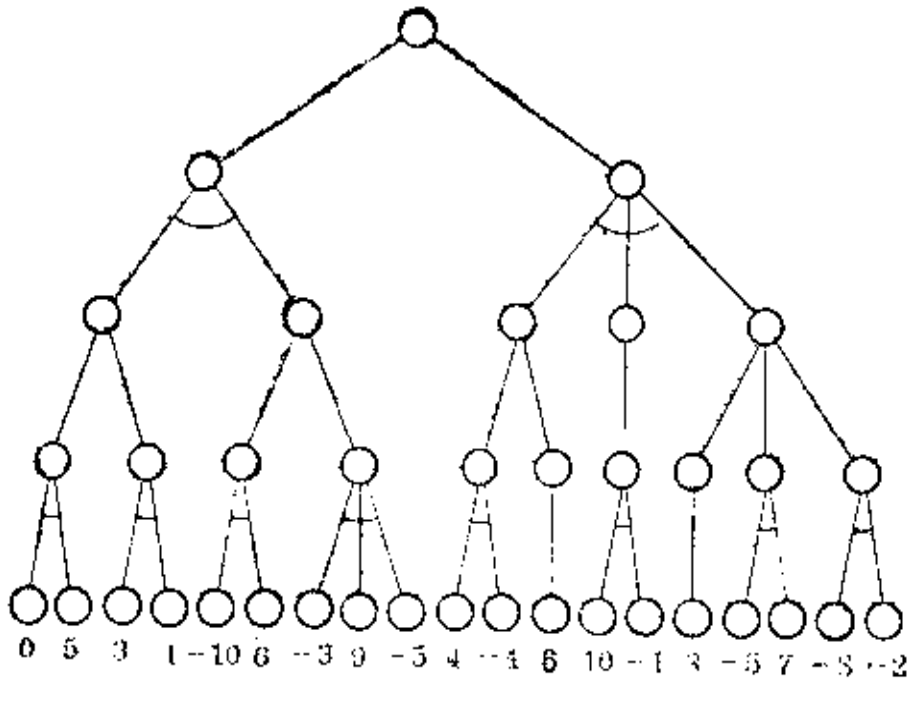


图 2.57　一棵博奕树

图 2.58　民间游戏的棋盘

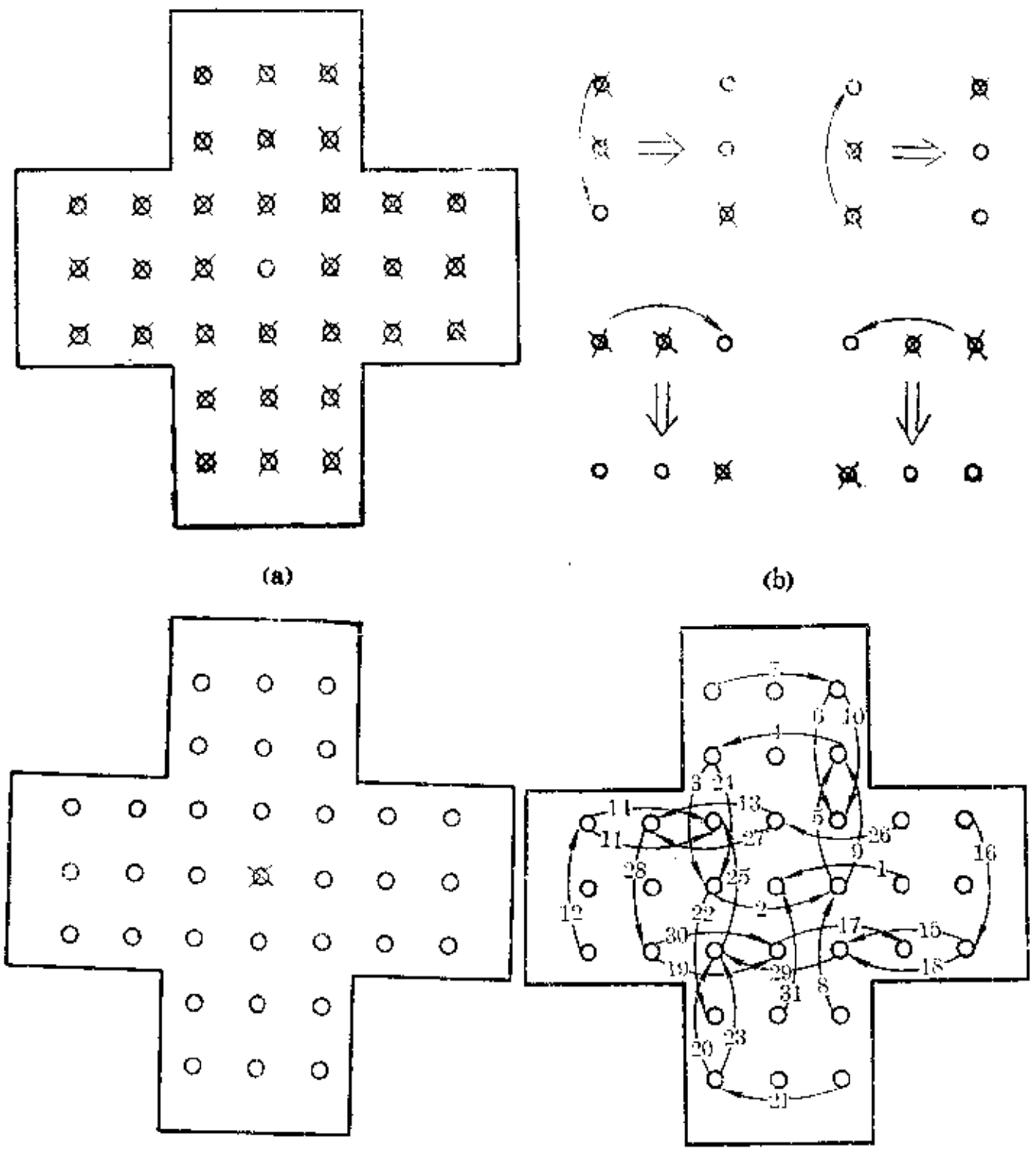


图 2.59　另一种跳棋

二十三、某省民间有一种二人玩的三子棋游戏。棋盘如图 2.58 所示。其玩法是比较复杂的。为方便起见，将它的游戏规则简化如下：

甲乙二人轮流在棋盘上的交叉点处放上自己的棋子一只，直到棋盘的各交叉点放满为止。哪方的棋子能达到某三子成一直线，就可得一分。谁得的总分最高就获胜。试为甲方画出几级博奕图。

二十四、画出图 2.54 的非确定程序流程图。

二十五、画出猴子与香蕉问题的非确定程序流程图。

二十六、图 2.59 所示的是另一种跳棋游戏。其开局如图 2.59 (*a*)所示，只有中间一个空格，其余都布满棋子，图中 × 表示棋子。图 2.59(*b*) 示出各种可能的跳法及跳的结果。在经过 31 次跳跃之后，最终结果如图 2.59(*c*)所示。这是能够做得到的。而且方法也有多种。图 2.59(*d*)是其中的一种。试用非确定程序方法解之。

第三章　自动定理证明

人们在认识事物中，有一个很重要的才能就是有抽象能力。人们能够从大量观察的事例中抽象出其共同的形式，识别其共同的原理。如何把这个能力赋予机器，直至目前为止还是一个很大的难题。但是人们往往将已经提出的定理之类的证明工作进行了一番探索，利用计算机来协助自己做些工作。所以利用计算机来证明某些定理，往往是比较机械，因而有人称这种证明为**自动定理证明**，或称为**机器定理证明、机械定理证明**。

一般说来，证明定理的方法有这样几种：一种是归纳法。例如数学归纳法这种从特殊真理推论至普遍真理的归纳过程，在被检验时所涉及到的数目不太多的情况下，可以采用穷尽列举的方法——完全归纳法。给这个方法予系统化的手段就可用真值表。在这章中，我们在证明某些定理时就是采用这种方法。另一种方法是演绎法。这是本章着重要介绍的方法。在本章中主要介绍命题演算和谓词演算。在谓词演算中，除了介绍 Herbrand 提出的方法外，着重介绍 Robinson 于 1965 年提出的分解原理。

3.1　命 题 演 算

3.1.1　基本符号

在日常生活中，我们有时感兴趣的是每个有意义文句的内容是真或是假，而不深入考究其内部的细致含意（如主词与谓词之间的关系）。将非真即假的一些陈述句称之为**命题**，用大写拉丁字母 A，B，…表示。这些表示命题的符号 A，B，…称为**原子公式**。每个原子公式看成为一个整体。

给原子公式设定的“真”（用 T 表示）或者“假”（用 F 表示），称为原子公式的**真值**。“真值”并不是“真”，而是包含 T 和 F 二值。例如“这部计算机在运行着”，“天在下雪”两者都是非真即假的命题，可分别用 A，B 表示。如果在某时刻，这部计算机确实在运行着，但天却不在下雪，则原子公式 A、B 的真值可分别设定为 T、F。

还可以用一些逻辑联结词把命题联结起来而得出新的**复合命题**。例如，“这部计算机在运行着与天在下雪”，“如这部计算机在运行着，则天在下雪”都是一个复合命题。可用联结词“与”，“如…则…”来表示成“A 与 B”，“如 A 则 B”。这类联结词有六个，其相应的联结符号为：“～”（读成“**非**”）、“∧”（读成“**与**”、“**合取**”）、“∨”（读成“**或**”、“**析取**”）、“→”（读成“**如…则…**”、“**蕴含**”）、“→→”（读成“**如…则…否则…**”）以及“⟷”（读成“**等价**”）。

命题经过这些联结词联接起来的复合命题，也是命题。复合命题还可用这些联结词再联结起来而成新的复合命题。所有这些命题都可称为**合式公式**。

合式公式在命题演算中可递归定义如下：

---

* 本章主要参考文献见“参考文献”1、9、10、11 和 16。

1. 原子公式是合式公式；

2. 如 A，B，C 都是合式公式，则 (~A)，(A∧B)，(A∨B)，(A→B)，(A→→B,C) 以及 (A←→B) 都是合式公式。

3. 所有的合式公式必须都能反复应用以上规则而生成。

根据以上定义，可以知道表达式 (A→) 及 (A~B) 都不是合式公式，而表达式 ((A→B)→→ ((A∧B)←→(~C)),(A∨B)) 却是一个合式公式。可图示于式 (3.1) 中：

$$\underbrace{\underbrace{((\underbrace{A\to B}_{\text{合式公式}})\to\to \underbrace{((\underbrace{A\wedge B}_{\text{合式公式}})\leftrightarrow(\underbrace{\sim C}_{\text{合式公式}}))}_{\text{合式公式}}}_{\text{合式公式}}, \underbrace{(A\vee B)}_{\text{合式公式}})}_{\text{合式公式}} \tag{3.1}$$

这六个联结词所确定的合式公式的真值与原子公式的真值之间的关系如下定义：

1. ~A 为真的充要条件是 A 为假；~A 为假的充要条件是 A 为真。

2. (A∧B) 为真的充要条件是 A、B 都为真；否则 (A∧B) 为假。

3. (A∨B) 为假的充要条件是 A、B 都为假，否则 (A∨B) 为真。

4. (A→B) 为假的充要条件是 A 为真且 B 为假；否则，(A→B) 为真。

5. (A→→B,C) 的真值如下式定义（其中≜表示“根据定义等于”）：

$$(A\to\to B,C)\triangleq\begin{cases}B，\text{如果 A 为真，}\\ C，\text{如果 A 为假。}\end{cases}$$

6. (A←→B) 为真的充要条件是 A、B 具有相同的真值；否则 (A←→B) 为假。

以上定义可用表 3.1 来表示。

表 3.1 六个联结词的真值表

A	B	C	~A	A∨B	A∧B	A→B	A→→B,C	A←→B
F	F	F	T	F	F	T	F	T
F	F	T	T	F	F	T	T	T
F	T	F	T	T	F	T	F	F
F	T	T	T	T	F	T	T	F
T	F	F	F	T	F	F	F	F
T	F	T	F	T	F	F	F	F
T	T	F	F	T	T	T	T	T
T	T	T	F	T	T	T	T	T

必须注意：(1) (A→B) 不应理解为表示存在于原因和结果之间的那种关系。虽然都是“如 A 则 B”的叙述形式，但在因果关系中，如果 A 不成立，B 成立与否是没有结论的；而在 (A→B) 中，当 A 为假时，(A→B) 却为真。不过，(A→B) 与因果关系两者仍然有一个共同点：在(A→B)为真的情况下，由 A 的成立(为真)可以推出 B 也成立 (也为真)。

(2) (A→→B,C) 中的逗号“,”，不是联结词符号，而是标点符号。因此应将 A→→B,C当作为一个整体，缺一不可。

由于合式公式的定义是递归的，因此由联结词符号联结而成的一些合式公式，还可以再由联结词符号联结而成新的合式公式。这样，在一个合式公式中就有多个联结符号。为了表示这些联结符号之间的主从关系，在此引入括弧“(”、“)”。每个合式公式都用一对括弧括起来。式 (3.1) 就是一例（在此省略了原子公式的括弧对）。左右括弧都不是联结词符号，而是标点符号之类。按照惯例，左右括弧必须成对出现，而且是依序内外层配对。分不清内外层关系的括弧对可视为并列关系。

如果一至三个括弧对之间由联结词符号紧密地（即根据该联结词的联接规则）联结起来的话，那么其括弧对内的联结符就是这个联结符的**从联结符**。当然，从联结符中可能还有它的从联结符。

为了省略一些括弧时，除了合式公式的最外层括弧对一概省略之外，人们常常仿照算术运算中的先乘除后加减等约定，而规定其先后运算次序为～，∧，∨，→，→→，←→。例如，((～A)∨(A→(A∧B)))可简略为～A∨(A→A∧B)。

在本章中，我们也参照这个先后运算次序的约定，但为了能一目了然，通常还是给予适量的括弧对。甚至于为了能够方便地判定表达式是否合式公式，除了最外层的括弧对省略外，其余尽量不省略。

在本章中还约定，一个合式公式只有一个主联结符。所谓“一个合式公式的**主联结符**”就是在该合式公式中该联结符不能成为其他联结符的从联结符。显然，如果合式公式能最终表示为～A，A∧B，A∨B，A→B，A→→B,C，A←→B 等某一形式的话，～，∧，∨，→，→→，←→就分别称为它的主联结符。相反地，除了原子公式而外的表达式，根据合式公式的定义，如果找不到其主联结符的话，则该表达式必不是合式公式。因此通过找主联结符可以判别该表达式是否合式公式，而且可以将其各原子公式的联结关系搞清楚。

寻找合式公式的主联结符的算法

可设置一个括弧计数器。当一个表达式输入时，自左至右依序一个个符号进行判别。并且自始至终对括弧计数器计数：如果是左括弧，则括弧计数器加 1；如果是右括弧，则括弧计数器减 1。在只省略最外层括弧的情况下，当括弧计数器的数值恰为零时，紧接着的联结符就是该合式公式的主联结符。

判别表达式是否合式公式的基本思想是：

(1) 一旦出现括弧计数器的数值小于 0，则立即可判定该表达式不是合式公式。

(2) 找不出主联结符，则该表达式不是合式公式。

(3) 一旦找到主联结符，可以根据主联结符的具体含意，按照六个联结符的定义，立即把该表达式分成一至三个（比原表达式还短的）表达式。例如，如主联结符是联结符～时，则将该联结符的右边的子表达式分出来，～的左边不能有其它表达式，否则，可判定该表达式不是合式公式；如主联结符是联结符∧，∨，→，←→时，则将该联结符的左右部分分成二个子表达式；如主联结符是→→，则将该联结符的左边部分分成一个子表达式，而从该主联结符的右边起，一直到括弧计数器的数值恰为 0 时的逗号“,”为止，分成另一子表达式；该逗号以右部分分成第三个子表达式。如果分出的子表达式有最外层括弧对时，则要删去。

(4) 按(3)的方法分离出的子表达式，如果是原子公式，则不再分离。否则就把它作为表达式再分别找主联结符，且按照(3)的方法再分离出新的子表达式来。反复如此操作。

一旦其中某个子表达式出现(1)(2)情况，或者是空的，则原先的整个表达式就不是合式公式。

一旦分离出的每个表达式都是原子公式而无法再分离时，则整个表达式就是一个合式公式。

例 3.1　分析一下表达式

$$(A\rightarrow\rightarrow(A\rightarrow\rightarrow B,C),B)\vee((A\wedge(C\rightarrow(\sim B)))\longleftrightarrow(\sim(A\vee B)))$$

该表达式的主联结符是“$\vee$”，故可分成二个子表达式 $A\rightarrow\rightarrow(A\rightarrow\rightarrow B,C),B$ 以及$(A\wedge(C\rightarrow(\sim B)))\longleftrightarrow(\sim(A\vee B))$。再分离这两个子表达式。为清楚起见，用图 3.1 的树表之。

从图 3.1 可以看其端节点全是原子公式 A，B，C，根据联结词的意义，各节点的分枝数是合理的，故该表达式是合式公式。如果树中有一个端节点为空的话，则它不是合式公式。如 $A\rightarrow$，$A\rightarrow\rightarrow B$，等表达式，就会出现空节点。

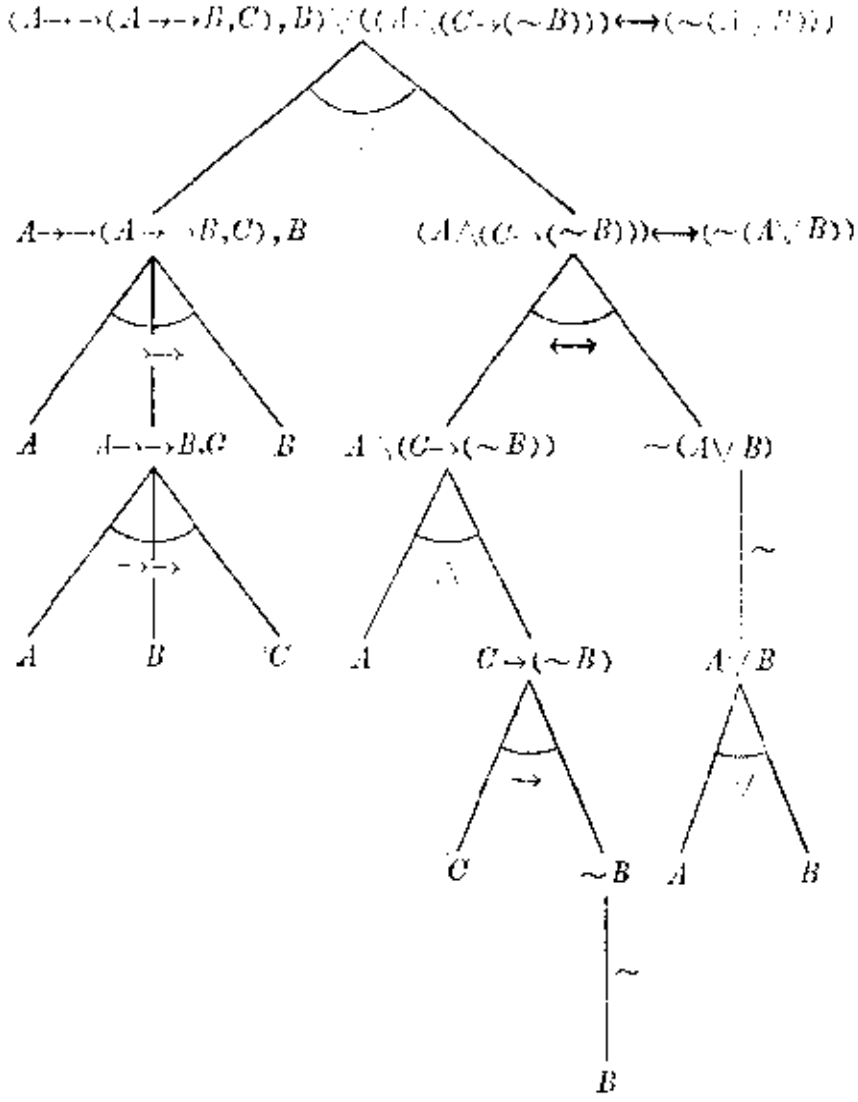


图 3.1　$(A\rightarrow\rightarrow(A\rightarrow\rightarrow B,C),B)\vee((A\wedge(C\rightarrow(\sim B)))\longleftrightarrow(\sim(A\vee B)))$ 的树

3.1.2　解释 I

在表 3.1 中有三个原子公式，就可能有 8 种不同的取值。对应于每一种取值，其合式公式就有相应的一种真值。一般说来，对于一个给定的合式公式 G，如果 G 中出现的原子公式有 n 个（设为 $A_1,\cdots A_n$），而每个 A_i 都可设定为 T 或 F 值，则就有 2^n 种不同的取值。每种取值就是一种**解释**。因此合式公式 G 的一种解释 I，就是对其原子公式 $A_1,\cdots,A_n$ 的真值的一种设定。通常称合式公式是真或假，也是指在某种解释中其值为真或假。

合式公式 G 被称为关于解释 I 为真的充要条件是 G 在这种解释中取值为真；否则 G 称为关于解释 I 为假。

解释 I 其实就是由集合$\{A_1,\cdots,A_i,\cdots A_n\}$至集合$\{m_1,\cdots,m_i,\cdots m_n\}$（其中 m_i 或取 T 或取 F 值）的一种映照。有时为了方便起见，m_i 就用 A_i 或$\sim A_i$ 来书写。如果把 A_i 映照到 T 值上，则 m_i 写为 A_i；如果把 A_i 映照到 F 值上，则 m_i 写为$\sim A_i$。其实，合式公式也是关于 I 的一种映照。

例 3.2　P≙雪是白的，Q≙人总是要死的。而解释 I 是把{P,Q}至{F,F}的一种映照。

试求 R≜（雪是白的而且人总是要死的），关于 I 的真值。

根据 I：{F,F}，则 R≜P∧Q = F∧F = F。即命题 R 关于解释 I 是假的。

3.1.3 联结词的可约性

前述的六个联结词对于命题演算来说并不是缺一不可的。实际上 A∧B 可由～((～A)∨(～B)) 来表达，即两者是等值的。所谓**等值**就是二个合适公式关于每一种解释其值都相同，也称为**逻辑恒等**。在本章中用符号“⇐⇒”表示等值关系。为简化起见，有时也常用符号“=”表示。将六个联结词的等值关系列一部分于式 (3.2) ～式 (3.7) 中

$$A \longleftrightarrow B \Longleftrightarrow (A \to B) \wedge (B \to A) \tag{3.2}$$

$$A \to\to B, C \Longleftrightarrow (A \to B) \wedge ((\sim A) \to C) \tag{3.3}$$

$$A \to B \Longleftrightarrow (\sim A) \vee B \tag{3.4}$$

$$\sim(A \vee B) \Longleftrightarrow (\sim A) \wedge (\sim B) \tag{3.5}$$

$$\sim(A \wedge B) \Longleftrightarrow (\sim A) \vee (\sim B) \tag{3.6}$$

$$A \Longleftrightarrow \sim(\sim A) \tag{3.7}$$

由上可见，只需二个联结词（～，∨）或者（～，∧）就可以代替六个联结词了。其实只需用一个联结词→→就够了。其等值性示于式 (3.8) ～式 (3.13) 中。

$$\sim A \Longleftrightarrow A \to\to F, T \tag{3.8}$$

$$A \wedge B \Longleftrightarrow A \to\to B, F \tag{3.9}$$

$$A \vee B \Longleftrightarrow A \to\to T, B \tag{3.10}$$

$$A \to B \Longleftrightarrow A \to\to B, T \tag{3.11}$$

$$A \longleftrightarrow B \Longleftrightarrow A \to\to B, (\sim B) \Longleftrightarrow A \to\to B, (B \to\to F, T) \tag{3.12}$$

$$\sim(A \to\to B, C) \Longleftrightarrow A \to\to (\sim B), (\sim C) \Longleftrightarrow A \to\to (B \to\to F, T), (C \to\to F, T) \tag{3.13}$$

为要证明二个合式公式的等值性，可采用列真值表的方法。即用列表的方法来证明二个合式公式关于各种解释的映照都相同。

例 3.3　欲证式 (3.3) 的等值式，则可列其真值表于表 3.2 中。从中可以看出二者关于各种解释 I 是等值的。

表 3.2　$A \to\to B, C$ 与 $(A \to B) \wedge ((\sim A) \to C)$ 真值表

A	B	C	～A	$A \to\to B, C$	$A \to B$	$(\sim A) \to C$	$(A \to B) \wedge ((\sim A) \to C)$
F	F	F	T	F	T	F	F
F	F	T	T	T	T	T	T
F	T	F	T	F	T	F	F
F	T	T	T	T	T	T	T
T	F	F	F	F	F	T	F
T	F	T	F	F	F	T	F
T	T	F	F	T	T	T	T
T	T	T	F	T	T	T	T

3.1.4 永真公式和永假公式

如果一个合式公式G关于各种可能解释I都是真（或假）的话，则称该合式公式G为**永真公式**（或**永假公式**）。不是永真公式（或不是永假公式）的合式公式称为**非永真公式**（或**非永假公式**）。

由该定义显然有如下性质：

(1) 合式公式为永真公式的充要条件是它的非为永假公式。

(2) 合式公式为永假公式的充要条件是它的非为永真公式。

(3) 合式公式为非永真公式的充要条件是至少存在有一种解释，使得该合式公式关于该解释为假。

(4) 合式公式为非永假公式的充要条件是至少存在有一种解释，使得该合式公式关于该解释为真。

(5) 永真公式一定是非永假公式。但非永假公式不一定是永真公式。

(6) 永假公式一定是非永真公式。但非永真公式不一定是永假公式。

例如$A\vee(\sim A)$为永真公式，$A\wedge(\sim A)$为永假公式。而$A\wedge B$为非永真公式，也为非永假公式。

欲证合式公式是永真（或永假）公式，可采用列真值表的方法。

例3.4 欲求$P\triangleq((A\rightarrow B)\wedge A)\rightarrow B$，$Q\triangleq(A\rightarrow B)\wedge(A\wedge(\sim B))$，则可列出如表3.3的真值表。可见P为永真公式，而Q为永假公式。

表3.3 P与Q的真值表

A	B	$A\rightarrow B$	$(A\rightarrow B)\wedge A$	$P\triangleq((A\rightarrow B)\wedge A)\rightarrow B$	$A\wedge(\sim B)$	$Q\triangleq(A\rightarrow B)\wedge(A\wedge(\sim B))$
F	F	T	F	T	F	F
F	T	T	F	T	F	F
T	F	F	F	T	T	F
T	T	T	T	T	F	F

3.1.5 范式

如果能将任一个合式公式变换成另一个我们所需要的统一的标准形式（称为**范式**），那么今后只要对这类范式加以深入研究就行了。本节着重研究两种范式：合取范式和析取范式。

为了定义范式，在此引入基本式的概念。所谓**基本式**，就是原子公式或原子公式的非之集合。即基本式$=\{A,\sim A\mid A$为原子公式$\}$

如果一个合式公式G是如下形式的话，则称它为**合取范式**。

$$G\triangleq A_1\wedge A_2\wedge\cdots\wedge A_n\qquad(n\geqslant 1)\tag{3.14}$$

其中A_1，A_2，…，A_n是基本式的析取。

例3.5 如P，Q，R为原子公式的话，则$G\triangleq(P\vee Q\vee(\sim R))\wedge((\sim P)\vee Q)\wedge(P\vee(\sim R))$是一个合取范式。此处$A_1=P\vee Q\vee(\sim R)$，$A_2=(\sim P)\vee Q$，$A_3=P\vee(\sim R)$都是基本式P，$\sim P$，Q，R，$\sim R$的析取。

一个合式公式G，如果是如下形式的话，则称为**析取范式**，即

$$G \triangleq A_1 \vee A_2 \vee \cdots \vee A_n \qquad (n \geqslant 1) \tag{3.15}$$

其中 A_1，A_2，$\cdots A_n$ 是基本式的合取。

例 3.6　如果 P，Q，R 为原子公式的话，则 $G \triangleq (P \wedge (\sim Q)) \vee (P \wedge R) \vee ((\sim P) \wedge (\sim R))$是一个析取范式。此处 $A_1 = P \wedge (\sim Q)$，$A_2 = P \wedge R$，$A_3 = (\sim P) \wedge (\sim R)$都是基本式 P，$\sim P$，$\sim Q$，R，$\sim R$ 的合取。

为了把一个合式公式变成合取范式或析取范式，经常用到一些等值公式。现列于 表 3.4 中。其证明留给读者。表中的 A，B，C 都是合式公式，而□表示永假公式，■表示永真公式。

表 3.4　等　值　公　式

(1) $A \vee B = B \vee A$　　(2) $A \wedge B = B \wedge A$

(3) $A \to B = (\sim B) \to (\sim A)$

(4) $A \vee (B \vee C) = (A \vee B) \vee C = A \vee B \vee C$　　(5) $A \wedge (B \wedge C) = (A \wedge B) \wedge C = A \wedge B \wedge C$

(6) $A \longleftrightarrow (B \longleftrightarrow C) = (A \longleftrightarrow B) \longleftrightarrow C$

(7) $A \vee (B \wedge C) = (A \vee B) \wedge (A \vee C)$　　(8) $A \wedge (B \vee C) = (A \wedge B) \vee (A \wedge C)$

(9) $A \vee A = A$　　(10) $A \wedge A = A$

(11) $A \vee □ = A$　　(12) $A \wedge ■ = A$

(13) $A \vee ■ = ■$　　(14) $A \wedge □ = □$

(15) $A \vee (\sim A) = ■$　　(16) $A \wedge (\sim A) = □$

(17) $\sim(\sim A) = A$

(18) $\sim(A \vee B) = (\sim A) \wedge (\sim B)$　　(19) $\sim(A \wedge B) = (\sim A) \vee (\sim B)$

(20) $A \to B = (\sim A) \vee B$

(21) $■ \to B = B$　　(22) $□ \to B = ■$

(23) $A \to ■ = ■$　　(24) $A \to □ = \sim A$

(25) $A \longleftrightarrow B = (A \to B) \wedge (B \to A) = ((\sim A) \vee B) \wedge ((\sim B) \vee A) = (A \wedge B) \vee ((\sim A) \wedge (\sim B))$

(26) $A \longleftrightarrow ■ = A$　　(27) $A \longleftrightarrow □ = \sim A$

(28) $A \to\to B, C = (A \to B) \wedge ((\sim A) \to C) = ((\sim A) \vee B) \wedge (A \vee C) = (A \wedge B) \vee (B \wedge C) \vee ((\sim A) \wedge C)$

(29) $A \to\to ■, ■ = ■$　　(30) $A \to\to □, □ = □$

(31) $A \to\to ■, □ = A$　　(32) $A \to\to □, ■ = (\sim A)$

(33) $A \to\to ■, C = A \vee C$　　(34) $A \to\to □, C = (\sim A) \wedge C$

(35) $A \to\to ■, (\sim A) = ■$　　(36) $A \to\to □, A = □$

(37) $A \to\to B, ■ = (\sim A) \vee B$　　(38) $A \to\to B, □ = A \wedge B$

(39) $A \to\to A, ■ = ■$　　(40) $A \to\to (\sim A), □ = □$

给定一个合式公式，通过以下几个基本步骤可以变换成所需的范式，而且新的范式与原来的合式公式是等值的。

(1) 应用表 3.4 中的(20)～(40)，可消去联结符→，←→，→→，而得到只有联结符～，∧，∨的合式公式。

(2) 反复应用表中的(15)～(19)，可以消去那些不是原子公式的合式公式的联结符～，

而使得带有～的合式公式都是原子公式。

(3) 反复应用表 3.4 中的(7)(8)及其他的等值公式加以化简，就可以得到合取范式或析取范式。

以上几个步骤在具体变换中，如能灵活交替使用，则可以加速完成。但对于让机器自动证明定理时，如上述的机械性步骤可能是有利的。

例 3.7 将 $A\rightarrow\rightarrow(B\rightarrow(B\vee C)),(A\leftrightarrow B)$ 化为范式的

原式 $=A\rightarrow\rightarrow((\sim B)\vee(B\vee C)),((A\wedge B)\vee((\sim A)\wedge(\sim B)))$

$=A\rightarrow\rightarrow\blacksquare,((A\wedge B)\vee((\sim A)\wedge(\sim B)))$

$=A\vee(A\wedge B)\vee((\sim A)\wedge(\sim B))$ (已是析取范式)

$=((A\vee(\sim A))\wedge(A\vee(\sim B)))\vee(A\wedge B)$

$=(\blacksquare\wedge(A\vee(\sim B)))\vee(A\wedge B)$

$=A\vee(\sim B)\vee(A\wedge B)$ (也是析取范式)

$=(A\vee(\sim B)\vee A)\wedge(A\vee(\sim B)\vee B)$ (是合取范式)

$=(A\vee(\sim B))\wedge\blacksquare$ (也是合取范式)

$=A\vee(\sim B)$ (也是析取范式)

由例 3.7 可以看出：(1)析取范式与合取范式可以互换。

(2) 其范式不是唯一形式的。

3.1.6 逻辑推论

对于合式公式 $F_1,F_2,\cdots F_n$ 及 G，如果对于能使 $F_1\wedge F_2\wedge\cdots\wedge F_n$ 为真的所有解释 I，G 关于解释 I 也为真的话，则称 G 为 F_1，F_2，…，F_n 的**逻辑推论**。即 G 是从 $F_1,F_2,\cdots,F_n$ 逻辑推理而来的。$F_1,F_2,\cdots,F_n$ 就称为 G 的**前提**。反过来也成立。

根据这个定义，可得出以下两条定理：

定理 3.1 对于合式公式 F_1，F_2，…，F_n 和 G，G 是 F_1，F_2，…，F_n 的逻辑推论的充要条件是合式公式 $(F_1\wedge F_2\wedge\cdots\wedge F_n)\rightarrow G$ 为永真公式。

证：先证必要性。设 G 是 F_1，F_2，…，F_n 的逻辑推论。令 I 是一个任意解释，则根据定义，如果 F_1，F_2，…，F_n 关于解释 I 都为真的话，G 关于任何解释 I 也为真。因而 $(F_1\wedge F_2\wedge\cdots\wedge F_n)\rightarrow G$ 关于 I 为真。另一方面，如果 F_1，F_2，…F_n 关于 I 有一个为假的话，则由于 $F_1\wedge F_2\wedge\cdots\wedge F_n$ 为假，故 $(F_1\wedge F_2\wedge\cdots F_n)\rightarrow G$ 关于 I 为真。因此，$(F_1\wedge F_2\wedge\cdots\wedge F_n)\rightarrow G$ 关于任一解释为真。故 $(F_1\wedge F_2\wedge\cdots\wedge F_n)\rightarrow G$ 为永真公式。

再证充分性。设 $(F_1\wedge F_2\wedge\cdots\wedge F_n)\rightarrow G$ 是永真公式。对于任一解释 I，如果 $(F_1\wedge F_2\wedge\cdots\wedge F_n)$ 关于 I 是真，则 G 必为真。因此根据定义，G 是 F_1，F_2，…，F_n 的逻辑推论。

定理 3.2 对于合式公式 F_1，F_2，…，F_n 和 G，G 是 F_1，F_2，…，F_n 的逻辑推论之充要条件是合式公式 $F_1\wedge F_2\wedge\cdots\wedge F_n\wedge(\sim G)$ 为永假公式。

证明：由于 $F_1\wedge F_2\wedge\cdots\wedge F_n\wedge(\sim G)=\sim(\sim(F_1\wedge F_2\wedge\cdots\wedge F_n)\vee G)=\sim((F_1\wedge F_2\wedge\cdots\wedge F_n)\rightarrow G)$，如果 $(F_1\wedge F_2\wedge\cdots\wedge F_n)\rightarrow G$ 为永真公式的话，则 $\sim((F_1\wedge F_2\wedge\cdots\wedge F_n)\rightarrow G)$ 为永假公式。即 $F_1\wedge F_2\wedge\cdots\wedge F_n\wedge(\sim G)$ 为永假公式。把这些应用到定理 3.1，就可得出：G 是 $F_1,F_2,\cdots,F_n$ 的逻辑推论之充要条件是 $(F_1\wedge F_2\wedge\cdots\wedge F_n\wedge(\sim G))$ 是永假公式。

定理 3.1 和定理 3.2 相当重要，给我们予启示：欲证某特定合式公式的一个有限集合的

逻辑推论，则等效于证明某个对应合式公式是永真公式或永假公式。

如果G是$F_1,F_2,\cdots,F_n$的逻辑推论，则合式公式$(F_1\wedge F_2\wedge\cdots\wedge F_n)\rightarrow G$称为定理，而G也称为定理的结论。象其他领域一样，在数学领域中，也可以把许多问题公式化为定理证明的问题。

例3.8　令合式公式$F_1\triangleq(P\rightarrow Q)$，$F_2\triangleq\sim Q$，$G\triangleq\sim P$，试证G是$F_1$和$F_2$的逻辑推论。

证法1　根据定义用真值表方法指出对于能使$F_1\wedge F_2$（即$(P\rightarrow Q)\wedge(\sim Q)$）为真的所有解释I，G（即$\sim P$）关于解释I也为真。真值表示于表3.5中。从表中可见，使$(P\rightarrow Q)\wedge(\sim Q)$为真的所有解释只有一个$\{\sim P,\sim Q\}$。关于该解释$\{\sim P,\sim Q\}$，$(\sim P)$也为真。故G是$F_1$和$F_2$的逻辑推论。

表3.5　$(P\rightarrow Q)\wedge(\sim Q)$与$\sim P$的真值表

P	Q	$P\rightarrow Q$	$\sim Q$	$(P\rightarrow Q)\wedge(\sim Q)$	$\sim P$
F	F	T	T	T	T
F	T	T	F	F	T
T	F	F	T	F	F
T	T	T	F	F	F

证法2　根据定理3.1，证明$((P\rightarrow Q)\wedge(\sim Q))\rightarrow(\sim P)$为永真公式。而在此又有两种证明方法。一种是像证法1那样列出真值表，在此从略。另一种是把它变成析取范式，而最终证明它为永真公式。用该法把原式推导如下：

$$\begin{aligned}((P\rightarrow Q)\wedge(\sim Q))\rightarrow(\sim P) &= (\sim((P\rightarrow Q)\wedge(\sim Q)))\vee(\sim P)\\ &= (\sim(((\sim P)\vee Q)\wedge(\sim Q)))\vee(\sim P)\\ &= (\sim(((\sim P)\wedge(\sim Q))\vee(Q\wedge(\sim Q))))\vee(\sim P)\\ &= (\sim(((\sim P)\wedge(\sim Q))\vee\square))\vee(\sim P)\\ &= (\sim((\sim P)\wedge(\sim Q)))\vee(\sim P)\\ &= (P\vee Q)\vee(\sim P)\\ &= Q\vee\blacksquare\\ &= \blacksquare\end{aligned}$$

证法3　根据定理3.2，证明$((P\rightarrow Q)\wedge(\sim Q))\wedge(\sim(\sim P))$为永假公式。同样也可用真值表法，但还可用公式推导如下。

$$\begin{aligned}(P\rightarrow Q)\wedge(\sim Q)\wedge(\sim(\sim P)) &= ((\sim P)\vee Q)\wedge(\sim Q)\wedge P\\ &= (((\sim P)\wedge(\sim Q))\vee(Q\wedge(\sim Q)))\wedge P\\ &= (((\sim P)\wedge(\sim Q))\vee\square)\wedge P\\ &= (\sim P)\wedge(\sim Q)\wedge P\\ &= \square\wedge(\sim Q)\\ &= \square\end{aligned}$$

3.2 一阶谓词演算

虽然命题演算能表达一些思想、证明一些定理和求解一些问题。但是有许多思想是无法用命题来表达的。例如，一个简单的三段论：P≜人总是要死的。Q≜张三是人。R≜张三是要死的。显然，P是大前提，Q是小前提，R是P和Q的逻辑推论。但是在命题演算中却无法证明这个结果。这是由于在命题中不考虑陈述中的主词和谓词之间的关系。**谓词演算**就是研究这些问题的。

命题演算中所使用的符号、联结词及一些演算规则，除了特加叙述之外，都适用于谓词演算。谓词演算的原子公式一般理解为陈述中的谓词。例如上例中的P,Q应理解为“是要死的”，“是人”。有了命题演算的基础，对谓词演算就较易理解了。

3.2.1 基本概念

一阶谓词演算的组成要素有八个，其中也涉及到一些基本定义。分述如下：

1. 标点符号：，、（、）。

2. 逻辑符号（联结词）：此处的逻辑符号～、∧、∨、→、→→、←→，与命题演算的六个联结词之意义完全相同。在命题演算中所讨论的性质，在此也适用。

3. 常量：常用事物的英文名字或汉语拼音或标识符或数来表示。例如，BEIJING，FATHER，9，8，…

4. 变元：常用小写拉丁字母x,y,z等表示，代表未确定的对象。

如果在所讨论问题的范围内所有的常量或变元的取值都是某个非空集合D的成员，则称D为所研究表达式的**定义域**或**论域**。

5. **n位函词**：常用小写拉丁字母及其后n个空位来表示，如f（　，…，　）其中有n个空位。**零位函词**称为**常量**，就可直接用小写拉丁字母表示，如f,g,h。

n位函词中的n个空位可以用常量或变量填入，所以n位函词是一个从定义域D的n次笛卡尔积集合D^n到值域A（A⊂D）的映照。有时就简化A为D。简记为$f:D^n\to D$。例如，若一位函词$father$（　）表示“…的父亲”，则$father(x)$表示“x(变元)的父亲”。定义域D为“人的集合”，值域A为“有子女的男人的集合”。A⊂D。

又如，若二位函词jia（　，　）表示“…+…”，则$jia(4,3)$表示4+3。

$father(x)$并没有表达出完整的陈述，无法判别该陈述是真是假。故称$father(x)$为项。

通常，**项**的递归定义如下：

(1) 常量是一个项。

(2) 变元是一个项。

(3) 如果f（　，…，　）为n位函词，而且$t_1,\cdots,t_n$是项，则$f(t_1,\cdots,t_n)$也是一个项。

(4) 所有的项必须是有限次地应用上述规则而产生的。

例如，$father$（$father(x)$）也是项，表示为x的祖父。而jia（$jia(x,3),1$）也是项，表示$(x+3)+1$。

6. **n位谓词**：常用大写拉丁字母及其后n个空位来表示，如P（　，…，　），其中有n个空位。**零位谓词**称为命题，直接用大写拉丁字母表示，如P，Q。

n位谓词的n个空位可以用项填入，而表达成完整的陈述。

例如，若一位谓词 HONG（ ）表示"…戴红帽子"。则 HONG(x)表示 "x 戴红帽子"。

又如，若二位谓词 FATHER(,)表示 "…是…的父亲"，则 FATHER(x, $father$ (y))表示 x 是 y 的祖父。

从上例可看出，n 位谓词一旦由项填入空位，就能表达完整的陈述，而且还可判定该陈述是真是假。因而，n 位谓词 P(,…,)是一个从定义域 D 的 n 次笛卡尔积集合 D^n 到真假集合{T,F}的映照。简记为P：$D^n \to \{T,F\}$。

n 位谓词一旦其空位填入项时，就是一个原子公式了。

通常，原子公式的递归定义如下：

(1) 命题（零位谓词）是原子公式。

(2) 如P(,…,) 是 n 位谓词，而 $t_1,\cdots,t_n$ ($n\geqslant 1$) 是项，则表达式 P ($t_1,\cdots,t_n$) 是原子公式。

(3) 其他的表达式不能称为原子公式。

例如，HONG ($father(x)$) 是原子公式，而 $father$ (HONG(x)) 不是原子公式。

有了原子公式的定义之后，可得出**合式公式**的递归定义如下：

(1) 原子公式是合式公式。

(2) 如果 A,B,C 是合式公式，则～A，A∧B，A∨B，A→B，A→→B,C，及 A←→B 都是合式公式。

(3) 如果 A 是合式公式，且 x 是在 A 中的自由变元，则$(\forall x)$A，$(\exists x)$A 都是合式公式。

(4) 所有的合式公式都只能有限次地应用以上三条规定而产生。

以上定义中的第(1)、(2)条是很易理解的。例如，FATHER(x,y)→→HONG(x)，(～HONG(x)) 与FATHER(x,y)都是合式公式。

至于定义中的第(3)条，后面立即介绍。为了使定义完整起见，在此也列出来，读者可在理解了后面的第7、8个组成要素之后，再回来复习一下这个定义。

7. **全称量词**：常用符号（∀ ）来表示。一般出现的形式是 $(\forall x)$ M (x,y)，其中 x 为个体变元，M(x,y)是含有变元 x 的合式公式（这里用 y 来表示除了 x 之外的其他多个或者一个的变元）。如果 M(x,y)是由二个以上谓词所组成，为了清楚地表示 ∀ 的作用域，则规定对 M(x,y)必须加一对括弧来表示 ∀ 的作用范围。例如，如果 $M(x,y)=P(x,y)\wedge Q(x,y)$，则 $(\forall x)M(x,y)=(\forall x)(P(x,y)\wedge Q(x,y))$。而$(\forall x)P(x,y)\wedge Q(x,y)=((\forall x)P(x,y))\wedge Q(x,y)\neq(\forall x)(P(x,y)\wedge Q(x,y))$。

$(\forall x)M(x,y)$ 之意义是"对于变元 x 在其定义域 D 中的所有取值（或叙述为"任一取值"）都能使 M(x,y) 为真"。换句话说，如果变元 x 在定义域 D 中的可能取值为x_1，x_2，…，x_n，则$(\forall x)M(x,y)$为真的充要条件是 $M(x_1,y)\wedge\cdots\wedge M(x_n,y)$为真。当然，$x$ 的取值可能有无限多个，要判定 $M(x_1,y)\wedge\cdots\wedge M(x_n,y)\wedge\cdots$为真是很困难的。但是要判定$(\forall x)M(x,y)$ 为真是可能的，只要能将 M(x,y)推导成这种形式 $(\forall x)(P(x,y)\vee(\sim P(x,y)))$ 即可获得判定。

在全称量词之后紧接出现的变元称为**"全称量词化变元"**。全称量词化变元无论是在紧接着其全称量词之后的出现或者是在全称量词的作用域内的出现，都称为**约束出现**。变元的出现如果是非约束的话，则称为**自由出现**。例如，$(\forall x)M(x,y)$中的第一个 x 及第 2 个 x 的

出现都是约束出现，而变元 y 在此合式公式中的出现却是自由出现。

如果变元在合式公式中的出现至少有一次是约束出现，则称该变元是在合式公式中的**约束变元**。如果变元在合式公式中的出现至少有一次是自由出现，则称该变元在合式公式中是**自由变元**。

可见，一个变元可以同时为约束变元和自由变元。例如 $((\forall x)M(x,y))\wedge((\forall y)N(y))$ 中的变元 y 在此出现三次。第一次是自由出现，而第二、三次是约束出现。

有了全称量词及谓词，有些在命题演算中无法解决的问题，在此就可解决了。

例 3.9 如果“人总是要死的”，“张三是人”为真，则“张三是要死的”也为真。可以这样来证明：

令 M( )≜“…是人”，D( )≜“…是要死的”。

则 $A_1\triangleq(\forall x)(M(x)\rightarrow D(x))$

$A_2\triangleq M(ch)$

$A_3\triangleq D(ch)$

即欲证：若 A_1,A_2 为真，则 A_3 也为真。

由于 $M(x)\rightarrow D(x)$ 对于所有的 x 都为真，所以 x 用 ch 代入也为真。即 $M(ch)\rightarrow D(ch)$ 也为真。又 $M(ch)$ 为真，故 $D(ch)$ 也为真。

8. **存在量词**：常用符号（∃ ）来表示。一般出现的形式是 $(\exists x)M(x,y)$，其中 x 为个体变元，$M(x,y)$ 是含有变元 x 的合式公式（在此，同样也用变元 y 表示除了 x 之外的其它变元）。

$(\exists x)M(x,y)$ 之意义是“对于变元 x 在其定义域 D 中至少有一个取值（或叙述为“存在着一个取值”）能使 $M(x,y)$ 为真”。换句话说，如果变元 x 在定义域 D 中的可能取值为 $x_1,\cdots,x_i,\cdots$（可能有无限多个），则 $(\exists x)M(x,y)$ 为真的充要条件是 $M(x_1,y)\vee\cdots\vee M(x_i,y)\vee\cdots$ 为真。当然，要判定它为真是很容易的。而要判定它为假，则可想法把它推导成这种形式 $(\exists x)(M(x,y)\wedge(\sim M(x,y)))$。

“存在量词化变元”、“自由变元”、“约束变元”以及“自由出现”、“约束出现”等等概念，与全称量词部分类似，不赘述。

例 3.10 令一元函词 s（ ）表示“…是一条直线”。二元谓词 P(,)表示“…平行于…”。那么，$(\exists x)$ $P(s(x),s(a))$ 为真吗？

(Ex) $P(s(x),s(a))$ 的意义是“对于已知直线 a，可找到一条直线，使之平行于 a”。欲证其为真的话，并不要求所有的直线都平行于 a（事实上不可能做到这点），只需找到一条直线平行于 a 即可。显然这是很容易找到的。故 $(\exists x)P(s(x),s(a))$ 为真。

在一阶谓词演算中除了全称量词 $\forall$ 和存在量词 ∃ 符号之外，其他符号的基本等值关系与命题演算中所述的相同。在此着重介绍与量词有关的几个基本的等值关系式。

量词与“非”的关系如下：

$$\sim((\forall x)M(x,y))=(\exists x)(\sim M(x,y)) \tag{3.16}$$

$$\sim((\exists x)M(x,y))=(\forall x)(\sim M(x,y)) \tag{3.17}$$

量词有关的分配律如下：

$$(\forall x)M(x,y)\vee G(y)=(\forall x)(M(x,y)\vee G(y)) \tag{3.18}$$

$$(\exists x)M(x,y)\vee G(y)=(\exists x)(M(x,y)\vee G(y)) \tag{3.19}$$

$$(\forall x)M(x,y)\wedge G(y)=(\forall x)(M(x,y)\wedge G(y)) \tag{3.20}$$

$$(\exists x)M(x,y)\wedge G(y)=(\exists x)(M(x,y)\wedge G(y)) \tag{3.21}$$

式(3.16)～式(3.21)是很容易理解的。但要注意，对于$(\forall x)M(x,y)\vee G(x)$，则应理解为$(\forall z)(M(z,y)\vee G(x))$，而不理解为$(\forall x)(M(x,y)\vee G(x))$。因为$G(x)$中的变元$x$不是约束变元，而且在合式公式中的每个约束变元可认为是一种**虚拟变元**，$(\forall x)M(x,y)=(\forall z)M(z,y)$。这常称之为**约束变元换名法则**。注意，换名时，新名不能重新出现在该合式公式中除了被换名的位置以外的其它位置上，以免再重新引起混淆。例如$(\forall x)M(x,y)$中把x换成y而得$(\forall y)M(y,y)$是不行的，因为新名y在未取代过的位置（第三个y）上又重新出现了。

量词有关的分配律还有：

$$(\forall x)M(x,y)\wedge(Ax)N(x,y)=(\forall x)(M(x,y)\wedge N(x,y)) \tag{3.22}$$

$$(\exists x)M(x,y)\vee(\exists x)N(x,y)=(\exists x)(M(x,y)\vee N(x,y)) \tag{3.23}$$

以上二式的证明是很容易的。留给读者作为练习。

但是要注意，$\forall$关于$\vee$的分配律，$\exists$关于$\wedge$的分配律却是不成立的。即

$$(\forall x)M(x,y)\vee(\forall x)N(x,y)\neq(\forall x)(M(x,y)\vee N(x,y))$$

$$(\exists x)M(x,y)\wedge(\exists x)N(x,y)\neq(\exists x)(M(x,y)\wedge N(x,y))$$

例3.11　令$M(x)$表示x是老人，$N(x)$表示x是儿童。而x的定义域D是老人与儿童。可见，“$M(x)\vee N(x)$”表示“或是老人或是儿童”，“$M(x)\wedge N(x)$”表示“既是老人又是儿童”。显然，$(\forall x)(M(x)\vee N(x))$为真，$(\exists x)(M(x)\wedge N(x))$为假。但是$(\forall x)M(x)$及$(\forall x)N(x)$都为假，而$(\exists x)M(x)$及$(\exists x)N(x)$都为真。因此上述二个不等式是成立的。

虽然如此，但对于这二个分配律若采用换名法则，还是可以成立的。甚至于推而广之而得如下四式。

$$(\forall x)M(x,y)\vee(\forall x)N(x,y)=(\forall x)(\forall z)(M(x,y)\vee N(z,y)) \tag{3.24}$$

$$(\exists x)M(x,y)\wedge(\exists x)N(x,y)=(\exists x)(\exists z)(M(x,y)\wedge N(z,y)) \tag{3.25}$$

$$(\forall x)M(x,y)\vee(\exists x)N(x,y)=(\forall x)(\exists z)(M(x,y)\vee N(z,y)) \tag{3.26}$$

$$(\forall x)M(x,y)\wedge(\exists x)N(x,y)=(\forall x)(\exists z)(M(x,y)\wedge N(z,y)) \tag{3.27}$$

3.2.2　前束范式

在命题演算中曾有合取范式和析取范式。在一阶谓词演算中也有一个范式，称为前束范式。有了前束范式可以简化证明过程。

前束范式　一阶谓词演算中的合式公式P称为**前束范式**形式的充要条件是该合式公式表示如下形式：

$$P\triangleq(Q_1x_1)\cdots(Q_nx_n)M \tag{3.28}$$

其中每个(Q_ix_i)，$i=1,\cdots,n$，或是$(\forall x_i)$或是$(\exists x_i)$，而且M是一个不包含量词的合式公式。$(Q_1x_1)\cdots(Q_nx_n)$称为**前束**，M称为合式公式P的**母式**。

可见，前束范式的特点是量词都在最左面，而且其他联结符都在母式中。

任何合式公式都可以通过以下步骤变为前束范式。

步骤1　消去多余的前束。有时，在母式的化简过程中，其某些合式公式由于推导出永

真公式（~P(x)∨P(x)）或永假公式（~P(x)∧P(x)）而消去，就可能使得某些约束变元也随之在母式中消失。因而可能出现在母式中没有其前束中的变元的情况。该前束是多余的，应该消去。这个步骤在化简开始或在今后步骤中都应随时采用。

步骤 2　反复应用式(3.2)～式(3.4)，使所有的联结符↔，→，→消去。

步骤 3　反复应用式(3.5)～式(3.7) 以及式(3.16)、式 (3.17)，使所有的符号～都 在紧接于原子公式的前面。

步骤 4　如有需要，可采用变元换名法则（例如可应用式(3.24)～式(3.27)）。

步骤 5　反复应用式(3.18)～式(3.27)，就可把量词移至整个公式的最左面，从而得到一个前束范式。

例 3.12　将如下合式公式化为前束范式

$$(\forall x)(\forall y)(\exists z)((\forall u)P(x,u)\rightarrow((\exists x)(((P(x,y)\rightarrow Q(x,y))\vee P(x,y))\wedge Q(y,z))))$$

$$\begin{aligned}
\text{原式} &= (\forall x)(\forall y)(\exists z)(\sim(\forall u)P(x,u)\vee((\exists x)(((\sim P(x,y)\vee Q(x,y))\vee P(x,y))\wedge Q(y,z))))\\
&= (\forall x)(\forall y)(\exists z)((\exists u)(\sim P(x,u))\vee((\exists x)Q(y,z)))\\
&= (\forall x)(\forall y)(\exists z)((\exists u)(\sim P(x,u))\vee Q(y,z))\\
&= (\forall x)(\forall y)(\exists z)(\exists u)(\sim P(x,u)\vee Q(y,z))
\end{aligned}$$

3.2.3　Skolem 标准形

从命题演算可知，前束范式中的母式M还可变换成合取范式和析取范式。前者称为**前束合取范式**，后者称为**前束析取范式**。

前束范式的前束部分中全称量词与存在量词之间的先后顺序不可调换。例如$(\forall x)(\exists y)P(x,y)$与$(\exists y)(\forall x)P(x,y)$有不同意义。$(\forall x)(\exists y)P(x,y)$表示“对于 x 的任一取值，至少存在着一个 y，使 $P(x,y)$为真”。此处 y 的确定是与 x 有关的，$y=g(x)$。而 $(\exists y)(\forall x)P(x,y)$ 表示“至少存在着一个y，使得对于 x 的任一取值，$P(x,y)$ 都为真”，此处 y 的取值与 x 无关，$y=a$。可见对于存在量词的约束变元，只要适当地选取其数值，存在量词也可消去。

Davis 和 Putnam 的基本思想就是：(1) 把合式公式变换成前束合取范式；(2)只要不产生前后矛盾，可适当选取 Skolem函词，把前束部分的存在量词消去，而得到 Skolem 标准形。

Skolem函词的定义如下：

令$(Q_1x_1)\cdots(Q_nx_n)$M是前束合取范式（其中 $Q_j(j=1,\cdots n)$为 ∀ 或 ∃）。设 $Q_r(1\leqslant r\leqslant n)$ 是前束部分中的一个存在量词 ∃。

(1) 若 Q_i $(1\leqslant i\leqslant r)$ 都是存在量词（即 Q_r 前面都是存在量词）或者 $r=1$（即 Q_r 是最前面的存在量词）的话，则选择与M中出现的常量有所区别的新常量 c，来取代M中出现的所有 x_r。从而把前束部分中的 (Q_rx_r) 消去。

(2) 若 Q_i $(1\leqslant i<r)$ 有全称量词，设这些全称量词为 Q_{s1}，$Q_{s2}\cdots Q_{sm}$ $(1\leqslant s_1<s_2<\cdots<s_m<r)$，则选择与M出现的函词符号有所区别的新的 m 位函词符号 f，而后以 $f(x_{s1},x_{s2},\cdots,x_{sm})$ 取代M中出现的所有 x_r。从而把前束部分中的 (Q_rx_r) 消去。

在前束合取范式中用于取代母式中存在量词化变元的常量和函词，称为 **Skolem 函词**。

简称S 函词。

在对前束部分中所有的存在量词都应用以上过程之后所得到的新的前束合取范式，就称为 **Skolem 标准形**。简称 **S 标准形**。

例 3.13　化前束合取范式$(\exists x)(\exists y)(\forall z)(\exists u)(\forall v)(\exists w)(P(x,y,z,u,v,w)\wedge Q(x,y,z,u,v,w))$为 S 标准形。其中，$(\exists x)$之前没有量词，$(\exists y)$之前没有全称量词，故 x, y 的 S 函词分别是常量 a,b。就用 a,b 分别代入母式的谓词 P，Q 中的 x,y。$(\exists u)$之前只有$(\forall z)$，故 u 的 S 函词为一位函词 $f(z)$。$(\exists w)$之前有$(\forall z)$及$(\forall v)$，故 w 的 s 函词为二位函词 $g(z,v)$。将各函词取代母式中相应的变元，就得到其 S 标准形为

$$(\forall z)(\forall v)(P(a,b,z,f(z),v,g(z,v))\wedge Q(a,b,z,f(z),v,g(z,v)))$$

注意，在将合式公式变换成 S 标准形的过程中，消去存在量词符号时总是取用尽可能简单的 S 函词的，即采用变元数目最少的 S 函词。但并不是说一个合式公式的 S 标准形只有一个。而是有多个。因此，将一个合式公式W变换成 S 标准形W_s之后，W_s仅是W的特例而已。如果W为非永假公式时，通常，W与W_s两者并不等价。

例如，若$W\triangleq(\exists x)P(x)$且$W_s\triangleq P(a)$，显然，$W_s$是W的S标准形。然而，如果 I 是如下定义的解释：定义域 $D=\{1,2\}$。而且 a 的设定值为 1，关于 P 的设定值为：$P(1)=F, P(2)=T$。那么，W显然关于 I 为真，但W_s关于 I 却为假。可见W与W_s不等价。

但是，如果合式公式W为永假公式时，W的 S 标准形 W_s 也为永假公式。由 W 变换成 W_s 时，并不影响其永假特性。

如果将基本式的析取式称为**子句**的话，这些子句所组成的集合就称为 **子 句 集**。可见。S 标准形的母式就是子句的合取式。

例如$((\sim P(x))\vee(\sim P(y))\vee P(f(x,y)))\wedge((\sim P(x))\vee Q(x,g(x)))\wedge((\sim P(x))\vee(\sim P(g(x))))$有三个子句 $(\sim P(x))\vee(\sim P(y))\vee P(f(x,y))$，$(\sim P(x))\vee Q(x,g(x))$ 及 $(\sim P(x))\vee(\sim P(g(x)))$。

不含有基本式的子句称为**空子句**。可见，空子句没有任何能满足某种解释的子句。因而空子句的取值总为假。故也用符号□来表示空子句。

由于 S 标准形的母式中的变元都是全称量词化的，所以其所有的子句中的变元也是全称量词化的。可见，可以把 S 标准形简单地以其母式来取代，而且其母式又都是子句的合取式。如果把这些子句组成一个集合的话，称该子句集为其标准形的**子句集**。因而可进一步把S 标准形简单地以其对应的子句集来取代。

如果子句集的所有成员的合取式不为真，则称该子句集为**不可满足**。

定理 3.3 是相当重要的，由于篇幅所限，其证明从略。

定理 3.3　若 S 是合式公式W的 S 标准形之子句集，则W为永假公式的充要条件是 S 为不可满足。

甚而，如果$W=W_1\wedge W_2\wedge\cdots\wedge W_n$(W，$W_i$ 都为合式公式)，又若分别得到 W_i 的 S 标准形的子句集 S_i，$i=1,\cdots,n$，那么令 $S=S_1\cup S_2\cup\cdots\cup S_n$，则从定理 3.3 不难看出，W为永假公式的充要条件是 S 为不可满足。因此在求证定理时，如有 $W=W_1\wedge W_2\wedge\cdots\wedge W_n$ 形式，则不必要先求W的 S 标准形而后再求 S。可以对W_i分别求标准形及其 S_i，最后再求出 $S=S_1\cup S_2\cup\cdots\cup S_n$。

从定理 3.2 和定理 3.3 可知，W为永真公式的充要条件是～W 的 S 标准形的子 句 集 S 为不可满足的。

从现在起，假设所有欲证的问题都归结成子句集的形式。因而对于子句集，将用“不可满足”（或“可满足”）来代替“永假”（或“永真”），用符号□（或■）表示。

例 3.14　欲求$W = W_1 \wedge W_2 \wedge W_3$的子句集。其中

$$W_1 = (\forall x)(\forall y)(\exists z)P(x,y,z)$$

$$W_2 = (\forall u)Q(u,a)$$

$$W_3 = (\forall v)P(a,b,v)$$

则可分别求出W_1，W_2和W_3的子句集 S_1，S_2 和 S_3。

$S_1 = \{P(x,y,f(x,y))\}$，$S_2 = \{Q(u,a)\}$，$S_3 = \{P(a,b,v)\}$。

故 $S = \{P(x,y,f(x,y)), Q(u,a), P(a,b,v)\}$。

3.2.4　Herbrand 全域和 Herbrand 基

根据定义，合式公式W为永假公式的充要条件是对于整个定义域上的所有解释，W的值为假。为方便起见，首先可以把它确定在某一个特殊定义域 D 上，使得W为永假公 式 的 充要条件是对于该定义域 D 上的所有解释，W为假。而后从该特殊定义域 D 拓广至整个定义域上。对任何一个合式公式W，确实有这样一个定义域，称为 **Herbrand 全域**简称 **H 全域**。用 H(S)表示W的子句集 S 的 H 全域。

合式公式W的子句集 S 的 H 全域，递归定义如下：

(1) 在 S 中出现的所有个体常量的集合属于 H(S)。如果 S 中不出现个体常量，则就设定任意一个个体常量（记为 a），a 属于 H(S)。

(2) 假设项 $t_1,\cdots,t_n$ 在 H(S)之中，则 S 中的 n 位函词 $f(\ ,\cdots,\ )$用 t_i 代入而得的 $f(t_1,\cdots,t_n)$也属于 H(S)。

(3) 只有以上定义的项才属于 H(S)。

由于对基本式仍可设定 T 和 F 值，故 H(S)本身也是一个最一般的定义域了。如果能指出在 H(S)上 S 是不可满足的话，则就能确保在任意定义域上 S 也是不可满足的。

通常，H(S)是无限的，但却是可列个的。所以它的成员总是可以按某方法整理 次 序。

例 3.15　子句集 S 为　　$\{P(x) \vee Q(a) \vee \sim P(f(x)),\ \sim Q(b) \vee P(g(x,y))\}$

S 中出现的常量为 a，b。而出现 的函词有一位函词 $f(\ )$及二 位函 词 $g(\ ,\ )$。H(S)则是可列个的无限集合$\{a,\ b,\ f(a),\ f(b),\ g(a,a),\ g(a,b),\ g(b,a),\ g(b,\ b),\ f(f(a)),\ f(f(b)),\ g(a,\ f(a)),\ g(f(a),\ a),\ \cdots\}$。

如果对于 S 中的子句给予 H(S) 上的一个解释，那么对于它的原子公式就指定了 真 值（T 或 F）。假设 $P(\ ,\cdots,\ )$是 S 中的 n 位谓词。原子公式 $P(x_1,\cdots,x_n)$ 中 的 变元 x_1，$\cdots,x_n$ 用 H(S)中的成员代入之后就得到 $P(x_1,\cdots,x_n)$的基本例子，$P(x_1,\cdots,x_n)$ 的值是根据所有这些基本例子而定的。由于 H(S)常常是无限集合，故每个原子公式的基本例子也常常是无限多个。要全部计算出来是不可能的。但是，并不要全部把它们算出来就可以证明 S 是不可满足的。

在 S 中的所有原子公式都用 H(S)的成员代入之后的所有基本例子所组成的集合，定义为 S 的 **Herbrand 基**，简称为 **H 基**。H 基的所有成员称为**原子**。显然，H 基也是一个可列个

的集合，且可以整理次序。理过序的H基可表示成有序集合{$P_1,P_2,\cdots$}。

例3.16　例3.15中的原子公式有$P(x),Q(x)$，故S的H基为

$$\{P(a),Q(a),P(b),Q(b),P(f(a)),Q(f(a)),P(f(b)),Q(f(b)),\cdots\}$$

3.2.5　语义树

语义树是一棵二值外向树。所谓**二值树**是该树的每个节点，如果有子节点的话，一定是恰恰有二个子节点。当然可以把语义树的概念扩大至具有二个以上分枝的k值外向树。但是在此所用的语义树是二值外向树。

在此所用的**语义树**是这样构成的。从始根出发，取H基（理过序的）中第一个原子P_1来定义其树枝。左分枝表示P_1的真值为真(T)，而右分枝表示P_1的真值为假(F)。从而产生两个子节点。从这两个子节点，再取H基中下一个原子P_2按上述方法来定义各自的左右二分枝，从而产生各自的二个子节点。这个过程一直进行到H基中的原子都取遍（而且每个原子只取一次）为止。显然，如果H基是个无限集合的话，其对应的语义树也是无限的。

对于S中子句的某种解释，在语义树中就有一条从始根至某一端节点的路相对应。一棵完整的语义树必含有所有可能的路，它们表示S中子句的所有可能的解释。语义树之名也就因此而得。

例3.17　研究一下子句集$S=\{P(x)\vee Q(y),\sim P(a),\sim Q(b)\}$的语义树。在此情况下，S的H全域是有限的。

$$H(S)=\{a,b\}$$

而且其H基也是有限的，且次序整理为：

$$\{P(a),Q(a),P(b),Q(b)\}$$

其语义树示于图3.2。

从图中可看出由始根至端节点的一条通路就给出了S集的一个解释。例如从始根至端节点（用记号1表示）的通路有$P(a)$，$\sim Q(a)$，$\sim P(b)$，$Q(b)$。这就是提供的一种解释。把这样的通路组成一个集合。

$$M_1=\{P(a),\sim Q(a),\ \sim P(b),Q(b)\}$$

称这样的集合为关于子句集的一个**模式**。

同理，从始根至节点2的另一个模式M_2为$\{\sim P(a),\sim Q(a),\sim P(b),\sim Q(b)\}$。图3.2中就有16个模式。

把某个模式具体地去检验一下原来子句集S中的每个子句。只要有一子句的一个基本例子的真值为假的话，即这个模式不能满足这子句，则称这个模式不满足集合S。

例如，M_1中第一个成员$P(a)$使得原先S集的第二个成员$\sim P(a)$的真值为假。何况，M_1中的$Q(b)$也使得S中的$\sim Q(b)$为假。所以M_1不满足S，而M_2虽然能满足S中的第二、第三个子句（$\sim P(a)$及$\sim Q(b)$）。但它不满足S中的第一个子句$P(x)\vee Q(y)$，因为$P(x)\vee Q(y)$的基本例子$P(a)\vee Q(b)$关于M_2的具体解释为假，故M_2不满足S。同样如果把所有十六个模式都检验一下的话，则可以发现，它们都不满足S。

实际上并不需要一个模式一个模式地检验。例如在该例中，从始根开始的左分枝为$P(a)$，它已经不满足其子句集S中的子句$\sim P(a)$了。可以肯定再从该子节点而往下发展的八个模式

都是不满足 S 的。称这类节点为**失效节点**，用实心园点表示之（如图 3.2 中所示）。

在子句集 S 的语义树中，如果从起始节点至某节点的路所组成的模式，不满子句集S，但从起始节点至其父节点的模式却可满足子句集 S。则称该节点为**失效节点**。

可见，语义树中每个失效节点的后裔节点就没有产生的必要了。换句话说，可以在失效节点处进行修剪。修剪的结果如图 3.3 所示。

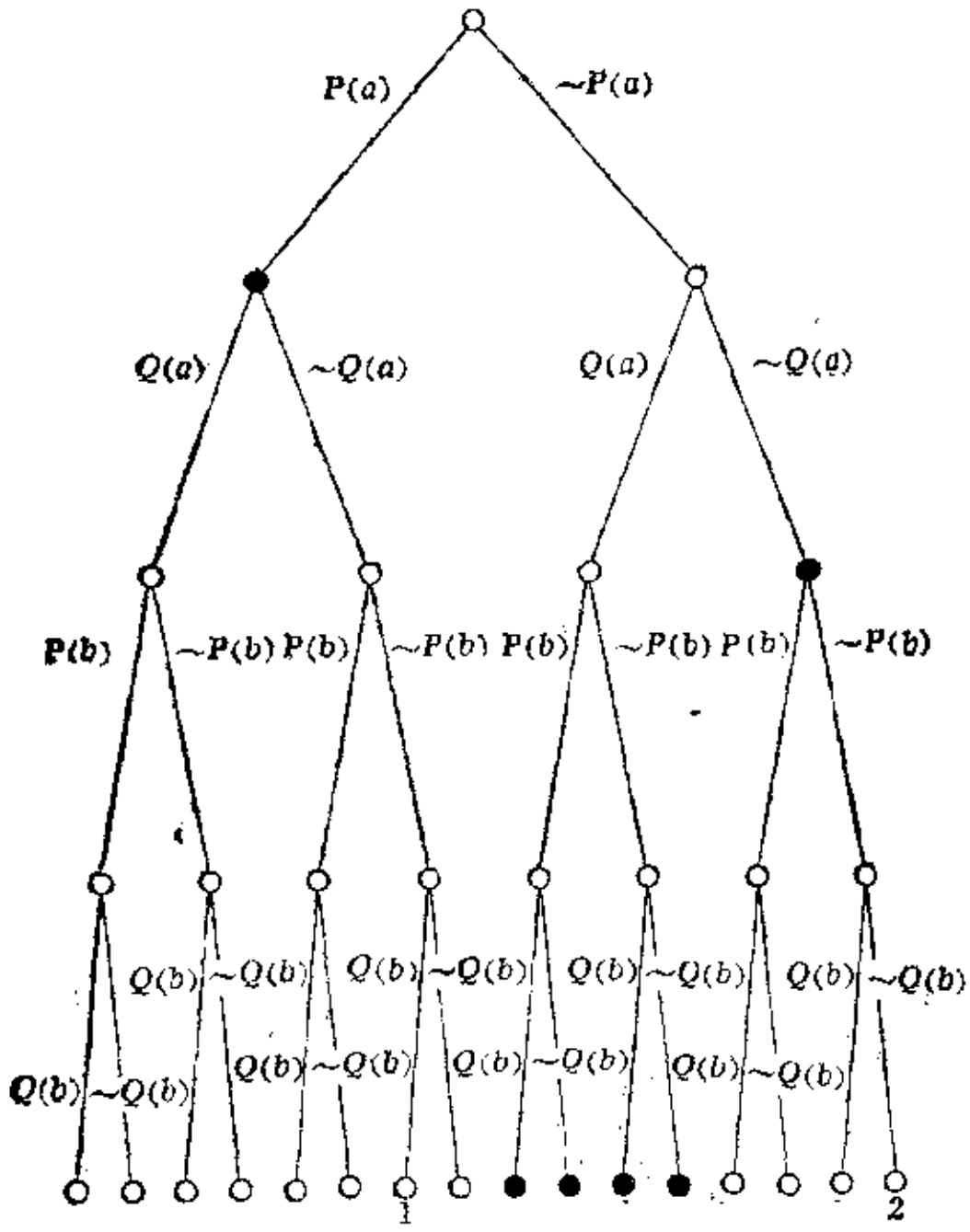


图 3.2 语义树的具体例子

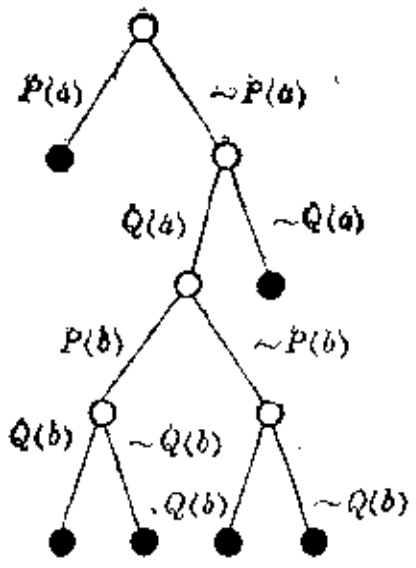


图 3.3 对图 3.2 的修剪结果

如果子句集 S 的语义树的所有路都有失效节点的话，称该语义树为关于 S **封闭**的，或者说被失效节点所封闭的。例如，图 3.3 的语义树是关于 S 封闭的。

封闭语义树一定是有限的，其失效节点的先辈节点是有限个的。

3.2.6 Herbrand定理及实现方法

Herbrand 定理是一个很重要的定理，在自动定理证明中也是最现代证明过程（分解原理）的一个基础。Herbrand 定理有二种叙述形式，现分别介绍如下：

定理 3.4（**Herbrand 定理**，Ⅰ型）子句集 S 是不可满足的充要条件是对应于 S 的每棵完整的语义树，都是一棵有限的封闭语义树。

先证充分性。由于对应于 S 的每棵完整的语义树 T，都有一棵有限的封闭语义树，则 T 的每个由始根至端节点的分枝系列都至少含有一个失效节点。即每个模式都不满足 S，故

S 是不可满足的。

后证必要性。利用反证法。假设对应于 S，如果存在着一棵非封闭的语义树 T，那么 T 中必有一个由始根至端节点的分枝，不含有任何失效节点。即有一个模式能满足 S。换句话说，S 是可满足的子句集。这就导出矛盾。因而 S 是不可满足的话，对应于 S 的每棵完整的语义树，都有一棵有限的封闭语义树。

定理 3.5（**Herbrand 定理，** Ⅱ 型）子句的集合 S 为不可满足的充要条件是有一个 S 子句的基本例子的有限的不可满足集合 S′。

证明必要性。根据定理 3.4，只需对于有限的封闭语义树 T′ 中的每个失效节点，找出其所有对应的子句的基本例子（它们都是不可满足的）。而组成一个有限的集合 S′。这个 S′ 就是本定理中所需的有限的不可满足集。故得证。

证明充分性。设有一个 S 的子句的基本例子的有限的不可满足集合 S′。因为 S 的每个解释 I 都含有 S′ 的一个解释 I′，如果 I′ 不满足 S′，则 I 也必须不满足 S′。然而每个解释 I′ 都不满足 S′。从而 S 的每个解释 I 都不满足 S′。因此 S 的每个解释 I 不满足 S。所以 S 是不可满足的。

例 3.18　欲证子句集 $S=\{\sim P(x)\vee Q(x),\ P(f(y)),\ \sim Q(f(y))\}$ 是不可满足的。根据 Herbrand 定理，可以有二个办法。

第一种办法是可以找到一个在 S 中子句的基本例子的不可满足集 $S'=\{\sim P(f(a))\vee Q(f(a)),\ P(f(a)),\ \sim Q(f(a))\}$。因而 S 是不可满足的。

第二种办法是首先求出其 Herbrand 全域　$H(S)=\{a,f(a),f(f(a)),f(f(f(a)),\cdots\}$，显然 H 基是无限的，但却可得到其封闭语义树，如图 3.4 所示。因而 S 是不可满足的。

将 Herbrand 原理付之实现的方法有许多种，此处介绍一下 Davis 和 Putnam 提出的方法。该方法由如下规则组成（令 S 是子句集）。

(1) **同义反复规则**：因为同义反复对于任何解释都为永真，故从 S 中删去那些同义反复的所有子句，余下的集合 S′ 与原来的集合 S 两者的不可满足性仍然等价。例如，若 $S=\{\sim P(x)\vee P(x),\ Q(f(y)),\ R(g(x))\}$，则删去其中的同义反复 $\sim P(x)\vee P(x)$，而得 $S'=\{Q(f(y)),\ R(g(x))\}$。S 与 S′ 仍具有相同的不可满足性。

(2) **一个基本式规则**：如果在 S 中有某个子句恰恰仅是一个基本式 L 的话，则就从 S 中删去含有 L 的所有子句，从而得到集合 S′。如果 S′ 是空集，则 S 是可满足的，故这时用符号{■}来表示空集 S′。如果 S′ 是非空集合，则再从 S′ 中删去各子句的基本式 ~L，而得到集合 S″。（注意，如果 S′ 中的子句恰恰是仅含有这个基本式 ~L，而没有其他基本式的话，则当 ~L 被删去时，该子句就变成为□。）这么一来，S″ 与 S 的不可满足性是等价的。

例 3.19　设 $S=\{P,P\vee\sim Q,P\vee\sim R\}$。因第一个子句仅有一个基本式 P，故删去所有含有 P 的子句，而得到空集 S′。$S'=\{■\}$。故 S 是可满足的。

例 3.20　设 $S=\{P\vee Q\vee\sim R,\ P\vee\sim Q,\ \sim P,\ R,\ \sim P\vee U,\ U\}$。因第三个子句仅有基本式 ~P，故得 $S'=\{P\vee Q\vee\sim R,\ P\vee\sim Q,\ R,\ U\}$。再从 S′ 中删去基本式 P，而得 $S''=\{Q\vee\sim R,\ \sim Q,\ R,\ U\}$。则只要能验证出 S″ 是不可满足的话，就可肯定 S 也是不可满足的。实际上，对于 S″ 还可关于 ~Q 再使用本规则而得 $(S'')'=\{Q\vee\sim R,\ R,\ U\}$ 及 $(S'')''=\{\sim R,\ R,\ U\}$。

对(S″)″再关于 R 使用本规则，而得((S″)″)′ = {~R, U}，从而最终得到((S″)″)″ = {□, U}。由于有空子句□存在，所以((S″)″)″是不可满足的。因而(S″)″也是不可满足的。可见，S″ 也是不可满足的。从而证得 S 是不可满足的。

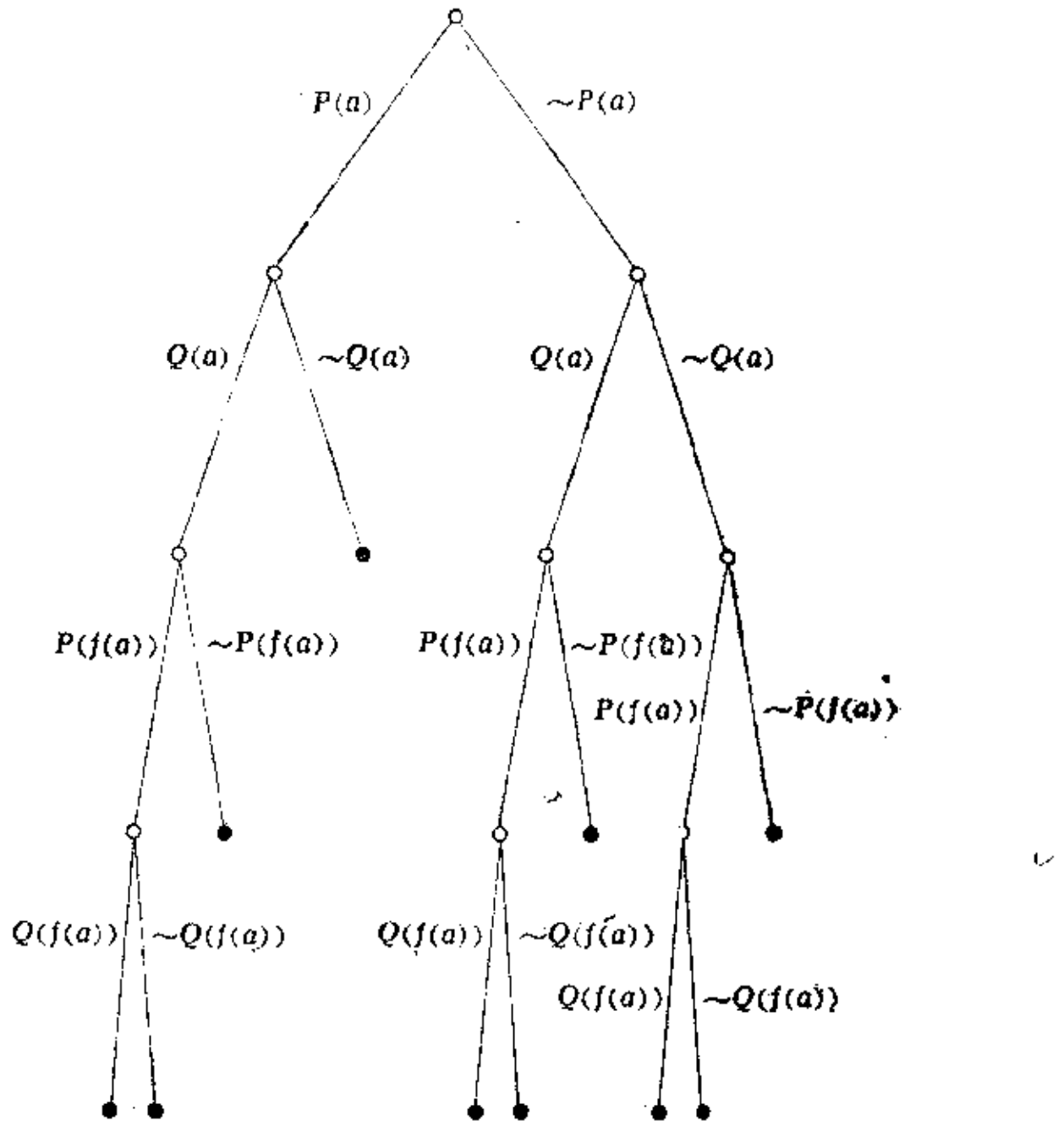


图 3.4　不可满足集合{~P(x) ∨ Q(x)，P(f(y))。~Q(f(y))}的封闭语义树

(3) 纯基本式规则：如果有某个基本式 L 出现在 S 的某个子句中，但其互补基本式~L 却不出现在 S 中的任一子句中，则称该基本式 L 在 S 中是纯的。如果基本式 L 在 S 中是纯的，那么就删去含有 L 的所有子句，而得到集合 S′，则 S′ 与 S 的不可满足性仍然是等价的。注意，如果 S′ 为空集，则 S 为可满足的，故此时用{■}表示空集 S′。

例 3.21　设 S = {P ∨ Q, P ∨ ~Q, ~R ∨ Q, ~R ∨ ~Q}。基本式 P 和~R 在 S 中都是纯的，则先后关于 P, ~R 对 S 进行删除（同时删除也是可以的），则得 S′ = {■}。因此 S 是可以满足的。

(4) 分裂规则：如果集合 S 能够表示成如下形式

$$S = \{A_1 \vee L, \cdots, A_m \vee L, B_1 \vee \sim L, \cdots, B_n \vee \sim L, R_1, \cdots R_k\}$$

其中 A_i, B_i 和 R_i 都是不含有 L 或~L 的基本式的析取式，那么就根据 L 和~L 把它分裂成二个集合 S_1 和 S_2，即 $S_1 = \{A_1, \cdots, A_n, R_1, \cdots, R_k\}$ 和 $S_2 = \{B_1, \cdots, B_n, R_1, \cdots, R_k\}$。则 S 是不可满足的充要条件是 S_1 和 S_2 都是不可满足的。如果把 S，S_1，S_2 都写成为合取范式的话，则

$$S=(A_1\vee L)\wedge\cdots\wedge(A_m\vee L)\wedge(B_1\vee\sim L)\wedge\cdots\wedge(B_n\vee\sim L)\wedge R_1\wedge\cdots\wedge R_k$$

$$S_1\vee S_2=(A_1\wedge\cdots\wedge A_m\wedge R_1\wedge\cdots\wedge R_k)\vee(B_1\wedge\cdots\wedge B_n\wedge R_1\wedge\cdots\wedge R_k)$$

两者的不可满足性是等价的。

例 3.22　设 $S=\{P\vee\sim Q,\sim P\vee Q,Q\vee\sim R,\sim Q\vee\sim R\}$，关于 P 来分裂而得

$$S_1\vee S_2=((\sim Q)\wedge(Q\vee\sim R)\wedge((\sim Q)\vee(\sim R)))\vee(Q\wedge(Q\vee\sim R)\wedge((\sim Q)\vee(\sim R)))$$

只需证出 $S_1\vee S_2$ 的不可满足性就可得出 S 的不可满足性。

注意，在解题过程中，可以根据需要采用以上四个规则。如上例，在采用分裂规则之后，还可以对 S_1、S_2 分别关于~Q、Q 采用一个基本式规则而分别得 $S_1''=\{\sim R\}$，$S_2''=\{\sim R\}$。再对 S_1''、S_2'' 关于~R 采用一个基本式规则而分别得到■和■。由于两者都不是不可满足的，故 S 是可满足的。

3.3 分 解 原 理

分解原理的基本思想是：检验一下 S 是否含有空子句□。如果 S 含有□，则 S 是不可满足的。如果 S 不含有□，那么接着检验一下能否从 S 导出□。在这节中首先介绍命题演算的分解原理，而后介绍谓词演算的分解原理。

3.3.1 推 理 节 点

从语义树的讨论中可知，对于不可满足的子句集 S 的语义树一定是封闭的。从这可引伸出预解式的概念。

例 3.23　有不可满足子句集 $S=\{\sim P(x)\vee Q(x),\ P(f(y)),\ \sim Q(f(y))\}$，其 H 全域是 $H(S)=\{a,f(a),f(f(a)),f(f(f(a)),\cdots\}$。虽然其 H 基是无限的，但由于 S 是不可满足的，其语义树是封闭的。如图 3.5 所示。

在图 3.5 中，子句$\sim P(x)\vee Q(x)$与$\sim Q(f(y))$分别在节点 1 和 2 失效。称节点 1 和 2 为失效节点。但在节点 3 却不是失效节点。在语义树中，节点本身不是失效节点，但其两个子节点都是失效节点，称该节点为**推理节点**。节点 3 就是一个推理节点。从推理节点可以觉察出来，将会有一个新子句（当然不在子句集 S 中），它自己已在该推理节点处失效或者在该推理节点的先辈节点处就失效。例如，若有一个解释 I 满足分别在节点 1 和 2 失效的子句$\sim P(x)\vee Q(x)$和$\sim Q(f(y))$，x 用 $f(y)$置换之后，I 当然应满足$\sim P(f(y))\vee Q(f(y))$和$\sim Q(f(y))$。而$\sim P(f(y))\vee Q(f(y))$中的$Q(f(y))$与$\sim Q(f(y))$为互补基本式，不可能同时满足。因而 I 应满足$\sim P(f(y))$。这个$\sim P(f(y))$就是从在失效节点 1、2 处失效的子句$\sim P(x)\vee Q(x)$与$\sim Q(f(y))$推出来的新子句。称该新子句为原来二个子句的**预解式**。预解式$\sim P(f(y))$不在子句集 S 中，但是在推理节点 3 之处失效。

值得注意的是：关于非空子句的不可满足集合 S 之任一封闭语义树，至少必须有一个推理节点。否则，每一节点将至少有一个非失效的后裔节点，就和语义树是封闭的假设有矛盾了。

这么一来，就可从 S 的封闭语义树的推理节点出发，从 S 中的两个子句推出预解式 C。C 与 S 构成一个新子句集 $S_1'=\{C\}\cup S$。那么关于 S 的语义树也是关于 S_1' 的语义树。但有一点区别，对于 S_1' 而言，该推理节点或其先辈节点已是失效节点了。换句话说，S_1' 的封闭

语义树与原来 S 的封闭语义树相比较一下，关于其近树根处的非失效节点数目方面，前者比后者少，至少少一个非失效节点。而 S_1' 的封闭语义树又至少有一个推理节点，又可推导出新的预解式，再构成一个更新的子句集 S_2'。如此重复以上过程。封闭语义树的近根处的非失效节点数目就一次次减少。经过有限步之后，树根也是失效节点了。因为树根仅是对于空子句□失效，故最终得到的子句集 S_n' 必然含有空子句□。换句话说，如果子句集 S 是不可满足的话，则从 S 必可推出空子句□。

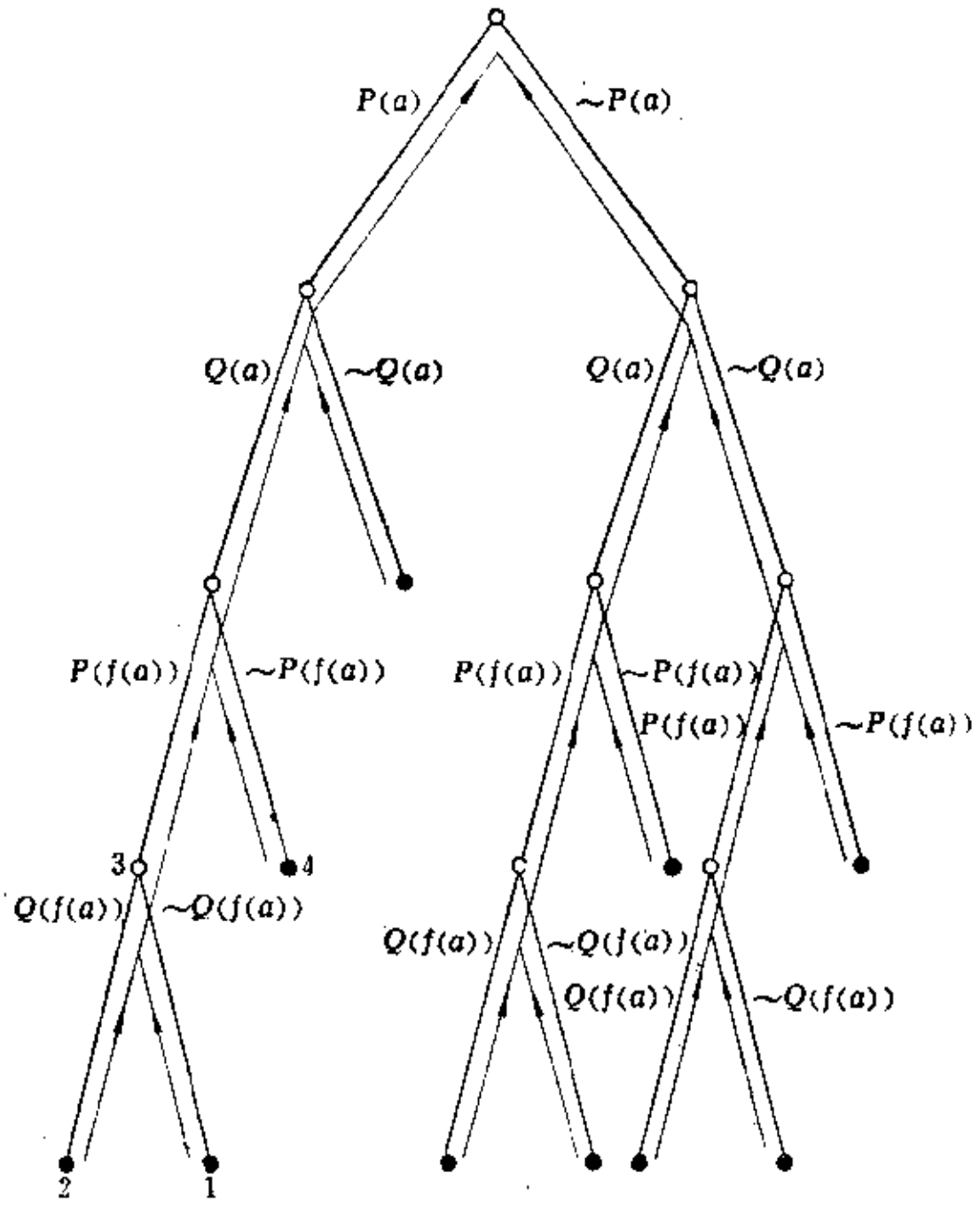


图 3.5　树根失效的推理方法

3.3.2　命题演算的分解原理

分解原理　对于任意二个子句 C_1 和 C_2，如果在 C_1 中有一个基本式 L_1，而在 S_2 中也有与 L_1 成互补的基本式 L_2，则分别从 C_1 和 C_2 中删去 L_1 和 L_2，而且将其子句剩余部分构成析取式。这个新构成的子句就是 C_1 和 C_2 的**预解式**。

可用表达式来表示分解原理。令 $C_1 = L \vee C_1'$，$C_2 = (\sim L) \vee C_2'$，其中 L 为基本式，C_1'、C_2' 为基本式的析取式，$C_1 \in S$，$C_2 \in S$，则 C_1 和 C_2 的预解式 $C = C_1' \vee C_2'$。当 $C_1' = C_2' = \square$ 时，$C = \square$。注意，如果 L 及 $\sim L$ 不分别同时存在于 C_1 及 C_2 中的话，则预解式 C 不存在，但不能得出结论 $C = \square$。

定理 3.6　二个子句 C_1 和 C_2 的预解式 C 是 C_1 和 C_2 的逻辑推论。

证：假设 $C_1 = L \vee C_1'$ 和 $C_2 = (\sim L) \vee C_2'$ 关于解释 I 为真，则需要证明 $C = C_1' \vee C_2'$ 关于解释 I 也为真。

因为关于解释 I，L 和～L 两者中必有一个为假。设L为假。则 C_1' 必为真（否则，C_1 为假，这与前面的假设有矛盾）。

同理，如果关于解释 I，～L 为假，则 C_2' 必为真。因而 $C = C_1' \vee C_2'$ 关于解释 I 为真。

可见，C 是 C_1 和 C_2 的逻辑推论。证毕。

集合 $S = \{C_1, C_2, \cdots\}$ 比集合 $S_1' = \{C, C_1, C_2, \cdots\}$少一个成员 C。如果 S_1' 是不可满足的话，则对于不满足 S_1' 的任一解释I，有二种可能：一，如 I 使 C 为真，则 I 必使 S_1' 中除 C 外的某一子句为假，即 I 必使 S 中的某一子句为假。因而，S 也是不可满足的。二，如果 I 使 C 为假，则根据定理 3.6，I 或者使 C_1 为假，或者使 C_2 为假，二者必居其一。因而，S 也是不可满足的。

显然，S 为不可满足集合的话，S_1' 必为不可满足。因而，S 与 S_1' 的不可满足性是等价的。由 S_1' 应用分解原理而得到的集合 S_2' 也具有等价的不可满足性。为此可得到子句集 $S_1', S_2', \cdots, S_n'$。若 S_n' 含有空子句（即预解式是空子句□），就可肯定 S 是不可满足的。

例 3.24　试证集合 $S = \{P \vee Q, (\sim P) \vee Q, P \vee \sim Q, \sim P \vee \sim Q\}$的不可满足性。

证法1.　画出语义树，检验是否为封闭语义树。

S 的语义树如图 3.6 所示。S 中子句 $P \vee Q, (\sim P) \vee Q, P \vee \sim Q, \sim P \vee \sim Q$ 分别在节点 4，3，2，1 处失效。故该语义树是封闭的。证毕。

还可以从图 3.6 中的推理节点 5 与 6 出发，推出树根也为失效节点。实际上，新子句 Q 与～Q 分别在节点 6 与 5 处失效。树根又成为推理节点。因而可推出树根为失效节点。

证法2.　导出空子句□。

从 S 中 $P \vee Q$ 与 $\sim P \vee Q$ 推出预解式 Q 而得 $S_1' = \{P \vee Q, \sim P \vee Q, Q, P \vee \sim Q, \sim P \vee \sim Q\}$。

从 S_1' 中 $P \vee \sim Q$ 与 $\sim P \vee \sim Q$ 推出预解式 $\sim Q$ 而得 $S_2' = \{P \vee Q, \sim P \vee Q, Q, P \vee (\sim Q), \sim P \vee Q, \sim Q\}$。

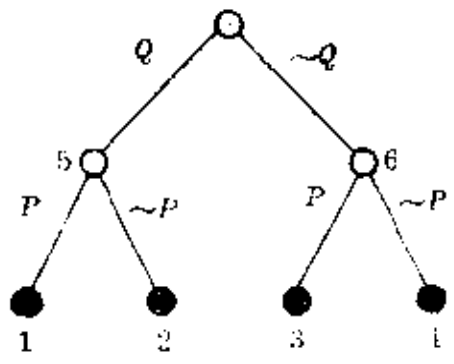


图 3.6　$\{P \vee Q, \sim P \vee Q, P \vee \sim Q, \sim P \vee \sim Q\}$的语义树

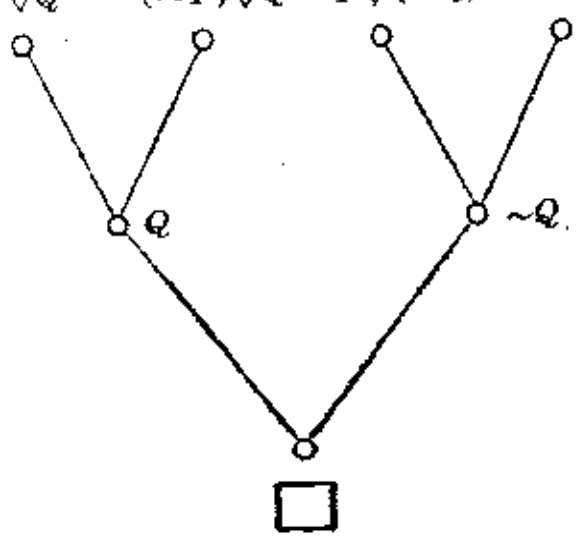


图 3.7　演绎树

从 S_2' 中 Q 与～Q 推出预解式□。

由于推出□，故 S 为不可满足的。

证法3。 作出演绎树。

证法 2 用树形来表示，则如图 3.7 所示。它是一棵内向树。称该树为**演绎树**。也称为**反驳图**。

三个方法是一致的。演绎树实际上也是从语义树中的推理节点出发推出树根为失效节点的过程。

3.3.3 谓词演算的分解原理

从上节中可以清楚地看出：应用分解原理过程中很重要的关键之一是寻找二个子句中成为互补的基本式。在命题演算中，子句是不含变元的，那是很简单的。然而，如果子句含有变元，那就较复杂了。例如，若 $S=\{P(y)\vee Q(y),\ (\sim P(f(x)))\vee R(x)\}$。该两子句中并没有互补的基本式，因谓词 P( )中的项不相同。但是如果在第一个子句中用 $f(a)$置换 y，在第二个子句中用 a 置换 x，则得 $S'=\{P(f(a))\vee Q(f(a)),\ (\sim P(f(a)))\vee R(a)\}$。这么一来，就可求出其预解式为 $Q(f(a))\vee R(a)$ 了。但是如果在第一子句中用 $f(x)$ 置换 y 的话，则得 $S^*=\{P(f(x))\vee Q(f(x)),\ (\sim P(f(x)))\vee R(x)\}$。这样就得另一预解式$Q(f(x))\vee R(x)$。它的特例就是前者的预解式。由于采用不同的置换，会得出不同的预解式。如果所有预解式都是其中某个预解式的特例的话，则称该预解式为**最普通预解式**。

因此，在谓词演算中应用分解原理，必须着重解决如下几个问题：

(1) 子句集 S 中两子句之间基本式的文字符号虽然相同（如上例中 $P(y)$与 $P(f(x))$ 的文字 P( )是相同的），但其中项的具体内容（此例为y，$f(x)$）却不同。因此，必须寻找出它们之间不一致的项。

(2) 如何置换才能使它们一致起来。

(3) 如何求出预解式，从而推出空子句。

这些问题将一个个解决。

不一致集合 将需要寻找其不一致所在的表达式组成一个非空集合 $W=\{E_1,\cdots,E_k\}$。W的**不一致集合** D 可以这样求得：

(1) 对 W 中的 E_1，E_2，…，E_k 都自左至右依序地检查其符号：(a) 如果所有表达式 E_1，…，E_k 在某位置的符号都相同，则接下去检查下一个符号（转到(1)）；(b)如果 E_1，…，E_k 在某个位置上的符号并不完全相同，则说明有不一致符号存在。

(2) 如一发现有不一致的符号存在，则对 E_1，…，E_k 分别再加以处理：(a) 该不一致符号取出来作为子表达式中的第一个符号；(b)下一个符号若不是左括弧“（”，则该子表达式就到该符号为止；下一个符号若是左括弧“（”，则一方面对括弧符号进行成对性计数（左括弧加 1，右括弧减 1），另一方面同时往下查，直到括弧计数器为 0 时，那么该子表达式就到此右括弧为止。

(3) 将 E_1，…，E_k 各自的如上取出的子表达式分别作为集合的成员，从而求出 W的不一致集合 D。

例3.25 求$W=\{P(x,\ f(y,z)),\ P(x,a),\ P(x,g(h(k(x))))\}$的不一致集合 D。

对W中三个表达式依序进行判别。各表达式的第一至第四个符号都是相同的，依序都是

^{D}、(、x、$,$。而第五个符号却不相同。那么，对第一个表达式进行判别、取出其子表达式为$f(y,z)$，第二、第三个表达式的子表达式分别为a，$g(h(k(x)))$。因此，W的不一致集合为 $D=\{f(y,z),\ a,\ g(h(k(x)))\}$。

置换 对于表达式集 $E=\{E^1,E_2\cdots,E_k\}$，，若以项$t_i(i=1,\cdots,n)$置换在E中出现的变元x_i，则用有限集合 $\theta=\{t_1/x_1,\ t_2/x_2,\cdots,t_n/x_n\}$ 表示置换。注意，θ中的每个x_i都是变元，而且$x_i\neq x_j(i\neq j)$，每个t_i都是项，$t_i\neq x_i$。若$t_i=x_i$，则应从θ中删除该成员t_i/x_i。置换的集合用希腊字母表示。如果θ是一个空集合，则称θ为**空置换**。用ε表示空置换。

例3.26 下列的集合是置换：$\theta=\{f(y)/x,\ x/z\}$，$Y=\{a/x,\ f(g(b))/y,\ f(z)/z\}$。但是，$\{f(a)/x,\ g(y)/x,\ g(z)/z\}$不是置换，因为有两个变量同为$x$，搞不清应以$f(a)$置换$x$，还是以$g(y)$置换$x$。$\{g(y))/x,\ {}^{b}/_{a}\}$也不是置换，因为有一个$x_i=a$，不是变元。

置换 $\theta=\{t_1/x,\ \cdots,\ t_n/x_n\}$对于表达式E进行置换而得的新表达式，记为Eθ，其规则如下：

(1) 凡是每个变元$x_i(i=1,\ \cdots,\ n)$在E中出现的每个地方，都用相应的t_i置换之。

(2) 每个变元x_i的置换必须是同时一次进行的。

例3.27 令置换 $\theta=\{x/y,\ a/x\}$，$E=P(x)\vee Q(y)\vee R(x)\vee W(z)$。则 $E\theta=P(a)\vee Q(x)\vee R(a)\vee W(z)$。注意：$E\theta\neq P(a)\vee Q(y)\vee R(a)\vee W(z)$（因为θ中的变元$y$在E中出现的地方Q(　)，没有被置换）；$E\theta\neq P(a)\vee Q(x)\vee R(x)\vee W(z)$。因为θ中的变元$x$在E中出现的两个地方 P(　)与R(　)，有一处 R(　)没有被置换)；虽然 $E\theta=P(a)\vee Q(x)\vee R(a)\vee W(z)$，其中有变元$x$，但不能再进行一次置换，将$x$用$a$置换而得 $P(a)\vee Q(a)\vee R(a)\vee W(z)$。$E\theta\neq P(a)\vee Q(a)\vee R(a)\vee W(z)$；对E关于θ中某个成员（如$x/y$）先进行置换而得 $P(x)\vee Q(x)\vee R(x)\vee W(z)$，而后关于θ中另一个成员（如$a/x$）再进行置换，最终得到$P(a)\vee Q(a)\vee R(a)\vee W(z)$，这是不允许的，因为这个置换不是同时进行的。

Eθ 称为E的**特例**

置换的合成 设有二个置换 $\theta=\{t_1/x_1,\ \cdots,\ t_n/x_n\}$和 $\lambda=\{u_1/y_1,\ \cdots,\ u_m/y_m\}$。表达式E先关于θ进行置换之后而得到新表达式Eθ，再对Eθ关于λ进行置换之后得到 $(E\theta)\lambda$。θ与λ的合成用 $\theta\circ\lambda$ 表示，即 $(E\theta)\lambda=E(\theta\circ\lambda)$。$\theta\circ\lambda$ 仍是一个置换。用如下有限集合表示。

$$\theta\circ\lambda=\{t_1\lambda/x_1,\ \cdots,\ t_n\lambda/x_n,\ u_1/y_1,\ \cdots,\ u_m/y_m\}$$

注意：(1) $\theta\circ\lambda$中$t_i\lambda$表示对θ中的项t_i关于λ进行置换而得的新项。

(2) 如果在 $\theta\circ\lambda$ 中出现 $t_i\lambda=x_i(i=1,\ \cdots,\ n)$时，则根据置换的定义，应从 $\theta\circ\lambda$ 中全部删去这类成员 $t_i\lambda/x_i$。

(3) 在 $\theta\circ\lambda$ 中如果出现$y_i=x_k$，$(k=1,\ \cdots,\ n)$则应从 $\theta\circ\lambda$ 中全部删去这类成员 u_i/y_i。理由是：如果θ的t_j中不含有变元x_k(即y_i)的话，则E关于θ置换之后就再也不可能出现变元y_i了。$\theta\circ\lambda$中的u_i/y_i已成为多余的了；如果θ的t_j中含有变元y_i(即x_k)的话，Eθ再关于λ置换之后，t_j中的y_i（即x_k）已由u_i所置换，而在$\theta\circ\lambda$中$t_j\lambda$也含有“t_j中的y_i由u_i置换”之意，故u_i/y_i也成为多余的了。

(4) 不能由于(3)的理由而得出结论：“因为在 $\theta\circ\lambda$ 中有 $t_j\lambda$ 的保证，即使不出现 $y_i=$

x_k 的条件，也可将 u_i/y_i 从 θ ∘ λ 中删去。”事实上如果 E 中有变元 y_i，关于 θ 置换之后变元 y_i 仍然原封不动保留着（因 $y_i \neq x_k$）。而 Eθ 关于 λ 置换之后，y_i 已被 u_i 所置换。但如果 θ ∘ λ 中没有 u_i/y_i（因被删除了），又没有一个 x_k 与 y_i 相同，那么 E 关于 θ ∘ λ 置换之后，变元 y_i 仍然没有被 u_i 所置换。故在 $y_i \neq x_k$ 时 θ ∘ λ 中 u_i/y_i 不可删除。

(5) 置换的结合律成立，即 $(\theta \circ \lambda) \circ \mu = \theta \circ (\lambda \circ \mu)$；空置换具有左右同一性，即 $\varepsilon \circ \theta = \theta \circ \varepsilon = \theta$。但置换的交换律不成立，即 $\theta \circ \lambda \neq \lambda \circ \theta$。

例3.28 令 $\theta = \{t_1/x_1,\ t_2/x_2\} = \{f(y)/x,\ z/y\}$。

$$\lambda = \{u_1/y_1,\ u_2/y_2,\ u_3/y_3\} = \{a/x,\ b/y,\ y/z\}$$

则 $\theta \circ \lambda = \{t_1\lambda/x_1,\ t_2\lambda_2/x_2,\ u_1/y_1,\ u_2/y_2,\ u_3/y_3\}$

$$= \{f(b)/x,\ y/y,\ a/x,\ b/y,\ y/z.\}$$

但其中 y/y 应删除(因为 $t_2\lambda = x_2 = y$)；a/x、b/y 也应删除(因 $y_1 = x_1 = x$，$y_2 = x_2 = y$)。故最终应得

$$\theta \circ \lambda = \{f(b)/x,\ y/z\}$$

置换的目的是为了使 S 的子句之间能有较多的构成互补的基本式，从而可顺利地应用分解原理。因此必须寻找能使若干个表达式一致的置换。

一致化 对于表达式的集合 $\{E_1, E_2, \cdots, E_k\}$，如果有一个置换 θ 存在，使得 $E_1\theta = E_2\theta = \cdots = E_k\theta$，则称 θ 为关于集合 $\{E_1,\ E_2,\ \cdots,\ E_k\}$ 的**一致置换**，称集合 $\{E_1,\ E_2,\ \cdots,\ E_k\}$ 为**能一致化**。

例如，集合 $\{P(a, y),\ P(x, f(b))\}$ 是能一致化的。因为置换 $\theta = \{a/x,\ f(b)/y\}$ 是关于该集合的一致置换。

某集合的一致置换也不是唯一的，可能有许多个。

如果对于表达式集合 $\{E_1,\ \cdots,\ E_k\}$ 的每个一致置换 θ 而言，都可找到一个置换 λ，使得 $\theta = \sigma \circ \lambda$，则称 σ 为关于该集合的**最普通一致置换**。我们所希望的就是寻找最普通一致置换。其算法如后。

一致化算法

步骤 1. 设置 $k = 0$，$W_k = W$ 且 $\sigma_k = \varepsilon$。

步骤 2. 如 W_k 是单个的，则停止；σ_k 就是 W 的一个最普通一致置换。否则，就求出 W_k 的不一致集合 D_k。

步骤 3. 如果 D_k 中有成员 v_k 和 t_k 存在，使得 v_k 是一个不在 t_k 中出现的变量，则转至步骤 4。否则，停止。W是不能一致化的。

步骤 4. $\sigma_{k+1} \Leftarrow \sigma_k \circ \{t_k/v_k\}$，$W_{k+1} \Leftarrow W_k\{t_k/v_k\}$。（注意，$W_{k+1} = W\sigma_{k+1}$。）

步骤 5. $k \leftarrow k + 1$ 且转至步骤 2。

其流程图如图 3.38 所示。

该一致化算法是可行的。对于任一有限的非空集合表达式 D 而言，该算法总是可终结的。如果能无限止地运行下去的话，则会产生一个无限序列 $W\sigma_0$，$W\sigma_1$，$W\sigma_2$，…。而且每一个集合 $W\sigma_{k+1}$ 总比其前一个集合 $W\sigma_k$ 少含有一个变量（譬如，$W\sigma_k$ 含有 v_k，但 $W\sigma_{k+1}$ 却没有）。这是不可能的。因为 $W\sigma_0$（即W）所具有的不同变量仅有有限个。而且还可证得：只要W是能一致化的，则一致化算法一定在步骤 2 终结，而且最终的 σ_k 就是W的

一个最普通一致置换。

在一个子句 C 中有时有二个以上的同符号的基本式。如 $C=P(x)\vee P(f(y))\vee\sim Q(x)$ 中第一、第二个基本式具有同符号 P( )。如果子句 C 的二个以上同符号的基本式有一个最普通一致置换 σ，则 Cσ 称为 C 的**因子句**。如果 Cσ 是单元子句（即只有一个基本式），则称之为 C 的**单元因子句**。

例 3.29　$C=P(x)\vee P(f(y))\vee\sim Q(x)$ 中的二个基本式 $P(x)$ 与 $P(f(y))$ 有一个最普通一致置换 $\sigma=\{f(y)/x\}$。因而，$C\sigma=P(f(y))\vee\sim Q(f(y))$ 是 C 的因子句。

分解原理　令 C_1 和 C_2 为不具有完全相同变元的两个子句。令 L_1 和 L_2 分别是 C_1 和 C_2 中的二个基本式。如果 L_1 和 $\sim L_2$ 有一个最普通一致置换 σ，则称子句

$$(C_1\sigma-L_1\sigma)\cup(C_2\sigma-L_2\sigma)$$

为 C_1 和 C_2 的二元预解式。称 C_1 和 C_2 为其二元预解式的**先辈子句**。基本式 L_1 和 L_2 称为**分解所依据的基本式**。

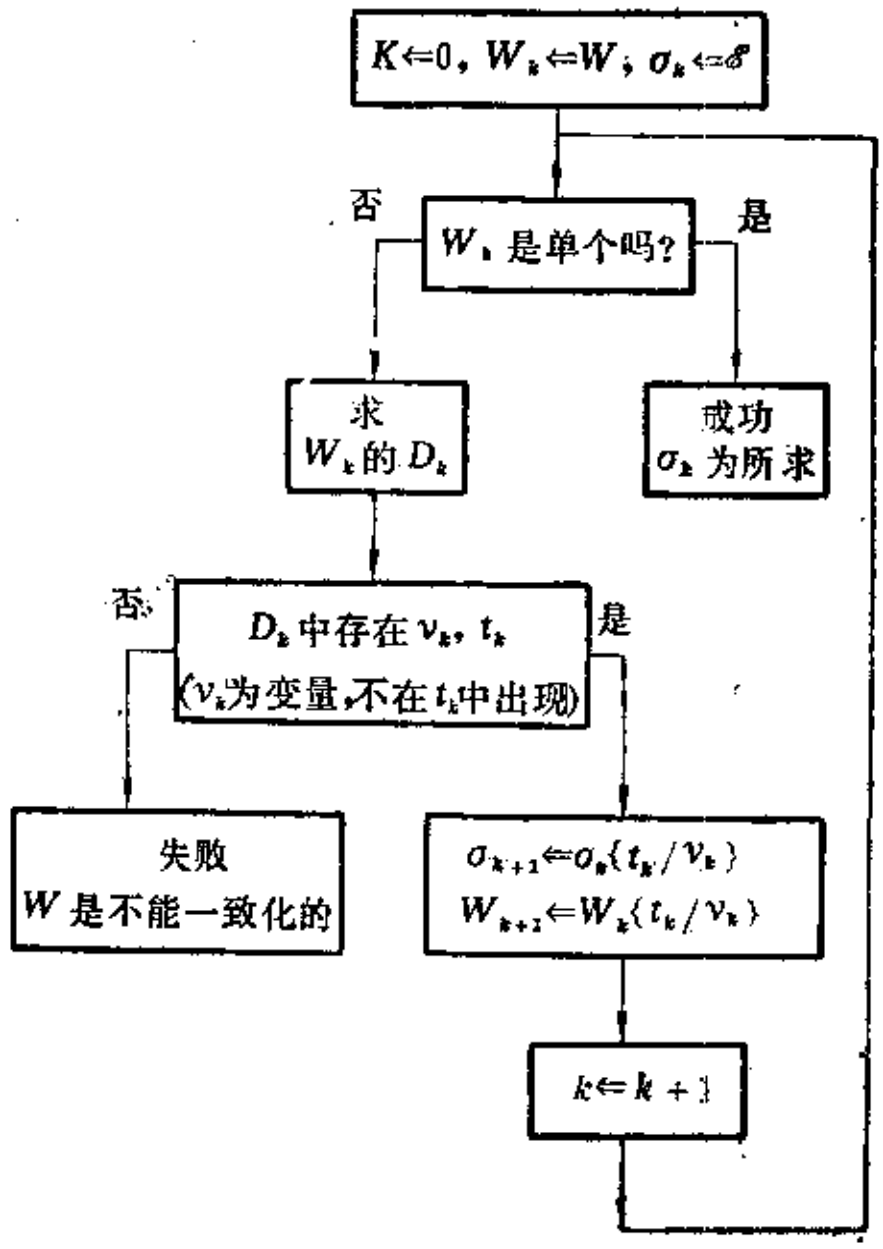


图 3.8　一致化算法的流程图

例 3.30　求 $C_1=P(x)\vee Q(x)$，$C_2=\sim P(a)\vee R(x)$ 的预解式。

选取 $L_1=P(x)$，$L_2=\sim P(a)$。由于 $\sim L_2=P(a)$，所以 L_1 和 $\sim L_2$ 有最普通一致置换 $\sigma=\{a/x\}$。因此

$$\begin{aligned}&(C_1\sigma-L_1\sigma)\cup(C_2\sigma-L_2\sigma)\\&=(\{P(a),Q(a)\}-\{P(a)\})\cup(\{\sim P(a),R(a)\}-\{\sim P(a)\})\\&=\{Q(a)\}\cup\{R(a)\}\\&=\{Q(a),R(a)\}\\&=Q(a)\vee R(a)\end{aligned}$$

$Q(a)\vee R(a)$ 是 C_1 和 C_2 的二元预解式。C_1 和 C_2 是 $Q(a)\vee R(a)$ 的先辈子句。$P(x)$ 和 $\sim P(a)$ 是分解所依据的基本式。

下列二元预解式之一都是（先辈）子句 C_1 和 C_2 的预解式：

(1) C_1 和 C_2 的二元预解式

(2) C_2 的因子句和 C_1 两者之二元预解式。

(3) C_1 的因子句和 C_2 两者之二元预解式。

(4) C_1 的因子句和 C_2 的因子句两者之二元预解式。

以上四种二元预解式是不难理解的。就以第(4)种二元预解式为例加以举例说明。

例 3.31　求 $C_1=P(x)\vee P(f(y))\vee Q(x)$，$C_2=\sim P(u)\vee\sim P(f(w))\vee R(u)$ 的二元预解

式。

方法 1　对 C_1，C_2 先求它们的因子句，而后求两因子句的二元预解式。

C_1 的 σ_1 为 $\{f(y)/x\}$，$C_1\sigma_1 = P(f(y)) \vee Q(f(y))$；

C_2 的 σ_2 为 $\{f(w)/u\}$，$C_2\sigma_2 = \sim P(f(w)) \vee R(f(w))$

关于 $C_1\sigma_1$ 和 $C_2\sigma_2$ 两者之 σ_3 为 $\{y/w\}$，

$(C_1\sigma_1)\sigma_3 = P(f(y)) \vee Q(f(y))$，　$(C_2\sigma_2)\sigma_3 = \sim P(f(y)) \vee R(f(y))$

那么 $(C_1\sigma_1)\sigma_3$ 和 $(C_2\sigma_2)\sigma_3$ 两者之二元预解式为 $Q(f(y)) \vee R(f(y))$，即 C_1 和 C_2 的二元预解式为 $Q(f(y)) \vee R(f(y))$。

方法 2　直接求 C_1，C_2 的二元预解式。

C_1 和 C_2 的 σ 为 $\{f(y)/x,\ f(y)/u,\ y/w\}$；

$C_1\sigma = P(f(y)) \vee Q(f(y))$，　$C_2\sigma = \sim P(f(y)) \vee R(f(y))$

故 C_1 和 C_2 的二元预解式为 $Q(f(y)) \vee R(f(y))$

可见两者方法是一致的。而且 $\sigma = (\sigma_1 \circ \sigma_2) \circ \sigma_3$。当 σ_1，σ_2 分别为 $\sigma_1 = \sigma_2 = \varepsilon$，$\sigma_1 = \varepsilon$，$\sigma_2 = \varepsilon$ 等三种情况时，则第(4)种二元预解式就分别变成第(1)、(2)、(3)种了。

利用分解原理来解决实际问题，一般有如下步骤。

(1) 将与欲证明的问题有关的公理或已证明过的定理等，写成谓词演算有关表达式（命题可看成是谓词的特例）。

(2) 将欲证明的问题写成谓词演算有关的表达式。注意，我们利用反证法，设其结论为不正确，而后导出矛盾。因此，欲证的结论应取否定形式。

(3) 变成 Skolem 标准形，组成子句集 S（欲证该子句集为不满足集合）。

(4) 利用分解原理（即求出不一致集合，利用置换，使之一致化，而求出预解式）导出空子句，从而证得子句集 S 为不满足集合。

例 3.32　试证梯形的对角线与上下底相交的内错角相等。参看图 3.9。

证：令 $T(x,y,u,v)$ 表示具有左上顶点 x，右上顶点 y，右下顶点 u 和左下顶点 v 的一个梯形 $xyuv$；令 $P(x,y,u,v)$ 表示线段 xy 平行线段 uv；令 $E(x,y,z,u,v,w)$ 表示 $\angle xyz$ 等于 $\angle uvw$。

则有如下已知条件：

A_1：$(\forall x)(\forall y)(\forall u)(\forall v)(T(x,y,u,v) \rightarrow P(x,y,u,v))$（根据梯形定义）

A_2：$(\forall x)(\forall y)(\forall u)(\forall v)(P(x,y,u,v) \rightarrow E(x,y,v,u,v,y))$（平行线的内错角相等）

A_3：$T(a,b,c,d)$（根据图 3.9）

从这些已知条件，欲证结论 $E(a,b,d,c,d,b)$ 为真。因而

$\sim G$：$\sim E(a,b,d,c,d,b)$

故欲证：$A_1 \wedge A_2 \wedge A_3 \wedge \sim G$ 永假公式

将它变换成 S 标准形之后而构成如下子句集 S

$S = \{\sim T(x,y,u,v) \vee P(x,y,u,v)$，

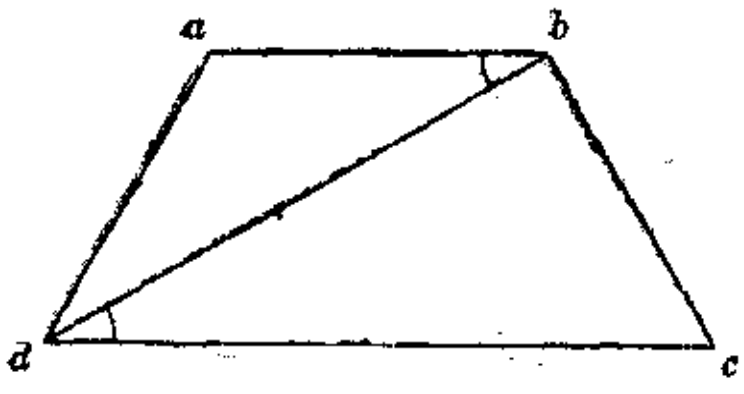


图 3.9　梯形 *abcd*

$$\sim P(x,y,u,v) \vee E(x,y,v,u,v,y),$$
$$T(a,b,c,d), \sim E(a,b,d,c,d,b)\}$$

可见，S 是具有四个子句的一个集合。现在就利用分解原理证明 S 为不可满足的。推导过程如下：

(1) $\sim T(x,y,u,v) \vee P(x,y,u,v)$　　(S 的第一个子句)

(2) $\sim P(x,y,u,v) \vee E(x,y,v,u,v,y)$　　(S 的第二个子句)

(3) $T(a,b,c,d)$　　(S 的第三个子句)

(4) $\sim E(a,b,d,c,d,b)$　　(S 的第四个子句)

(5) $\sim P(a,b,c,d)$　　((2)和(4)的预解式)

(6) $\sim T(a,b,c,d)$　　((1)和(5)的预解式)

(7) □　　((3)和(6)的预解式)

由于推出□，故 S 为不可满足的集合，因 而 $\sim E(a,b,d,c,d,b)$为假。即 $E(a,b,d,c,d,b)$为真。$\angle abd = \angle cdb$ 的定理得到证明。

将上述推导过程用演绎树来表示。如图 3.10 所示。

值得指出的是：以上的推导过程与文字叙述是相一致的。若用文字来叙述(5)、(6)、(7)式时，就可清楚地看出这点。

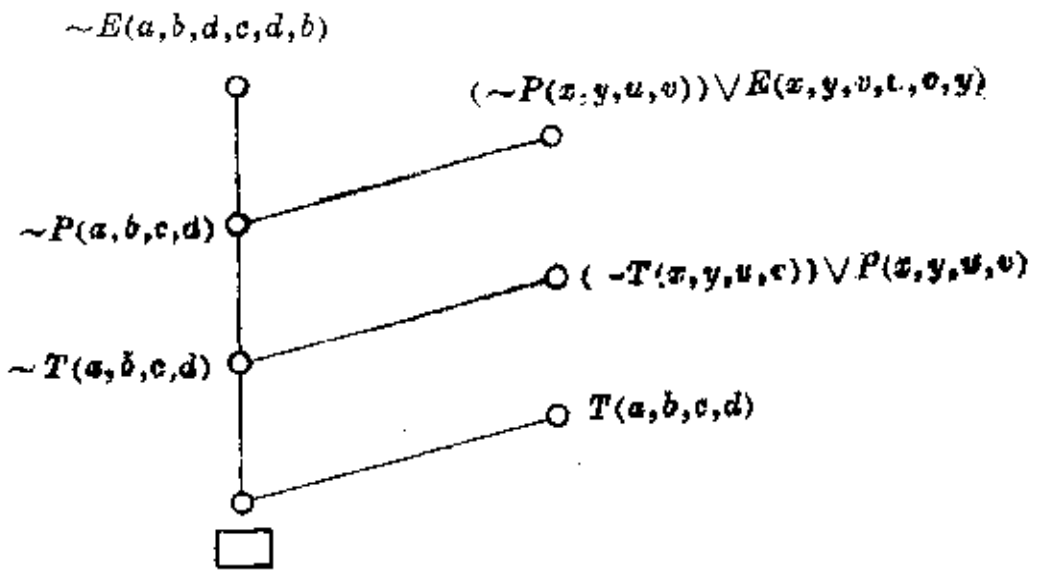


图 3.10　演绎树的例子

(5)式：由于如果线段 xy 平行于线段 uv，则内错角必相等。$\angle xyv = \angle uvy$。但是却假设两夹角不等，$\angle abd \neq \angle cdb$。故线段 ab 必不平行于线段 cd。

(6)式：由于如果 $xyuv$ 是梯形，则上下底必平行，$xy \parallel uv$。但已证得 ab 不平行 cd，$ab \nparallel cd$。故 $abcd$ 不是梯形。

(7)式：已推出的结论“$abcd$ 不是梯形”是与原来的前提“$abcd$ 是梯形”发生矛盾的。

3.3.4　删除策略

由计算机自动地求预解式，并不是每求出一个预解式对解问题都是很有成效的。

例如上例中，将 S 中的子句做为原始集合 S^0 的成员。再由 S^0 中每两个成员求出可能的预解式，而组成集合 S^1。S^1 的成员数不一定是 S^0 中成员的组合数。因有时某两个子句间没有预解式存在；而有时某两个子句间却有两个以上的预解式。(如 P∨Q 与~P∨~Q 就有两个预解式 Q∨~Q 及 P∨~P。) 再重复上述过程，一直到推出空子句的预解式为止，就产生了序列 S^0，S^1，…。该方法称为**级饱和（分解）方法**。用数学式表达如下：

$S^0 = S$

$S^n = \{C_1$ 和 C_2的预解式$|C_1 \in (S^0 \cup \cdots \cup S^{n-1})$，$C_2 = S^{n-1}\}$，$n = 1$，$2\cdots$

例 3.33　在例 3.32 中

S^0：(1) $\sim T(x,y,u,v) \vee P(x,y,u,v)$

(2) $\sim P(x,y,u,v) \vee E(x,y,v,u,v,y)$

(3) $T(a,b,c,d)$

(4) $\sim E(a,b,d,c,d,b)$

S^1：(5) $\sim T(x,y,u,v) \vee E(x,y,v,u,v,y)$ (由(1)、(2)产生)

(6) $P(a,b,c,d)$ (由(1)、(3)产生)

(7) $\sim P(a,b,c,d)$ (由(2)、(4)产生)

(注：(1)与(4)、(2)与(3)、(3)与(4)之间分别都没有预解式)

S^2：(8)$\sim T(a,b,c,d)$ (由(1)、(7)产生)

(9) $E(a,b,d,c,d,b)$ (由(2)、(6)产生)

(10)$E(a,b,d,c,d,b)$ (由(3)、(5)产生)

(11)$\sim T(a,b,c,d)$ (由(4)、(5)产生)

(12)□ (由(6)、(7)产生)

(注：(1)与(5)、(1)与(6)，…等没有预解式)

从上述过程可见，不仅是(5)(8)(9)(10)(11)是多余的，而且(9)与(10)、(8)与(11)是重复的。如果万一S^2中未推出空子句，(9)与(10)会重复产生预解式，而造成不必要的存贮空间和演算时间的浪费。为此，就必须应用删除策略。

在介绍删除策略之前，先介绍子句包含的概念及包含算法。

包含 对于两个子句 C 与 D，如果有一个置换 σ 存在使得 $C\sigma \subseteq D$，则称**子句 C 包含着子句 D**。称 **D 为被包含子句**。

注意，C 与 D 相同，或者 D 是 C 的一个特例，则也称 C 包含着 D。

例 3.34 设 $C = P(x)$，$D = P(a) \vee Q(a)$。如果 $\sigma = \{a/x\}$，则 $C\sigma = P(a)$。因为 $C\sigma = P(a) \subseteq P(a) \vee Q(a) = D$，所以 C 包含着 D。

包含算法 令 C 和 D 是子句，令 $\theta = \{a_1/x_1, \cdots, a_n/x_n\}$，其中 $x_1, \cdots, x_n$ 是 D 中出现的变元，且 $a_1, \cdots, a_n$ 是既不在 C 中也不在 D 中出现的新的各不相同的常量。假设 $D = L_1 \vee L_2 \vee \cdots \vee Lm$，则 $D\theta = L_1\theta \vee L_2\theta \vee \cdots \vee L_m\theta$。$\sim D\theta = \sim L_1\theta \wedge \sim L_2\theta \wedge \cdots \wedge \sim L_m\theta$。那么测试 C 是否包含着 D 的算法如下：

步骤 1：令 $W = \{\sim L_1\theta, \cdots, \sim L_m\theta\}$

步骤 2：设置 $k = 0$　$U^0 = \{C\}$

步骤 3：如果 U^k 含有空子句□，则算法结束，说明 C 包含着 D；否则，令 $U^{k+1} = \{C_1$ 和 C_2 的预解式$|C_1 \in U^k$ 且 $C_2 \in W\}$。

步骤 4：如 U^{k+1} 是空的，则算法结束，说明 C 不包含着 D；否则，设置 $k \leftarrow k+1$，且转到步骤 3。

举二例说明如何应用该算法。

例 3.35 令 $C = \sim P(x) \vee Q(f(x),a)$，$D = \sim P(h(y)) \vee Q(f(h(y)),a) \vee \sim P(z)$。问 C 是否包含着 D？

(1) y 和 z 是 D 中的变元。令 $\theta = \{b/y, c/z\}$。注意，b 和 c 都不在 D 中。则 $D\theta = \sim P(h(b)) \vee Q(f(h(b)),a) \vee \sim P(c)$。$\sim D\theta = P(h(b)) \wedge \sim Q(f(h(b)),a) \wedge P(c)$。因此

$W=\{P(h(b)),\ \sim Q(f(h(b)),a),\ P(c)\}$

$U^0=\{\sim P(x)\vee Q(f(x),a)\}$

(2) 因为 U^0 不含有□，则得

$$U^1=\{Q(f(h(b)),a),\ \sim P(h(b)),\ Q(f(c),a)\}$$

(3) 因为 U^1 不是空集，且不含有□，则得

$$U^2=\{□,\ □\}$$

(4) 因为 U^2 含有□，则算法终结。从而得出 C 包含着 D 的结论。

例 3.36　令 $C=P(x,x)$，$D=P(f(x),y)\vee P(y,f(x))$。问 C 是否包含着 D?

(1) x、y 是 D 中的变元。选择不在 C、D 中出现的新常量 a，b。令 $\theta=\{a/x,\ b/y\}$。则

$$D\theta=P(f(a),b)\vee P(b,f(a))$$

$$\sim D\theta=\sim P(f(a),b)\wedge\sim P(b,f(a))$$

因此　$W=\{\sim P(f(a),b),\ \sim P(b,f(a))\}$

$U^0=\{P(x,x)\}$

(2) 因为 U^0不含有□，则得

$$U^1=\phi\ (=\{\ \})$$

(3) 因为 U^1 是空集合，则算法终结。得出 C 不包含着 D 的结论。

有了以上算法，在级饱和方法中就可以采用如下的删除策略了。

删除策略　在上述级饱和方法中，首先把 $(S^0\cup\cdots\cup S^{n-1})$ 中的子句按序列入表中；而后把 $S^0\cup\cdots\cup S^{n-1}$ 中的每个子句 C_1 与 S^{n-1} 中的一个（表中列于 C_1 之后的）子句 C_2 加以比较，而求出预解式。如果该预解式是同义反复（即形如 $P\vee\sim P$）的，或者是被 $S^0\cup\cdots\cup S^{n-1}$ 表中的任一子句所包含，则删除之；否则，将该预解式列于表的末尾。

例 3.37　$S=\{P\vee Q,\sim P\vee Q,P\vee\sim Q,\sim P\vee\sim Q\}$

读者可自行试一试。如果不采用删除策略，则直到推出□为止，S^1有八个成员，而 S^2 有27个成员，共有 39 个成员。但采用删除策略，则只有 9 个成员，列于下：

S^0: (1) $P\vee Q$
(2) $\sim P\vee Q$
(3) $P\vee\sim Q$
(4) $\sim P\vee\sim Q$
} S

S^1: (5) Q　　由(1)、(2)推出
(6) P　　由(1)、(3)推出
(7) ~P　　由(2)、(4)推出
(8) ~Q　　由(3)、(4)推出

S^2: (9) □　　由(5)、(8)推出

还有许多有关分解原理的改进方法。篇幅所限，从略。

第三章 习 题

一、把下列陈述用命题形式表达出来：

(1) 如果湿度很高的话，不是今天下午下雨，就是晚上下雨。

(2) 除非查清癌症的病因并且找到治癌新药，否则癌症无法征服。

(3) 如果能买到火车票的话，我就到北京开会去。否则，买得到飞机票，我也去开会，再买不到飞机票，就只好不去开会了。

(4) 要登上珠峰，就需要有足够的勇气、充沛的体力和互助的精神。

二、为了给学生食堂的优劣予评价，设 A 表示“食物是好吃的”，B 表示“服务态度是佳的”，C 表示“价格是便宜的”，D 表示“学生对食堂还是满意的”，E 表示“学生对食堂很满意”。试将下列各命题用语言表达出来：

(1) $A \wedge B \wedge C \leftrightarrow E$;

(2) $\sim C \rightarrow \sim E$;

(3) $\sim E \rightarrow \sim A \vee \sim B \vee \sim C$

(4) $(A \wedge B) \vee (A \wedge C) \vee (B \wedge C) \rightarrow D$

(5) $D \wedge \sim A \rightarrow B \wedge C$

(6) $C \rightarrow\rightarrow D, (A \wedge B \rightarrow D)$

三、把公式 $(\sim P \vee Q) \wedge (\sim (P \wedge \sim Q))$ 的真值表列出来

P	Q	$\sim P$	$\sim Q$	$\sim P \vee Q$	$P \wedge \sim Q$	$\sim (P \wedge \sim Q)$	$(\sim P \vee Q) \wedge (\sim (P \wedge \sim Q))$

四、下列公式是永真公式、或永假公式、或非永真公式、或非永假公式，或它们兼有之？为什么？

(1) $\sim (\sim P) \rightarrow P$　　(2) $P \rightarrow (P \wedge Q)$

(3) $\sim (P \vee Q) \vee \sim Q$　　(4) $(P \vee Q) \rightarrow P$

(5) $(P \rightarrow Q) \rightarrow (\sim Q \rightarrow \sim P)$　　(6) $(P \rightarrow Q) \rightarrow (Q \rightarrow P)$

(7) $P \vee (P \rightarrow Q)$　　(8) $(P \wedge (Q \rightarrow P)) \rightarrow P$

(9) $P \vee (Q \rightarrow \sim P)$　　(10) $(P \vee \sim Q) \wedge (\sim P \vee Q)$

(11) $\sim P \wedge (\sim (P \rightarrow Q))$　　(12) $(P \wedge Q) \rightarrow\rightarrow P, Q$

五、把下列公式变为合取范式和析取范式：

(1) $(\sim P \wedge Q) \rightarrow R$　　(2) $P \rightarrow ((Q \wedge R) \rightarrow S)$

(3) $P\rightarrow\rightarrow Q,(P\wedge R)$　　(4) $(P\rightarrow\rightarrow Q,\square)\rightarrow R$

(5) $\sim(P\wedge Q)\wedge(P\vee Q)$　　(6) $(P\rightarrow Q)\rightarrow R$

六、令 $P(x)$和 $Q(x)$分别表示 "x 是一个有理数" 和 "x 是一个实数"。将下列陈述用符号表示出来：

(1) 每个有理数都是一个实数。

(2) 某些实数是有理数。

(3) 不是每个实数都是有理数。

(4) 某些实数不是有理数。

七、令 $S(x)$表示 "x 是三好学生"，$J(x)$表示 "x 是健康的"。x的定义域为学生。将下列各式译成汉语：

(1) $(\exists x)S(x)$

(2) $(\exists x)J(x)$

(3) $(\forall x)(S(x)\rightarrow J(x))$

(4) $(\exists x)(S(x)\wedge\sim J(x))$

(5) $(\exists x)(J(x)\wedge\sim S(x))$

八、将下列句子表示成合式公式：

"对于任意二个点，必有而且仅有一条直线通过这二个点。"

九、设有如下解释 ($D=\{a,b\}$)：

$P(a,a)$	$P(a,b)$	$P(b,a)$	$P(b,b)$
T	F	F	T

试求出下列各公式的真值

(1) $(\forall x)(\exists y)P(x,y)$

(2) $(\forall x)(\forall y)P(x,y)$

(3) $(\exists x)(\forall y)P(x,y)$

(4) $(\exists y)(\sim P(a,y))$

(5) $(\forall x)(\forall y)(P(x,y)\rightarrow P(y,x))$

(6) $(\forall x)P(x,y)$

十、考虑到如下解释：

定义域：$D=\{1,2\}$

常量 a 和 b 的设定值为

a	b
1	2

函词 f（　）的设定值为

$f(1)$	$f(2)$
2	1

谓词 P(,)的设定值为

P(1,1)	P(1,2)	P(2,1)	P(2,2)
T	T	F	F

试求出在上面解释下，下列公式的真值：

(1) $P(a,f(a))\wedge P(b,f(b))$

(2) $(\forall x)(\exists y)P(y,x)$

(3) $(\forall x)(\forall y)(P(x,y)\rightarrow P(f(x),f(y)))$

十一、令 F_1 和 F_2 为

F_1:$(\forall x)(P(x)\rightarrow Q(x))$

F_2:$\sim Q(a)$

试证明$\sim P(a)$是 F_1 和 F_2 的逻辑推论。

十二、将下列公式变成前束范式

(1) $(\forall x)(P(x)\rightarrow(\exists y)Q(x,y))$

(2) $(\exists x)(\sim((\exists y)P(x,y))\rightarrow((\exists z)Q(z)\rightarrow R(x)))$

(3) $(\forall x)(\forall y)((\exists z)P(x,y,z)\wedge((\exists u)Q(x,u)\rightarrow(\exists v)Q(y,v)))$

十三、对下列各式找出其 Skolem 标准形

(1) $\sim((\forall x)P(x)\rightarrow(\exists y)(\forall z)Q(y,z))$

(2) $(\forall x)((\sim E(x,a))\rightarrow((\exists y)(E(y,g(x))\wedge(\forall z)(E(z,g(x))\rightarrow E(y,z)))))$

(2) $(P \vee Q) \wedge (R \vee Q) \wedge \sim R \wedge \sim Q$

(3) $(P \vee Q) \wedge (\sim P \vee Q) \wedge (\sim R \vee \sim Q) \wedge (R \vee \sim Q)$

二十、根据 Davis 和 Putnam 提出的规则，试证明下列公式是可满足的。

(1) $P \wedge Q \wedge R$

(2) $(P \vee Q) \wedge (\sim P \vee Q) \wedge R$

(3) $(P \vee Q) \wedge \sim Q$

二十一、应用分解原理证明子句集$S = \{P \vee Q \vee R, \sim P \vee R, \sim Q, \sim R\}$，是不可满足的。

二十二、对于子句集 $S = \{P \vee Q, \sim Q \vee R, \sim P \vee Q, \sim R\}$，试用分解原理从 S 导出空子句。

二十三、令 $\theta = \{a/x, b/y, g(x,y)/z\}$是一个置换以及 $E = P(h(x), z)$。试找出 $E\theta$。

二十四、令 $\theta_1 = \{a/x, f(z)/y, y/z\}$ 和 $\theta_2 = \{b/x, z/y, g(x)/z\}$。试找出 θ_1 和 θ_2 的复合。

二十五、下列各个集合能否一致化？如能一致化，则找出其最普通的一致置换。

(1) $W = \{Q(a), Q(b)\}$

(2) $W = \{Q(a,x), Q(a,a)\}$

(3) $W = \{Q(a,x,f(x)), Q(a,y,y)\}$

(4) $W = \{Q(x,y,z), Q(u,h(v,v),u)\}$

二十六、分别找出下列子句的所有可能的预解式

(1) $C = \sim P(x) \vee Q(x,b)$，$D = P(a) \vee Q(a,b)$

(2) $C = \sim P(x) \vee Q(x,x)$，$D = \sim Q(a, f(a))$

(3) $C = \sim P(x,y,u) \vee \sim P(y,z,u) \vee \sim P(x,v,w) \vee P(u,z,w)$

$D = P(g(x,y), x, y)$

(4) $C = \sim P(v,z,v) \vee P(w,z,w)$，

$D = P(w, h(x,x), w)$

二十七、应用分解原理证明第十八题中的子句集是不可满足的。

二十八、应用分解原理证明 $(\sim Q \rightarrow \sim P)$ 是 $(P \rightarrow Q)$ 的逻辑推论。反之亦然。

二十九、令 $C = P(x,y) \vee Q(z)$ 及 $D = Q(a) \vee P(b,b) \vee R(u)$。试问，C 包含着 D 吗？

三十、令 $C = P(x,y) \vee R(y,x)$ 和 $D = P(a,y) \vee R(z,b)$。试证明 C 不包含着 D。

（以上两题用包含算法求之）

第四章 模 式 识 别*

模式识别就是对某些过程或事件的集合加以判别或分类。这些被判别或分类的过程或事件可以是物理或化学或生物等等客体，也可以是心理状态。人类在其生活活动的几乎是一瞬间当中，已执行了模式识别的任务。在观察了愈来愈多的模式之后，人们就给予一定分类，并注意有否新模式类出现。模式分类的数目常常根据所关心的特殊应用来确定。例如识别英文字母问题，就是一个 26 类问题。而如果要区别出英文字母和汉字，就是一个 2 类问题。

模式识别的研究已有了可观的成就，应用也很广。可用机器识别英文印刷体和某些手写体；可识别数字 0～9 的手写体；可用来识别染色体，文字；可用来分析处理气象云图、空中摄影及图片识别等等。

用于模式识别的方法很多，粗略分为三种：决策论方法、句法方法和机器视觉。最后一部分放在《智能机器人》一章中介绍。

4.1 决 策 论 方 法

决策论方法的基本识别系统由二部分组成：特征抽取器和分类器。如图 4.1 所示。当一个模式输入时，特征抽取器就从该模式中抽取它的特性测度（称为**特征**）。而分类器通常根据这些特征由划分特征空间的办法来识别每个模式，从而做出决策，把它们分至某一个模式类。假设这些特征是不变量，或者对于常遇到的变化和畸变较不敏感，而且还包含较少的多余物。在这些前提下，模式识别问题就归结成二个子问题。第一个子问题是从输入模式中应取什么样的特征。这点是人为定出的，也依赖于实际情况。例如，测度的有效性和测度的成本等。遗憾得很，关于特征测度的选择，至今很少有一般理论。第二个子问题是从根据所选特征而抽取的测度中进行分类。本节着重介绍分类法。

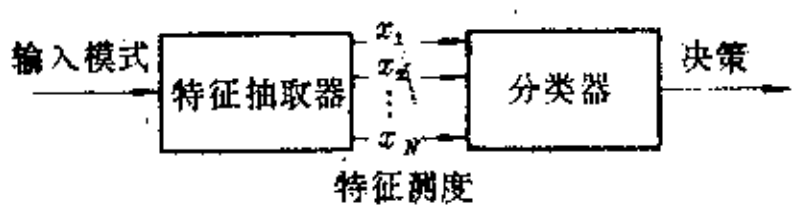
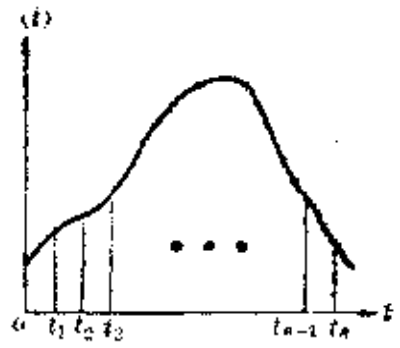


图 4.1 一个模式识别系统

根据所选择的特征，对一个输入模式抽取一组测度，而构成一个矢量。称它为**特征矢量**。例如，对一个声波的测量，若是时间采样，则 n 个采样值就构成一个特征矢量，如图 4.2(a)所示。再如对英文字母（如 a）的测量，若采用网格方式，则第一网格至第 n 个网格的测度（有字迹或空白），也构成一个特征矢量，如图 4.2(b) 所示。这些测度 $x(t_i)$或 x_i 都是随机变量。故每个模式可用 n 维空间中的一个随机矢量表示，用黑体字母 **X** 或 $x(t_i)$ 记之，或者看成是 n 维空间中的一个点。

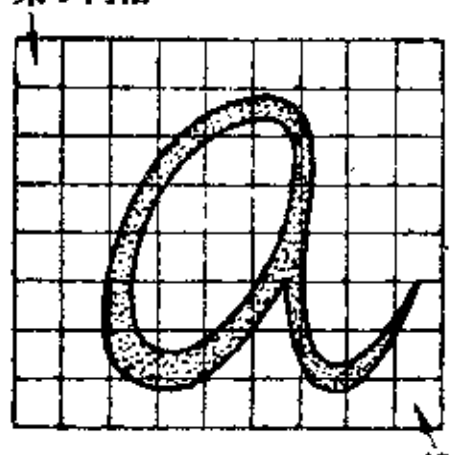
---

* 本章主要参考文献见“参考文献”2、4、5、6、7、8 和 14。



$$\mathbf{X}=\begin{bmatrix}X_1\\X_2\\\vdots\\X_n\end{bmatrix}=\begin{bmatrix}X(t_1)\\X(t_2)\\\vdots\\X(t_n)\end{bmatrix}$$

(a)



$$\mathbf{X}=\begin{bmatrix}X_1\\X_2\\\vdots\\X_n\end{bmatrix}=\begin{bmatrix}X(1)\\X(2)\\\vdots\\X(n)\end{bmatrix}$$

(b)

图 4.2　模式的矢量表示

模式分类的概念可以表示为特征空间的划分，或者表示为从特征空间至决策空间的映照。分类的问题是将特征空间中每种可能的矢量或点指定为本征模式类。这能解释为特征空间至互不复盖的区域的一个映照。从数学角度来看，分类问题能够公式化成为**判别函数**。

4.1.1　样本匹配法

一般说来，**样本匹配法**可解释为图 4.1 中的特征抽取的一种特殊情况。其中，样本是按照特征测度而存贮起来的。而且分类器采用专门的分类匹配法。

例 4.1　语音识别系统的一种简图，如图 4.3 所示。讲话者可先读单词，经过 n 个带通滤波器 Δf_i 加以采样。也可以用等间隔时间 Δt_i 采样。其 n 个特征组成的矢量就作为样本存贮起来。各单词经过训练之后，讲话者再读入单词时，机器就将输入模式的特征矢量与样本加以比较。如果与某个样本相匹配，则判定该单词属于该样本这类。但讲话者在读同一单词时，难免有些差异。为了避免这种差异，也可以一个单词重复讲几次。或者求其平均值作为样本，或者同一单词有多种样本。

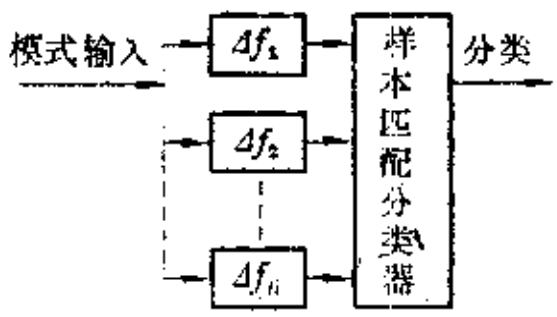


图 4.3　一种语音识别系统简图

样本匹配法是比较简单的。但是如果模式的分类较多，而且有若干细小差异的话，则样本的种类就很多，计算机的容量就成问题了。

4.1.2　基本的非参量决策论分类法

令 $\omega_1,\omega_2,\cdots\omega_m$ 表示欲被识别的 m 种可能的模式类，$\mathbf{X}$ 是图 4.2 所示的输入模式的特

征矢量。用 $X\sim\omega_i$ 表示作出输入模式 X 属于模式类 ω_i 的决策。而 $D_j(X)$表示与 ω_j 有关的**判别函数** $(j=1,2,\cdots m)$。如果 $X\sim\omega_i$ 的话，则 $D_i(X)$的值必须是最大的。也就是说，对于所有的 $X\sim\omega_i$，则

$$D_i(X)>D_j(X) \qquad i,j=1,\cdots m,\ i\neq j \tag{4.1}$$

因此，在特征空间中，在分别与类 ω_i 和 ω_j 有关的区域之间的交界（称为**决策边界**）由如下方程表示

$$D_i(X)-D_j(X)=0 \tag{4.2}$$

可见，只要能找到判别函数 $D_i(X)$的具体形式，对于每个输入模式，根据式 (4.1)，求出最大的 $D_i(X)$，从而可得出 $X\sim\omega_i$ 的结论。

下面讨论若干重要的判别函数。

1. 线性判别函数及其线性分类器的训练

线性判别函数就是将 $D_i(X)$选择为特征测度 $x_1,x_2\cdots x_N$ 的线性组合，即

$$D_i(X)=\sum_{k=1}^{N}(W_{ik}x_k+W_{i,N+1}) \quad i=1,\cdots,m \tag{4.3}$$

上式可用矢量（或矩阵）方式表达如下：

$$D_i(X)=\begin{pmatrix}x_1\\x_2\\\vdots\\x_N\\1\end{pmatrix}^T\begin{pmatrix}w_{i1}\\w_{i2}\\w_{iN}\\\vdots\\w_{i,N+1}\end{pmatrix}=\begin{pmatrix}X\\\\\\1\end{pmatrix}^T\begin{pmatrix}w_{i1}\\w_{i2}\\w_{iN}\\\vdots\\w_{i,N+1}\end{pmatrix}=Y^TW \tag{4.4}$$

其中 X 是特征矢量，Y 称为 X 的**增广特征矢量**，T 表示矩阵或矢量的转置。

在特征空间中，与 ω_i、ω_j 有关的决策边界为如下形式：

$$\begin{aligned}D_i(X)-D_j(X)&=\sum_{k=1}^{N}((W_{ik}-W_{jk})x_k+(W_{i\,(N+1)}-W_{j\,(N+1)}))\\&=\sum_{k=1}^{N}W_kx_k+W_{N+1})=0\end{aligned} \tag{4.5}$$

其中 $W_k=W_{ik}-W_{jk}$，$W_{N+1}=W_{i\,(N+1)}-W_{j\,(N+1)}$。这是**超平面**。

如果 $m=2$，根据式 (4.4)，$i,j=1,2(i\neq j)$，就把如图 4.4 所示的一个阈值逻辑器件作为线性判别函数的线性分类器。在图 4.4 中，令$D(X)=D_1(X)-D_2(X)$，

$$\begin{aligned}&\text{如}\quad D(X)>0,\ \text{输出}=+1,\ \text{则}\quad X\sim\omega_1\\&\text{如}\quad D(X)<0,\ \text{输出}=-1,\ \text{则}\quad X\sim\omega_2\end{aligned} \tag{4.6}$$

模式类的数目若大于 2$(m>2)$，就可把若干个阈值逻辑器件加以并联。因此，从M个阈值逻辑器件的输出组合，就足够区别出 m 类$(m\leqslant 2^M)$了。

如果从不同类来的模式都可能由式 (4.4) 加以区分的话，那么只要式 (4.5) 中的系

数(或权) $W_1, W_2 \cdots W_{N+1}$ 具有适当的值，就有可能得到完美的正确识别。然而，实际上，原先的权通常并不是有效的。在这种情况下，就要将分类器设计得使它具有能从输入模式而估算权的最佳值的能力。其基本思想是这样的，为了得到正确的识别，经过观察事先已知其类别的一些模式，分类器要能够自动地调整这些权。所用的模式愈多，分类器的性能应当更得到改善。这个过程称为**训练**或**学习**，而用于输入的模式称为**训练模式**。

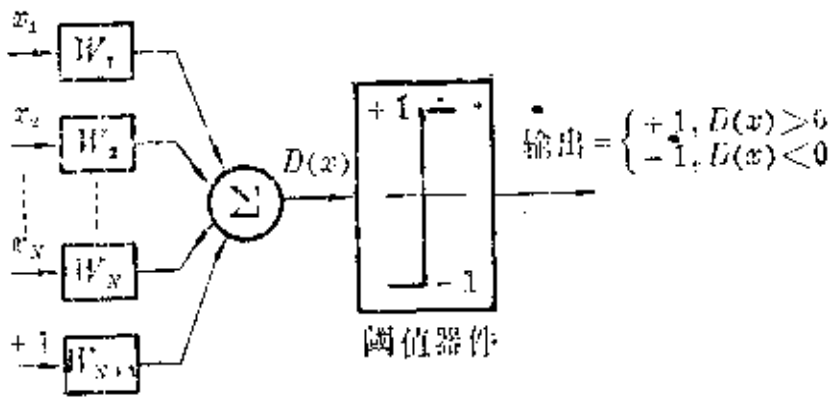


图 4.4　线性的二类分类器

令分别属于二个不同模式类 ω_1 和 ω_2 的训练模式的二个集合为 T_1' 和 T_2'。对应于这两个训练集合就分别有增广矢量的二个集合 T_1 和 T_2。而在 T_1 和 T_2 中的每个成员都是分别由 T_1' 和 T_2' 中的模式加以增广而得到的。两个训练集合能线性地区分，就意味着存在一个权矢量（称这个解为**权矢量**）使得

$$\mathbf{Y}^T\mathbf{W}>0 \qquad \text{对于每个 } \mathbf{Y}\in T_1 \tag{4.7}$$

而且

$$\mathbf{Y}^T\mathbf{W}<0 \qquad \text{对于每个 } \mathbf{Y}\in T_2 \tag{4.8}$$

其中 $\mathbf{Y}$ 和 $\mathbf{W}$ 与式 (4.4) 相同。

线性分类器的所谓**“纠错”**过程可概述如下：对于任一 $\mathbf{Y}\in T_1$，$\mathbf{Y}^T\mathbf{W}$ 必是正的，即 $\mathbf{Y}^T\mathbf{W}>0$。如果分类器的输出是错误的（即 $\mathbf{Y}^T\mathbf{W}<0$）或不确定（即 $\mathbf{Y}^T\mathbf{W}=0$），则令新的权矢量为

$$\mathbf{W}'=\mathbf{W}+\alpha\mathbf{Y} \tag{4.9}$$

其中 $\alpha>0$ 称为**纠正增量**。另一方面，对于 $\mathbf{Y}\in T_2$，$\mathbf{Y}^T\mathbf{W}<0$。如果分类器的输出有错误（即 $\mathbf{Y}^T\mathbf{W}>0$）或不确定，则令

$$\mathbf{W}'=\mathbf{W}-\alpha\mathbf{Y} \tag{4.10}$$

在训练开始之前，$\mathbf{W}$可以根据方便而预置任意值。

α 的选择规则有三种：

(1) **固定增量规则**：即 α 是任意的固定正数。

(2) **绝对纠正规则**：即 α 取为比 $\dfrac{|\mathbf{Y}^T\mathbf{W}|}{\mathbf{Y}^T\mathbf{Y}}$ 略大的最小整数 (4.11)

(3) **部分纠正规则**：即选取 α 使得

$$|\mathbf{Y}^T\mathbf{W}-\mathbf{Y}^T\mathbf{W}'|=\lambda|\mathbf{Y}^T\mathbf{W}| \quad 0<\lambda\leqslant 2 \tag{4.12}$$

或者等效于

$$\alpha=\lambda\frac{|\mathbf{Y}^T\mathbf{W}|}{\mathbf{Y}^T\mathbf{Y}} \tag{4.13}$$

3. 最小距离分类器

最小距离分类器就是把输入模式和特征空间中参考矢量集合的成员之间的距离作为分类标准。假设给定 m 个参考矢量 $\mathbf{R}_1$，$\mathbf{R}_2$，…，$\mathbf{R}_m$（其中 $\mathbf{R}_j$ 是模式类 ω_j 的参考矢量）。

如果 $|\mathbf{X}-\mathbf{R}_i|$ 为最小值，则 $\mathbf{X}\sim\omega_i$ (4.14)

其中 $|\mathbf{X}-\mathbf{R}_i|$ 是 $\mathbf{X}$ 和 $\mathbf{R}_i$ 之间定义的距离。例如，$|\mathbf{X}-\mathbf{R}_i|$ 可以定义为

$$|\mathbf{X}-\mathbf{R}_i| = \sqrt{(\mathbf{X}-\mathbf{R}_i)^{\mathrm{T}}(\mathbf{X}-\mathbf{R}_i)} \tag{4.15}$$

由式 (4.15) 可得

$$|\mathbf{X}-\mathbf{R}_i|^2 = \mathbf{X}^{\mathrm{T}}\mathbf{X} - \mathbf{X}^{\mathrm{T}}\mathbf{R}_i - \mathbf{R}_i^{\mathrm{T}}\mathbf{X} + \mathbf{R}_i^{\mathrm{T}}\mathbf{R}_i \tag{4.16}$$

因为 $\mathbf{X}^{\mathrm{T}}\mathbf{X}$ 不是 i 的函数，所以对应的最小距离分类器的判别函数可以是

$$\mathrm{D}_i(\mathbf{X}) = \mathbf{X}^{\mathrm{T}}\mathbf{R}_i + \mathbf{R}_i^{\mathrm{T}}\mathbf{X} - \mathrm{R}_i^{\mathrm{T}}\mathbf{R}_i \quad i=1,\cdots m, \tag{4.17}$$

这是线性的。因而该分类器也是一个**线性分类器**。最小距离分类器的性能当然依赖于参考矢量集合的选择。

3. 近邻域法（分段线性判别函数）

令 $\mathbf{R}_1$，$\mathbf{R}_2$，…，$\mathbf{R}_m$ 是 m 个参考集合，其中 $\mathbf{R}_j(j=1,\cdots,m)$ 是关于模式类 ω_j 的参考集合，$\mathbf{R}_j=\{\mathbf{R}_j^{(1)},\cdots,\mathbf{R}_j^{(u_j)}\}$，$u_j$ 是 $\mathbf{R}_j$ 中参考矢量的数目。输入的特征矢量 $\mathbf{X}$ 与 $\mathbf{R}_j$ 之间的距离定义为 $\mathbf{X}$ 与 $\mathbf{R}_j$ 中每个矢量之间距离的最小者。如下式所示。

$$\mathrm{d}(\mathbf{X},\mathbf{R}_j) = \underset{k=1,\cdots,u_j}{\mathrm{Min}}|\mathbf{X}-\mathbf{R}_j^{(k)}| \tag{4.18}$$

在这种情况下，判别函数为

$$\mathrm{D}_j(\mathbf{X}) = \underset{k=1,\cdots,u_j}{\mathrm{Max}}\{\mathbf{X}^{\mathrm{T}}\mathbf{R}_j^{(k)} + (\mathbf{R}_j^{(k)})^{\mathrm{T}}\mathbf{X} - (\mathrm{R}_j^{(k)})^{\mathrm{T}}\mathbf{R}_j^{(k)}\} \quad j=1,\cdots,m \tag{4.19}$$

令

$$\mathrm{D}_j^{(k)} = \mathbf{X}^{\mathrm{T}}\mathbf{R}_j^{(k)} + (\mathbf{R}_j^{(k)})^{\mathrm{T}}\mathbf{X} - (\mathbf{R}_j^{(k)})^{\mathrm{T}}\mathbf{R}_j^{(k)} \tag{4.20}$$

则

$$\mathrm{D}_j(\mathbf{X}) = \underset{k=1,\cdots,u_j}{\mathrm{Max}}\{\mathrm{D}_j^{(k)}(\mathbf{X})\} \quad j=1,\cdots,m, \tag{4.21}$$

可以看出，$\mathrm{D}_k^{(k)}(\mathbf{X})$ 是特征矢量 $\mathbf{X}$ 的线性组合。因而用式 (4.19) 或式 (4.21) 的分类器，常称为**分段线性分类器**。$\mathrm{D}_j(\mathbf{X})$ 称为**分段线性判别函数**。

例 4.2　在手写体英文字母中，a 与 o 的手写体可有多种，如图 4.5 所示的两个参考集合 a 与 o。各有五个参考矢量。

现在输入一个如图 4.6 所示的手写字样。该输入字样与 a 类的参考矢量比较，与 a_3 最接近；同样，与 o 类中的 o_5 最接近。a_3 与 o_5 分别作为 a 类和 o 类的代表。输入字样与 a_3 更接近，则分至 a 类；另一种情况，则分至 o 类。

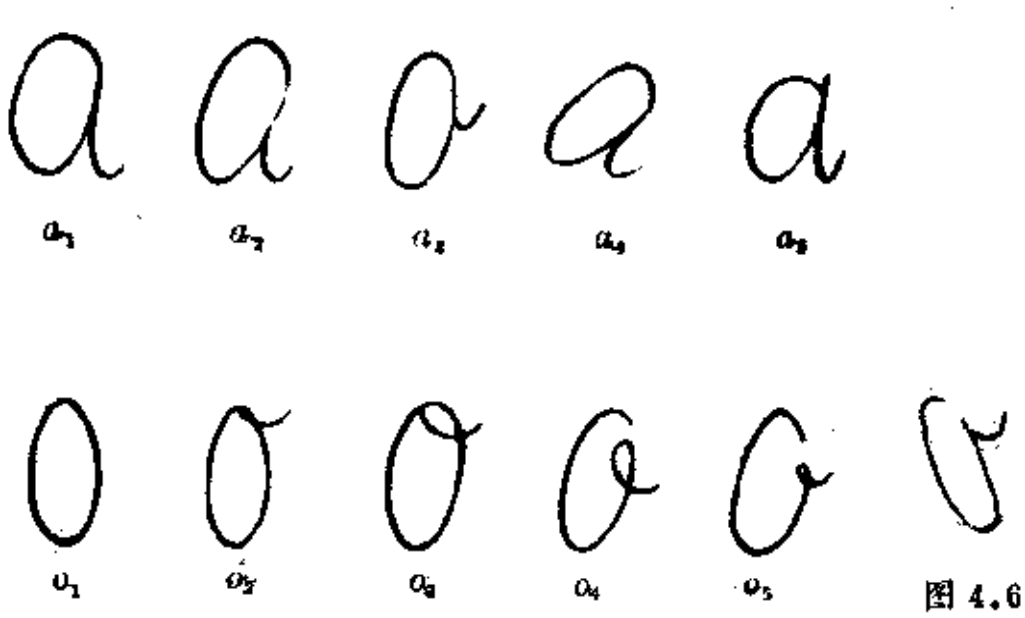


图 4.5　手写体 a 与 o 的字形

图 4.6　输入字样

4. 多项式判别函数

一个 r 阶多项式判别函数可表达为

$$D_i(\mathbf{X}) = \omega_{i_1} F_1(x) + \omega_{i_2} F_2(x) + \cdots + \omega_{i_L} F_L(x) + \omega_{i,L+1} \tag{4.22}$$

其中 $F_j(x)$是 $X_{k_1}^{n_1} X_{k_2}^{n_2} \cdots X_{k_r}^{n_r}$ 的形式，$k_1, k_2, \cdots, k_r = 1, \cdots, N$，且 $n_1, n_2, \cdots n_r = 0$ 或 1。在任意二类之间的决策边界也是 r 阶多项式的形式。

特别地，如 $r=2$，则称其判别函数为**二阶判别函数**。在这种情况下

$$F_j(\mathbf{X}) = X_{k_1}^{n_1} X_{k_2}^{n_2} \quad k_1, k_2 = 1, \cdots, N，且 n_1, n_2 = 0 或 1 \tag{4.23}$$

而且

$$L = \frac{1}{2} N(N+3) \tag{4.24}$$

$$D_i(\mathbf{X}) = \sum_{k=1}^{N} \omega_{kk} x_k^2 + \sum_{j=1}^{N-1} \sum_{k=j+1}^{N} \omega_{jk} X_j X_k + \sum_{j=1}^{N} \omega_j X_j + \omega_{L+1} \tag{4.25}$$

4.1.3 Bayes（参量）分类

在本小节中，所讨论的特征矢量都是随机矢量。

如果每个模式不是属于 ω_1 类，就是属于 ω_2 类，则这类分类问题称为二类问题。对于一个输入模式所测的特征矢量 $\mathbf{X}$，若要判别它属于 ω_1 或 ω_2 类，则可简单地根据如下的概率决策规则加以判别：

$$\begin{cases} 若\ P(\omega_1|\mathbf{x}) > P(\omega_2|\mathbf{x})，则\ \ \mathbf{X} \sim \omega_1 \\ 若\ P(\omega_2|\mathbf{x}) > P(\omega_1|\mathbf{x})，则\ \ \mathbf{X} \sim \omega_2 \end{cases} \tag{4.26}$$

其中 $P(\omega_i|\mathbf{x})$表示特征矢量为 $\mathbf{X}$ 的模式是属于 ω_i 类的概率。这个判别规则是可行的。因为既然所测得的特征 $\mathbf{X}$ 能使 $P(\omega_1|\mathbf{x}) > P(\omega_2|\mathbf{x})$，我们自然倾向于作出决策 $\mathbf{X} \sim \omega_1$，根据式（4.26），对于 $\mathbf{X}$ 的错误决策概率 $P(e|\mathbf{x})$如下：

$$P(e|\mathbf{x}) = \begin{cases} P(\omega_1)|\mathbf{x}), & 若\ \ \mathbf{X} \sim \omega_2 \\ P(\omega_2)|\mathbf{x}), & 若\ \ \mathbf{X} \sim \omega_1 \end{cases} \tag{4.27}$$

式（4.26）的判别规则强调了后验概率的作用。利用 Bayes 定理，后验概率 $P(\omega_i|\mathbf{x})$ 可由先验概率 $P(\omega_i)$和条件密度函数 $P(\mathbf{X}|\omega_i)$来计算，即

$$P(\omega_i|\mathbf{x}) = \frac{P(\mathbf{x}|\omega_i)P(\omega_i)}{P(\mathbf{x})} \qquad i = 1, 2 \tag{4.28}$$

其中

$$P(\mathbf{x}) = \sum_i P(\mathbf{x}|\omega_i)P(\omega_i)$$

因而将式（4.28）代入式（4.26），再考虑到 $P(\mathbf{x})$是相同的，则得到了 **Bayes 决策规则**：

$$\begin{cases} 若\ P(\mathbf{x}|\omega_1)P(\omega_1) > P(\mathbf{x}|\omega_2)P(\omega_2)，则\ \ \mathbf{X} \sim \omega_1 \\ 若\ P(\mathbf{x}|\omega_2)P(\omega_2) > P(\mathbf{x}|\omega_1)P(\omega_1)，则\ \ \mathbf{X} \sim \omega_2 \end{cases} \tag{4.29}$$

如果模式类别 ω_i 有 m 类，$i = 1, \cdots, m$，即 m 类问题，那么式（4.26）的决策规则为

$$若\ P(\omega_i|\mathbf{x}) > P(\omega_j|\mathbf{x}),\ j = 1, \cdots, m,\ j \neq i，则\ \ \mathbf{X} \sim \omega_i \tag{4.30}$$

而 Bayes 决策规则如下：

$$\begin{aligned} &若\ P(\mathbf{x}|\omega_i)P(\omega_i) > P(\mathbf{x}|\omega_j)P(\omega_j),\ (j = 1, \cdots, m,\ j \neq i) \\ &则 \qquad\qquad \mathbf{X} \sim \omega_i \end{aligned} \tag{4.31}$$

式（4.31）也可改写成：

$$\lambda(\mathbf{x}) = \frac{P(\mathbf{x}|\omega_i)}{P(\mathbf{x}|\omega_j)} > \frac{P(\omega_j)}{P(\omega_i)} \quad j = 1, \cdots, m, j \neq i \quad 则\ \mathbf{X} \sim \omega_i \tag{4.32}$$

其中 λ(x)称为**似然比**。$P(\omega_j)/P(\omega_i)$为判别的似然比的阈值。

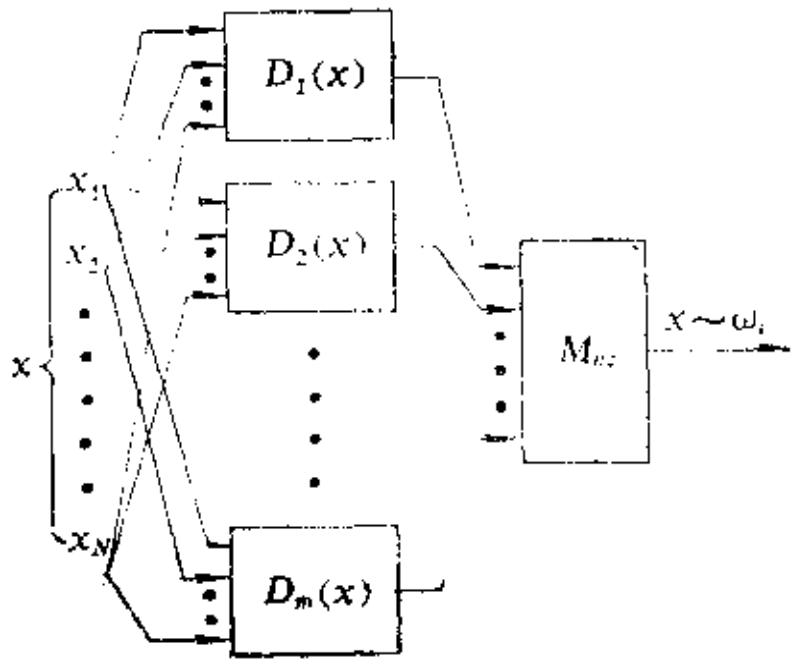
实现 Bayes 决策规则的分类器称为 **Bayes 分类器**。从式 (4.26) 及式 (4.30) 可见，由 Bayes 分类器实现的相应的判别函数 $D_i(X)$可有如下形式：

$$D_i(X) = P(\omega_i|x) \qquad i = 1,\cdots,m \tag{4.33}$$

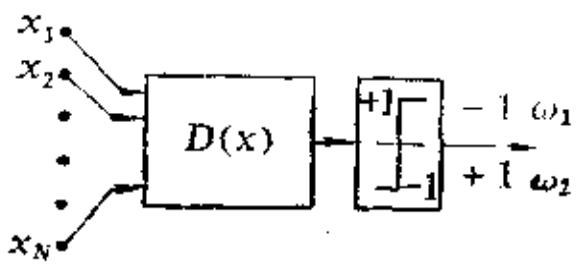
$$D_i(X) = P(x|\omega_i)P(\omega_i) \qquad i = 1,\cdots m, \tag{4.34}$$

$$D_i(X) = \log[P(x|\omega_i)P(\omega_i)] \qquad i = 1,\cdots,m \tag{4.35}$$

Bayes 分类器的基本形式如图 4.7(a) 所示。它由二部分组成：判别函数计算器及最大值选择器。



(a) 多类问题分类器



(b) 二类问题分类器

图 4.7 Bayes 分类器

如果是二类问题，则判别函数 D(X) 适当修改为 $D(X) = D_1(X) - D_2(X)$。此处$D_1(X)$和 $D_2(X)$ 如式 (4.33)～式 (4.35)所示。而决策规则就改为下式，

$$\begin{cases} D(X) > 0 \quad 则 \quad X \sim \omega_1 \\ D(X) < 0 \quad 则 \quad X \sim \omega_2 \end{cases} \tag{4.36}$$

其判别函数 D(X)的具体形式如下：

$$D(X) = P(\omega_1|x) - P(\omega_2|x) \tag{4.37}$$

$$D(X) = P(x|\omega_1)P(\omega_1) - P(x|\omega_2)P(\omega_2) \tag{4.38}$$

$$D(X) = \log\frac{P(x|\omega_1)}{P(x|\omega_2)} + \log\frac{P(\omega_1)}{P(\omega_2)} \tag{4.39}$$

二类问题的 Bayes 分类器如图 4.7(b)所示。

不管判别函数写成何种形式，其决策规则的效果都是把特征空间划分成 m 个决策区域 $R_1, R_2, \cdots R_m$。

如果 $D_i(X) > D_j(X)$对 $j \neq i$ 都成立，

则 X 在 R_i 区域内， 因而就作出决策 $X \sim \omega_i$ (4.40)

而且 R_i 与 R_j 区域之间的边界也可用方程式 $D_i - D_j = 0$ 表示。相应于式(4.33)～式(4.35)的区域 R_i 与 R_j 之间的边界方程如下：

$$P(\omega_i|x) - P(\omega_j|X) = 0 \tag{4.41}$$

$$P(x|\omega_i)P(\omega_i) - P(x|\omega_j)P(\omega_j) = 0 \tag{4.42}$$

$$\log\frac{P(x|\omega_i)}{P(x|\omega_j)} + \log\frac{P(\omega_i)}{P(\omega_j)} = 0 \tag{4.43}$$

图 4.8 示出了决策边界及决策区域的例子。分类器也可看作为把特征空间划分成决策区域的一种装置。

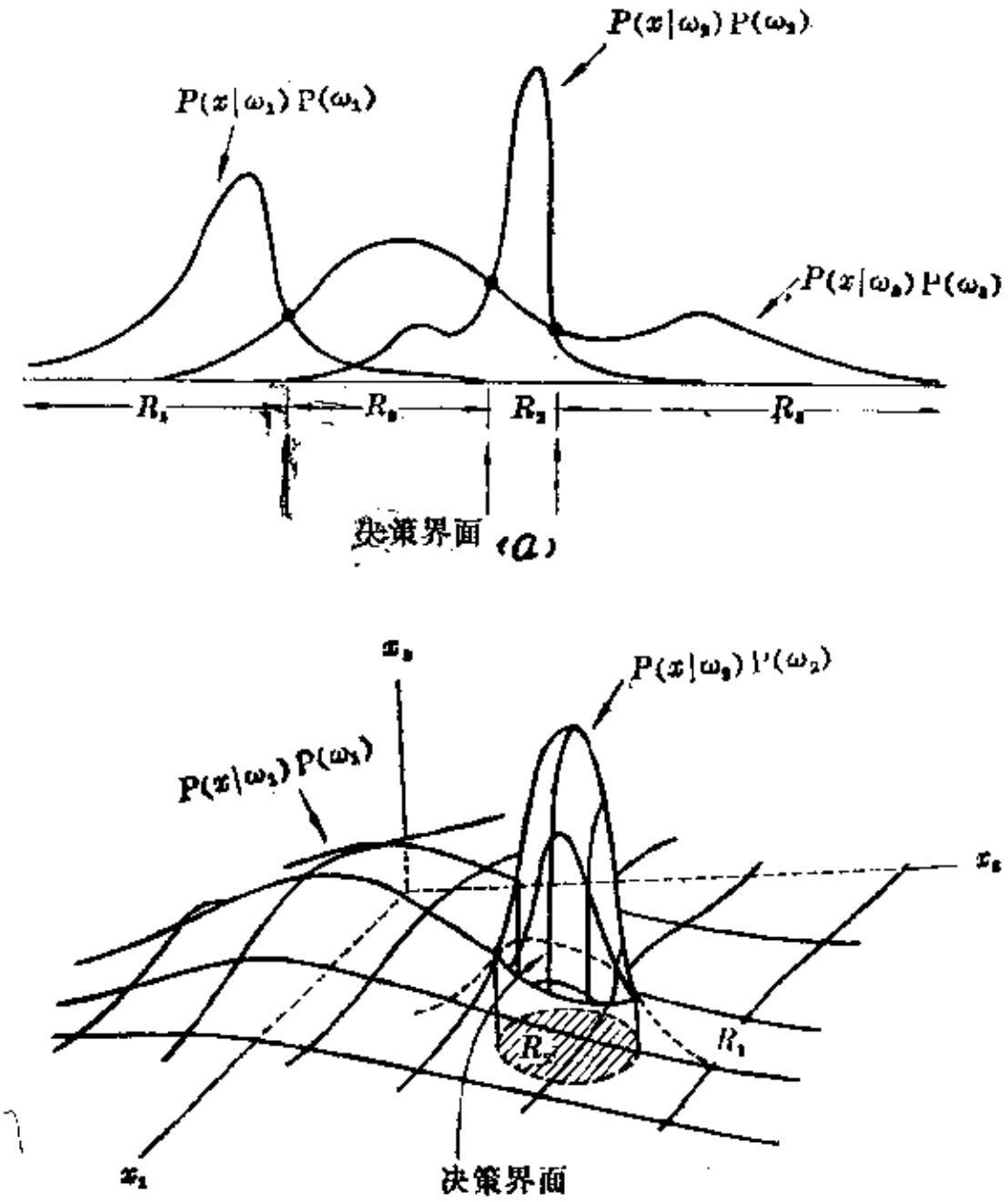


图 4.8　决策边界和决策区域的例子

以上的决策规则是**最小错误概率**的 Bayes 决策。但是不能得到完美的分类。虽然分类划分得正确是有所得益的，但常常由于分类的错误而造成损失更大。因此还应从其反面来考虑：这样的分类，出现错误的概率如何？损失如何？等等。

假定对测得的一个特征矢量 X，作出了决策 $X\sim\omega_i$（用 d_i 简记之）。如果真正的类别应是 ω_j，这就会导致**损失** $L(X\sim\omega_i,\omega_j)$，简记为 L_{ij}。由于 $P(\omega_j|x)$ 是当 x 的真正类别为 ω_j 时的概率，所以同决策 $X\sim\omega_i$ 有关的条件损失为

$$R(X\sim\omega_i|x) = \sum_{j=1}^{m} L(X\sim\omega_i,\omega_j)P(\omega_j|x) \tag{4.44}$$

$R(\mathbf{X}\sim\omega_i|\mathbf{x})$是与决策 $\mathbf{X}\sim\omega_i$ 有关的，故 $R(\mathbf{X}\sim\omega_i|\mathbf{x})$也称为**条件风险**，可简记 为 $R(d_i|\mathbf{x})$。因而对于每个 x，就根据式 (4.44) 计算出关于每个决策 $\mathbf{X}\sim\omega_i$ 的条件 风 险 $R(d_i|\mathbf{x})$。而后选择使$R(a_i|x)$为最小值的决策 $\mathbf{X}\sim\omega_i$，即

$$\text{若 } R(d_i|\mathbf{x})<R(d_j|\mathbf{x}),\ i\neq j \quad \text{则} \quad \mathbf{X}\sim\omega_i \tag{4.45}$$

这样的决策称为**最小风险的 Bayes 决策规则。**

式 (4.44) 中的 $L(d_i,\omega_j)$ 根据具体问题而定。而特别使人感兴趣的一个损失 函 数是 **(0,1) 损失函数**，或称为**对称损失函数**：即对正确的决策，损失为 0；而 对任何不正确的决策，损失为 1 。用下式表示：

$$L(d_i,\omega_j)=\begin{cases}0 & i=j\\ 1 & i\neq j\end{cases}\quad i,\ j=1,\cdots,m. \tag{4.46}$$

式 (4.46) 代入式 (4.44) 得

$$R(d_i|\mathbf{x})=\sum_{j=1}^{m}L(d_i,\omega_j)P(\omega_j|\mathbf{x})=\sum_{j\neq i}P(\omega_j|\mathbf{x})$$

$$=1-P(\omega_i|\mathbf{x}) \tag{4.47}$$

欲使 $R(d_i|\mathbf{x})$最小，则应使 $P(\omega_i|)$最大。因此对于 (0,1) 损失函数，其决策规 则 就 是式 (4.30) 了。

对于二类问题，式 (4.44) 就可写为

$$R(d_1|\mathbf{x})=L_{11}P(\omega_1|\mathbf{x})+L_{12}P(\omega_2|\mathbf{x})$$
$$R(d_2|\mathbf{x})=L_{21}P(\omega_1|\mathbf{x})+L_{22}P(\omega_2|\mathbf{x}) \tag{4.48}$$

再根据式 (4.45) ，则得

$$\text{若}\quad (L_{21}-L_{11})P(\omega_1|\mathbf{x})\ \gtrless\ (L_{12}-L_{22})P(\omega_2|\mathbf{x}),\quad \text{则}\ \begin{matrix}\mathbf{X}\sim\omega_1\\ \mathbf{X}\sim\omega_2\end{matrix} \tag{4.49}$$

由于作出错误决策所引起的损失比作出正确决策的损失要大，所以可合理地假设 $L_{21}-L_{11}$ 和 $L_{12}-L_{22}$ 均大于 0 。再利用概率论的 Bayes 规则，用 $P(\mathbf{x}|\omega_i)P(\omega_i)$ 代 替 $P(\omega_i|\mathbf{x})$，则式 (4.49) 等价为下式

$$\text{若}\quad \frac{P(\mathbf{x}|\omega_1)}{P(\mathbf{x}|\omega_2)}\ \gtrless\ \frac{(L_{12}-L_{22})P(\omega_2)}{(L_{21}-L_{11})P(\omega_1)}\quad \text{则}\ \begin{matrix}\mathbf{X}\sim\omega_1\\ \mathbf{X}\sim\omega_2\end{matrix} \tag{4.50}$$

则二类问题的 Bayes 决策规则也可理解为：若似然比超过 (或低于) 某一阈值(与 X 无关)，则作决策 $\mathbf{X}\sim\omega_1$(或 $\mathbf{X}\sim\omega_2$)。

对于 (0,1) 损失函数而言，式 (4.50) 就变为式 (4.32)了，其中 $m=2$。

如果 $P(\mathbf{x}|\omega_i)$，$i=1,\cdots m$，是多变量正态密度函数，即

$$P(\mathbf{x}|\omega_i)=\frac{1}{(2\pi)^{N/2}|\mathbf{K}_i|^{1/2}}\exp\left[-\frac{1}{2}(\mathbf{X}-\mathbf{M}_i)^T\mathbf{K}_i^{-1}(\mathbf{X}-\mathbf{M}_i)\right]$$
$$i=1,\cdots,m \tag{4.51}$$

其中 $\mathbf{M}_i$ 是 N 维均值矢量，$\mathbf{K}_i$ 是 $N\times N$ 协方差矩阵，$(\mathbf{X}-\mathbf{M}_i)^T$ 是 $(\mathbf{X}-\mathbf{M}_i)$的转置，$\mathbf{K}_i^{-1}$ 是 $\mathbf{K}_i$ 的逆阵，$|\mathbf{K}_i|$是 $\mathbf{K}_i$ 的行列式。应用式 (4.35) ，其判别函数为

$$D_i(\mathbf{X})=-\frac{1}{2}(\mathbf{X}-\mathbf{M}_i)^T\mathbf{K}_i^{-1}(\mathbf{X}-\mathbf{M}_i)-\frac{N}{2}\log 2\pi-\frac{1}{2}\log|\mathbf{K}_i|+\log P(\omega_i)$$

(4.52)

若 $\mathbf{K}_i=\mathbf{K}$，$i=1,\cdots,m$，则式（4.52）中的$(N/2)\log 2\pi$和$(1/2)\log|\mathbf{K}_i|$就与i无关了，故可省略之。如果再考虑到m个先验概率$P(\omega_i)$都相同，则$\log P(\omega_i)$项又可省略了。式（4.52）简化为

$$D_i(\mathbf{X})=-\frac{1}{2}(\mathbf{X}-\mathbf{M}_i)^T\mathbf{K}^{-1}(\mathbf{X}-\mathbf{M}_i) \tag{4.53}$$

此时决策规则就可简单叙述为：为了对一个特征矢量$\mathbf{X}$进行决策分类，只要计算一下它到各类的均值矢量$\mathbf{M}_i$的 Maharanobis 距离平方，$(\mathbf{X}-\mathbf{M}_i)^T\mathbf{K}_i^{-1}(\mathbf{X}-\mathbf{M}_i)$，而后把它归类到最近均值的所在类$i$。

将式（4.53）展开，并注意到$\mathbf{X}^T\mathbf{K}^{-1}\mathbf{X}$同$i$无关，又可省略之，式（4.53）就变成

$$D_i(\mathbf{X})=\mathbf{M}_i^T\mathbf{K}^{-1}\mathbf{X}-\frac{1}{2}\mathbf{M}_i^T\mathbf{K}^{-1}\mathbf{M}_i \tag{4.54}$$

即使m个先验概率$P(\omega_i)$不相同，只是在式(4.54)中再加一个与$\mathbf{X}$无关的常数项$\log P(\omega_i)$而已。由于判别函数是线性的，所以决策边界仍是超平面。

在以上所讨论的 Bayes 决策规则中，对于某一个特征矢量$\mathbf{X}$，都应将它分类至某一类ω_i中。如果出现$D_i(\mathbf{X})=D_j(\mathbf{X})>D_k(\mathbf{X})$（对所有的$k\neq i,j$）时，则可任意地做出决策$\mathbf{X}\sim\omega_i$或$\mathbf{X}\sim\omega_j$。这样的决策，其误识率可能是较高的。就以二类问题为例。图 4.9 中的线 AB 是决策边界。在线 CD 和线 EF 之间的决策，误识率并不理想。而在许多实际问题中，误识所带来的损失常常非常之大，因而没有相当之把握，仍可不识（不对$\mathbf{X}$进行分类）。如

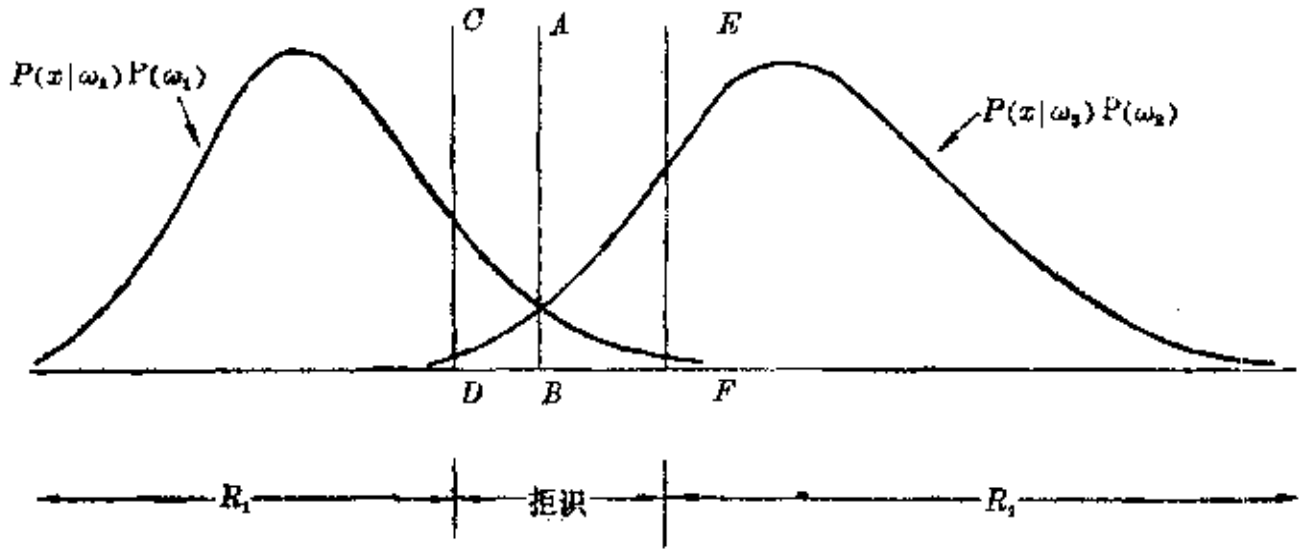


图 4.9　错误概率的组成

图 4.9 中落入 CD 与 EF 之间的区域的$\mathbf{X}$，则不加以分类而拒识。因而模式识别中常有技术指标：**正识率、误识率、拒识率**。例如式（4.32）对于二类问题就可改写为

$$\begin{aligned}&\text{若 } D(X)>A \quad \text{则 } \mathbf{X}\sim\omega_1\\&\text{若 } D(X)<B \quad \text{则 } \mathbf{X}\sim\omega_2\\&\text{若 } B<D(X)<A \quad \text{则拒识}\end{aligned} \tag{4.55}$$

4.2　句法（语言学）方法

在某些模式识别问题中，所考虑的模式通常是很复杂的，而且所需的特征数目常常是很大量的。例如指纹和脸谱辨识问题，红白血球计数，癌细胞识别，连续语音识别，汉字识别问题等等。如果把其每个描述都定义为一个类，那是不实际的。因此，与其用简单的分类

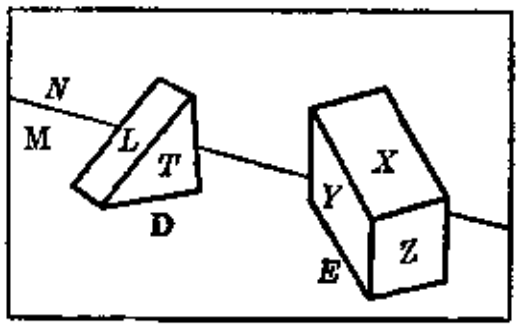
法，不如用另一种描述方法，更能满足识别的需要，即以较小的子模式根据一定的层次组合来描述一个复杂的模式。这就是**句法（语言学）方法**。

4.2.1 句法（语言学）方法的基本思想

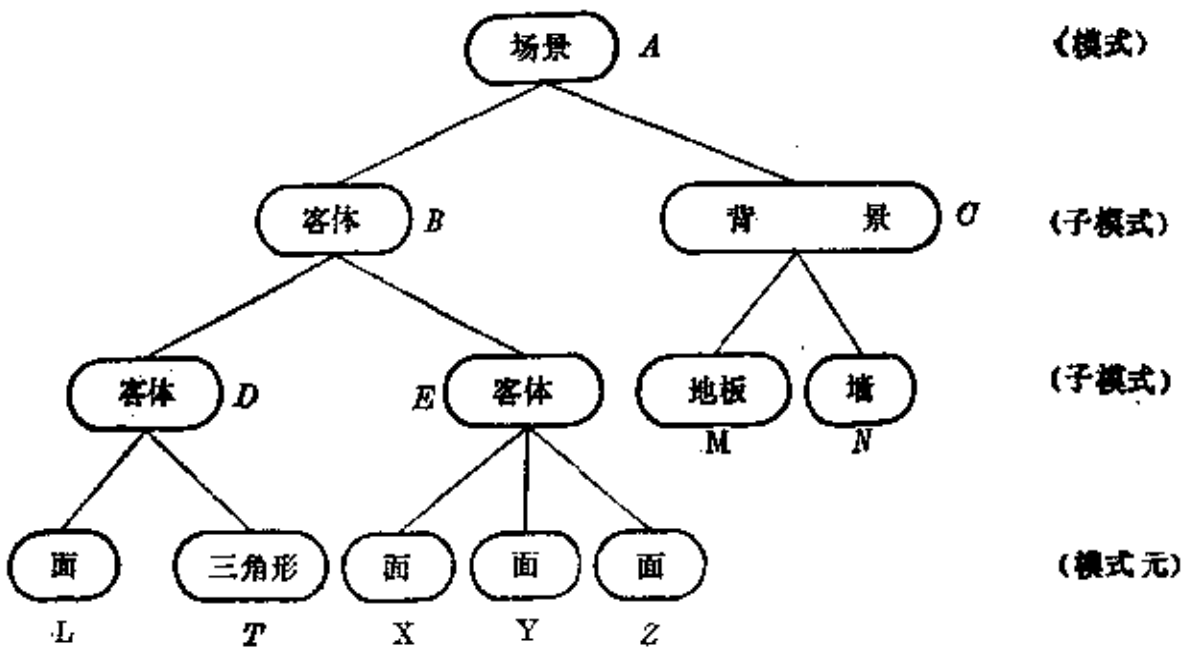
在英语中，短语与句子是由单词根据一定的语法关系组合而成，而单词又是由字母连接而成。**句法模式识别**方法就是模仿语言学中句法的层次结构，故也称为**语言学的模式识别**方法。该方法是将模式用较简单的子模式按照一定规则组合而成（或称**组合操作**）。而较简单的子模式又由更简单的子模式组合而成，……以此类推。其最简单的子模式称为**模式元**，构成一个模式元集合。

采用该方法时，识别模式元显然比识别原来的模式要容易得多。

该方法在描述模式由子模式、子模式又由子模式，……子模式由模式元构成的结构信息中，层次结构是很明显的。因此可由图论中的图或树来描述。其起始节点就是模式本身，而端节点就是模式元。如图 4.10(a) 中的场景就可由图 4.10(b)的树来描述。



(a)



(b)

图 4.10　场景及其树形描述法

在模式元的描述中，由模式元组成子模式、子模式组成模式所遵循的规则，称为**文法**。而**句子**则是根据文法来描述某具体模式的。称句子的集合为**语言**。

因此在对给定模式的每个模式元都被辨识之后，就对描述给定模式的句子进行分析，予以确定它对于给定的文法是否在文法（或句法）方面正确。称该分析为**句法分析**。与此同时，句法分析也产生表示给定模式的句子（通常用树结构形式）的结构描述。

可见句法模式识别系统应当由如下几个主要部分组成（参看图 4.11）：预处理、模式表示和句法分析。

预处理的功能包括模式编码与近似，以及滤波、恢复与增强。一个输入模式首先要用对后面进行处理较方便的某些方式来加以编码或近似。例如，黑白图就可以根据 0 和 1 的栅格（或矩阵）来编码，而波形就可以用它的时间采样或一个截尾的 Fourier 级数展开式来近似。为了使在系统的后一级进行处理时更加有效，在这一级常常采用“数据压缩”。滤波、恢复、增强用来消除噪声、恢复被丢失的东西，增强被编码（或近似）的模式的特质。在预处理器输出的模式大约就具有适当的“好特质”。每个预处理过的模式，则由一个语言结构（如一个模式元串或一个图）来表示。

模式表示的操作则是由模式分割与模式元（特征）抽取两部分组成。为了用其子模式来表示模式，就必须把模式加以分割，同时对模式元及其在模式中的关系加以识别(或抽取)。换句话说，它把预处理过的模式根据预先给定的句法或组合操作，分割为子模式和模式元。而且依序用模式元的一个给定集合来识别每个子模式。到此为止，每个模式已经由具有特定句法操作的模式元集合来表示。有些系统甚至连在模式里的各种句法关系也删除掉。

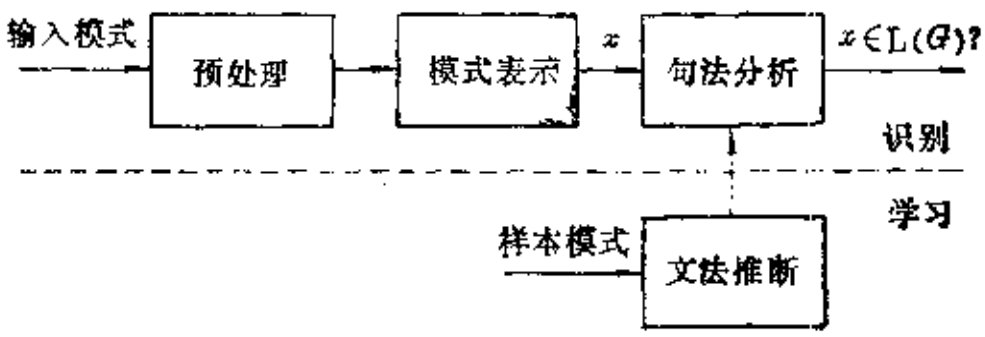


图 4.11　句法模式识别系统的框图

句法分析就是用一个“句法分析器”来决策该模式表示在文法上是否正确，即决策它属于模式（也是由给定的句法或文法来描述）的哪一类。一个模式类就有一种特定的文法及其分析树。当执行句法分析时，一种文法如能接受这种模式时，就把这模式分到这一类。否则，就由另一种文法来分析，从而确定能否接受。都不能接受时，则拒绝这个模式。

如有一个文法推理机那就更好了。它可以从训练模式（也是用文法表示的）的给定集合中推断出一种文法来。这相当于前述的决策论方法中的“学习”过程。

现在我们主要的研究课题就是模式表示及句法分析。即模式元如何分割与抽取，其之间关系如何表示，以及如何分析其文法。

4.2.2　模式元与子模式的表示与选择

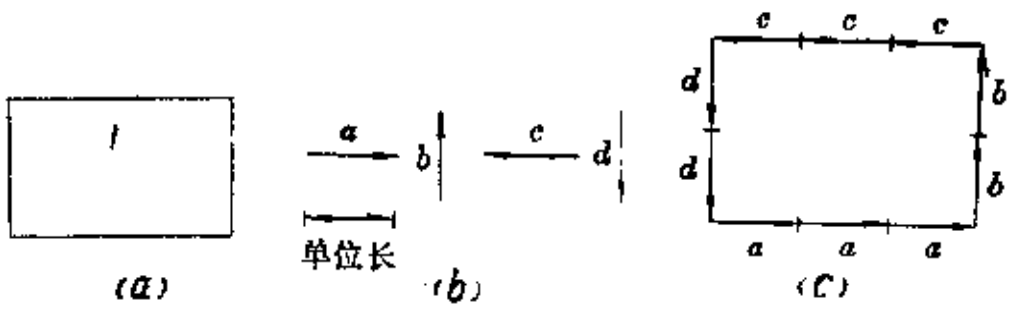
模式应当由哪些模式元（即模式元集）来构成较好？这个模式元集的确定，其影响的因素很多。如数据的性质，特殊应用以及实现该系统的技术有效性等都会有很大影响。因此，模式元的选择问题，没有一个共同的方法。不过，通常有两个要求：

(1) 模式元应是基本的模式单位，应能根据指定的结构关系提供简洁而又足够的模式描述。

(2) 模式元要容易地由现存的非语言学方法来抽取或识别。因为这些模式元被认为是简单且细密的，而且它们的结构信息量是不重要的。

例 4.3　矩形可以用单位矢量来描述，其各种关系或组合操作可用逻辑与/或的数字操作来表示。如图 4.12(a) 中的矩形。可由图 4.12(b) 中的四个模式元来描述，如图 4.12(c) 所示，即用串 $aaabbcccdd$ 来描述该矩形。如果其组合操作规定为仅有“头尾连接”的操作，用“+”表示，如图 4.12(d) 所示。那么，图 4.12(c) 的矩形就可用串 $a+a+a+b$

$+b+c+c+c+d+d$ 来表示，其对应的树形结构则如图 4.12(e) 所示



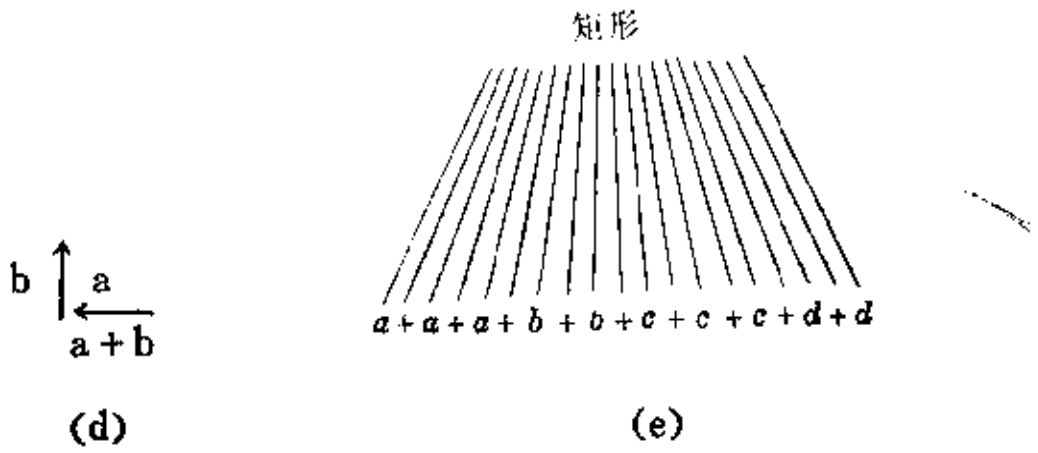


图 4.12　矩形及其模式元和表示法

不同尺寸的矩形集合可由如下语言来表示。

$$L=\{a^n b^m c^n d^m \mid n,\ m=1,2,\cdots\} \tag{4.56}$$

当 $m=n$ 时，即不同尺寸的正方形集合，就由如下语言来示。

$$L=\{a^n b^n c^n d^n \mid n=1,2,\cdots\} \tag{4.57}$$

稍为复杂的例子就如图 4.13 所示。

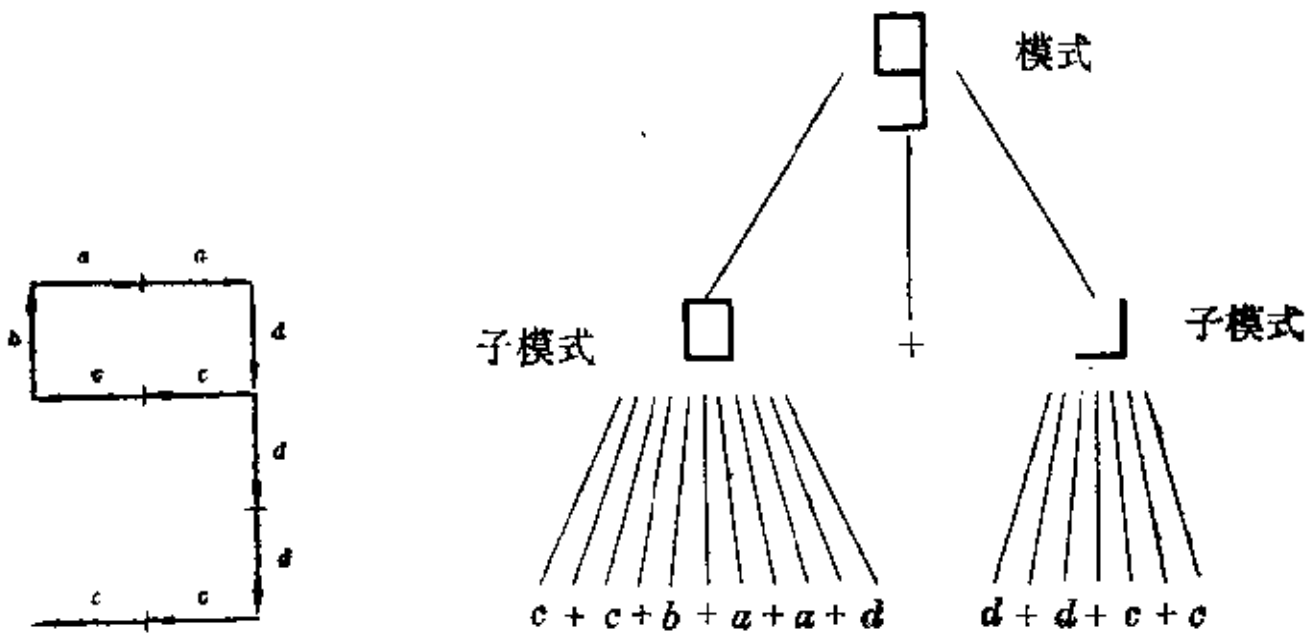


图 4.13　模式“9”和它的结构描述

例 4.4　汉字识别的结构描述。汉字常由偏旁等构成字。因此汉字的模式元集可取一些偏旁、笔划和简单字来组成，而它们之间的结构关系可由简单的分割操作来表示，如图 4.14 所示的结构关系。对于这些操作可以递归地应用，即可以重复用这几种操作之任一种而分割每个子模式。这样就把一个汉字分割成子模式与模式元。例如，“侗”字的结构关系就如图 4.15 所示。如果现成的技术已能识别或抽取出汉字的模式元集合，那么就可用给定的结构关系集合，采用句法方法来描述汉字。而识别其模式元却是较简单的。而且只有几个模式元，由于不同的结构关系，却可组合成许多汉字。

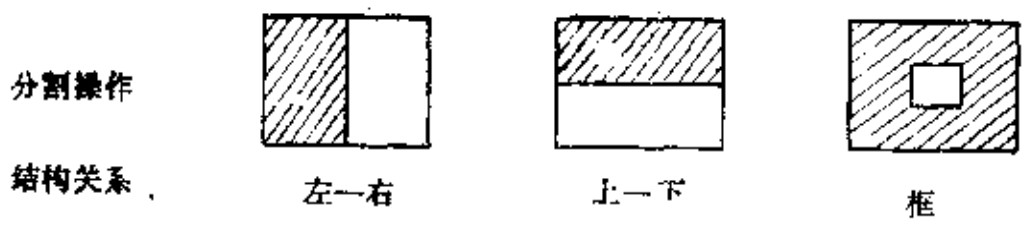


图 4.14　汉字的分割操作和结构关系

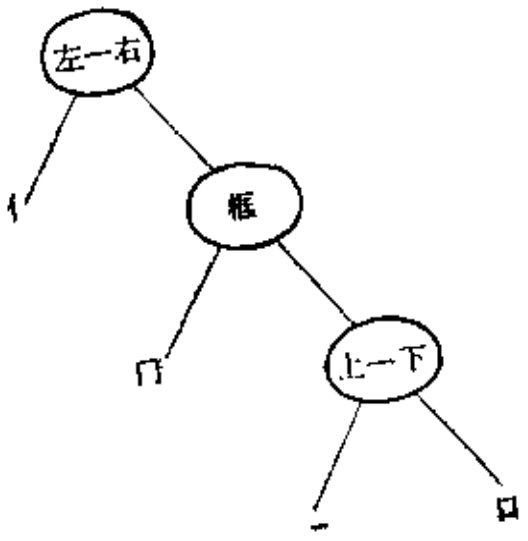


图 4.15　“侗”字的结构关系

例 4.5　英文手写草体的模式元集合可由式（4.58）的三元组形式的线段组成

$$\sigma_j=[(x_{j_1},y_{j_1}),(x_{j_2},y_{j_2}),\theta_j] \quad (4.58)$$

其中 (x_j, y_j) 表示线段端点的近似位置，θ_j 表示从第一端点沿着线段至第二端点的旋转。如果顺时针转，则 θ_j 为正；如果逆时针转，则 θ_j 为负。其中典型的四个模式元示于图 4.16 中。

在图 4.16 中，“棒”、“钩”、“拱”和“环”

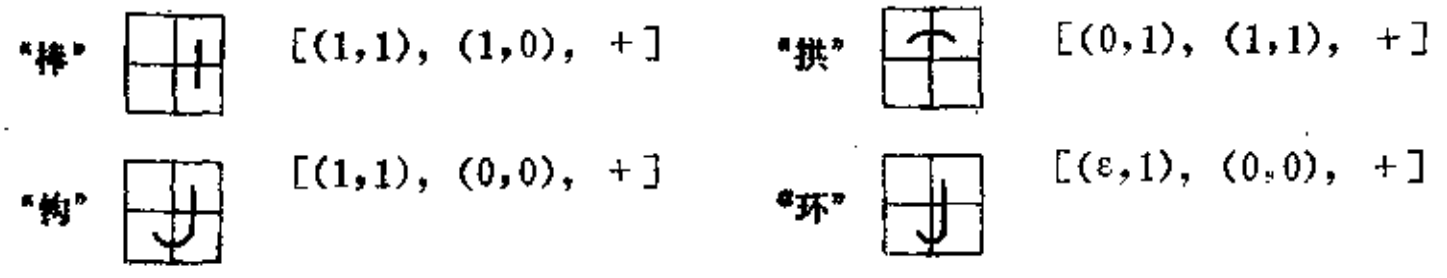


图 4.16　典型的四个模式元

的端点座标 (x_j, y_j) 可根据图 4.17 所示那样来定义。即在 x、y 座标轴上各划分成 0，1 两个线段。由它们所定义的区域 a，b，c 和 d 的 (x_j, y_j) 座标就可分别表示为（0，0），（0，1），（1，0）和（1，1）了。例如，图 4.16 中，“钩”的第一端点座标为(1,1)，而第二端点座标为（0，0），第一端点是按顺时针方向沿着线段转至第二端点的。故其θ_j取为“+”。因而“钩”的 σ_j 表示为 [(1,1), (0,0), +]。为了把“环”与“钩”加以区别，引入了 ε。ε 表示其位置虽然投影在 1 的座标段上，但有偏移，较靠近 0 的座标段。故“环”的 σ_j表示为 [(ε, 1), (0,0), +]，与“钩”是有区别的。

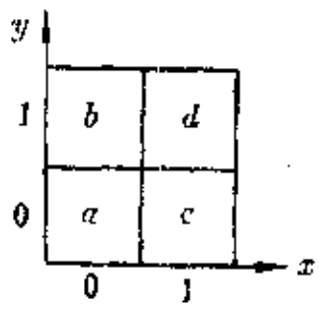
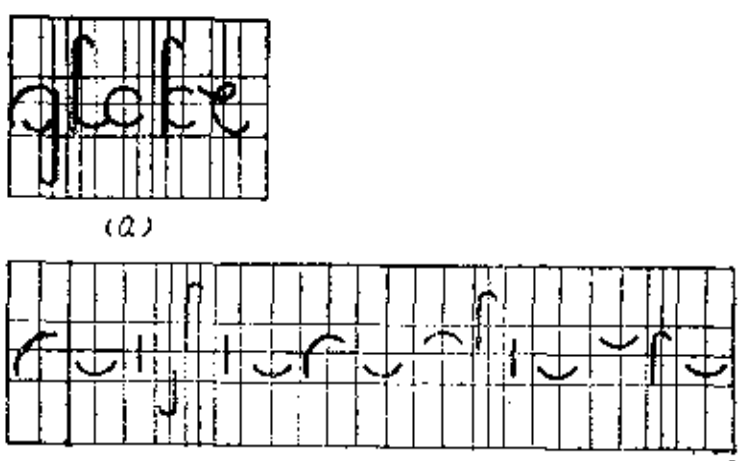


图 4.17　端点的座标值定义

只要对图 4.16 的四个模式元的 (x_j, y_j) 及 θ_j 的不同值进

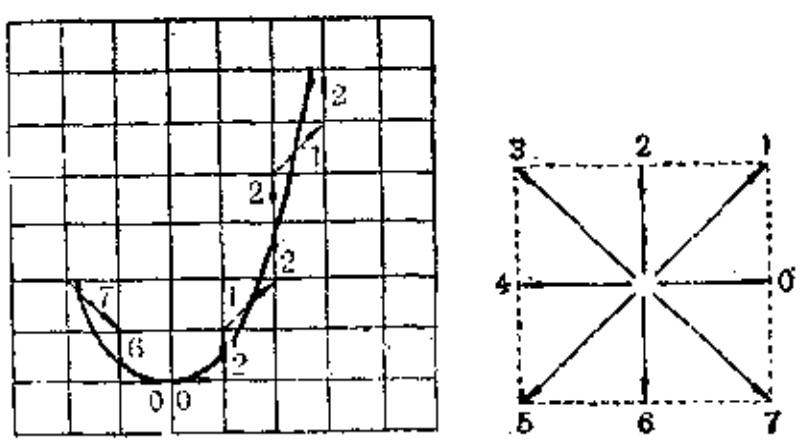
行组合，可得 32 种线段。由于有重复，故只有 28 种线段。但是在共同使用的英语手写草体笔划中较感兴趣的仅有 9 种。即可仅取其中 9 种作为模式元。英文字母就可由这些模式元依序组合来表示。例如，图 4.18(a) 所示的 "*globe*" 手写体，可以由图 4.18(b) 中所示的笔划（模式元）序列来表示。但至今还没有描述手写体的形式语言。



(a)

(b)

图 4.18 "*globe*" 的笔划序列表示法



(a) (b)

图 4.19 曲线的链码

例 4.6 利用链码来描述模式。对于二维模式，例如任意曲线，区域边界线，轮廓线之类，可以在其上先打下矩形网格，参看图 4.19(a)。而后把最接近于模式（即图中所示的粗实线，方向是由左至右）的网格点，就近地依序连接起来，如图中的虚线所示，其中箭头表示其顺序关系，每个虚线矢量其实就是模式元。在此类问题中，其模式元集由图 4.19(b) 中所示的 8 种模式元组成。从中可看出，模式元是这样产生的：网格中在水平轴正方向的矢量，编码为 "0"；而后沿网线逆时针每转 45°，其编码增加 1（模 8 运算）。从而形成八种不同编码的模式元。用这个模式元集，就可以采用链码形式来描述如上所述的任意二维模式。因此图 4.19(*a*) 中的线段就可由链码 7600212212 来表示。如果线段的方向改为由右上至左下，则其链码变为 6566564423。

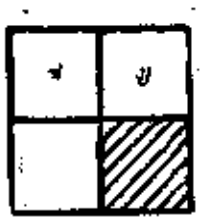


图 4.20 区域边界寻找法

表4.1 边界走向规则

X	Y	下一个边界点
0，1	0	转向右面
0	1	直　　走
1	1	转向左面

对于二值数字图片，如何寻找其区域的边界线或轮廓线呢？同样采用网格法。每个网格就是一个图元。图片在每个网格中，其值为 0 或 1。为了要把 "1" 值网格连成片的区域边界线寻找出来，假设在原先的查寻中，"1" 区域是在已查路线的右面，如图 4.20 所示。其中箭头表示路线的走向，而斜影网格表示 "1" 值区域，X、Y 网格表示查寻到交叉点时所遇到的两个未查新网格。那么可根据 X、Y的值来决定下一个边界点，如表 4.1 所示。其

结果必然是边界以顺时针方式绕“1”值区域包围着，即边界走向的右侧是“1”值区域。而后再另找新区域。以此类推。直到把所有“1”值区域的边界线都查寻完为止。

例 4.7　任意多边形（或任意图形）用它的有限个凸多边形的并集表示，而这些凸多边形又可表示为有限个半平面的交集。

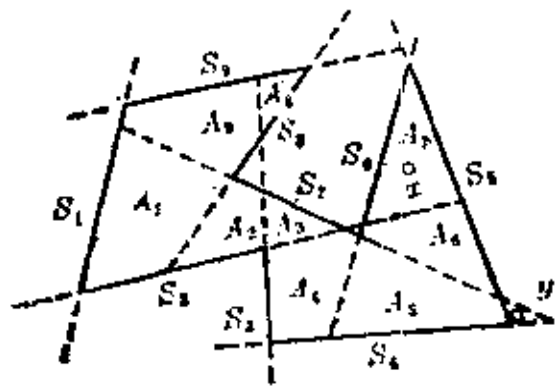


图 4.21　多边形

设 A 是一个闭合多边形，其边为 S_1，S_2，…S_n。平面上的点 x 关于该多边形 A 的某边 S 是正或负的定义如下：该点 x 处于该边 S（及其延长线）的某一侧，而多边形 A 处于该边 S 本身（即不包括延长线）的某一侧。如果两者都是同一侧，则称为**点 x 关于边 S 是正**；如果两者不是同一侧，则称为**点 x 关于边 S 是负**。如图 4.21 所示的多边形 A，点 x 关于 S_6 和 S_8 为正，但关于 S_7 为负。同样，点 y 关于 S_4 和 S_7 为正，但关于 S_5 为负。把 A 的边两方向延长，从而**横截**多边形 A。A 就被分成 A_1，A_2，…，A_9 等 9 个凸多边形。虽然 A 可由这 9 个凸多边形的并集来表示。但这不是所需要的，因为这些凸多边形应尽可能大，而凸多边形的数目应尽可能少。例如 A_1，A_2，A_3 就可并成一个凸多边形。

虽然，关于某一边为正的所有的点可以形成一个以该边为其边的半平面。令 h_i 表示关于边 S_i 的半平面，Q 表示所有半平面的交（必属于 A），如 A 是凸多边形，则 A = Q，如 A 不是凸多边形，则 Q 可以是空集，或纯粹就与 A 不同。

令 Q_I 表示除了 S_{i_1}，…S_{i_k} 外所有半平面的交（其中索引集 $I=\{i_1, i_k\}$），则定义 Q_i 的序列如下：

$$Q=\bigcap_{i=1}^{n} h_i,\qquad Q_j=\bigcap_{\substack{i=1\\ i\neq j}}^{n} h_i,\qquad Q_{jk}=\bigcap_{\substack{j=i\\ i\neq j\ \ i\neq k}}^{n} h_i\cdots\cdots \tag{4.59}$$

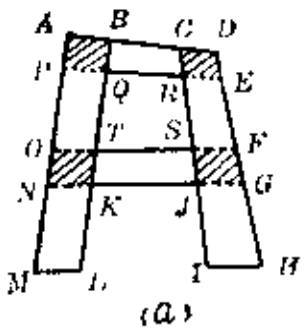
因为 $Q\subset Q_j\subset Q_{jk}\cdots$，所以式（4.59）是一个递增序列，称为**Q 序列**。序列的最后成员就是整个平面。而且由于 j，k，…的不同取值也产生不同的序列。如果各序列中有一个属于 A 的最大成员，则那些最大成员的集合就称为 A 的**本原（凸多边形）子集**。如果 Q 序列中某个属于 A 的非空成员的前面成员都是空集的话，则称该成员为 A 的**核**。这么一来，A 的本原子集的并集恰是 A。

对于一个给定多边形，可以用形成所有序列 Q，Q_j，Q_{jk}…并搜索其最大成员的办法来找出本原子集，而且是唯一的。这样，其并集就表示原多边形 A 了。例如，图 4.21 中多边形的本原子集 = $\{A_1+A_8+A_9,\ A_1+A_2+A_3,\ A_3+A_4+A_5,\ A_5+A_6+A_7\}$。

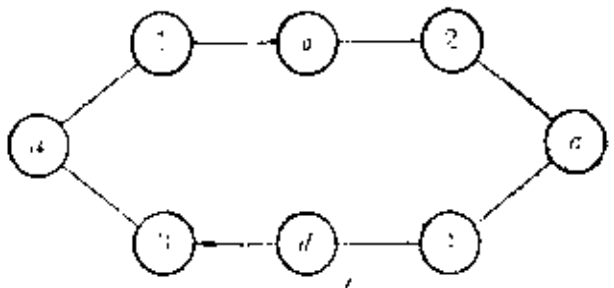
利用多边形还可表示其他多边形形式的图形，如图 4.22 的“A”和“E”。采用图论中的图结构描述法。其节点对应于核与本原子集，其分枝就把每个核连至包含该核的所有本原子集上。

4.2.3　模式文法

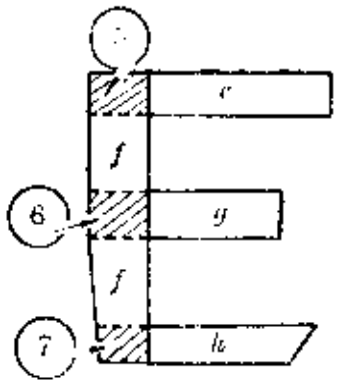
文法 G 在形式上是由四元组(V_N，V_T，P，S)等形式表示的。其中 V_N 是**非终止符**集合（其成员用大写拉丁字母表示），V_T 是**终止符**集合（其成员用小写拉丁字母表示），P 是**生成式**集合，它们都是有限集。S 是**起始符号**。通常，$S\in V_N$，$V_N\cap V_T=\phi$，且定义 $V=V_N\cup$



(a)


(b)

记号	图元或核
a	ABLM
b	ADEP
c	CDHI
d	OFGN
1	ABQP
2	CDER
3	OTKN
4	SFGJ


(c)

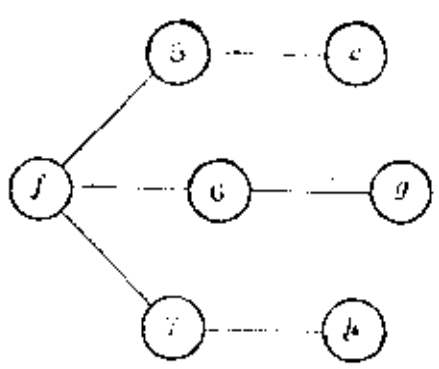

(d)

图 4.22　多边形图与对应的本元

V_T，V^+是由 V 中符号组成的**符号串**（简称为**串**）之集合，V_T^+为终止符组成的符号串集合。$V^* = V^+ \cup \{\phi\}$, $V_T^* = V_T \cup \{\phi\}$, $P = \{\alpha \to \beta \mid \alpha \in V^+, \beta \in V^*\}$。生成式 $\alpha \to \beta$ 的具体含意是：对 V^* 中任意的串 γ、δ，如果$\alpha \to \beta$作用于串 $\gamma\alpha\delta$ 上之后，使该串 $\gamma\alpha\delta$ 得到 $\gamma\beta\delta$。而且称 $\gamma\beta\delta$ 为在文法 G 中 $\gamma\alpha\delta$ 的**直接推导**。用记号 $\gamma\alpha\delta \Rightarrow \gamma\beta\delta$ 表示之，读成“$\gamma\alpha\delta$ 直接推导出 $\gamma\beta\delta$”。生成式 $\alpha \to \beta$ 也可理解为$\gamma\alpha\delta \Rightarrow \gamma\beta\delta$的导出规则。如果 $\alpha_1, \cdots, \alpha_m \in V^*$，而且 $\alpha_1 \Rightarrow \alpha_2, \alpha_2 \Leftarrow \alpha_3, \cdots, \alpha_{m-1} \Rightarrow \alpha_m$，即通过对 P 的某些生成式的应用，能够从 α_1 **推导**出 α_m，则我们认为α_1与 α_m之间有$\alpha_1 \overset{*}{\Rightarrow} \alpha_m$ 的关系。如果α_1 为起始符号 S，且α_m 为终止符串 ω，即 $S \overset{*}{\Rightarrow} \omega$，$\omega \in V_T^+$的话，则称 ω 为**句子**。这就是句子的形式定义。

所谓由 G 所产生的**语言**（记为L(G)），就是能从 S 导出的终止符串的集合，即 $L(G) = \{\omega \mid S \overset{*}{\Rightarrow} \omega, \omega \in V_T^+\}$。也可称 L(G)为 G **可接受的语言**。如果 $L(G_1) = L(G_2)$，则称文法 G_1和 G_2是**等价**的。

由于对 P 的限制不同，就有不同类型的文法。上述的定义称为“**0 型文法**”。如果对 P 中的每个 $\alpha \to \beta$ 再给予限制$|\beta| \geqslant |\alpha|$（即串 β 的符号个数不少于串 α 的符号个数）的话，则该 G 称为“**1 型文法**”或“**上下文有关文法**”。如果对 1 型文法再加限制：α 为单个非终止符及 β 为任意的非空的串，则称该 G 为“**2 型文法**”或“**上下文无关文法**”。如果对 2 型文法中的 $\alpha \to \beta$ 再加限制：β 为 aB 或 a，$a \in V_T$，$B \in V_N$，则称该 G 为“**3 型文法**”或“**正则文法**”或“**有限状态文法**”。这四种类型文法所产生的语言分别称为**0 型语言，上下文有关语言，上下文无关语言及正则语言。**

有了一种文法就会产生一类语言。

例 4.8　3 型文法 $G=(V_N, V_T, P, S)$，其中 $V_N=\{S, A_1, A_2, B_{10}, B_{20}, B_{30}, B_{21}, B_{31}, B_{32}, C_1, C_2, C_3\}$，$V_T=\{a, b, c\}$。P 由下列生成式组成：

(1) $S\to aA_1$　(2) $S\to aB_{10}$　(3) $A_1\to aA_2$

(4) $A_1\to aB_{20}$　(5) $A_2\to aB_{30}$　(6) $B_{10}\to bC_1$

(7) $B_{20}\to bB_{21}$　(8) $B_{21}\to bC_2$　(9) $B_{30}\to bB_{31}$

(10) $B_{31}\to bB_{32}$　(11) $B_{32}\to bC_3$　(12) $C_1\to c$

(13) $C_2\to cC_1$　(14) $C_3\to cC_2$

根据该文法由如下的三种导出过程可以分别产生语言 $L_1(G)=\{abc\}$，$L_2(G)=\{a^2b^2c^2\}$，$L_3(G)=\{a^3b^3c^3\}$。下列过程中的记号 "$\overset{(N)}{\Rightarrow}$" 表示 "根据第 (N) 个生成式而导出"。

$$S\overset{(2)}{\Rightarrow}aB_{10}\overset{(6)}{\Rightarrow}abC_1\overset{(12)}{\Rightarrow}abc;$$

$$S\overset{(1)}{\Rightarrow}aA_1\overset{(4)}{\Rightarrow}aaB_{20}\overset{(7)}{\Rightarrow}aabB_{21}\overset{(8)}{\Rightarrow}aabbC_2\overset{(13)}{\Rightarrow}aabbcC_1\overset{(12)}{\Rightarrow}aabbcc;$$

$$S\overset{(1)}{\Rightarrow}aA_1\overset{(3)}{\Rightarrow}aaA_2\overset{(5)}{\Rightarrow}aaaB_{30}\overset{(9)}{\Rightarrow}aaabB_{31}\overset{(10)}{\Rightarrow}aaabbB_{32}$$
$$\overset{(11)}{\Rightarrow}aaabbbC_3\overset{(14)}{\Rightarrow}aaabbbcC_2\overset{(13)}{\Rightarrow}aaabbbccC_1\overset{(12)}{\Rightarrow}aaabbbccc。$$

综上所述，该种文法可以产生语言 $L(G)=\{a^nb^nc^n\mid 1\leqslant n\leqslant 3\}$。这可用来描述边长为1，2，3单位长的等边三角形的语言。由上述三种推导过程，可以用其派生树来形象化地描述，如图 4.23 所示。

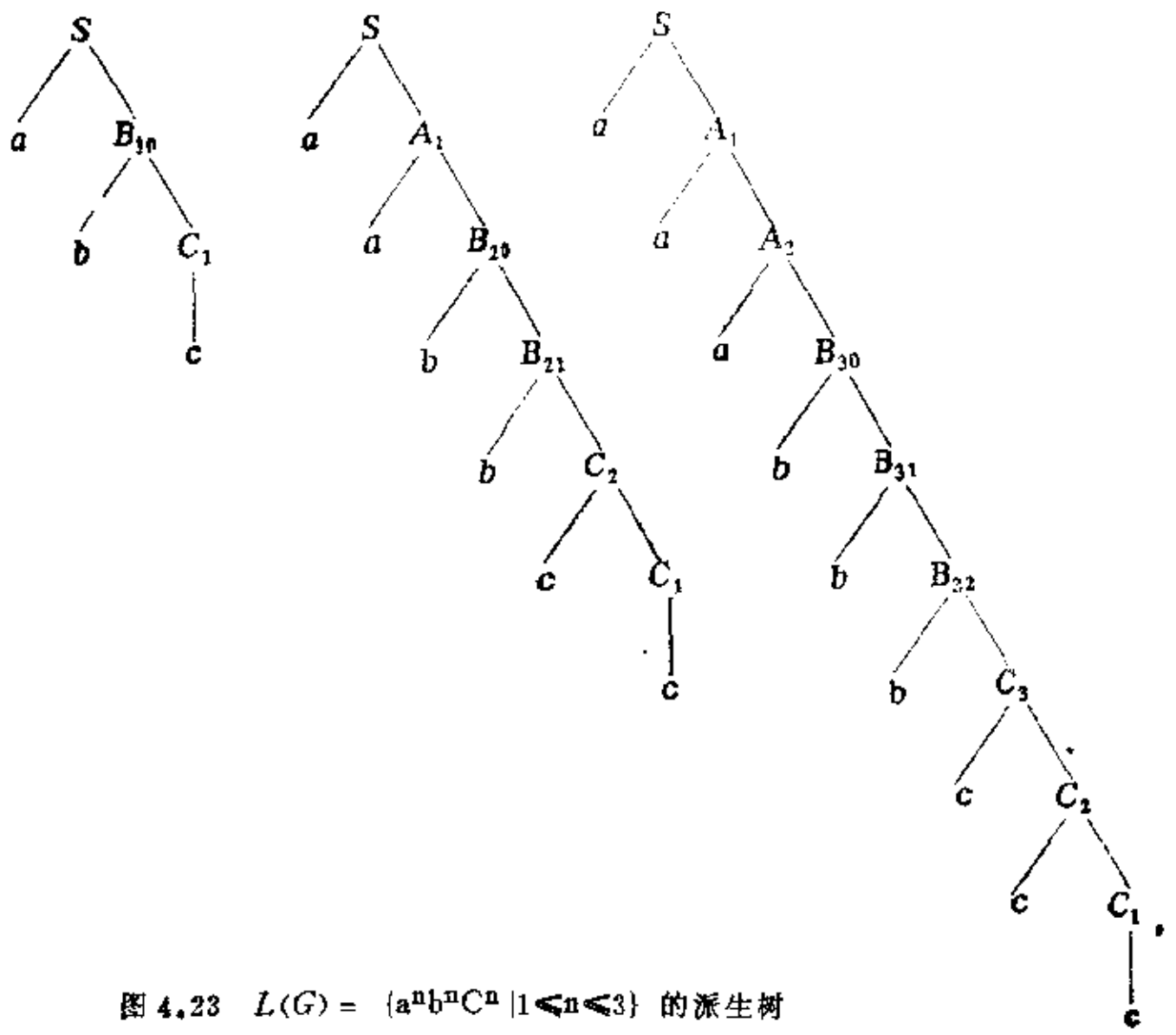


图 4.23　$L(G)=\{a^nb^nC^n\mid 1\leqslant n\leqslant 3\}$ 的派生树

实际上，凡是 2 型和 3 型文法都有其派生树。通常，**派生树**可这样下定义：设 $G=(V_V, V_T, P, S)$ 为 2 型文法，满足下列五个条件的树，称为 G 的**派生树**，其中第五条是前四条的必然结果。

(1) 每个节点都有一个属于V的符号做为其标记；

(2) 树根的标记为 S；

(3) 非端节点的标记一定属于 V_N，端节点的标记一定属于 V_T。

(4) 如果节点 α 的子节点，从左到右的次序是子节点 $\beta_1, \beta_2, \cdots \beta_k$，则 $\alpha \to \beta_1\beta_2\cdots\beta_k$ 一定是 P 中的一个生成式。

(5) 如果树的所有端节点自左到右的次序为 $\beta_1, \beta_2, \cdots, \beta_K$，则串 $\beta_1\beta_2\cdots\beta_K$ 恰恰是由 G 所产生的语言 L(G)。

语言往往可由文法来描述。

例 4.9 能接受语言 $L=\{a^nb^nc^n \mid n\geqslant 1\}$（可理解为描述边长为 1, 2, …$n$ 单位长的等边三角形）的文法有 1 型文法 $G=(V_N, V_T, P, S)$。其中 $V_N=\{S, B\}$，$V_T=\{a, b, c\}$，P由下列生成式组成：

(1) $S\to aSBc$　(2) $S\to abc$　(3) $cB\to Bc$　(4) $bB\to bb$

例 4.10 能接受语言 $L=\{a^nb^mc^nd^m \mid n, m\geqslant 1\}$（它可描述矩形图形）的文法有 0 型文法 $G=(V_N, V_T, P, S)$。其中，$V_N=\{S, A, B, C, D, E\}$，$V_T=\{a, b, c, d\}$。P 由下列生成式组成：

(1) $S\to aAC$　(2) $A\to aAC$　(3) $A\to bBD$

(4) $B\to bBD$　(5) $B\to E$　(6) $ED\to DE$

(7) $DEC\to EcD$　(8) $DEc\to cED$　(9) $bEc\to bcE$

(10) $DE\phi\to Ed$　(11) $DEd\to Edd$　(12) $cEc\to ccE$

(13) $cEd\to cd$

描述某类具体语言的文法并不是唯一的，可以有许多文法来描述它。例 4.11 就说明了这点。

例 4.11 例 4.8 中的语言 $L=\{a^nb^nc^n \mid 1\leqslant n\leqslant 3\}$ 还可以用 2 型文法 $G=(V_N, V_T, P, S)$ 来描述。其中 $V_N=\{S, A_1, A_2, B_1, B_2, B_3, C\}$，$V_T=\{a, b, c\}$。P 由下列生成式组成：

(1) $S\to aA_1C$　(2) $A_1\to b$　(3) $A_1\to aB_2C$

(4) $A_1\to aA_2C$　(5) $A_2\to aB_3C$　(6) $B_3\to bB_2$

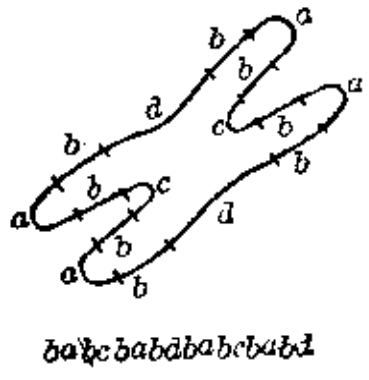
(7) $B_2\to bB_1$　(8) $B_1\to b$　(9) $C\to c$

模式文法的应用是相当广的。在此仅再举一个描述染色体的例子。

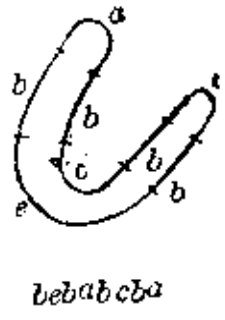
例 4.12 用上下文无关文法来描述染色体。

染色体有三类。第一类称为中央着丝点染色体；第二类称为亚中央着丝点染色体；第三类称为端部着丝点染色体。Ledly, Rotolo, Golab, Jacobson, Ginsburg 和 Wilson 等人于 1965 年提出了用 2 型文法来描述染色体的思想。为尊重原著起见，在此把染色体分为如图 4.24(a) 和图 4.24(b) 所示的亚中央着丝点染色体(submedian chromosome) 和端部着丝

点染色体 (telocentric chromosome) 二类。这二类染色体都可由如图 4.25 所示的图元组合而成。为方便起见，在此我们分别将这些图元定名为 a，b，c，d 和 e，如图中所示。



babcbabdbabcba



bebabcba

图 4.24　染色体

图 4.25　染色体的图元集合

那么，描述染色体的文法 G 就可由四元组组成。

$G = (V_N, V_T, P, \{\langle$亚中央着丝点染色体$\rangle, \langle$端部着丝点染色体$\rangle\})$

其中，$V_N = \{\langle$亚中央着丝点染色体$\rangle, \langle$端部着丝点染色体$\rangle, \langle$臂偶$\rangle, \langle$左部分$\rangle, \langle$右部分$\rangle, \langle$臂$\rangle, \langle$边$\rangle, \langle$底$\rangle\}$

$V_T = \{a, b, c, d, e\}$

P：

〈亚中央着丝点染色体〉→〈臂偶〉〈臂偶〉

〈端部着丝点染色体〉→〈底〉〈臂偶〉

〈臂偶〉→〈边〉〈臂偶〉

〈臂偶〉→〈臂偶〉〈边〉

〈臂偶〉→〈臂〉〈右部分〉

〈臂偶〉→〈左部分〉〈臂〉

〈左部分〉→〈臂〉c

〈右部分〉→c〈臂〉

〈底〉→b〈底〉

〈底〉→〈底〉b

〈底〉→e

〈边〉→b〈边〉

〈边〉→〈边〉b

〈边〉→b

〈边〉→d

〈臂〉→b〈臂〉

〈臂〉→〈臂〉b

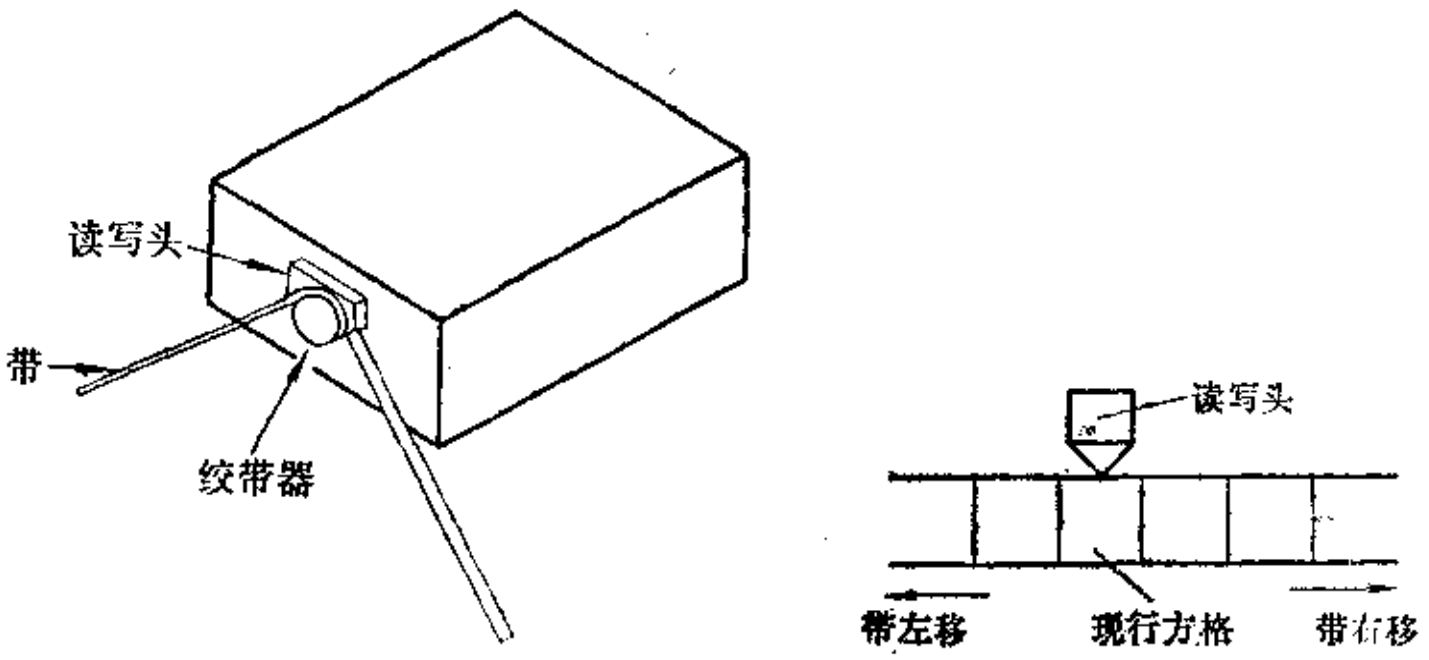
〈臂〉→a

根据这个文法，图 4.24 所示的二个具体的染色体就可分别描述成符号串 *babcbabdbabcba bd* 和 *bebabcba*。

4.2.4　自动机

综上所述可知，根据给定文法而产生的语言，实际上就是一个串。对于模式识别而言，描述不同类型的模式，就会有不同的语言及其对应的文法。因而欲对某输入模式进行分类的

话，就是对描述该模式的语言，用不同的文法试探一下能否接受。何类文法能接受，该输入模式就属于对应这类文法的类。现在的问题是用什么识别器来识别对应于同类文法的语言。**自动机**就是这种识别器。



(a) (b)

图 4.26 *Turing* 机及其带

在现代计算机发展之前，Turing 早在 1936 年就提出了自动机的模型。称这个模型为 **Turing 机**，如图 4.26(a) 所示。它有三个基本部分：控制部件，读写头及无限长的**存贮带**（**简称带**）。带是能够存贮信息、读出信息和改变信息的外部介质。把带分成一个个方格。如图 4.26(b) 所示。在控制部件的操作下，读写头能够而且只能够在它接近的一个方格（简称它为**现行方格**）上读出该格原来“印”的符号（包括没有任何符号的空白）。在读出一次之后，在控制部件的操作下，读写头可能在现行方格上写上新的符号或空白（即清除）。还可能使带左移一格，使读写头位于现行方格的右边方格（该方格就成为现行方格，而原来的现行方格已不成为现行方格了）上。或者停机。

有时为了叙述的方便，常把带的左移一格或右移一格说成是“读写头前进一格”或“读写头后退一格”。

甚至还可设想带上的方格不仅是一道信息道，还可以是二道、三道等多道信息道。Turing 机的功能是相当完备的。因而，Turing 机可以识别 0 型文法所产生的语言。但完备的功能也带来了算法的复杂性。为了方便起见，对于其他三种文法所产生的语言，可以不用 Turing 机来识别。通常采用稍简单的线性界限自动机、下推自动机和有限状态自动机来分别识别 1 型文法、2 型文法和 3 型文法所产生的语言。

现在就由低级到高级地介绍一下四种自动机的基本意义，以及如何由已知的文法来构造自动机的方法。

如果读者原来已掌握编译技术的基本知识的话，那么就很容易把这些自动机在计算机中加以实现了

有限状态自动机

有限状态自动机是一个五元组 (I, Q, , δ, q_0, F)。其中 I 是输入符号的有限集合，

Q是个有限的非空的状态集合，δ是Q×I至Q的一种映照，$q_0 \in Q$是起始状态，$F \subseteq Q$是终止状态集合。

对于这个形式定义，可以这样理解：有一个输入带，带上“印”有输入符号串。有限状态自动机附有一个现在状态寄存器，开始时，该寄存器存放q_0，并且从带最左端开始，一个个符号自左至右依序地加以读入。根据现在状态q（$\in Q$）及现行读入的符号a（$\in I$），自动机A就选择q_1，q_2，……，q_k（$\in Q$）的任一个作为下一个状态（即取之置换掉现在状态寄存器的内容），并控制带读入头前进一格。用$\delta(q, a) = \{q_1, \cdots, q_k\}$表示。也可称之为“自动机A对任何$i$进入状态$q_i$”。这些就是形式定义中映照δ的具体含意。可见，$\delta(q, a_1a_2) = \delta(q_i, a_2)$，$q_i \in \delta(q, a_1)$。如果读完最后一个符号时，现在状态恰好是F的一个成员，则称自动机A可接受带上所描述的串。用$T(A) = \{x \mid p \in \delta(q_0, x)$且$p \in F\}$表示。如果$\delta(q, a) = \{q_1, \cdots, q_k\}$中$k \neq 1$，即下一状态并不唯一确定，则称A为**不确定的有限状态自动机**；$k = 1$时，则称为**确定的有限状态自动机**。从理论上可以证明：凡是由不确定的有限状态自动机所能接受的语言，一定可构造出一个确定的有限状态自动机，它也能接受该语言。因此除非特殊需要外，我们就不加以区分地统称为**有限状态自动机**。

对于一个正则文法$G = (V_N, V_T, P, S)$，按照如下方法，可构造出一个有限状态自动机$A = (I, Q, \delta, q_0, F)$，使得该自动机A所接受的串（即语言）与文法G所产生的语言（即串）完全等价（记为$T(A) = L(G)$）。构造方法如下：

(1) $I = V_T$

(2) $Q = V_N \cup \{T\}$（T为附加状态）

(3) $q_0 = S$

(4) 如果P含有生成式$S \to \phi$（ϕ表示空白），则$F = \{S, T\}$；否则，$F = \{T\}$。

(5) 凡是P中含有生成式$B \to a$（$B \in V_N$，$a \in V_T$），则相应地定义$\delta(B, a) = \{T\}$

(6) 凡是P中含有生成式$B \to aC$（$C \in V_N$），则相应地定义$\delta(B, a) = \{C\}$

(7) 再附加定义$\delta(T, a) = \phi$（$a \in V_T$）

由后三项就定义了δ的集合。

例4.12　从例4.8中的正则文法，可构成如下的有限状态自动机$A = (I, Q, \delta, q_0, F)$，使得$T(A) = L(G)$。其中，$I = \{a, b, c\}$

$Q = \{S, A_1, A_2, B_{10}, B_{20}, B_{30}, B_{21}, B_{31}, B_{32}, C_1, C_2, C_3, T\}$

$q_0 = S$

$F = \{T\}$

$\delta(S, a) = \{A_1, B_{10}\}$　　$\delta(A_1, a) = \{A_2, B_{20}\}$

$\delta(A_2, a) = \{B_{30}\}$　　$\delta(B_{10}, b) = \{C_1\}$

$\delta(B_{20}, b) = \{B_{21}\}$　　$\delta(B_{21}, b) = \{C_2\}$

$\delta(B_{30}, b) = \{B_{31}\}$　　$\delta(B_{31}, b) = \{B_{32}\}$

$\delta(B_{32}, b) = \{C_3\}$　　$\delta(C_1, c) = \{T\}$

$\delta(C_2, c) = \{C_1\}$　　$\delta(C_3, c) = \{C_2\}$

$\delta(T, a) = \delta(T, b) = \delta(T, c) = \phi$

为了更形象地看出有限状态自动机的状态变化，可以画出其状态传递图。状态传递图是

采用图论中的有向耗散图的。把耗散图的所有节点都与有限状态自动机的状态集Q的所有成员一一对应起来。图中连接节点的边都与该自动机的映照δ相对应起来。而边的耗散值是状态转变的条件（即该自动机的输入符号集I的成员）。就构成了**状态传递图**。例如，对于任意的$\delta(B, a)=\{C_1, C_2, C_3\}$，则在有向耗散图中，应当有三条有向边从节点B分别指向至节点C_1，C_2，C_3。如图4.27所示。而三条边的耗散值为$a(\in I)$。用这种方法就可将该例的状态传递图作出，如图4.28所示。从该传递图可看出，对于语言$L=\{ab^2c\}$，该自动机是拒绝接受的。因为从S状态开始，串abbc输入后，状态是无法传递到终止状态T的。参看图中的路1，当串abbc读至第三个符号b时，自动机的状态已无法从C_1状态传递出去了。而对于$L=\{a^2b^2c^2\}$，却是可接受的。从图中的路2可看出，状态可传递到终止状态$T(\in F)$。

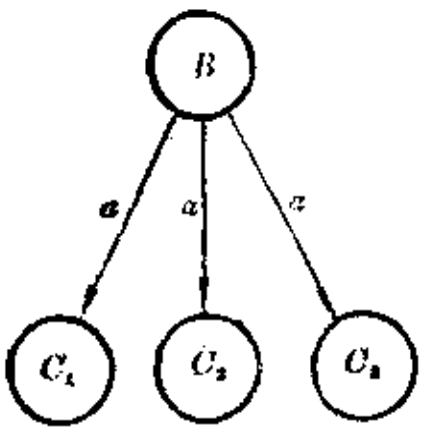


图4.27　状态传递表示法

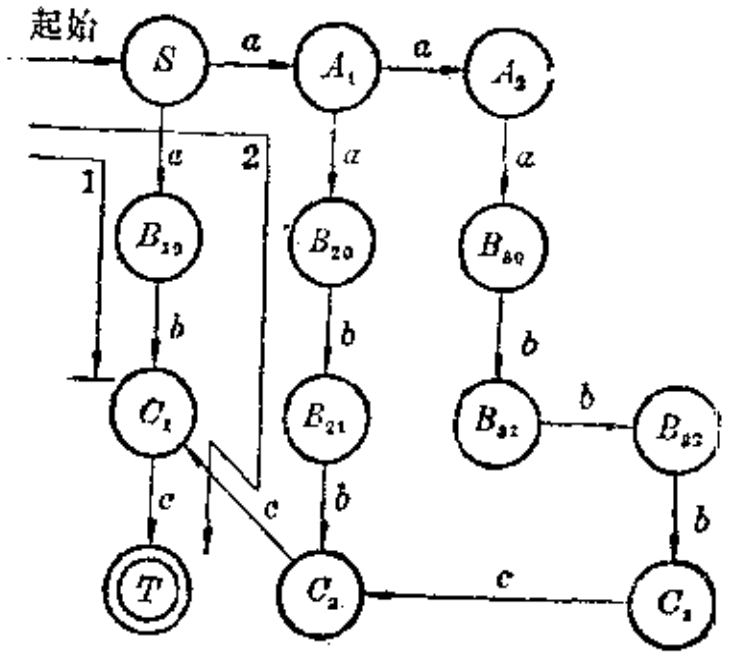


图4.28　状态传递图

同理，给定了一个有限状态自动机$A=(I, Q, \delta, q_0, F)$，也可构成有限状态文法，使得$L(G)=T(G)$。

下推自动机

能识别由上下文无关文法所接受的语言之**下推自动机**是一个七元组$A=(I, Q, \Gamma, \delta, q_0, Z_0, F)$。其中，I是输入符号的有限集。Q是状态的一个有限集。Γ是压入下推表的符号的一个有限集合。δ是一个从$Q\times(I\cup\{\phi\})\times\Gamma$到$Q\times\Gamma^+$（$\Gamma^+$是由Γ中符号组成的串集合）的有限子集的映照。$q_0\in Q$是起始状态。$Z_0$表示一开始就压入下推表Γ的符号。$F\subseteq Q$是终止状态集合。

δ的具体意义如下：

(1) 对于映照$\delta(q, a, Z)=\{(p_1, r_1), \cdots, (p_m, r_m)\}$的解释是：当输入符号为$a$以及下推表的栈顶符号为Z时，处于状态$q$的下推自动机能够对任何$i$进入状态$p_i(\in Q)$，从Γ中取走Z而压入$r_i$，并使带的读入头前进一格。在此$r_i$是串，其最右符先压入Γ中，依序地，最左符后压入Γ中。如r_i为空串，则对Γ不压入符号。

(2) 对于映照$\delta(q, \phi, Z)=\{(p_1, r_1), \cdots, (p_m, r_m)\}$的解释是：当栈顶符号为Z时，处于$q$状态的下推自动机不依赖于所读入的符号，而对任何$i$进入状态$p_i$，并从Γ中

取走 Z 而压入 r_i。在这种情况下，带的读入头不前进。

输入带上的语言能否被下推自动机 A 所接受，有两种完全等价的定义。一种是与有限状态自动机相类似的；另一种是用“空下推表”来定义。即在读带过程中，如果下推表已空（栈顶符号为Z_0），则表示可接受。称之为**空栈接受**。空栈接受的下推自动机就不必再定义终止状态集 F 了。

欲从一个上下文无关文法 $G=(V_N, V_T, P, S)$ 来构造一个空栈接受的下推自动机 A 的话，只需令 $A=(V_T, \{q\}, V_N, \delta, q, S, \phi)$ 即可。其中 δ 的具体内容是：只要 P 中有 $B\to ar_i(i=1, \cdots, m)$，则 δ 就有 $\delta(q, a, B)=\{(q, r_1), \cdots, (q, r_m)\}$。

例 4.13　从例 4.11 中的 2 型文法可构造出下推自动机 $A=(\{a, b, c\}, \{q\}, \{S, A_1, A_2, B_1, B_2, B_3, C\}, \delta, q, S, \phi)$。其中，δ 由下列组成：

$\delta(q, a, S)=(q, A_1C)$　　$\delta(q, b, A_1)=(q, \phi)$

$\delta(q, a, A_1)=\{(q, B_2C), (q, A_2C)\}$　　$\delta(q, a, A_2)=(q, B_3C)$

$\delta(q, b, B_3)=(q, B_2)$　　$\delta(q, b, B_2)=(q, B_1)$

$\delta(q, b, B_1)=(q, \phi)$　　$\delta(q, c, C)=(q, \phi)$

线性界限自动机

线性界限自动机能识别由上下文有关文法所接受的语言。它是一个六元组 $(I, Q, \Gamma, \delta, q_0, F)$。其中，I 是输入符号集。Q 是状态的一个有限集。$F\subseteq Q$ 是终止状态集。Γ 是允许“印”在带上的符号(包括空白符号 B)的有限集，δ 是一个从 $Q\times\Gamma$ 到 $Q\times(\Gamma-\{B\})\times\{L, R\}$的映照（其中 L 及 R 分别是读写头左、右移），$q_0\in Q$是起始状态，$F\subseteq Q$ 是终止状态集。

该自动机规定：带上输入符号串的左右端加有两个特定符号 # 和 *，表示左右端标记。读写头可在带上某格读出符号，也可写入符号（包括空白符号），因而可更改带上的符号。因此在处理过程中，该机可在带上的 * 后面增加长度为 $(a-1)n$ 格的符号串。其中 n 为输入符号串的长度，而 a 为常数，代表该自动机的性能。为方便起见，可看成长度为 n 的带上共有 a 道。δ 的意义是根据现行状态 q_i 与带上读入的符号 A_i 而决定进入新状态 q_j，将该符号改写为 A_j，并且使读写头左移或右移一格。

从上下文有关文法而构造一个线性界限自动机的方法相当多，仅举一例说明之。

例 4.14　从例 4.9 中的 1 型文法可构造出线性界限自动机。其中 $a=2$，带有 2 道。第 1 道为输入符号串，如图 4.29(a) 所示。第 2 道用于记录派生符号串。每道都有 # 和 * 的左右端符号。而且将起始符号 S 安排在道 2 的最左单元。该自动机就按照例 4.9 中的生成式 P 完成如下各步动作：

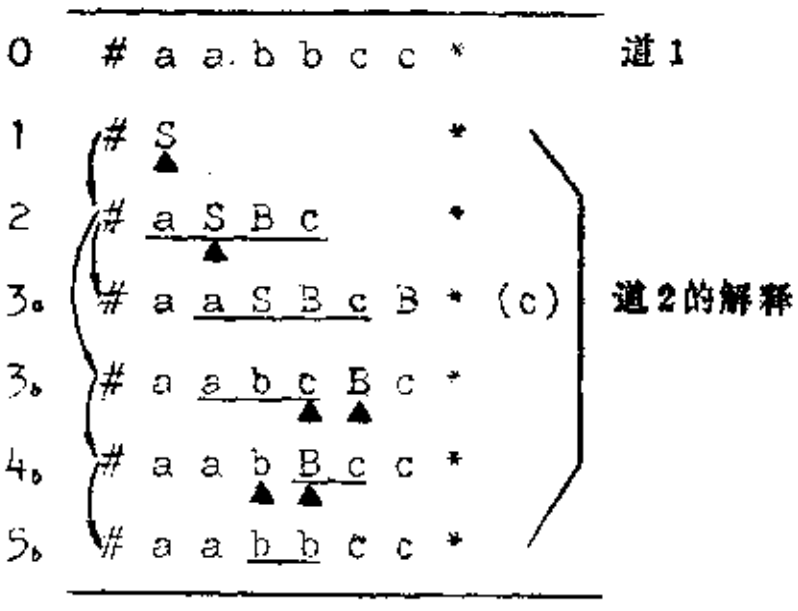
(1) 从道 2 中选择连续的符号子串 α（如图 4.29(b)所示的“▲”部分），使得 α 是 P 中生成式 $\alpha\to\beta$ 的 α 部分。

(2) 在道 2 中用子串 β（如图中的划线部分）代替原先的子串 α。如果 $|\beta|>|\alpha|$，则将道 2 上子串 α 右边的符号串都相应右推 $(|\beta|-|\alpha|)$ 格。

图中的箭号表示道 2 上符号串的改变过程。

(3) 如果右推的符号与 * 重迭，则失败而退出（如图中的 3_a 就是一例）；如果道 2 与道 1 的串完全相同，则成功而退出，表示可接受道 1 的串（如图中的 5_b），否则，就反复(1)

(2) 步骤，一直到对各生成式的各种可能组合都寻找完毕之后，而仍然不成功，就停机，表示拒绝接受道 1 的串。



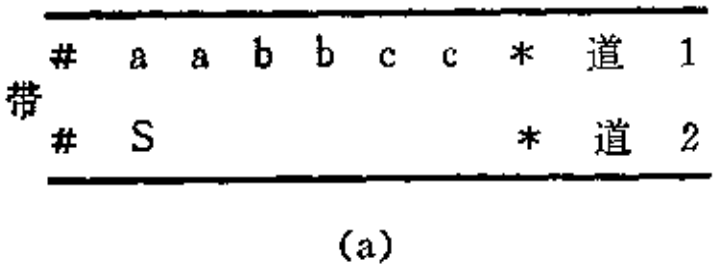


图 4.29 线性界限自动机的带

Turing 机

Turing 机能接受 0 型文法的语言。它的形式定义与线性界限自动机一样。但 Turing 机的带是单向（或双向）无穷带。两者的δ意义也很类似。从两者的实例中是不难看出 的。在此不赘述。

从 0 型文法构成 Turing 机的方法更是灵活。也仅举一例。

例 4.15 从例 4.10 描述的矩形语言$L=\{a^1b^2c^1d^2\}$，可构造出 Turing 机。也是采 用 2 道带。但道 2 没有右端符号，只有左端符号#，且将起始符号S安放在# 之后面。该 Turing 机依生成式P完成如下各步动作：

(1) 从道 2 中选择连续的符号子串 α，使得该 α 是 P 中α→β 的 α部分。

(2) 在道 2 中用子串 β 代替原先的子 串 α。根据 $|\beta| \gtrless |\alpha|$ 的情况，将子串 α 的右边符号串都相应依序地右推或左 推 $||\beta| - |\alpha||$ 格。注意，线性界限自动机是不会出现左推的。

(3) 如果道 2 与道 1 的串完全相同，则成功而退出，表示可接受道 1 的串；否则，就反复如上步骤，一直到对各生成式的各种可能组合都寻找完毕之后，而仍然不成功，就停机，表示拒绝接受道 1 的串。

参看图 4.30 中对$L=\{ab^2cd^2\}$ 的 解 释 过程，就不难理解了。

在此所介绍的文法及自动机的有关知识，不仅适用于模式识别，而且适用于一般的形式

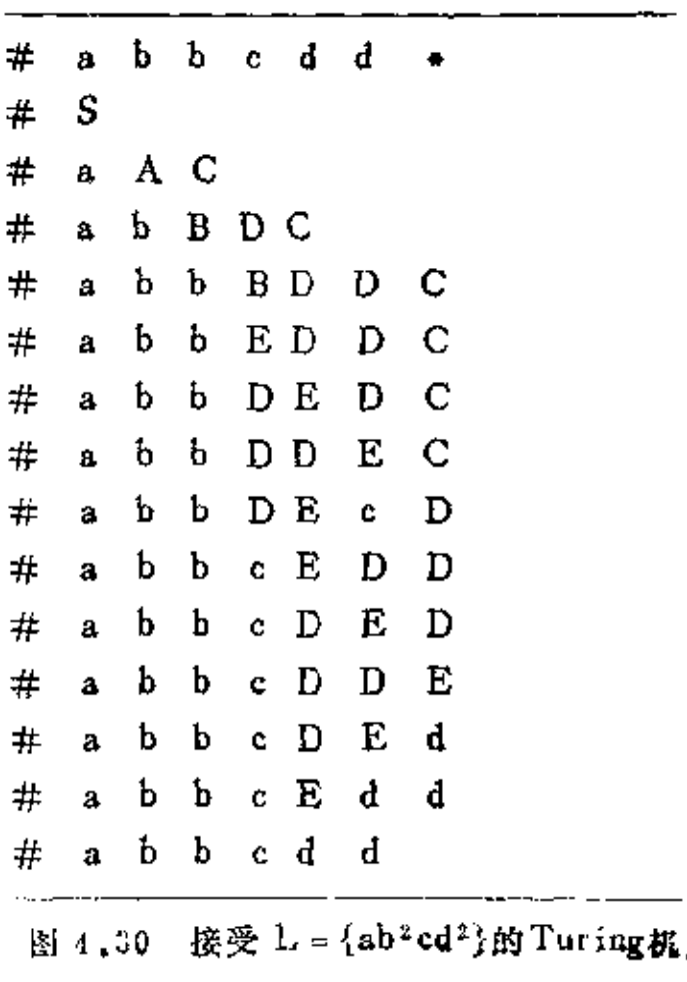


图 4.30 接受 $L=\{ab^2cd^2\}$的Turing机的带

语言与自动机方面。本应专列一章而加以介绍，由于篇幅所限，就仅介绍一些基本方法。至于自动机如何在计算机中实现，这牵涉编译技术问题。请读者再参考有关书籍。

第四章　习　　题

一、已知一个二类（ω_1和ω_2）的线性分类器，其权矢量分别为：

$$W_1=\begin{bmatrix}5\\4\\8\\2\\3\end{bmatrix}\qquad W_2=\begin{bmatrix}1\\7\\4\\3\\6\end{bmatrix}$$

试对如下的输入特征矢量进行分类：

(1) $X=\begin{bmatrix}1\\0\\1\\1\end{bmatrix}$　(2) $X=\begin{bmatrix}1\\1\\1\\1\end{bmatrix}$　(3) $X=\begin{bmatrix}3\\1\\2\\2\end{bmatrix}$

(4) $X=\begin{bmatrix}4\\3\\2\\2\end{bmatrix}$　(5) $X=\begin{bmatrix}1\\3\\2\\2\end{bmatrix}$　(6) $X=\begin{bmatrix}0\\1\\0\\1\end{bmatrix}$

二、求出上题的决策边界。

三、已知一个二类（ω_1，ω_2）的线性分类器，其权矢量为：

$$W=\begin{bmatrix}3\\-5\\6\\-4\\2\end{bmatrix}$$

若对于$D(X)>0$，$X\sim\omega_1$；对于$D(X)<0$，$X\sim\omega_2$，试对如下输入特征矢量进行分类：

(1) $X=\begin{bmatrix}2\\-2\\0\\1\end{bmatrix}$　(2) $X=\begin{bmatrix}1\\-2\\1\\0\end{bmatrix}$　(3) $X=\begin{bmatrix}-2\\-1\\1\\-1\end{bmatrix}$

(4) $X=\begin{bmatrix}-1\\-1\\1\\-2\end{bmatrix}$　(5) $X=\begin{bmatrix}2\\0\\1\\-1\end{bmatrix}$　(6) $X=\begin{bmatrix}0\\2\\-1\\1\end{bmatrix}$

四、根据式(4.7)及式(4.8)，已知两个训练集合T_1'和T_2'分别为：

$$T_1' = \left\{ \begin{bmatrix} 1 \\ 2 \\ -1 \\ 1 \end{bmatrix}, \begin{bmatrix} -1 \\ 1 \\ 1 \\ 2 \end{bmatrix}, \begin{bmatrix} -1 \\ 0 \\ 1 \\ -1 \end{bmatrix} \right\}$$

$$T_2' = \left\{ \begin{bmatrix} -2 \\ -2 \\ 1 \\ 0 \end{bmatrix}, \begin{bmatrix} 0 \\ 1 \\ -2 \\ 1 \end{bmatrix}, \begin{bmatrix} 1 \\ -1 \\ 0 \\ -2 \end{bmatrix} \right\}$$

试对线性分类器进行训练，找出其权矢量。

五、设分别对应于模式类$\omega_i (i=1,2,3,4)$的四个参考矢量 $\mathbf{R}_i (i=1,2,3,4)$为：

$$\mathbf{R}_1 = \begin{bmatrix} -1 \\ -1 \\ 2 \\ -2 \end{bmatrix} \qquad \mathbf{R}_2 = \begin{bmatrix} 0 \\ 1 \\ -1 \\ 0 \end{bmatrix}$$

$$\mathbf{R}_3 = \begin{bmatrix} 2 \\ 0 \\ 1 \\ 0 \end{bmatrix} \qquad \mathbf{R}_4 = \begin{bmatrix} 1 \\ 1 \\ 0 \\ 2 \end{bmatrix}$$

试用最小距离分类法对如下的输入特征矢量进行分类：

(1) $\mathbf{X} = \begin{bmatrix} 0 \\ 0 \\ 0 \\ 0 \end{bmatrix}$ (2) $\mathbf{X} = \begin{bmatrix} 1 \\ 1 \\ 1 \\ 1 \end{bmatrix}$ (3) $\mathbf{X} = \begin{bmatrix} -1 \\ -1 \\ -1 \\ -1 \end{bmatrix}$

(4) $\mathbf{X} = \begin{bmatrix} -2 \\ -2 \\ -2 \\ -2 \end{bmatrix}$ (5) $\mathbf{X} = \begin{bmatrix} 2 \\ 2 \\ 2 \\ 2 \end{bmatrix}$ (6) $\mathbf{X} = \begin{bmatrix} 0 \\ 1 \\ 0 \\ 1 \end{bmatrix}$

六、如果上题中的 $\mathbf{R}_1$ 和 $\mathbf{R}_2$ 是关于模式类 ω_1 的参考集合，而 $\mathbf{R}_3$ 和 $\mathbf{R}_4$ 是关于模式类 ω_2 的参考集合。试用近邻法对上题中的输入特征矢量进行分类。

七、分别对几千个成年男子和女子进行了体高的调查，所得的统计数据如下表：

百分比 / 高度 / 性别	<1.3米	1.3米～1.5米	1.5米～1.6米	1.6米～1.7米	1.7米～1.9米	>1.9米
男	0.001	0.009	0.224	0.613	0.133	0.020
女	0.030	0.181	0.554	0.221	0.013	0.001

设在社会人口统计中男女人数比例约为：男子占 49 %，女子占 51 %。现对几个人测得的体高分别为 1.25 米，1.41 米，1.67米，1.75 米和 2.1 米。应将他（她）们分别判别为男

性或女性？

八、在二类（ω_1 和 ω_2）问题中，已知它们的先验概率相等，而条件密度函数为正态分布：

$$P(x|\omega_1) = (b\sqrt{2\pi})^{-1} \quad exp\left[-\frac{(x-a_1)^2}{2b^2}\right]$$

$$P(x|\omega_2) = (b\sqrt{2\pi})^{-1} \quad exp\left[-\frac{(x-a_2)^2}{2b^2}\right]$$

试求出该二类问题的判别函数 D(x)。

九、在某地区的地震预报工作中，在发出地震警报之后，如果确实发生地震，则损失0.1亿元；如果并不发生地震，则损失 12 亿元。在不发出地震警报的情况下，如果实际上发生了地震，则损失达 100 亿元；如果确实不发生地震，则没有任何损失。现在根据所有获得的情报资料的分析结果，如果发生地震的可能性分别为 11%和 10%，试问，对这二种情况要否发出地震警报？

十、试描述“模”、“式”、“识”、“别”四个汉字的结构关系。

十一、试将图 4.31 所示的曲线用链码描述出来。

十二、下列文法是何类型文法？写出它们所生成的语言。

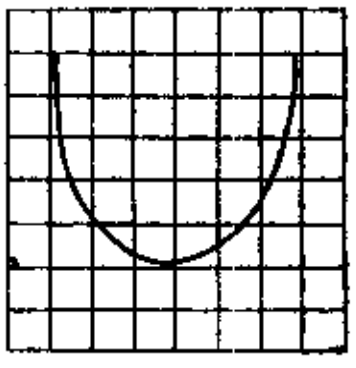


图 4.31 曲线

(1) $G=(V_N, V_T, P, S)$, $V_N=\{S, A\}$, $V_T=\{a,b\}$

P: $S\to aS$, $S\to bA$, $A\to bA$, $A\to aS$, $S\to a$, $S\to b$。

(2) $G=(V_N, V_T, P, S)$, $V_N=\{S,A,B\}$, $V_T=\{a,b\}$

P: $S\to bA$, $A\to bB$, $B\to bB$, $B\to aS$, $B\to a$。

(3) $G=(V_N, V_T, P, S)$, $V_N=\{S,A,B\}$, $V_T=\{a,b,c\}$

P: $S\to cSc$, $S\to AB$, $A\to aaA$, $B\to bbB$, $A\to a$, $B\to b$。

(4) $G=(V_N, V_T, P, S)$, $V_N=\{S\}$, $V_T=\{a,b\}$

P: $S\to aSb$, $S\to ab$

十三、设 G 的生成式 P 为：$S\to AB$, $S\to Ba$, $A\to aS$, $A\to a$, $B\to BS$, $B\to b$, $B\to bA$。试问，它属于何型文法？而 *ababba* 属于 L (G) 吗？若属于，试画出其派生树。其派生树是唯一的吗？

十四、对于上述二题，构造出相应的自动机。

十五、G 的生成式如下：

P: $S\to aSBC$, $S\to aBC$, $CBC\to BCC$, $aB\to ab$, $bB\to bb$, $bC\to bc$, $cC\to cc$。试问，G 属于何型文法？能接受何类语言？试构造其相应的自动机。

十六、足球场地管理员在潮湿天气（即当天下过雨）时无事可做；在干燥天气（当天不下雨）时通常他都要推着剪草机将场地的草剪掉；但是一旦连续干燥到三天以上（包括第三天）时，他不剪草而要给草地洒水一次。试将他的工作状况用状态传递图表示出来。写出其文法，并构造其相应的自动机。

第五章　智 能 机 器 人*

智能机器人是机器人发展的高级阶段。但是，关于智能机器人的定义，国际上还没有统一。据国际自动控制联合大会的一份报告称，智能机器人是能够独立决策和适应性地行动的机器人。日本工业机器人学会一九七六年在制订的工业机器人用语草案中规定：智能机器人是一种能够根据感觉和识别机能而自行决定动作的机器人。不管如何，智能机器人的课题应当是把智能模拟的许多技术综合在一起，建立一个“人”的模型。

本章并不把所有的综合技术都加以介绍，也不详细介绍机器人的具体结构。只着重介绍某些技术，包括神经元、记忆、机器视觉等等。

5.1　概　　述

美国 Stanford 研究所（简称 SRI）的 Nillson 认为智能机器人主要有如下四种软件：

(1) 机器人能独立地发现一种方法，以完成人给予的任务。（问题求解）

(2) 机器人用眼、耳和触觉等感觉器官来了解周围的状况。（感觉，模式识别）

(3) 能记忆并灵活运用诸如机器人工作环境的状态、规则以及自己的经验等知识。（知识的表达）

(4) 能理解并会讲英语、日语等语言。（自然语言处理）

这些并不是独立的课题，而是有相互密切联系的。图 5.1 示出了智能机器人的软件系统。

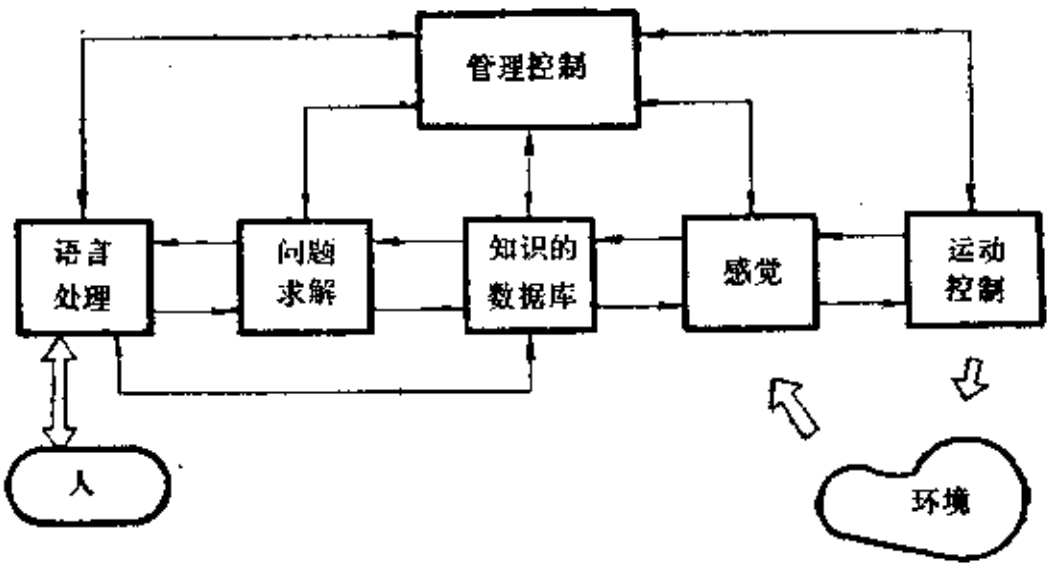


图 5.1　智能机器人的软件系统

智能机器人的硬件系统并不一致。根据不同的研究目的及应用而异。机器人的外形并不一定象人，甚至于有四只脚或六只脚的。目前，国际上也有单位在研究与人的外形相似

---

* 本章主要参考文献为“参考文献”3、12、13、14及17。

的机器人。比较有代表性的智能机器人的硬件系统如图 5.2 所示。把它的各部分与人的器官对比一下：

脑——中央计算机。

眼——电视摄象机，预处理计算机。

手——多自由度的机械手、接触与力检测元件、控制计算机。

足——车，接触、接近检测元件，控制计算机。

耳——话筒，预处理计算机。

口——喇叭，发声用计算机。

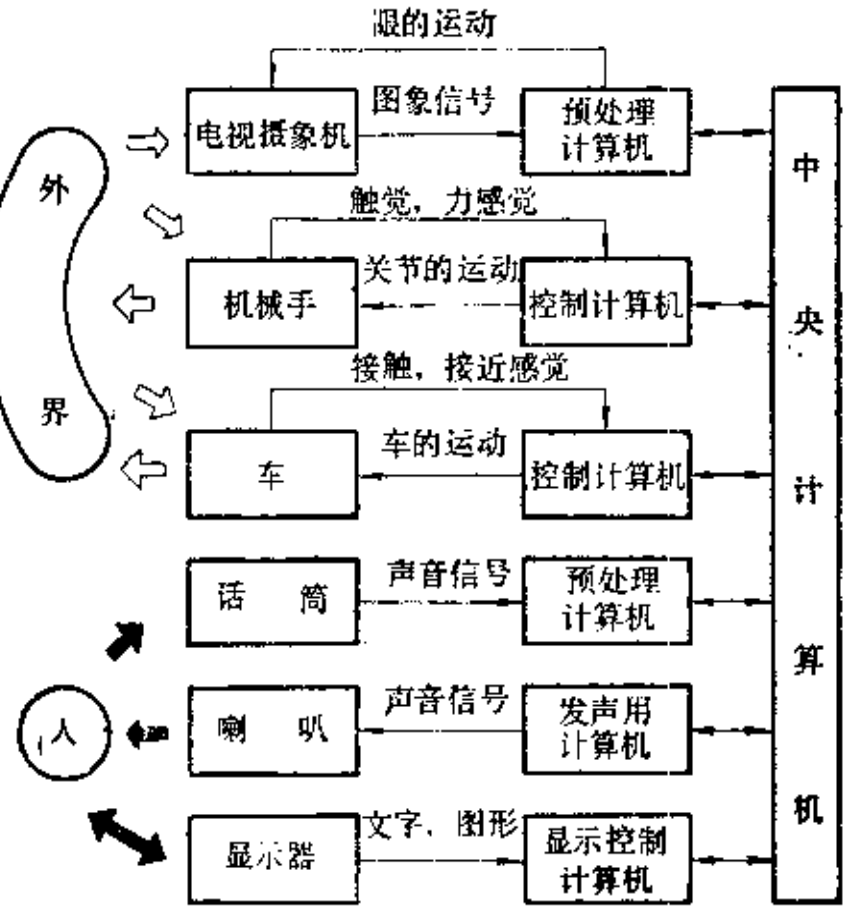


图 5.2　智能机器人的硬件系统

当然，人的各个器官的功能是综合性的。就以手为例，既有触感、握物、提物等能力，还有冷热感等。再以脚为例，除了皮肤的触感及冷热感外，在行走过程中还要保持整个身体的平衡，以免跌跤。这是一个难题，虽然有人在研究步行机构，但是通常用简单的机构（车）来代替。在图 5.2 的系统中，五官中还缺少鼻子。实际上，有时嗅觉也是很重要的。

图 5.3 是 SRI 提出的智能机器人的结构。

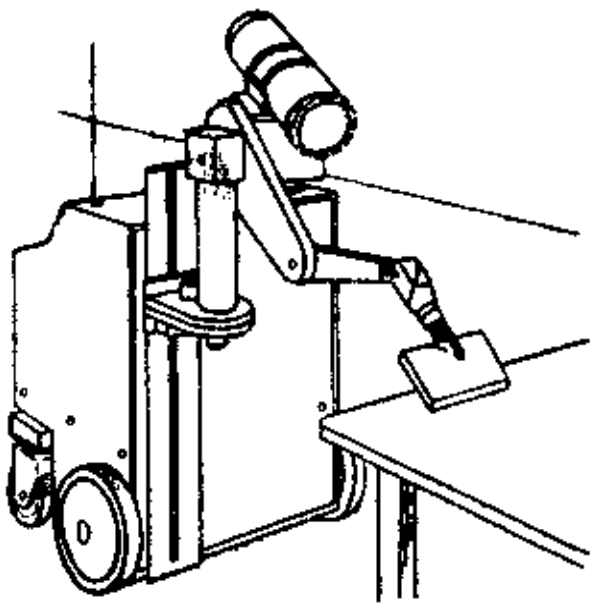


图 5.3　智能机器人的一例（SRT）

眼　机器人的眼，通常采用电视扫瞄摄象机，或飞点管扫瞄输入装置，将图象（景物或文字等）信号输入，经预处理之后送入中央计算机。根据需要还可调节摄象机的位置或焦距。

手　除了选出模仿人的多关节的手的机械手，且对机械手进行多自由度控制外，为了控制机械手握物时的握力，有的还设有触觉、力感觉装置。图 5.4 示出了日本谷江等提出的具有触感和压感的测定原理。图 5.4(a) 表示刚握上物体的状态。物体的宽度为 α，触觉部分的触角长度为 t_1，t_2，刚握上时触角受力使平板内装着的通/断传感器接通，此时板间距离为 x_1，x_2，夹片的开口量为 l。再用力握物体时，如图 5.4(b)所示，物体可能变形。α变为 α'，而x_1、x_2 分别改变为 x_1'、x_2'，l 也变为 l'。设 t_1，t_2 的变化量可忽略，即 $t_1 \doteqdot t_1'$，$t_2 \doteqdot t_2'$。那么，$l = x_1 + x_2 + t_1 + t_2 + \alpha = x + t + \alpha$，$l' = x_1' + x_2' + t_1' + t_2' + \alpha' = x' + t + \alpha'$，$\Delta l = l - l'$。则

$$\frac{\alpha-\alpha'}{\Delta l}=1-\frac{x-x'}{\Delta l} \tag{5.1}$$

因此，对于被握物体的硬度可由下式来评价。

$$Hd=\frac{x-x'}{\Delta l}=\frac{\Delta\varepsilon}{\Delta l} \tag{5.2}$$

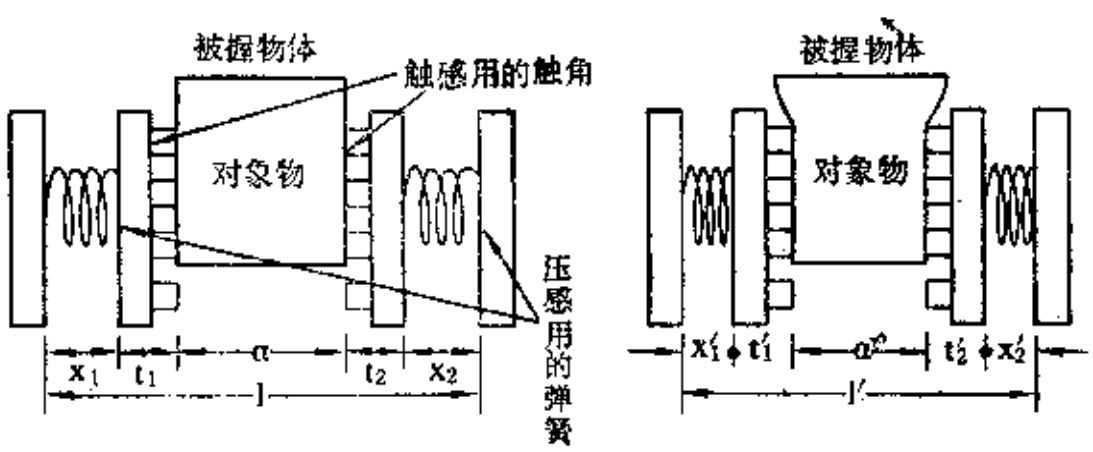


图 5.4　具有触感和压感的测定原理

耳　为了研究机器听觉，对于人耳的生理构造有必要从声理学角度进行粗略的了解。当外界声音从外耳、中耳传到耳蜗时，耳蜗就将外界声音进行频谱分析，变成电信号，通过听神经传导到大脑皮层。耳蜗具有初步分析声音的能力。

耳蜗的形状如蜗牛的壳，故得名。图 5.5 示出了人耳耳蜗的神经纤维排列。耳蜗从底部到顶部，高约 5 毫米，底部宽约 9 毫米。骨管依着一锥形的蜗轴，盘旋 $2\frac{3}{4}$ 转，而形成螺旋状。其表面有毛细胞（因细胞上端有纤毛而得名）。每个毛细胞都接触听神经的末梢纤维。这些纤维从蜗轴中的螺旋神经节发出，而分别和各毛细胞直接接触。每个毛细胞可以和几条或许多条纤维接触。神经纤维的排列也是螺旋式的。这些富有弹性的纤维称为底膜。底膜的宽度并不相同。如图 5.6 所示。

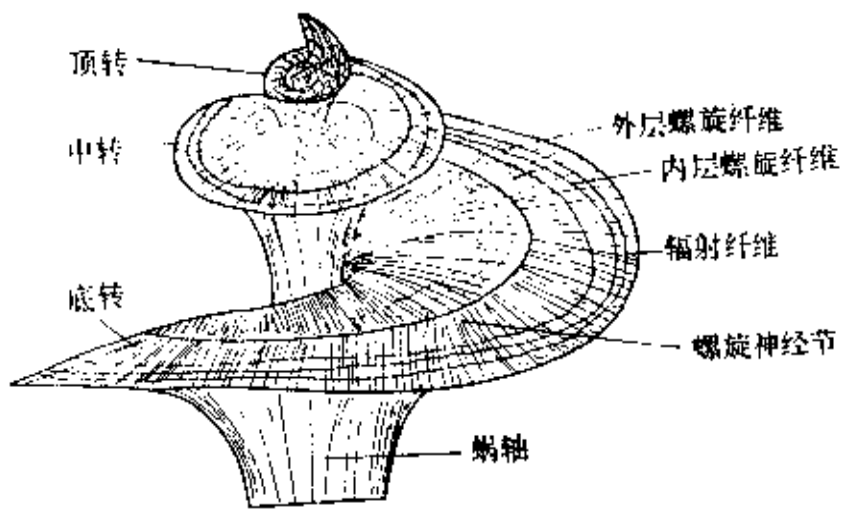


图 5.5　人耳的耳蜗

由于耳蜗是螺旋状分布的，声波到达时，从顶端至底部，其共振空间由大到小。因而从顶端至底部，引起底膜振动的频率也由低到高连续分布。从顶端至底部分布的毛细胞也连续分布式地分别受到 16HZ 至 20KHZ 的频率的刺激而发生兴奋，转换成电能，再引起听神经末梢发生冲动，传到脑中枢。不同的部位对不同的频率较敏感。当某一种频率的声波进入时，可引起某部位上多条纤维的振动。当声音的响度较大（即振动的幅度较大）时，则受刺激也大，所影响的底膜面积也较广，发生兴奋的毛细胞数目也较多。

由上述可知，我们可将不同的语音或声音通过话筒变为电信号，而后用硬件或软件的方

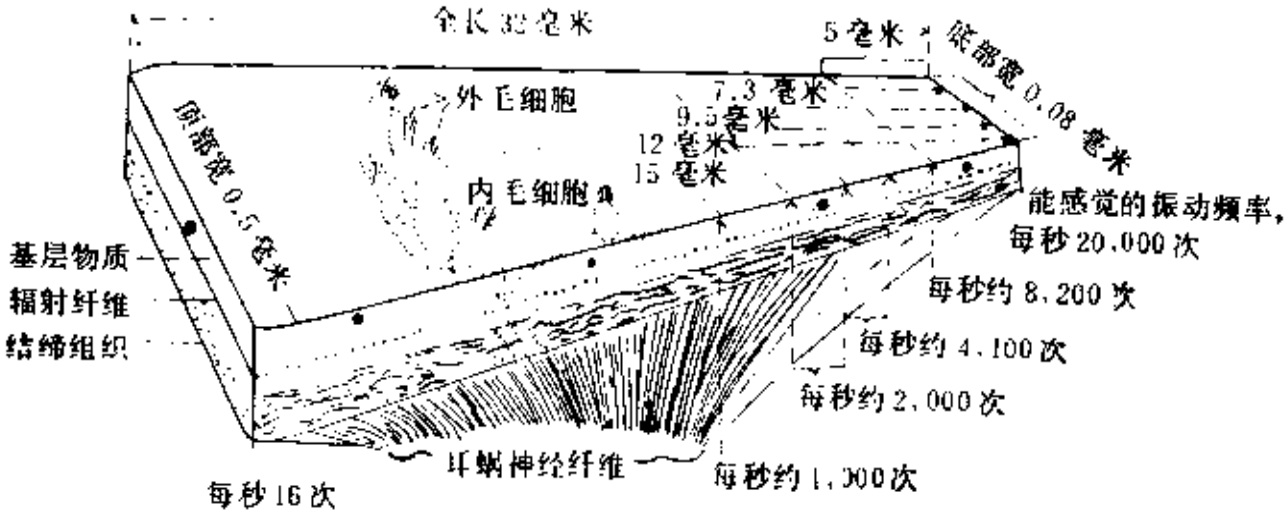


图 5.6　耳蜗神经纤维的排列

法对它进行频谱分析，作为预处理。而后由机器再进行识别。

口　通常采用喇叭。对许多不同频率的振荡进行控制合成送至喇叭，发出声音。但由于对声音的分析还不甚清楚，所发出的声音与人的语音还有较大差别。美、日等国也有研究模拟喉咙的发声器的。

脚　简单办法是采用车子。美、日等国也在研究二脚的步行机。需用计算机来控制其重心位置。

5.2　神经元与记忆

脑的智能活动的研究，一直从脑的神经结构及脑的功能这方面进行。脑的结构的研究已从神经生理和神经化学方面进行了大量的工作。虽然还不能用它来阐明脑的智能活动，但对这方面有所了解还是必要的。

神经元　很多学者认为，神经细胞及其所有的轴突，以及这些轴突的所有末梢，是神经系统的基础结构。Wuldeyer 把这种神经成分的综合体称为**神经元**。神经元如图 5.7 所示。人脑约有 100 亿个神经元。每个神经元平均与 1000 个左右的其它神经元联接。神经元之间是用脉冲传递信息的。每个神经元通过轴突接受由其它神经元来的信息，将脉冲变为细胞膜电位的变化，将各神经元来的信息加以综合，膜电位超过一定阈值时，送出等幅度的脉冲到其它神经元。脉冲频率与兴奋信号成正比。因而每个神经元就是一个复杂的信息处理单元。目前已提出 100 多种不同类型的神经元模型，从各方面表现了神经元的时空总和、阈值作用、不应期、疲劳效应和可逆性等特性。从智能模拟角度来说，早期的工作是 W.S.McCulloch和 W.H.Pitts（1943 年）的神经模型。这是神经元轴突模型中最原始最基本的模型，如图

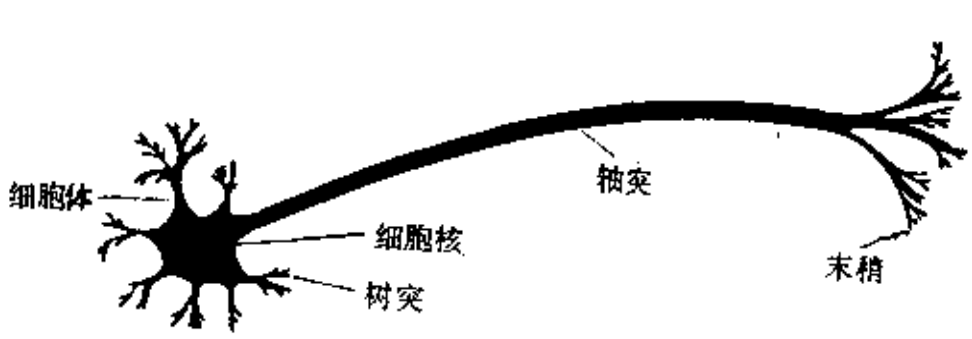


图 5.7　神经元

5.8 所示。它不是一个实际神经元的全生理学模型，而是模拟在神经元中的较简单的逻辑处理，从而产生二状态的模型。其输入有兴奋和抑制二类。在图 5.8 中的每个输入线都加权为 1。而 T 值为触发阈值。当兴奋输入的和E大于或等于触发阈值T，且没有任何抑制输入作用时，该神经元就被触发，输出值为 1。否则，输出值为 0。形式表示如表 5.1 所示。因此对于单个神经元来说，在时间 $t+\Delta t$ 上的输出，不是在时间 t 上的输出的函数。称这个模型为 **McCulloch-Pitts 模型**。

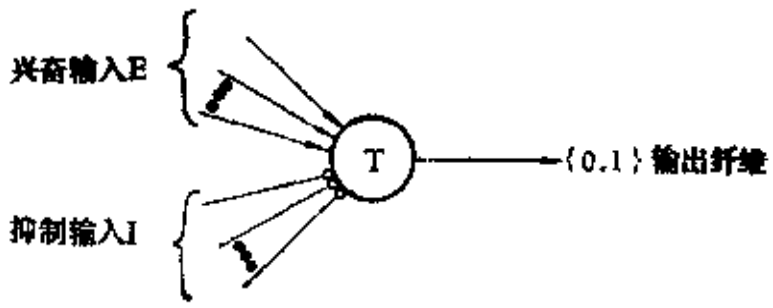


图 5.8 McCulloch-Pitts 神经元模型

从这个一般结构可以实现普通的布尔逻辑函数。图 5.9 示出了一些例子。

表 5.1 McCulloch-Pitts 模型的输入输出表

输入条件	输出
E≥T, I = 0	触发
E≥T, I > 0	不触发
E<T, I = 0	不触发
E<T, I > 0	不触发

对 McCulloch-Pitts 模型稍加变更，就是图 5.10(a) 所示的线性加权模型。其输出条件如式 (5.3) 所示。图 5.10(b)是它实现某种布尔逻辑的一例。

$$\text{输出} = \begin{cases} 1，\text{触发} & \text{当} \sum E - \sum I \geqslant T \text{时} \\ 0，\text{不触发} & \text{当} \sum E - \sum I < T \text{时} \end{cases} \tag{5.3}$$

记忆 人脑的记忆能力与计算机的记忆能力有很大的不同。到底人脑如何实现记忆功能，目前仍未完全解开这个谜。在一个神经阵列内如何实现记忆的方法，在这方面，有些学者提出了一些假设。了解一下也是有益的。

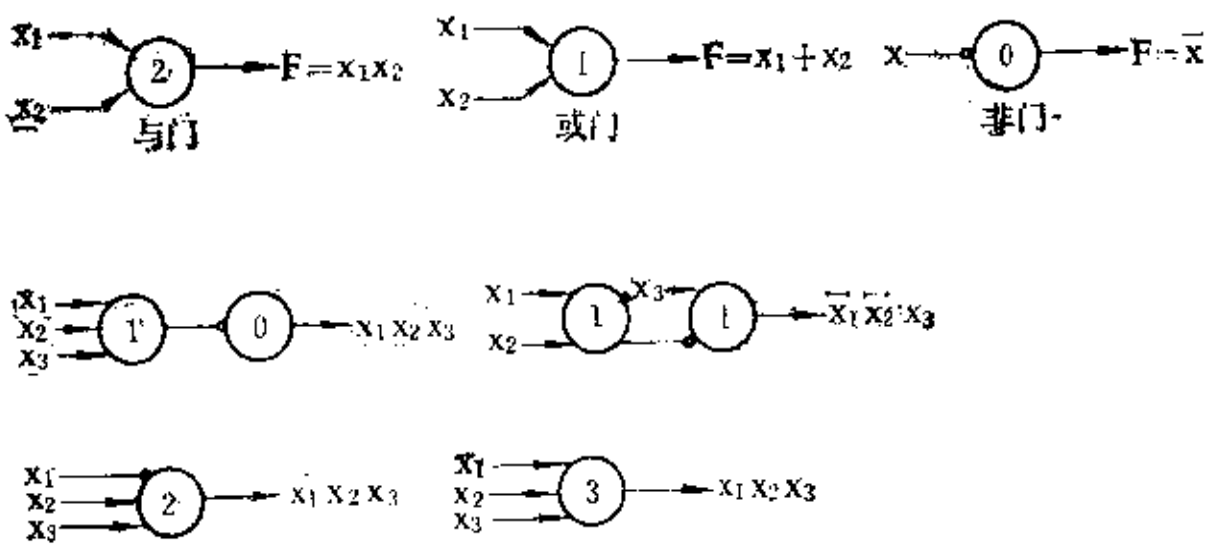
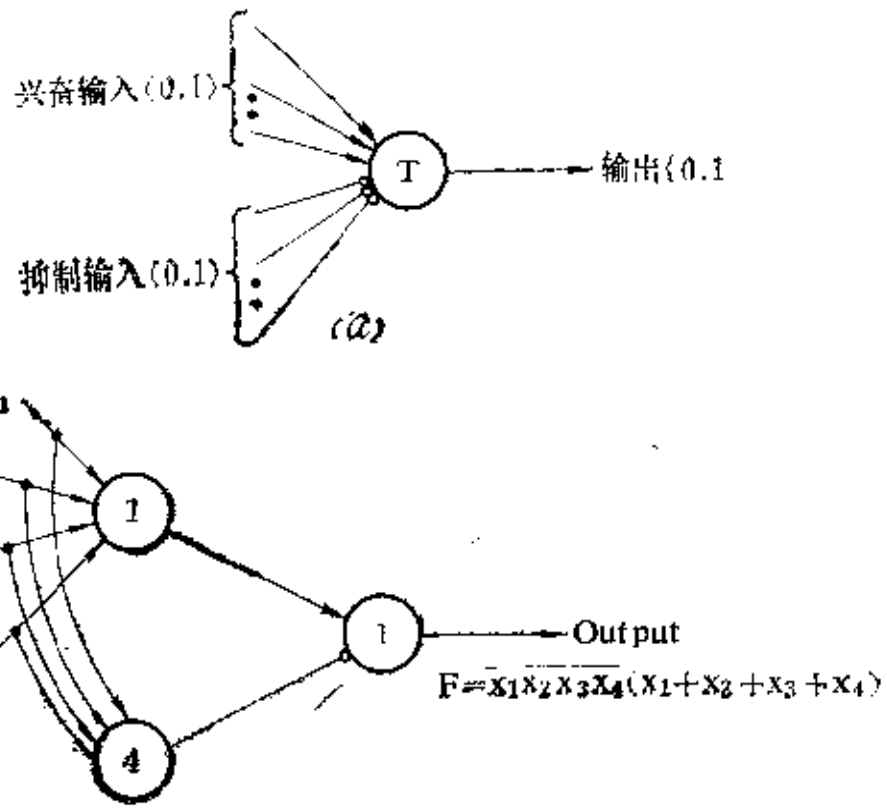


图 5.9 用McCulloch-Pitts神经元模型实现布尔逻辑函数的几个例子

由于在通过一个神经元的信号传播中，存在着一个固有的延迟。因而可认为有一个如图 5.11(a)所示的单位延迟元件。其输入—输出关系如图5.11(b)所示。在神经元的反馈装置中若用了这一延迟特性，就可组成如图 5.11(c) 所示的记忆单元。当抑制线为 0 时，加入写线的单个脉冲可以通过神经元而连续地传播，一直到抑制线作用到来为止。一旦抑制线作用，则该脉冲就被“遗忘”了。

人的“短期”记忆是容易遗忘的，但若适时地经常谈起他自己的“短期”记忆，则会变成一个“长期”记忆。因而采用将反复使用过的神经元的阈值降低的办法，一个神经元就能



(b)

图 5.10　线性加权模型及应用一例

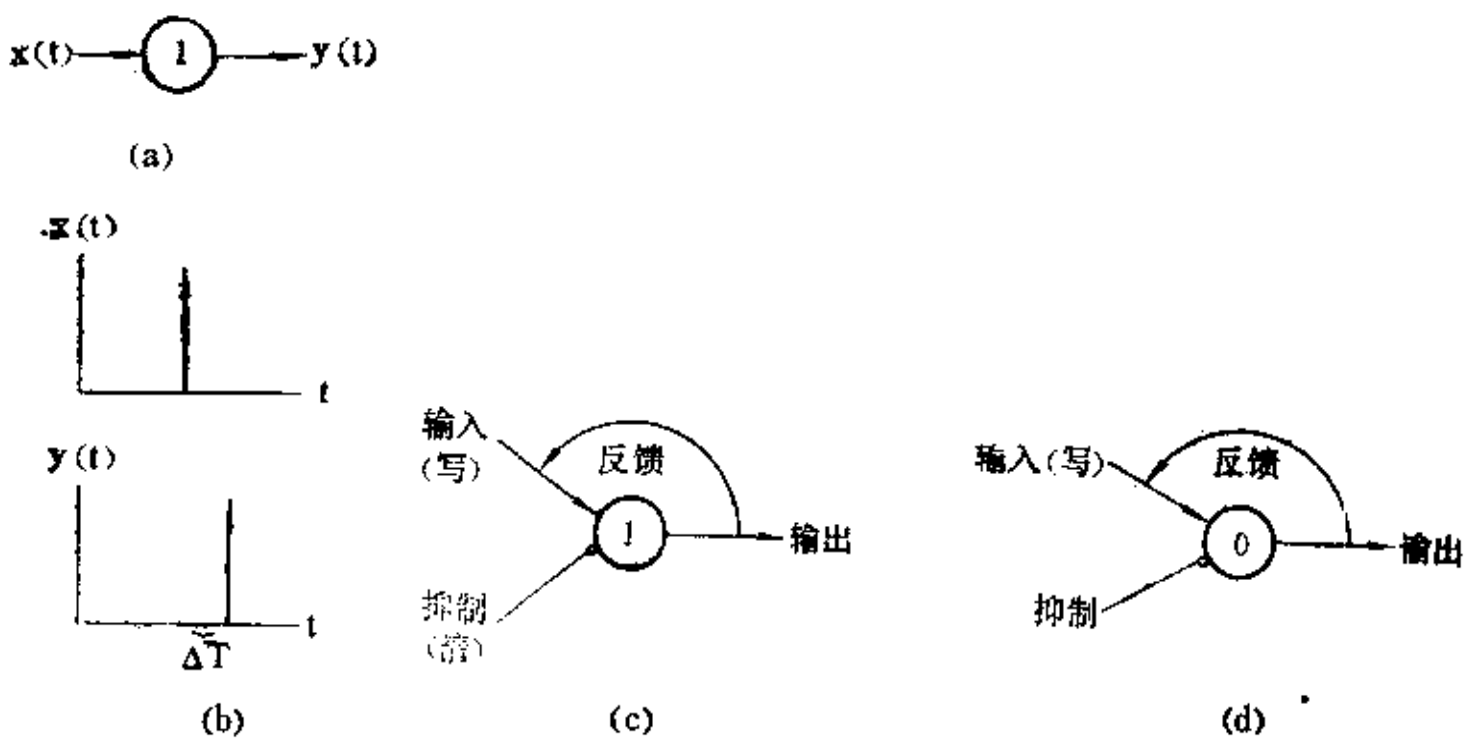


图 5.11　延迟元件及记忆单元

用它来完成这个功能。图 5.11(*d*) 就是表示如图 5.11(*c*) 的神经元在被反复触发之后，其阈值变得很低，直至零。在这状态下，抑制不起作用。因而该元将无限期地触发，从而完成永久的记忆。

联想记忆　人脑不仅能记住某些事物，而且有联想能力。例如，一看到野外的房子，就会联想到田野、树林、山丘和河流等等；一看到汉字“中”，就会联想到词汇“中间”、“中央”、“中部”、“中华”、“中国”、“中等”、“中学”，甚至于“当中”、“其中”等等。人的联想能力可以是很广泛的。可以从某一事物的某一特征而联想起一大堆事物来。可以从近义性联想，也可从反义性联想。儿童在学习中往往就是从这些联想中加深记忆，增强智力。虽然这些联想能力的生理机理目前尚不清楚，它涉及到生物工程学和心理学，但人们却在热心地研究着如何使机器也具有联想能力。

在计算机中具有联想记忆的联想式存贮器，可根据各个不同的具体问题而采用不同的方式。在此介绍其中一种方式。从中有助于理解联想式存贮器与普通的地址式存贮器的区别。

图 5.12(a) 示出了普通的地址式存贮器。在其记忆领域上付与地址。一旦从外部指定了地址，就读出该相应地方的记忆内容。可是它是根据地址来读出其记忆内容的。而图 5.12(b) 示出的联想式存贮器，则是根据询问的信息来读出其记忆内容的。从图中可理解这点。在记忆领域中，有“参照栏”和“信息栏”。一旦从外部给予“询问”的信息，就先在参照栏中查出其信息和询问信息一致的地方，而后读出它们相应的信息栏的内容。参照栏的数字不必象地址那样从 0 开始依序排列。可根据用户方便来排序。因此，它们有两点不同：

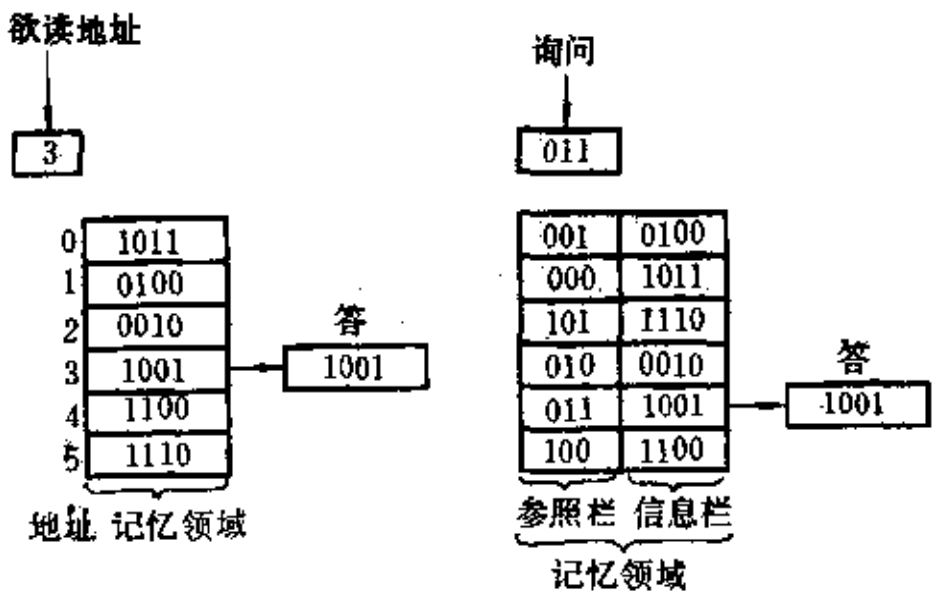


图 5.12 地址式存贮器与联想式存贮器

(1) 在地址式存贮器中，根据已作成的硬件来进行地址的解读，而且无法更改；但在联想式存贮器中，可改写参照栏的信息，这就很自由了。

(2) 由(1)而得出的必然结果是在联想式存贮器中，对于一个询问信息，可以有多个栏和它一致的地方，而且必须有能力处理它。

当然，根据需要，询问信息可以用不同方式给出。例如给出询问信息“1φφ”(即 100，101，110，111)，则从图 5.12(b) 中的参照栏查出与 1φφ一致的地方有二栏，即101，100。因而，其信息有 1110，1100。

5.3 机器视觉

机器视觉 对用肉眼看到的景物（包括二维图形）进行分析与解释，来识别其三维空间的物体，这方面的研究就是机器视觉。最初，从非常简单的场面开始研究，如识别长方体、

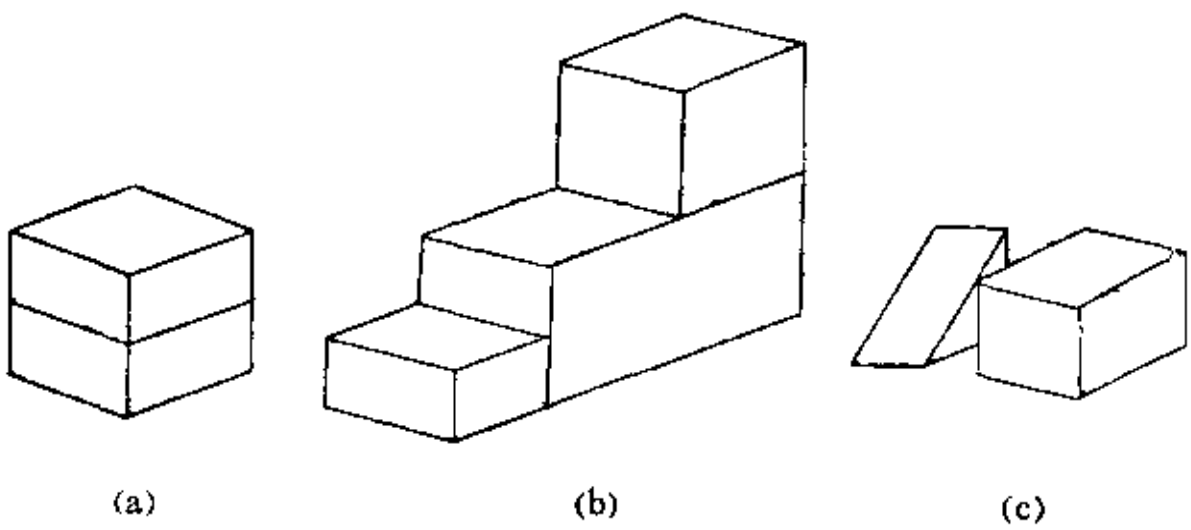


图 5.13 多面体的某些例子

三棱柱体及由它们组成的复合体。最近，已研究室内的景物及野外风景了。

多面体识别 最早用计算机识别物体的是 L.Roberts。因为大多数多面体都可以切割成长方体、三棱柱体，所以 Roberts 研究了长方体及三棱柱体及由它们组成的复合体。在无灯光（无阴影）情况下，将其二维图形用微分方法抽取出其轮廓线条，而成几何图形的线条画，如图 5.13 所示的几种线条画。而后再根据面与面之间的关系来确定哪些面是表示一个物体的。

在线条画中最主要的特征部分是线段的交接处及分叉点等。从各种图形的分析中，可归结为如图 5.14 所示的九种类型。图中的圆弧表示两个面有结合关系。其下面的英文字是其对应的名称。

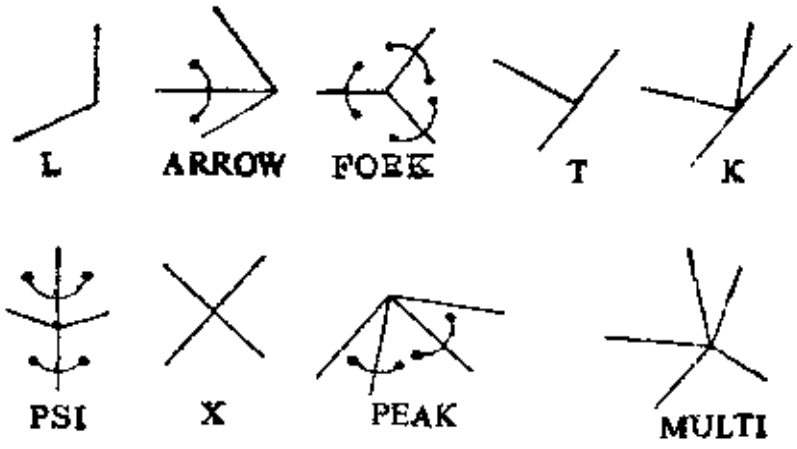


图 5.14 线段交接处和交叉点的几种类型

根据图 5.14 的规则可以对长方体、三棱柱体及由它们组成的复合体的图形进行分析，如图5.15(*a*)及(*c*)所示。而后再把图中各平面之间的结合关系用图论的节点、边的方法加以表示。图 5.15(*b*) 及 (*d*)分别是图 5.15(*a*)及(*c*)的结合关系图。

为了确定由那些面表示一个物体，在此引入如下二个假设：

(1) 连接有二重结合关系的面（图中用节点表示）就构成一个物体。如图 5.15(*b*) 的 A、B、C三个面是一个物体。D、E二个面也是一个物体。而（A、B、C）及（D、E）二者不是一个物体，可确定二者仅是重迭在一起而已。而图 5.15(*d*)中显然可看出A、B、C三面是一个物体。D、E 也是属于一个物体。G、H、I 也是一个物体。

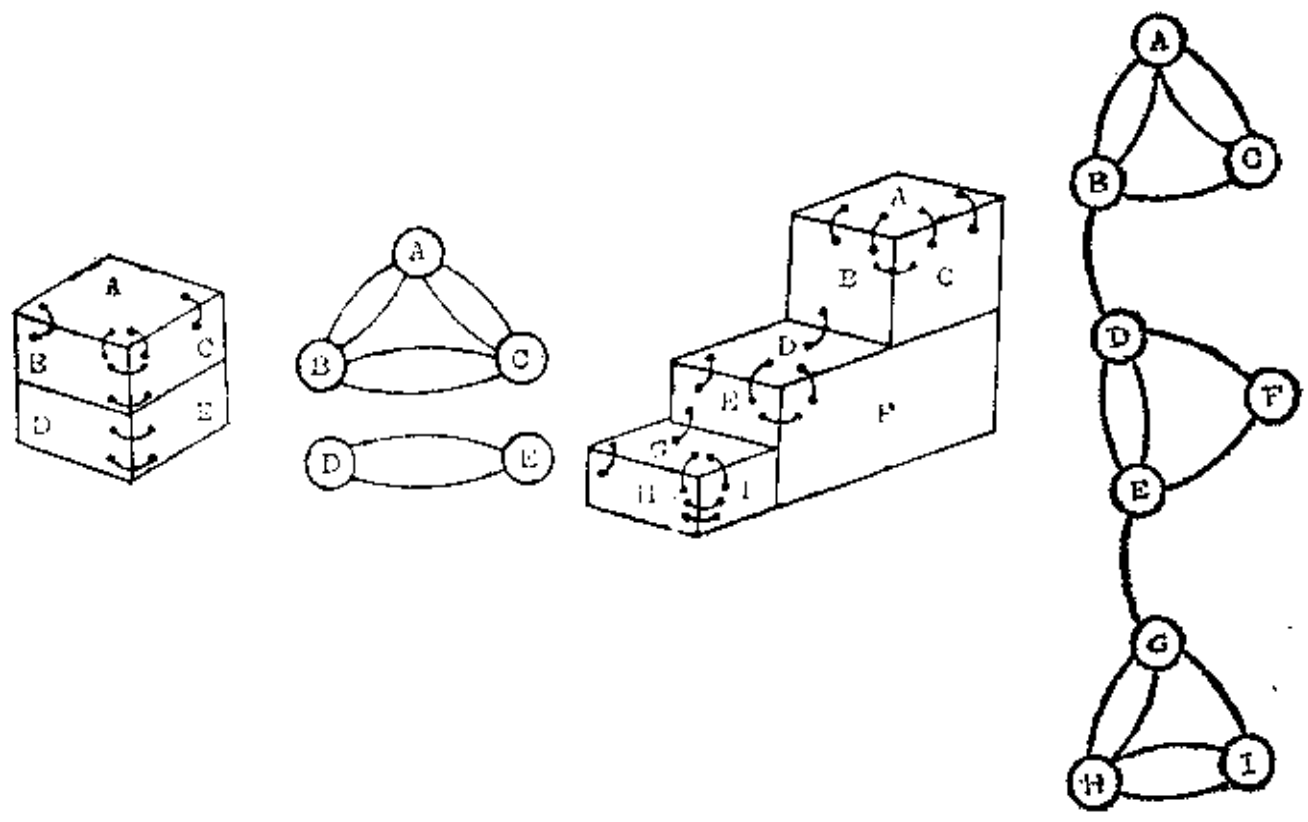


图 5.15 表示面与面间结合关系的二个例子

(2) 有二重结合关系的节点，可把它们当作一个节点考虑。如图 5.15(*d*) 中的 D、E 可作为一个节点。那么 F 与 (DE) 的关系就有了二重结合关系了。所以可以认为 DEF 构成一个物体。而B与D之间以及 E 与 G 之间都只有一重结合关系，故都不是同属一个物体。

这方法有缺陷。如图 5.13(*c*)所示的稍为复杂的场合，就无法确定 ABC三个面是一个物体了。

A.Guzman 补充一些假设，对这方法加以改进。其补充假设之一，如图 5.16 所示。在ARROW情况下，若有一条和 L_1 平行而且又与由 P 引出的其它线连着的线L_2存在的话，就在P 点的面 A和B 用**弱结合关系**（虚圆弧线表示）来强化之。原来用实圆弧线表示的结合关系称为**强结合关系**。

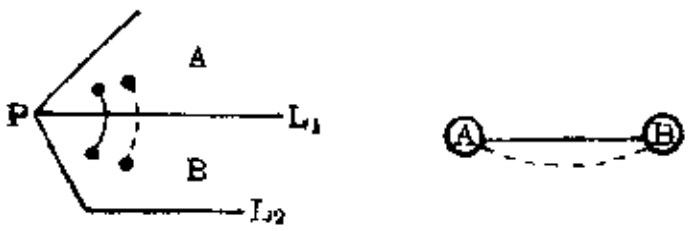


图 5.16 Guzman 的补充假设

这么一来，在图 5.13(*c*) 的稍复杂的场合，Guzman 方法就可以确定 ABC三个面构成一个物体，如图 5.17 所示。

弱结合关系不能和强结合关系等同看待，仅是“启发式”的。之所以称为**启发式结合关系**，是由于可能胜利也可能失败，但总的说来，有很大改善，提供了取胜的条件。例如，即使图5.18所示的简单场合，不用说，它由二个物体组成。Roberts 方法无法确定是几个物体。而 Guzman 的方法引入了弱结合关系。可能把AB CD面确定为一个物体，也可能把(B C)、(AD)分别看成二个物体。但却提供了一种可能，即 (AB) 及 (C D) 分别为二个物体。

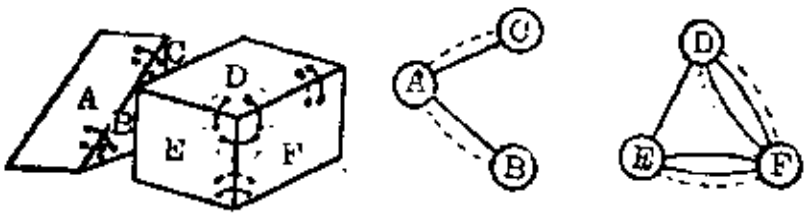


图 5.17 稍复杂的场合

M.Clowes, Huffman 及D.Waltz 对这方法加以发展。首先，作了一些限制，

(1) 没有阴影和裂缝。

(2) 所有顶点都是严格的三个面的交点，排除那些如图 5.19 所示的出现四个面的交点的情况。

(3) 观察点要选择得好，不能由于眼睛稍为移动一下，就改变了交点的拓扑结构。因此，图 5.20 所示的观察点是不允许的。

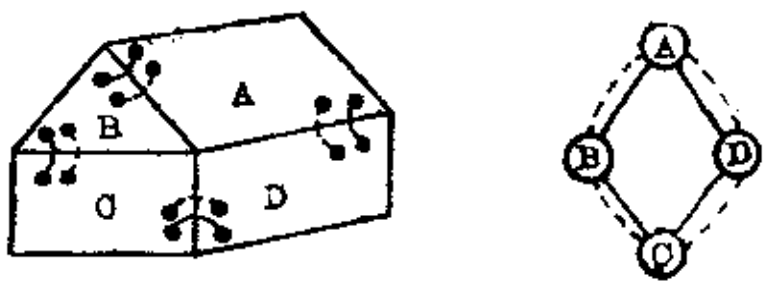


图 5.18 简单的难题

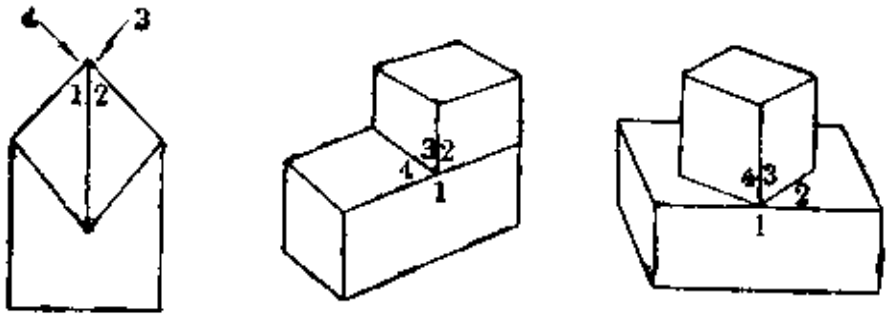


图 5.19 出现四个面的交叉点的情况

在上述限制下，按下列步骤（参看图5.21），根据线的性质而附加一些符号，就可把线的性质都表示出来。

首先，在区别物体和背景的线上标以箭头。使得物体的表面在箭头方向的右侧。

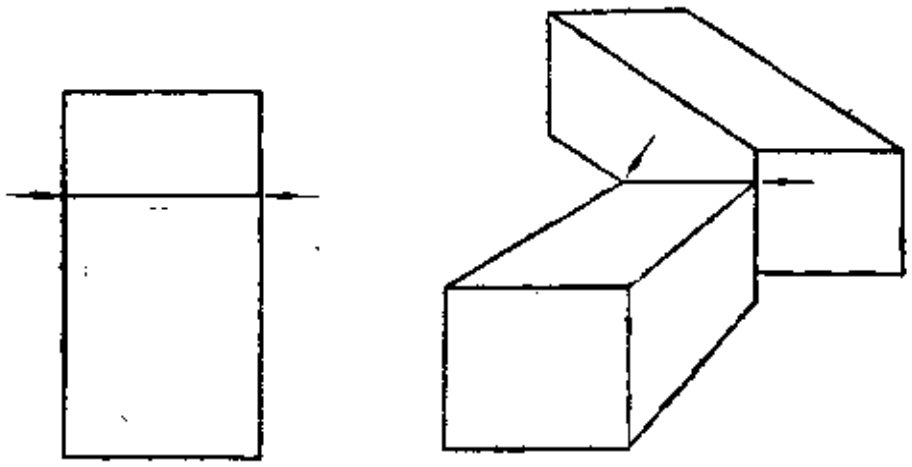


图 5.20　不允许的观察点

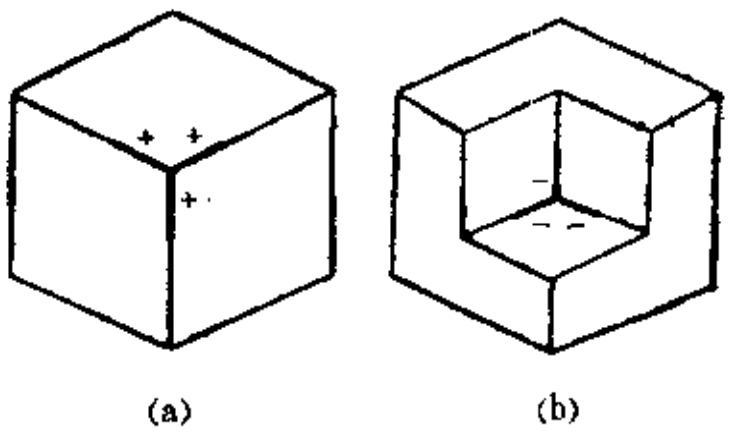


图 5.21　线的十、一号表示

其次，两个平面在外侧上以凸形式相交的线，给予“+”的记号，如图 5.21(*a*) 所示；以凹形式相交的线，给予“−”的记号，如图5.21(*b*)所示。

再对交接点的各种可能组合情况加以研究，可得出如图5.22所示的几种类型的交接点。它也作为判别规则。

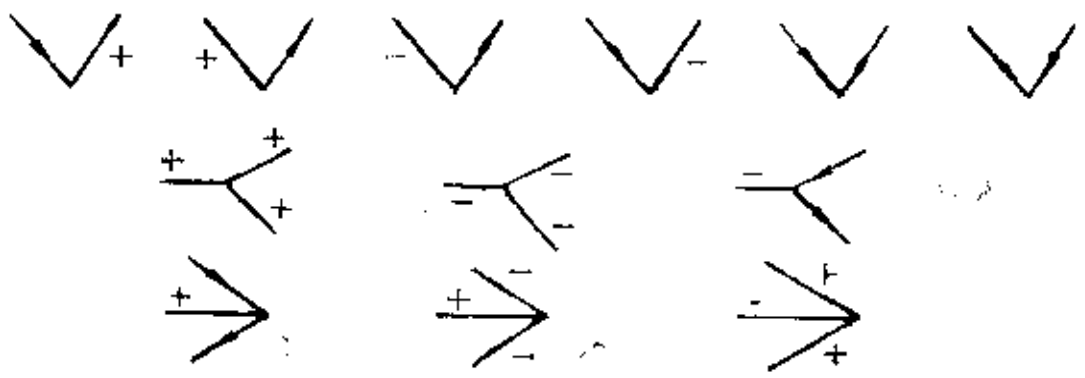


图 5.22　交接点的判别规则

根据图 5.22 的规则，可对图 5.23 所示的多面体进行识别。首先把物体的轮廓线标以箭号，如图5.23(*a*)所示。而后由箭号再标出与之相连接的其它线条的＋、－号，如图5.23(*b*)所示。这样，由外及里直至标完所有的线条标记为止，如图 5.23(*c*)及(*d*) 所示。

对多面体的识别还有许多改进方案。例如。有单个灯光照射的情况下，如何去除阴影，等等。在此不一一列举了。

景物分析　自对多面体的识别进行研究以来，至今已经对球和圆筒等较简单的曲面体的识别，对茶杯等日用品的识别，对脸谱的识别以及对简单的室内外景物的识别，有了一些试验。它们的共同之处是：

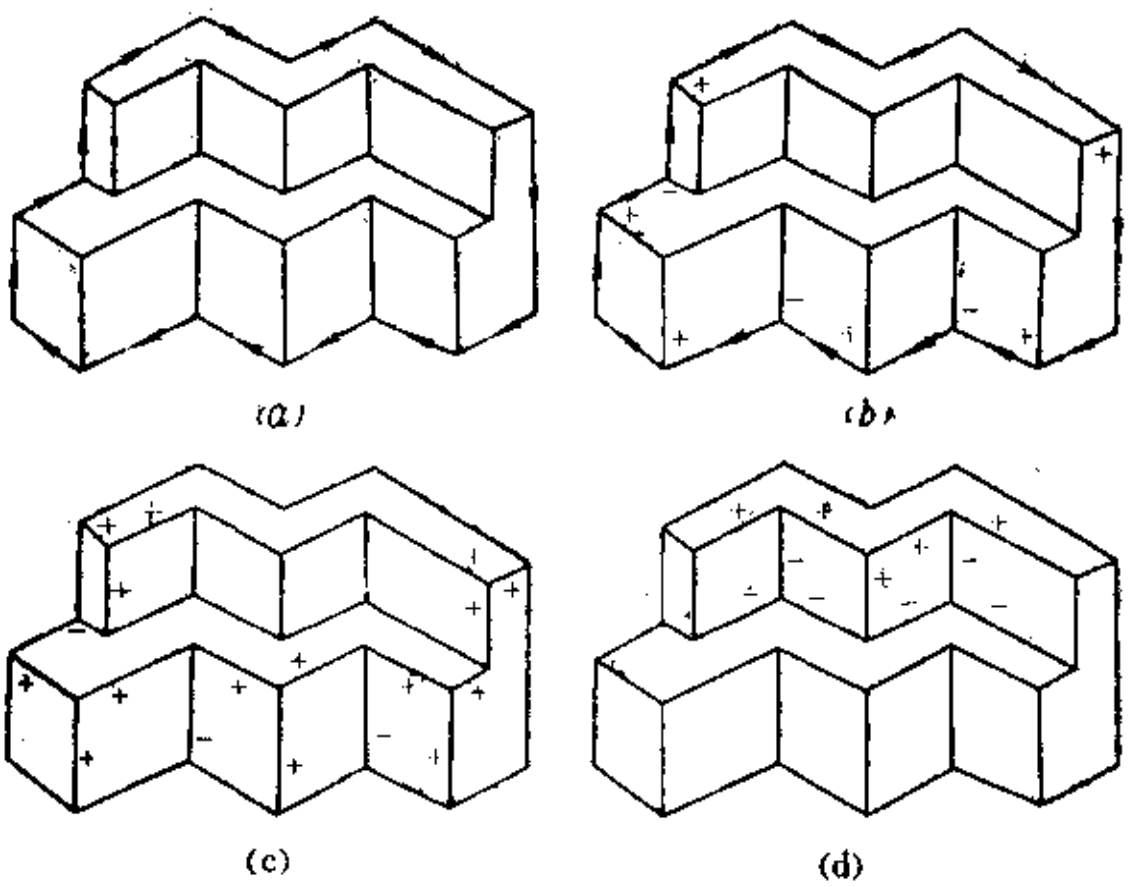
(1) 从图象中抽取出合适的直线和曲线。

(2) 对于不同的环境的景物，要知道它可能包含哪些物体，其外形的主要特征及付特征是什么。预先存放在计算机中。

(3) 根据这些特征来识别待识的图象。

在此仅以书桌上的景物识别为例加以说明。

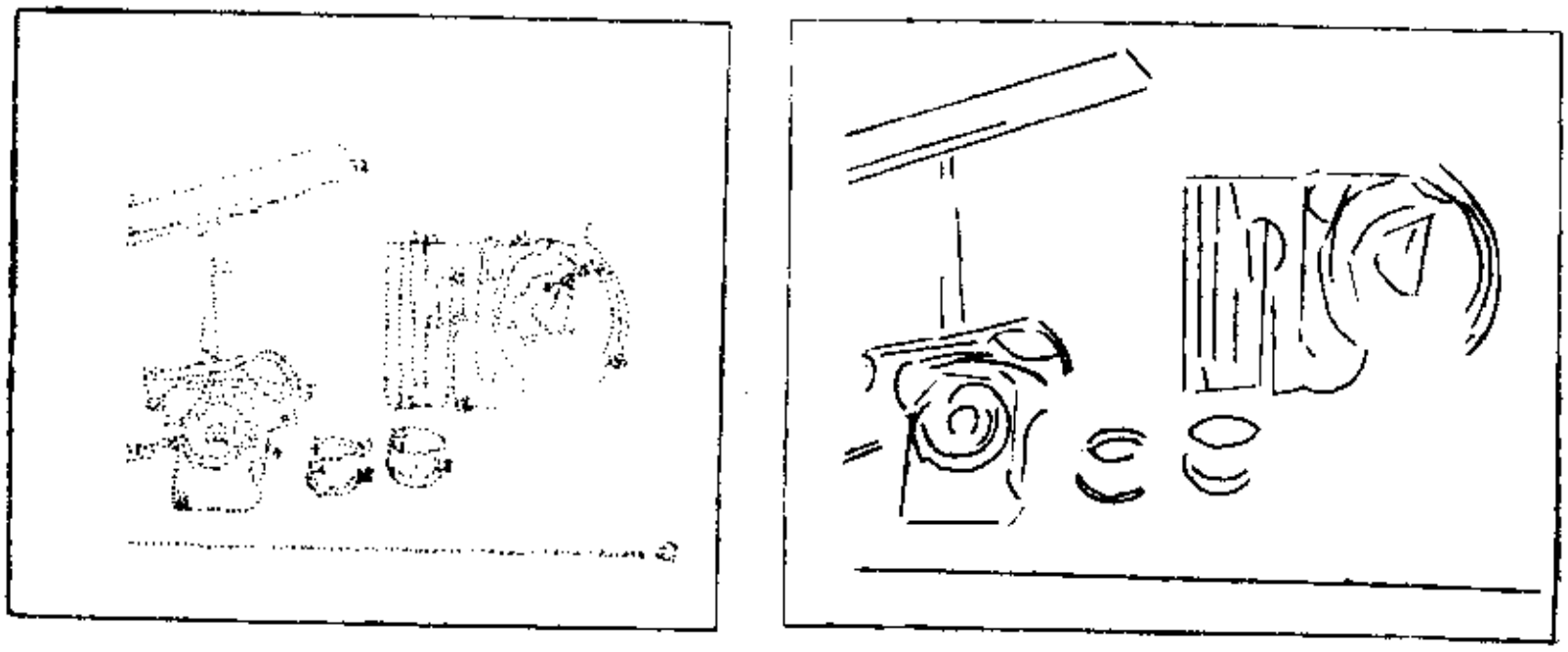
日本电总所的白井等研究了桌上电话机和台灯等的识别。先从原先照片上取出其附近明



(a) (b)

(c) (d)

图 5.23 识别多面体的例子



(a) (b)

(c)

图 5.24 桌上景物的识别

暗变化较大的点，如图 5.24(a)所示。而后圆滑成合适的直线和曲线，如图5.24(b)所示。最后按一定的搜索顺序来识别。例如电话机。首先要搜索其主要特征——拨号盘。其次搜索电话机的外形。而欲搜索拨号盘的话，则对所有曲线进行调查，取出其圆形曲线来；而后再把包围着它的一连串曲线也找出来。如能做到这点，那么它是电话机确实无疑了，就成功地抽取出电话机来了。如还不甚确切，再对把这圆形曲线围在中间的线详细地调查一下，而加以判别。实在找不到确切的外形曲线，就否定它。又如对台灯的识别。先搜索其罩子，而后再搜索其灯柱，等等。图 5.24(c) 示出了在识别景物时所用到的线。

5.4 眼—手 装 置

眼-手装置一般用于装配或堆积物件。日本日立中央研究所于 1971 年试验研究的装配用的眼-手装置，是典型例子之一。它能够根据三个分视图把散放在桌子上的积木搭成所要求的图形。图 5.25 是其示意图。图 5.26 是这种装置的硬件系统。

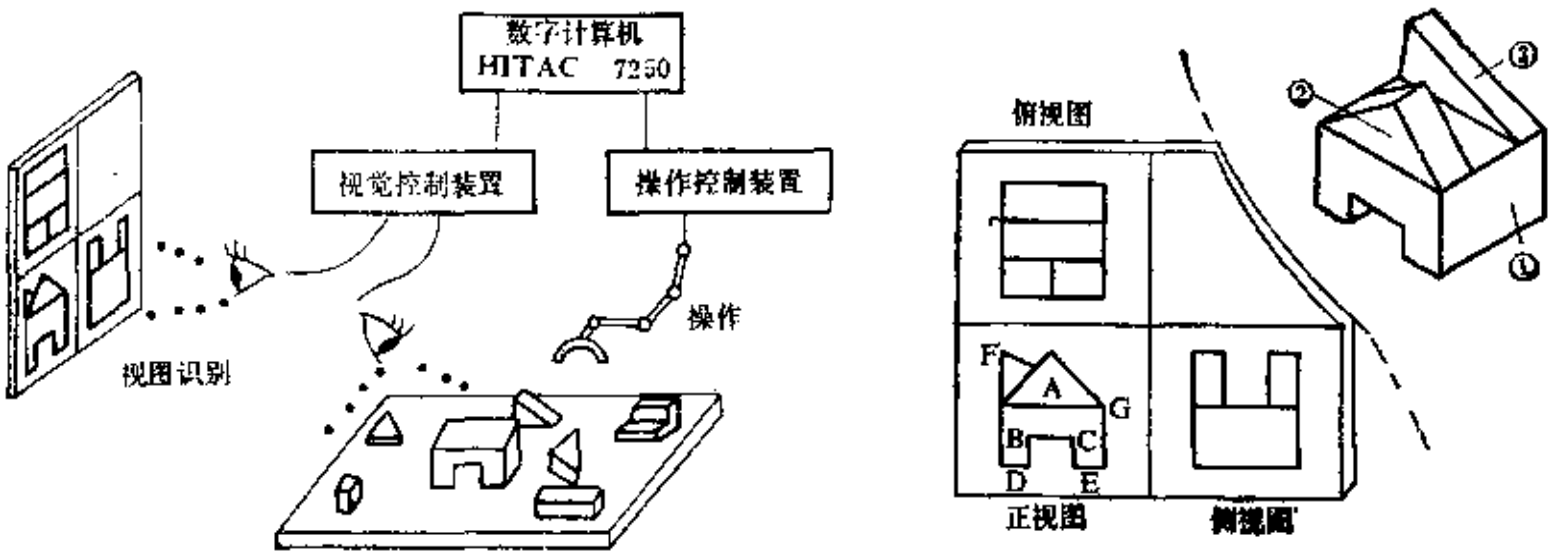


图 5.25 眼-手装置示意图

该装置的硬件系统中用二个电视摄象管作为眼睛，一个看图纸，另一个看积木的现场。采用与标准电视系统极类似的扫描系统进行操作。图象被划分为 76800 个象素（水平方向 320 个，垂直方向 240 个）。每个象素的信息通过模数转换器转换为 5 位数字信息，并送入

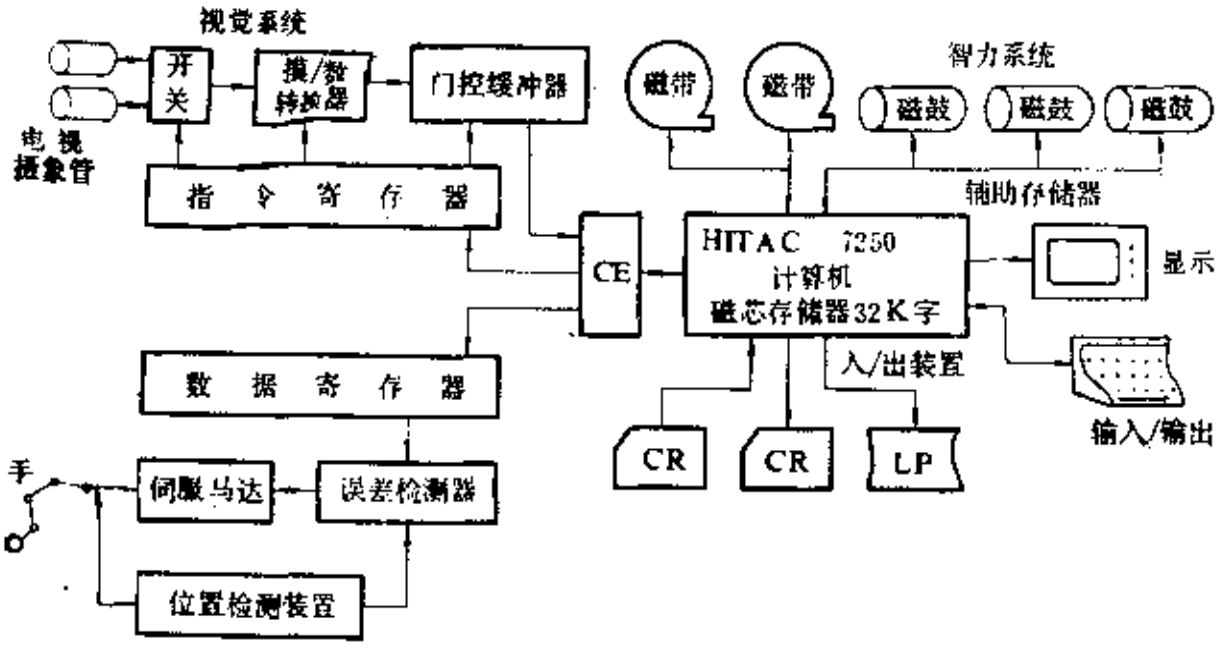


图 5.26 眼-手装置的硬件系统

处理控制机 HITAC7250（内存 32K，字长 16 位，操作周期为 2 微秒，加法时间为 4.5 微秒）。

其软件系统如图 5.27 所示，分为三个主要程序：图形识别程序，物体识别程序，操作程序。整个软件系统所占的存贮容量达 400K 字。存放在 512K 字的磁鼓中。

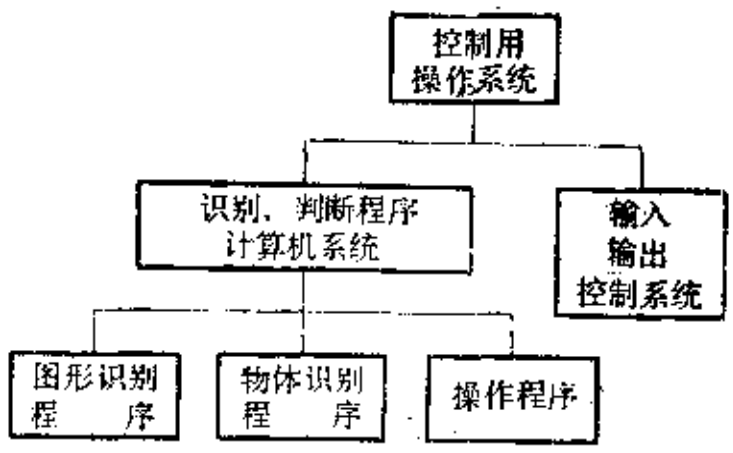


图 5.27　眼-手装置的软件系统

5.5　眼—车 装 置

SRI 的机器人 Shakey 是较典型的眼-车装置之一。Shakey 具有识别外界的感觉器官——电视扫描器做成的眼睛和自由行动的脚——车轮，以及操作外界的手段——一边用身体推着物体一边移动的办法来移动物体。它所能行动的世界是在室内。如图 5.28 所示。有简单形状的房间及使房间之间连通的门。在房间中间还放有类似于图 5.17 所示的长方体箱子和楔形箱子。Shakey 就按照人的命令把放在不同位置的三个箱子推在一起，并把楔形箱子并在长方体的边上作为斜坡，利用这个斜坡登上长方体的顶上。

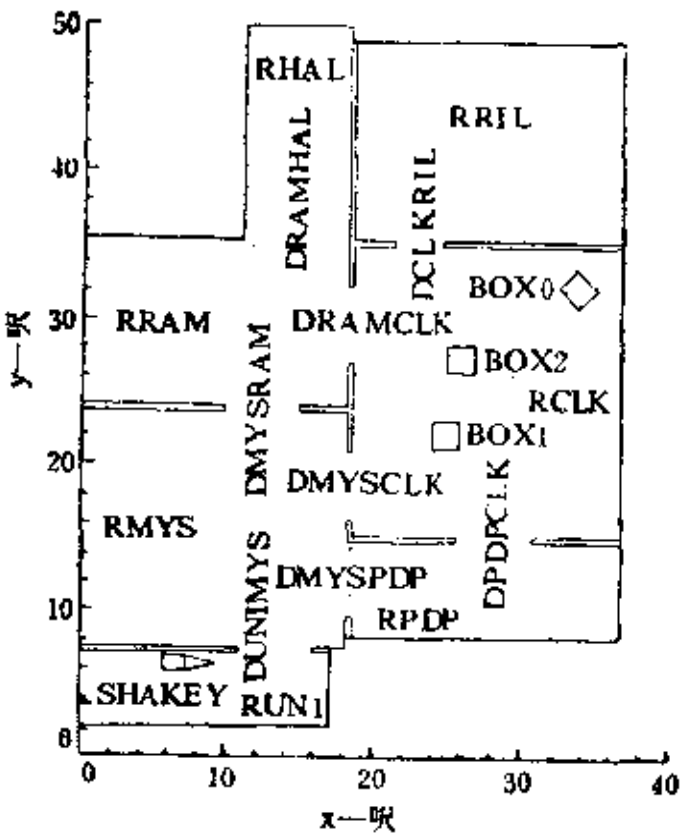


图 5.28　Shakey 的活动世界

从这个例子中可看出，要让机器人在所处环境中活动，必须将人对于外界的经验知识预先告诉计算机，让它能在活动过程中计划自己的行动。称这种计划行动为**规划**。

Shakey 的规划是根据称为 STRIP 的问题求解程序来进行的。STRIP 巧妙地把问题求解方法与自动定理证明的方法结合起来而进行 Shakey 的规划的。

在此，机器人的规划要解决几个问题。

(1) 知识表达问题。要把机器人所处的环境和人的经验用某种表达方式送入计算机。例如，STRIP 把图 5.28 的现状用一阶谓词逻辑式来表示。

$$\text{INROOM (ROBOT, RUN1)} \tag{5.4}$$

$$\text{JOINROOM (DMYSRAM, RRAM, RMYS)} \tag{5.5}$$

$$\text{BOX (BOX1)} \tag{5.6}$$

$$\text{SHAPE (BOX1, BOX)} \tag{5.7}$$

……

还有一般的知识表达方法：

$$(\forall x)(\mathrm{BOX}(x)\rightarrow\mathrm{PUSHABLE}(x)) \tag{5.8}$$

……

式(5.4)～式(5.7)的ROBOT，RUN1，DMYSRAM，…等都是表示特定的常量——机器人，房间和门。而式（5.8）的 x 是变量。由式（5.6）和式（5.8）可导出式（5.9）。

$$\mathrm{PUSHABLE}\ (\mathrm{BOX1}) \tag{5.9}$$

这是可理解的。因为从式(5.8)可知，如果箱子是 x 的话，就可能把 x 推走。而式(5.6)告诉我们，箱子是 1 号箱，因而 1 号箱就可能被推走（这就式(5.9)的意义）。

(2) 设定好操作的集合。由于机器人的活动不仅受外界现状的限制，而且对外界的状态也有影响（箱子被推走了，外界状态就发生变化了），所以要预先将各种可能的操作进行分析：操作的先决条件是什么？操作之后原先的状态改变了哪些？

STRIP 就是把 Shakey 实行某种操作的先决条件用逻辑式表示。例如，动作 PUSH (ob, x, y) 的先决条件为

$(\exists r)(\mathrm{INROOM}(\mathrm{ROBOT}, r)\wedge\mathrm{INROOM}(ob, r)\wedge\mathrm{LOCINROOM}(x, y, r)\wedge\mathrm{PUSHABLE}(ob))$

它表示“物体 ob，机器人 ROBOT (Shakey)，移动地点(x, y)都在同一房间 r 内而且该物体又是可推走时，这个操作就可能执行了”。

STRIP 还用“删除表”(Delete-list)和“附加表”(Add-list)来分别表示动作执行之后原先状态取消哪些，又出现哪些新状态。

(3) 当外界状态变化，原来可行的规划变为不可行时，要有改变规划的能力。

例如，STRIP 根据 BLOCKED(DPDPCLK，RCLK，BOX2)(从房间 RCLK 中用 2 号箱子堵住门DPDPCLK)，做出了如下规划：

GOTO1 (DUNIMYS)，GOTHRUDR (DUNIMYS，RUN1，RMYS)

GOTO2 (DMYSCLK)，GOTHRUDR (DUMYSCLK，RMYS，RCLK)

BLOCK (DPDPCLK，RCLK，BOX 2)

如果有 3 号箱子把门 DMYSCLK 堵住了，则立刻把上述规划改变如下：

GOTO1(DMYSPDP)，GOTHRUDR (DMYSPDP，RMYS，RPDP)

GOTO2(DPDPCLK)

GOTHRUDR (DPDPCLK，RPDP，RCLK)

BLOCK (DPDPCLK，RCLK，BOX 2)

图 5.29 表示执行之后的状态。

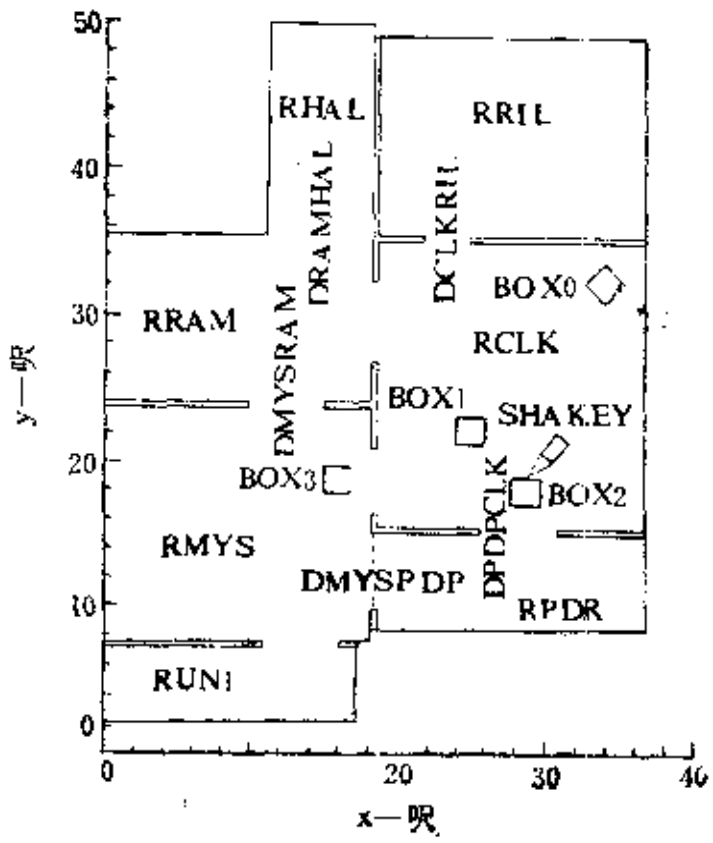


图 5.29 Shakey 改变规划后的执行状态

(4) 可能要有更高级的知识表达。在此，仅举时间概念和观察点移动这二个方面作为例子加以说明。

时间概念，随着时间的推移，外界状态也会有所改变。因而在解决机器人的某些问题时，在模型中有必要引入时间概念。

例如，在一阶谓词逻辑中，“如果在地点 x 下雨，人 p 在地点 x 处，且 p 在外面的话，则 p 淋湿”这句话可用下式表达：

$$(\forall x)(\forall p)(\mathrm{RAINING}(x)\wedge\mathrm{AT}(p,\ x)\wedge\mathrm{OUTSIDE}(p)\rightarrow\mathrm{WET}(p)) \qquad (5.10)$$

但是式 (5.10) 不能表达如下这句话的意义：“如果在地点 x 下雨，人 p 在地点 x 处，且 p 在外面的话，则 p 将会淋湿”。

为解决这种时间问题，J.McCarthy 引入了 $\mathrm{F}(\lambda s',\ \pi(s'),\ s)$。$\mathrm{F}(\lambda s',\ \pi(s'),\ s)$ 表示：就依照状态 s，在未来的状态 s' 中，$\pi(s')$ 成为真的。因而后一句话可以用下式表示。

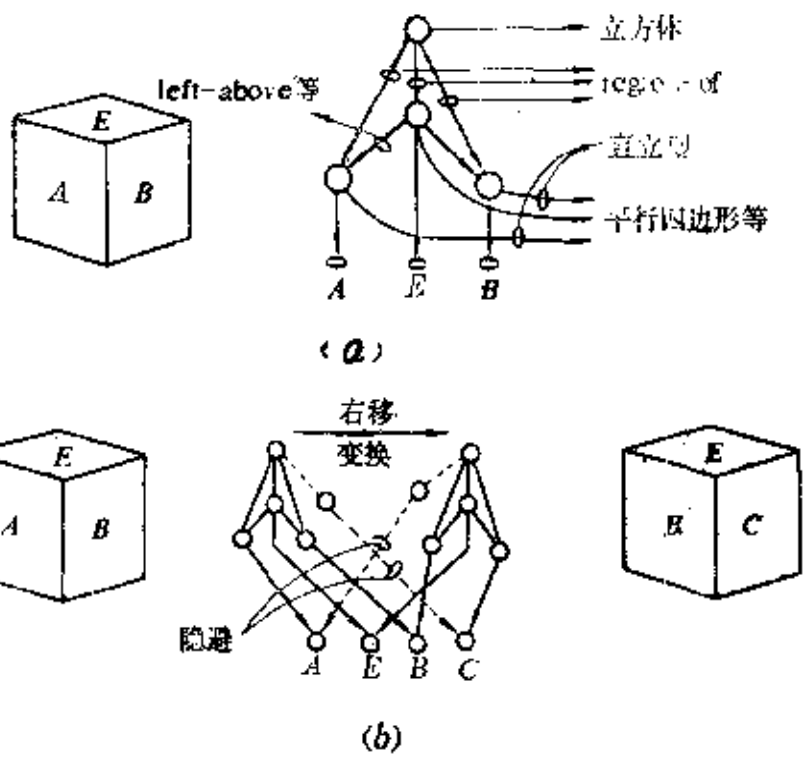
$$(\forall x)(\forall p)(\forall s)(\mathrm{RAINING}(x,\ s)\wedge\mathrm{AT}(p,\ x,\ s)\wedge\mathrm{OUTSIDE}(p,\ s)\rightarrow\mathrm{F}(\lambda s',\ \mathrm{wet}(p,\ s'),\ s)) \qquad (5.11)$$

式 (5.11) 读为“如果在状态 s 中，在地点 x 下雨，人 p 在地点 x 处，且 p 在外面的话，就依照状态 s，在未来的状态 s' 中，那个人受淋”。

象这样的谓词演算，有人称它为**高阶谓词演算**。

观察点移动 机器人是在所处的世界中活动的，因而对于同一物体，观察点也在改变，所得的图形结构可能改变。

M.Minsky 于 1974 年提出一种知识表达形式——**骨架**。本书中没有对此专述，但读者接触到它时也会一下子理解的。用骨架形式来表达从某一观察点对某一立方体所观察的图形结构，如图 5.30(a) 所示。而如果观察点从左至右移动的话，则其骨架也产生变换。如图 5.24(b) 所示。这样的表达法是好的，但还没有具体的系统。



(a)　　(b)

图 5.30　根据骨架形式，观察点移动的表达法

(5) 要能理解语言。不管机器人是接受文字或声音等语言，都必须能理解人给予的语言。除非人给予的语言是原先约定好的几句固定格式的语言，否则，只要一超出这些“死规定”，那怕是使用很简单的简单句，机器要理解它也是不容易的。首先，机器要懂得句法和语法（本书中统称之为句法），其次，要懂得语义。因此，语言理解系统必须包含句法分析和语义分析这二部分。有关这方面内容在下一节中一起叙述。

5.6 对答系统

将在某范畴内的一些事实告诉机器之后，人们若对机器提出有关这范畴内的一些问题，机器能够正确地回答。若对问题中有不明确的地方，机器还会反提问题来请教。这种系统称为**对答系统**，或**问题回答系统**。这是智能模拟的重要课题之一。

对答系统的组成部分如图 5.31 所示。为了能够对答，它必须对所提供的事实和所提的问题等语言进行句法和语法的分析，还必须进行语义分析以了解其意义，并导出解答来。为要管理这些信息，数据库管理当然是需要的。

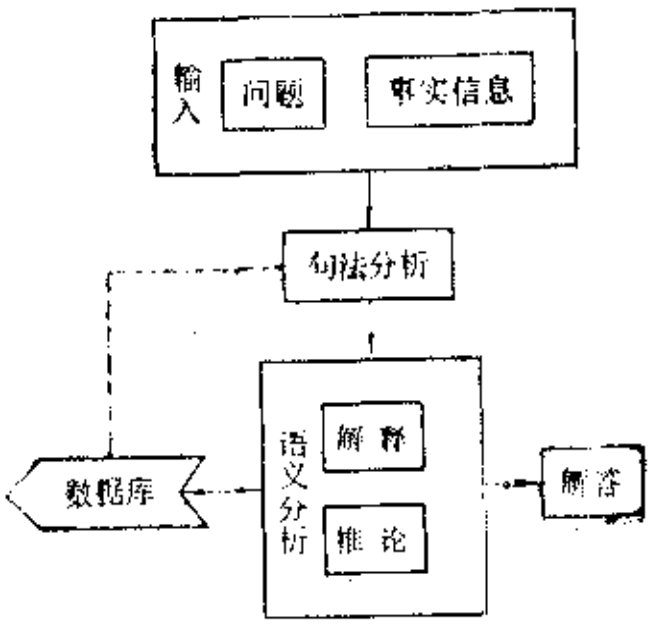


图 5.31 对答系统的组成部分

在目前，对答系统还是初级阶段。因而，在此列举英语中一些简单的例子来说明这个系统的最基本思想。而且为方便起见，在本节中所涉及到的句法、语法方面的词汇，并不完全采用语法教科书中的词汇，而是根据需要另加一些词汇。

5.6.1 句法分析

先引入名词组、介词组和动词组的概念，而后介绍句法分析。

名词组 在英语中，名词往往带有冠词、形容词之类修饰语，可把它们看成一个整体，其核心是名词。故把它定义为**名词组**。在此，把 a，the，this，that，these，those，one 等冠词、数词定义为**限定词**。因此，名词组可由限定词、形容词和名词组成。当然，有些名词（特别是专有名词）是不带任何修饰语的，也把它做为一个名词组看待。为方便起见，可参照第四章第二节的文法知识，用状态传递图表示名词组的组成及其语法约束。如图 5.32 (*a*)所示的状态传递图就表示一个**基本名词组**。其中，S_1 表示名词组的初始状态，S_3 表示终止状态(即可接受状态)。从中还可看出，它可以不带形容词，也允许任意有限个形容词，对它加以修饰。

例如，“a big red pyramid”、“the pyramid”、“China”在名词组状态传递图中是可以从初始状态到达终止状态的。

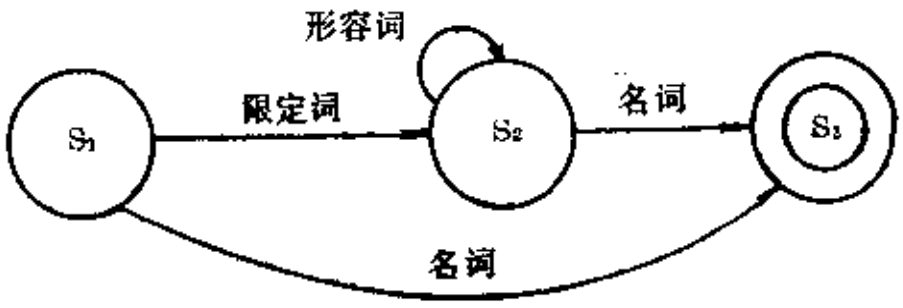
介词组 **介词组**是由一个介词及紧跟其后的一个名词组所组成。其状态传递图如图5.32 (*b*)所示。介词组的这个介词称为该介词组的**标识介词**，或**标识词**。

例如，“on the big block”、“near the empty box”和“by the furry purple cube”都是介词组。在传递图中都能到达终止状态。其标识介词分别为 on，near 和 by。

在名词组之后可能还带有若干个介词组，而这介词组都是修饰该名词组的，故都把它们归入名词组中。其状态传递图如图 5.32(*c*) 所示。

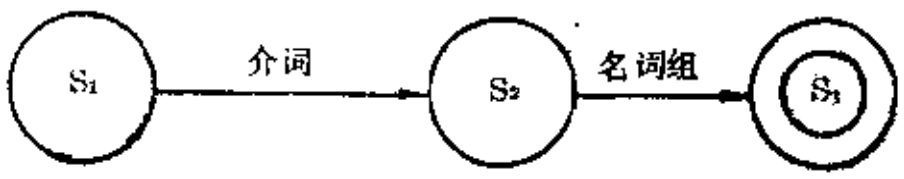
例如，“a red pyramid
on the big block
near the empty box
by the furry purple cube…”

基本名词组



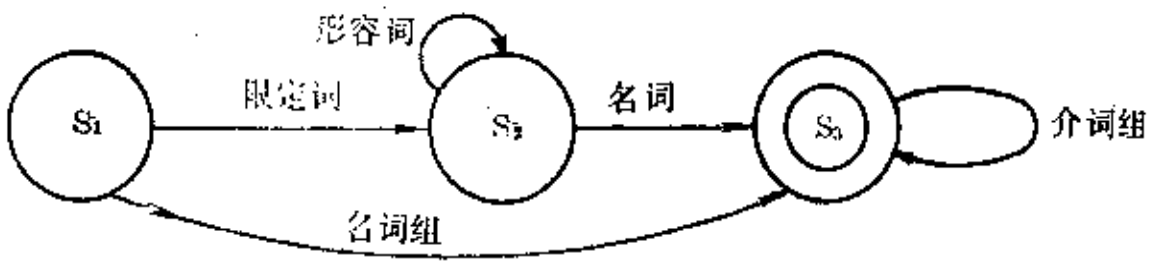
(a)

介词组



(b)

名词组



(c)

图 5.32　描述语法约束的传递图

是一个名词组。

把图 5.32　用文法形式表示出来，则为

〈名词组〉→〈名词〉

〈名词组〉→〈限定词〉{〈形容词〉}$_0^n$〈名词〉{〈介词组〉}$_0^n$

〈介词组〉→〈介词〉〈名词组〉

动词组是由助动词、付词及动词等组成的。读者自己也可给予定义。在此仅研究动词组只由一个动词组成的情况。

句法分析　**句法分析**就是对于给予的句子加以分析，分析它是否符合句法规则。符合者，称为在句法上正确。否则，称为在句法上出错。

对英语的简单句中的一些只带有一个动词的陈述句略加分析一下就可知，句子中除了一

个动词之外，主语、宾语都是名词组。因而，一个句子实际上可分成一个动词及若干个名词组和介词组。例举如下：

Robbie read the book in the library, at a desk by the wall, under a picture, near the door.
名词组 动词 名词组 介词组 介词组 介词组 介词组 介词组

如果要给予形式定义的话，则可用四元组 $G=(V_N, V_T, P, S)$表示。其中，V_N, V_T, P, S 的意义与第四章中模式文法的 V_N, V_T, P, S 的意义相同。在上面这句英语的具体情况下，可把 V_N, V_T, P 具体列出来（S 是起始符号）：

V_N = {S, 〈名词组〉, 〈动词〉, 〈介词组〉, 〈限定词〉, 〈名词〉}

V_T = {Robbie, read, the, a, book, library, desk, wall, picture, door, at, in, by, under, near}

P: S→〈名词组〉〈动词〉〈名词组〉{〈介词组〉}[5]

〈名词组〉→〈名词〉

〈名词组〉→〈限定词〉〈名词〉

〈介词组〉→〈介词〉〈名词组〉

〈动词〉→{read}

〈限定词〉→{a, the}

〈介词〉→{at, in, by, under, near}

〈名词〉→{Robbie, book, library, desk, wall, picture, door}

句法分析仅是必要的第一步。若句法分析之后认为是正确的话，其语义也不一定正确。例如，The book reads Robbie in the wall.在语法上是正确的，但语义上不对。另外，有些词本身有多种意义，单靠句法分析是解释不清的。有些英语专业的翻译人员，虽然语法及词都懂，但无法翻译科技专业的书，原因就在此。因而语义分析是不可少的。

5.6.2 语义分析

在语义分析中，把句子看成是以动词为核心，而后分析其他的名词组、介词组与动词的关系。介词组总含有名词组。所以，**语义分析**就是分析各个名词组（包括介词组中的名词组）与动词的关系。换句话说，语义分析就是把所有的名词组在句中所扮演的角色(所起的作用)分析清楚。我们把名词组在句中所起的作用称为**名词组角色**。

名词组角色 假设动词是表示动作的，那么可用图 5.33 来形象地表示名词组角色与动词的关系。

"动作对象"是动作使之改变状态的人、事、物。通常，宾语就是动作对象。如 Robbie hit the ball 中的 ball。有时主语也属于动作对象。如 The ball was hit by Robbie中的ball。

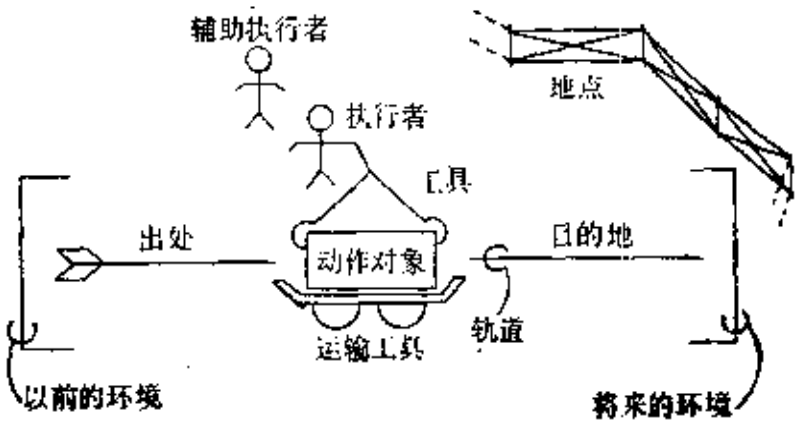


图 5.33 动作与名词组的关系

"执行者" 是产生动作的主角。通常，句子的主语是执行者。但可能出现在介词组中的名词是执行者。如 The ball was hit by Robbie 中的 Robbie。

"工具" 是执行者所用的东西。常常伴随着词 with，这是工具型名词组。例如，Robbie assembled another robot with a screwdriver 中的 a screwdriver。

"辅助执行者" 是给执行者当配角来一起执行动作的。常伴随着介词 with。例如 Robbie assembled another robot with Suzie 中的 Suzie。

"出处" 与 **"目的地"** 就是动作的出发点与目标。可以是物理空间的起点与终点。如 Robie went from the dining room to the kitchen。也可以是非物理空间。如 Robbie's mood went from bad to worse。

"运输工具" 常伴随着 by 等介词。如 Robbie always goes by train。

"轨道" 从出处到目的地的移动都是在轨道上发生的。常伴随着介词 through，over 等。如 "They went in through the front door"。"Robbie carried her over the threshold"。

"地点" 是动作发生的地方。常伴随着 in，at，by，under，near。如 He studied in the library，at a desk，by the wall，under a picture，near the door。

在图 5.33 中没有表示出来的还有：

"受益者" 动作是为他而执行的，他可以是人，也可以不是人。常伴随着介词 for。如 Robbie cleaned the house for mother 中的 mother。

"原材料" 是造出产品的物质。常伴随着介词 out of 。如 Robbie made a calculator out of integrated circuits 中的 integrated circuits。

"持续时间" 是动作发生的时刻。常伴随着介词 during，while，before，after。有时也有单个名词的。如 today。

从以上分析中可见，同一名词组，由于伴随着的介词不同，该名词组角色的作用也就不同了。可见，标识介词是很重要的。反过来，同一标识介词，由于其伴随的名词组不同，意义也不尽相同。例如，介词 by，其名词组角色可以是执行者、或是运输工具，或者地点。但不是工具、原材料或受益人。下面列出几个标识介词与其所伴随的名词组角色的关系。

from→出处

to→目的地

by→执行者或运输工具或地点

with→工具或辅助执行者

for→受益者或持续时间

除此之外，动词组在句子的地位等都对名词组角色有影响。可参阅图 5.34。

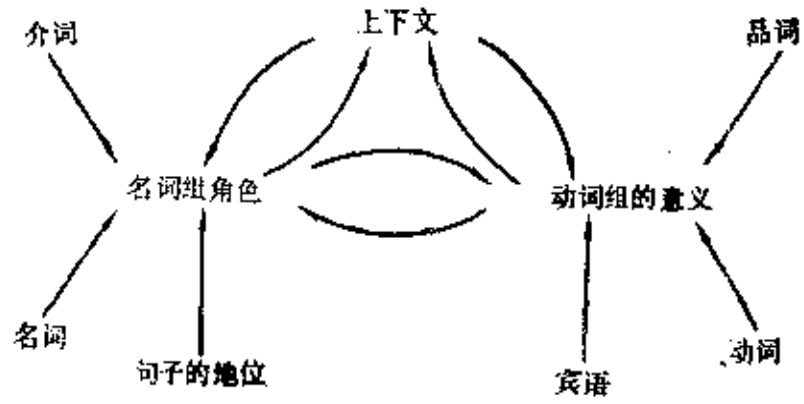


图 5.34 各词组的互相约束

动词的多义性 只要一翻字典，常可发现一个动词有多种意义供选择。人们只能从所涉及的范围、上下文之间的关系以及该句中各种词组的约束当中，选择出一个恰当的意义，至少也要尽可能缩小范围。这是语言理解的关键之一。

首先，上下文对动词意义有约束。也就是说，机器人所处的世界或所交谈的范围一确定下来，动词的许多意义就可删除了。

其次，品词（如虚词、冠词、介词、连接词等）对动词意义的选择大有帮助。在动词组中考察一下动词所伴随的品词，就可将其意义缩小在一个小范围内。举例如下：

He threw some food.（他扔一些食物。）

He threw away some food.（他扔掉一些食物。）

He threw up some food.（他呕吐一些食物。）

She picked up some candy.（她拾起一些糖果。）

She picked out a nice assortment.（她掏出一颗好什锦糖。）

再有，不同的宾语对动词的意义也有约束。例如，

He shot the rabbit.（他射兔子。）

He shot the picture.（他拍照片。）

当然，还有许多因素对动词的意义有很大的约束。可参看图 5.34。

为方便起见，假设所有的名词组对描述动词的动作都是有帮助的。而且忽略那些名词组修饰其他名词组的情况。那么对简单的陈述句进行句法分析之后，其语义分析的粗略步骤如下：

第一步，寻找句子中的主要动词。而后就所对答的范围内选出该动词可能存在的多种意义。特别要注意，品词的存在与否，对于删除某些动词意义是很有帮助的。

第二步，寻找动作对象。

如果动词不是被动态，则从没有标识介词的名词组（往往位于动词之后）中寻找动作对象。

如果动词是被动态，则较有可能是，语义上的动作对象就在动词的前头。这就是语法教科书上称之为主语的。

如果动词是及物的话，较大可能是动作对象跟在动词后面。如果在动词后面仅有一个没有标识介词的名词组，假设它不是时间修饰词的话，则它是动作对象。通常，时间名词组是很容易从其构词识别出来的。在时间和地点名词组挑出之后，如果有二个没有标识介词的名词组跟在动词后面的话，则要吗第一个是动作对象，要吗第二个是动作对象。而且较大可能是第二个为动作对象。

如果动词是不及物的话，则不要求有动作对象。如果又有如上的名词组，则判定这种情况是错误的。

动作对象在握的话，就再有一个机会把那些与该动作对象有矛盾的动词意义除去。

第三步，注意那些没有标识介词的名词组。残存的动词意义可以提出它们需要什么名词类，在什么位置上需要它等等，来指定选择的优先权。例如，许多及物动词要求一个明显的执行者，而且它要在句子的主语位置上。这种动词附带要求，对于那些除动作对象之外所找到的一、二个没有标识介词的名词组，大概有足够条件指定其角色了。

知道了没有标识介词的名词组角色之后，要确定那些有标识介词的名词组角色，就比较简单了。办法是分析一下标识介词在句子中允许导出那些角色。例如，句子中含有带标识介词 by 的名词组。by 就导出要吗执行者，要吗运输工具，要吗地点。如果执行者角色已经由

句法的主语谈及了，则仅留下运输工具和地点两种可能性。

通常，这些残留的动词多义性，可由在名词组中有关词的知识或者由动词的辞典陈述的需要来解决。

最后，某些角色一确定好，也可以帮助解决遗留下来的动词多义性。

举个例子加以说明之。

假设 Robbie 和 Suzie 都是机器人。他们用很简单而有限的英语对话。所涉及的内容如下：

- diseases（疾病）
  - cold（发冷）
  - quinsy（扁挑腺炎）
- time（时间）
  - day（天）
  - hour（点钟）
- things（事物）
  - medicine（药物）
    - aspirin（阿斯匹林）
    - herd tea（汤药）
  - toys（玩具）
    - box（箱子）
    - ball（球）
- people（人物）
  - Suzie（女性机器人）
  - Robbie（男性机器人）

就有关以上范围，动词 take 的意义有如下八种：

TAKE 1　有输送之意。将出现的角色或有出处，或有目的地，或两者皆有。

TAKE 2　有骗取之意。当这个意义是一种打算时，就不存在着目的地角色。只有人才会（不明真象时）受骗。

TAKE 3　有服药之意。药有 aspirin 之类。受益人就是执行者本身。

TAKE 4　有偷盗之意。人是不会被偷盗走的。

TAKE 5　有和其他人执行某社会事件之意。常用品词 out。

TAKE 6　有搬迁之意。总是用品词 out。人是不能被搬迁的。

TAKE 7　有接管之意。品词 over 是这意义的预兆。

TAKE 8　有从身上取下什么东西之意。总是用品词 off。

把这些不同意义与名词组组合起来。假设所有的被动句都恰恰有一个没有标识介词的名词组（传统的句法的主语），而且它在动词前面出现。所有的主动句都有一个这种无介词的名词组（在动词前面），而且有一个或二个跟在动词后面（传统的间接和直接宾语）。

下面列出对于不同角色，要求所伴随的标识介词以及填入角色的约束条件。

角　色	标识介词	填入角色的约束
动作对象	……	……
出　处	from	……
目 的 地	to	……
执 行 者	by	必须是人物
运输工具	by	必须不是人物
辅助执行者	with	必须是人物

工　具	with	必须不是人物
受 益 者	for	必须是人物
持续时间	for	必须是时间
原 材 料	out of	必须不是人物
旧 环 境	out of	……
新 环 境	into	……

为了更好地理解，就举几个例子：

Robbie took aspirin. 由于没有品词，故 TAKE 5 至 TAKE 8 全排除。根据词序，而且无别的选择，故 Robbie 是执行者， aspirin 是动作对象。不象是 TAKE 1， 因为没有可作为出处、目的地的名词组。 TAKE 2 也可排除， 因为 aspirin 不是人物， 不能受骗。因此，这句子只有两个意思。即“Robbie 吃了阿斯匹林”或“Robbie 偷走了阿斯匹林”。 到底是那个意思，就要从上下文等来分析。

Robbie took aspirin for Suzie. 通过上一句的分析已知，只有 TAKE 3 和 TAKE 4 是可能的。for 可以标识受益者或持续时间。 但由于 Suzie 不是时间而是人物。照理她是受益者。因为 TAKE 3 中服药的要求是受益者与执行者相同。 所以 TAKE 3 又被排除了。故 Robbie 偷窃阿斯匹林是无疑的了。当然，可能是 Suzie 再三恳求 Robbie 吃药，Robbie 由于精神的力量而为了 Suzie才 吃药的。 Suzie 则是精神上的受益者。但这已超出了本系统的语义分析的深度了。

Robbie took out Suzie。 品词 out 把动词意义限制于 TAKE5 和 TAKE6，即幽会与搬迁。但 TAKE6 要求动作对象是非生命的，所以其意为 Robbie 幽会 Suzie。

Robbie took out the box。 box 是非生命的，因而是搬迁而不是幽会。

Robbie took the ball to Suzie。 一确定 ball 是动作对象，可供选择的就是 TAKE 1（输送）和 TAKE4（偷盗）了。因为有一个确定的目的地角色，所以是 TAKE1 了。即 Robbie 把球拿给 Suzie。

Robbie took Suzie the ball。 ball 又是动作对象了，但是由于没有标识介词，使得Suzie 的角色不清楚。这样，在 TAKE1 和 TAKE4 二类中就无法抉择了。

Robbie took Suzie。 Suzie 是动作对象，可能性是 TAKE1 和 TAKE2，即输送与骗取。因为没有出处和目的地，故可能性是 Robbie 骗了 Suzie。

Robbie took Suzie to the ballet。 由于有目的地，所以不象是骗取之意了。Robbie 带Suzie 到巴蕾舞剧团。

The ball was taken out of town by Robbie by car for a day for Suzie。 由于句子是被动态，ball 就是动作对象了。这就把动词意义限制于 TAKE1 或 TAKE4。这个二义性就保留着。介词 out of 可能标识着或者原材料或者旧环境。一知道原材料是什么，或者知道在此 take 不是一种生产形式，就可解决这二种可能性了。 但在该简单分析中没有提供这类信息。car 这个词，事先没有告诉过，所以它可能是执行者或运输工具， 只好作为疑案留下。但是由于 Robbie 是人物，故他必须是执行者，因而把它填入执行者表中，而迫使car 就一定是运输工具了。最后，应用通常的约束来检验一下，就可知 a day 和 Suzie 就是持续时间和受益者了。

5.6.3 对答

简单对答 语义分析之后就可作一些简单对答了。只要把句子的角色结构做出来，就可回答一些简单问题了。例如，这样的一句陈述句

Robbie made coffee for Suzie with a percolator。把其中四个名词组在句中的角色确定好。

动作对象 → coffee
执 行 者 → Robbie
工　　具 → a Percolator
受 益 者 → Suzie

现在对这句如果提出四个有关问题："What was made?"、"Who made it?"、"With what was it made?"以及"For whom was it made?"，如何回答得出来？对于疑问句也要进行句法分析和语义分析，搞清其问的主要内容（其对应的角色）。这只要找出疑问词 who、whom、whose、what、which、when、where 等以及其可能有的标识介词（称之为**疑问词组**）之后，就可确定疑问词在疑问句中的角色。然后再到原来句子中找出相应的角色，这就是欲回答的关键词汇了。例如，

问　　题		所问角色		关键词
What was made?	→	动作对象	→	coffee
Who made it?	→	执 行 者	→	Robbie
With what was it made?	→	工　　具	→	a Percolator
For whom was it made?	→	受 益 者	→	Suzie

骨架形式 把机器人所处的世界或交谈所涉及的范围用骨架形式加以表示，对于所提的问题可以从骨架中找出其有关的内容加以回答。

例如，机器人所处的世界如图 5.35 所示的积木世界。为简化起见，只考虑积木块之间的支撑关系，不考虑其前后、左右等关系。每个积木块用一个简单代码表示之。就可用图 5.36 所示的骨架形式表示出来。

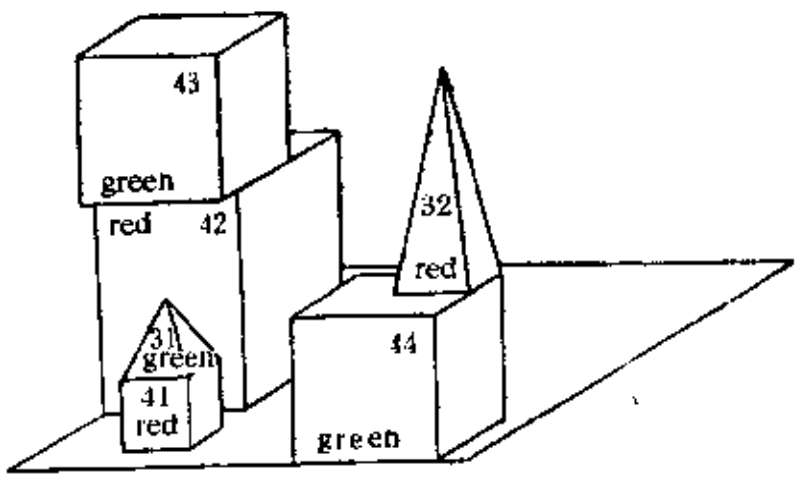


图 5.35　机器人所处的简单世界

现在如果要问两个问题：

What cubes are red?

How many cubes are red?

为要回答这两个问题，可从图 5.36 中找出 red 和 cube 二个节点，指向它们的分别是 41，42，32 和 41,42,43,44。因此，第一个问题的答案是 41,42。第二个问题的答案是 2。

模式匹配 这样的骨架形式可以存入机器中，形式如下：

41　SUPPORTS　31　(5.12)
42　SUPPORTS　43　(5.13)

44	SUPPORTS	32	(5.14)
31	HAS-COLOR	GREEN	(5.15)
32	HAS-COLOR	RED	(5.16)
41	HAS-COLOR	RED	(5.17)
42	HAS-COLOR	RED	(5.18)
43	HAS-COLOR	GREEN	(5.19)
44	HAS-COLOR	GREEN	(5.20)
31	IS-A	PYRAMID	(5.21)
32	IS-A	PYRAMID	(5.22)
41	IS-A	CUBE	(5.23)
42	IS-A	CUBE	(5.24)
43	IS-A	CUBE	(5.25)
44	IS-A	CUBE	(5.26)

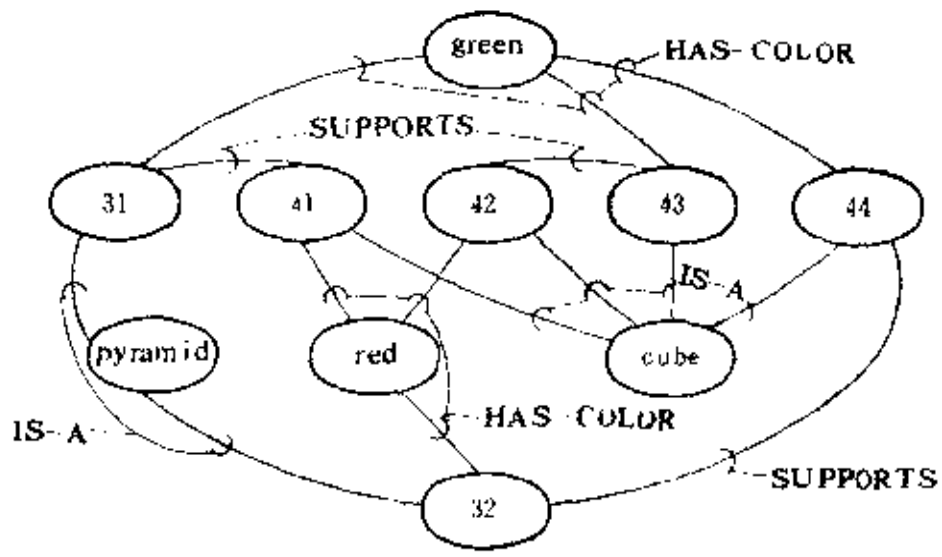


图 5.36 简单的积木世界的骨架表示形式

对于上面所提的两个问题，可以转换成如下程序：

```
FIND  ALL   X SUCH    THAT
      ? X   HAS-COLOR   RED          (5.27)
      ? X   IS-A        CUBE         (5.28)
```

再把式 (5.27) 及式 (5.28) 所表示的问题模式，与式 (5.12) ～式(5.26) 所表示的模式进行匹配检查。除了“?X”部分可不相同以外，其余部分应完全相同。如果确实达到这个要求，则称该两模式匹配。可见，式(5.27)与式(5.16)、式(5.17)、式(5.18)相匹配；而式 (5.28) 与式 (5.23) 、式 (5.24)、式 (5.25) 、式 (5.26)相匹配。式 (5.27)的答案是 32,41,42；式 (5.28) 的答案是 41,42,43,44。因此，对第一个问题的答案是 41,42 。而对 “How many” 问题的答案，则不需具体列举，只需将其数目累加即可。

5.6.4 专家系统

以上只叙述对答系统的很基本的功能。一个对答系统还必须将机器人做过的事情（或谈过的话）及其过程加以记忆。如果现场改变了状态的话，还需有修改能力。还要有推论能

力，如从局部推论到全局，从全局推论到局部。例如，“一个人有两只手。”“凡是人都有两条腿。”这二个事实给定之后，则可回答如下问题：“五个人有几只手？”及“张三是人，他有几条腿？”。

从本节可知，即使很简单的对答系统，也要求有三个专家系统：

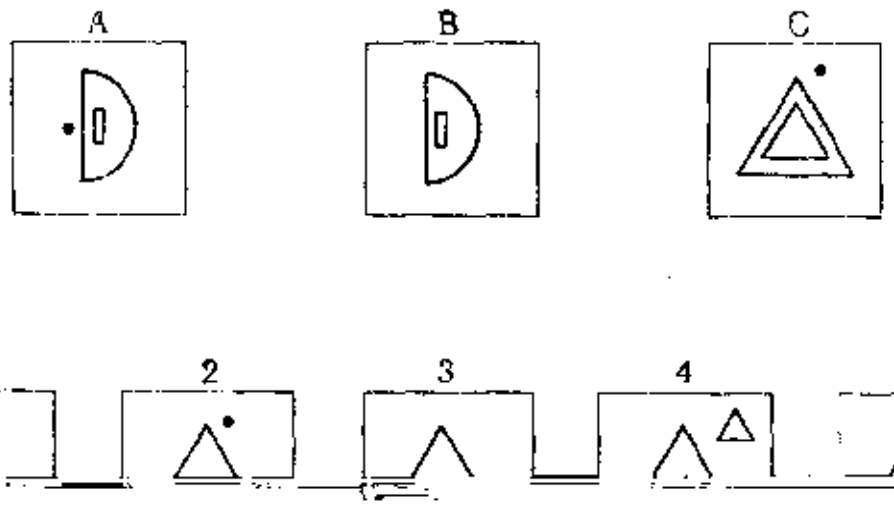
(1) **句法专家** 句法专家对句子进行句法分析。

(2) **句子专家** 句子专家是进行语义分析的。

(3) **编剧专家** 编剧专家必须理解单独的句子与其他句子的关系以及与它们一起交谈的一般故事情节的关系。即使在最简单的情节中，也必须理解句子组是怎样传达含意、唤起期望、回忆、建立上下文、制造情绪和发展情节。

5.7 类推（类比）

在图 5.37 中，如果已被告知，从源图 A 变成终图 B 的话，现欲从图 1 至图 5 中找出一个图 X 来，使得图 C 变至图 X 的规则与图 A 变至图 B 的规则相匹配。 虽然图 A 和图 B 的

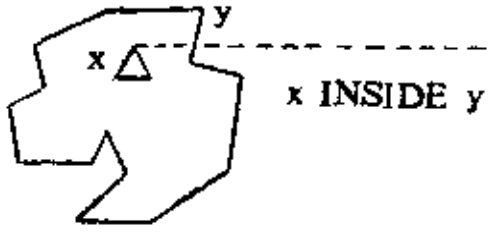


的描述。它有如下三个部分：

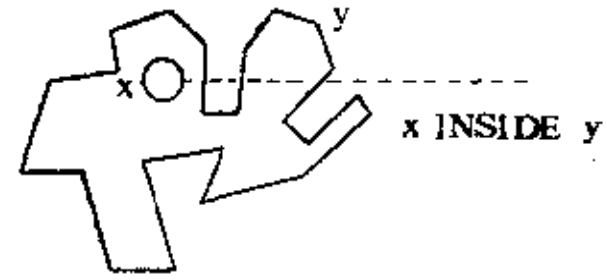
(1) 源图的描述。即在源图中子图之间的关系。

(2) 终图的描述。即在终图中子图之间的关系。

这些关系可能有 ABOVE(在…上)、LEFT-OF(左)、INSIDE(在…内)。两子图形的 ABOVE 与 LEFT 关系是这样确定的：通过其中一子图(X)的区域中心画与水平线成 45° 的一对斜线（如图 5.38 所示)作为判别准线。在此，关系 UNDER(在…下)和 RIGHT-OF(右)可不必取用，因为 X UNDER Y，X RIGHT-OF Y 可分别由 Y ABOVE X, Y LEFT-OF X 来代替。



欲确定互不邻接的两子图 X、Y 的 INSIDE 关系，可以从其中某一子图 X 上任一点沿任一方向虚构出一条射至版图边界的半射线，如图 5.39 所示。如果该虚构的线与另一子图 Y 相交的次数为奇数次，则 X INSIDE Y；如是偶数次，则没有 INSIDE 关系。



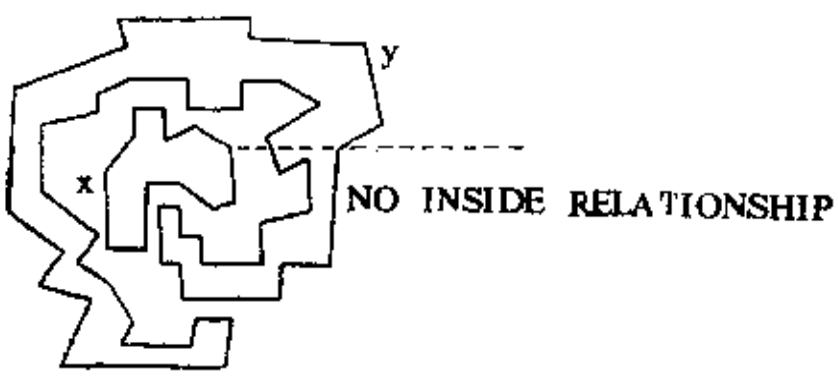


图 5.39 INSIDE 关系的确定

(3) 在源图中的子图如何变至在终图中的子图之描述。比较典型的变换有：**变大变小** (SCALE CHANG)，如图 5.40 中的放大、缩小；**旋转** (ROTATION)，如图中的正方形被旋转了 45°；**反射** (REFLECTIO-N)，如图中的子图 B 被以 Y 轴为对称地对称变换一下，之后又以 X轴为对称轴地被对称变换一次；或者以上三种的组合；**增添** (ADD) 子图；**删去** (DELETED) 子图，如图中的子图 B 被添上了子图 A，而后又删去子图 B，最终留下子图 A。

有了以上三种描述，就可找出最佳匹配了。如图 5.41 所示的源图 A、终图 B 及图 C、图 1、2、3，现欲从图 1～图 3 中找出图 X，使得 C→X 与 A→B 为最佳匹配。先把子图间的三种描述关系列出如下：

规则	部分 1			部分 2			部分 3	
A→B	*l*	ABOVE	*m*	*m*	ABOVE	*n*	*l*	DELETED
	l	ABOVE	*n*				*m*	SCALED 1 : 1
	n	INSIDE	*m*					ROTATED 0°
							n	SCALED 1 : 2
								ROTATED 45°

C→1	x	ABOVE	y	y ABOVE z	x	DELETED
	x	ABOVE	z		y	SCALED 1:1
	z	INSIDE	y			ROTATED 0°
					z	SCALED 1:2
						ROTATED 45°
C→2	x	ABOVE	y	z ABOVE y	x	DELETED
	x	ABOVE	z		y	SCALED 1:1
	z	INSIDE	y			ROTATED 0°
					z	SCALED 1:2
						ROTATED 45°
C→3	x	ABOVE	y	x ABOVE z	y	DELETED
	x	ABOVE	z		x	SCALED 1:2
	z	INSIDE	y			ROTATED 0°
					z	SCALED 1:2
						ROTATED 45°

其中，规则部分 1 是关于M→N 的M 部分的描述；规则 部分 2 是 关 于M→N 的 N部分的描述；规则部分 3 是关于M→N 的变换关系的描述。

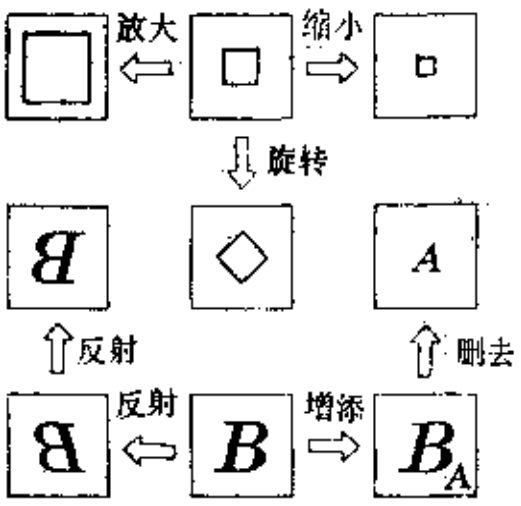


图 5.40　图形的几种变换

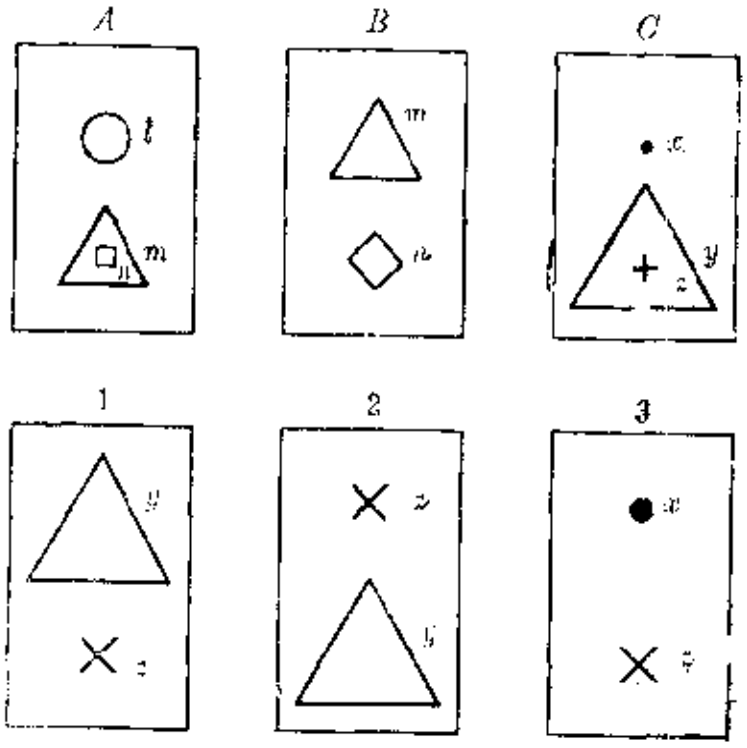


图 5.41　类推的 例子

现在，只要关于 l、m、n 与 x、y、z 的对应关系寻找得恰当，使得规则部分 1 至规则部分 3 都相匹配的话，最佳匹配就找到了。在此，只需 l 与 x 对应，m 与 y、n 与 z 对 应。 那么A→B 与 C→1 的三个规则部分就可匹配了。其余的 C→2 、C→3 就不能匹配。

5.8　简单概念的学习

教师为了教给儿童一个新概念，除了要举出一些正确的例子以外，还需举出一些似是而非的例子。只有这样，才能给儿童树立完整的正确的概念。如何使计算机也能象儿童一样，

在教师（即人）的一些正确与似是而非的例子的教育中，树立正确的概念。这是智能模拟的课题之一。这种**概念学习**的学习方式称为**有教师的学习**；这种机器称为**可教育的机器**。P.H.Winston 在这方面做了一些工作。

机器在得到一个景物模型时，就把它表示为附有一般性连接概念的骨架形式的网络，把这网络作为模型。而后，每接受一个“似是而非”或正确的例子时，就把这些例子也表示为骨架网络。再把它与旧模型加以比较，适当修改旧模型的连接概念而成强调性的连接概念，从而建立新模型。这就是机器的学习过程。举例来说明之。

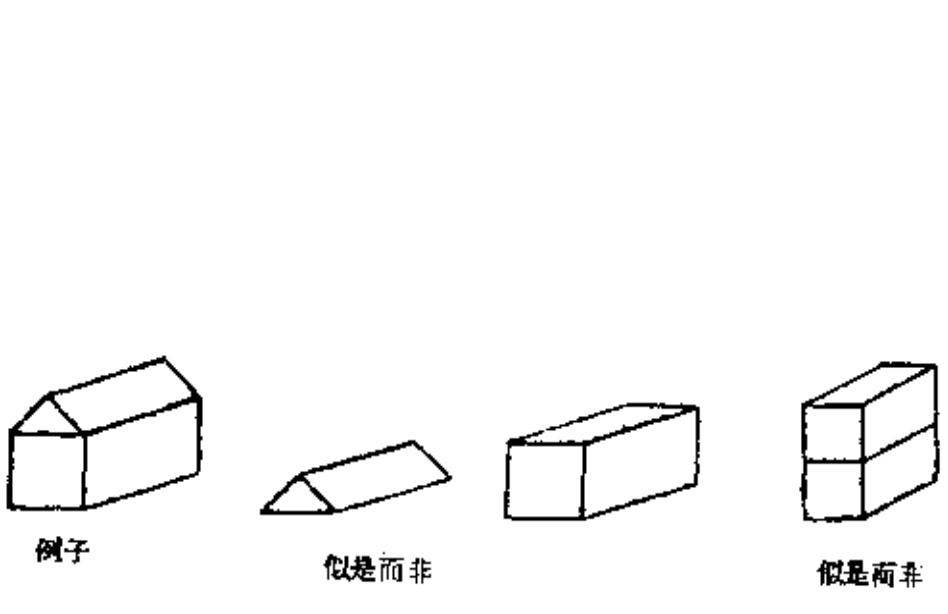


图 5.42　房子概念的学习序列

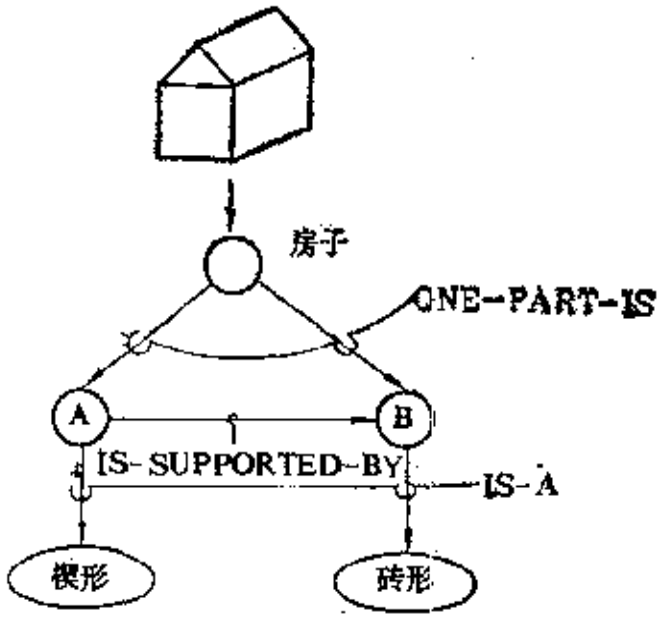


图 5.43　“房子”的骨架网络

假设要让机器掌握“房子”这个简单的概念。如图5.42所示，先教给它一个正确的样品图案，而后教给它二个似是而非的图案。机器则按如下方法建立、修改模型。

首先，根据“房子”的概念，建立如图 5.43 所示的骨架网络，作为模型。房子的某一部分是 A 或 B，所以节点“房子”与 A、B 节点的连接概念是 ONE-PART-IS。而 A 由 B 支撑着，故节点 A 与节点 B的连接概念是 IS-SUPPORTED-BY。A 的图案是一个楔形、B 的图案是一个砖形，故连接概念为 IS-A。所有这些连接概念都是**一般性连接概念**。

当接受到第一个似是而非的例子时，机器先把它表示为网络（如图 5.44 的右上角所示），而后与刚建立的模型（如图 5.44 的左上角所示）加以比较。可以发现，其不同点在于 A 由 B 支撑着的连接概念没有了。因此，必须对这个连接概念加以强调。故修改为 MUST-BE-SUPP-

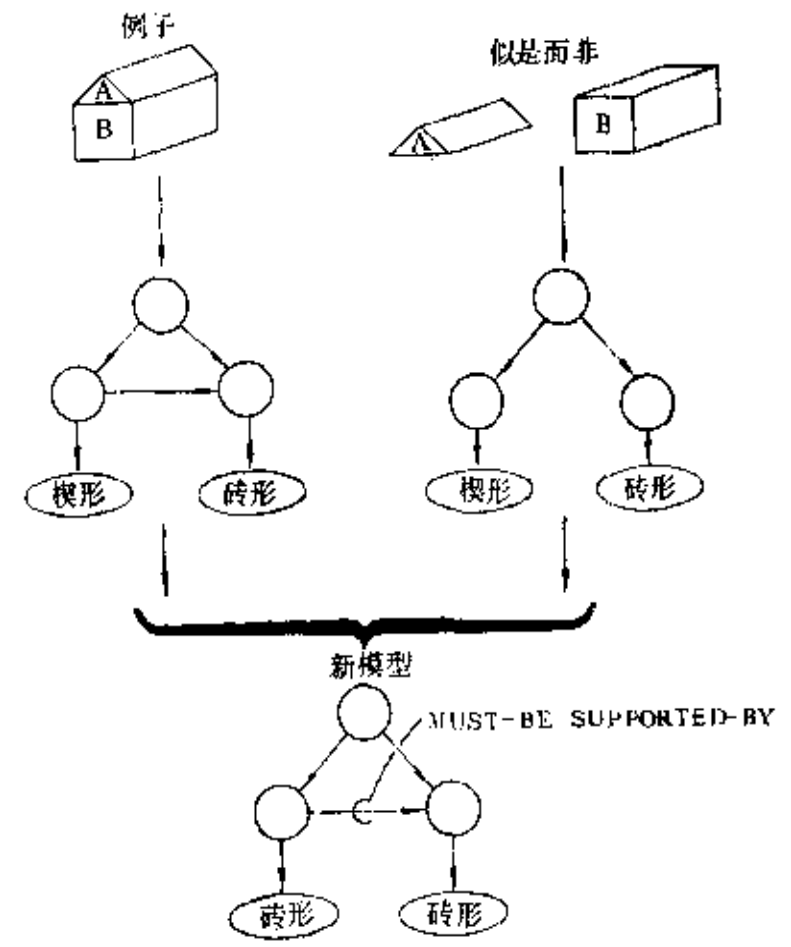


图 5.44　第一次学习

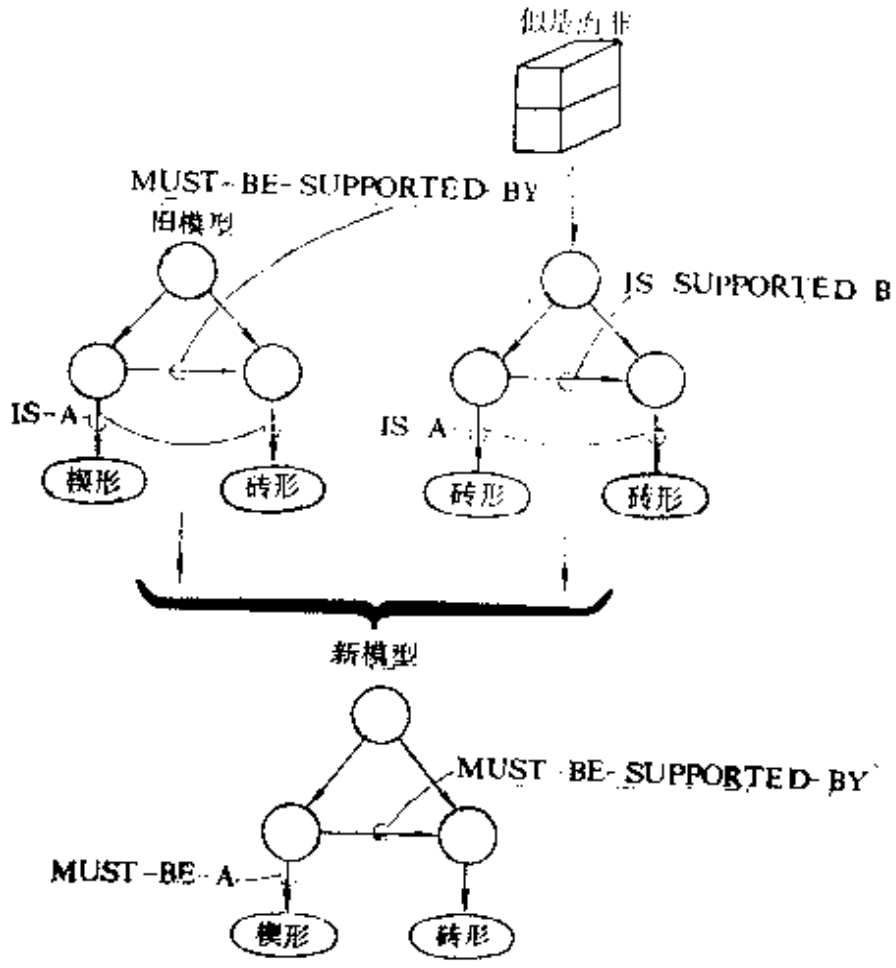


图 5.45 第二次学习

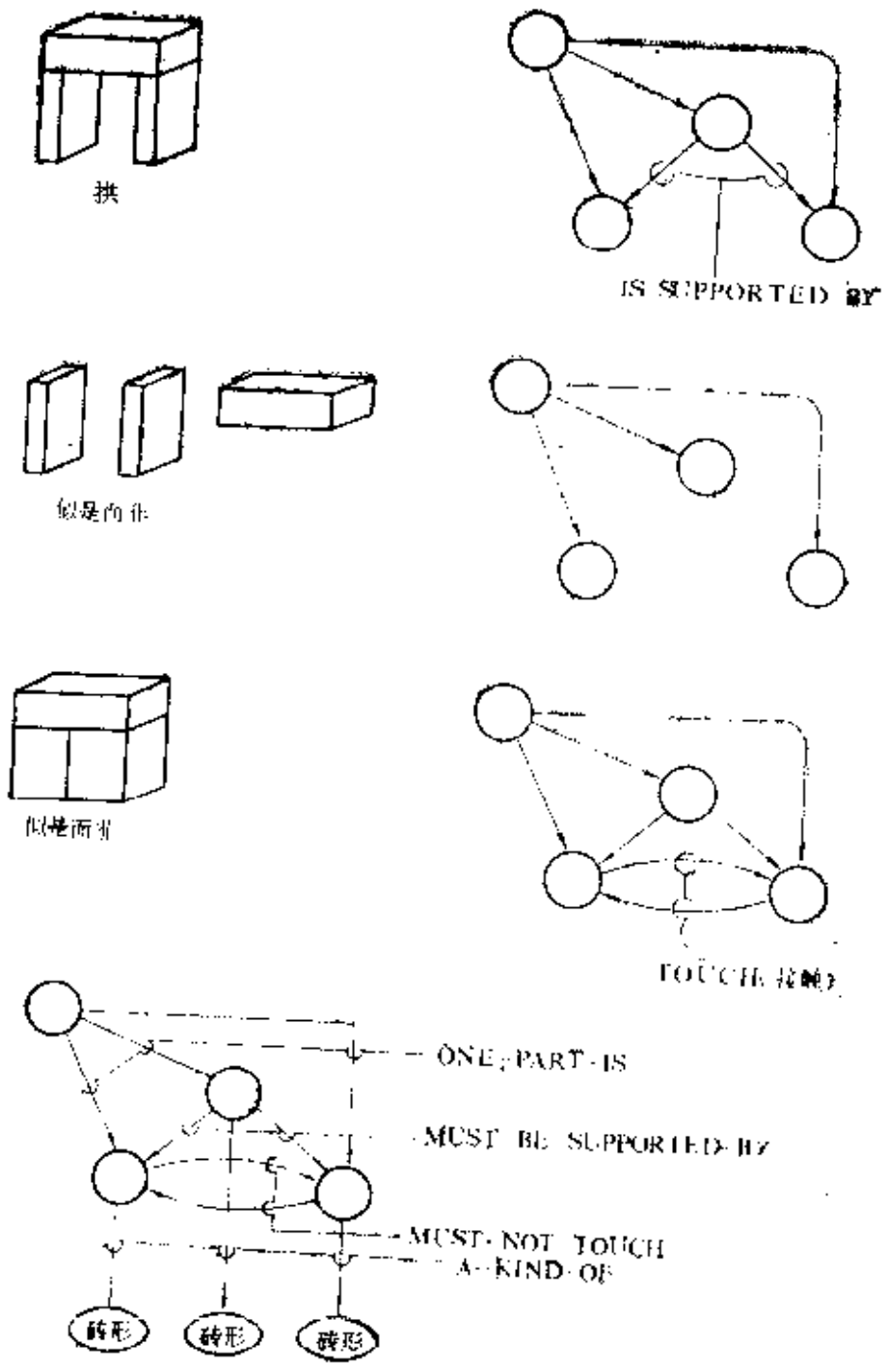


图 5.46 “拱”概念的学习

ORTE-BY。这是一个**强调性连接概念**。。从而产生新模型（如图中的下半部分所示）。

同样地，接受到第二个是似而非的例子时，就可象图 5.45 所示那样加以修改。

在此例子中砖形 B 上的东西必须是楔形。所以用 MUST-BE-A 的强调性连接概念。但在某些场合下，只要是属于某一类的东西都行。因此，把连接概念IS-A，MUST-BE-A 都改为连接概念 A-KIND-OF。但强调性连接概念的 KIND 比起一般性连接概念的 KIND 来说，所属范围更广了。现在就以“拱”为例来说明之。

用拱的一个样品及二个似是而非的例子教给机器，机器就建立了一个模型，如图 5.46 所示。图中底部的模型就是学习之后的模型。

如果再用另一个正确的样品教给机器的话，机器就要用其他方法来修改模型。如果正确的样品与旧模型不同的话，说明模型中原来的约束条件要放宽或加严。

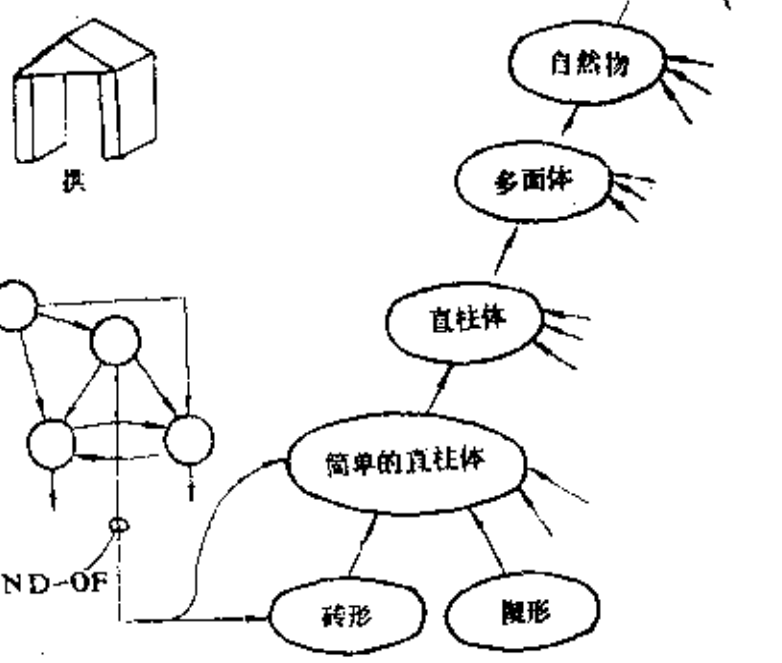


图 5.47　类别的修改

如图 5.47 所示，旧模型中的拱顶，原来是砖形之类物体（如图中虚线箭头所表达）。由于新样品（图中左上角所示）的拱顶是楔形，就引入新条件（如图中的右下“楔形”部分）。拱顶的约束条件放宽了，变为“简单的直柱体”。因而拱顶的 A-KIND-OF 箭头修改为指向“简单直柱体”框。该框的其它箭头表示引起放宽约束条件的内容。根据这个思路，可能在其他概念中会放宽至“事物”（包括有形与无形）为止。描述之间的类别差异就是一种学习过程。

把以上过程加以系统化。先看一个简单例子。旧模型与所教的“似是而非”例子或新的正确的样品加以比较，找出其间的相似与差异来。如图 5.48 所示。图中的虚线表示对应的节点。从中可发现两骨架网络的差异。把两骨架网络之间的差异归结为如图 5.49 所示的几类。那么两景物之间的差异可由如下五种关系来表达：SUPPLEMENTARY-POINTER（**增补指针**）、SATELLITE-PAIR（**伴偶**）、INTERSECTION（**交叉**）、MERGE（**合并**）及 EXIT（**退出**）。

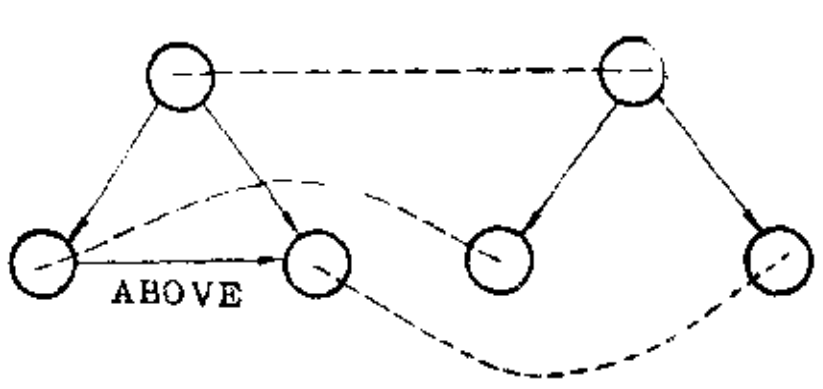


图 5.48　差异的简单例子

对于这些差异，机器就可能要对模型执行一些修改操作。下面就列出教给一个似是而非的例子时，机器面对所出现的不同格局而采取哪些措施。

格　　局	措　　施
在模型中出现“增补指针”或“退出”	用 MUST-BE 替换指针说明
在似是而非例子中出现“增补指针”或“退出”	用 MUST-NOT-BE 替换指针说明
在模型中出现“增补指针”或“退出”；而指针说明是否定的。	用 MUST-NOT-BE 替换指针说明
在似是而非例子中出现“增补指针”或“退出”，而指针说明是否定的。	在模型中替换掉 MUST-BE 说明
否定的“伴偶”。	用 MUST-BE 替换指针说明
MUST-NOT-BE“伴偶”。	不做什么
在模型的节点上“合并”。	对似是而非的节点用 MUST-NOT-BE 替换指针说明
在似是而非例子的节点上“合并”。	用 MUST-BE 替换指针说明
其它的“合并”，第一个选择。	对似是而非的节点用 MUST-NOT-BE 替换指针说明
其它的“合并”，第二个选择。	用 MUST-BE 替换指针说明

如果教给一个正确的样品，机器则根据下面所示的不同格局采取相应措施。

格　　局	措　　施
在模型中出现“增补指针”或“退出”	摘下指针
在样品中出现“增补指针”或“退出”	不做什么事
在模型中出现“增补指针”或“退出”而指针说明是MUST-BE。	矛盾
在模型中出现“增补指针”或“退出”而指针说明是 MUST-NOT-BE。	不做什么事
否定的“伴偶”	摘下指针
MUST-NOT-BE“伴偶”	矛盾
“合并”，第一个选择。	把指针移至“合并”点上
“合并”，第二个选择。	摘下指针

一旦做了这些操作而把旧模型变为新模式，就是机器的所谓学习了。

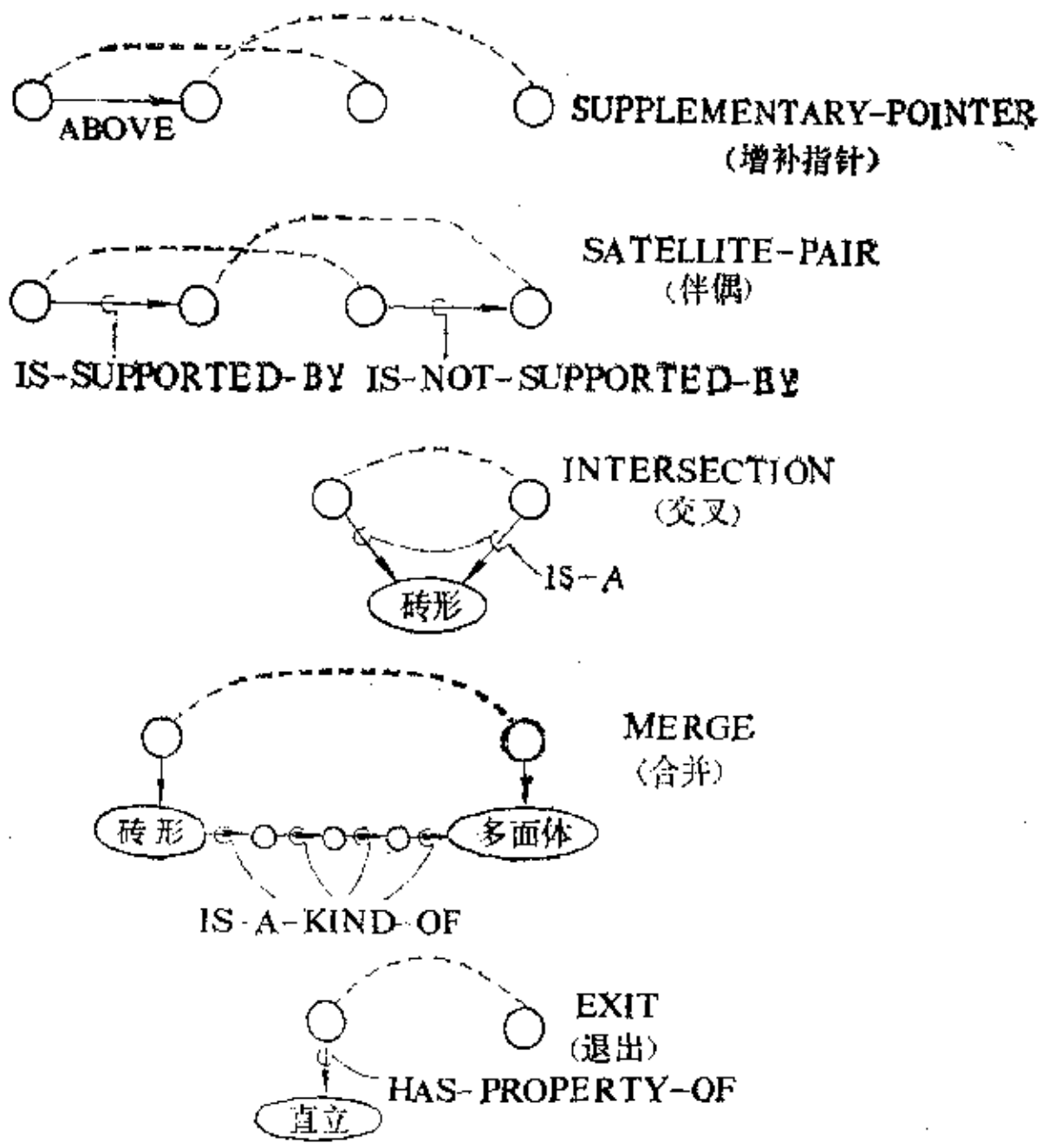


图 5.49　差异的种类

第五章　习　　题

一、由McCulloch-Pitts 神经元模型来实现如下命题逻辑：

(1)$\sim(A\wedge B)$，　(2)$\sim(A\vee B)$，　(3)$(A\vee B)\wedge C$，

(4)$A\rightarrow B$，　(5)$A\leftrightarrow B$，　(6)$A\rightarrow\rightarrow B,C$。

二、试用 Roberts 和 Guzman 方法对如下多面体进行识别。

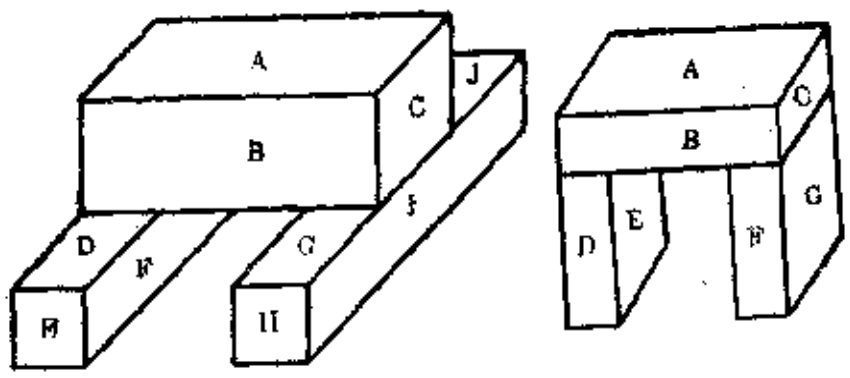


图 5.50　二个多面体

三、用 Clowes、Huffman 及 Waltz 方法识别如下多面体。

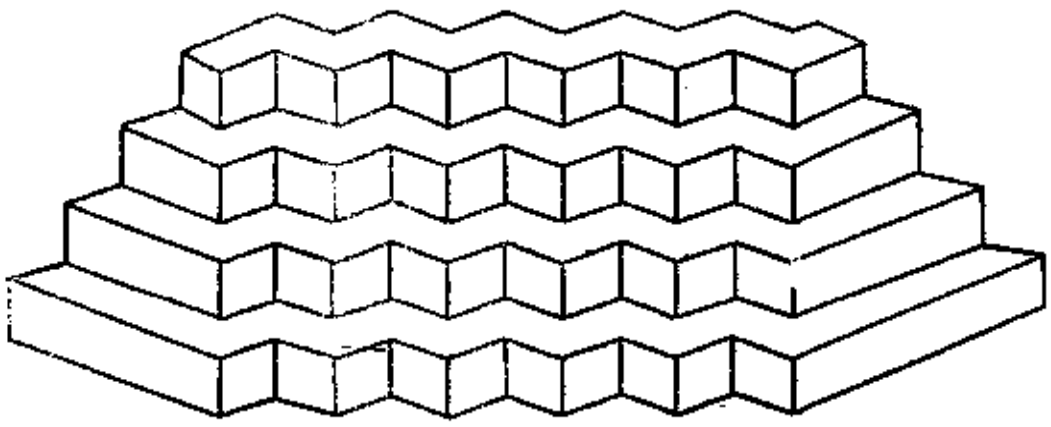


图 5.51　多面体

四、试作一个语义分析练习，从分析名词组角色、动词 turn 的多义性等方 面着 手，而后分析几个具体的简单句。设所涉及的内容如下：

- 人物 (sb.)
  - Robbie
  - Suzie
- 事物
  - 划船 (to boat)
  - 灯 (light)
  - 街角 (street corner)
  - 船 (boat)
  - 不幸事情 (unfortunate event)
  - 求救 (for help)
  - 门 (door)
  - 休息 (rest)
- 动作
  - 关 (灯) (turn off)
  - 沿…拐弯(turn……)
  - 进入到…(turn into)
  - 返回(turn back)
  - 进入(turn in)
  - 开 (灯) (turn on)
- 人物间的动作
  - 劝说 (turn sb. to(verb))
  - 求救 (turn to sb.(for help))
  - 发怒 (turn on sb.)
- 事故
  - 突然发生 (turn up)
  - 翻船(turn over)
- 身体状况
  - 头昏(sb.'s head turns
  - 恶心、呕吐(sb.'s stomach turns)

对如下几句英语进行语义分析：

Robbie turns Suzie to boat.

Robbie turns off the light.
Robbie and Suzie turn a street corner.
Robbie and Suzie turn into a boat.
A unfortunate event turns up.
The boat turns over.
Suzie turns to Robbie for help.
Robbie's head turns.
Suzie's stomach turns.
They turn back.
They turn in at the door.
Robbie turns on the light.
Suzie turns on Robbie.
Robbie turns Suzie to rest.

五、如图5.52所示。设从源图A变至终图B。试从图1至图5中找出一个图X来，使得图C变至图X的规则与之相匹配。

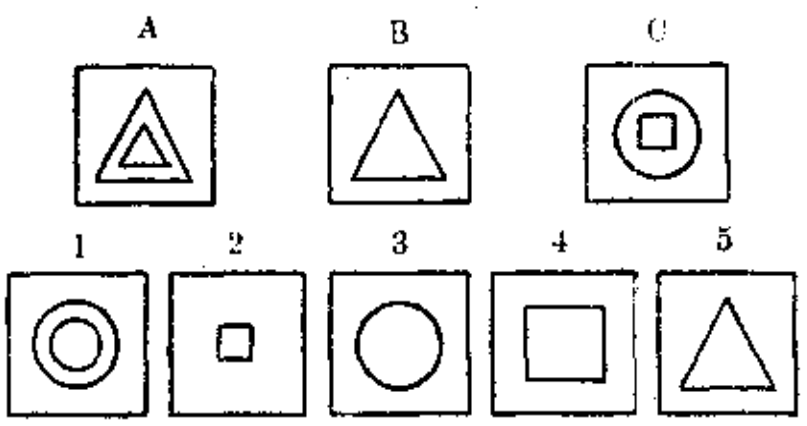


图 5.52　类推的图

六、试做一个“三轮车”的简单概念学习的模型。先教给它一个正确样品——前面有一个轮子、后面有二个轮子的三轮车。第二次教给一个似是而非的例子——二个轮子的自行车。第三次教给另一个似是而非的例子——前后有二个轮子、后面有二个轮子的四轮车。最后教给它一个正确的样品——前面有二个轮子、后面有一个轮子的三轮车。

参 考 文 献

1. Chin-Liang Chang, Richard Char-Tung Lee; “Symbolic Logic and Mechanical Theorem Proving”, ACADEMIC Press, 1973.
2. Duda R. O., Hart P. E.; “Pattern Classification And Scence Analysis”, Wiley-Interscience 1973.
3. Glorioso R.M.; “Engineering Cybernetics”, Prentice-Hall, Englewood Cliffs, New Jersey 1975.
4. Hopcroft J.E., Ullman J.D.; “Formal Languages and Their Relation To Automata”, Addsion-Wesley Publishing Company, 1969.
中译本：莫绍揆、段详、厄秀芬译：《形式语言及其与自动机的关系》，科学出版社，1979.
5. Hopkin D., Moss B.; “Automata”, MACMILLAN Press, 1976.

6. Hunt E.B.: "Artificial Intelligence", ACADEMIC Press, 1975.
7. Fukunaga K.: "Introduction To Statistical Pattern Recognition", ACADEMIC Press, 1972.
中译本：福永圭之介著，陶笃纯译：《统计图形识别导论》，科学出版社，1978.
8. K.S.Fu: "Digital Pattern Recognition》,Spring-Verlage, Berlin Heidelberg New York, 1976.
9. Lightstone A.H.: "Mathematical Logic——An Introduction To Model Theory", Plenum Press, New York,1978.
10. Manna Z.: "Mathematical Theory Of Computation", McGraw-Hill, 1974.
11. Nilsson N. J. : "Problem-Solving Methods In Artificial Intelligence", McGraw-Hill, 1971.
12. Winston P.H.: "Artificial Intelligence" Addison-Wesley, 1977.
13. 辻三郎：知能ロボット，《数理科学》，1976,9
14. 白井良明编：《ロボット》，共立，1976.
15. 尾崎弘，白川功，翁长健治："グラフ理论"，コロナ社，1975.
16. 马希文：机器证明及其应用，《计算机应用与应用数学》，1978.4～1978，8.
17. 涂序彦、郭荣江：智能控制及其应用，《自动化》，1977 年 1 卷 1 期.

词汇索引（以拼音字母为序）

D

E

F

G

H

J

K

L

M

T

W

X

Y

Z

α

β

α

β

α

β

α

β

α

β

α

β

α

β

α

β

α

β

α

β

α

β

α

β

α

β

α

β

α

β

α

β

α

β

α

β

α

β

α

β

α

β

α

β

α

β

α

β

α

β

α

β

α

β

α

β

α

β

α

β

α

β

α

β

α

β

α

β

α

β

α

β

α

β

α

β

α

β

α

β

α

β

α

β

α

β

α

β

α

β

α

β

α

β

α

β

α

β

α

β

α

β

α

β

α

β

α

β

α

β

α

β

α

β

α

β

α

β

α

β

α

β

α

β

α

β

α

β