计 算 机 科 学 丛 书

人工智能

Artificial Intelligence
A New Synthesis

(美) Nils J. Nilsson 著
郑扣根 庄越挺 译
潘云鹤 校

机械工业出版社
China Machine Press
MORGAN KAUFMANN

人工智能 Artificial Intelligence: A New Synthesis
(美) Nils J. Nilsson 著
机械工业出版社
MORGAN KAUFMANN

计算机科学丛书

人工智能

(美) Nils J. Nilsson 著

郑扣根 庄越挺 译

潘云鹤 校

机械工业出版社
China Machine Press

本书从一个新颖的角度对人工智能各方面的问题进行了探讨。由浅入深地介绍了整个人工智能系统和agent的发展历程。首先，描述了仅能对周围环境中可感知特征做出反应的原始agent，以及它们所涉及的机器视觉、机器学习和机器进化等问题；然后，逐步介绍了agent 可以从无法立即感知的任务环境中获取信息的技术。本书不仅是对人工智能技术的介绍，而且能为人工智能的研究提供参考和建议。

本书作为人工智能的入门教材，适合所有对人工智能这门学科感兴趣的读者参考，尤其适合大专院校的计算机专业及相关专业的学生用做教材或教学参考书。

本书版登记号：图字：01-2000-1856

图书在版编目（CIP）数据

人工智能/（美）尼尔森（Nilsson, N. J.）著；郑扣根等译. -北京：机械工业出版社，2000.9

（计算机科学丛书）

书名原文：Artificial Intelligence: A New Synthesis

ISBN 7-111-07885-3

I. 人… II. ①尼 … ②郑… III. 人工智能 IV. TP18

中国版本图书馆CIP数据核字（2000）第30597 号

机械工业出版社(北京市西城区百万庄大街22号 邮政编码 100037)

责任编辑：陈贤舜

北京第二外国语学院印刷厂印刷 · 新华书店北京发行所发行

2000年9月第1版第1次印刷

787mm × 1092mm 1/16 · 20.5印张

印数：0 001-5 000册

定价：30.00元

译 者 序

人类总是在不断深入地研究自然界，最复杂的研究对象便是人类本身。人工智能在20世纪50年代诞生并随之兴起，此后掀起了用机器来研究与模拟人类思维的阵阵热浪。人工智能的发展历史反复经历了高峰和低谷的转换。起初，研究者们对之极为乐观，期望在不远的将来，人类所创造的机器能够成为作曲家、艺术家、工程师和象棋大师等。在这种目标的激励下，人工智能的研究热潮一浪高过一浪，很快在问题求解、博弈、演绎逻辑及机器自动证明理论和技术等方面有了突飞猛进的发展。但是人类智能之复杂及计算机软硬件之局限，决定了人工智能的发展道路崎岖不平。过高的期望也给许多人带来了失望。跳棋程序到了一定程度水平难以再提高、通用解题程序遇到了难以逾越的困难、归结方法难以克服“组合爆炸”等等，人工智能的研究曾一度陷入低谷。但近几年来，通过人工智能研究者们的不断争论、探索和创新，并随着相关领域、特别是计算机技术的飞速发展，人工智能又进入一个新的发展期，一些新技术、新观念被集成到这个领域，本书即是作者在此背景下撰写而成的。

本书的作者Nilsson教授是人工智能研究中逻辑学派的奠基人之一，他早期的一些人工智能著作堪称为该领域的经典，曾经在全世界许多国家被用做大学的教材。本书是Nilsson教授在综合了当前人工智能领域的最新研究、计算机及相关领域的最新发展，增添了大量新的研究内容之后撰写完成的。本书取材新颖，概念清楚，通俗易懂，在广度、深度及先进性方面都作了综合的考虑。与作者的其他著作风格不同的是，在每章的最后一节均附有“补充阅读和讨论”，系统地总结了与所讨论的章节相关的参考文献和评论，为读者进一步深入了解书中的基本原理提供了很大的帮助。

本书的翻译力求忠于作者原意。考虑到本书可作为一本人工智能的基础教材，我们在许多人工智能的专业术语后面的括号内注上英文原文。这一方面是为了方便读者能对照理解，为其以后的学习打下基础；另一方面也为了避免以往就存在的不同中文译法而带来的歧义，以节省读者的宝贵时间。另外，本书频繁地使用了“agent”这一术语，书中可译为“智能体”或“智能主体”，但由于“agent”这一单词近几年在人工智能和计算机界使用非常广泛，且大都沿用原文，故我们在本书中也按此习惯，未将其译为中文。译者希望本书的翻译既能吸引更多的读者进入这个领域，同时也能给人工智能研究者们提供知识的更新和补充。

本书的翻译由郑扣根和庄越挺合作完成，全书由潘云鹤教授校阅。

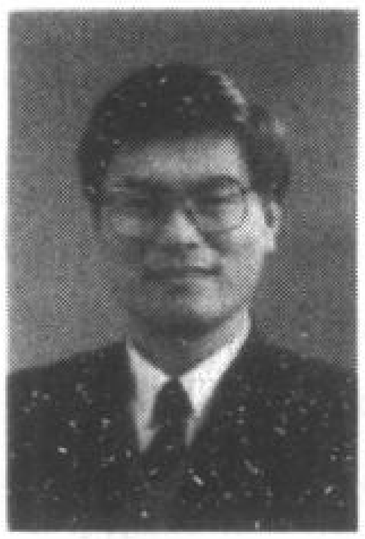
在本书的翻译过程中，得到了马社亮、田鹏、任海波、田稷等同志的许多帮助，在此表示深深的谢意。同时也非常感谢机械工业出版社的编辑们给予我们耐心的等待和支持。

由于种种原因，书中错误和不妥之处在所难免，恳请读者批评指正。

译 者

2000年4月

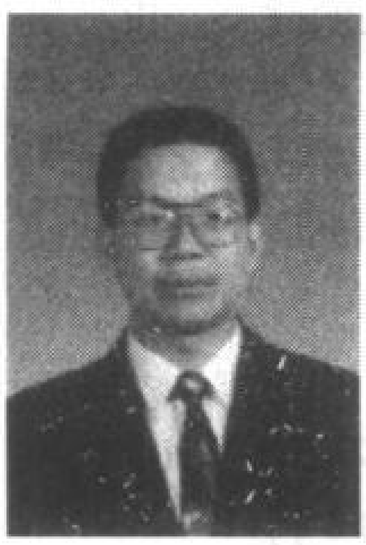
译、校者介绍



潘云鹤，男，1946年11月生，浙江省杭州市人。计算机专家，中国工程院院士，现任浙江大学教授、校长兼研究生院院长。

1970年毕业于上海同济大学建筑学专业。1970年8月至1978年10月在湖北省襄樊市工作，曾任襄樊市自动化研究所所长和市科委副主任。1978年至1981年为浙江大学计算机应用专业研究生，1981年毕业并获硕士学位，后留校任教，1985年9月晋升为副教授。1986年至1988年在美国卡内基—梅隆大学计算机系作访问学者。1990年任浙江大学人工智能研究所所长，1990年8月晋升为教授，1993年被评为博士生导师。1991年9月至1994年7月任浙大计算机系主任，1994年7月至1995年5月任浙江大学副校长，1995年5月起任浙江大学校长。现兼任中国计算机学会理事、中国人工智能学会理事、中国智能CAD/CAM专委理事长、《中国科学》等5种杂志编委等。

郑扣根，男，1964年11月出生于江苏镇江。1986年东北重型机械学院自动控制系本科毕业，同年考入浙江大学科仪系攻读硕士学位，1987年公派至英国Warwick大学继续攻读工程硕士，并于1990年获Warwick大学博士学位。同年至英国Leicester 大学做博士后，1993年12月回国至浙江大学计算机系做博士后，1994年5月晋升为副研究员，1996年1月出站并留在浙江大学计算机系任教至今。主要研究方向为操作系统、人工智能、地理信息系统、并行算法等。



庄越挺，1965年6月出生于浙江慈溪。获浙江大学计算机应用博士学位。现为浙江大学计算机系教授、系副主任、博士生导师，浙江省“151人才工程”第一层次培养人员，曾多次获浙江大学优秀青年教师称号。主要的研究领域为多媒体数据库及信息检索、智能动画、人工智能、CAD 等，主持过国家基础研究“攀登计划”、国家“八五”攻关、国家自然科学基金等10多个项目，并取得丰硕的科研成果，92年获中科院科技进步一等奖，93年获国家科技进步二等奖。共发表50多篇学术论文。1997年2月至1998年8月间，获浙江大学包氏奖学金资助，赴美国伊利诺斯大学Urbana-Champaign(UIUC) 的计算机系和Beckman研究中心作访问学者。

前　言

本书从一个新颖的角度对人工智能(Artificial Intelligence，简称AI)各方面的问题进行了探讨，由浅入深地介绍整个人工智能系统或agent的发展历程。首先，将介绍仅能对周围环境中可感知特征做出反应的原始agent，以及这些简单的机器所涉及的机器视觉、机器学习和机器进化等问题；然后，将逐步介绍使agent可以在无法立即感知的任务环境中获取信息的技术。这些信息可以采用环境状态、环境图标模型、状态空间图和逻辑表示等描述性信息的形式。因为AI的发展历程类似于动物的进化过程，因此我称其为演化人工智能(*evolutionary artificial intelligence*)。希望本书不仅是对人工智能技术的介绍，而且能为研究人工智能提供参考(建议)。为此，书中的例子为人工智能的学习提供激励和基础。

尽管我运用agent来说明人工智能技术，但人工智能技术本身拥有更广泛的应用。许多人工智能研究者的思想已经渗透到计算机科学中，AI已广泛应用于专家系统、自然语言处理、人机交互、信息检索、图形图象处理、数据挖掘和机器人技术(对此将会举例说明)。这里，agent旨在把一系列看似不相关的主题组织到一起。

本书涉及的范围，将力图控制在理论和实践之间的中间地带。这一地带拥有丰富而重要的人工智能的思想，并且，在本书中我将尽力选取并说明那些在人工智能领域中具有经久不衰的价值观点(当然，在选取论题并做出结论时不可避免地会出现遗漏和错误)。同时，在书中将对某些论题进行深入探讨——不仅因为这些论题更加重要，而且我想要在书中提供一些深层剖析的例子。虽然书中出现了伪代码算法，但本书并非人工智能编程的教材("人工智能技术"的书包括：[Shoham 1994，Norvig 1992，Tracy & Bouthoorn 1997])。我并不对所有重要的理论结果给出证明，但会对那些形式证明尽力提供直观的论据和引用。我的目的是为一个学期的大学初级课程提供一本厚度适中的入门教材，激发学生和读者的兴趣，为进一步学习更高级的人工智能课程做好准备，同时使大量关于人工智能的文献易于查阅。

本书的一个打破常规之处是机器学习(*machine learning*)并未作为单独的论题讨论，而是将其贯穿本书始终。首先讨论神经网络(*neural net*)和受监督的学习(*supervised learning*)的基本思想；接着在"搜索(*search*)"章节中将讨论学习启发式搜索和动作策略的技术；然后，在有关"逻辑(*logic*)"章节之后将讨论规则学习(*rule learning*)，归纳逻辑编程(*inductive logic programming*)和基于解释的学习(*explanation-based learning*)；最后，在讨论了基于逻辑的计划(*logic-based planning*)之后，将讨论有关学习规划(*learning plan*)。

我以前的书中每章末尾均提供"参考书目和历史评价"(有的读者或许觉得仍有用)，但是在这本书中我并没有这样做。因为随着人工智能的发展，它所包含的内容已经愈加广泛，而且另一本更加详尽的教材已经作了这项工作(Russell & Norvig 1995)。但我在本书中引入了适当的评价和引用，并且在多数章节末尾的讨论小节中给出了另外一些讨论。那些有志于以人工智能为研究方向的学生可以查阅这些参考书，希望这些大量的引用不会给一般读者带来困扰。

每章末尾均附有习题，有些只是书中概念的简单应用，有些则稍具挑战性，难度不一。我希望教师能根据自己的需要扩充习题，包括上机练习和项目编程(为了与以思想而不是以程序

为中心的初衷保持一致，我并没有在书中涉及任何计算机上机练习和项目编程。在专门的人工智能编程技术教材中可找到相关内容。）

本书使用如下的排版约定：黑体大写字母如W和X用来表示矢量、矩阵和模式操作符。小写希腊字母表示谓词演算表达式和子表达式所涉及的元变量，有时也表示替换。大写希腊字母用来表示谓词演算公式的集合。小写字母p表示概率。

通过万维网（WWW），学生和研究者可以找到大量关于人工智能的资料，这里并没有列出它们的网址，因为现在列出的清单数月后会变得不完整和不确切。另外，通过Web搜索引擎，读者可迅速查找到应用实例、常见问题、参考书目、研究论文、程序、交互演示、研究所和会议的公告及研究者的主页等等。

在出版者的Web站点www.mkp.com/nils的网页上可找到本书的相关资料，如发现错误，请通过以下地址给出版商发电子邮件：aibugs@mkp.com。错误及更正可以在以下网址中找到：http://www.mkp.com/nils/clarified。

我的前一本人工智能教材《人工智能原理》（Principles of Artificial Intelligence）现在已经过时，但书中某些内容仍有价值，因此本书的编写直接采用了这些内容。同时，与其他人工智能教材（特别是[Russell & Norvig 1995，Rich & Knight 1991，Stefik 1995]）相互对照学习，亦十分有益。

目　录

第1章 绪 论

我认为，理解智能包括理解：知识如何获取、表达和存储；智能行为如何产生和学习；动机、情感和优先权如何发展和运用；传感器信号如何转换成各种符号；怎样利用各种符号执行逻辑运算、对过去进行推理及对未来进行规划；智能机制如何产生幻觉、信念、希望、畏惧、梦幻甚至善良和爱情等现象。我相信，对上述内容有一个根本的理解将会成为与拥有原子物理、相对论和分子遗传学等级相当的科学成就。

— James Albus “答复 Henry Hexmoor”，摘自URL：
http://tommy.jsc.nasa.gov/er/er6/mrl/papers/symposium/albus.txt
1995年2月13日

1.1 什么是人工智能

广义地讲，人工智能是关于人造物的智能行为，而智能行为包括知觉、推理、学习、交流和在复杂环境中的行为。人工智能的一个长期目标是发明出可以像人类一样或能更好地完成以上行为的机器；另一个目标是理解这种智能行为是否存在于机器、人类或其他动物中。因此，人工智能包含了科学和工程的双重目标。本书主要从工程角度讨论AI，集中说明构成智能机器设计基础的重要概念和思想。

长期以来，围绕着人工智能有很多争议。“机器是否能思考？”这一问题吸引了许多哲学家、科学家和工程师。在一篇著名的文章中，计算机科学的创始人之一，艾仑·图灵（Alan Turing），重述了这一问题，使其更经得起一种实验的测试，这种测试后来被称为图灵测试[Turing 1950]。下面将描述这一测试，但图灵同时指出对“机器是否能思考”这一问题的答案取决于人们如何定义“机器”和“思考”。他也许还可指出，这一问题还依赖于人们如何定义“能”。

让我们先来考虑“能”这个词。我们认为机器现在或将来能思考吗？我们认为原则上机器应该可以思考吗（即使我们不可能制造出这样的机器）？或者，我们真的要求实际的演示吗？由于人造物尚未具有广泛的思考能力，这些问题就变得非常重要。

一些人认为，能够思考的机器必定十分复杂且拥有复杂的经验（如与其所处的环境和其他能够思考的机器交流）。以致于我们永远也无法设计并制造出它们。产生全球气象的过程是一个很好的例子。尽管我们知道有关天气的一切重要现象，这些知识也无法让我们完整、详尽地复制天气现象。因为再没有比地球表层、大气层和海洋这些存在于宇宙之中、汲取太阳的光和热并受潮汐影响的更简单的系统能够完整详尽地复制天气现象了。同样，完全与人类相当的智能会十分复杂，或者至少会十分依赖于人类严密的生理机能，从而使其不能脱离处于特定环境的人的主体（*embodiment*）而单独存在（关于“主体”这一概念的重要性的讨论，可参见[Lakoff 1987， Winograd & Flores 1986， Harnad 1990， Mataric 1997]）。至于我们是否能造出与人类水平相当的能思考的机器仍无定论。但人工智能朝着这一目标的发展是坚定不移的，虽然这一进展比早期开创者们的预计要慢。我对我们最终的胜利持乐观态度。

接着，我们考虑“机器”这一词。许多人认为，机器是一种相当愚钝的东西，它总让人联

想起齿轮转动、蒸汽嘶嘶、钢铁锵锵的景象。这样的机器能思考吗？但是，如今计算机已大大延伸了“机器”这一概念。同时，我们对生物机制的理解也有了前所未有的进展。譬如：一种名为E6抗菌素的简单过滤性病毒（如图1-1所示），它头部含过滤性病毒DNA。它用尾部须根与一个细菌的细胞壁相连，先刺入细胞壁，再将其DNA注入此细菌中，然后这些DNA使此细菌产生成千上万这一过滤性病毒DNA的复制品。这些复制品自动集合而形成新的过滤性病毒后，离开这一细菌，再重复以上过程。这一完整的集合看起来、运作起来均像一台机器，我们还不如称之为由蛋白质构成的机器。

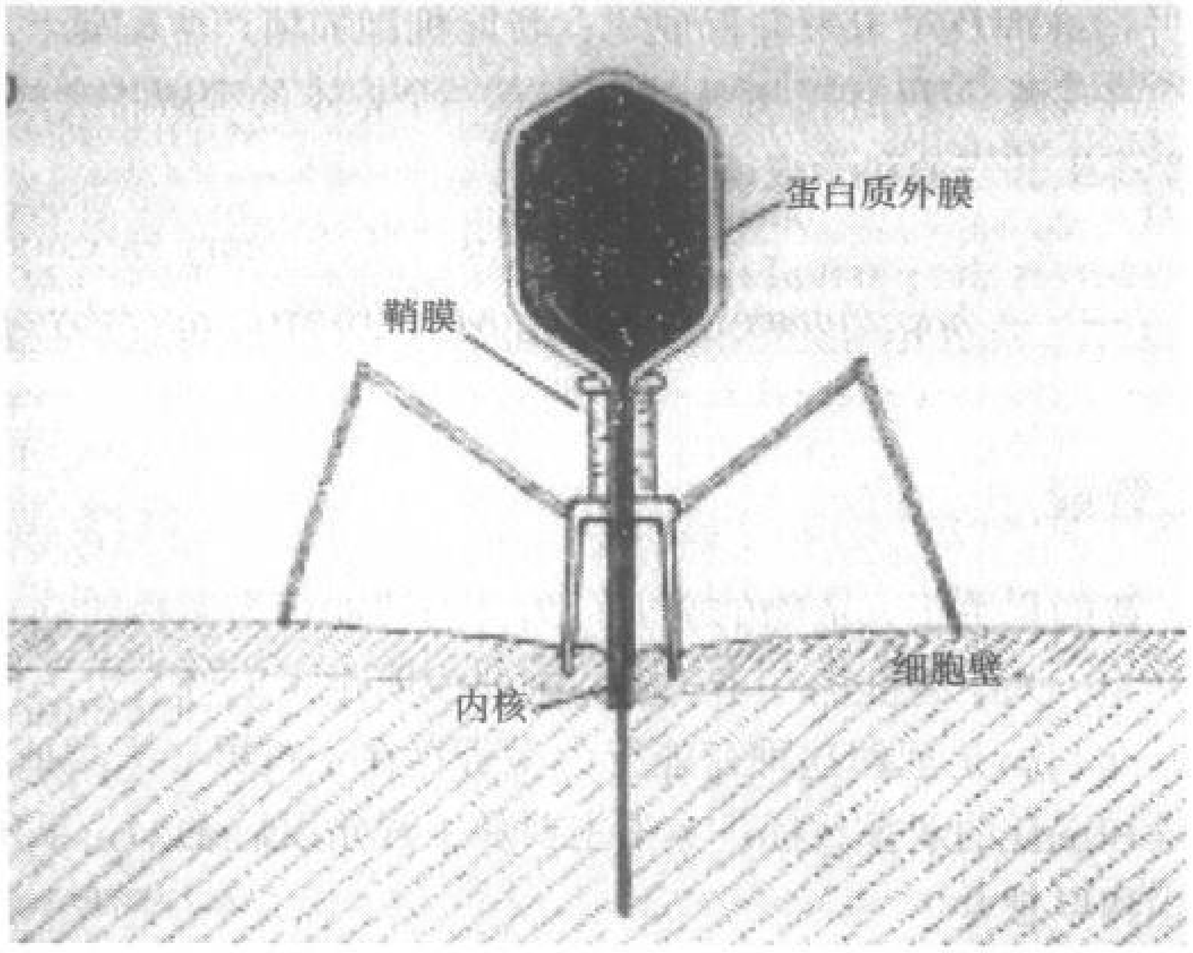


图1-1　E6抗菌素的示意图

其他生物过程和有机物又如何呢？最近，细菌Haemophilus influenzae Rd的基因排列已经完全被破解[Fleischmann, et al. 1995]。这一基因有1 830 137个基对（由字母A、G、C和T组成），大概占3.6×10^6比特，约半兆字节。尽管科学家们还不清楚其中1743个基因的功能，但他们已经开始像解释机器(当然是非常复杂的机器)的发展和功能那样解释这些机制的发展和功能。运用逻辑电路时序图这一计算机科学家熟知的技术，对理解可感染细菌的过滤性病毒的复杂生化基因的规则是十分有益的[McAdams & Shapiro 1995]。对其他有机物包括人类的基因组的排序工作正在进行中。一旦知道这些结果，我们会把细菌、寄生虫、果蝇、老鼠、海豚与人类一起视为机器吗？如果人类是机器，那么机器便能思考！因为我们拥有活生生的证据。只是我们还不知道人类这种机器是如何运作的。

即使我们就什么是“机器”达成一致观点，这种观点仍存在其他许多争议。尽管由蛋白质构成的机器能思考，由硅片构成的机器则未必能。一位知名哲学家John Searle认为，我们由什么构成直接影响着我们的智能[Searle 1980, Searle 1992]。他认为思考仅发生在那些十分特殊的机器上——有生命且由蛋白质构成的机器。

与Searle的观念（和前面所提到的“主体”概念）截然相反，Newell 和 Simon提出了物理符号系统假说[Newell & Simon 1976]。这一假说指出，物理符号系统具备必要且足够的方法来进行普通智能行为。Newell 和 Simon指出物理符号系统是类似数字计算机的机器，具备灵活处理符号数据的能力——加数、重排符号序列（如按字母顺序排列一组姓名）及符号替换等等。

这一假说的重要之一是它指出这种物理符号系统由什么构成并不重要！这一假说是完全中性的。一个智能实体只要能处理符号，它可以由蛋白质、机械传动、半导体或其他什么构成[⊖]。

还有一些人也认为，机器是由蛋白质还是由硅片构成无关紧要，但他们认为智能行为是他们所谓的“亚符号处理”，即“信号处理”，而不是“符号处理”的结果。如识别熟悉面孔对人类来说易如反掌，而我们却不知道机器该如何运作。他们认为这一过程最好的解释便是人类把图象或图象各部分作为多维信号而不是符号来处理。

能列出很多其他关于什么样的机器才具有人类的思维能力的看法，我们经常听到的有：

- 人脑对信息进行并行处理，而传统的计算机则是串行处理。我们需建造各种新型的并行计算机来加快人工智能的发展。
- 传统的计算机以非真即假（双态）逻辑为基础，而真正的智能系统应运用某种模糊逻辑。
- 动物神经元远比开关——计算机的基本模块——要复杂，我们需要在智能机器中运用更现实的人造神经元。

尽管许多人工智能研究者接受物理符号系统这一假说，但在人工智能领域关于究竟需要哪种机器达成共识还为时过早。

最后，我们来看看“思考”这一最难的词。图灵没有企图对这个词下定义，只是提出了“图灵测试”。通过这一测试即可判断某一特定机器是不是智能机器。这一测试最初被描绘成一种游戏。从图灵的文章中摘录如下[Turing 1950]：

> 游戏由一男（A）、一女（B）和一名询问者（C）（性别不限）进行。询问者单独在一间房间里与其他两人分别通过电传打字机联系。在游戏中，询问者的目的是分辨两人的性别。开始，他只知道两人的称呼X、Y，最终，他需要在“X是A，Y是B”或者“X是B，Y是A”中选择答案。询问者允许问A和B以下问题：
>
> C：X能告诉我你的头发的长度吗？
>
> 如果X是A，那么他必须回答。游戏中，A必须尽力使C判断错误。
>
> · · ·
>
> 而B的任务则是帮助询问者。
>
> · · ·
>
> 现在我们提出这样一个问题：一个机器代替游戏中的A会如何？询问者会依然像当游戏由一男一女进行时一样经常判断错误吗？这些问题代替了最初的问题：机器能思考吗？

图灵测试常被简化为让一个机器试图使询问者相信它是一个人。许多更简单的测试层出不穷，然而由于就连一些陈旧的机器也可以愚弄询问者一段时间，这些简单的测试已经不再被视为测试机器智能的良方了。譬如：Joseph Weizenbaum的ELIZA程序运用一些相当简单、但对一个宽容的使用者却是虽显空洞却十分现实的对话技巧。Mauldin的JULIA程序是更新和更复杂的对话程序[Mauldin 1994][⊜]。

除了运用图灵测试，我们有必要在标榜一台机器是智能机器之前，了解这样的机器应具备

⊖ 当然，如果我们考虑到速度、永久性、可靠性、并行处理的适合性和温度敏感度等实际因素，模块材料必定有好有坏。

⊜ 1991年，Hugh Loebner开始举行一个有奖竞赛，他向第一个能通过无限制图灵测试的计算机程序的开发者提供10万美元的奖金。另外，每年这一竞赛都为能通过有限图灵测试的最佳程序的开发者提供数额略少的奖金。

怎样的能力。许多计算机程序已经完成了大量不可思议的事——设计高效省油的最佳航空路线、模拟全球气象状况、统筹安排工厂的机器使用等等。这些是智能程序吗？它们能体现人工智能的主旨吗？本书一开始我便描绘那些难以被人们称为智能机器的机器，随着其复杂性的增强，它们会变得越来越智能吗？毫无疑问，别人会有不同的观点，但至少我这样认为。

1.2　人工智能的研究方法

尽管人工智能已经创造了一些实用系统，但人们不得不承认这些远未达到人类的智能水平。正因为如此，就选择人工智能研究的最佳方法——既为人工智能的最终研究目标打好基础，又能创造出短期效益——存在大量的讨论和争辩。这样，在过去的四十年里涌现出大量方法，每一种方法均有其拥护者，有些甚至有趣得令人爱不释手。也许所有这些方法应该综合起来运用。总之，所有这些拥护者都认为自己的研究方法具有突破性进展，值得特别关注。其中的一些方法可分为两大类。

第一类包括符号处理的方法。它们基于Newell和Simon的物理符号系统的假说。尽管不是所有人都赞同这一假说，但几乎大多数被称为“经典的人工智能”（即哲学家John Haugeland所谓的“出色的老式人工智能”或GOFAI）均在其指导之下。这类方法中，突出的方法是将逻辑操作应用于说明性知识库。最早由John McCarthy的“采纳意见者”备忘录提出[McCarthy 1958]，这种风格的人工智能运用说明语句来表达问题域的“知识”。这些语句基于或实质上等同于一阶逻辑中的语句。采用逻辑推理可推导这种知识的结果。这种方法有许多变形，包括那些强调对逻辑语言中定义域的形式公理化的角色的变形。当遇到“真正的问题”，这一方法需要掌握问题域的足够知识，通常就称作基于知识的方法。许多系统的构建都运用了这些方法，在本书后面将会提到一些。

在大多数符号处理方法中，对需求行为的分析和为完成这一行为所做的机器合成要经过几个阶段。最高阶段是知识阶段，机器所需知识在这里说明。接下来是符号阶段，知识在这里以符号组织表示（例如列表可用列表处理语言LISP来描述），同时在这里说明这些组织的操作。接着，在更低级的阶段里实施符号处理。多数符号处理采用自上而下的设计方法，从知识阶段向下到符号和实施阶段。

第二类包括所谓的“子符号”方法。它们通常采用自下而上的方式，从最低阶段向上进行。在最低层阶段，符号的概念就不如信号这一概念确切了。在子符号方法中突出的方法是“*Animat approach*”。偏爱这种方式[Wilson 1991，Brooks 1990]的人们指出，人的智能经过了在地球上十亿年或更长时间的进化过程。他们认为，为了制造出真正的智能机器，我们必须沿着这些进化的步骤走。因此，我们必须集中研究复制信号处理的能力和简单动物如昆虫的支配系统，沿着进化的阶梯向上进行。这一方案不仅能在短期内创造实用的人造物，又能为更高级智能的建立打好坚实的基础。

第二类方法也强调符号基础。[Brooks 1990]将物理符号系统和他的物理基础假说相对照。在物理基础假说中，一个agent不采用集中式的模式而运用其不同的行为模块与环境相互作用来进行复杂的行为（然而，他也承认，要达到人类智能水平的人工智能也许需要将两种途径相结合）。

机器与环境的相互作用产生了所谓的“自然行为（*emergent behavior*）”。一名研究人员这样说[Maes 1990b，p.1]：

> 一个agent的功能可视作该系统与动态环境密切相互作用的自然属性。agent本身对其行为的说明并不能解释它运行时所表现的功能；相反，其功能很大程度上取决于环境的特性。不仅要动态地考虑环境，而且环境的具体特征也要运用于整个系统之中。

由子符号派制造的著名样品机器包括所谓的“神经网络（*Neural network*）”。受到生物学方法的启发，这些系统主要因其学习的能力而十分有趣。根据模拟生物进化方面的进程，一些有趣的机器应运而生，包括：Sexual crossover、Mutation和Fitness-proportional reproduction。其他自下而上、含animat风格的方法是基于控制理论和动态系统的分析（参见[Beer 1995，Port & van Gelder 1995]）。

介于自上而下和自下而上之间的方法是一种动机“环境自动机（*situated automata*）”[Kaelbling & Rosenschein 1990，Rosenschein & Kaelbling 1995]的方法。Kaelbling 和Rosenschein 建议编写一种程序设计语言来说明agent在高水平上所要求的行为，并编写一编译程序，以从这种语言编写的程序中产生引发行为的线路。

1.3 人工智能简史

当20世纪40~50年代数字计算机研制成功时，几位研究者就编写了能够完成原始推理工作的程序。其中突出的是第一个可以下国际象棋[Shannon 1950， Newell，Shaw & Simon 1958]、担当实验员[Samuel 1959，Samuel 1967]和证明平面几何定理[Gelernter 1959]的计算机程序。1956年，John McCarthy和Claude Shannon合作编著了一本名为《Automata Studies》（自动机研究）的书[Shannon & McCarthy 1956]。由于对书中主要针对automata的数学理论感到遗憾，所以McCarthy决定把1956年的Dartmouth会议用人工智能来命名。在该次会议上发表了许多重要论文，包括由Allen Newell、Cliff Shaw和Herbert Simon 编写的名为《Logic Theorist》（逻辑理论家）[Newell，Shaw & Simon 1957]的程序，它可以证明命题逻辑中的定理。尽管人们试着用许多其他名称来为该领域命名，包括复杂信息处理、机器智能、启发式编程和认知技术，但人工智能这一名称最终保留下来。毫无疑问，这主要归因于一系列的教科书、大学课程、会议和期刊均用这一命名。

很久以前，亚里士多德（公元前384~322年）在着手解释和编纂他称之为三段论的演绎推理时就迈出了向人工智能发展的早期步伐。一些使智能自动化的努力对于今天来说显得太不实际。一位加泰罗尼亚的诗人兼神秘主义者，Ramon Llull（大约1235~1316年），构建了一套称为Ars Mgna的转轮，据说是一部可以回答任何问题的机器。同时许多科学家和数学家开始探讨推理自动化。Martin Gardner[Gardner 1982，p. 3]把“有一天所有的知识，包括精神和无形的真理，能够通过通用的代数演算放入一个单一的演绎系统”的梦想归功于莱布尼兹（1646~ 1716年）。莱布尼兹称这个系统为微积分原理机，或推理机。当然，这个梦想运用当时的技术设备是无法实现的。直到布尔[Boole 1854]建立并发展了命题逻辑，这方面才有了实质性的进展。布尔的意图是要“把有关人类意识的本质和构成的某些可能的暗示收集起来”。到了19世纪末期，Grottlieb Frege提出了用于机械推理的符号表示系统，从而发明了我们现在熟知的谓词演算[Frege 1879]，他称之为Begriffsschrift，可以译为“概念书写（*concetp - writing*）”。

1958年，John McCarthy 建议在他称之为“意见采纳者”的系统中采用谓词演算这种语言来表示和运用知识。这一系统被告知它所需要知道的而不是事先程序设计好的知识。Covdell

Green在他所谓的QA3系统中[Green 1969a]适度地、颇有影响地实现了这些思想。尽管在研究者中还存在大量争议，谓词演算和一些它的变形构成了人工智能知识表示的基础。

20世纪的逻辑学家，包括Kurt Cödei、Stephen Kleene、Emil Post、Alonzo Church和Alan Turing，对哪些能和哪些不能由逻辑和计算机系统完成的任务做了形式化分类。最近，计算机科学家，包括Stephen Cook和Richard Karp证明有些计算在原则上可能需要根本不切实际的时间和存储空间。

许多从逻辑学和计算机科学中所得到的结果是："真理不可能被演绎"、"计算不可能被执行"。也许这些负面的发现令许多哲学家和其他人振奋，他们将之理解为再一次否定了人类的智能可以机械化[Lucas 1961, Penrose 1989, Penrose 1994]，他们猜想人类不存在机械所固有的计算局限。然而多数逻辑学家和计算机科学家却认为这些负面结果并不暗示机器具有人类所不具有的任何局限。

在现代，第一篇讨论把人类智能机械化的可能性的文章是由Alan Turing所著的（前面已经引用）[Turing 1950]。同一时期，Warren McCulloch和Walter Pitts总结出简单计算元素和生物神经元之间关系的理论[McCulloch & Pitts 1943]。他们证明了运用逻辑网络系统计算可计算功能的可能性（参见 [Minsky 1967]有关McCulloch Pitts神经元计算方面有价值的论述）。另外，由Frank Rosenblatt[Rosenblatt 1962]所著的书中探讨了称作perceptrons的网络由类似于神经元的部件组成运用于学习和模式识别的可行性。一些其他学派的工作，如控制论[Wiener 1948]、认知心理学、计算语言学[Chomsky 1965]和自适应控制理论[Widrow & Hoff 1960]，均对人工智能的发展作出了贡献。

许多人工智能的早期工作（从20世纪60年代至70年代初）探讨了问题表示、搜索技术和通用启发等一系列问题——并把它们运用于计算机程序中来解迷、博奕和检索信息，其中有影响的程序是由Allen Newell、Cliff Shaw和Herbert Simon[Newell，Shaw & Simon 1959，Newell & Simon 1963]共同编写的通用问题求解程序（General Problem Solver（GPS））。由这些早期系统解决的实例问题包括符号集成[Slagle 1963]、代数词汇问题[Bobrow 1968]、类比难题[Evans 1968]及机器人的控制[Nilsson 1984b]。在这些系统中，许多都是《Computers and Thought》这卷书中的主题[Feigenbaum & Feldman 1963]。

为了应用于更重要的现实问题而对这些程序和技术进行升级的尝试表明这些系统只能解决"玩具问题"。更有效的系统要求对应用领域具有更多内在的知识。20世纪70年代末80年代初发展了一些更高级的程序，包括在完成一定任务时模拟专业人员的知识，如分析、设计和诊断等。一些表达具体问题的知识得到了探讨和发展。第一个能演示具体领域知识的重要程序DENDRAL是一个根据所提供的化学分子式和质谱分析图来预测有机物分子结构的系统[Feigenbaum，Buchanan & Lederberg 1971，Lindsay, et al. 1980]。接着，其他"专家系统"，包括医疗诊断[Shortliffe 1976，Miller，Pople & Myers 1982]、计算机系统的配置[McDermott 1982]和矿藏评估（evaluated potential ore deposits）[Campbell，et al. 1982，Duda，Gaschnig & Hart 1979]。[McCorduck 1979]撰写了这一阶段的人工智能简史。

通过升级游戏问题，博奕这一领域有了实质性的进展。1997年5月11日，一个名为"深蓝"的IBM程序在六局比赛中以3.5比2.5的总比分战胜了世界象棋冠军Garry Kasparov(盖利·卡斯帕洛夫)。这次成功是运用复杂的搜索算法、高速计算机和国际象棋专用硬件才得以实现的。

人类的智能包括洞察和分析可视场景、理解并运用语言等许多方面的能力。关于这些能力的专题均得到了高度重视。Larry Roberts开发了早期场景分析程序之一[Roberts 1963]。这一工作之后对机器视觉作了大量研究（[Nalwa 1993]是一本很好的通用教材），对动物视觉系统[Letvinn, et al. 1959, Hubel 1988, Marr 1982]也作了研究。一个早期自然语言理解系统也由Terry Winograd开发成功[Winograd 1972]。20世纪70年代，在一个多方项目中，连续语言理解系统原型被开发出来；由William Woods开发的LUNAR系统能回答用口语提出的关于由美国航空航天局（NASA）从月球收集的岩石样品的问题。尽管现在存在一些自然语言理解系统，但它们的能力仅局限于特定的话题和词汇。打破这些局限有待于开发出更大量的常识表示。CYC项目[Guha & Lenat 1990, Lenat & Guha 1990, Lenat 1995]的一个目标就是尽可能多地收集、表达这些所需的知识。

20世纪50年代末在Frank Rosenblatt所领导的开创性工作之后，对神经网络的研究虽然一度萎靡，但是到20世纪80年代又恢复了活力。具有强度可调互连系统的非线性元素网络如今已被视为一类重要的非线性建模工具。现在已存在神经网络方面的一些重要应用。动态方法与神经网络相结合，促使人工智能的研究集中到把符号处理过程与处于物理环境中的机器人的传感器与受动器联系起来的问题上来。

立足当前，展望未来，我认为人们将重视集成的、自治的系统——机器人和Softbots。Softbots[Etzioni & Weld 1994]是在互联网中查找他们认为用户会感兴趣的信息的软件agent。今后，不断提高和完善机器人和软件agent的能力将促进并引导人工智能研究。

1.4 本书规划

许多人工智能研究者已提出了一些有关智能机械化的观点和技术，我会在介绍一系列逐步弹大逐步复杂的agent时陈述这些内容。我们本可以考虑各类agent及其环境，如：在太空失重的情况下、在海底深水域中、在办公楼或工厂中及在互联网的“符号数据世界”中的机器人。然而，在这样的真实世界中，真正实用的agent将会十分复杂，这样会难以清晰地展示赋予agent智能的人工智能概念。因此，我将在“网格空间世界”这一假想空间中采用一系列“玩具”agent。虽然简单世界很容易描述，但各方面的发展使之变得太复杂而迫使其中的agent需要有智能才行。

网格空间世界是一个三维空间，它以二维的地面为界限，而地面是由一系列单元格组成。单元格集合可以容纳具有各种特性的物体。单元格集合之间可能会存在像墙一样的边界，agent不能离开地面，但可以在单元格之间移动。物体必须在地面上或必须由在地面上的其他物体支撑。有时我会采用仅包含地面的二维子空间。一个典型的网格空间世界如1-2图所示。图中有两个机器人，一个是原始的二维机器人，它用感知邻近单元格是否为空的传感器来判定是否向其移动；另一个略微复杂，它有一个可操纵物体的手臂。

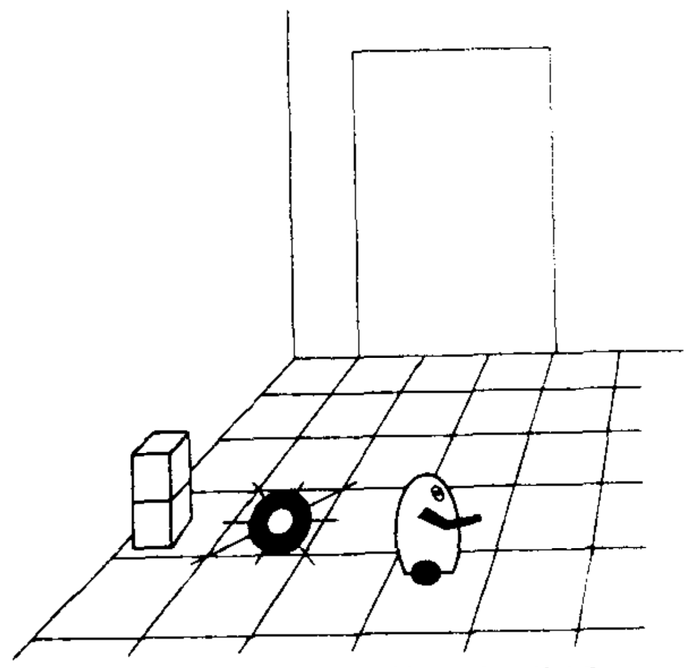


图1-2 网格空间世界

熟悉人工智能历史的读者会发现网格空间世界能够被定制为其他许多用于人工智能研究

的“世界”，包括积木世界、瓷砖世界［Pollack & Ringuette 1990］、wumpus世界［Russell & Norvig 1995，pp. 153以后］和蚂蚁世界［Koza 1992，pp. 54以后］。这些世界仅包含有限的位置、agent、物体和时间点。从这一意义上讲，它们均是离散的。我所描述的多数人工智能技术仅适用于离散世界，而且需要用子符号处理，使它们与连续世界相连。

本书的开头介绍响应agent，它们运用不同方式感知世界并活动于其中。更复杂的agent具有记忆世界特性并存储世界内部模型的能力。任何情况下，响应agent的行为都是所感知和记忆的世界的过去或现在状态的函数。它们能进行相当复杂的知觉和驱动处理。虽然在第6章将详细介绍视觉感知，但因篇幅所限，有关机器人驱动的底层控制的内容无法囊括其中。

多数人工智能系统对它们所处的世界和任务采用某种模型和表示。从广义上讲，模型是与世界紧密相关的任何符号结构和计算的集合，在此基础上计可算产生agent所需的有关世界的信息。这一信息可能是当时agent所处世界或今后可能所处世界的状态。在由浅入深地介绍人工智能agent的过程中，我把人工智能系统分为两种模型。一种是“图标”模型，包括模仿agent环境各方面以及agent的行为对环境产生的影响的数据结构和计算。在一张8×8单元格数组中，表示棋局状况便是图标表示的一例。如果这一表达包括所有棋子位置的信息，则这一棋局图标模型就是完整的。

另一种是“基于特征”的模型。这种模型对环境进行描述。譬如说在棋局中有两个特征，一个特征可能是车或王是否安全，另一个可能是王被将了几次军。描述特征集往往不完整——这是基于特征的表示方式的优点，它们能容忍agent对复杂世界的知识了解的残缺。

接下来介绍的一系列agent能够预计其行为的效果，并采取那些能够达到预期目的的行动。这样的agent可以说具备规划能力。一些研究者认为，这种能力是判断机器是否具备智能的标准，而人工智能也应该由此开始发展。在不能完全感知和模型化的世界中采取行动的agent也需要时刻了解它们的行为是否达到预期效果。

多数网格空间世界具有与真实世界的特性类似的隐含约束条件。如一个物体已有一个具体位置，在同一时刻它就不可能有另一个位置。能够考虑这些约束条件的agent通常更有效力。然后，我将介绍一系列具有推理能力的agent，它们能演绎出隐含于约束条件中的、所处世界的特性。

最后，我将介绍已有其他agent占据的世界中的agent。它们出色的表现有时依赖于对其他agent行为的预期和影响。agent之间的交流十分重要。

在不断加大agent复杂程度的过程中，我总是会讨论agent学习它们所处环境的方法。除了计划能力，学习能力也被视为一个智能系统品质的证明，这一点，我与[Russell & Wefald 1991，p. 18]的观点一致。Russell 和Wefald 写到：

> 学习是自治性的一个重要方面。一个系统可称之为自治的，即它的行为是由其自身的当前输入和过去的经验而不是设计者的输入和经验来决定。agent往往为一类环境专门设计，这类环境中的每一种情况都已经存储在agent中，并且与设计者所了解的真正环境相一致。这样一个在固有的假设基础上操作的系统只有当这些假设完全真实时才可能成功运行，因而缺乏灵活性。如果给予充分的应变时间，一个真正自治的系统应能在任何环境中成功地运行。原则上，这一系统的内部知识结构应该可以根据其自身对世界的经验而进行构造。然而不能把自治系统与tabula rasa系统等同起来。一个合理折衷的观点是在一开始根据设计者对世界的知识来设计大部分系统的行为，但所有这些假设必须尽可能明确且易

于为agent 所修改。这种意义的自治和我们最初关于智能的概念也完全吻合。

除了用来统一集中论述人工智能技术而使用的网格空间，我还会不断地介绍一些解决现实问题的重要应用。

1.5 补充读物和讨论

关于反对使用图灵测试的争论（不幸的是这似乎也是放弃人工智能达到人类智能水平的宏伟目标的争论），请参阅[Hayes & Ford 1995]，但图灵测试在小说《Galetea 2.2》[Powers 1995]中却十分有趣。

人工智能的自上而下和自下而上的两种研究方法均受到对人类和动物行为研究的启发。进行自下而上研究的人倾向于集中对可以通过组织类神经元的计算元素或逻辑门而实现的行为（通常是简单的行为）进行研究。像动物行为学家一样，他们借此创造了动物行为的各种计算模型。本章提到的许多所谓基于行为的动态人工智能的研究方式是从动物模型中得来的。就动物和机器人之间的比较的讨论，请参阅[Anderson & Donath 1990，Beer，Chiel & Sterling 1990]。

像心理学家那样，自下而上的研究者和神经科学家也已经发展了试图解释某些人类知觉和行为的神经网络模型。这样的神经网络系统有的可以学会朗读手写体语句[Sejnowski & Rosenberg 1987]，有的可以识别尺寸、方位和姿态不同的字母数字字符[Minnix，McVey & Iñigo 1991]，有的具有在书信识别方面上下文敏感的洞察力[McClelland & Rumelhart 1981，McClelland & Rumelhart 1982]。

当自上而下的研究者从动物和人类的行为中获取灵感时，他们倾向于将研究的焦点集中在那些可用符号来处理最佳模型化的领域，包括认知心理学家研究的问题求解、语言和记忆任务。开发人类问题求解的计算机模型的两位先驱是Herbert Simon 和Allen Newell（参阅[Newell & Simon 1972，Newell 1991]）。至于后一本书的评论和Newell所做的回答，请参阅[Artificial Intelligence，vol 59，1993]。而关于认知科学和计算机科学之间的关系，请参阅[Johnson-Laird 1988]。

当然，你也可以认为无论动物和人类如何完成智能行为与设计智能人造物这一工程问题毫无关系，就如同飞机并不像鸟或昆虫那样飞行。设计优良的智能机器，尽管也许能够超越人类，但也许与自然发生的智能行为根本不同。通常引用“蛮力（*Brute-force*）”搜索(因此假定为非人类所为)方式在许多领域的成功应用，包括：玩游戏、列时间表和规划，[Ginsberg 1996]推测机器更高级的思考方式也许会与人类和动物的大脑截然不同，以至拘泥于这种体系结构的人工智能将无法复制人和动物的行为[Dreyfus 1979，Dreyfus 1992，Dreyfus & Dreyfus 1986]。

其他学科对人工智能的影响并不总是定位于自下而上或自上而下的方式。譬如说，[McFarland & Bösser 1993]讨论了许多发生在动物身上的计算实例，然而他们的观点却与经济学和公用事业理论紧密相连。他们认为，动物是经济agent。本书第3章将着重讨论公用事业理论在动物模型中的作用。一个名为Michael Wellman的人工智能研究者发明了一种称为“面向市场的程序设计”方法[Wellman 1996]。[Shoham 1996]，也陈述了相关观点。

如何定义智能行为正是区别自下而上和自上而下两种方式的根本所在。不管一种行为是如何“计算”出来，只要它是正确的，就能称之为智能行为吗？实际上存在两种相反的意见。[McFarland & Bösser 1993，p. 6]认为“智能行为是能够得出正确答案的行为，而不管这一答案

是如何得出的”；相反，[Russell & Wefald 1991, p. 1]认为，“……由于任一物理系统在推理能力方面不可避免的局限使其不可能在任何时候都能做出正确的举动”，“智能系统的设计者需要忽略其举动的正确性而去考虑设计正确的系统……”。这一观点引出了基于“有限合理性”的方法。他们主张智能系统必定是在两种行为中决策：一种是在世界中的行为；另一种是旨在完善对其所在世界中最佳行为的估计的计算行为（请参阅[Russell 1997]）。

可计算性的确限制了智能系统所能完成的任务。根据复杂性分析[Garey & Johnson 1979]的原则，人们更加注重对“易处理的”（即可在多项式空间与时间代价下实现的）任务的计算。然而复杂性分析通常涉及最坏情况（而不是平均情况）的结果，许多有趣的人工智能计算都具有最坏情况的指数计算复杂度。我想这样回答那些对许多人工智能算法的不易处理性忧心忡忡的人比较合适，即我们寻找具有平均情况表现良好的算法，并且在许多情况下愿意寻找粗略的、非最佳的解决方法。

对日益丰富的agent的发展的详细说明与动物智能进化的某些步骤息息相关，这对我来说并不新鲜。[Dennett 1995, pp 373以后]提出了一个相似的agent的进化顺序，他将之命名为Darwinian，Skinnerian，Popperian和Gregorian。

尽管我们仍远远无法创造出具有一般人类智能的系统，但想一想这样做的结果却十分重要。当然，廉价的机器人、Softbot、自然语言系统和专家系统具有相当的经济价值。运用这些系统会导致大量失业吗？或者会像早期的工业技术那样创造出比淘汰的更多的就业机会吗？如果说多数这些工作能够由它们这些智能系统完成又会如何呢？（至于我对这些问题的早期想法，请参阅[Nilsson 1984a]）。Joseph Weizenbaum[Weizenbaum 1976]曾担心另外一个问题，即将这些人工智能系统运用到他认为不适当的任务（如辩护、教学和审判）中去的危险。因为自动系统往往造成一种假像，即它们能完成对它们来说实际上根本无法完成的任务，所以会出现对人工智能系统一味草率依赖的危险。然而对这些稍有弥补的是，自动系统比人类少出错。至于人工智能系统对各方面产生的影响的文章，请参阅[Trappl 1986]。

也许，人工智能对我们理解人类自身所产生的影响最深刻。哥白尼和其他天文学家把我们从宇宙的中心带到了不计其数的银河系中的一个小行星上；达尔文和其后的进化论者又使我们从天地万物的中心变成了现在的基于DNA的不计其数的生命形态之一。我们过去都曾难以接受所有这些观念的变化，那么，如果我们真的成功造出像我们一样聪明的机器，我们又将面临什么呢?

本节末尾，我简要列出了有关人工智能的资料出处，它们会随本书后面章节中介绍的子论题的增多而增多。一些重要的期刊包括：《Artificial Intelligence》、网上的《Journal of Artificial Intelligence Research》、《Computational Intelligence》和《Journal of Experimental and Theoretical Artificial Intelligence》。重要的会议包括：年度国家人工智能会议（由AAAI主办），两年一次的国际人工智能联合会议（IJCAI）。一些国家和地区举办的会议还发表会议论文集，如欧洲人工智能会议（ECAI）。AAAI还举办年度春季专题讨论会和秋季研讨会，借此宣布及讨论最新的研究成果。计算机械协会（ACM）还拥有一个特殊的人工智能兴趣组（SIGART），他们发布通讯。AAAI还出版《人工智能》杂志。

PC AI杂志发表有关人工智能技术应用的文章——主要针对决策支持和专家系统。杂志《Engineering Applications of Artificial Intelligence》(EAAI)包括侧重于实时系统的文章。在《IEEE Expert》中有关“智能系统及其应用”的系列文章介绍了世界各地致力于应用研究和工

业技术转化的人工智能实验室。

对不同人工智能论题的总结可查阅《The Encyclopedia of Artificial Intelligence》[Shapiro 1992]、《The Handbook of Artificial Intelligence》中的几卷[Barr & Feigenbaum 1981, Barr & Feigenbaum 1982, Cohen & Feigenbaum 1982, Barr, Cohen & Feigenbaum 1989]以及《Exploring Artificial Intelligence》[Shrobe 1988]。一些人工智能子领域中的重要论文再版在名为“Readings in *X*”的出版物中（*X*代表各种具体领域）。

习题

1.1 给出你自己关于机器的定义。你认为人类是机器吗？不论你的看法如何，运用你的定义和有关人类各种能力的证据证明你的观点。

1.2 你能列举出用蛋白质而不是硅片制造思维机器的实际好处吗？

1.3 假设你是图灵测试中的询问者，想出问*X*或*Y*的五个用于判断它们哪一个是人和哪一个不是人的问题。

1.4 批判地对用图灵测试来判定非人机器是否能思考进行评价，至少提出一种不同观点。

1.5 一些人工智能研究者主张人工智能的目标是建造能“帮助”人们进行智能任务的机器，而不是去“完成”那些任务。不严格地讲，去“帮助”有时被称为“弱人工智能”（*weak AI*），而去“完成”有时被称为“强人工智能”（*strong AI*）。你怎样认为？为什么？

第一部分　响应机器

当你思考这一领域的问题时，会不自觉地不知所措。思考中的你正处于实际生活中的某一周里，那正是阳光明媚的时候。一旦深入这一游戏，你会恍然大悟：你只需对你所看到的作出响应。

——Albert Lewis,cornerback of the Oakland Raiders,"……Sam Farmer 引自《San Jose Mercury News》，1D页，1996年8月30日。

第2章　刺激响应agent

2.1　感知和动作

在本章中，我将介绍不具有内部状态而仅对其所处环境的即刻刺激有所反应的机器。我们称之为刺激响应（*stimulus-response*, S-R）agent。各种机器人应运而生，它们能通过电机对传感器即刻输入的十分简单的功能反应而展示相当有趣的行为。这种机器人中最早的例子是Grey Walter的*Machina speculatrix*——一个有轮子、配有电机、光电管和两个真空管的装置[Walter 1953]，能够朝光度适中的地方移动而避免强光。Braitenberg也描述过类似的机器[Braitenberg 1984]。

用一个图例来开始我们的讨论。先来看这样一个在二维网格空间世界里的机器人，如图2-1所示。这一机器人的世界有完整的边界线，可能还包括如图所示的其他庞大的固定物体，这一世界里没有"稠密空间（*tight space*）"（即物体与边界线之间的距离只有一个单元格）[⊖]，我们设计机器人时可利用这一点。

我们要求这个机器人完成以下动作：走到与一边界或物体毗邻的单元格中，然后沿着它的边界一直走下去。要能够完成这一沿边界运动的行为，机器人必须能够感知一个单元格是否空缺而可以向其移动，并且必须能够做一些基本的动作。

这个机器人能够感知出它周围八个单元格是否空缺。这些传感器输入用二进制变量s_1、s_2、s_3、s_4、s_5、s_6、s_7和s_8表示。当相应的单元格能被机器人占据时（如图2-1所示），它们的变量值为0，反之为1。如果机器人处于有X标记的地方，传感器的输入值（从s_1开始顺时针计算）为（0，0，0，0，0，0，1，0）。

该机器人能够向与它同行或同列的毗邻的（空缺）单元格移动，共有如下四种动作：

north：机器人在网格中向上移动一个单元
east：机器人在网格中向右移动一个单元
south：机器人在网格中向下移动一个单元
west：机器人在网格中向左移动一个单元

⊖ 有关"不存在稠密空间"的具体论述请参阅 www. mkp.com/nils/clarified。

所有这些动作均有其效果，除非该机器人企图向非空缺的单元格移动；如果这样，则此动作无效。

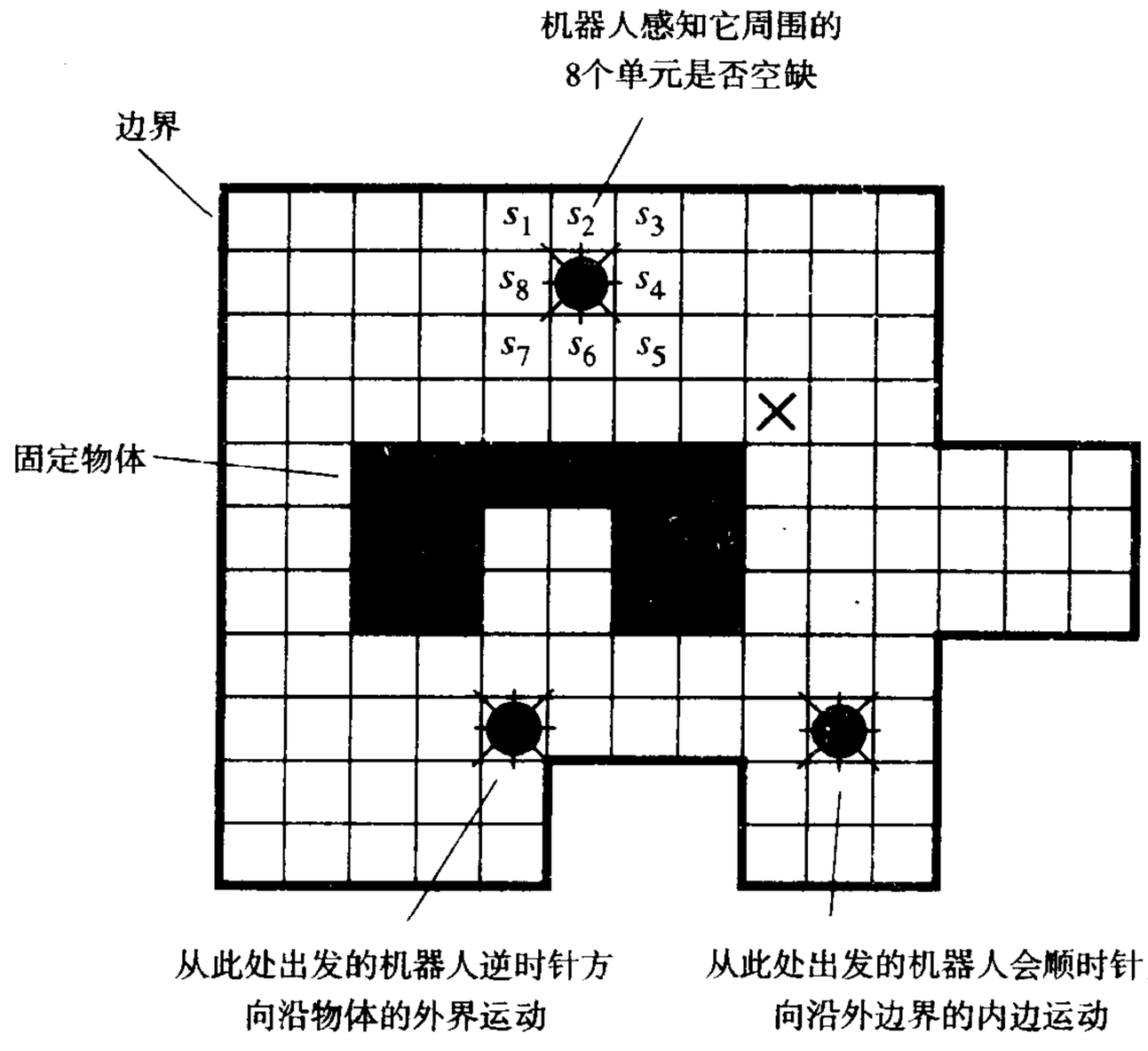


图2-1　一个在二维网格空间中的机器人

给定了机器人适应的某种世界的特性（如图2-1所示）、机器人完成的任务（沿边界移动）和机器人传感器和电机的功能，设计者的工作就是说明为此任务选择适当动作的传感器输入（示例中表示为s_1，…，s_8）的功能。通常我们把从传感器信号中计算动作的过程分为两个分开的阶段，如图2-2所示。知觉处理阶段产生一个特征向量**X**（x_1,… x_i,…,x_n）。动作计算阶段选择一个以特征向量为基础的动作：各特征值既可以是真正的数字（*numeric feature*，数字特征），也可以是范畴（*categorical feature*，范畴特征）（范畴特征的值是名字或特性，譬如：特征值“颜色”可能是“红”、“蓝”或“绿”）。二进制特征这一特殊例子既可视作数字（0，1），也可视作范畴（真，假）。设计者选择特征来将其与机器人的环境特性相联系，而此环境特性又与由此特征描述的状态中机器应做的动作密切相关。

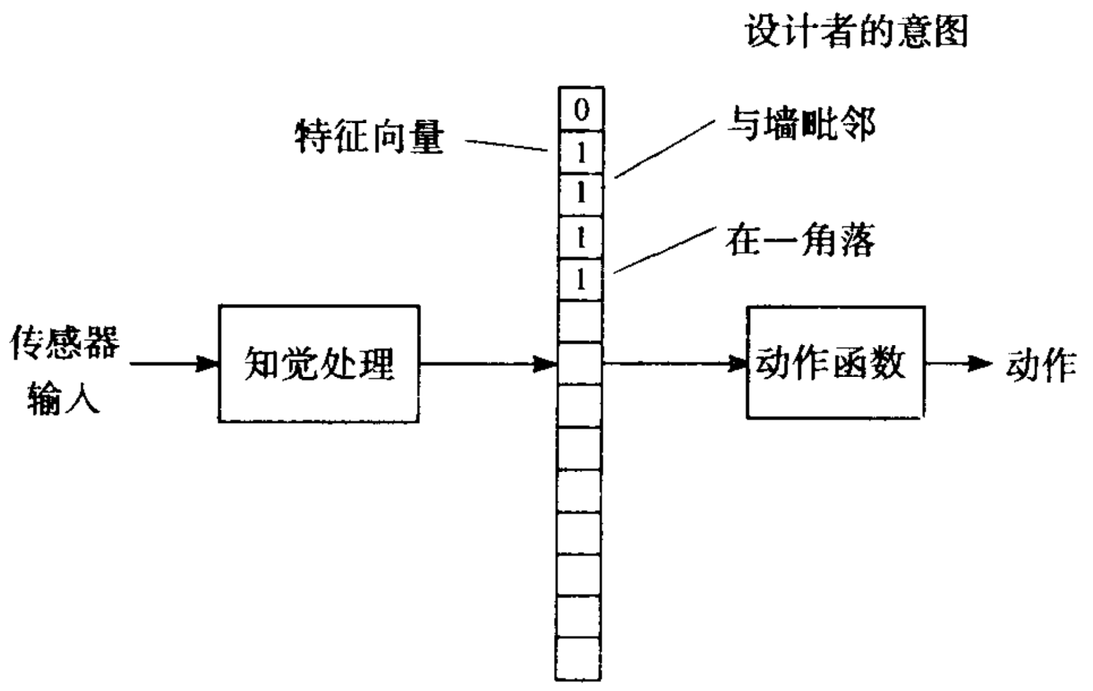


图2-2　知觉和动作部分

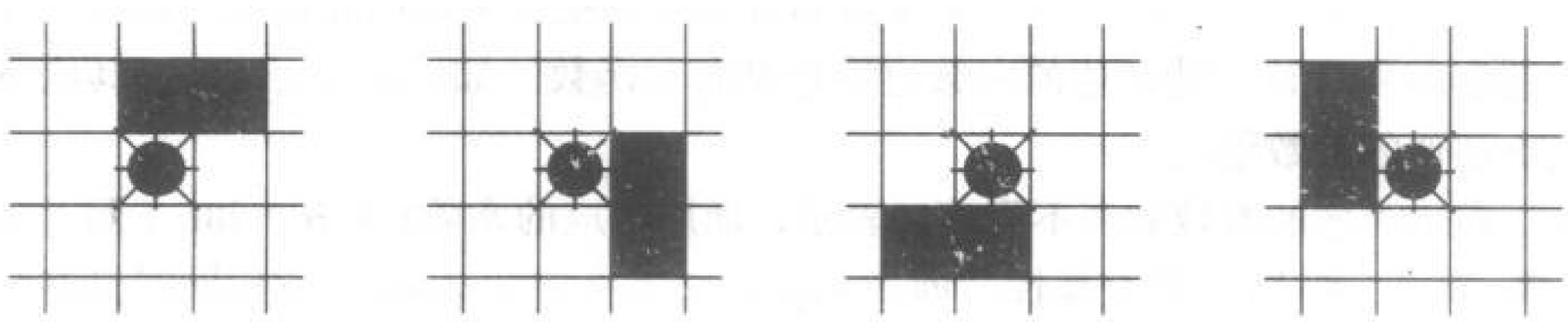
当然，知觉与动作的分界是完全任意的。可以把整个过程要么看作知觉（世界被感知，处于一个应该采取动作“北”的状态），

也可看作动作（根据原始传感器的数据而计算得出，应该采取动作“北”）。通常，分界可使得在完成各项预期任务时重复使用相同的特征。这样，不同的任务可拥有相同的特征向量和不同的动作函数。这些计算机程序对传感器信号中的特征进行的计算通常被视作重复使用库程序——被许多不同的动作计算所需要。如何分界这两个过程是这些机器的设计艺术，对此就无庸赘述了。

完成分界工作后，我们还有两个问题要解决：一是把传感器的数据转换成特征向量；另一个是确定动作函数。在后面的示例中会分别简述这两个问题。

2.1.1 感知

沿边界运动的机器人的传感器输入包括s_1, ..., s_8的值，这些值共有2^8= 256种组合方式。但在我们的环境中，由于稠密空间的限制，一些组合方式已被排除在外。对当前的任务，刚好有四个对计算适当动作有用的传感器的二进制特征值，分别用x_1、x_2、x_3和x_4来表示。它们的定义如图2-3所示。例如，x_1=1，当且仅当s_3=1或s_2=1。



在每个图中，当且仅当至少一个阴影单元为非空时，所示的特征值为1

图2-3 沿边界运动的特征

本例中知觉处理过程的计算相对比较简单，而为更复杂的世界以及具备更复杂的传感器和任务的机器人设计这一过程，将极富挑战性。而且在许多实际任务中，知觉处理过程偶尔会发出有关机器人所处环境的错误、模糊或不完全的信息。尽管这样的错误会导致不恰当的动作，但只要不经常发生，这些依任务和环境而定的动作并不会造成太大的损失。在第6章中，我会再回到知觉这一问题中来。

2.1.2 动作

有了这四个特征，现在我们必须确定它们的函数以便选择适当的沿边界的动作。我们首先提出，如果特征值都不是1（即机器人感知到它周围的单元格全部空缺），则它可向任一个方向移动直至遇到边界。我们先让它向北移动。若至少有一个特征值为1，沿边界的行动则按以下规则完成：

若x_1= 1且x_2 = 0，则向东移；

若x_2= 1且x_3= 0，则向南移；

若x_3= 1且x_4= 0，则向西移；

若x_4 = 1且x_1= 0，则向北移。

其中，这些机器人可做出不同动作的条件恰好是特征值的布尔组合，这些特征值本身也是传感器输入的布尔组合。既然一些重要的知觉和动作选择方法涉及到布尔函数，那么在继续讨论实

例之前我们有必要讨论一下它们。

2.1.3　布尔代数

布尔函数$f(x_1, x_2,..., x_n)$表示n元组（每个值为（0,1））与{0，1}集合之间的对应关系。布尔代数是说明布尔函数的一种便捷的表达方法。布尔代数使用“·”、“+”、“-”三个连接符号。两个变量的“与”函数记作$x_1 \cdot x_2$，通常连接符号可以省略，即可以记作$x_1\ x_2$。函数$x_1\ x_2$值为1，当且仅当x_1和x_2的值均为1；否则，其值为0。两个变量的或记作x_1+x_2，x_1+x_2值为1当且仅当x_1、x_2两者均为1或其中任一个为1；否则值为0。变量x的补(*complement*)或非(*negation*)记作$\bar{x}_1$。$\bar{x}_1$的值为1当且仅当x的值为0；否则其值为0。

以下便是布尔代数所给出的严格定义：

$1 + 1 = 1,\ 1 + 0 = 1,\ 0 + 0 = 0$

$1 \cdot 1 = 1,\ 1 \cdot 0 = 0,\ 0 \cdot 0 = 0$

$\bar{1} = 0,\ \bar{0} = 1$

譬如，表达式$\bar{x}_1\ \bar{x}_2\ \bar{x}_3\ \bar{x}_4 + x_1\bar{x}_1$给出了沿边界运动的机器人应向北移动的情况。在这个例子中，从传感器信号中计算特征值的函数也恰好是布尔函数。如，$x_4 = s_1 + s_8$。其他特征值和动作规则也由相似的函数给出。

有时，布尔函数的参数和值不用1和0表示，而用相应的常数T（真）和F（假）来表示。

连接符“·”和“+”具交换性。即，$x_1x_2=x_2x_1$；$x_1 + x_2 = x_2 + x_1$。它们同时具有结合性，即$x_1(x_2x_3)=(x_1x_2)\ x_3$；$x_1+(x_2+x_3)=(x_1+x_2)\ +x_3$。这样，我们可去掉括号，书写如下：$x_1\ x_2\ x_3$和$x_1 + x_2 + x_3$。

由单个变量组成的布尔函数，如x_1，称为原子(*atom*)。由单个变量或其补组成的布尔函数，如$\bar{x}_1$，称为文字(*literal*)。

在复合表达式中，先“·”后“+”的顺序不能颠倒。而且，布尔代数遵守DeMorgan定律（用前面的定义可证明此定律）：

$$\overline{f_1 f_2} = \bar{f}_1 + \bar{f}_2$$

$$\overline{f_1 + f_2} = \bar{f}_1 \bar{f}_2$$

DeMorgan定律可简化布尔函数。如，$x_1\bar{x}_2 = (s_2 + s_3)\overline{(s_4 + s_5)} = (s_2 + s_3)\bar{s}_4\bar{s}_5$.

另一个重要定律为分配律：

$$f_1(f_2 + f_3) = f_1f_2 + f_1f_3$$

2.1.4　布尔函数的类别和形式

布尔函数有多种形式，其中一种重要的形式为：$\lambda_1\lambda_2...\lambda_k$，这里$\lambda_i$为文字，这样书写的函数称为“文字合取式（*conjunction*）”或“单项式(*monomial*)”。这一合取式本身称为一个项式。x_1x_7和$x_1x_2\bar{x}_4$均为项式。项式的大小即为其所含文字的总数。刚才所举的两个项式的大小分别为2和3。

我们很容易证明n个变量可组成3^n种可能的单项式（参看习题2.4）。大小为k或更小的单项式的总数由以下公式约束：

$$\sum_{i=0}^{k} C(2n, i) = O(n^k)$$

其中，$C(2i,j)=\frac{i!}{(i-j)!j!}$为二项式系数。

一个子句是形如$\lambda_1+\lambda_2+\ldots+\lambda_k$的表达式，其中$\lambda_i$为文字。这样的形式称作“文字析取式(*disjunction*)”。$x_3+x_5+x_6$, $x_1+\bar{x}_4$ 均为子句。子句的大小也是其所含文字的总数。共存在3^n种可能的子句和少于$\sum_{i=0}^{k}C(2n,i)$ 种大小为k或小于k的子句。若f为项式，则（根据DeMorgan定律）$\bar{f}$为子句，反之亦然。这样，子句和项式互为对偶（*dual*）。

若一个布尔函数可以书写成项式析取式则称为“析取范式(*disjunctive normal form*, DNF)”。$f=x_1x_2+x_2x_3x_4$和$f=x_1\bar{x}_3+\bar{x}_2\bar{x}_3+x_1x_2\bar{x}_3$ 均为DNF。任何布尔函数都能写成DNF形式。由k个项式组成的DNF析取式称为“k项DNF表达式”；若其中最大的项式的大小为k，它就属于k-DNF这一类。刚才所举的两个例子分别称为二项式和三项式，二者均属3DNF这一类。

析取范式有一个对偶：合取范式（*conjunctive normal form*, CNF）。若一个布尔函数可书写成子句合取式(*conjunction*)，则称为CNF。$f=(x_1+x_2)(x_2+x_3+x_4)$即为CNF的一例。所有布尔函数均有CNF形式。如果一个表达式是k个子句的合取，那么它叫做k子句CNF表达式；如果其中最大子句的大小为k，那么它属于kCNF类。上面的例子是一个2子句表达式，属于3CNF类。若f是DNF形式，则根据DeMorgan定律，其CNF为$\bar{f}$，反之亦然。

2.2 动作函数的表达和执行

若存在R种可能的动作，则我们必须找出一个恰当的R值的特征向量函数来计算动作。人们已对多种表达和执行动作函数的方法进行了研究，下面介绍其中的一部分。

2.2.1 产生式系统

产生式系统是动作函数的简单表达形式之一。一个产生式系统包含一个有序规则序列，称为产生式规则(*production rule*)或产生式(*production*)。每一规则写作“$c_i\rightarrow a_i$”，其中c_i是条件部分(*condition part*)，而a_i是动作部分(*action part*)。一个产生式系统包含以下规则集：

$$
\begin{aligned}
&c_1 \longrightarrow a_1\\
&c_2 \longrightarrow a_2\\
&\quad\vdots\\
&c_i \longrightarrow a_i\\
&\quad\vdots\\
&c_m \longrightarrow a_m
\end{aligned}
$$

一般来说，一个规则的条件部分可能是对传感器输入的感知处理而产生的特征的任一二进制值（0，1）函数。通常，它是一个单项式——一个布尔合取。为了选择一个动作，选择规则如下：从第一个规则$c_1\rightarrow a_1$开始，按顺序寻找第一个条件部分值为1的规则，并选择那个规则的动作部分。该动作部分可以是一个简单动作、对另一个产生式系统的调用或者是一个要同时执行的动作集合。通常，序列中最后一条规则的条件部分的，值为1；若在此之前没有其他条件部分的值为1的规则，则缺省地执行最后一条规则中的动作。传感器的输入和以其为基础的特征值随着动作的执行也不断改变。假设条件被不停地检测，使在任一时刻执行过程中的动作与

在那一时刻的条件值为1的第一个规则相匹配。

运用布尔代数以及先前为沿边界行动的机器人而定义的特征文字，可以产生下面的沿边界行动路线的产生式系统表示：

$x_4\overline{x_1} \longrightarrow$ north

$x_3\overline{x_4} \longrightarrow$ west

$x_2\overline{x_3} \longrightarrow$ south

$x_1\overline{x_2} \longrightarrow$ east

$1 \longrightarrow$ north

沿边界行动是持续过程的一例—— 一个永不停止的(*durative*)程序，机器人永远执行这一动作。与此相反，一些任务要在满足目标(*goal*)条件后停止运动，这一目标通常用基于特征的布尔条件来表示。譬如，我们不让机器人永远沿边界行动而让其在走到一个角落后停在那儿。给出一个角落搜索特征c，当且仅当机器人在一个角落时其值为1，下面的产生式系统会让机器人去一个角落（如果确实存在一个可到达的角落）：

$c \longrightarrow$ nil

$1 \longrightarrow$ b-f

这里，nil表示什么都不做，而b-f表示沿边界运动过程，对此我们已经做了定义。

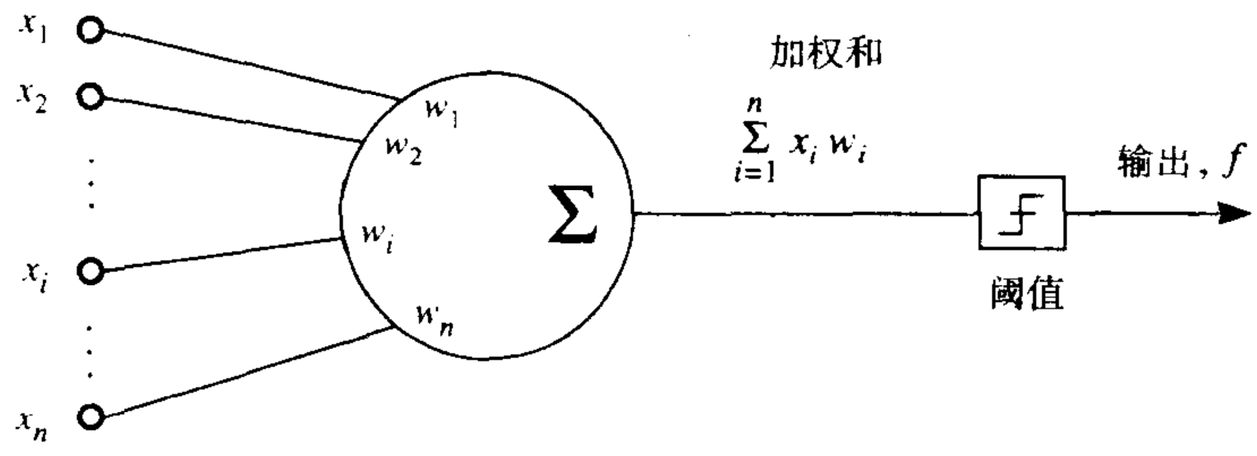
在目标实现的产生式系统中，处于规则序列首位的规则c_1的条件部分说明了我们要让动作完成的总体目标。一旦达到这一目标，agent便停止动作。当状态不满足c_1而满足c_2，通常选择条件c_2和动作a_2，然后动作a_2的完成将最终达成c_1的目标。以此类推，这种产生式系统形成了一个被称作*teleo-reactive*（T-R）程序[Nilsson 1994]的形式化基础。在这个T-R程序中，每正确完成规则序列中一条规则中的动作，就满足了此规则中一个更高的条件。若给出为agent设置的总体目标（使用基于特征的条件说明），书写此特征的产生式系统将十分容易。T-R程序也十分健全，动作一直朝着目标行进。只要感知精确度合理，由错误感知、不恰当的执行动作或对环境信息处理的偏差所造成的偶而的错误是可以得到修正的，而且动作通常能实现其设计效果[⊖]。除这些特征外，当T-R程序被调用时，可以带上参数，而且可以调用其他T-R程序，还能够递归调用自己。

2.2.2　网络

用计算机程序执行布尔函数和产生式系统并不困难。另外，它们也能直接通过电路实现[⊜]，电路输入可以是传感器信号本身。在逻辑电路中，布尔函数由逻辑门（AND，NAND，OR等）网络系统实现。一种常用的电路包括阈值元件或其他能计算其输入加权总和的非线性函数元件的网络系统，其例子之一便是如图2-4所示的阈值逻辑单元（*threshold logic unit*, *TLU*）。它对其输入的加权总和进行计算，并将结果与一阈值比较；如果超过阈值则输出1，否则输出0。

⊖ T-R程序不仅不断地向目标“行进”，而且能对它们所处的环境做出反应。前缀teleo 起源于希腊语“终极”或“目的”。

⊜ 尽管如此，这种电路的行为通常被某种电路模拟计算机程序所模拟。然而，把这种行为视为由一个电路而不是一个程序产生，能帮助我们理解动作对即刻传感器输入的状态依赖。



$$\text{如果} \sum_{i=1}^{n} x_i w_i \geq \theta \text{，} f=1$$

否则， $f=1$

图2-4 一个阈值逻辑单元

可由TLU实现的布尔函数称为线性可分函数(*linearly separable function*)（TLU用一个线性平面——在n维空间中称为超平面 —— 将产生高于阈值响应的输入向量空间与产生低于阈值响应的输入向量空间分开）。许多（但并非全部）布尔函数都是线性可分的。譬如，任一单项式（文字的合取）或任一子句（文字的析取）都是线性可分的，如图2-5所示，我给出了一个单项式的TLU实现。TLU权值绘制在其响应的输入线旁，而画在圈内的阈值表示TLU。两个变量的“异或”函数（$f=x_1\bar{x}_2+\bar{x}_1x_2$）就是一个线性不可分的函数的例子。

沿边界行动的机器人使用的函数能通过TLU实现。譬如，图2-6显示了由单个TLU对函数$\bar{x}_1\bar{x}_2=(s_2+s_3)(\overline{S_4+S_5})=(s_2+s_3)\ \bar{S}_4\ \bar{S}_5$的实现过程。

在仅有两种动作可能的应用中，若给出特征向量的代码表示输入，那么单一的TLU便可计算出正确的动作，而对更复杂的问题，则需要由这样的元件组成的神经网络系统(*neural network*)。由于TLU被认为是生物神经元的简单模型，这样的电路称为神经网络系统。生物神经元是否被激励取决于若干输入的强度的加权和。我们将在下一章中对神经网络系统进行更深入的研究。

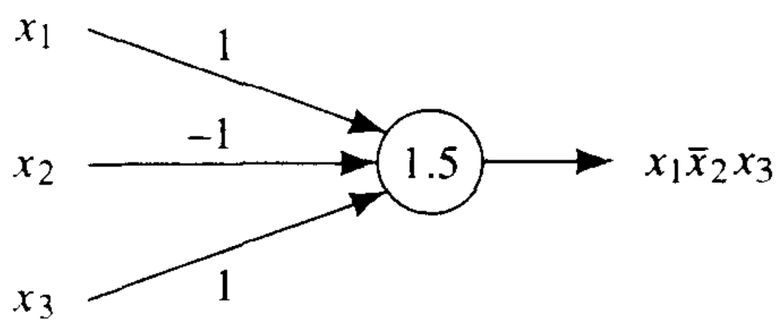


图2-5 用一个TLU实现一个单项式

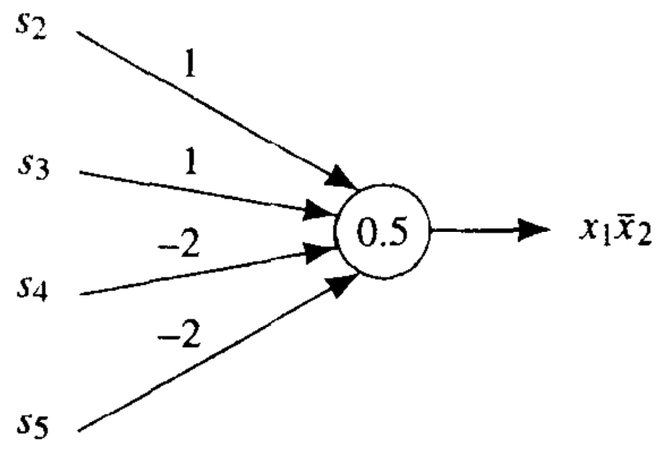


图2-6 沿边界运动的函数的实现

Katz㊀提出通过反相器和逻辑与门(AND)组合的简单网络系统结构，可用来实现任一T-R程序。图2-7中表达了我对这种网络系统的理解。网络系统的输入是状态c_i的二进制值（0，1），网络系统的输出引发相应的动作a_i（这里并未显示产生c_i的计算，既然c_i通常是文字的合取，那么它们也可以通过TLU实现）。T-R程序中的每一条规则由一个拥有两个输入和两个输出的子电

㊀ Edward Katz, private communication, 1996年4月。

路（称为TISA，即测试、禁止、压制和动作）实现。TISA中的一个TLU计算其中一个输入和另一个输入的补的合取值；另一个TLU计算两个输入的析取值。当以上规则均不满足正确的状态时，禁止输入为0；当对应规则的条件c_i满足时，测试输入为1；若测试输入为1而禁止输入为0时，动作输出为1（即引发相应的动作a_i）。当测试输入和禁止输入任一为1时，压制输出为1（即禁止以下的所有单元）。运用可编程门阵列或等价形式的动态电路结构，可通过调用一个T-R程序来动态构造运行时的电路。

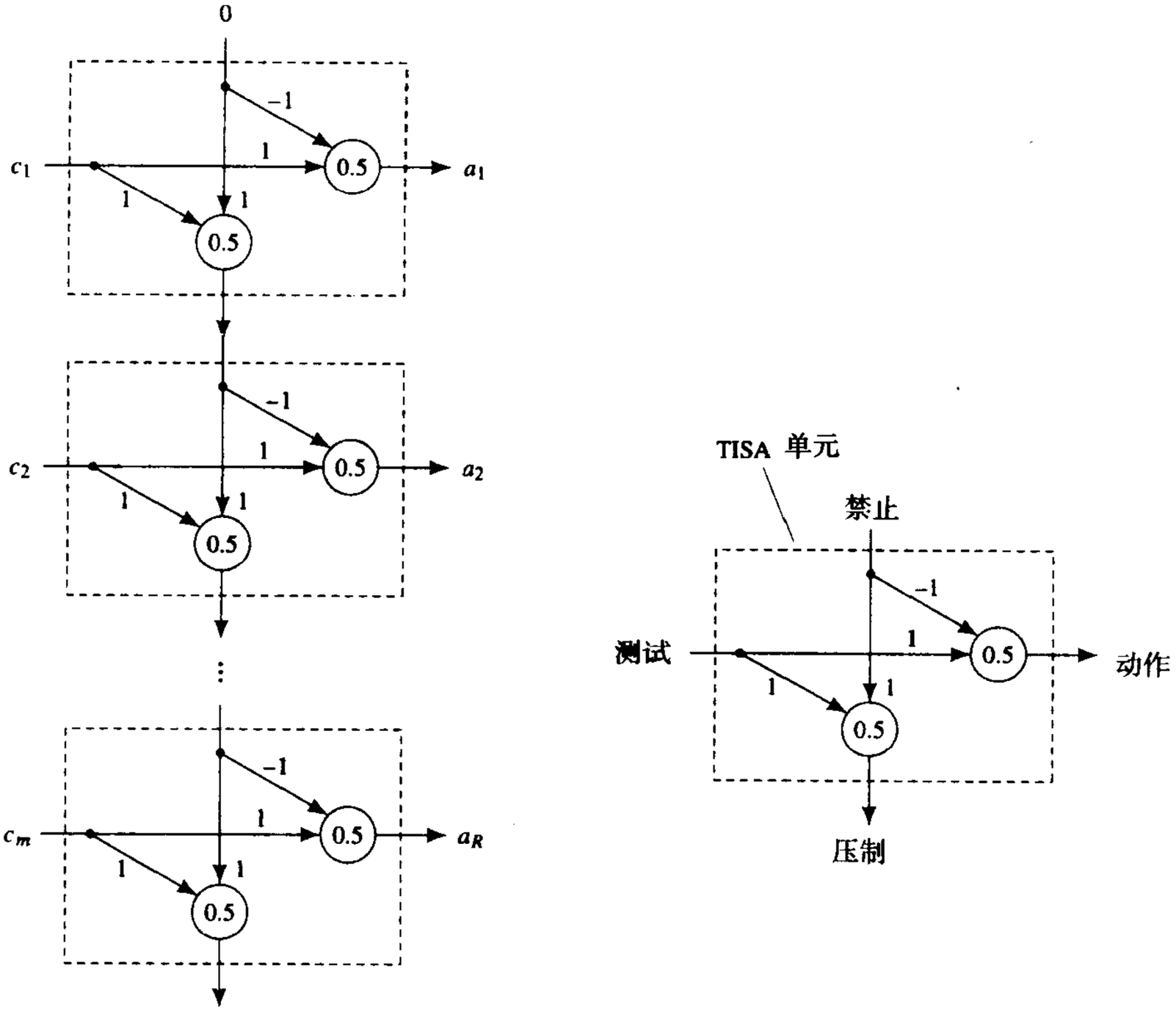


图2-7　由TISA单元组成的网络系统

2.2.3　包含体系结构

实际上还存在其他几种把传感器的即刻输入转换成动作的形式化描述。其中之一便是Rodney Brooks提出的包含体系结构（*subsumption architecture*）[Brooks 1986, Brooks 1990, Connell 1990]。虽然对如何组成这一体系结构还没有确切的定义，但其大体含义是，用一组“行为模块”来控制agent的行为。每一个模块直接从世界接受传感器信息，当传感器输入满足此模块特定的先决条件时，就执行此模块特定的行为程序。一个行为模块可将另外一个行为模块包含其中。在图2-8中，上层模块均可将下层模块包含其中。当模块*i*将模块*j*包含其中时，如果满足*i*的条件，就用模块*i*的程序替换模块*j*的程序。与“垂直”体系结构相反，Brooks称其为“水平”体系结构。Brooks和他的学生演示证明，相对简单的反应机器和复杂环境的相互作用可产生极其复杂的行为[Matatic 1990，Connell 1990]。与其他人工智能机器相比，Brooks的机器并不依赖于其所处环境的复杂的内部表达或对这些表达进行的推理[Brooks 1991a, Brooks

1991b]⊖。然而，必须意识到，尽管所有的S-R机器能做出许多有趣的动作，但它们不可避免地存在相当大的局限性([Kirsh 1991]对Brooks 的方法写了一个有趣的评论)。

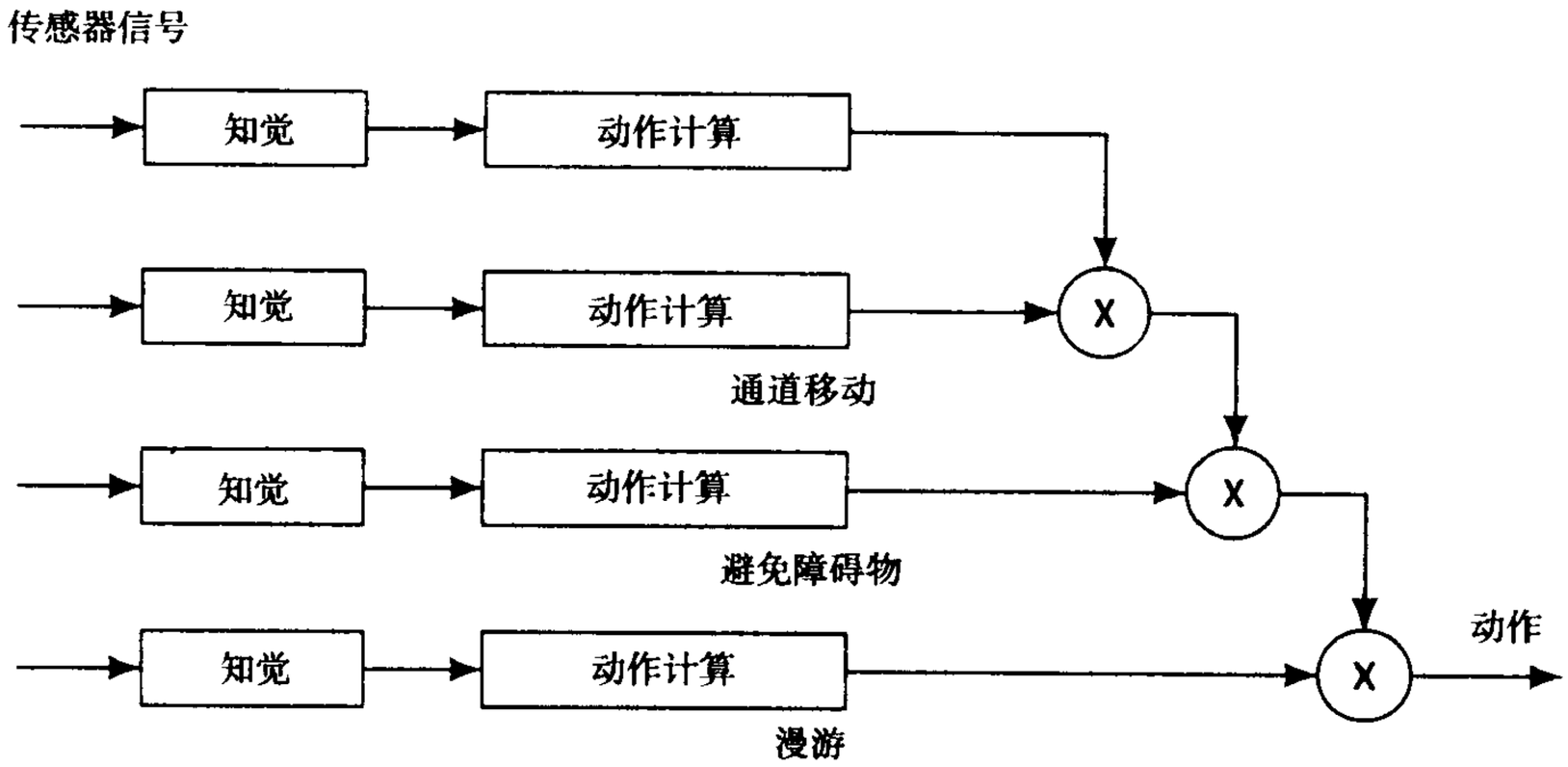


图2-8 包含模块

2.3 补充读物和讨论

S-R agent在现代电子世界中无处不在。恒温器、维持车速的巡逻控制、计算机操作系统中的中断驱动部分以及工厂里成千上万的自动装置均是 S-R agent。一般来说，这种系统和装置并不被视为人工智能的主要论题。我之所以将其纳入本书的论述，是因为它是更高级智能系统的基础。

在一些有关人工智能和机器人的实验课程中，学生们从运用Lego元件构造S-R机器人开始（请参阅[Resnick 1993]）。探索基于行为的控制策略的人工智能研究者已经研制出由探测器指引的可漫游、躲避障碍且沿墙运动的S-R机器人[Mataric 1990, Connell 1990]。

我所提出的感知与动作的分界在agent设计中也十分常见。根据[Kaelbling & Rosenschein 1990]，我们将从传感器的原始输入中计算特征向量的任务分配给感知处理，将选择基于此特征向量的动作的任务分配给动作函数。Kaelbling 和Rosenschein认为与Brooks的“水平”体系结构相反，这种体系结构方式是“垂直”的。他们对这两种方式的比较如下[Kaelbling & Rosenschein 1990 pp. 36-37]：

> Brooks认为，对感知和动作的水平分离是一种实用的agent设计研究方法……水平研究方法允许设计者同时考虑那些感知—动作的有限方面来得出特定的行为……而另一种垂直策略则建立在分离的系统模块基础之上，一部分从复合信息来源（如：感知）中提取广泛、有价值的信息，其他部分利用这些信息来达到复合目的（如：动作）。这种垂直研究方法由于其对提取信息和引发动作的固有结合而能有效利用程序设计者的成果，从而十分引人注目。

实际上，感知与动作的分离也适用于水平研究方法。特征向量可分为不同的分量——每个分量由 特定的感知设备来计算并引发单独的动作或行为，像包含系统结构一样。

⊖ 因为Brooks 的许多包含机器确实有少量的内部状态，本应该在第5章中对其进行介绍。我们之所以将其归入本章，是因为它与T-R程序有相似之处。

T-R程序和其他响应系统与动物行为学中不同的动物行为模型紧密相关。这些模型中，对动物动作目标的实现顺序的排列表明，一个动作结果成为“释放” 序列中下一动作的“刺激物”或“触发器”。T-R程序正是以这种选择条件和执行动作的方式运行。关于T-R的工作部分受到[Deutsch 1960]的启发。

动物所展示的寻觅并向其刺激物运动的特性称为趋向性。如趋光性动物朝有光的方向移动。[Genesereth & Nilsson 1987]称本章所讨论的这种agent为“趋向性agent”。

一些实验研究运用模拟而非实际的S-R机器人。有人讽刺“模拟注定会成功”[Brooks & Mataric 1993, P. 209]，但是，模拟中的不足之处可以帮助我们进一步完善感知和动作策略。而且，模拟有时确实是“真实的”东西，如计算机教育、娱乐和戏剧艺术中活生生的角色与其模拟环境及用户的交互[Bates 1994, Blumberg 1996]（Blumberg 的ALIVE系统是另一个运用动物行为学来构造交互角色的例子）。

有关布尔函数的其他介绍，请参阅[Unger 1989]。产生式系统可视为布尔函数的普遍化，称为“策略表”[Rivest 1987]。

习题

2.1 将下面的布尔函数改写为DNF形式：

$f=(x_1+x_2)(x_3+x_4)$

2.2 证明：$x_1 x_2 x_3+\bar{x}_1 x_2 x_3=x_2 x_3$。

2.3 指出下列由三个变量组成的布尔函数中，哪个能通过加权输入的单阈值元件实现，你无需求权值和阀值。

1) x_1

2) $x_1 x_2 x_3$

3) $x_1+x_2+x_3$

4) $(x_1 x_2 x_3)+(\bar{x}_1 \bar{x}_2 \bar{x}_3)$

5) 1

2.4 证明：n个变量可组成3^n个单项式，3^n个子句。

2.5 参照2.1.1节中x_1、x_2、x_3和x_4的定义及2.1.2节的动作规则。证明：在二维网格空间世界中不存在“稠密空间”这一假设暗示不可能同时满足任意两个动作规则。

2.6 手工设计一个神经网络，使其接收传感器信号输入（$s_1,s_2,\ldots,s_8$），产生由TISA单元构成的网络所需的条件输出来完成2.1.2节中为沿墙运动的机器人设计的动作规则。

2.7 本章中指出，由T-R程序指挥的行为是“一直向目标行进”。严格地讲，这一陈述正确吗？你能给出T-R程序不在最高层条件结束的情况吗？

第3章 神经网络

3.1 引言

本书其他地方将讨论机器学习(*learn*)的方法。本章将讨论S-R机器执行动作的选择计算是如何通过与每个输入的适当的动作配对的一组输入实例来学习的。虽然存在许多不同的可用的计算结构，但在这儿将集中讨论具有可调节权值的TLU网络。网络系统通过不断调节权值，直至其动作计算表现令人满意来完成学习。如上一章所述，TLU网络称为神经网络(*neural network*)是因为它模仿了生物神经元的一些特性。这里不讨论神经网络与大脑或部分大脑功能的关系，而是把这些网络严格地视为有趣且有用的工程设备。

考虑以下学习问题：给定一个由n维向量**X**组成的集合Ξ，分量为x_i，$i=1,...,n$。这些向量将是由一个响应agent的感知处理单元计算出的特征向量。这些分量的值可以是数值，也可以是布尔值（如前所述）。也已知集合Ξ中每个**X**所对应的恰当的动作a。这些动作也许是学习者所观察到的一个教师对一组输入的响应。这些相关的动作有时称为向量的“标号（*label*）”或“类别(*class*)”。集合Ξ与相应的标号组成了“训练集合(*training set*)”。机器学习的问题就是寻找一个函数，如$f(\mathbf{X})$，“令人满意地”与训练集合的成员相对应。通常，我们希望，由f计算出的动作尽可能与Ξ中向量的标号一致。因为同时给出了标号和输入向量，所以我们认为这一学习过程“受到监控”。

即使能找到一个可对训练集合做出正确响应的函数，我们又有何理由相信它能对训练过程中未遇到过的输入做出正确的响应呢？一些实验证据证明它可以，另外，还有一种理论证明：（在一定条件下）若此训练集合对于其他可能会遇到的输入种类来说具备“代表性”，那么，这些输入“可能”会“引发”“几乎正确”的输出。有关此论题的其他论述，请参阅有关“可能接近正确（*probably approximately correct*, PAC）”的机器学习理论[Kearns & Vazirani 1994, Haussler 1998, Haussler 1990]。实际上，许多方法可用来评测一个已学习过的函数对相似（但尚未觉察）输入的反应的准确程度。我会在本章的后面部分介绍其中一部分。

3.2 训练单个TLU

3.2.1 TLU几何学

首先介绍如何调节或“训练”单个TLU的权值，从而使其对某一训练集合产生正确的输出。若用一个TLU来计算一个动作，则其输入应为数值（这样才能计算加权总和）；若感知处理过程产生类别特征，那么应用某种方法将其转为数字。当然，用单个TLU来计算动作的响应机器只能根据TLU的两种可能输出做出两种动作。单TLU也称为*Perceptron*和*Adaline*（即可调节线性元素）。Rosenblatt和Widrow[Rosenblatt 1962, Widrow 1962]对它们的使用进行了广泛的探讨。对一个TLU的训练是通过调节其可变权值来实现的。用几何学解释TLU如何对其输入作出反应，能帮助我们直观地理解TLU的训练方法。

一个TLU由其权值和阈值来定义。权值（$w_1,\cdots,w_i,\cdots,w_n$）可由一个权向量**W**来表示。我用$\theta$来表示TLU的阈值。这里，我们假设输入向量**X**，具有数字分量（这样才能计算这些数字分量的加权总和）。若向量点积$s = \mathbf{X} \cdot \mathbf{W}$，比$\theta$大，则TLU输出为1；否则为0。如图3-1所示，TLU用一个线性边界把输入向量的空间分开。在二维空间里，此边界为一条线，而在三维空间里为一个平面。在多维空间里此线性边界称为“超平面”(*hyperplane*)。此超平面把$\mathbf{X} \cdot \mathbf{W}-\theta > 0$的向量与$\mathbf{X} \cdot \mathbf{W}-\theta < 0$的向量分开。超平面方程为$\mathbf{X} \cdot \mathbf{W}-\theta = 0$。我们可以通过调节阈值来改变超平面（相对原点）的位置，而通过调节权值可以改变其方向。

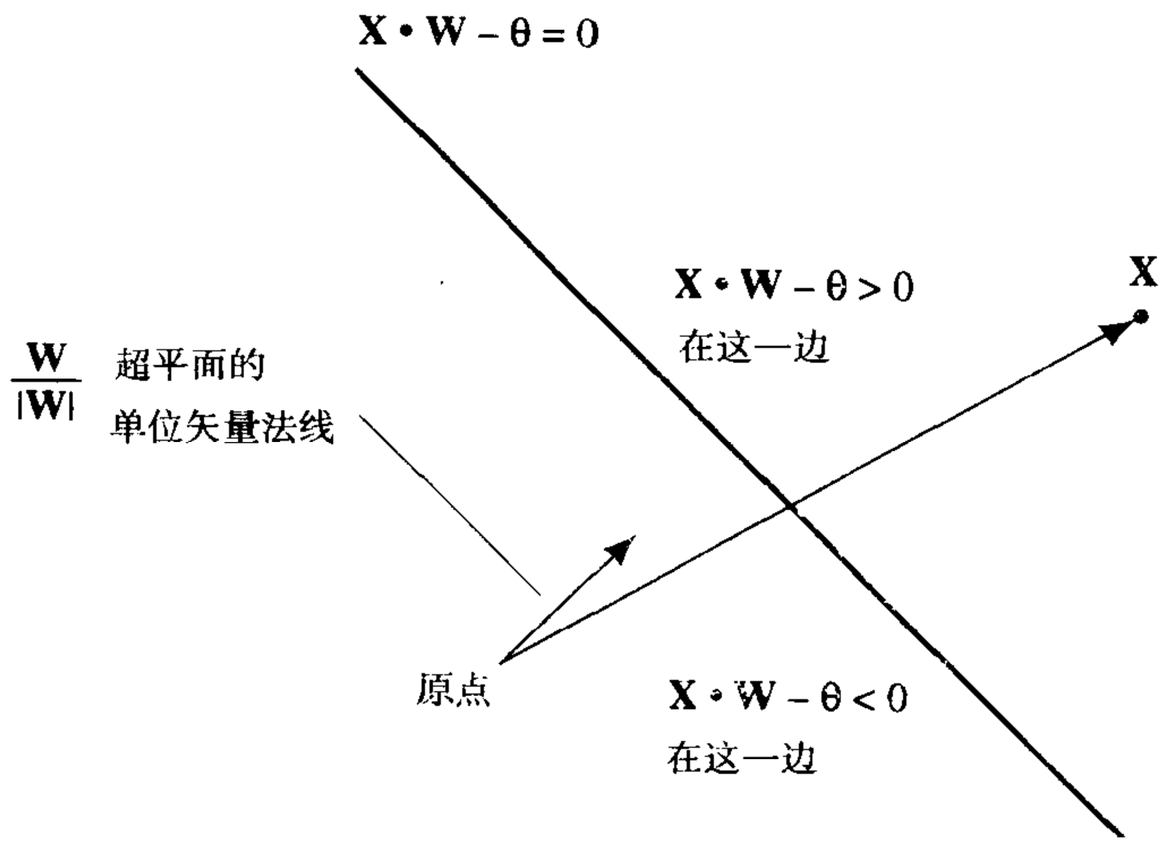


图3-1　TLU几何学

3.2.2　扩充向量

实际上，调节一个TLU的权值有好几个方法。如果我采用TLU的阈值总是等于0这一常规约定，就会简化对这些方法的解释。与此不同，我们运用$n+1$维“扩充”向量来实现任意阈值。扩充输入向量的第$n+1$个分量的值总为1；扩充权向量的第$n+1$个分量，w_{n+1}，设定为所希望的阈值θ的负数。今后，当我们运用$\mathbf{X}-\mathbf{W}$这一符号表示时，采用的就是$n+1$维扩充向量。那么，当$\mathbf{X} \cdot \mathbf{W} \geqslant 0$时，TLU的输出为1；否则为0。

3.2.3　梯度下降方法

研究训练一个TLU使其能对训练向量作出恰当响应的方法之一，就是定义一个误差函数，使其能通过调节权值达到最小值。通常使用的误差函数是平方差：

$$\varepsilon = \sum_{\mathbf{X}_i \in \Xi} (d_i - f_i)^2$$

式中，f_i是TLU对输入$\boldsymbol{X}_i$的实际响应，d_i是所希望的响应。然后我们对训练集合中的所有向量求和。对于固定的Ξ来说，ε（通过f_i）依权值而定。我们可通过下降梯度求到ε的最小值。为了计算下降梯度，先计算“权空间”中ε的梯度；然后把权向量沿此梯度的反方向（下山）移动。计算ε的梯度的困难之一，就是ε依Ξ中所有输入向量而定。通常，我们倾向于先用Ξ中的一个成员，对其权值进行调节后，再用Ξ中的另一个成员——一个运用由已作标号的输入向量所组

成的序列Σ的递增训练过程。当然，递增训练的效果只能接近所谓的批处理方式的效果，然而这一近似值通常十分有效。这里，对递增方式作一个描述。

一个单输入向量**X**（当所希望的输出为d时引发输出f）的平方差为：

$$\varepsilon = (d-f)^2$$

ε对权值的梯度为：

$$\frac{\partial \varepsilon}{\partial \mathbf{W}} = \left[\frac{\partial \varepsilon}{\partial w_1}, \cdots, \frac{\partial \varepsilon}{\partial w_i}, \cdots, \frac{\partial \varepsilon}{\partial w_{n+1}}\right]$$

（标量ϕ对向量**W**的梯度有时表示为 $\nabla_w \phi$）

由于ε对**W**的依赖完全通过点积$s = \mathbf{X} \cdot \mathbf{W}$产生，故可用链式规则（*chain rule*）书写如下：

$$\frac{\partial \varepsilon}{\partial \mathbf{W}} = \frac{\partial \varepsilon}{\partial s}\frac{\partial s}{\partial \mathbf{W}}$$

然后，因为$\frac{\partial \varepsilon}{\partial \mathbf{W}} = \mathbf{X}$，故

$$\frac{\partial \varepsilon}{\partial \mathbf{W}} = \frac{\partial \varepsilon}{\partial s}\mathbf{X}$$

注意到：$\frac{\partial \varepsilon}{\partial s} = -2(d-f)\frac{\partial f}{\partial s}$。这样，

$$\frac{\partial \varepsilon}{\partial \mathbf{W}} = -2(d-f)\frac{\partial f}{\partial s}\mathbf{X}$$

在求f对s的偏导数时我们遇到一个问题。由于阈值函数的存在，TLU的输出f对s而言不是连续可导的。点积中许多微小的变化根本无法改变f，而且f一发生变化就从1变到0或从0变到1。我将介绍解决此问题的两种程序：一种是忽略阈值函数，且令$f = s$；另一种用一个可求导的非线性函数替代此阈值函数。

3.2.4 Widrow-Hoff程序

假设我们试图通过调节权值从而使每个标号为1的训练向量产生的点积精确地等于1，使每个标号为0的向量产生的点积正好等于－1。这里，$f = s$，则增量平方差$\varepsilon=(d-f)^2=(d-s)^2$，且$\frac{\partial f}{\partial s} = 1$。这样，梯度为：

$$\frac{\partial \varepsilon}{\partial \mathbf{W}} = -2(d-f)\mathbf{X}$$

把权向量沿梯度的反方向移动，并把因数2融入学习率（*learning rate*）参数c，则权向量的新值为：

$$\mathbf{W} \longleftarrow \mathbf{W} + c(d-f)\mathbf{X}$$

当$(d-f)$为正时，把一部分输入向量加到权向量中去，这令点积变大而（$d-f$）变小；当（$d-f$）为负时，从权向量中减去一部分输入向量——则产生相反效果。这一程序就是著名的*Widrow-Hoff*或*Delta*规则[Widrow & Hoff 1960]⊖。当然，在找出一组可求出平方差的最小值的

⊖ 熟悉数字方法的人均会发现Widrow-Hoff程序是解决线性方程的relaxation method之一。

权值后（设$f=s$），就可重新考虑阈值函数来求出f的值（0或1）。

3.2.5　一般化Delta程序

Werbos[Werbos 1974]提出了另一种处理不可求导的阈值函数的程序，其他几位研究者对此也分别作了探讨，如[Rumelhart,et al. 1986]。这一方法涉及用S型可求导函数、即所谓的"sigmoid"⊖来替换阈值函数。常用的阈值函数为$f(s)=\frac{1}{1+e^{-s}}$，这里s为输入，f为输出。图3-2显示了一个sigmoid 函数及相应的阈值函数。

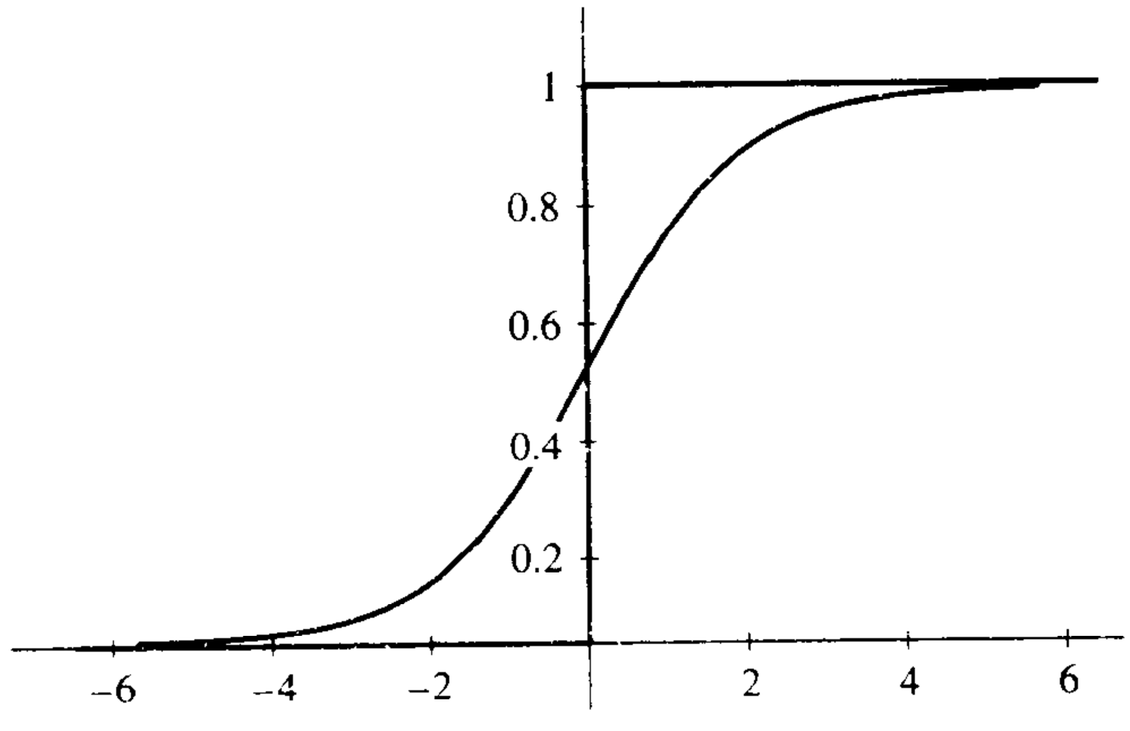


图3-2　一个sigmoid函数

选择这一sigmoid函数可得出如下偏导数：

$$\frac{\partial f}{\partial s}=f(1-f)$$

把此表达式代入$\frac{\partial \varepsilon}{\partial \mathbf{W}}=-2(d-f)\frac{\partial f}{\partial s}\mathbf{X}$，得

$$\frac{\partial \varepsilon}{\partial \mathbf{W}}=-2(d-f)f(1-f)\mathbf{X}$$

于是得到了下面的权变化规则，即"一般化Delta程序"：

$$\mathbf{W}\longleftarrow\mathbf{W}+c(d-f)f(1-f)\mathbf{X}$$

以下是Widrow-Hoff与一般化Delta的不同之处：

1) 前者所希望的输出d为1或−1，而后者为1或0；

2) 前者的实际输出f等于点积s，而后者的f为sigmoid的输出；

3) 由于sigmoid函数的存在，后者的表达式中多出了$f(1-f)$这一项。$f(1-f)$的值随sigmoid函数的变化而在0到1之间变化。当f为0时，$f(1-f)$也为0；当f为1时，$f(1-f)$也为0；当f为1/2时，$f(1-f)$为最大值1/4（即当sigmoid的输入为0时）。sigmoid函数可视为一个"模糊的"超平面。对于远离此模糊超平面的输入向量，其$f(1-f)$的值接近0，而且无论所希望的输出如何，一般化Delta程序对权值的改变甚微，甚至无法改变。在围绕模糊超平面的区域之内（这是惟一一个权变化对f颇有影响的地方）才有权变化，而且这些变化不断地纠正错误。

在一般化Delta程序中找到一组权值后，如有需要还可再用阈值函数替换sigmoid函数。

⊖ [Russell & Norving, P.595]把此观点的运用归功于[Bryson & Ho 1969]。

3.2.6 纠错程序

在下面这种方法中，我们保留阈值不变（并不替换成sigmoid函数），而且仅当TLU的响应出错时（即当（$d-f$）的值为1或-1时）才调节权向量。这一程序称为“纠错程序”。其改变权值的规则如下：

$$\mathbf{W} \longleftarrow \mathbf{W} + c(d-f)\mathbf{X}$$

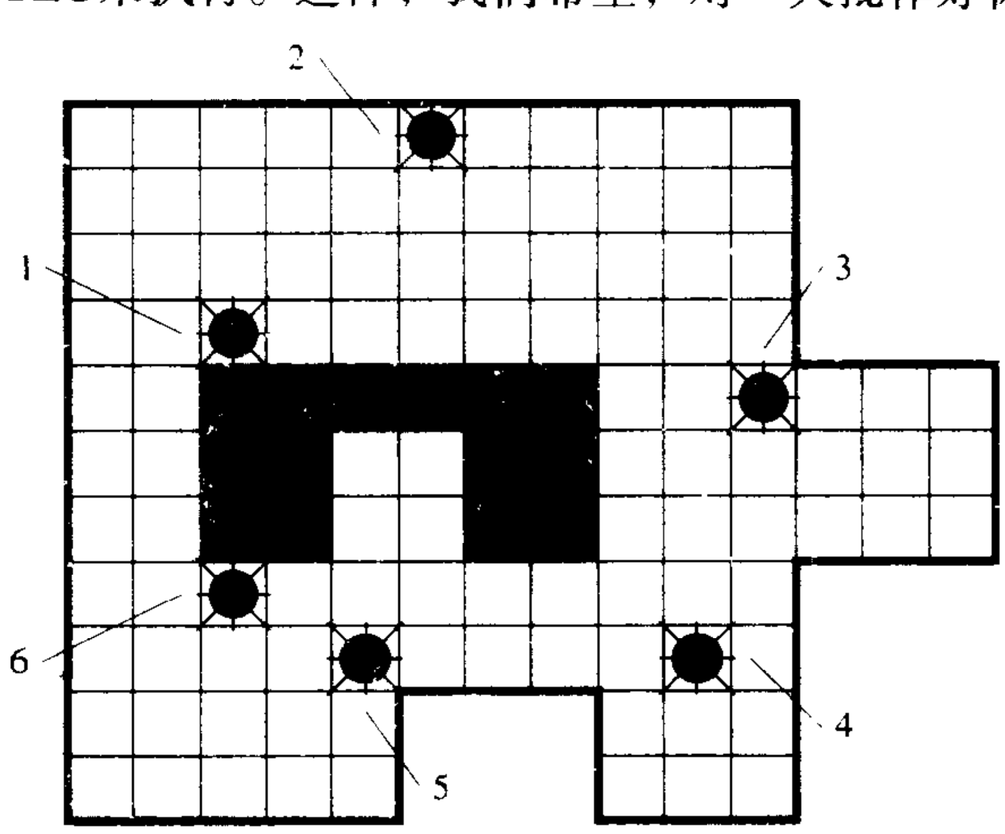
当然，这一改变是趋向于纠正错误的（也许能彻底纠正错误，这依赖于学习率参数c的值）。这一规则与Widrow-Hoff的区别是：前者的d和f的值为0或1，而后者的d为1或-1，且$f=s$为点积的值。

可以证明，若存在某一个为Ξ中所有输入向量产生正确输出的权向量**W**，那么，在经过有限的输入向量的表达式之后，纠错程序最终能得到此权向量，从而不再做权值的调节（尽管这一程序确保在表达完有限个Σ中的输入向量之后便终止，但这一证明要求每个输入向量均在训练序列Σ中出现无限次）。对于非线性可分的输入向量集合来说，纠错程序会永不停止，从而不能用来寻找一个“最佳”权向量来作为判定错误的标准。然而，Widrow-Hoff和一般化Delta程序都可找到最小平方差解，尽管这时最小误差不为0。

在第2章中，我们碰到过线性可分函数。回想一下为沿边界运动的机器人设定的产生式规则。下面是其中的一个规则：

$x_1\overline{x_2} \longrightarrow \mathtt{east}$

将其改写为与传感器输入有关的表达式，得到 $x_1\overline{x_2} = (s_2 + s_3)\overline{s_4}\,\overline{s_5}$ 。这一函数是线性可分的，并且可以由如图2-6所示的TLU来执行。这样，我们希望，对一大批作好标号的传感器的向量



输入数量	传感器矢量	$x_1\overline{x_2}$ 向东移动
1	00001100	0
2	11100000	1
3	00100000	1
4	00000000	0
5	00001000	0
6	01100000	1

图3-3 学习何时向东移动的一个训练集合

的纠错训练，最终可使TLU正确地区分让机器人东移的输人和不让其东移的输人。如图3-3 所示，我们可把训练集合排列到一起。你也许想用这些向量以及其他已作标号的输入向量来试试纠错程序（千万别忘了输入$s_9 \equiv 1$和加权值w_9！）可是，经过训练的TLU如何处理那些未训练过的输人呢？训练集合应包括多少输人向量才能使TLU的表现令人满意呢？

至于纠错程序的背景、引用、证明和实例，请参阅［Nilsson 1965］。

3.3 神经网络

3.3.1 动机

实际存在许多单TLU学不会的刺激—响应集合（这些训练集合可能是线性不可分的）。这种情况下，TLU网络却可以给出正确的响应。由TLU网络执行的函数不仅依赖于其拓扑，而且依赖于单TLU的权值。“前向网络(*feedforward networks*)”没有循环：在此网络中，任一TLU的输人并不（通过0个或多个中间TLU）依赖于这一TLU的输出(不具前向性的网络称为“递归网络(*recurrent network*)”。在后面的章节中，我们将学习一例递归网络)。若用层来组织前向网络中的TLU，且层j 的组件只接受来自层$j-1$的TLU的输人，那么，我们称这样的网络为“分层前向网络(*layered feedforward network*)”。图3-4中的网络执行函数$f = x_1x_2 + \bar{x}_1\bar{x}_2$（称为“偶校验（*even-parity*）”函数），它是一个有两个层的（具有权值）分层前向网络（有些人在计算TLU层个数时，把输人也算作一层，因此，他们会称此图中的网络为三层网络）。

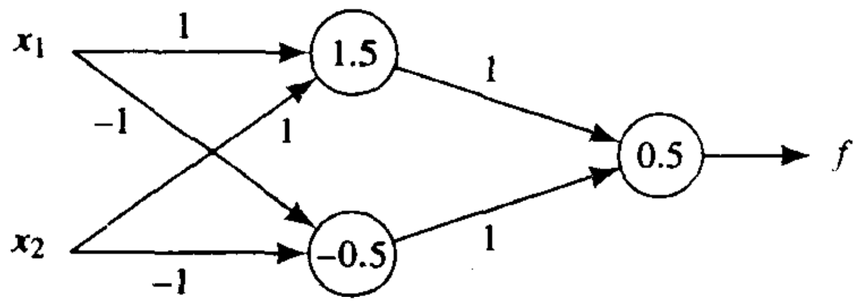


图3-4　执行偶校验函数的TLU网络

下一节将介绍训练多层前向网络的通用程序，即运用梯度下降的“反向传播程序(*backpropagation*)”。既然将求出误差函数对权向量的偏导，我们就把网络中所有阈值函数换成sigmoid函数（训练完毕，再把sigmoid函数换成阈值函数）。

3.3.2 表示符号

在图3-5中，我给出了一个由sigmoid单元组成的通用k层前向网络。除了在最后一层中的，其他所有的sigmoid单元称为“隐藏单元(*hidden unit*)”，因为它们的输出仅间接影响最终输出。图3-5中的网络只有一个输出单元；当存在两种以上的动作和输人类型时，应该使用几个输出单元（这时，应用一个译码表（*decoding scheme*）来把输出向量转换成类型[Brain, et al. 1962]采用的是基于最大移位寄存器序列的代码，［Dietterich & Bakiri 1991, Dietterich & Bakiri 1995］也对相似的纠错代码进行了研究）。

用图3-5来介绍一些有用的表示符号。正如同输人特征是一个输人向量的分量一样，我们认为每一个sigmoid单元层的输出也是向量的分量。sigmoid的第j（$1 \leqslant j < k$)层将把向量$\mathbf{X}^{(j)}$作为其输出。然后，这一向量成为$j+1$层的输人向量。输人向量用$\mathbf{X}^{(0)}$来表示，（第k层TLU的）最终输出为f。每一层中的每一个sigmoid均有一个权向量（与其输人相关）；第j层的第i个sigmoid单元的一个权向量用 $\mathbf{W}_i^{(j)}$来表示。如前所述，我们假设“阈值权”是相关权向量的最后一个分量。我们用 $s_i^{(j)}$ 来表示j层的第i 个sigmoid单元的加权总和输人。一个sigmoid单元的这一输人可称为

这一个单元的“激励（*activation*）”，由以下公式给出：

$$s_i^{(j)} = \mathbf{X}^{(j-1)} \cdot \mathbf{W}_i^{(j)}$$

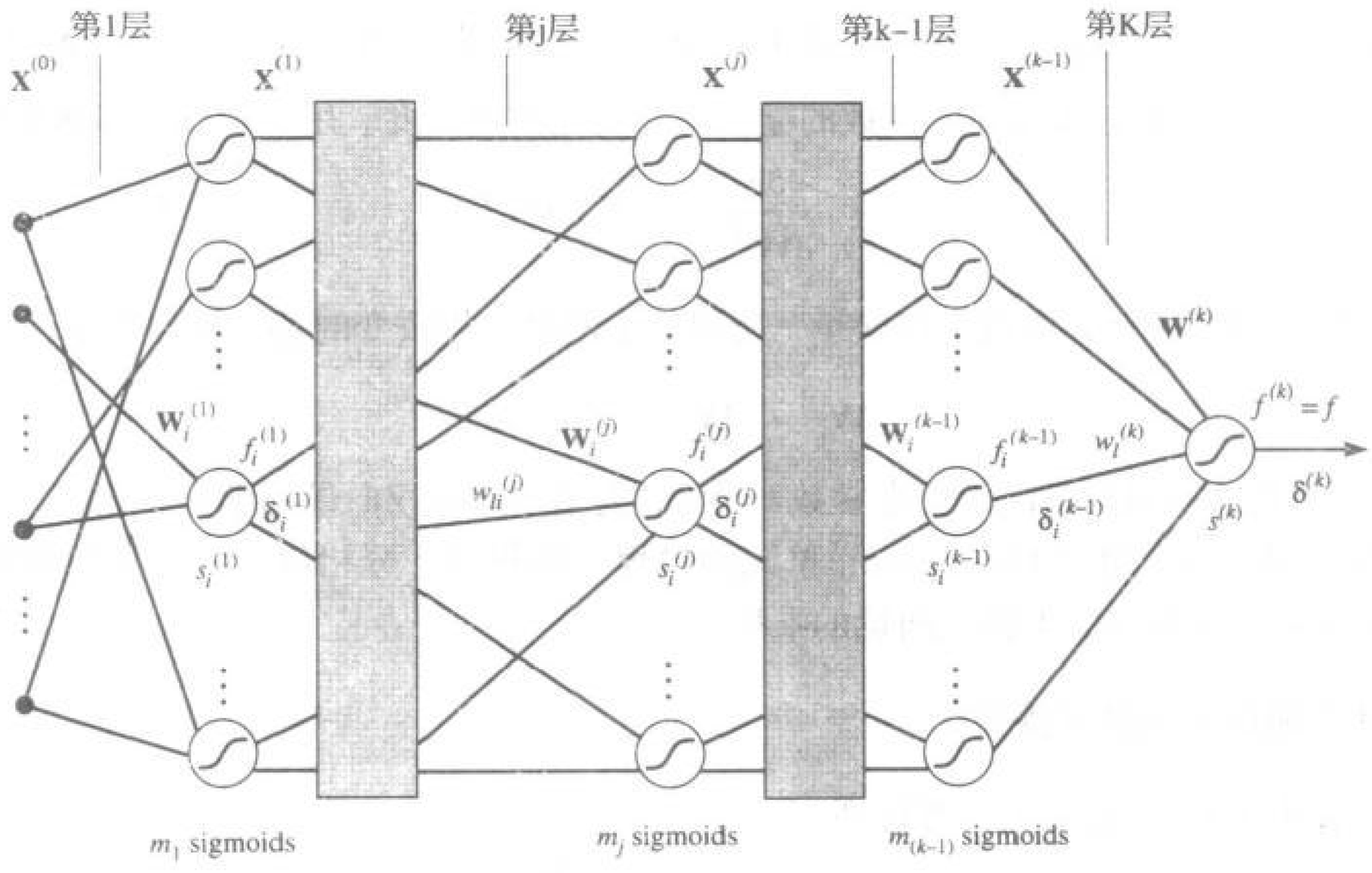


图3-5 *k*层sigmoid单元网络

第*j*层中sigmoid单元的数目用m_j来表示。向量$\mathbf{W}_i^{(j)}$的组件为$W_{l,i}^{(j)}$，这里$l = 1, m_{(j-1)}+1$。

3.3.3 反向传播方法

我们将计算平方差函数$\varepsilon = (d-f)^2$的梯度。此处用来计算梯度的权向量应包含网络中所有权值，然而，推导ε相对于单sigmoid的权向量相应的各组权值的偏导比较容易。ε相对$\mathbf{W}_i^{(j)}$的偏导为：

$$\frac{\partial \varepsilon}{\partial \mathbf{W}_i^{(j)}} \stackrel{\text{def}}{=} \left[\frac{\partial \varepsilon}{\partial w_{1i}^{(j)}}, \cdots, \frac{\partial \varepsilon}{\partial w_{li}^{(j)}}, \cdots, \frac{\partial \varepsilon}{\partial w_{m_{j-1}+1,i}^{(j)}} \right]$$

其中$\mathbf{W}_{li}^{(j)}$是$\mathbf{W}_i^{(j)}$的第l个分量。

如前所述，既然ε是完全通过$s_i^{(j)}$来依赖于$W_i^{(j)}$的，我们可用链式规则书写如下：

$$\frac{\partial \varepsilon}{\partial \mathbf{W}_i^{(j)}} = \frac{\partial \varepsilon}{\partial s_i^{(j)}} \frac{\partial s_i^{(j)}}{\partial \mathbf{W}_i^{(j)}}$$

而且因为$s_i^{(j)} = \mathbf{X}^{(j-1)} \cdot \mathbf{W}_i^{(j)}$，$\frac{\partial s_i^{(j)}}{\partial \mathbf{W}_i^{(j)}} = \mathbf{X}^{(j-1)}$。将其代入上式，得

$$\frac{\partial \varepsilon}{\partial \mathbf{W}_i^{(j)}} = \frac{\partial \varepsilon}{\partial s_i^{(j)}} \mathbf{X}^{(j-1)}$$

现在，$\frac{\partial \varepsilon}{\partial s_i^{(j)}} = \frac{\partial (d-f)^2}{\partial s_i^{(j)}} = -2(d-f)\frac{\partial f}{\partial s_i^{(j)}}$，所以

$$\frac{\partial \varepsilon}{\partial \mathbf{W}_i^{(j)}} = -2(d-f)\frac{\partial f}{\partial s_i^{(j)}}\mathbf{X}^{(j-1)}$$

量$(d-f)\frac{\partial f}{\partial s_i^{(j)}} = -\frac{1}{2}\frac{\partial \varepsilon}{\partial s_i^{(j)}}$在我们的计算中十分重要，可用$\delta_i^{(j)}$来表示。每一个$\delta_i^{(j)}$可反映出网络输出的平方差对相应的sigmoid函数的输入中的变化的灵敏度。用δ书写的ε的梯度如下：

$$\frac{\partial \varepsilon}{\partial \mathbf{W}_i^{(j)}} = -2\delta_i^{(j)}\mathbf{X}^{(j-1)}$$

因为我们将沿梯度的反方向改变权向量，所以整个网络的权改变的基本规则如下：

$$\mathbf{W}_i^{(j)} \leftarrow \mathbf{W}_i^{(j)} + c_i^{(j)}\delta_i^{(j)}\mathbf{X}^{(j-1)}$$

这里$C_i^{(j)}$是这一权向量的学习率常数（通常，网络中所有权向量的学习率相同）。我们发现，这一规则与用于一个单TLU或sigmoid单元的过程中的规则十分相似。一个权向量的改变值为一个因数与其（未加权的）输入向量的乘积。

3.3.4　计算最后一层的权值变化

接下来计算$\delta_i^{(j)}$。根据定义可得到：

$$\delta_i^{(j)} = (d-f)\frac{\partial f}{\partial s_i^{(j)}}$$

为了计算最后一个sigmoid单元的权变化，我们先来计算$\delta^{(k)}$

$$\delta^{(k)} = (d-f)\frac{\partial f}{\partial s^{(k)}}$$

因为f是sigmoid函数，$s^{(k)}$是其输入，所以得（如前）$\frac{\partial f}{\partial s^{(k)}} = f(1-f)$，这样：

$$\delta^{(k)} = (d-f)f(1-f)$$

所以，最后一层单元的反向传播权值调节可写为：

$$\mathbf{W}^{(k)} \leftarrow \mathbf{W}^{(k)} + c^{(k)}(d-f)f(1-f)\mathbf{X}^{(k-1)}$$

当最后一个sigmoid单元是网络中惟一的单元且$\mathbf{X}^{(k-1)} = \mathbf{X}$时，这一规则就与一般化Delta程序中所给出的规则相同。

3.3.5　计算中间层的权值变化

运用δ的表达式，我们可用相似的方法计算出如何改变网络中的任一权向量。回想一下：

$$\delta_i^{(j)} = (d-f)\frac{\partial f}{\partial s_i^{(j)}}$$

我们再次使用链式规则，最终输出f是通过$j+1$层的sigmoid的每一个总和输入来依赖$s_i^{(j)}$的。所以：

$$\delta_i^{(j)} = (d-f)\left[\frac{\partial f}{\partial s_1^{(j+1)}}\frac{\partial s_1^{(j+1)}}{\partial s_i^{(j)}} + \cdots + \frac{\partial f}{\partial s_l^{(j+1)}}\frac{\partial s_l^{(j+1)}}{\partial s_i^{(j)}} + \cdots + \frac{\partial f}{\partial s_{m_{j+1}}^{(j+1)}}\frac{\partial s_{m_{j+1}}^{(j+1)}}{\partial s_i^{(j)}}\right]$$

$$= \sum_{l=1}^{m_{j+1}} (d-f) \frac{\partial f}{\partial s_l^{(j+1)}} \frac{\partial s_l^{(j+1)}}{\partial s_i^{(j)}} = \sum_{l=1}^{m_{j+1}} \delta_l^{(j+1)} \frac{\partial s_l^{(j+1)}}{\partial s_i^{(j)}}$$

现在，还应计算 $\frac{\partial s_l^{(j+1)}}{\partial s_i^{(j)}}$ ，为了便于计算，先这样写：

$$s_l^{(j+1)} = \mathbf{X}^{(j)} \cdot \mathbf{W}_l^{(j+1)}$$

$$= \sum_{v=1}^{m_j+1} f_v^{(j)} w_{vl}^{(j+1)}$$

然后，因为权值不依赖于s，所以：

$$\frac{\partial s_l^{(j+1)}}{\partial s_i^{(j)}} = \frac{\partial \left[\sum_{v=1}^{m_j+1} f_v^{(j)} w_{vl}^{(j+1)} \right]}{\partial s_i^{(j)}} = \sum_{v=1}^{m_j+1} w_{vl}^{(j+1)} \frac{\partial f_v^{(j)}}{\partial s_i^{(j)}}$$

我们注意到，除非$v=i$（这时，$\frac{\partial f_v^{(j)}}{\partial s_v^{(j)}} = f_v^{(j)}(1-f_v^{(j)})$，$\frac{\partial f_v^{(j)}}{\partial s_i^{(j)}} = 0$。）所以：

$$\frac{\partial s_l^{(j+1)}}{\partial s_i^{(j)}} = w_{il}^{(j+1)} f_i^{(j)} (1-f_i^{(j)})$$

我们把上式代入 $\delta_i^{(j)}$ 的表达式中，得

$$\delta_i^{(j)} = f_i^{(j)}(1-f_i^{(j)}) \sum_{l=1}^{m_{j+1}} \delta_l^{(j+1)} w_{il}^{(j+1)}$$

上面这一等式是δ的递归等式（有趣的是，我们发现这一等式与误差函数无关，误差函数仅直接影响$\delta^{(k)}$的计算）。计算出$j+1$层的 $\delta_i^{(j+1)}$ 后，可用此等式来计算 $\delta_i^{(j)}$ 。我们已经计算出基数$\delta^{(k)}$：

$$\delta^{(k)} = (d-f) f (1-f)$$

在一般权变化规则中把这个表达式用于δ，即

$$\mathbf{W}_i^{(j)} \leftarrow \mathbf{W}_i^{(j)} + c_i^{(j)} \delta_i^{(j)} \mathbf{X}^{(j-1)}$$

这一规则虽略显复杂，但它有一个直观合理的解释。量$\delta^{(k)}=(d-f)f(1-f)$控制了网络中所有权值调节的整体数量和符号（权值调节随着最终输出渐渐趋向0或1而减小，因为它们对f的影响逐渐消失。）正如δ的递归等式所示，对“进入”j层的一个sigmoid单元里的权值的调节与此调节对这个sigmoid单元的输出的影响是成比例的（此比例因数为（$f^{(j)}(1-f^{(j)})$）。这些调节也与这个sigmoid单元的输出的任意变化对最终输出的一种“平均”影响成比例。这种平均影响取决于两个因素：一个是从j层sigmoid单元“输出”的权值（权值越小，它对“下游”的影响越小）；另一个是$j+1$层sigmoid单元输出的变化对最终输出的影响（由$\delta^{(j+1)}$来衡量）。通过反向的权值反向传播可轻松地完成这些计算（这一算法名为“backprop”）。有关其他backprop及其应用的信息，请参阅［Chauvin & Rumelhart 1995］。

作为反向传播处理过程的一例，我们来看看图3-6中训练神经网络的一个步骤——从图中

的随机权值开始（为了使运用backprop方程更加容易，这里采用标准的网络表达方式）。我们的目标函数是由两个二进制变量组成的偶校验函数。输入（包括阈值输入）和所希望的标号为：

1) $x_1^{(0)}=1$，$x_2^{(0)}=0$，$x_3^{(0)}=1$，$d=0$

2) $x_1^{(0)}=0$，$x_2^{(0)}=0$，$x_3^{(0)}=1$，$d=1$

3) $x_1^{(0)}=0$，$x_2^{(0)}=1$，$x_3^{(0)}=1$，$d=0$

4) $x_1^{(0)}=1$，$x_2^{(0)}=1$，$x_3^{(0)}=1$，$d=1$

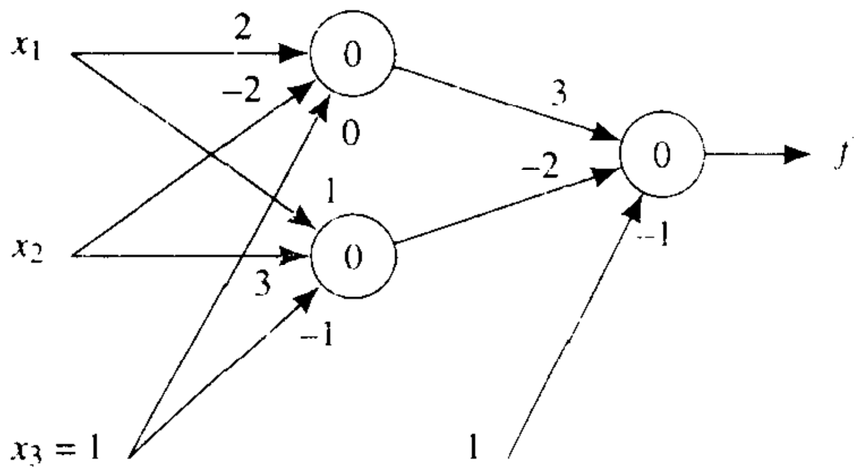


图3-6　用backprop训练的一个网络

第一个输入向量为(1, 0, 1)，它引发第一层面的输出$f_1^{(1)}=0.881$、$f_2^{(1)}=0.500$以及最终输出$f=0.665$。我们用基数方程算出$\delta^{(2)}=-0.148$。通过第二层的权值反向传播这个δ，得出$\delta_1^{(1)}=-0.047$和$\delta^{(1)}=0.074$。然后，运用权调节方程（和一个学习率常数$c=1$），得出新权值：

$$\mathbf{W}_1^{(1)}=(1.953,\ -2.000\ -0.047)$$
$$\mathbf{W}_2^{(1)}=(1.074,\ 3.000,\ -0.926)$$
$$\mathbf{W}^{(2)}=(2.870,\ -2.074,\ -1.148$$

我们注意到，因为$x_2^{(0)}=0$，$W_{2,2}^{(1)}$和$W_{1,2}^{(1)}$未被调节（即使$f_2^{(1)}=500$——这里是对调节的敏感度最高的地方）。对第一层的调节使$f_1^{(1)}$和$f_2^{(1)}$的值减小，然后它与第二层面的权值调节一起使f的值减小，同时也使这个输入向量的误差减小。试着写一个程序或用电子表格来继续这个训练过程，你定会受益匪浅。

3.4　一般化、准确度和过度拟合

上一节中介绍的神经网络的训练绝非典型。因为其维数太低，我们能训练所有四个可能的输入向量。根据图3-4，存在一组对所有输入向量进行正确分类的权值。然而，多数应用中的维数要高得多，通常为100或100以上。有100个二进制元素就有2^{100}种可能的输入向量。我们只可能训练其中的一小部分。而且，即使网络能对整个训练集合正确归类（这是不太可能的[⊖]），也无法保证它会按我们的要求对其他向量进行正确的归类。当一个网络可对不属于训练集合的向量进行正确的分类时，就称之为“一般化(*generalize*)”网络。一般化能力由其归类的准确度来衡量。后面会介绍如何估量一般化准确度。

为什么我们希望神经网络能够一般化？这与曲线拟合有些类似。当我们试图拟合一条直线或一条低次多项式曲线时，若已知大量数据且对这些数据的拟合十分贴切，那么，我们就有信心挖掘出隐含在数据中的函数关系。这一拟合曲线可用来估计（具有合理的可信度）在拟合过程中未曾使用过的新数据点。若一条直线对数据的拟合不贴切，那么我们也许会再用一个二次曲线来试试，依此类推。神经网络与此十分相似。一个神经网络计算出其输入的一个复杂的非线性函数。如果对这些输入的归类实际上是某一函数，这个函数与由网络执行的函数组中的某一个十分接近，同时，如果一个已经训练的网络对训练数据的拟合十分贴切——那么，对于大量输入来说，这一训练过程很可能挖掘出了输入与归类之间隐含的函数关系。这样的话，将能

⊖ 执行总会受噪声限制。输入向量的有些分量可能有噪声。如某自于有噪声的传感器。或者训练集合中的有些分量被标错了。前者为属性噪声，后者为分类噪声。

对新输入很好地一般化。

训练输入向量的数量应当比网络的自由度的数量（即可变权值的数量）大，这一点十分重要。我们再次运用曲线拟合这一类比来帮助理解。即使一条直线无法完全拟合数据，但若存在m个数据点，那么就必定能找到$m-1$次多项式来完全拟合数据。然而，由于不管这m个数据点如何布局，我们都可以找到这样一种拟合方式，因而无法发现数据的特殊之处。与此相反，我们过度拟合数据。若数据有噪声（也许，一条直线可拟合没有噪声的数据），那么，额外增加的自由度本质上正好拟合了噪声。众所周知，曲线拟合中，数据点的数量比用来拟合的多项式的次数大得多。而且，有了足够的数据，Occam的Razor原则指出，应选择那个能适度拟合数据的最低次多项式[⊖]。在图3-7中，通过图解说明了曲线拟合的一些观点。在图中，与没有像高次函数那样过度拟合数据的曲线和简单的直线相比，二次函数提供了一种更恰当的拟合方式。与此相似的原则也适用于神经网络。

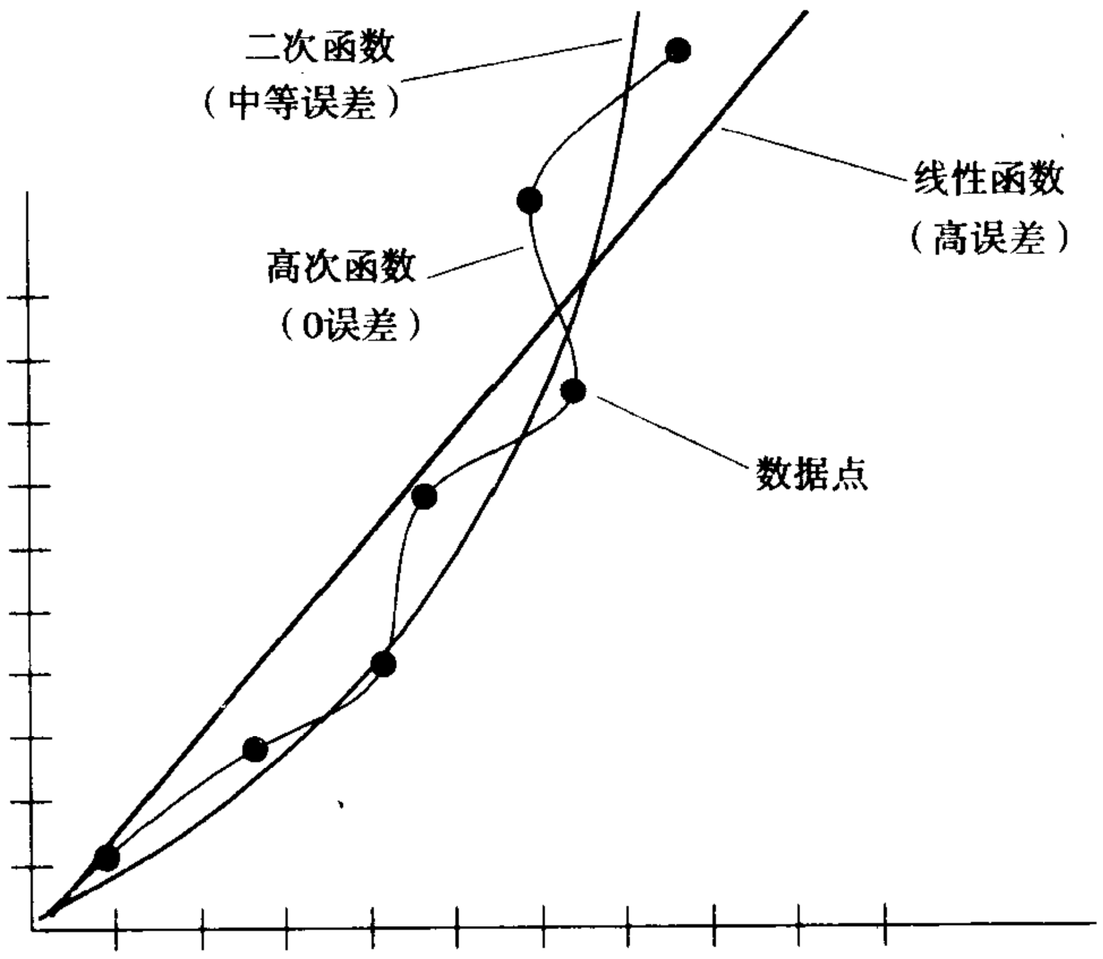


图3-7 曲线拟合

我们来考虑一下一个有n个输入、h个隐藏层单元和一个输出的二层前向神经网络。这样的网络有接近$nh+h=(n+1)h$个可变权值（不包括阈值权值）。对于固定的输入维来说，自由度实质上由隐藏单元的数量来控制。我们希望，对训练集合的归类的误差百分比随隐藏单元的数量的增加而减小——直至达到最小误差百分比（当然，若训练集合是线性可分的，那么，我们只要用一个隐藏单元就可以使训练集合达到零误差）。图3-8中，给出了一个典型的例子，其中训练集合误差是隐藏单元的数量的函数。

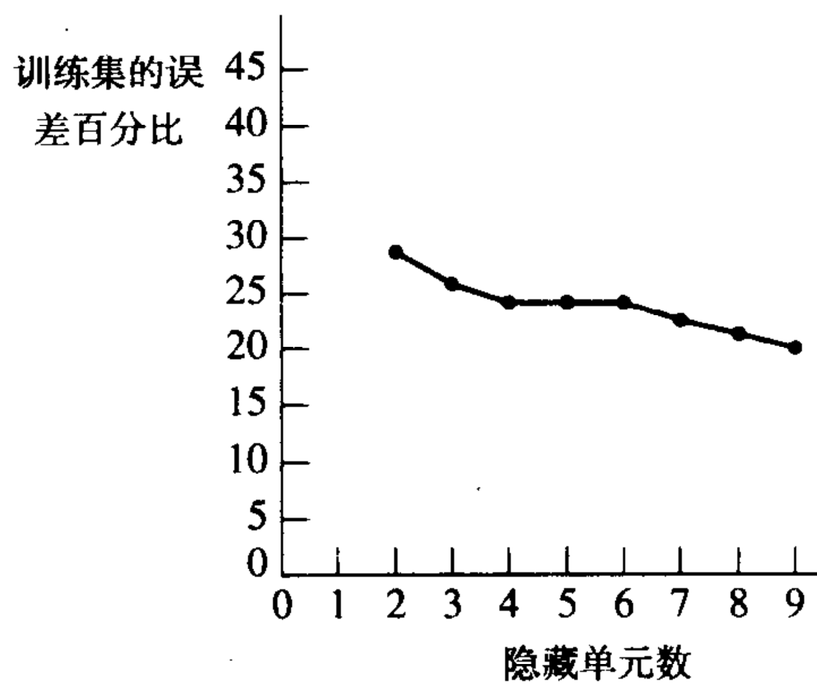


图3-8 误差与隐藏单元的数量（从[Duda, Hart & Stork 1998]中改编）

⊖ William of Occam (1285~1349年)是英国的一位哲学家，据说他曾这样说过：实体在没有需要时不应被扩大。

然而，如前所述，为了避免过度拟合，我们并不需要$(n+1)h$这么多的隐藏单元，因为它已接近训练输入向量的数量了。

尽管训练集合误差小，但其一般化并不一定好。我们可用各种方法来估计对不属于训练集合、但来自与训练集合相同的内在布局的误差率。这一误差率在统计学中称为“抽样（*out-of-sample-set*）”误差率。也许，最简单的技术就是把有待训练的输入向量分为两个互不相交的集合，然后用其中一个集合来进行训练，称之为“训练集合(*training set*)”。训练完毕，用另一个集合来估量抽样误差率，称之为“检验集合”。若两个集合中的向量的数量均很多，那么，检验集合所得出的误差率就是对一般化准确度的合理估量（当然，它常常高估了实际的抽样误差率。为什么？）。有经验的设计者会把2/3的可变向量归入训练集合，而另外1/3则归入检验集合。

另一种流行的估量一般化准确度的方法称为“交叉检验(*cross validation*)”。这种方法把有待训练的向量分成k（通常k为10）个互不相交的子集，即所谓的“fold”。然后，选其中一个fold作为检验集合而另外k-1个（组合起来）作为训练集合。这样做k次，每次选不同的一个fold作为检验集合而剩下的作为训练集合（在对相应的训练集合训练完毕后）。为每个检验集合计算误差率，并求出这些误差率的平均值作为对抽样误差的估量。在$k=m$的特例中，m为可使用的有标号的向量的数量，我们进行所谓的“排除一个（*leave-one-out*）”的交叉检验。实验结果表明，有10个fold的交叉检验所给出的一般化准确度（略显悲观）比较合理。图3-9中，通过图解说明了为解决这一典型的归类问题，以及检验集合对抽样误差率的估量是如何随隐藏单元的数量变化而变化的。注意，检验集合误差是怎样开始由于过度拟合而随隐藏单元的数量的增加而增大的。

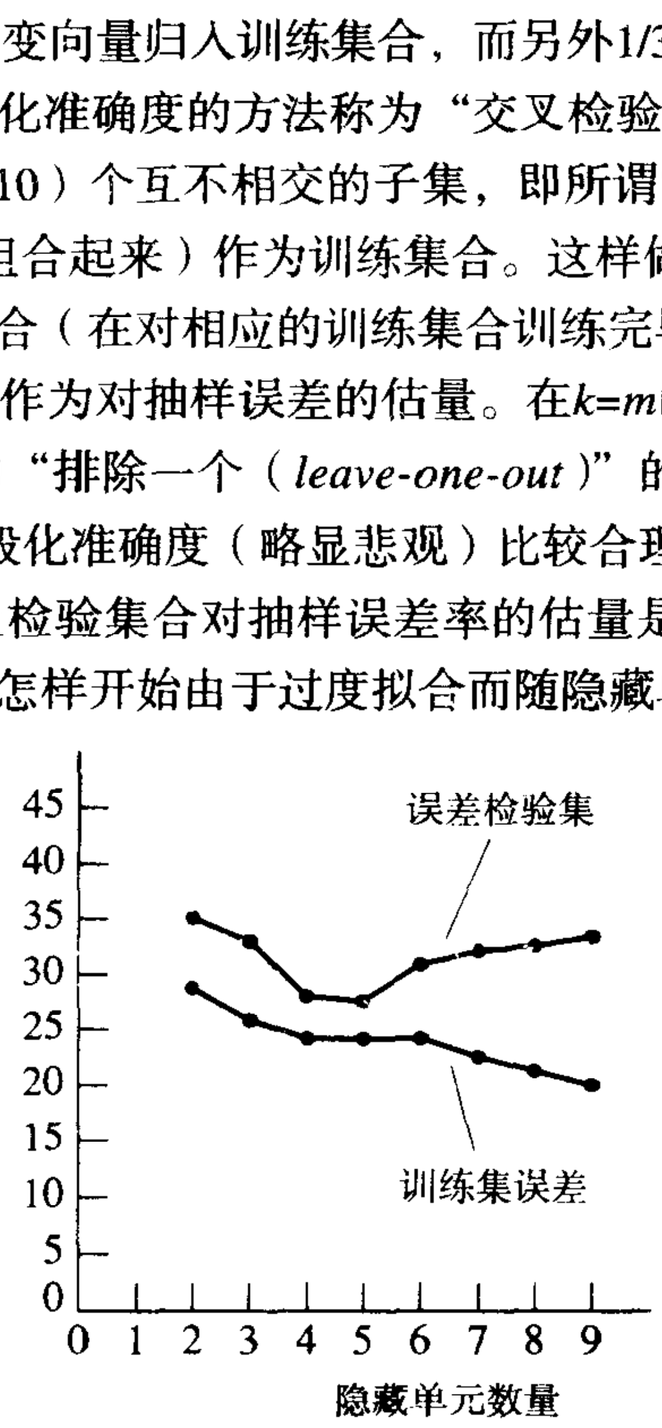


图3-9　一般化误差估计与隐藏单元数量（从[Duda,Hart, & Stork 1998]中改编）

3.5　补充读物和讨论

神经网络已经运用于解决模式识别、自动控制和大脑功能模型化的问题中，其中典型的例子有手写（ZIP译码）识别[LeCun, et al. 1989]、语音识别[Waibel, et al. 1988]和学习阅读手写文本中的语句[Sejnowski & Rosenberg 1987]。但值得注意的是，为这些应用而对神经网络的设计和训练仍是一门极需经验和实验的艺术。一些最佳的神经网络的研究成果和应用展示于年度“神经处理系统会议（NIPS）”，此会议发表的会议论文集名为“神经信息处理系统的进展(*Advances in Neural Information Processing Systems*)”。

前面提到，一个神经网络最基本的单元是TLU，它用一个超平面把输入向量的空间分开。

若两个训练输入子集的凸包（*convex hull*）不相交，那么超平面即可完成分割。可从本章所述的训练过程中或运用线性编程方法（众所周知，这种方法具有多项式复杂度）来得到一个分割超平面[Karmarkar 1984]。

我只介绍了分层前向网络。有关递归网络的行为的分析更加复杂，[Hertz, Krogh, & Palmer 1991]一书运用与物理学动态系统的类比对此作出了清晰的解释。反向传播这一算法已被（[Pineda 1987, Almeida 1987, Rohwer & Forrest 1987]）一般化，从而运用到递归网络中（当这样的网络汇集成稳定的状态时）（[Hertz, Krogh, & Palmer 1991, pp.172-176]介绍了这种算法）。

神经网络仅是许多机器学习结构的一种，另一种为“决策树”——它之所以深得一些人的喜爱是因为由它执行的函数（如DNF布尔函数）比神经网络中的函数更易理解。在人工智能领域中，Ross Quinlan首先提出决策树学习方法ID3[Quinlan 1979]和C4.5[Quinlan 1993]（统计学家们也已经开发出了相似的技术[Breiman, et al.1984]）。

本书的其他地方继续介绍其他学习技术，但在这里，我想提一下有关这一论题的信息资源。最重要的年度会议是国际机器学习会议（ICML），该会议发行会议论文集。计算学习理论进展研讨会（COLT）发表有关计算学习理论的论文。期刊主要有《Machine Learning》(机器学习)。重要的教材是[Mitchell, T. 1997, Langley 1996, Weiss & Kulikowski 1991]。有关神经网络的书有[Fu 1994, Haykin 1994, Hertz, Krogh, & Palmer 1991]。[Shavlik & Dietterich 1990]是一本论文集[Dietterich 1990]是对机器学习领域的一次出色的综述。

习题

3.1 一个具有权向量**W**和阈值θ的TLU完成一个超平面边界。推导一个表示此超平面与原点之间的欧几里德距离的表达式。从一个任意点**X**开始（参照图3-1）。

3.2 下面这个训练集合是线性可分的。

输入	输出
1 0 0	1
0 1 1	0
1 1 0	1
1 1 1	0
0 0 1	0
1 0 1	1

（手工）训练此训练集合中的线性阈值单元。你的单元包括执行阈值的输入在内的四个输入。设所有权值的初始值为0。用固定递增纠错程序来训练你的单元直至找到一个解。在每次训练循环后标出各组权值。以前面的输入为顶点画出一个三维立方体的草图，并根据最终权集画出分割平面的草图。

3.3 用一个TLU对一组n维输入向量归类。但假设，由于技术原因只能用非负权值。你打算如何完成这一归类?

3.4 设计（无需训练）一个前向网络来执行一个由两个输入组成的“或”函数。你的网络具备：（1）一个由接收输入x_1、x_2的线性阈值单元组成的隐藏层面。（2）一个把隐藏层的输出作为输入（但不是从x_1、x_2来的输入）的最终输出单元。

3.5 证明：用仅由一个隐藏层面组成的阈值单元网络便可实现任一由n个输入构成的布尔函数。

3.6 考虑下面图中的非线性单元：

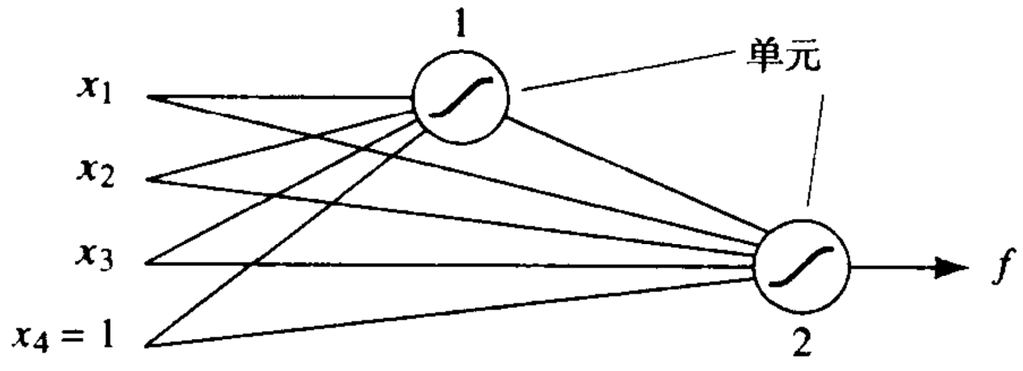
如果 $\mathbf{X}\cdot\mathbf{W}<-b$，$f(\mathbf{X})=0$

如果 $\mathbf{X}\cdot\mathbf{W}<-b$，$f(\mathbf{X})=1$

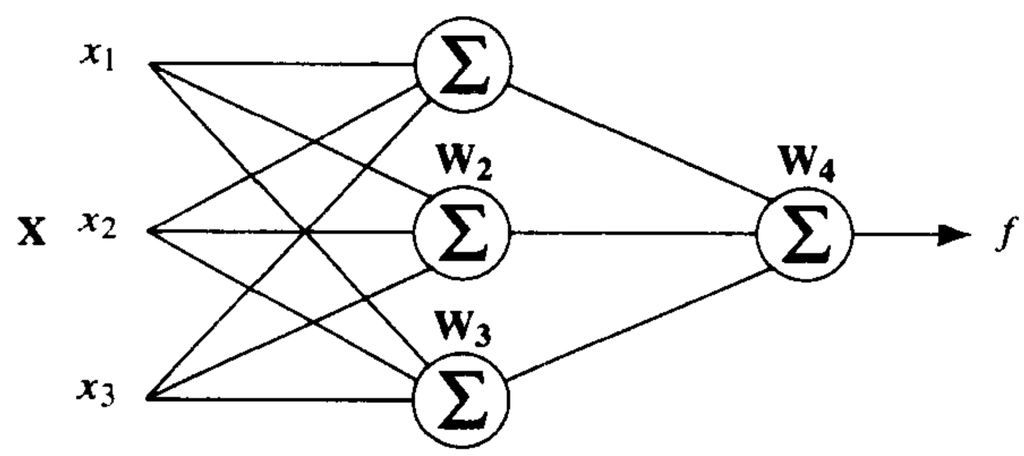
否则，$f(\mathbf{X})=(1/2b)(\mathbf{X}\cdot\mathbf{W}+b)$

与通常的阈值或sigmoid的非线性不同，我们现有一个如图所定义的“斜坡（ramp）”函数。通过在每个输入向量表达中，递增地应用下降最陡峭的程序，使平方差ε（即实际输出f与所希望的输出d之间的误差）最小，从而推导出权向量$\mathbf{W}$的权调节规则。评论一下你的结论。

3.7 考虑下图所示的“级联（cascade）”网络。它有三个输入，x_1、x_2和x_3。sigmoid单元1接收所有这些输入（包括一个“阈值输入”，$x_4\equiv 1$）。sigmoid单元2把sigmoid单元1的输出以及x_1、x_2、x_3和x_4作为其输入。所有sigmoid单元的输入均用可调节权值加权。基于网络输出与训练输入向量集合的标号之间的二次方差最小化，为该网络推导出一个适当实例化的反向传播式的增量调节程序。答案可用向量表达方式，即你无需具体到输入向量的分量x_i。



3.8 一个点积单元（DPU）计算一个输入向量$\mathbf{W}$和一个权向量$\mathbf{W}$的向量点积。它没有阈值；它的输出可简单地写为矩阵等式$\mathbf{WX}$，其中，$\mathbf{W}$是行向量而$\mathbf{X}$是列向量。考虑图中的DPU网络。证明整个网络等同于一个单DPU。这一DPU的权向量是什么？



3.9 1) 为一个sigmoid单元制定一个递增、梯度下降的权变化规则。在单元里，误差函数为$\varepsilon=|d-f|$，d为所希望的输出，$f=1/(1-e^{-s})$，$s=\mathbf{X}\cdot\mathbf{W}$，$\mathbf{X}$为一个$n+1$维输入向量，且$\mathbf{W}$为可被改变的$n+1$维权向量。

2) 解释一下你的规则与用二次方差作为标准的规则的不同之处。

3) 若采用这一新的误差标准，应对多层前向网络的反向传播规则做哪些改动？

第4章 机器进化

4.1 进化计算

上一章的学习系统，通过改变其行为来使其符合一组训练实例。这种学习模仿了生物系统学习的某些方面。生物系统采用另一种方法进行学习，即进化——子孙比祖先更先进。能运用与进化类似的过程来产生有用的程序么？本章将介绍一种达到此目的的技术。

生物进化过程中，先是双亲生产后代，然后这些后代中的一部分“有选择地生存”下来，并生产更多的后代。繁殖和选择性生存这两个方面足以生产繁殖能力越来越强的一代代个体。下面这个几何类比简单地描述了这一过程：想像这样一幅数学“风景画”：那里有山峰、山谷和平地，还居住着一群个体，它们随机地分布在这幅画的不同地方。个体在风景画中所处的高度是衡量此个体相对其他个体完成任务的能力。那些处于海拔低的个体停止“生存”，其概率随着其海拔越来越低而增大。那些处于海拔高的个体进行“繁殖”，其概率随着其海拔越来越高而增大。繁殖就是生产新的个体，使这些个体在风景画中的位置与其单亲或双亲的位置相关但不同。最有趣且有效的一种繁殖就是父亲和母亲联合生产新个体。这些子女在风景画中所处的位置是其双亲的位置的函数。在一些进化过程中，一个子女的位置（从某中意义上讲）处于其双亲的位置“之间”。

可把此过程视为搜索风景画的高峰过程。新生成的个体占据新地形，而那些处于海拔低的个体逐渐消失。不久，一些个体将位于峰顶。这个过程的有效度依个体不同于其单/双亲的方式和风景画的性质而定。子女所处的海拔高度也许会比它们的单/双亲低，但偶而也会更高。

把进化搜寻过程运用于计算机科学的主要目的有两个。函数优化是最直接的应用，在这种应用中，我们试图求出一个函数的最大值，如$f(x_1,\cdots,x_n)$，其中自变量$(x_1,\cdots,x_n)$说明个体的位置，f值为高度。John Holland[Holland 1975]提出了一种解决这类问题的算法——“遗传算法（GA）”。许多教材和论文均对此算法做了详细的介绍（请参阅 [Goldberg 1989，Mitchell，M.1996，Michalewicz 1992]）。

另一种应用就是用进化程序解决具体问题——如控制响应agent的程序。本书将对此进行介绍。“分类（*classifier*）系统”是GA的分支之一[Holland 1986]，且[Holland 1975，pp.171以后，第2版。已经成功地用于进化程序这一目的。而“遗传编程(*genetic programming*)（GP）”[Koza 1992，Koza 1994]是另外一种技术，它采用一种比GA更直接的方式来进化程序。在下一节我将举例说明GP过程。

4.2 遗传编程

4.2.1 遗传编程的程序表示

在遗传编程中，我们改进功能程序(*functional program*)，如LISP函数。这样的程序可以表

示为具有节点标记、有限的树。内部节点为带有一个或多个参数的函数、谓词或动作。叶节点为程序常数、或不带自变量的动作函数。在图4-1中，说明了如何用树结构来表示计算3+(5×4)/7的程序。本例中，叶节点为3、4、5和7，内部节点为函数+、×和/。

这里，我将介绍如何运用遗传编程来进化一个沿墙运动的机器人，这个机器人的任务与第2章所述相同。设此机器人所处的世界为二维网格世界，如图4-2所示。我们进化一个程序，此程序把机器人当前的传感器数据作为输入，并计算出一个动作。我们希望重复运行此程序来控制机器人，并把机器人从任一初始位置移到与墙毗邻的一个单元中，使其永远沿墙运动。

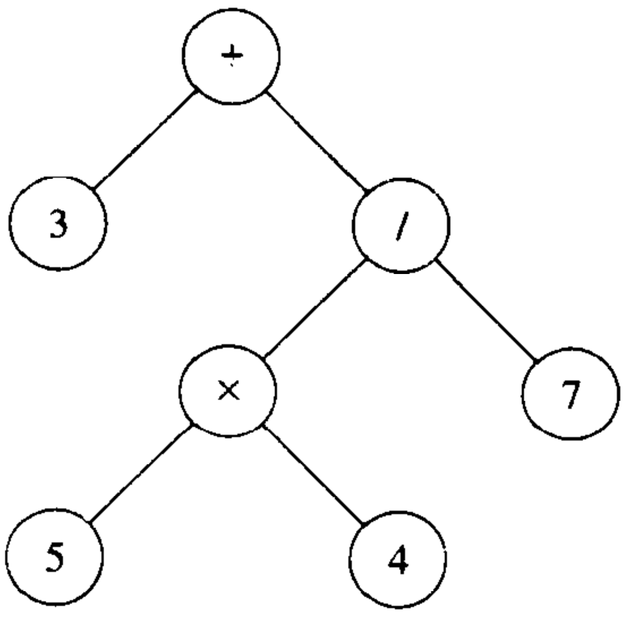


图4-1　用树结构表示的一个程序

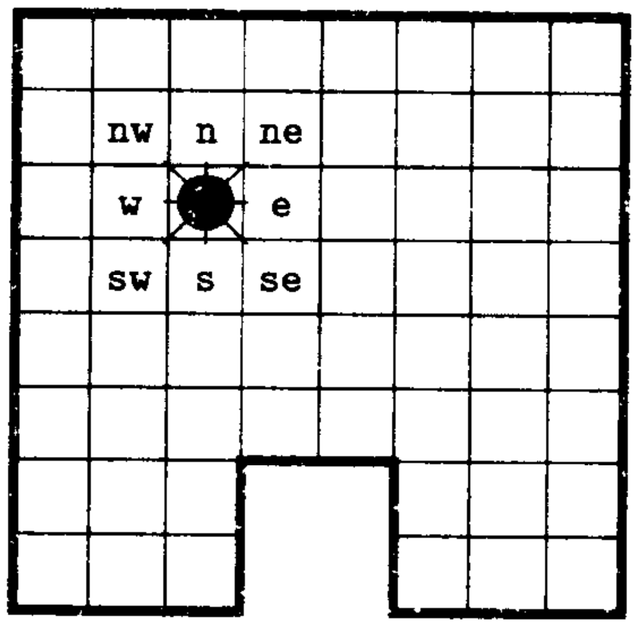


图4-2　网格世界里的一个机器人

此程序的基本函数包括：四个布尔函数——AND、OR、NOT和IF；四个动作——north、east、south和west。布尔函数具有它们通常的定义：

1) 当x=0时，AND(x,y)=0；否则为y。

2) 当x=1时，OR(x,y)=1；否则为y。

3) 当x=1时，NOT(x)=0；否则为1。

4) 当x=1时，IF(x,y,z)=y；否则为z。

四个动作定义如前：

north：将机器人在网格中向上移动一个单元。

east：将机器人在网格中向右移动一个单元。

south：将机器人在网格中向下移动一个单元。

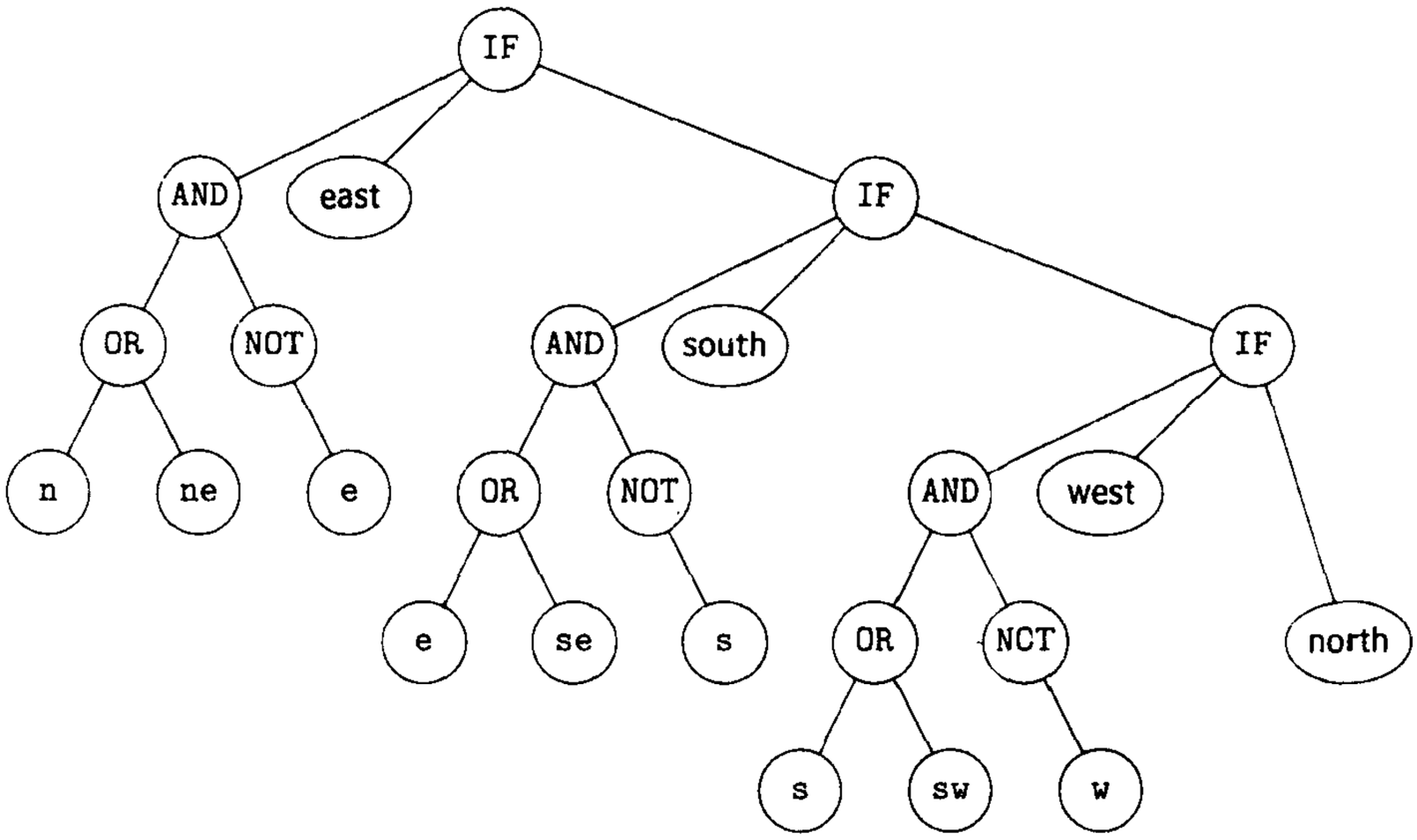
west：将机器人在网格中向左移动一个单元。

这些动作函数本身都有效果但无数值；因为对它们任意一个赋值便会终止程序，所以无须沿树结构向上传值。所有这些动作函数均有其指示的效果，当机器人企图移入墙里时，该动作无效，只能终止程序。当然，重复运行无效的程序最终也无效。

运用与以前一样的传感器输入，但在这里，为了便于记忆，我不用$s_1,\cdots,s_8$这种表示方式，而运用n、ne、e、se、s、sw、w和nw来表示。当相应的单元空缺使得机器人可移入时，输入值为0；否则为1。图4-2展示了将被感知的、与机器人相关的单元的位置。

在遗传编程中，必须保证程序中所有的表达式和子表达式（除了用来终止程序的表达式）对所有可能的参数来说均有值。例如，用函数*Divides*(*x*,*y*)来表示*x*/*y*的值。当*y*=0时，也应给*x*某一个值（也许是0）。这样，可保证树结构中的每个函数均有适当数量的参数，从而使此树结构描述一个可执行的程序。不久我们便会明白这一点的重要性。

在用遗传编程来进化一个沿墙运动的程序之前，图4-3给出了一个沿墙运动程序，同时也列出了此程序的树状表示和表结构表示。重复运行此程序将使机器人向北走到墙边，然后顺时针沿墙运动。这个程序可与第2章中为沿边界运动开发的产生式系统相比较。




```
(IF (AND (OR (n) (ne)) (NOT (e)))
    (east)
    (IF (AND (OR (e) (se)) (NOT (s)))
        (south)
        (IF (AND (OR (s) (sw)) (NOT (w)))
            (west)
            (north))))
```

图4-3 一个沿墙运动的程序

4.2.2 遗传编程过程

在遗传编程中，我们从随机程序开始，并运用那些能使程序对感兴趣的问题域有效的函数、常数和传感器输入。这些初始程序构成0代(*generation* 0)。0代程序的大小是遗传编程中的一个参数。在对遗传编程的阐述中，我们将从5000个随机程序开始来进化一个沿墙运动的机器人。从基本函数AND、OF、NOT和IF，动作north、east、south和west，传感器函数n、ne、e、se、s、sw、w和nw以及常数0和1中产生这些初始随机程序。每一代程序均接受评估，而且直到有一个程序的表现十分令人满意时才产生新的一代程序。

依据一个程序如何完成所设定的任务来对它进行评估。这里，我们把一个程序运行60次，并计算在这60次运行中被访问的与墙毗邻的单元个数（共有32个单元与墙毗邻，因此，从未走近墙边的程序的计数为0，而理想的程序的计数为32）。然后，我们让机器人分别在10个随机选择的不同初始位置进行前面这个步骤。在这10个运行中被访问的与墙毗邻的单元的总数即为此程序的“合理度（*fitness*）”。可能的合理度的最高值为320——机器人只有在这10个步骤的每一步的60次运行中均访问了所有与墙毗邻的单元，才可能达到这一最高值。

第*i*代按照下列方式建构第*i*+1代：

1) 把第*i*代的500个程序（10%）直接复制到第*i*+1代。用下列“锦标赛选拔（*Tournament Selection*）”过程来选择这500个程序：从5000个程序中随机选出7个，再从中选择最合适的那一个（“锦标赛选拔”是选择合适个体的一种有效的方法。数字7和直接复制的程序的百分比是遗传编程过程中的附加参数）。

2) 把4500个新的“孩子(*child*)”程序（90%）归入第*i*+1代。每一个孩子程序通过以下“交叉”操作由一个母亲(*mother*)程序和一个父亲(*father*)程序共同生产出来：这些双亲程序均从第*i*代的锦标赛选拔中选出，把母亲程序中一个随机选择的子树替换为父亲程序中的一个随机选择的子树，便产生出孩子程序（以下要求保证孩子程序的可执行度：即此程序的所有函数对所有参数的可能值来说均可执行）。在图4-4中，通过图解说明了这一交叉操作。双亲程序中的阴影节点即为随机选出的交叉点。这个孩子程序的合适度可能比其双亲高，也可能比其双亲低。因为一个主程序以及其合适的双亲的子表达式结合到了孩子程序中，所以我们认为此交叉操作可能有效。

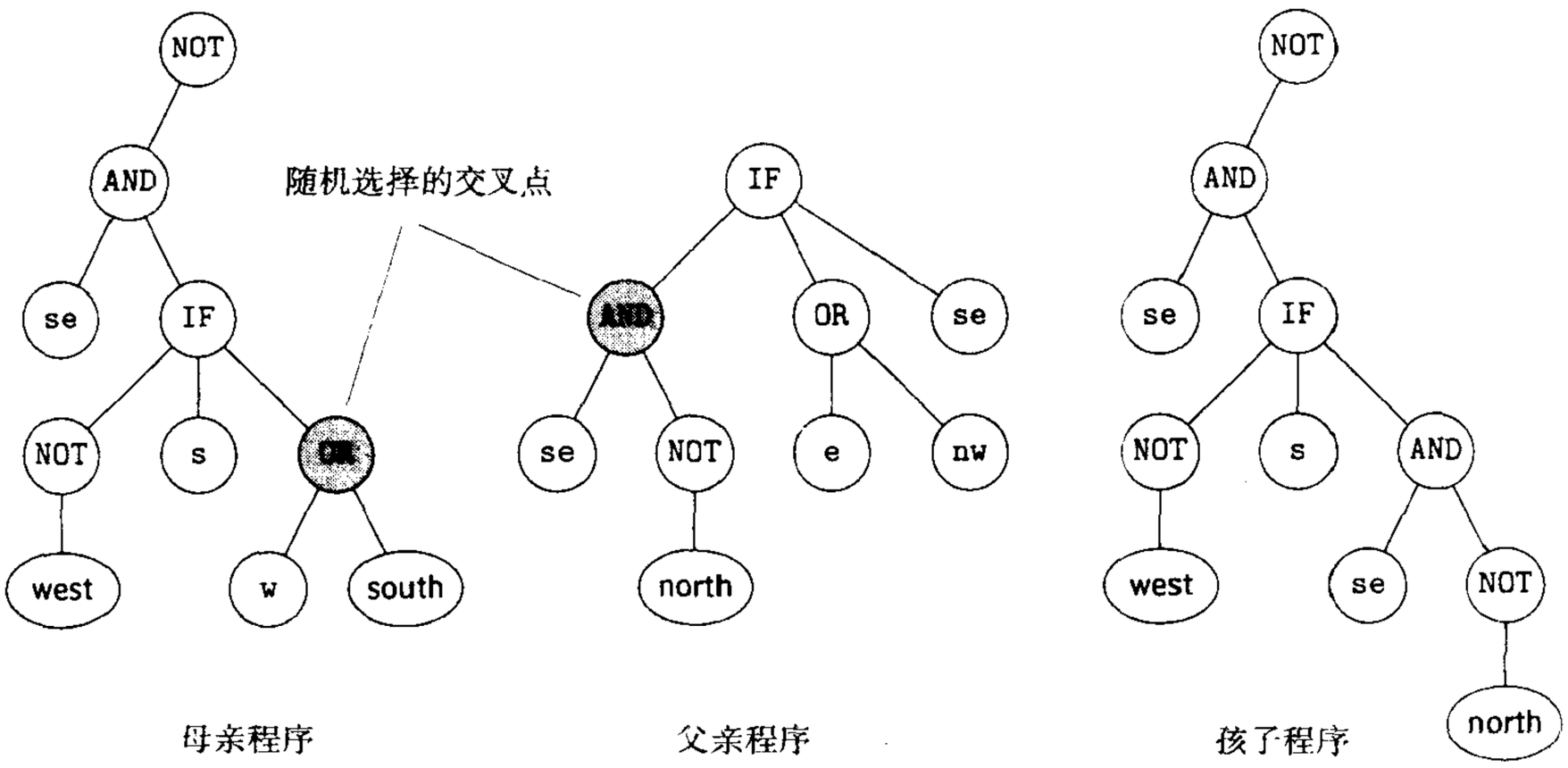


图4-4　两个双亲程序及其孩子程序

3) 有时，还用一个“变异算子(*mutation operator*)”来构造下一代个体，但它构造的下一代数目很少（也许只有1%）。这个变异算子用锦标赛选拔从*i*代中选出一个单亲，删除此单亲程序中随机选择的子树并替换为一个新成长的随机子树（此子树以0代个体生成的方式生成）。本例未用到变异。

我们注意到，在构造下一代的过程中，要设置几个任意参数，包括直接复制的数目、交叉生产数目、参与锦标赛选拔的程序数目以及变异百分比。示例中所用参数是遗传编程专家推荐的参数。

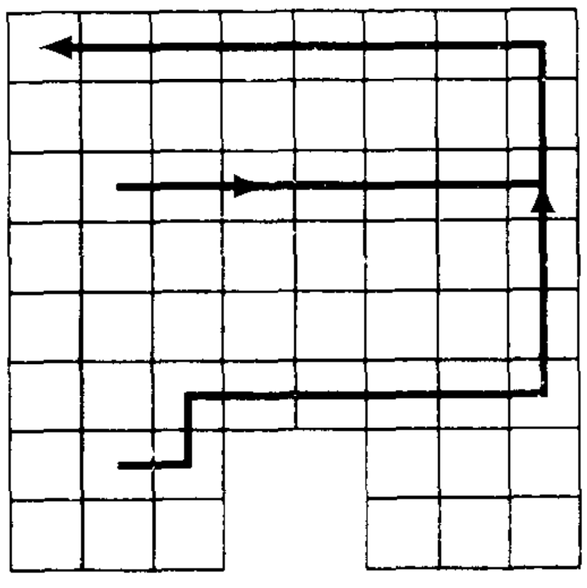
4.2.3　进化一个沿墙运动的机器人

从5000个随机程序开始，并运用刚才描述的技术，来开始进化一个沿墙运动的机器人的遗传编程[⊖]。许多0代随机程序根本什么也不会：如，(AND(sw)(ne))(合适度为0)给它的第一个参数赋值，若结果为0，则终止运行；否则，继续给第二个参数赋值，再终止运行。一些程

⊖　感谢David Andrew 为此实例编程。

序不管输入如何，只往一个方向运动。如，当程序（OR(e)(west)）赋值“west”，它朝西运动，然后终止运行。这个程序的合适度为5（一些随机程序会恰巧运动到与墙毗邻的单元中）。0代的这5000个程序可视为利用解决这类特殊进化问题资源来进行盲目空间搜索的计算机程序。

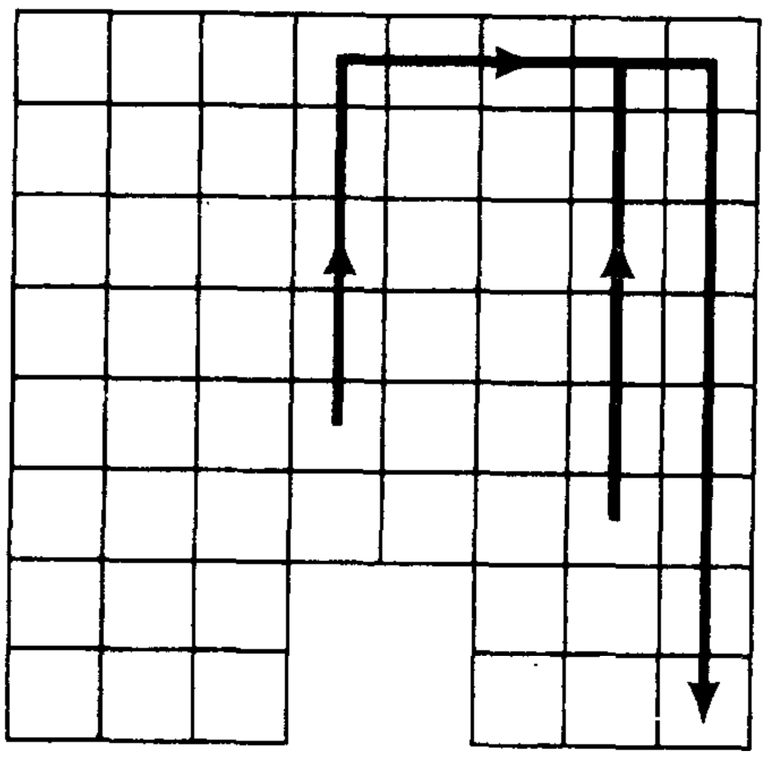
在图4-5中，给出了0代的最合适的程序（合适度为92）的表结构形式以及它的两个合适度的运行。和通常的遗传编程程序一样，这个程序很难理解而且存在许多冗余的操作（其中一些可被后处理器删除）。这个程序从任意单元开始东移直至到达与墙毗邻的单元；然后北移直至能再次东移，或者西移直至被困于左上角的单元中。



```
(AND (NOT (NOT (IF (IF (NOT (nw))
                       (IF (e)(north) (east))
                       (IF (west)(0) (south))
                   (OR (IF (nw)(ne)(w))
                       (NOT (sw)))
                   (NOT (NOT (north))))))
     (IF (OR (NOT (AND (IF (sw)(north)(ne))
                       (AND (south)(1))))
             (OR (OR (NOT (s))
                     (OR (e)(e)))
                 (AND (IF (west)(ne)(se))
                      (IF (1) (e)(e)))))
         (OR (NOT (AND (NOT (ne))(IF (east)(s)(n))))
             (OR (NOT (IF (nw)(east)(s)))
                 (AND (IF (w)(sw)(1))
                      (OR (sw)(nw)))))
         (OR (NOT (IF (OR (n)(w))
                      (OR (0)(se))
                      (OR (1)(east))))
             (OR (AND (OR (1)(ne))
                      (AND (nw)(east)))
                 (IF (NOT (west))
                     (AND (west)(east))
                     (IF (1)(north)(w))))))))
```

图4-5 0代的最合适个体

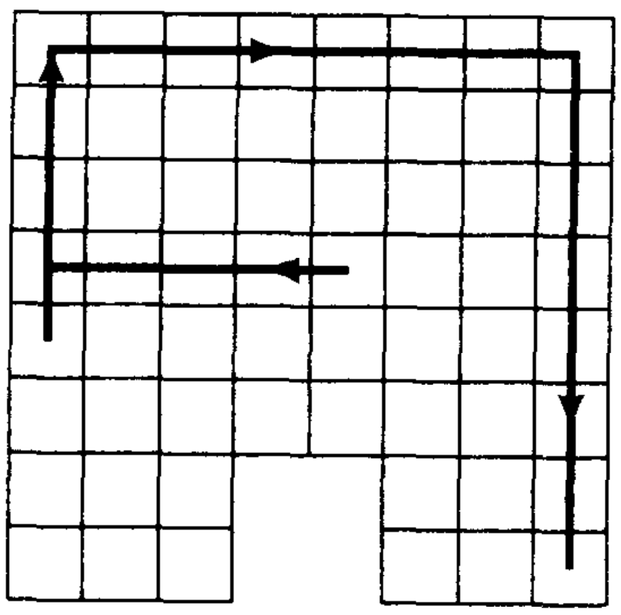
第2代的最佳程序的合适度为117。图4-6展示了这个程序及其两次典型的合适度运行的表现。这个程序比0代最佳程序短，但仍将被困于右下角。



```
(NOT (AND (IF (ne)
              (IF (se)(south)(east))
              (north))
          (NOT (NOT (e)))))
```

图4-6 第2代的最合适个体

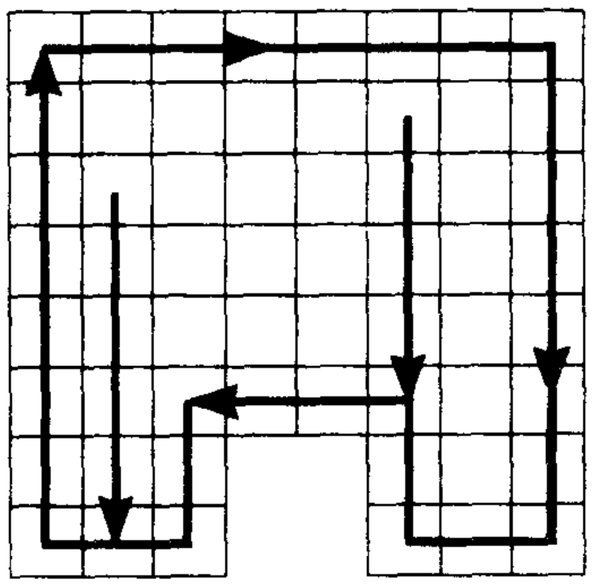
到第6代，程序的最高合适度达到163。其中最佳程序可理想地沿边界运动，但仍会被困于右下角，如图4-7所示。



```
(IF (AND (NOT (e))
         (IF (e)(s)(nw)))
    (OR (IF (1)(e)(south))
        (IF (north)(east)(nw)))
    (IF (OR (AND (0)(north))
            (AND (e)(IF (e)
                        (IF (se)(south)(east))
                        (north))))
        (AND (e)
             (NOT (IF (s)(sw)(e))))
        (OR (OR (AND (nw)(east))
                (west))
            (nw))))
```

图4-7　第6代的最合适个体

最后到第10代时，遗传编程过程已进化出一个能十分完美地沿墙运动的程序。图4-8展示了此程序以及从不同起点开始的两条路径。此程序顺时针沿墙运动，并且当它一开始不与墙毗邻时就南移并朝墙运动。



```
(IF (IF (IF (se)(0)(ne))
        (OR (se)(east))
        (IF (OR (AND (e)(0))
                (sw))
            (OR (sw)(0))
            (AND (NOT (NOT (AND (s)(se))))
                 (se))))
    (IF (w)
        (OR (north)
            (NOT (NOT (s))))
        (west))
    (NOT (NOT (NOT (AND (IF (NOT (south))
                            (se)
                            (w))
                        (NOT (n)))))))
```

图4-8 第10代的最合适个体

图4-9为每一代的最合适个体的合适度曲线。注意一代代逐渐的进步（通常进步不大）。

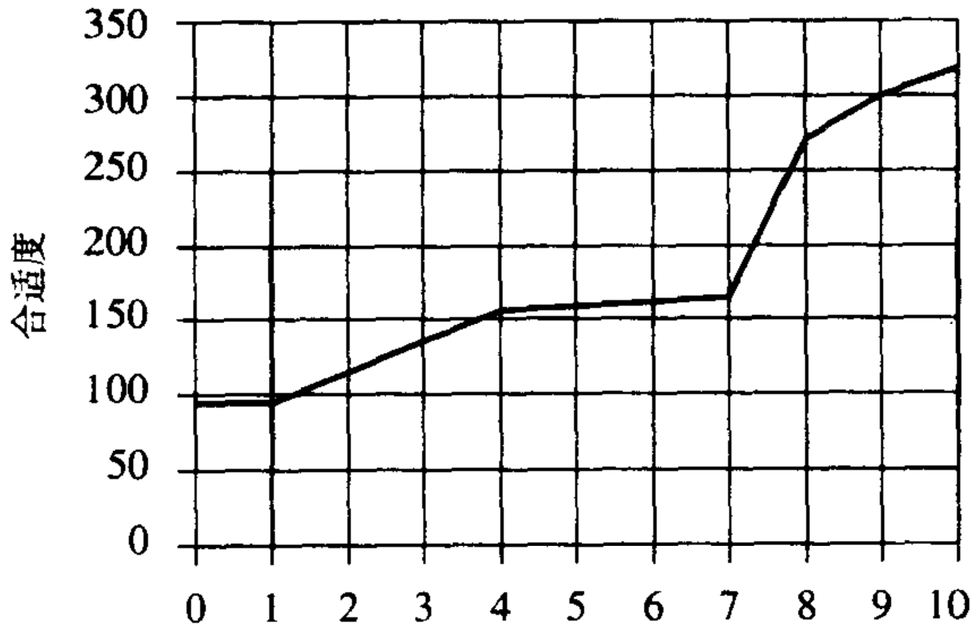


图4-9 合适度作为生产代数（*generation number*）的函数

4.3 补充读物和讨论

遗传算法和遗传编程已被广泛地应用于各种领域。遗传编程已经成功地用来进化响应agent，这些agent与其他人工智能研究者所研究的响应agent相似，其中包括推箱子、跟踪蚂蚁

路线、为载重拖车倒车及平衡反向钟摆等响应agent。[Koza 1992]描述并演示了这些应用以及其他技术的广泛应用。当我们扩充此方法使其允许进化子路线，继而借此来把主要程序当成基本函数时进行进化，这种方法会变得更加强壮。[Koza 1994]阐述了这个特征及其应用。最复杂且成功的遗传编程应用应是合成电子滤波器、放大器以及其他电路装置[Koza, et al. 1996]。已经有人做过用遗传编程来进化产生及运用储存器的程序、搜寻程序和递归程序等等这些基础性研究[Andrew 1995, Teller 1994, Brave 1996]。

研究遗传编程和遗传算法的论文出现于遗传编程会议的会议论文集、《IEEE Transactions on Evolutionary Computation》以及国际遗传算法会议的会议论文集中。

习题

4.1 指定用于进化下列agent的合适度函数

1) 控制一个电梯的agent

2) 控制一个城市主街道上的停车灯的agent

4.2 确定（生物学）进化论中“基因型（*genotype*）”与“表现型(*phenotype*)”这两个词的意义。如何运用它们来描述遗传编程?

4.3 若不严格要求“每个子树运行后必须返回一个值”，遗传编程交叉过程将会有何改变?

4.4 遗传编程中的交叉操作是从双亲的程序中各自随机选一个子树。谈谈你对随机选择出现以下偏见时所产生的影响的看法：

1) 倾向于选择在合适度测试中表现相当积极的子树；

2) 倾向于选择大的子树而不是小的子树，然后反之。

4.5 怎样用进化过程，如遗传编程过程，来进化

1) 神经网络?

2) 产生式系统?

请详细描述这些进化过程，特别是其中的交叉操作。谈谈你是否允许在神经网络的进化过程中运用Lamarckian进化。

4.6 你为什么认为变异有助于或不利于采用交叉的进化过程?

第5章 状 态 机

5.1 用特征向量来表示环境

一个S-R agent所用的特征向量可视为对与agent有关的环境状态的表示。这个S-R agent可从该特征向量中计算出一个适合环境状态的动作。如前所述，这个agent的传感器的局限使特征向量不可能完全精确地表示环境状态 ——尤其是那些从即刻(*immediate*)传感器输入中进行计算的特征向量。然而，通过考虑传感器的历史因素，可以提高其表示的精确度，因为agent以前也许曾感知过这些即刻无法感知的环境状态的某些重要方面。假设在某一时间$t = 0$时，存在一个对环境的初始表示（如设计者可事先输入这个初始表示）。再假设，在任一个接下来的时间步骤$(t+1)$时对环境状态的表示是$(t+1)$时的传感器输入、前一时间t时对环境状态的表示和t时所采取的动作的函数。我们称用这种方法跟踪其环境状态的机器为状态机。除即刻传感器输入以外，状态机必须具备用来存储环境模型的存储器。

这个模型可采用多种形式。最简单的表示就是一个特征向量的表示——与S-R所用的一样。但此时，特征向量$\boldsymbol{X}_{t+1}$要依另一个特征向量$\boldsymbol{X}_t$而定。图5-1说明了这种状态机。

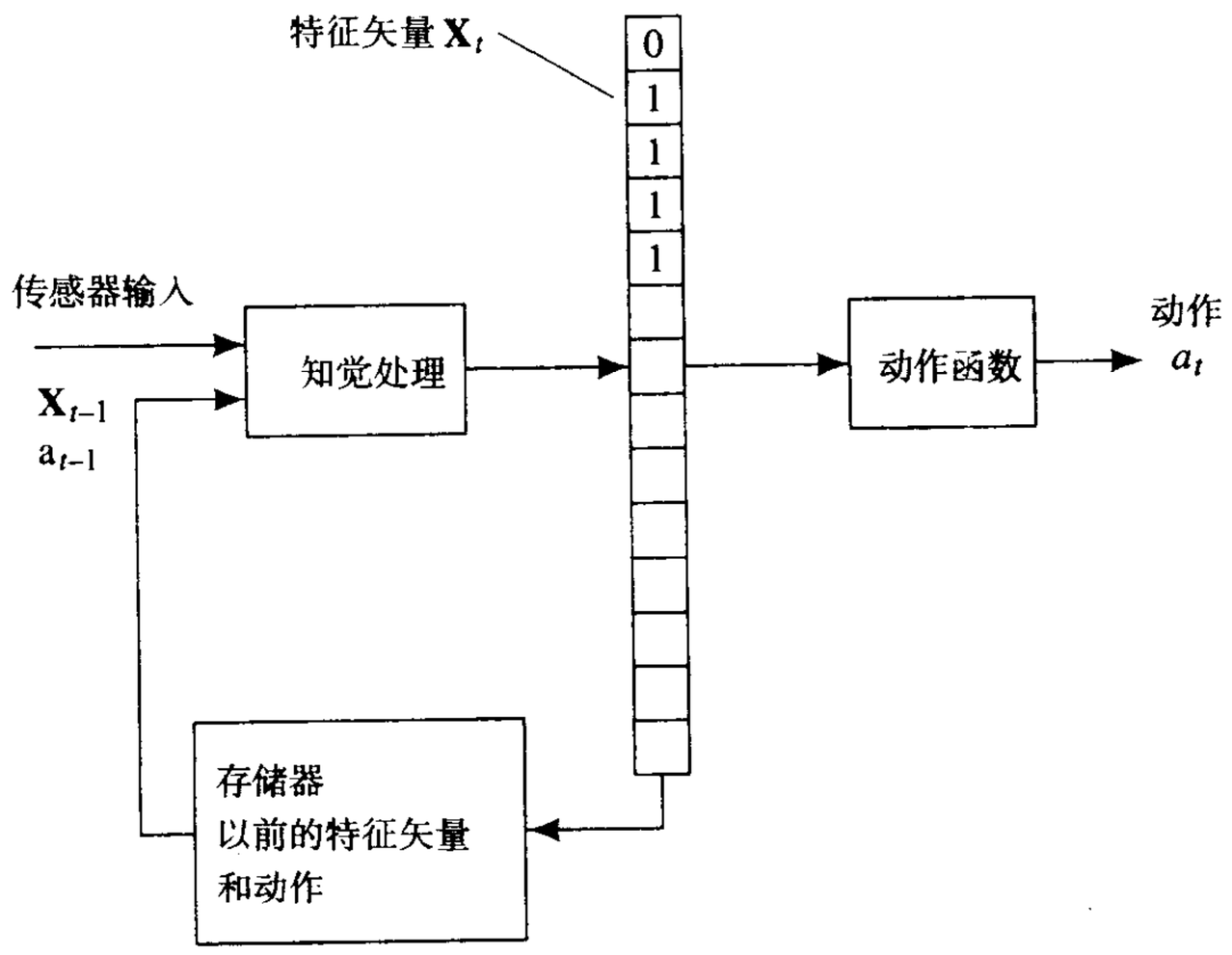


图5-1 一个状态机

由于agent 的环境可能存在的任意复杂性，一个特征向量总是无法完整地表示其环境。然而，为特定任务而设计的agent能够归并许多环境状态。因此，agent设计者应该设置一个能适当表示其环境状态——至少与agent的任务有关的环境状态的特征向量。

我们把第2章中沿边界运动的机器人作为一例状态机。回想一下，机器人通过从8个传感器输入中计算的特征值来构造其环境模型，这些传感器输入把围绕agent的单元是否空缺的信息传递给agent。让我们来看看沿边界运动的机器人的传感器受损时的情况。假设此机器人只能

感知与其毗邻的北、东、南、西方向的四个单元，这四个传感器输入为(s_2, s_4, s_6, s_8)，即它无法感知与其斜对角毗邻的另外四个单元。当北、东、南、西方向的四个单元都不空缺时，这些传感器输入的值均为1；否则为0。若机器人是从它的即刻传感器输入、前一个特征向量和刚完成的动作中计算所需特征向量，那么，即使存在上述缺陷，机器人仍能完成沿边界运动这一行为（再次假设环境中不存在“稠密空间”）。

为继续用一个特征向量来计算动作的实例，我们采用$w_i = s_i$，其中$i = 2, 4, 6, 8$，在其他四个无法感知的传感器输入的位置上换上w_1、w_3、w_5和w_7。当且仅当前一时间步骤中w_2值为1且机器人东移，w_1值为 1 。与此类似，当且仅当前一时间步骤中w_4值为1，且机器人北移，w_3值为 1 ，依此类推（否则，w_i值为0）。

下列产生式系统运用这些特征向量给出了沿墙运动的行为（在不存在“稠密空间”的世界中）：

$w_2\,\overline{w}_4 \rightarrow$ east

$w_4\,\overline{w}_6 \rightarrow$ south

$w_6\,\overline{w}_8 \rightarrow$ west

$w_8\,\overline{w}_2 \rightarrow$ north

$w_1 \rightarrow$ north

$w_3 \rightarrow$ east

$w_5 \rightarrow$ south

$w_7 \rightarrow$ west

1 → north

我们发现，第2章中的沿边界运动的机器人（其传感器未受损）不用存储前一次输入、特征和动作便能够完成任务。若机器人在任意所需时刻均能感知到环境的重要方面，那么就没有必要在存储器中存放一个环境模型。然而，传感器总有缺陷，故具有已存储的环境模型的agent通常能完成不具有存储器的机器人所不能完成的任务。

5.2　Elman网络

一个agent能用一种特殊的递归神经网络（被称为Elman网络[Elman 1990]）来学习如何从前一个特征向量和传感器输入中计算后一个特征向量和动作。这里将说明只能感知与其毗邻的北、东、南、西方向四个单元的沿墙运动的机器人是如何完成上述过程的。这个网络学会了储存那些适合于当前任务的前一次所感知的特征。图5-2显示了这样的一个Elman网络。该网络有8个隐藏单元，它们与前一节所讨论的机器人所需的特征一一对应⊖。网络的输入包括即刻传感器输入（s_2, s_4, s_6, s_8）以及前一时间步骤中8个隐藏单元的值。这8个附加输入称为“上下文单元(*context unit*)”，它们使网络将动作建立在已学会的前一次所感知的数据的特性之上——即机器人根据此上下文可找到自己即刻所处的位置。如前所述，有各种安排多网络输出的方法。这里，我们需要四个可区分的输出来与动作north、east、south、west一一对应。图5-2中的网络就有四个输出单元，分别代表四个动作。每个输出单元从同一组隐藏单元（特征）中计算动作。经过训练，我们可使一个单元所要求的输出为1，而其他单元所要求的输出为0。训

⊖　当然，在一个典型的学习情况中，我们并不知道需要多少隐藏单元（或多少个层面）。

练可通过带有相应合适动作标号的传感器输入序列来完成，这些动作类似于沿墙运动专家的动作。在实际操作中（训练后），我们选择具有最大输出的输出单元的动作。尽管Elman网络是第3章中提到的递归神经网络的一个特例，但仍可用普通反向传播来完成训练，这是因为反向权值（从隐藏单元到上下文单元）是固定的，而且上下文单元被看作是另一组输入。

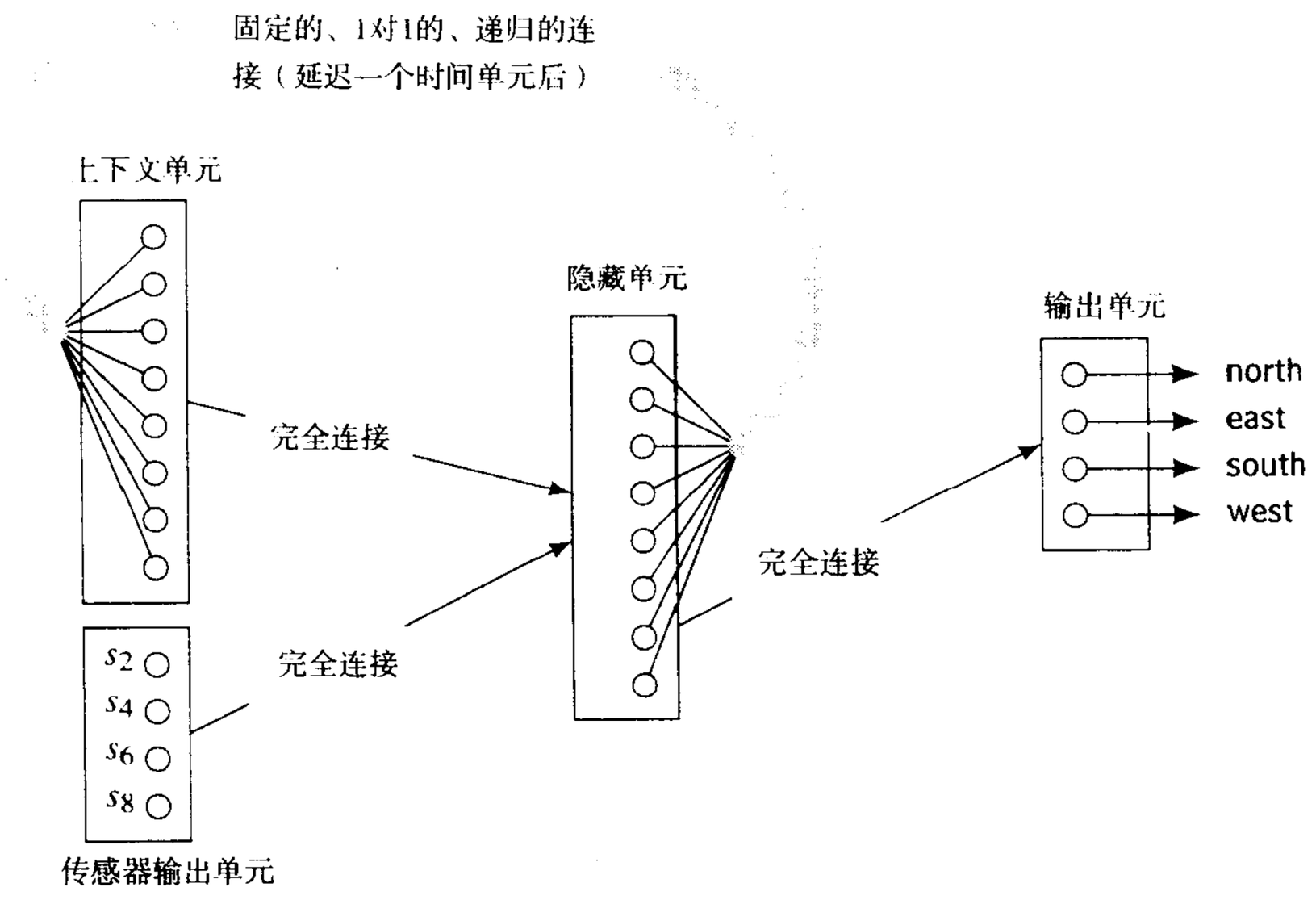


图5-2 一个Elman网络

5.3 图标表示

特征向量仅是表示环境的一种方法,我们还可以运用其他数据结构。比如，沿边界运动的机器人在感知时，可以用一个数组来存储一张由空缺单元和非空缺单元组成的映像。用特征或数据结构来表示世界的方法预示了人工智能的两大派别。其中的一派，我们称之为对世界“基于特征(*feature-based*)”的表示——即特征向量或属性向量；另一派我们称之为“图标（*iconic*）”表示——即数据结构，如地图，它被看作是对环境的重要方面的模拟（图标表示有时也称为“模拟表示(*analogical representation*)”。虽然我们很难泾渭分明地区分这两种表示，但在本书中，它们的确存在不同之处。

当一个agent运用图标表示时，它仍需计算适合于其任务及其当时（已模型化的）环境状态的动作。它对数据结构的响应方式与没有存储器的agent对传感器刺激的响应方式几乎一样：计算数据结构的特征。图5-3说明了这样组织agent的一种方法。首先用传感器输入将图标模型更新为恰当的模型，然后用与感知处理相似的操作来提取动作计算子系统所需的特征。这些动作包括改变图标模型以及影响实际环境的动作。从图标模型中得到的特征的表示环境的方式应适合于机器人所必须采取的动作的种类。

当一个处于网格空间的机器人的图标表示是一个由空缺单元和非空缺单元组成的矩阵时，此图标表示就可以像机器人的传感器一样，能感知网格中的单元是否空缺。比如，我们来讨论

这样一个机器人，它对局部环境的表示如图5-4所示。空缺单元相应的数组元素表示为0，非空缺单元表示为1，未知单元表示为？，机器人在周围的单元中所处的位置表示为*R*。一个机器人被设计为走到离其最近的墙边之后开始沿墙运动，它若处于图5-4中的数组所表示的环境状态中，将被激发动作“west”。当然，做完这个动作，机器人应更新其模型，从而改变自身的位置，并处理新的传感器数据。

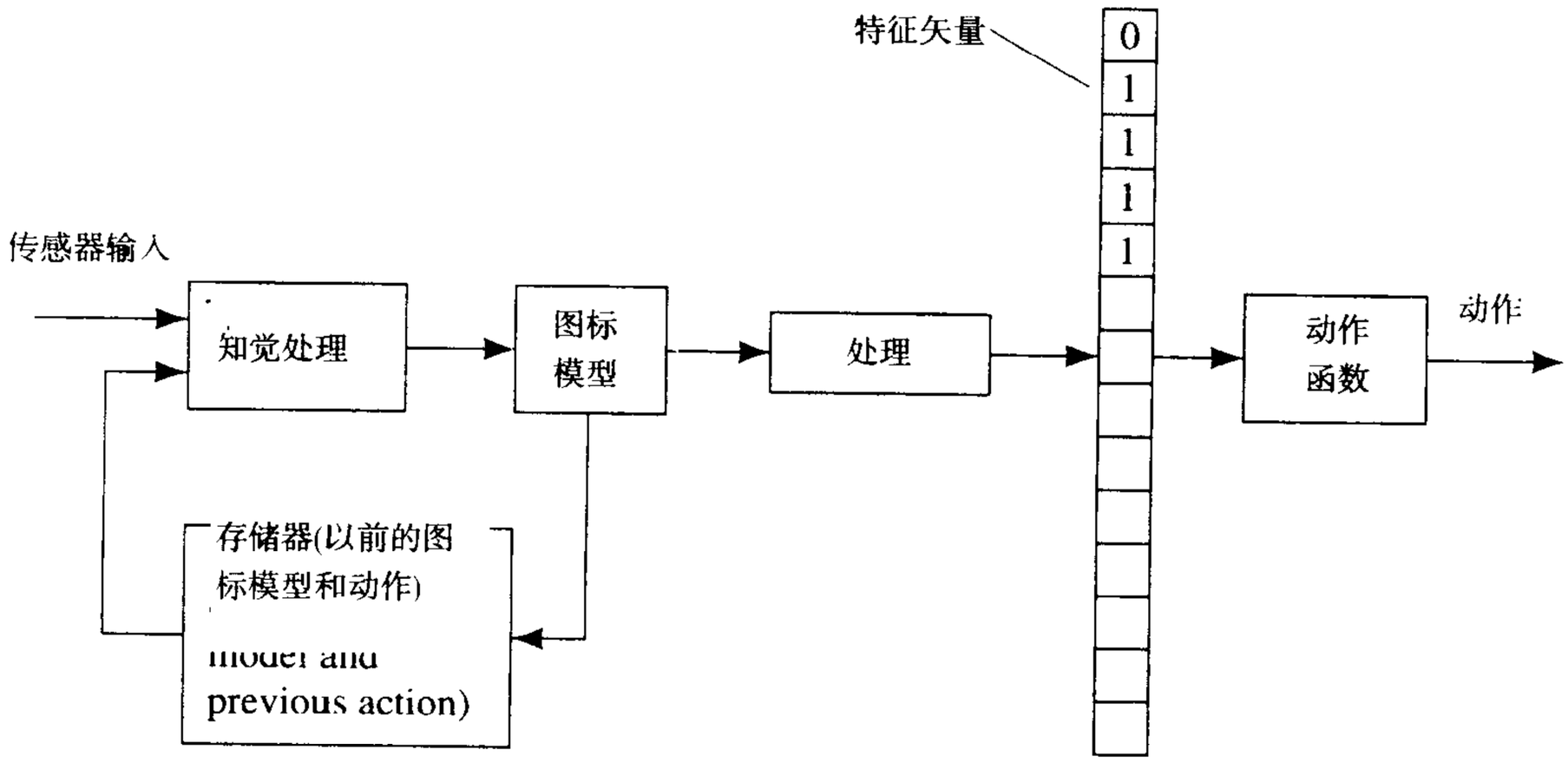


图5-3　从图标表示中计算特征的一个agent

另一种动作计算机制运用一种“人工势域（*Artificial potential field*）”模型。这种技术广泛地用来控制机器人的运动[Latombe 1991, 第7章]。它用一个二维势域来表示机器人的环境[⊖]。这个势域是一个吸引部分(*attractive component*)和一个排斥部分（*repulsive component*）的总和。吸引域与目标位置有关——即与机器人被命令到达的地方有关。典型的吸引函数为 $p_a(\mathbf{X})=k_1 d(\mathbf{X})^2$，其中$d(\mathbf{X})$是点$\mathbf{X}$到目标的欧几里德距离。在目标位置时，此函数的最小值为0。环境中的障碍激发一个排斥域。典型的排斥函数为 $p_r(\mathbf{X})=\frac{k_2}{d_0(\mathbf{X})^2}$，其中$d_0(\mathbf{X})$是点$\mathbf{X}$到障碍点的欧几里德距离。

	1	1	1	1	1	1	1	?
1	0	0	0	0	0	0	0	?
1	0	0	0	0	0	0	0	?
1	0	0	0	0	0	0	0	?
1	0	0	0	0	0	0	0	?
1	0	0	*R*	0	0	0	0	?
1	0	0	0	0	0	0	0	?
1	0	0	0	0	0	0	0	?
1	0	0	0	0	0	0	0	?
1	?	?	?	?	?	?	?	?
?	?	?	?	?	?	?	?	?

图5-4　一个类似地图的图标表示

总势为：$p = p_a + p_r$。机器人被引导沿着势域的梯度运动——即向下滑动。我们既可以事先计算势域并将其保存在存储器中，作为世界模型的一个方面；也可以在机器人决定移动之前将其所处位置的势域递增地计算出来。图5-5即是一例势域。在图5-5a中，机器人位于标记*R*处，目标位置在标记*G*处。图5-5b、图5-5c和图5-5d分别是吸引域、排斥域和总域。图5-5e是等位曲线和机器人的路线。这种方法不可避免地存在极小势，它将困住机器人。解决这个问题的许多方法已被研究出来 [Latombe 1991]。

⊖　当要控制许多自由度时，应采用多维域。

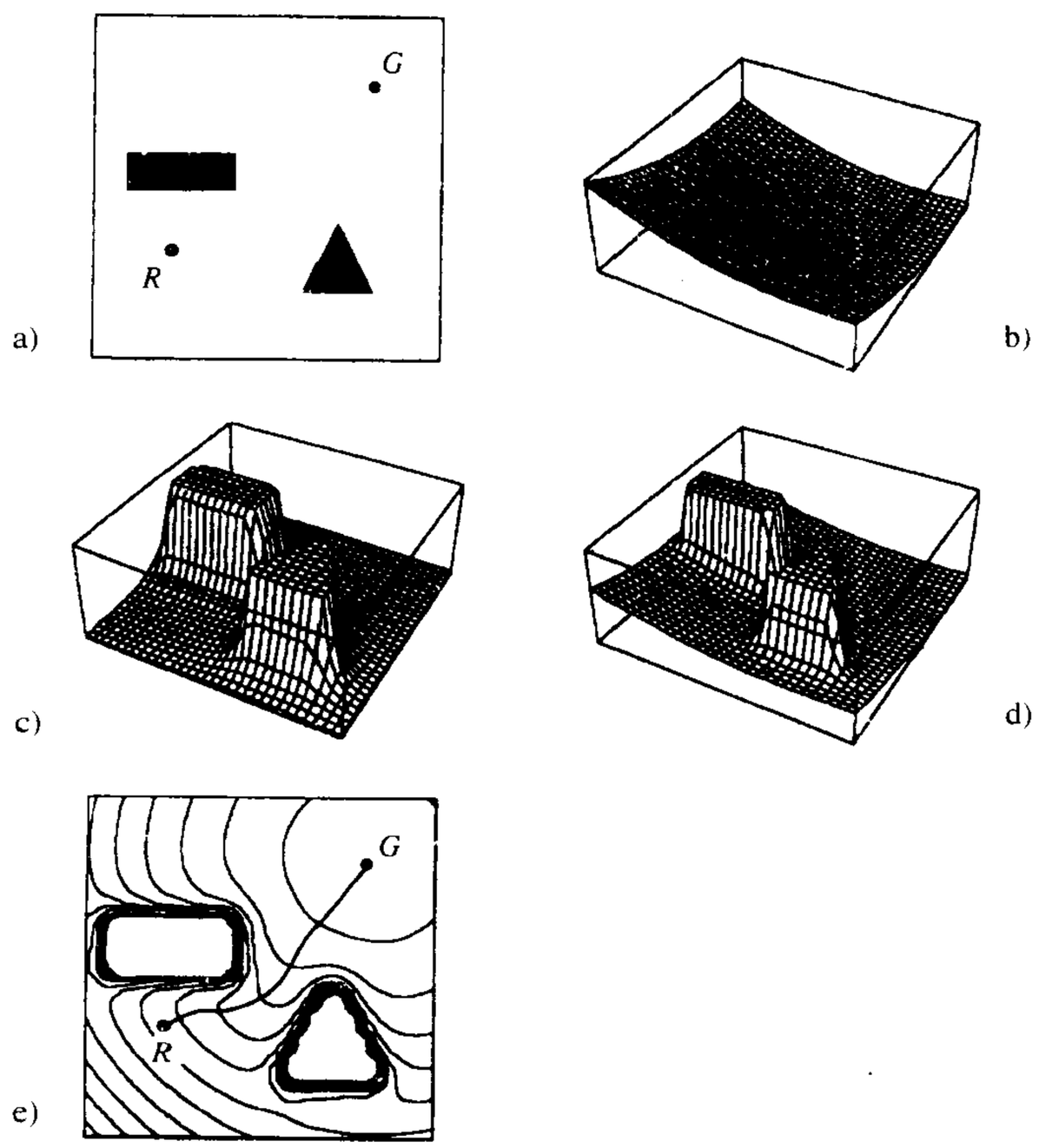


图5-5 一个人工势域的等位曲线（节选自[Latombe 1991]）

5.4 黑板系统

用来模型化世界的数据结构不必一定是图标的，尽管通常它们总是那样。一种重要的人工智能机器基于“黑板（*blackboard*）结构”[Hayes-Roth 1985, Nii 1986a, Nii 1986b]，这一结构运用一个被称为“黑板”的数据结构。一个被称为“知识源（*knowledge source*, KS）”的程序读取并修改这一黑板。黑板系统详细地阐述了前面所描述的产生式系统。每个知识源有条件部分和动作部分，条件部分计算一个特征的值，这个特征可以是任意有关黑板数据结构的条件，其值为1或0（也可以是“真”或“假”）。动作部分是任一改变数据结构或执行外部动作（或两者一起）的程序。当两个或两个以上的知识源的值为1时，“冲突解决（*conflict resolution*）”程序决策哪个知识源有效。知识源的动作不但能修改黑板，还能产生外部效果。另外，处理传感器数据的感知子系统也可以修改黑板。通常，我们通过分层来组织黑板数据结构，次级数据结构占据不同的层次。图5-6即是这样的一个黑板系统。

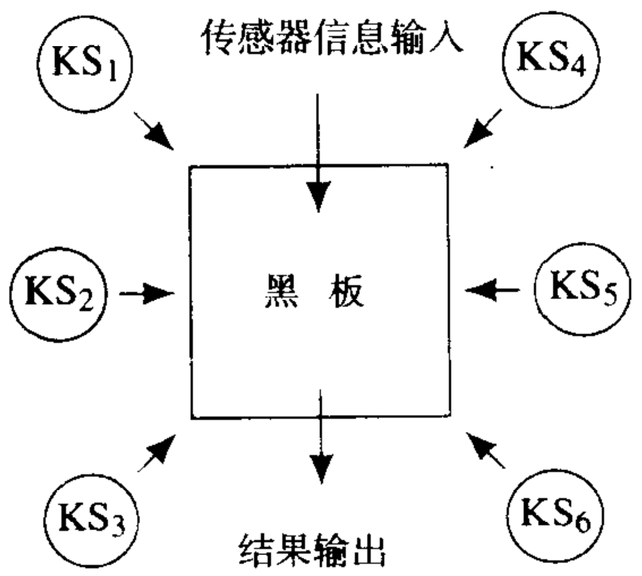


图5-6 一个黑板系统

知识源“精通”它们所管理的黑板的某一（或某几）部分的内容。当它们发现其管理的那一（或几）部分黑板有什么特别之处时，就会提出对黑板进行修改，而此修改如被选中，也可能激活其他知识源，依此类推。这样设计黑板，使得

当计算依次进行时，它最终变为这样一个数据结构：包含一个特殊问题的解或以所希望的方式改变世界的外部效果。这种黑板结构已运用于实际应用中，如语音理解[Erman, et al. 1980]、信号解释[Nii 1986b]和医学病人护理监控[Hayes-Roth, et al. 1992]。

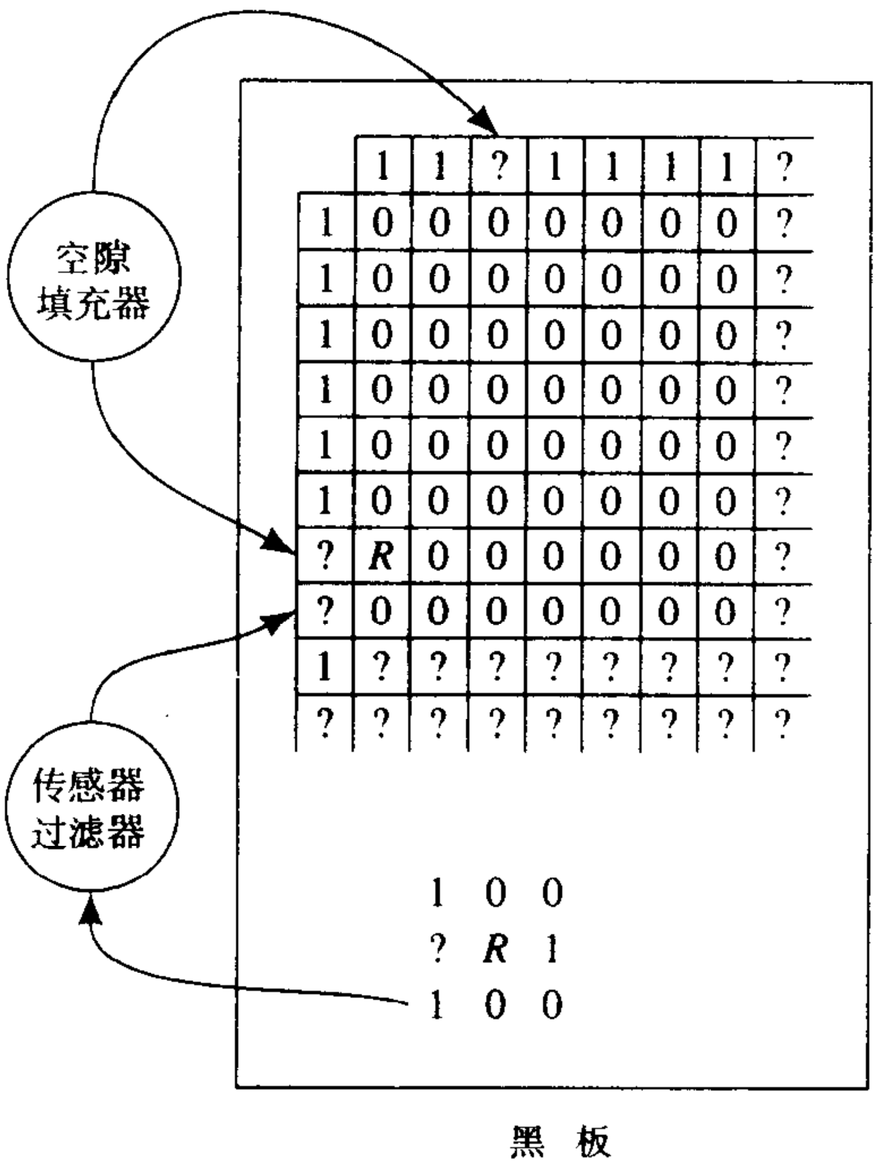


图5-7　一个机器人的黑板及其知识源

举个例子，我们再来讨论处于网格世界的机器人——这次，它能即刻感知所有围绕它的8个单元。然而，它的传感器（如同真实的传感器）有时会给出错误信息。这个机器人有一幅关于它所处世界的不完整的地图——如图5-4所示。由于传感器的错误，这一地图可能不完整并且存在错误。这样，表达地图的数据结构和包含传感器数据的数据结构组成了一个黑板。图5-7 是此机器人经历中某一时刻的黑板。

黑板有两个知识源来帮助机器人纠正错误。一个是“空隙填充器（*gap filler*）”，另一个是“传感器过滤器（*sensory filter*）”。前者在地图中寻找稠密空间（虽然已知不存在稠密空间），或者把1填入其中，或者扩展填充0（我想让你来推导填充和扩展的依据）。在图5-7中，空隙填充器决定在地图的顶部填充稠密空间。

传感器过滤器同时观察传感器数据和地图，并试图调和差异。图5-7中，传感器过滤器发现s_7的“已占据单元”的信号很强，而地图中相应的单元却是“？”。因此，它决定把地图中的“？”改为“1”来调和差异。相似地，根据地图的数据，它决定$s_4 = 1$的信号是错误的，因此并不按此信号来改变地图。然后，空隙填充器把左边一列中的那个“？”填充上“1”。

根据为机器人所设定的任务，其他一些知识源（对已调整过的地图作出响应）可向机器人建议一个动作。

5.5　补充读物和讨论

状态机比S-R agent应用更广泛。第2章所提到S-R agent与动物行为学模型的关系也适用于状态机。许多动物对那些无法即刻感知的内部世界和外部世界的特征至少会保存部分信息（参阅习题5.4）。对存储器和感知器同时作响应的产生式系统已被用来构造某些心理现象的模型。

Elman网络是学习有限状态机的一例。[Rivest & Schapire 1993]也曾研究过这个问题。

许多研究者[Kuipers, et al. 1993, Kuipers & Byun 1991]研究过如何学习空间地图的问题，这些地图均是图标表示的例子。当对即刻状态和动作效果不确定时，地图学习会更加困难。[Dean, Basye, & Kaelbling 1993]曾研究过这种情况。

[Genesereth & Nilsson 1987]把那些其行为依据过去的历史的agent命名为“*hysteretic agents*”。

习题

5.1 一个离散式电梯可以感知到它所处世界的以下信息：

1) 电梯应该停在哪一层。

2) 电梯里的乘客想要去哪几层。

3) 电梯外的哪几层有乘客想要乘电梯，他们想要往上乘还是往下乘。

4) 电梯门的状态（开或关）。

此电梯还能做以下动作：

1) 上升一层（除非已处顶层）。

2) 下降一层（除非已处底层）。

3) 开电梯门。

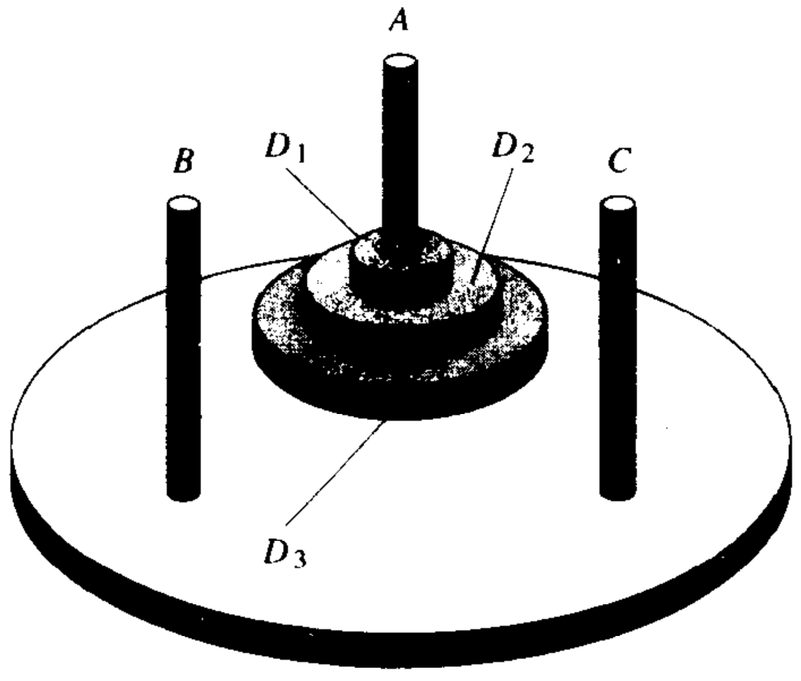
4) 关电梯门。

5) 等待△秒（△是使得电梯里的乘客出电梯、电梯外的乘客进电梯的充足时间）。

设计一个能有效控制电梯的产生式系统（如果电梯里仍有人想要去更高层或者电梯外的更高层仍有人想要乘电梯，该系统却改“上升”为“下降”，则此系统就不能有效控制电梯）。

5.2 “人工蚂蚁”生活在一个二维网格世界中，它能沿已作标记的单元所组成的连续“信息素踪迹(*pheromone trail*)”(宽为一个单元)的运动。这个蚂蚁占一个单元，它可以面向上、下、左、右。它能做五个动作：前移一个单元(m)；在同一单元中向左转(i)；在同一单元中向右转(r)；设置状态位元“开”(on)；设置状态位元“关”(off)。蚂蚁感知它的正前方(即其面朝的方向)是否有信息素踪迹且其状态位元是否为“开”（设状态位元起动为“关”）。说明可以控制这样一个跟踪以上路线的蚂蚁的一个产生式系统。设这个蚂蚁最初位于一个可感知的信息素踪迹单元中（请记住，产生式系统由一个有序的条件—动作规则集合组成；所执行的动作与首先满足的那个条件相应。每个规则的动作部分可以有一个以上的动作）。确保蚂蚁不会折反自己走过的路线。

5.3 “三盘汉诺塔问题”有三个桩*A*、*B*和*C*，三个中心有孔的圆盘D_3、D_2和D_1可以放在它们的上面。盘D_3比盘D_2大，盘D_2又比盘D_1大。通常，此问题把三个桩排成一排，但这里把它们放在一个圆周上。首先把三个圆盘放在*A*上，然后试图将这三个圆盘移到别的桩上（如下图）。其中的规则为：可把任一个圆盘移到任一个桩之上，但只能移动桩上位于顶部的圆盘，而且不能把大的圆盘放在小的圆盘之上。你也许熟悉解决这个问题的递归算法。



注释：有关更简单的算法，请参阅www.mkp.com/nils/clarified。

有趣的是，还存在另一种非递归算法，它的规则陈述如下：我们总是移动可以移动的最大的圆盘，但每次移动的方向不能破坏前一次的移动。我们尽可能在奇数次时顺时针移动，在偶数次时逆时针移动。当不可能遵守这个顺逆时针的规则时（因为不存在其移动不破坏前一移动的最大圆盘），我们就往不理想的方向移动，然后再重新执行原来的规则。

我们将说明一个执行这一算法的产生式系统。假设下列六个圆盘移动的动作：move(num, dir)，num可以是D_3、D_2和D_1，dir可以是CW（顺时针）或CCW（逆时针）。再假设传感器可以判断以下条件的真伪：B_1、B_2和B_3，这里B_i是指圆盘D_i为最大的可移动的圆盘。另外，以下“状态”条件将由我们的产生式系统来设定和删除：*CW*（指上一次移动为顺时针）；M_1、M_2和M_3，这里M_i指最后一次移动的圆盘是D_i；改变状态的动作有toggle——它改变*CW*状态（即，当执行动作toggle以后，*CW*从1变为0或者从0变为1）；moved(D)、moved(D_2)和moved(D_1)，这里moved(D_i)把M_i设置为真，M_j为假（j不等于i）。

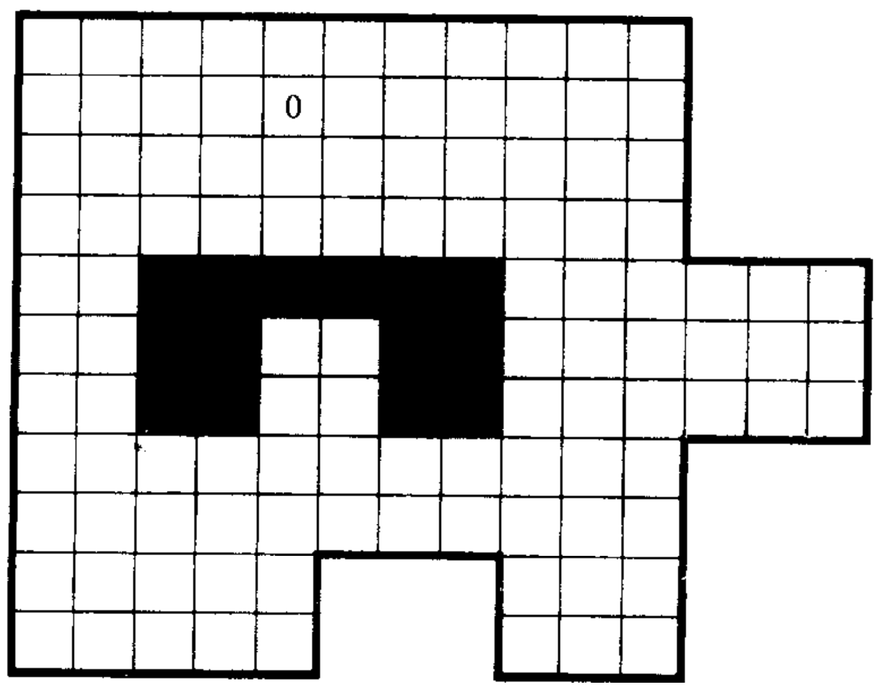
请用这些传感器和状态条件以及以上动作来说明一个运用此算法的产生式系统。别忘了指定状态条件的初始值！设这个产生式系统中的产生式规则已经排序，且所执行的动作与满足条件的最高规则相应。一个产生式规则的动作部分可以包含圆盘移动和状态改变的动作，且可同时执行这两种动作。

5.4 一个独处的母黄蜂Sphex把卵生在一只被它蛰晕并带回其巢穴的蟋蟀身上。母黄蜂孵出幼虫后，用此蟋蟀来给这些幼虫喂食。根据[Wooldridge 1968, p. 70]，黄蜂将做以下有趣的动作：

母黄蜂把蛰晕的蟋蟀带回巢穴放在穴口，进巢穴看看是否一切正常，然后回来把蟋蟀拉进巢穴。当黄蜂进穴做预备检查时，若此蟋蟀被移开了几英寸，黄蜂回来时会把它再移回穴口，然后再重复进穴检查是否一切正常。若黄蜂在穴中作检查时，蟋蟀又被移开了几英寸，黄蜂出来后会再次把它移回穴口，再进穴作最后检查……有一回，这个过程重复了40次，每次结果都一样。

请设计能使黄蜂这样做的特征、动作和一个产生式系统。

5.5 设计一个人工势函数(有吸引部分和排斥部分)，使其能指导机器人在二维网格世界中(如下图)从任一单元运动到标记为0的目标单元中(设此机器人能做的动作为向北、东、南、西方向移动)。吸引和排斥部分的和有局部最小值吗？若有，它在哪儿？为此网格世界设计一个特殊的全局势函数，它不具局部最小值，而且通过使用它能保证以最短路径到达目标。



第6章　机器人视觉

6.1　引言

迄今为止，所描述的S-R和状态机使用十分有限的传感器输入，它们只能提供网格世界中与其毗邻的单元的信息。实际上还存在许多其他表达agent所处世界的重要信息的传感器输入——音响、温度和压力，等等。传感器可用于各种机器以使其能对所处环境作出响应。

动物用眼睛一瞥，其视觉感官就能提供大量有关其所处世界的信息。赋予机器人“看”的功能正是“机器人视觉(*computer vision*)”这个学科所研究的问题之一。这一领域十分广阔，不仅包括通用技术，而且也包括为数众多的专用技术——如字符识别、相片解释、脸谱识别、指纹识别和机器人控制等等。

尽管对我们人类来说，“看”轻而易举，但这对机器来说却是一件非常困难的事。这其中的困难主要来源于难以控制的照明、影象和复杂而难以描述的物体，如那些室外场景中的物体、非刚性物体或啮合其他物体的物体。其中有些困难在人造环境中，如建筑物的室内景观，可得以减轻，而且在这种环境中研究计算机视觉往往更成功。这里限于篇幅，仅介绍机器人视觉的一些主要概念。

计算机视觉首先是在一组感光性原件上，如电视摄像机的光电管，生成一个场景的图象（对立体视觉需生成两个或两个以上的图象。本章的后面部分会介绍立体视觉）。这个图象是摄像机通过镜头对在视野中的场景进行一个透视投影，然后光电元件将其转换成一个二维的、随时间变化的亮度矩阵图象$I(x,y,t)$，其中x和y为光电元件在数组中的位置，t为时间（对有色视觉，需形成三个这样的矩阵来分别代表三原色。但我们在这里只考虑单色的情况，同时排除了可变时间——即假设一个静态场景）。一个由视觉引导的响应agent必须通过处理这个矩阵来产生这个场景的图标模型或者一组特征，从而使它能直接计算一个动作。

如图6-1所示，透视投影是多对一的变换。多个不同的场景可能生成相同的图象。更麻烦的是，图象易受到周围光线不足或其他因素的干扰，这样，我们就不能直接转换图象来重建场景。因此，agent通过运用可能处于有关场景中的物体的特定知识、有关场景中的各种表面的特性以及由这些表面反射回摄像机的周围照明度等一般知识来从图象中获取有用的信息。

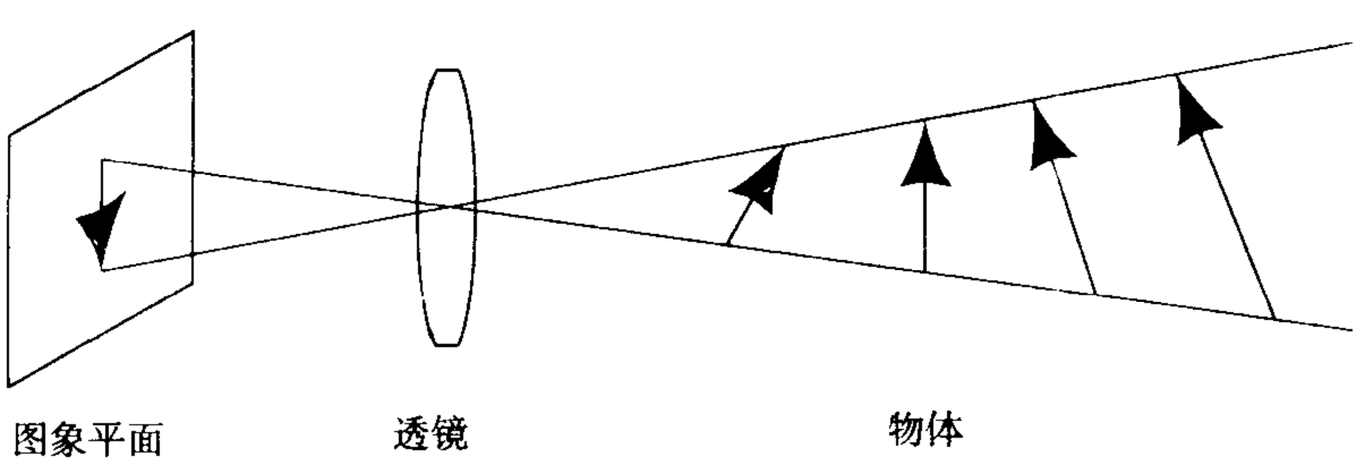


图6-1　图象生成过程的多对一特性

希望获取信息的种类取决于agent的目的和任务。若要让一个agent平安地通过一个混乱的

环境，这个agent必须了解其中物体的位置、边界、通路以及它所经路径表面的特性。agent若想要操纵物体，就必须知道这些物体的位置、大小、形状、成分和构造等。对其他目的而言，agent也许应了解颜色并能识别它们的类别。agent也许还应具备根据每隔一段时间所有以上信息的变化来预测将来可能的变化。从一个或多个图象中获取此类信息将极其困难，所以，如前所述，我只能给出这类技术的一个概况。

6.2　操纵一辆汽车

在S-R agent 的一些应用中，神经网络可用来把图象的亮度矩阵直接转换成动作。其中的一个突出的例子就是用来驾驶一辆汽车的ALVINN系统[⊖][Pomerleau 1991, Pomerleau 1993]。在介绍机器人视觉的一般过程之前，让我们先来看看这个系统。此系统的输入来自一个低解析度（30 × 32）的电视图象。一个电视摄像机被架在汽车上对准前面的道路，电视图象被采样并为神经网络产生一系列960维的输入向量。此网络如图6-2所示。

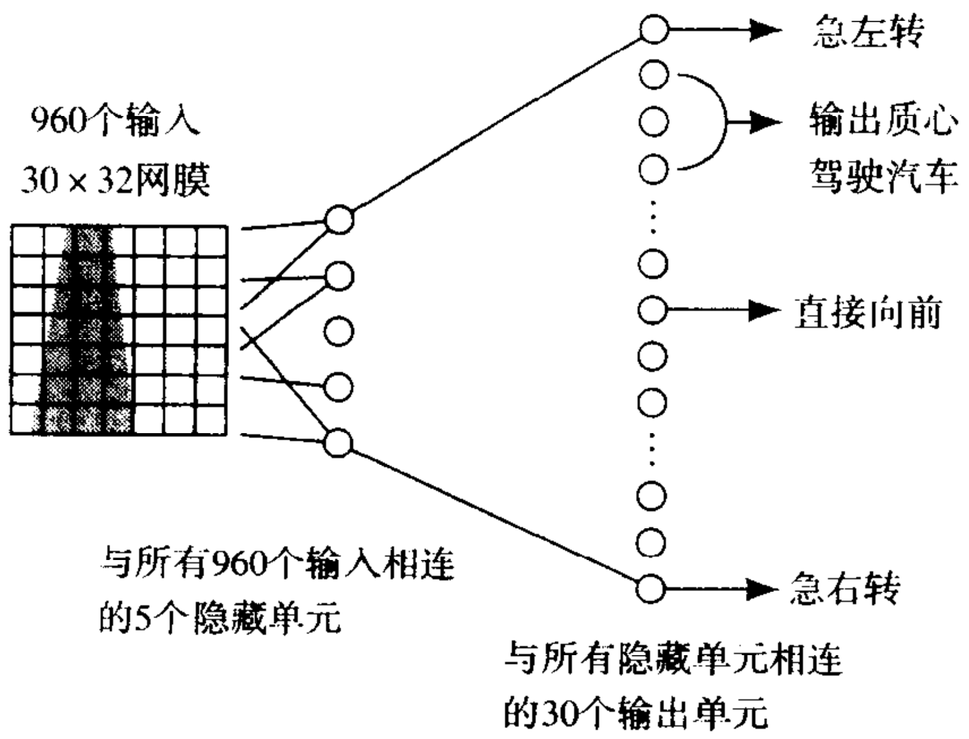


图6-2　ALVINN网络

网络的第一层有5个隐藏单元，第二层有30个输出单元，所有以上单元均为sigmoid单元。输出单元通过线性排列来控制汽车的角度。若此输出单元队列的顶端附近的一个输出单元的输出比其他大多数输出单元高，则车往左行驶；若在此队列的底端附近的一个单元的输出较高，则车往右行驶。计算出所有这些输出单元的响应的“质心”，并且把此车的驾驶角度设置为完全向左和完全向右之间相应的一个值。

此系统由一个改进过的“在空中（*on-the-fly*）”训练方式来训练：让一个真人驾驶员开车，实际的驾驶角度被作为相应输入的正确标志。网络以反向传播的方式递增地训练，从而使它能用驾驶员所指定的驾驶角度来响应实际驾驶车辆时出现的每一个视觉模式。整个训练大概要用五分钟的驾驶时间。

这个简单的训练过程经过改进后避免了两个潜在的问题。首先，因为驾驶员通常能很好地驾驶车辆，所以网络决不会经历任何偏离中心的车辆位置和不正确的车辆方位。其次，对长而笔直的路面，网络会在长时间的训练后，只知道产生笔直向前的驾驶角度；这种训练会使以前有关沿弯曲路面的训练无效。我们并不希望通过让驾驶员偶尔不稳定地驾驶来避免这些问题，因为系统将学会模拟这种不稳定的行为。

⊖　基于神经网络的自治的地面车辆。

所以，我们在软件中平移和旋转每个初始图象，从而产生14个补充图象，在这些图象中，车辆相对于路面的位置不同。再用一个模型来告知系统哪个图象应用哪个驾驶角度，并给出驾驶员为初始图象指定的驾驶角度，这样，系统将产生另外14个有标志的训练向量，并把它们加到一般训练过程中所遇到的那些训练向量上去。

经过训练，ALVINN可驾驶各种“试验”汽车在没有标线的铺筑过的路面、吉普车道、有标线的城市街道和州际高速公路上行驶。ALVINN曾在高速公路上以高于100公里／小时的速度连续行驶了120公里。

6.3 机器人视觉的两个阶段

尽管ALVINN的表演给人的印象很深，但许多机器人的任务需要对更高分辨率的图象进行更为复杂的处理。因为许多任务要求机器人了解场景中的物体，下面将集中讨论有关找出物体的技术。首先，什么是一个物体？在人造环境中，如建筑物的室内景观中，门口、家具、其他agent、人、墙及地板等等都是物体。在外部自然环境中，动物、植物、人造结构、汽车及道路等等都是物体。人造环境对机器人视觉来说通常比较容易，因为其中的物体有比较规则的边缘和表面。

有两种计算机视觉技术对勾勒出与场景中的物体相关的各部分图象的轮廓十分有用。一种技术是在图象中寻找“边缘”。一个图象边缘是图象的一部分，图象亮度或其他图象的特性在此处陡然变化。另一种技术试图把图象分为几个区域，一个区域也是图象的一部分，图象亮度或其他图象的特性在此处缓慢变化。图象中的边缘和区域之间的边界，经常但不总是与场景中产生图象的那些重要的、与物体相关的不连续点相对应。图6-3是一些不连续点的例子[⊖]。根据照明强度、表面特性和摄像机的角度，这些不连续点可由图象边缘和图象区域边界来表示。这样，获取这些图象特征成为机器人视觉要完成的一项重要的任务。

视觉处理过程可分成两个主要阶段，如图6-4所示。图象处理阶段主要把原始图象转换成更适合于景物分段的图象。图象处理包括降低噪声、增强边缘和寻找图象区域等不同的滤波操作。景物分析主要试图从已处理的图象中产生一个对原始场景的图标描述或基于特征的描述，并提供agent所处场景中与特定任务有关的信息。把机器人视觉分为两个阶段只是对这个过程的简化。实际的机器人视觉涉及更多的阶段，而且这些阶段一般都相互影响。

以后会详细讨论这两个阶段。但现在，为了了解一下概况，先来讨论一下图6-5所描绘的处

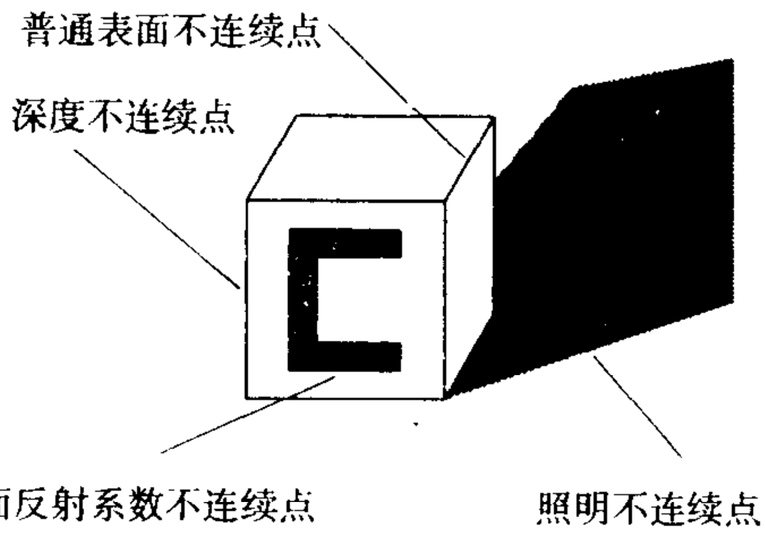


图6-3 场景的不连续点

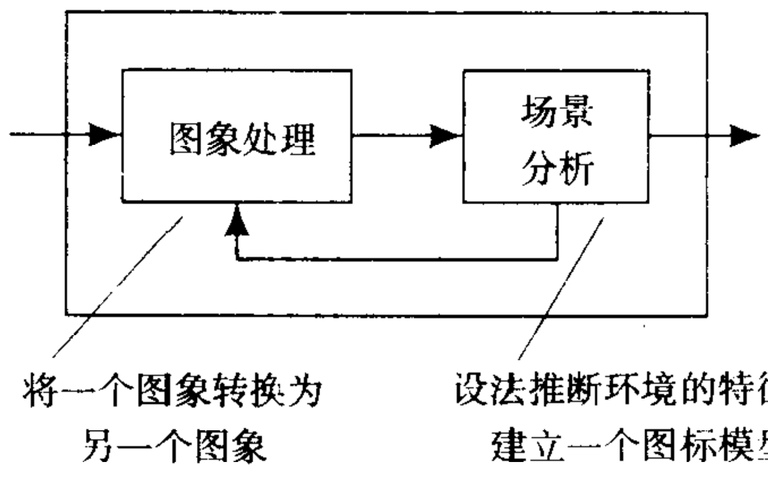


图6-4 机器人视觉的两个阶段

⊖ 节选自[Nalwa 1993, p.77]

于网格空间的机器人。机器人的视野中有三个标有*A*、*B*和*C*的玩具积木、一个门口和一个房间的一角。首先，图象处理排除伪造的噪声并增强物体的边缘以及其他不连续点。接着，已知世界中的物体的形状均由直线边界构成，景物分析会产生一个对此世界的图标表示——与用于计算机图形学中的模型相似。通常，这个图标模型用来更新存储在内存中的更全面的环境模型，然后计算出适合于这个假设环境状态的动作。

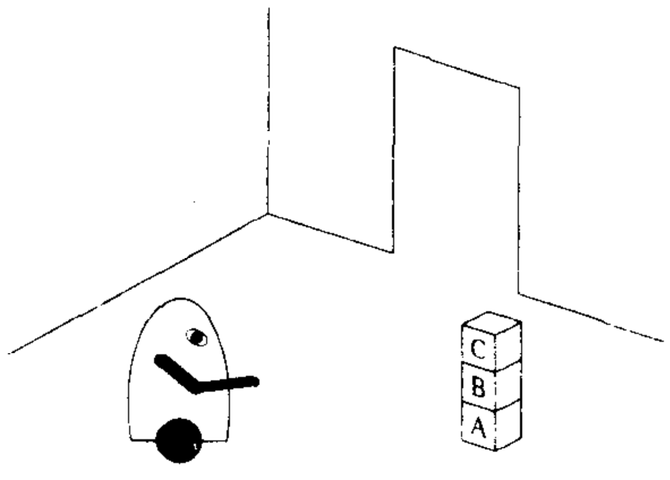


图6-5　机器人和元具积木都在一个房间里

根据其任务，图标模型并不一定要象计算机图形学模型那样描绘所有细节。若当前的任务仅与玩具积木有关，那么房间角落和门的位置就无关紧要了。我们不妨再假设只有积木的布局比较重要，那么，图标模型应为一个表结构((*C B A FLOOR*))，它表示*C*在*B*上，*B*在*A*上,而*A*在地板上。若*C*被移到地板上，那么图标模型应为((*C FLOOR*)(*B A FLOOR*))（也可以是((*B A FLOOR*)(*C FLOOR*)），但这里我们假设积木的相对水平位置无关紧要，这样，表结构的第一级元素的顺序就无表达意义）。因为每一个元件的最后一个元素均为*FLOOR*，所以我们可以去掉这一项来缩短表结构。

对于根本不用图标模型的机器人来说，景物分析会用另一种方法把处理过的图象直接转换成适合于机器人任务的特征。如，若机器人必须判定积木*C*上是否有其他积木，那么，一个对环境的描述应包括一个特征值，如*CLEAR_C*，积木*C*上无其他物体时这个特征值为1，否则为0（这里，为了使这些特征便于记忆，没有使用通常所用的x_i。必须记住，这些名字仅仅能帮助我们记忆，却无法帮助机器人记忆）。本例中，景物分析仅从已处理过的图象中计算特征的值。从以上例子中我们可以看出，景物分析完全根据所设计的机器人和它要完成的任务而定。

6.4　图象处理

6.4.1　平均法

假设初始图象可表达为一个$m \times n$数组$I(x,y)$，我们称之为“图象亮度数组(*image intensity array*)”。它把图象平面分成许多被称为“象素(pixel)”的单元。这些数字表示这幅图象中某点的光亮度。图象中一些不规则之处可通过求平均数的方法得以平滑。这个平滑操作就是把一个求平均数的窗口在整个数组中滑动。这一求平均数的窗口对准每个象素的中心，并计算出在求平均数窗口内的数字的加权总和，然后把此象素的初始值替换为这个加权总和。这种滑动并求和的操作称为“卷积(*convolution*)”。若我们希望所得的数组是二进制数字（1或0），那么就必须把这些加权总和与一个阈值比较。平均法不仅将压缩孤立的噪音点，而且将减小图象的卷曲度（*crispness*），并放弃那些微不足道的图象元素。

卷积是从信号处理中得来的操作。它通常被解释成对波形（沿时间轴滑动）的一维的操作。若我们沿一个信号$s(t)$滑动或卷积一个函数$w(t)$后，将得到平均信号$s^*(t)$：

$$s^*(t) = \int s(u)w(u-t)du = s(t) \star w(t)$$

用★来表示卷积。

图象处理中的二维离散式卷积如下：

$$I^*(x,y) = I(x,y) \star W(x,y) = \sum_{u=-\infty}^{\infty} \sum_{v=-\infty}^{\infty} I(u,v)W(u-x, v-y)$$

这里，$I(x,y)$是初始图象的数组，$W(u,v)$是卷积加权函数。假设$I(x,y)=0$当且仅当$x<0$或$x\geqslant n$，且$y<0$或$y\geqslant m$（这样，这个卷积操作会在图象的边界附近产生一些“边缘效应”）。

有时，我们把加权函数$W(x,y)$的值在x和y构成的长方形内看作1，长方形之外看作0。长方形的大小决定平滑度，长方形越大平滑度越高。图6-6展示了一个求平均数操作是如何对一个二进制图象先用一个长方形平滑函数平滑，然后将其与阈值比较来进行操作的（设图中的黑色象素的亮度值高而白色象素的亮度值低或为0。这个假设看起来好象是反的，但这样可以使图解简单些）。我们发现，这个平滑操作加粗了宽线，去除了窄线和微小细节。

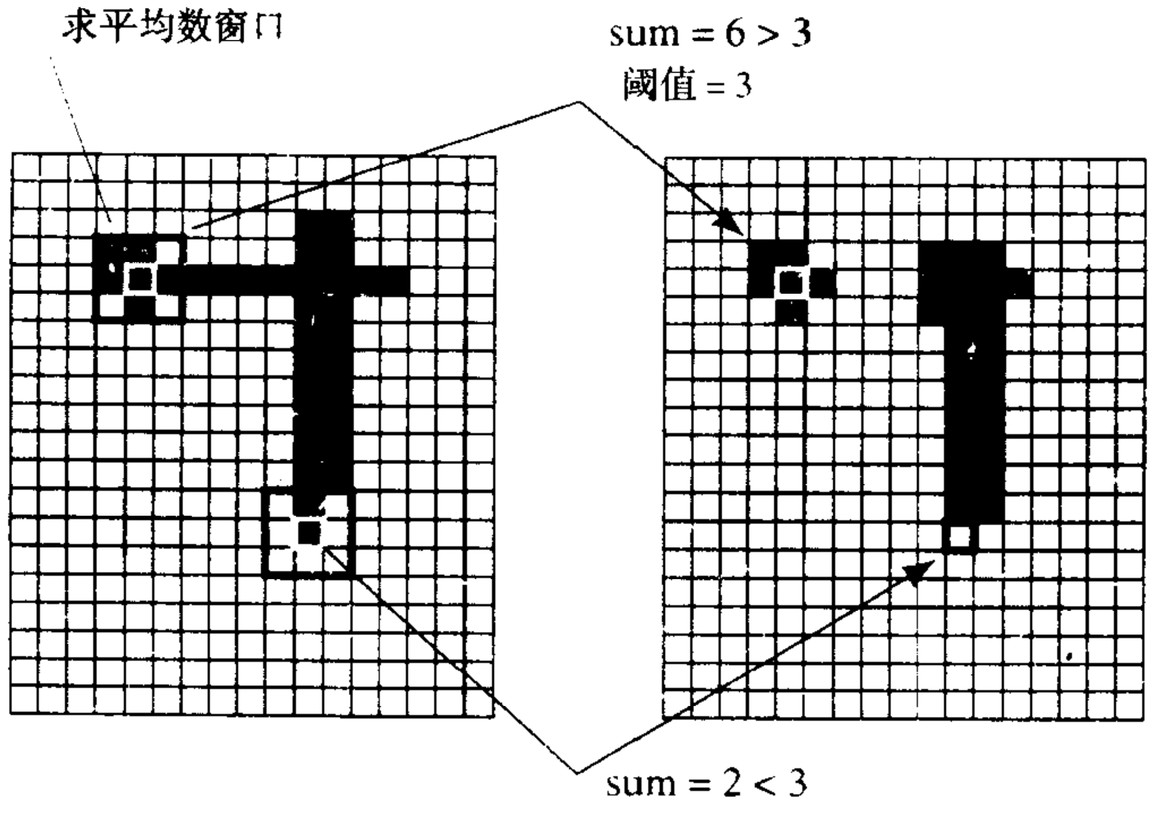


图6-6 求平均数操作的元素

用于平滑的常用函数是一个二维高斯函数（Gaussian）

$$W(x,y)=G(x,y)=\frac{1}{2\pi\sigma^2}e^{-\frac{x^2+y^2}{2\sigma^2}}$$

此函数描述的表面是铃铛形的，如图6-7（图中，移动了x和y轴的位置以便清楚地展示高斯表面）。高斯函数的标准差σ决定了表面的“宽度”，从而也就决定了平滑度。$G(x,y)$有一项是求x和y的积分。图6-8展示了采用不同的平滑因素对由积木和机器人组成的网格空间场景进行高斯平滑后的三个图象以及它们的初始图象[⊖]（通常，图象平滑和过滤的离散操作对离散值进行插值以增强表现效果）。

可以发现图6-8中的图象一个比一个模糊。我们可以这样来解释这种现象：假想图象亮度函数$I(x,y)$表示一个长方形导热板的初始温度场。随着时间的流逝，导热板各向同性地散热导致高温与低温混在一起。依照这种观点，图6-8中排列的图象按时间先后依次表现了当时的温度场。Koenderink[Koenderink 1984]指出，用标准差σ的高斯函数来卷积一个图象等同于在已知图象亮度域情况下求一个时间与σ成比例的扩散方程的解。

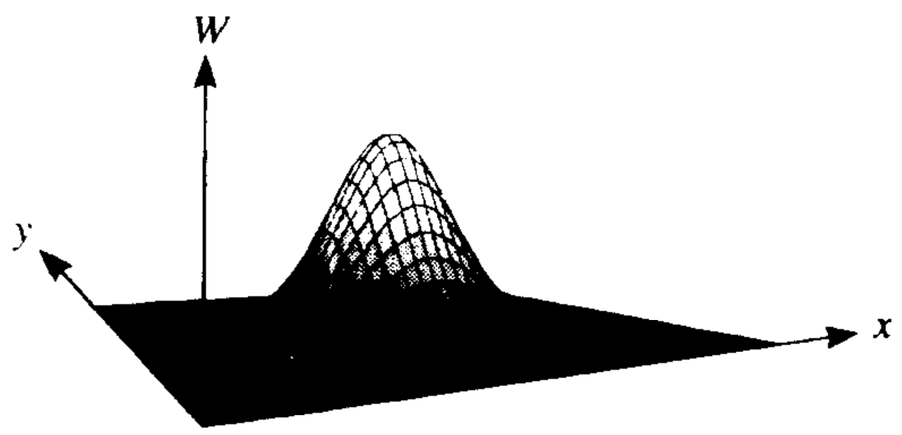


图6-7 高斯平滑函数

⊖ 在此，我要感谢Charles Richad，他创造了本章所用的图象并为这些图象的处理操作进行了编程。

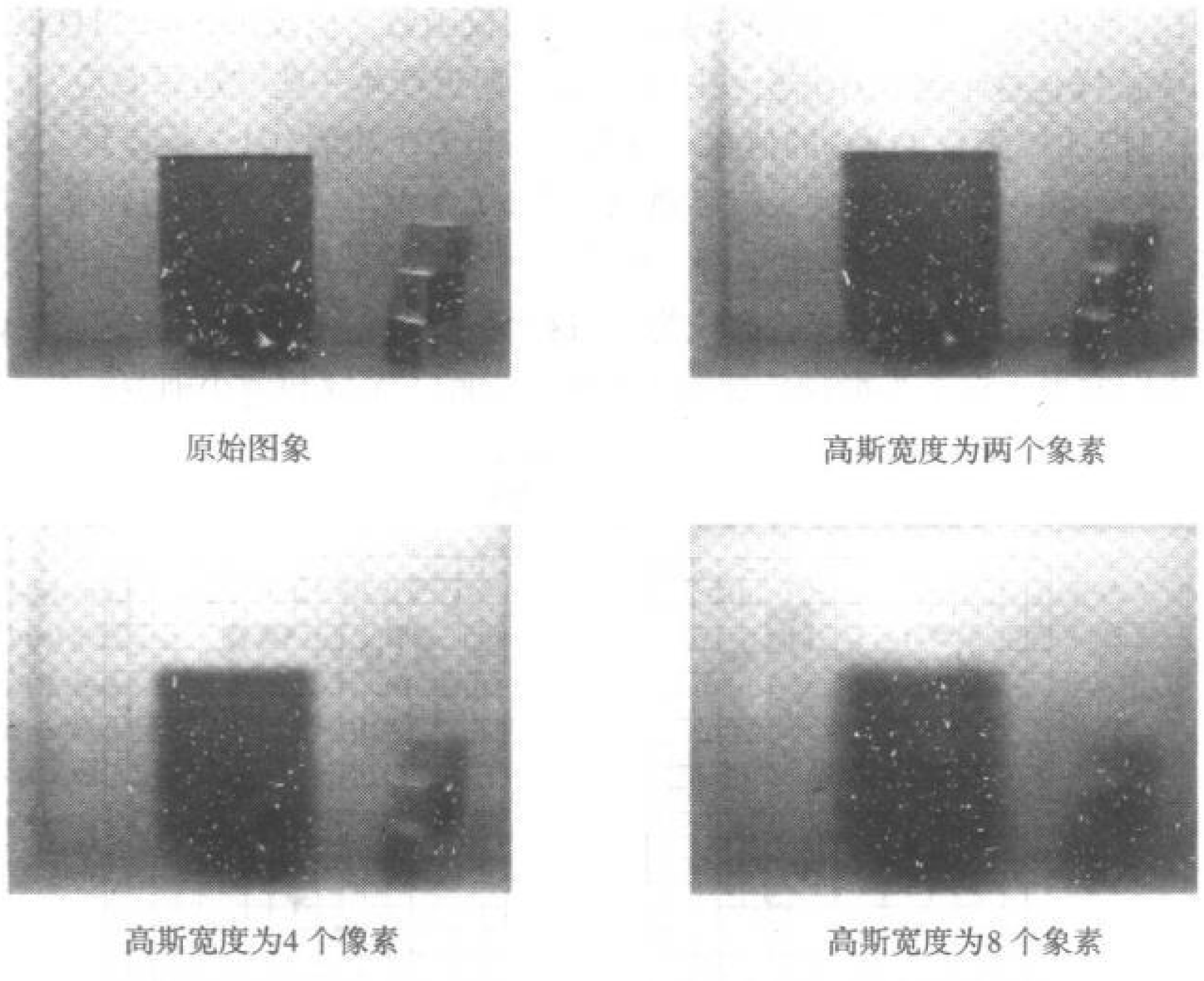


图6-8　用高斯函数过滤的图象平滑过程

6.4.2　边缘增强

如前所述，计算机视觉常常涉及图象边缘的提取，然后用这些边缘来把图象转换成某种线条图形。转换后的图象中的轮廓可与场景所包含的各类物体的原型（即模型）的轮廓特征相比较。获取轮廓的方法之一是先增强图象中的边界和边缘，边缘可以是图象各部分之间的任意边界，这些边缘的特性，如亮度，彼此之间明显不同。这样的边缘常常与图6-3所示的重要物体特性有关。

首先讨论一维图象，这意味着$I(x,y)$只随x的变化而变化，并不随y变化。然后介绍一些二维图象。我们可以通过在一维图象上卷积一个位于垂直线上的、一半为负一半为正的窗口（如图6-9）来增强这些图象的边缘强度。在图象的平均部分此窗口的总和为0。

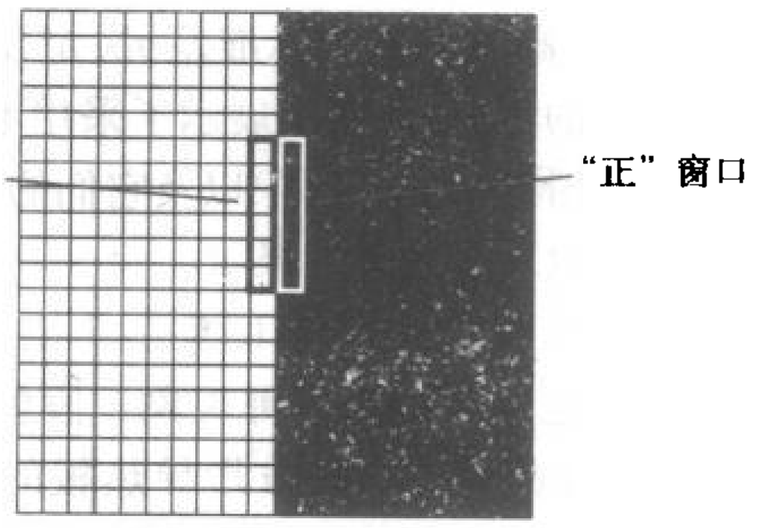


图6-9　边缘增强

若在图象上沿x方向卷积图6-9所示的窗口，那么，在与y方向连成一条线的边缘处会形成一个尖峰。这个操作与对图象亮度函数相对于x第一次求导（即dI/dx）极其相似。若我们对它们进行第二次求导，效果会更明显。由此，我们将得到一个经过0点的波形，边缘的一边为正波形而另一边为负波形。以上操作在一维空间的效果图如图6-10所示。这样，亮度的变化比较平滑，不象图6-9中的变化那么剧烈。当然，图象亮度的变化越陡，dI/dx的顶峰就会越窄。图象边缘出现在$d^2I/dx^2 = 0$处，即图象的二次导数的0点。

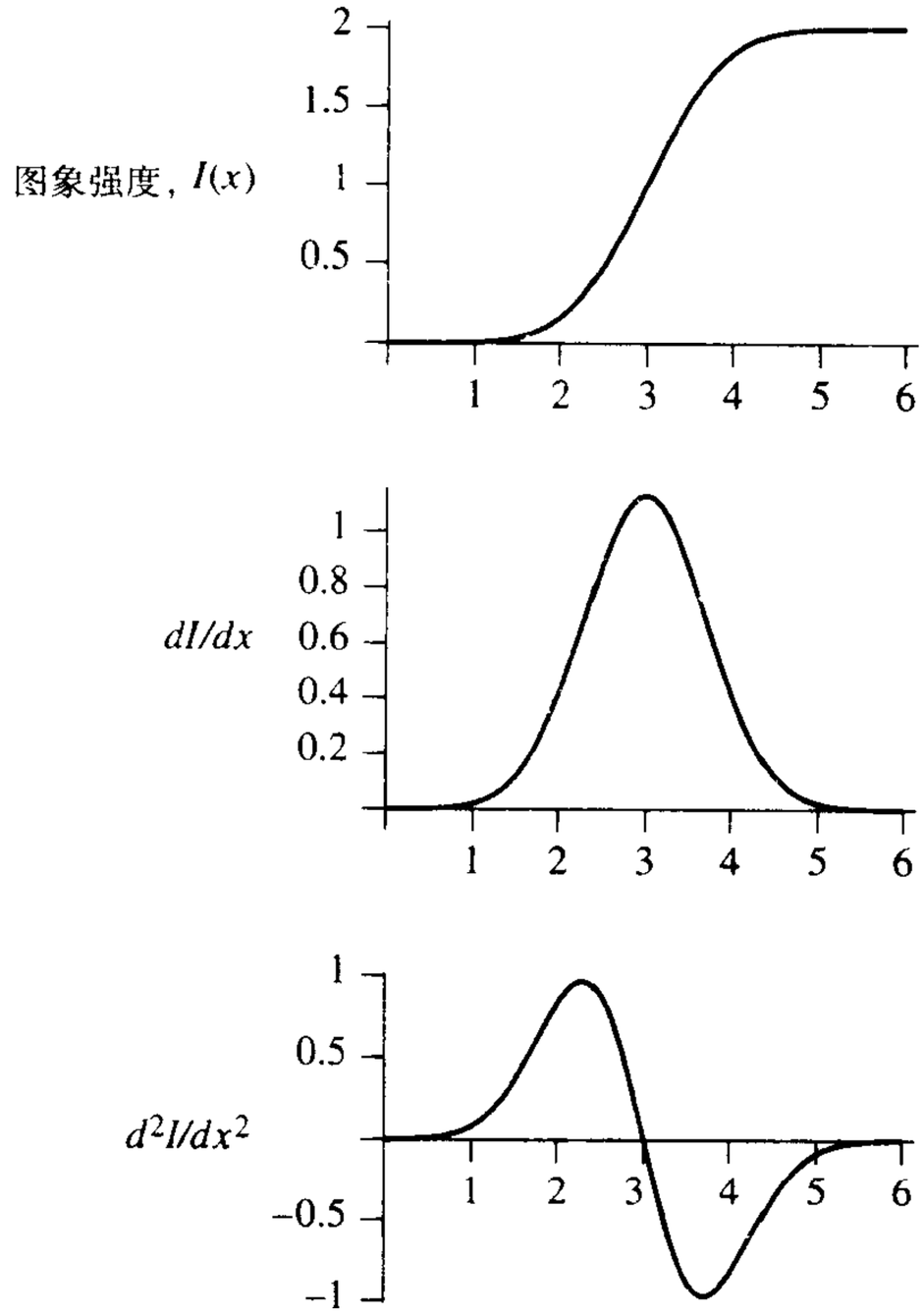


图6-10 对图象强度求导

6.4.3 边缘增强与平均法的结合

边缘增强本身将在增强边缘的同时突出图象中的假噪声元素。为了减小对噪声的敏感度，可以先用平均法再用边缘增强来把两种操作结合起来。于是，我们首先用一维高斯函数对连续的一维图象进行平滑处理。这一高斯函数为：

$$G(x) = \frac{1}{\sqrt{2\pi}\sigma} e^{-\frac{x^2}{2\sigma^2}}$$

其中，σ为标准差，它是此平滑函数的宽度的一个量度。用高斯函数平滑后得到过滤后的图象：

$$I^*(x) = I(x) \star G(x) = \int I(u)G(u-x)\,du$$

随后，通过边缘增强得出：

$$d^2[I^*(x)]/dx^2 = d^2[I(x) \star G(x)]/dx^2 = d^2\left[\int I(u)G(u-x)\,du\right]/dx^2$$

因为求导和求积分的顺序可互换，上式等同于$I(x)\bigstar d^2G(x)/dx^2$。这样把平滑过程与边缘增强结合后，我们可以用一个高斯曲线的二次导数来卷积一维图象，而不必对图象二次求导。

现在，来看看二维图象。我们需要一个二次求导型操作来增强任一方位的边缘强度。拉普拉斯变换就是这样的操作。$I(x,y)$的拉普拉斯变换的定义如下：

$$\nabla^2 I(x,y) = \partial^2 I(x,y)/\partial x^2 + \partial^2 I(x,y)/\partial y^2$$

如果要在二维空间中把边缘增强和高斯平滑结合起来，我们必须改变求导和卷积的顺序（与在一维空间所做的一样），因此得到

$$I(x,y) \star [\partial^2 G(x,y)/\partial x^2 + \partial^2 G(x,y)/\partial y^2]$$

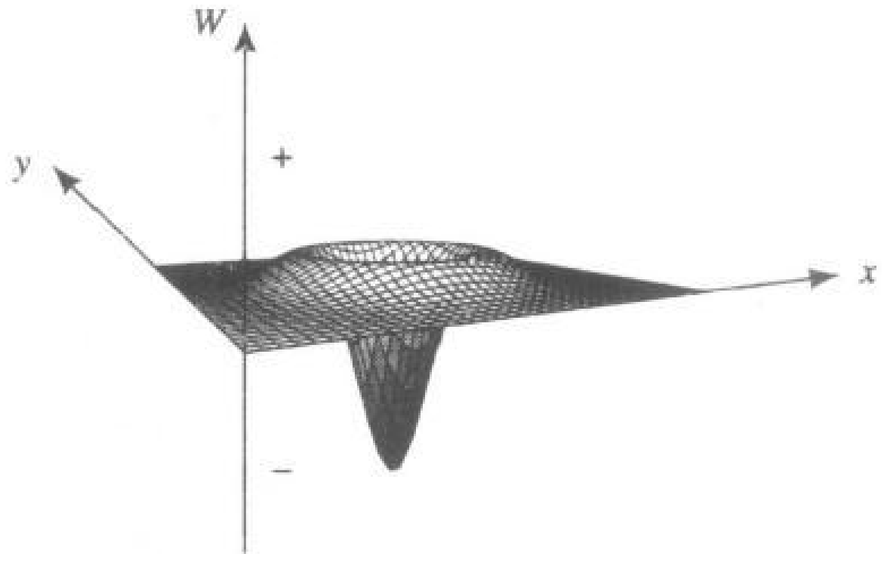


图6-11　用于拉普拉斯过滤的Sombrero函数

二维高斯函数的拉普拉斯变换有点像一顶倒置的帽子，如图6-11所示（这里，再次移动了坐标空间）。它又被称为“*sombrero*（宽边帽）函数”，帽宽决定了平滑度。

然后，通过用这个帽函数来卷积图象，就可以完成整个求平均和边缘寻找的操作。这个操作又被称为“拉普拉斯过滤(*laplacian filtering*)”，它产生的图象叫做“拉普拉斯过滤图象”。根据脊椎动物视网膜上的视觉处理已被发现与拉普拉斯过滤有些相似。而且，拉普拉斯过滤图象的过0点可用来产生大体轮廓图。拉普拉斯过滤和标出过0点这两个过程构成了所谓的“Marr-Hildreth” 算子[Marr & Hildreth 1980]（Marr-Hidreth算子的输出就是Marr所称的“原始轮廓”的一部分）。图6-12是以上操作对网格空间场景的图象所产生的效果[⊖]。我们注意到，Marr-Hilldreth算子为画出由简单边界组成的场景的各部分的轮廓素描提供了合理的基础，然而它却未能很好地画出图中位于门口的那个稍微复杂的机器人的轮廓。

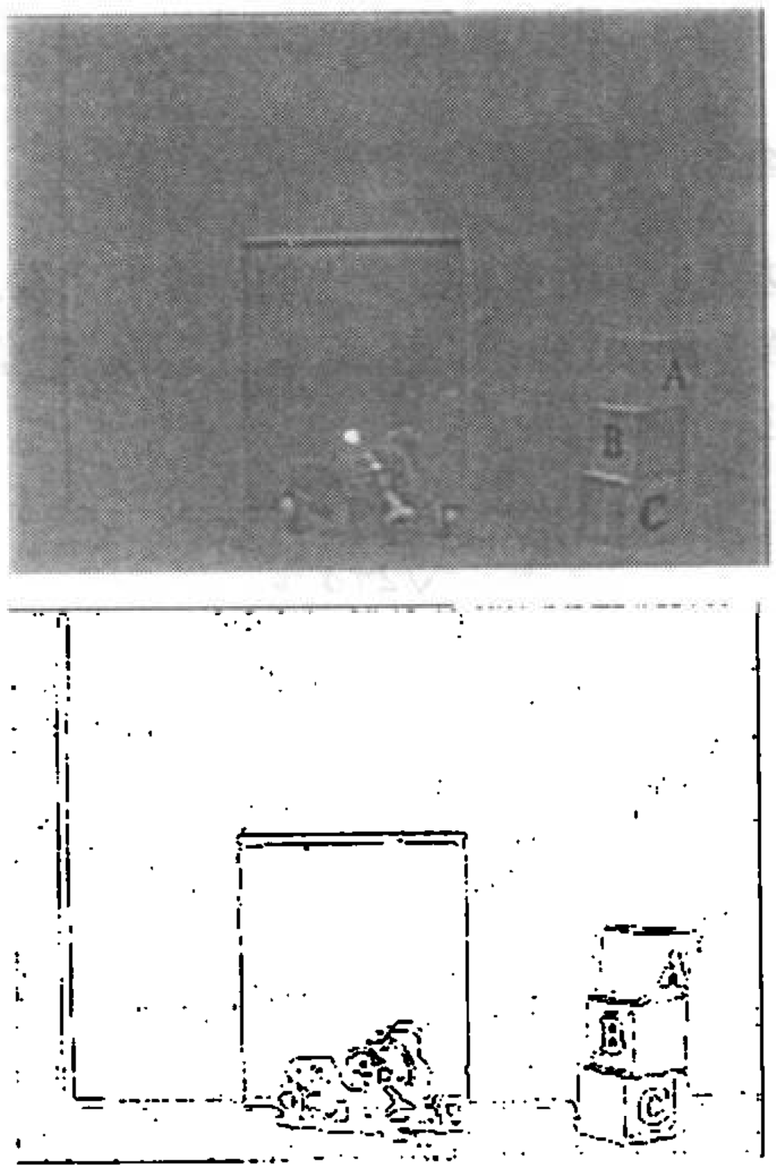


图6-12　拉普拉斯过滤和Marr-Hildreth算子（拉普拉斯变换的宽为1象素）

实际还存在许多其他边缘增强和线条寻找的操作，有些产生的效果比这种易于解释且十分

⊖　在计算图6-12的图象时，用段交叉点来代替过0点。图象亮度必须与0周围的波段相交才能被显现出来。

流行的拉普拉斯变换更好。其中突出的有：Canny变换（[Canny 1986]）、Sobel变换（归功于Irwin Sobel的[Pingle 1969]）、Hueckel变换（[Hueckel 1973]）和Nalwa-Binford变换（[Nalwa & Binford 1986]）。值得注意的是，Marr-Hildreth和其他边缘增强变换开始均把象素标号为可能处于图象边缘或线条的候选者，然后再把这些候选者连接起来形成轮廓或者其他简单的曲线。

6.4.4 区域查找

另一种处理图象的方法试图在图象中查找亮度或其他特性，如纹理等变化不突然的“区域”。从某种意义上来讲，查找区域是查找轮廓的对等物（*dual*）；这两种技术均把图象分割成我们所希望的与场景相关的若干部分，但由于二者均对噪声比较敏感，因此这两种技术通常用来互补。

首先，我们必须定义什么是图象的一个区域。一个区域就是一组满足以下特性的相互连接的象素：

1) 一个区域由类似的成分组成。常用的同质特性（*homogeneity property*）如下：

(a) 在这个区域中，象素的亮度值之间的差别不超过某个ε。

(b) k次多项式（k的值比较低且事先指定）的表面可与此区域内象素的亮度值以小于ε的最大误差（即表面与区域亮度值之间的误差）拟合。

2) 任意两个毗邻的区域内的所有象素的组合不满足同质特性。

通常，把一个图象分割成区域的方式不止一种，但每个区域总是与世界中的一个物体或其有意义的一部分相对应。

与边缘增强和线条查找一样，用来把图象分割成区域的技术有很多，下面介绍其中一种叫做“分割—合并(*split-and-merge*)”的方法[Horowitz & Pavlidis 1976]。若用一种简单的方法来描述，可首先把整个图象作为一个候选区域。为了便于图解，我们设此图象为一个由$2^l \times 2^l$数组组成的正方形。显然，这个候选区域并不满足“区域”这一定义，因为图象中所有的象素集不满足同质特性（具有相同亮度的图象除外）。于是我们把每个不满足同质特性的候选区域分割成另外四个等大的候选区域，并一直这样分割下去直至无需再进行分割。图6-13是一个人工的8×8图象的分割过程，其中运用了亮度的差别不超过1个单元这个同质特性。当无须再进行分割时，可以合并那些满足此同质特性的彼此毗邻的候选区域。可用不同的顺序进行兼并——所得的最终区域也会不同。实际上，在分割完成之前也可以进行合并。为了简化图解，图6-13中的所有合并在最后一步进行。

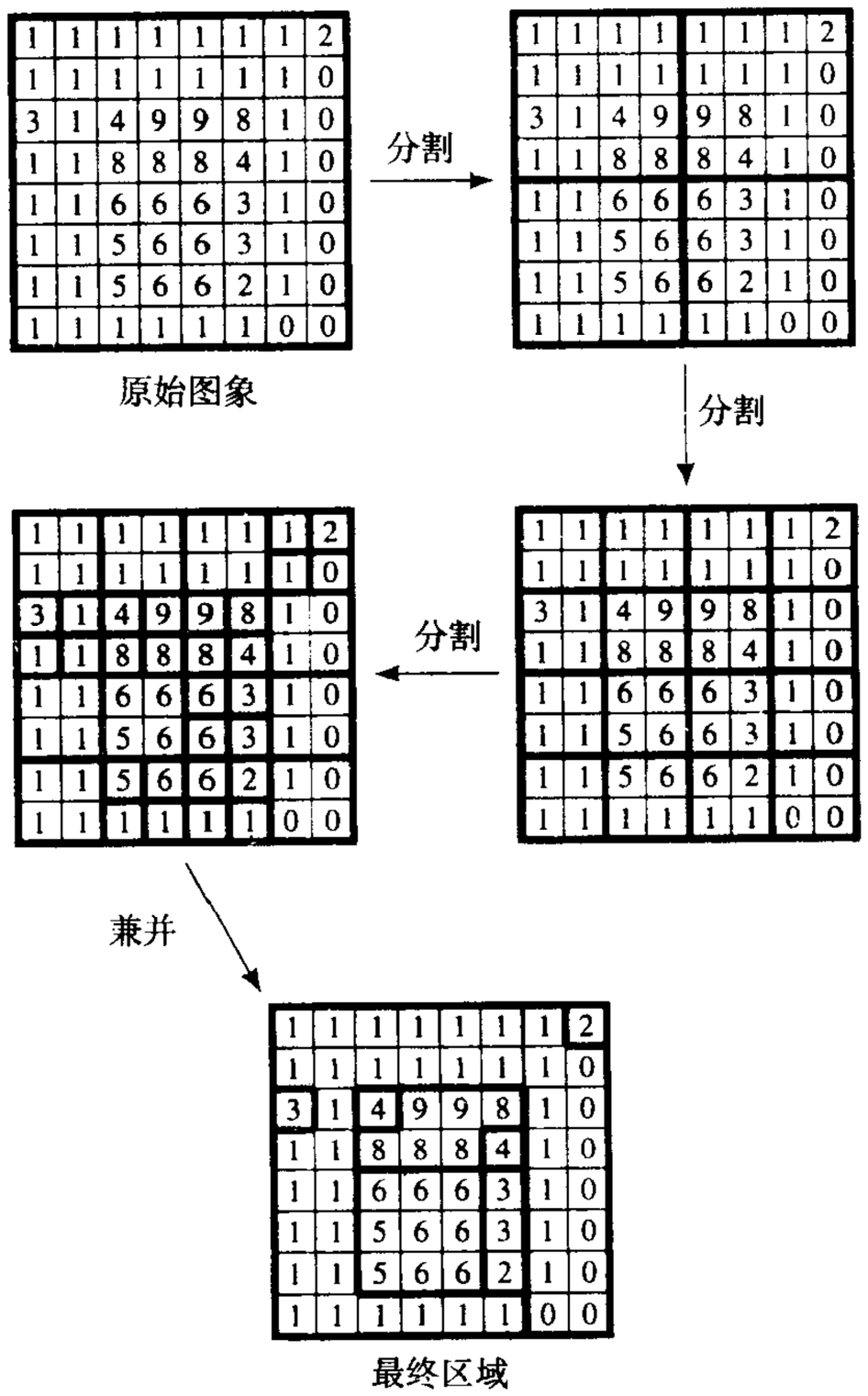


图6-13 分割和兼并候选区域

图6-13中，用的是低分辨率的图象来说

明区域查找过程。图6-14展示了一个高分辨率的图象经过此过程后的结果。与图6-13一样，我们发现了一些微小区域和许多不规则的区域边界。通常，可以通过去掉微小区域（其中有些是较大区域的过渡地带）、规整边界轮廓以及考虑可能处于场景中的物体的已知形状等等来“净化”用分割—合并算法找到的区域。图6-14中，网格世界里沿墙的亮度梯度产生了许多区域。

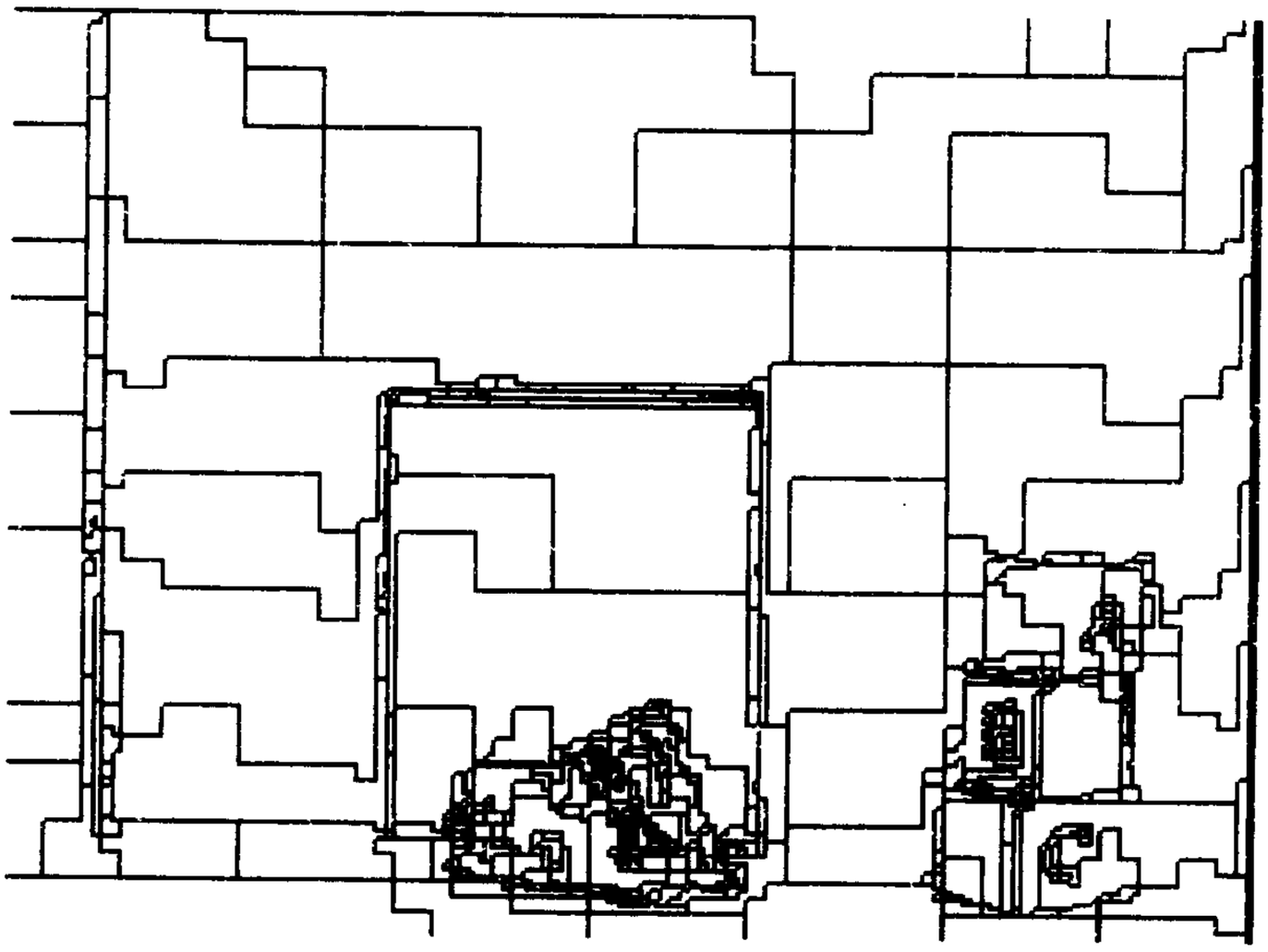


图6-14　在一个网格场景中用分割—合并算法查找而来的区域

回想一下，在讨论用高斯函数平滑图象时，提到了各向同性热扩散过程。Perona和Malik[Perona & Malik 1990]曾提出一个可用于生成区域的各向异性扩散过程。这个过程鼓励亮度变化小的平滑而拒绝亮度变化大的平滑。如[Nalwa 1993, p. 96]所述，这一过程结果将形成亮度相同的不同区域，而区域之间的边界的亮度梯度很高。

6.4.5　运用亮度以外的其他图象的属性

边缘增强和区域查找还可以基于除图象亮度的同质特性以外的其他图象属性。世界上众多物体的表面反光度有细微的差别，我们称之为视觉纹理。如一片草地、一块地毯、一簇树叶、动物的皮毛等等，它们的表面反光度均彼此不同。而这些物体反光度的强异会在图象高度上产生类似细微差别。

计算机视觉研究者已经能识别多种视觉纹理，并且开发了各种纹理分析工具，其中有结构化方法和统计方法。这些方法或者用来把某种具体纹理的图象的各个部分进行归类，或者用来把图象分割成各自拥有特殊纹理的不同区域。结构化方法力图用由原始“texels”（即是由黑白部分组成的微小形状）构成的棋盘形布置来表示图象区域（[Ballard & Brown 1982, 第6章]有完整的讨论）。

统计方法基于以下观点：图象区域的亮度值的概率分布能很好地描述图象的纹理。举一个简略的例子：有这样一张草地的图象，图中的草叶垂直生长。这个图象将有这样一个概率分布：被低亮度区域分割开的、窄且垂直分布的高亮度区域有峰值。最近，由Zhu及其同事所著的[Zhu, Wu, & Mumford 1998]介绍了估算各种视觉纹理的概率分布的方法。一旦我们知道这些分

布，就可以用它们来对纹理归类，并在此基础上分割图象。

除了纹理，我们可能还会用到其他图象属性。若用一个直接的方法来测量场景中的物体到摄像机的距离（如用一个激光距离探测器），则可以产生一个“距离图象”（图象中每个象素值表达场景中某点到摄像机的距离），并找出陡然的距离变化。至于运动和颜色等可感知或计算的其他特性，可以通过图象处理操作而得到。

6.5 场景分析

在用以上所讨论的技术对图象进行处理后，我们力图从中获取所需的有关场景的信息。计算机视觉的这个阶段被称为“场景分析(*scene analysis*)”。由于场景—图象的转换是多对一的，场景分析需要其他补充图象或有关将遇到的场景种类的大体信息。我将在后面描述立体视觉时讨论补充图象的运用；这里，将介绍用场景知识获取场景信息的不同方法。

这些所需的场景知识可能十分概括（如物体的反光度），也可能十分具体（如场景可能包括门旁的一些叠放在一起的箱子）。它可能清楚，也可能模糊。如，在线条查找算法的操作中就可能建有“什么构成了一条线”这个模糊知识。处于这两种极端之间的其他场景信息包括摄像机的位置、照明光源的位置和场景位于办公楼内还是室外等等。下面所讨论的场景知识均属于以上范围。有关这方面更详细的论述，请参阅计算机视觉的教材。

表面反光度特性和图象亮度的明暗常用来给出场景中光滑物体形状的信息。而图象明暗尤其能帮助我们计算物体的表面法线。Horn及其同事已经发明了从图象明暗中推出形状的方法（请参阅[Horn 1986]中的描述）。图象中所表达的纹理元素是场景中的元素的透视投影，这使从纹理中获取形状和质深（*qualitative-depth*）更加容易。

如前所述，我们有时需要一个场景的图标模型，但有时仅需场景的某些特征。图标景物分析通常力图建立一个场景或部分场景的模型。基于特征的景物分析仅获取当前任务所需的场景的特征。一种有代表性的基于特征的景物分析被称作“面向任务的(*task-oriented*)”或“意图(*purposive*)”视觉(可参见[Ballard 1991,Aloimonos 1993])。

6.5.1 解释图象中的线条和曲线

对已知包含直线物体的场景（如建筑物内部场景和网格世界的场景）进行分析时，其中关键的一步就是图象中线条的假定（这些线条后来将与场景的关键元素相关）。可以通过采用把直线段与边缘或区域的边界拟合的技术来生成直线。对于包含曲线物体的场景，我们可以把圆锥截面（如椭圆、抛物线和双曲线）与原始轮廓或区域的边界拟合（请参阅[Nalwa & Pauchon 1987]）来生成曲线。在经过去除短线、在端点处连接直线和曲线这些技术操作后，把图象转化成一个线条画(*line drawing*)，这幅线条画可用于进一步解释。

有很多把场景特性与线条画的元素相结合的策略。这样的结合称为“解释(*interpreting*)”线条画。这里，将介绍一种解释线条画的策略。在这种策略中，已知场景仅包含平面，从而使相交于一点的平面不超过三个（这种平面组合体称为“三面体顶点多面体(*trihedral vertex polyhedra*)”）。图6-15是此种场景的一个典型例子，它是一个由边界墙、地板、天花板和一地板上的正方体组成的室内场景。在这样的场景中，由两个相交平面组成的场景的边缘只有三种。一种边缘的两个相交平面的其中一个遮住了另一个（即在场景中只能看见其中的一个平面），这种边缘称为“*occlude*”。图6-15中用箭头（→）做标记的就是occlude，箭头沿边缘的指向使

得遮住另一个平面的平面位于箭头的右边。另两种边缘的两个相交平面在场景中均可见。其中形成的凸边称为“刀刃（*blade*）”，图中的标记为加号（+）；形成的凹边称为“折痕（*fold*）”，图中的标记为减号（-）。

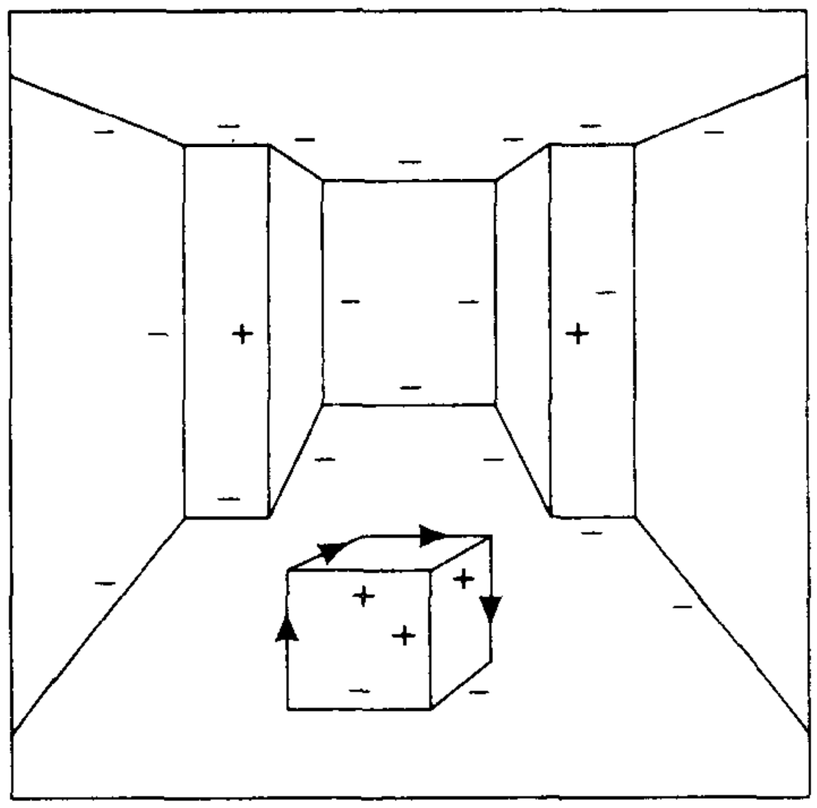


图6-15　一个房间的场景

假设能从这样的场景中得到一个可转化成线条画的图象。如图6-15所示。我们能通过这种给场景中的线条作标记的方法来精确地描述场景中的边缘的种类吗？在一定条件下，我们能。首先，从一个“一般的视点”来得到一个场景的图象——即这种视点不会使场景中的任两条边缘在图象中连成一条线。然后再用基于以下事实的方法来给图象中的线条作标记——（在我们假设的三面体顶点多面体中）只能存在有限种给图象中线条的连接点作标记的方法。图6-16便是这些标记方法。尽管还存在其他许多种给图象中线条的连接点作标记的方法，但在多面体的场景中，图6-16所示的就是所有可能存在的标记种类。

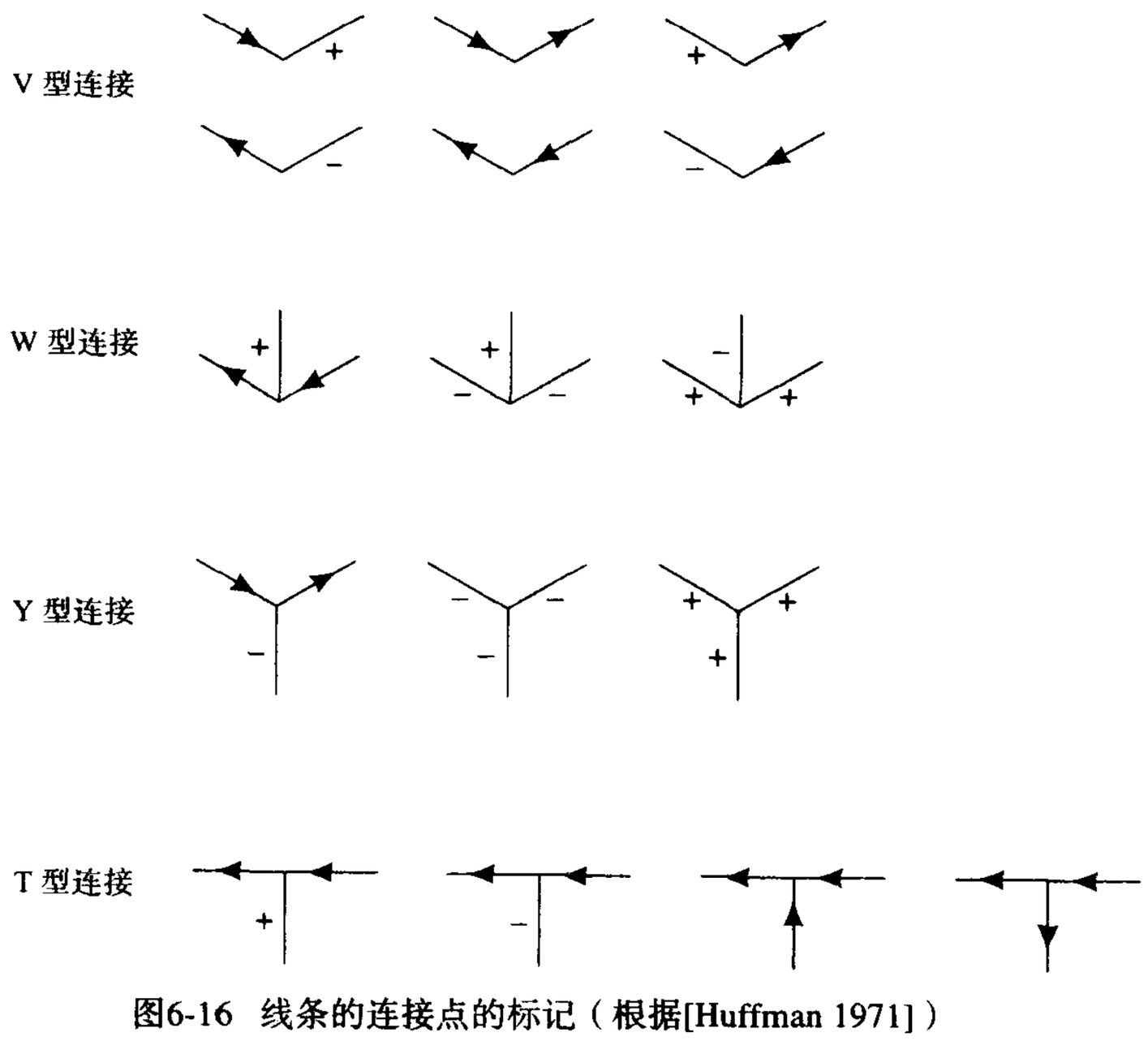


图6-16　线条的连接点的标记（根据[Huffman 1971]）

标记线条的景物分析过程如下：首先，根据线条连接的形状，给图象中所有的连接点分别标上V、W、Y或T。在图6-17的房间场景的图象中，已经按以上方法给连接点作好了标记。然后再给图象中的线条分别标上+、-或→，但必须遵循图6-16中的规则。而且，连接两个连接点的图象线条的标记必须前后一致。这些约束条件通常（但不是总是）导致只能有一种标记方法。若这些标记前后不一致，那么，在把图象转化成线条画时就会出错，或者这时所用的场景

不是三面体多面体。在给图象线条作标记时，由这些约束条件产生的问题在人工智能中称为“约束满足问题”。在后面会讨论解决这类普遍问题的方法，但你现在不妨试着给图6-17作出一种前后一致的标记（当然，如果图象的标记与图6-15中场景的标记一一相应，那么这就是一种前后一致的标记方法。然而，这个场景是被规定有那些标记的。你能想出一种自动地为图象线条作前后一致标记的方法吗？）。

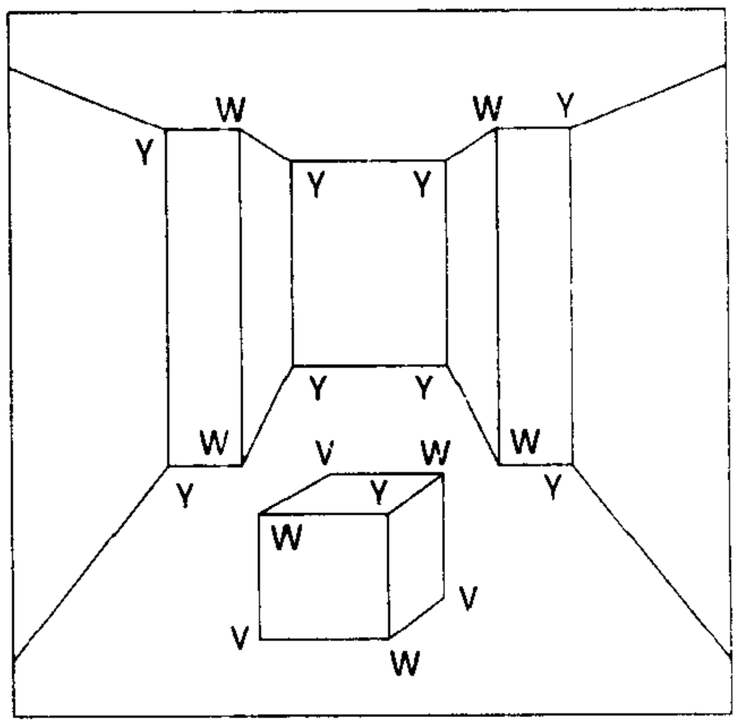


图6-17 按类型给图像连接点作标记

[Guzman 1986, Huffman 1971, Clowes 1977]首先开始研究给三面体顶点多面体图象中的线条作标记的景物分析技术，[Waltz 1975]以及其他书籍对此作了延伸。对包含非平面表面的场景的类似分析也获得了一定的成功。有关线条画解释的更详尽的解说（附引用），请参阅[Nalwa 1993, 第4章]。

大量有关场景的有用信息能从对线条画中线条和曲线的解释中得到。如机器人可预测到，向一个垂直的fold(场景中的一个凹形边缘)移动最终会使其到达一个角落。绕过多面体障碍可以通过绕过垂直的blade（凸形边缘）来实现。只要有足够的有关场景的一般知识，那么所需的特征或图标模型就可以直接从已解释过的线条画中得到。

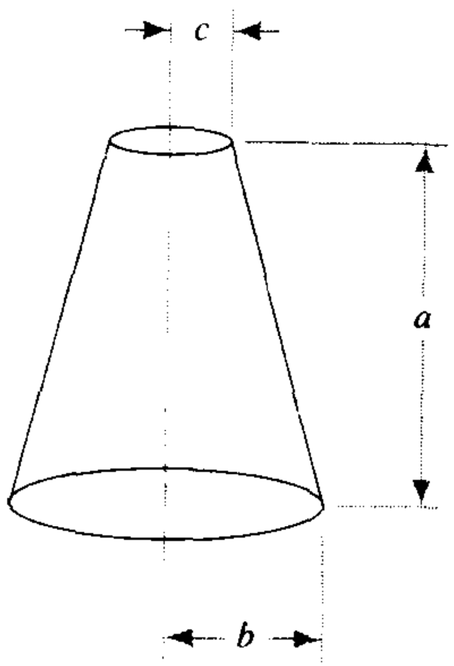
6.5.2 基于模型的视觉

计算机视觉技术运用越来越多的场景知识，现在来看看运用可能出现在场景中的物体模型的技术。例如，若已知场景由机器人组装所用的工业部件和零件所组成，那么，这些部件和零件的形状模型可用来帮助解释图象。下面概括性地介绍一些用于基于模型的视觉方法。若想作进一步的了解，请参阅[Binford 1982, Grimson 1990, Shirai 1987]。

正如线条和曲线可与图象中的区域的边界部分拟合，模型的透视投影也可与模型的各部分进行拟合。例如，如果已知一个场景包括一个平行六面体（如图6-15中的场景），我们可以把此平行六面体的一个投影与此场景图象的各个元素相拟合。此平行六面体有其给定大小、位置和方位的参数。调节这些参数，找到一组参数能使此平行六面体与图象中的一组线条很好地拟合。

研究者们还运用通用圆筒（*generalized cylinder*）[Binford 1987]作为建构模型的积木。此通用圆筒如图6-18所示。每个圆筒有九个参数。图6-19是用此种通用圆筒对一个人体的粗略的场景重建。这种方法也适用于分层表示，因为每个圆筒可与一组更小的、能更精确地表示图形的圆筒相连接。正如Nalwa所述，用分层的通用圆筒来表达场景中的景物“说起来容易，做起来难”[Nalwa 1993, p. 293]，但这种方法已经运用于一些物体识别的应用中[Brooks 1981]。有关三维结构模型表示的运用，请参阅[Ballard & Brown 1982, 第9章]。

我们可用不同的模型元素和模型拟合来生成一个整个场景的图标模型，或得到足够的有关场景的信息来获取当前任务所需的特征。通过把实际图象与用场景分析得来的图标模型构建的模拟图象进行比较，基于模型的方法能测试这些模拟图象的准确度。这些模拟图象必须由运用参数的模型来绘制，而这些参数与图象处理过程所用的参数（如摄像机角度等）相似。这样，就需要照明、表面反光特征以及计算机图形学的绘图过程的其他各方面的所有合适的模型。



a, *b*, *c* + 6]—个位置参数

图6-18　一个通用圆筒

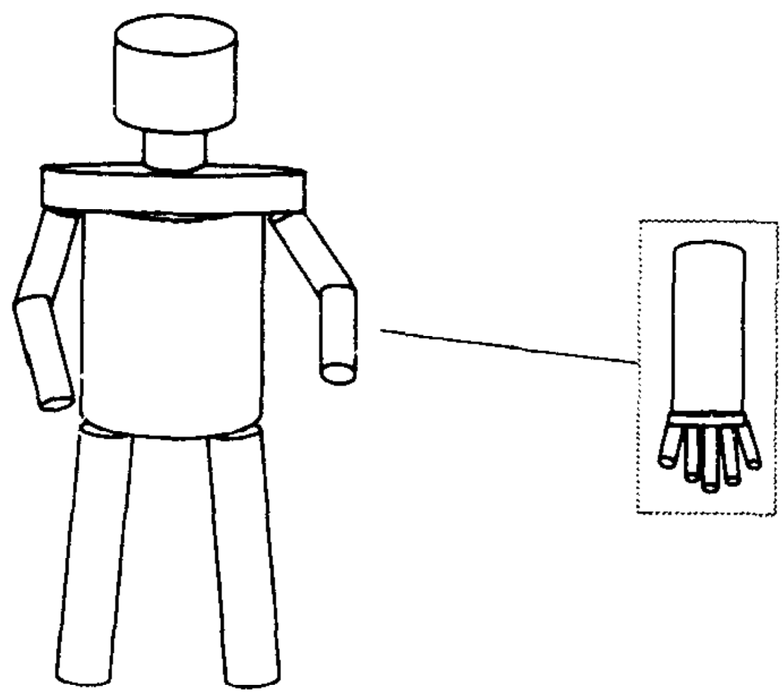


图6-19　运用通用圆筒的一个场景模型

6.6　立体视觉和深度信息

透视投影会使一个大而远的物体与一个与其相似的小而近的物体所产生的图象相同。这样，从单个图象估量物体的距离就十分困难了。但我们可运用立体视觉(*stereo vision*)来得到深度信息，这种方法是基于两个（或两个以上）图象的三角测量计算的运用。

然而，在讨论立体视觉之前，需要指出，在某些情况下如具备适当预备知识时，我们可以从单一图象中获取深度信息。例如，对图象的纹理分析（考虑场景纹理的透视变形这一因素）表明场景中有些元素比另一些更近。在一定的情况下，我们还能得到更精确的深度信息。例如，在一个办公室场景中，若已知所观察的物体在地板上和摄像机离此地板的高度，那么，运用从摄像机镜头的中心到图象上某一恰当的点的角度，可算出与物体的距离。图6-20图解了这种计算（角度α可用摄像机的焦距和图象的维数来计算）。可用相似的方法来计算与门口的距离以及物体的大小等等。

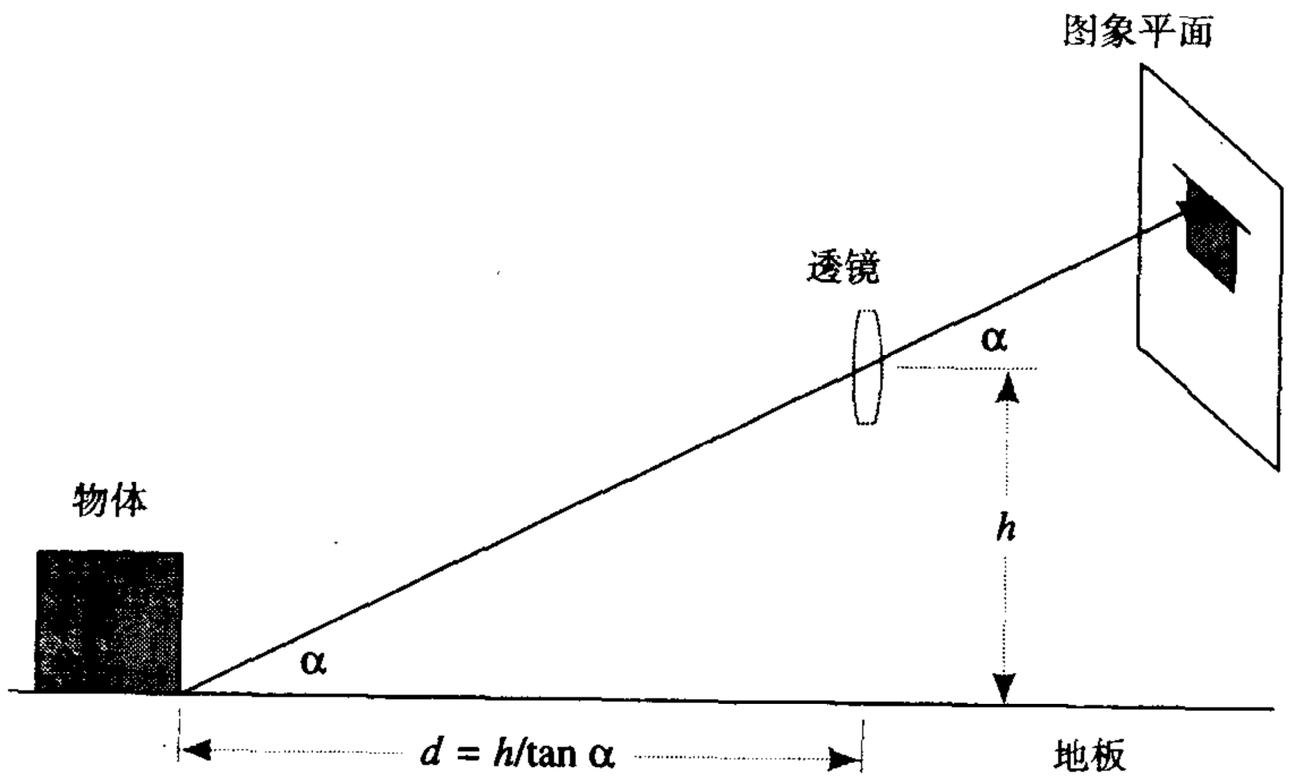


图6-20　从单个图象中计算深度

立体视觉也运用三角测量，其基本观点十分简单。我们来讨论一下图6-21所示的二维设置。这里有两个透镜，它们的中心被基线(*baseline*)*b*分开。与透镜距离为*d*的一个场景点通过这两个

透镜所产生的图象点如图所示。透镜到这些图象点的角度可用来计算d。若测量角度和基线的精确度不变，那么基线越长，物距越短，可得到的准确度越高。图6-21简化了实际情况，它假设光轴相互平行，两个图象平面位于同一平面，且场景点与两个平行光轴也位于同一平面。当把这些条件一般化时，会出现更复杂的几何运算，但三角测量这一大体观念不变(动物和一些机器人可把光轴旋转到场景中一个感兴趣的物体的一点上)。

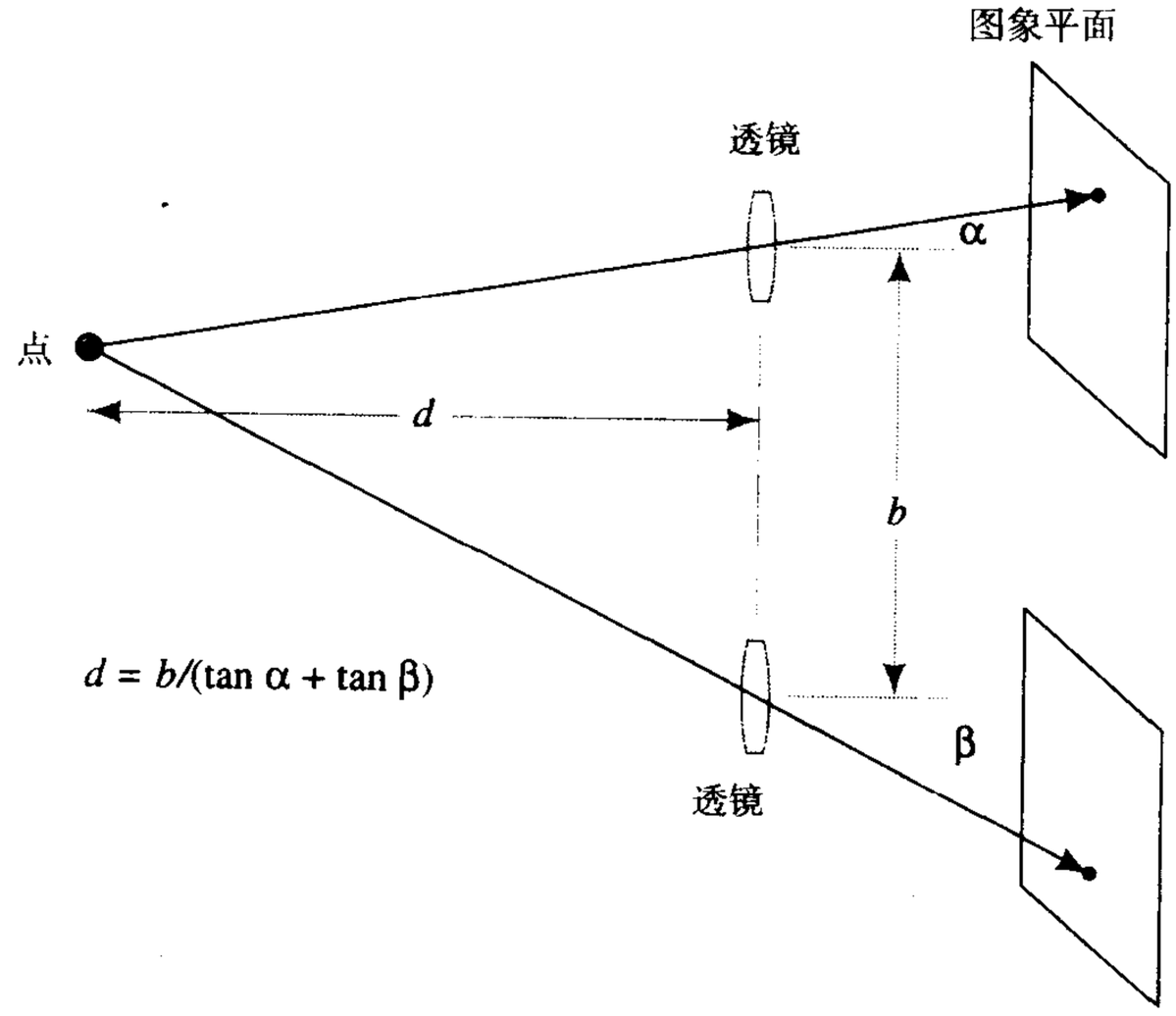


图6-21 立体视觉的三角计算

在立体视觉中，三角计算并不是最复杂的问题。在包含不止一个点的场景中（通常都是这样！），（为了计算出与此场景点的距离）必须判断两个图象中哪两个点与此场景点相对应。换句话说，若与一个场景点相对应的图象点落在一个图象所给定的象素中，我们就必须在另一个图象中找出与其相应的象素。这一过程被称为“对应问题（*correspondence problem*）”。本书篇幅有限，无法细致地讨论有关对应问题的技术，但将作些评价。

首先，几何分析表明，我们只需沿一维空间（而不是二维）搜寻一个图象中给定的一个象素在另一个图象中相应的象素。该一维空间称为“核线（*epipolar line*）”。一维搜寻可通过对两个图象中沿相应核线的亮度轮廓相互结合完成。其次，在众多应用中，我们无须找出图象点个体对之间的关联，而只需找出图象中更大的元素对（如线条[⊖]）之间的关联，而对每个图象的场景分析可提供相互关联的线条对的线索。有关立体视觉的精彩评论，请参阅[Nalwa 1993, 第7章]。

6.7 补充读物和讨论

开发ALVINN的小组又继续开发了其他的自动驾驶系统，有的技术与本章所讨论的内容有关（[Thorpe, et al. 1992]）。[Herbert, et al. 1997]也描述了一个在空旷地形中驾驶且配有立体和红外线传感器的自动交通工具的机动软件。

尽管许多应用要求从计算机视觉系统中获取整个场景的三维模型，但机器人通常只需要足

⊖ 或者是线条相交的图象点。

以引导其行动的信息。与力图计算完整场景模型的早期视觉研究不同，有些研究者却集中力量研究他们所认为的“意图视觉”的更相关的任务。[Horswill 1993]举出了一个简单的面向任务的机器人视觉系统的例子。这些面向任务的视觉系统足以满足汽车机器人的驾驶需要。请参阅[Churchland, Ramachandran, & Sejnowski 1994]。

通过对人类和动物视觉的心理学和神经生理学的研究，我们已经学到了有关视觉感知过程的很多知识。[Gibson 1950, Gibson 1979]还研究了其他一些现象，如变化中的视野是如何给出一个移动物体的环境信息的。[Julesz 1971]发现人类可利用对任一点的统计资料中的不连续点来感知深度。[Marr & Poggio 1997] 创建了一个较为合理的立体视觉神经模型。

对青蛙的实验表明[Letvinn, et al. 1959]，青蛙的视觉系统仅注意整体照明的变化（如一个渐渐逼近的庞大的动物的影子所引起的变化）和小而黑的物体（如苍蝇）的迅速运动。对猴子的实验表明[Hubel & Wiesel 1968]，它们视野中短的定向线段组才能刺激它们视觉皮层中的神经。对马蹄形蟹的实验表明，它们视觉系统中相毗邻的神经元彼此抑制(*lateral inhibition*)，从而使产生的效果类似后来我们所知的拉普拉斯过滤[Reichardt 1965]。有关生物学视觉的其他资料请参阅[Marr 1982, Hubel 1988]。

[Bhanu & Lee 1994]采用遗传技术来分割图象。

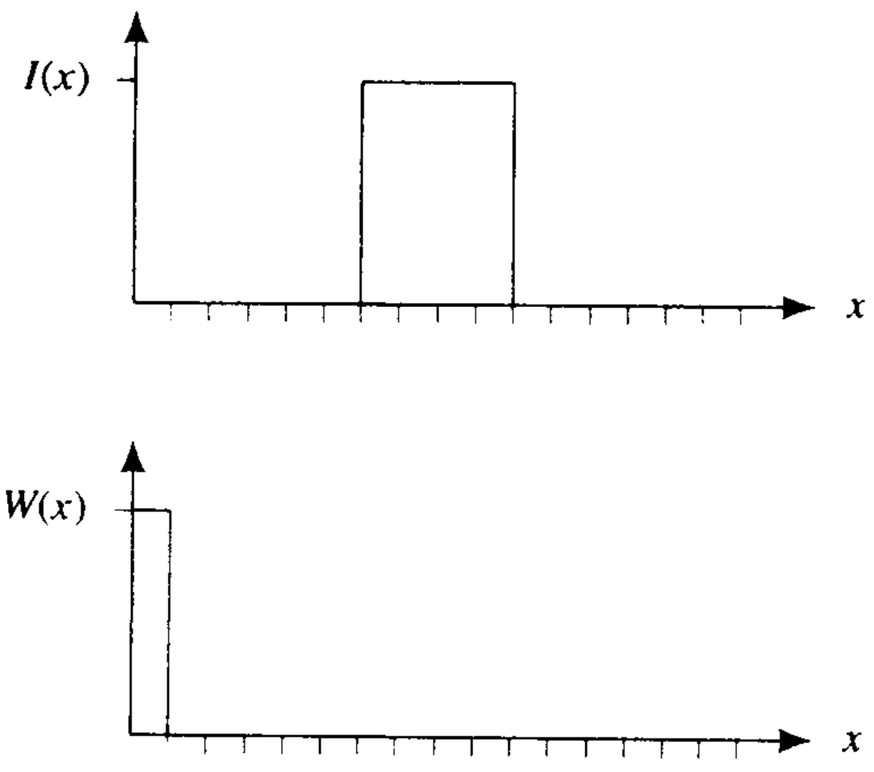
有关计算机视觉的主要会议有国际计算机视觉会议（ICCV）、欧洲计算机视觉会议（ECCV）以及计算机视觉与模式识别（CVPR）。权威的期刊是《International Journal of Computer Vision》。

计算机视觉的教材包括[Nalwa 1993, Horn 1986, Ballard & Brown 1982, Jain, Kasturi, & Schunck 1995, Faugeras 1993]。[Fischler & Firschein 1987]为重要的论文集。[Gregory 1996]通俗地说明了人类如何“看”。

习题

6.1 设计一个视觉系统来探测明亮背景中的黑色小物体。设这些物体的图象均为边长为5个象素的正方形。你的系统用来产生一个布尔特征，当单个这样的图象出现在一个100 × 100的图象数组的任一处时，其值为1；而出现在其他图象中时，其值为0。先用一个神经网络（有一个隐藏层面），再用一个特殊设计的加权函数进行卷积。描述一下你如何设计此系统。

6.2 一个一维图象函数$I(x)$，一个一维加权函数$W(x)$如下图所示。请画出$I(x) \bigstar W(x)$的图形。



6.3 一个简单的边缘查找（称为“Roberts cross”）用以下定义从图象$I(x, y)$中计算出另一个图象$I^*(x, y)$：

$$I^*(x,y) = \sqrt{(I(x,y) - I(x+\Delta, y+\Delta))^2 + (I(x, y+\Delta) - I(x+\Delta, y))^2}$$

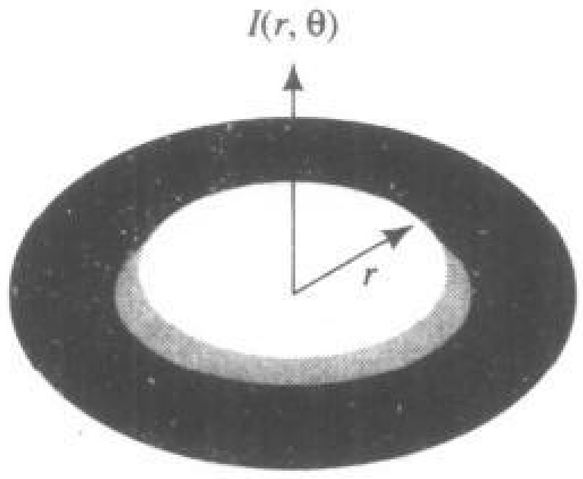
其中，Δ为一个单象素偏移量，并且采用正根。对小的象素，用微积分近似求出这个定义，再用此近似值算出$I^*(x,y)$，其中$I(x,y)$按极坐标来给定：

$$I(r,\theta) = 1 \text{ for } r < 9$$

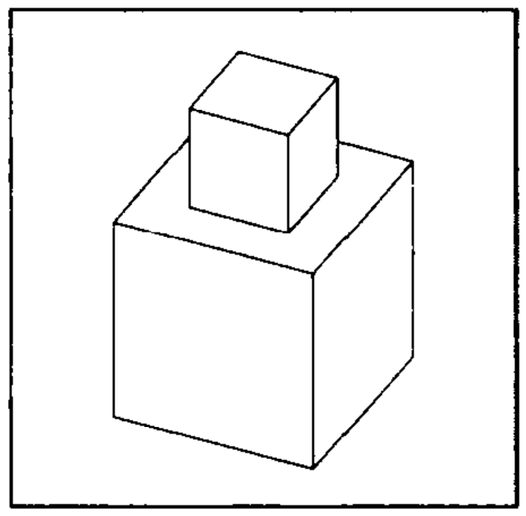
$$I(r,\theta) = 0 \text{ for } r > 10$$

$$I(r,\theta) = 10 - r \text{ for } 9 < r < 10$$

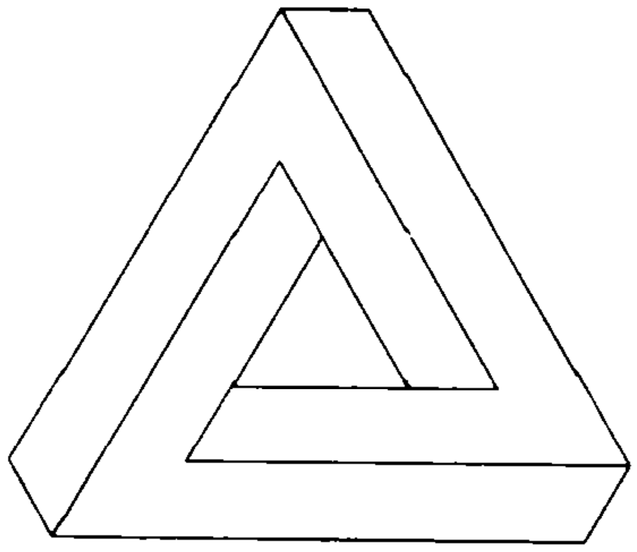
假设此圆形图象的半径为20（注意：用极坐标表示此图象有助于你得出最终解！）。图象亮度的草图如下：



6.4 复制下图并用6.5.1节中所讨论的顶点类别和线条标记给图中的线条作标记。若存在不止一种前后一致的标记方法，请列出你所想象的每一种，并描绘每一种的物理解释。



6.5 你能为下图中的著名的复杂图象（被称为“Penrose三角”）中的线条作出一种前后一致的标记吗？假设此图形被悬挂于空荡的空间。讨论一下“局部一致”与“全局一致”的关系。



第二部分　状态空间搜索

人无远虑，必有近忧。

——孔子

……[蚂蚁]知道必须采取一定的方案，但是它不知道如何去做。就像一个一只手拿着茶杯另一只手拿着三明治的人，想用一根火柴点燃一根香烟。这个人会想出一个办法，在拿起香烟和火柴前，先放下茶杯和三明治。而蚂蚁将会放下三明治，拿起火柴，然后放下火柴拿起香烟，又放下香烟拿起三明治，再放下茶杯拿起香烟，直到最后它放下三明治拿起火柴。这种方式倾向于依靠一系列的事件来达到目标，是一种不带任何思考的愚蠢做法……，Wart非常惊奇地观察了这个过程，它变得很着急，直到厌倦。他想问蚂蚁为什么不事先考虑一下呢……

——T. H. White《The Once and Future King》，第13章

第7章　能计划的agent

7.1　存储与计算

响应型agent(图2-2、图5-1和图5-3)的动作功能几乎没有做任何计算。从本质上讲，这些agent执行的动作或者由它们的设计者、或者通过学习、或者通过演化过程、或者是由以上几方面的组合而选择给它们的。这些动作能够通过表、产生规则描述给定特征向量动作的组合逻辑电路来实现。在计算机科学中，这种实现倾向于经典的时空权衡的“空间”一方。它们是基于空间或存储的实现——对设计者知识的汇编。

一个能在复杂环境下执行复杂任务的反应型机器需要大量（也许是无法计算的）的存储。而且，这样一个反应型机器的设计者需要有超人类的预见能力，要为该机器能遇到的所有可能情况预期一个合适的反应。这启发我们可以考虑用时间换取空间，用适应性代替显式的设计。首先，考虑反应型机器设计者必须做的一些计算的动作函数。这些计算当然会需要时间，但是它们将减少agent的存储要求和设计者的负担。

我们要考虑的一些计算是推测在任何给定的情况下，某些动作的可能结果。确实，一个有能力的反应型机器设计者必须把它的设计构架在这些预期结果的基础上，设计者（或者是进化，或是学习过程）必须指定这些计算是什么，但是计算（在需要它们时）程序通常比计算出结果所需要的空间要少得多。而且指定计算比作出所有可能情况的结果对设计者来说也更容易。也许最重要的是如果预期计算的结果能被自动地学习或演化，那么使用该结果的agent就能在那些连设计者都可能无法预见的情况下，也可以选择合适的动作来执行。

为了推测一个动作的结果，一个agent必须有一个自身所处环境的模型和一些结果模型，这些结果模型是agent对其环境模型的动作结果。因此，真正的动作只有在模拟环境是安全和有效时才会发生作用。

7.2　状态空间图

作为一个例子，让我们考虑一个有*A*、*B*、*C*三个玩具积木的网格空间，开始时，三个积木都在地板上。假如机器人的任务是把它们堆起来以便*A*在*B* 的上面，*B*在*C*的上面，*C*在地板上。当然，对我们来说采取什么动作是很明显的，但对机器人来说就不一样了。假定机器人能够对其每一个动作对环境的建模结果，它可以通过一对环境模型——一个代表动作执行前的环境状态，另一个代表动作执行后的环境状态——来建模。为了达到此目的，假设机器人能够把其上没有任何其他积木的积木 *x*移到另一个地方*y*，*y*或是地板或是其上没有其他积木的积木。可以通过一个模式的实例对这些动作建模，该模式表示为move　(*x*, *y*)，其中*x*可以是*A*、*B*或C中的任何一个，*y*可以是*A*、*B*、*C*和地板中的任何一个。我们也知道这个方法中的一些实例（如move(A,A)）是不可执行的动作 。这个方法的实例，如move(A,C)被称为算子(operator)。因此，算子是动作的模型。

用列表结构图标模型，可表示所有积木都在地面时能采用的所有动作的模型，参见图7-1[⇨]。（为了清楚起见，每种情况包括一个图示化的略图。图画对人是直观的，但是对处理列表的agent，列表可能更直观）。在这些结果中的（(*AB*)(*C*)）和（(*A*)(*BC*)）在某些方面似乎比其他更接近于我们的目标（(*ABC*)）。因此，仅考虑单个动作的预期结果，机器人可能宁愿执行动作move(A,B)和move(B,C)。当然，机器人还没有采取真正的动作。

在一个模拟环境中，只向前看一步常常就能产生有用的预期效果，但是多看几步，也许直到任务完成的所有步骤都看到后就会发现一些捷径，从而避免走弯路。跟踪几个可选动作序列结果的最有用的结构是有向图。一个agent通过它的动作产生的环境集合能用一个有向图表示。有向图的节点代表每个环境，弧代表算子（这里先使用一些不规范的图论术语；在本章的后面，将介绍一些我们要用到的图论概念的规范定义）。节点表示可以是基于图标的或基于特征的，尽管两种类型都会考虑，但先从图标表示开始。

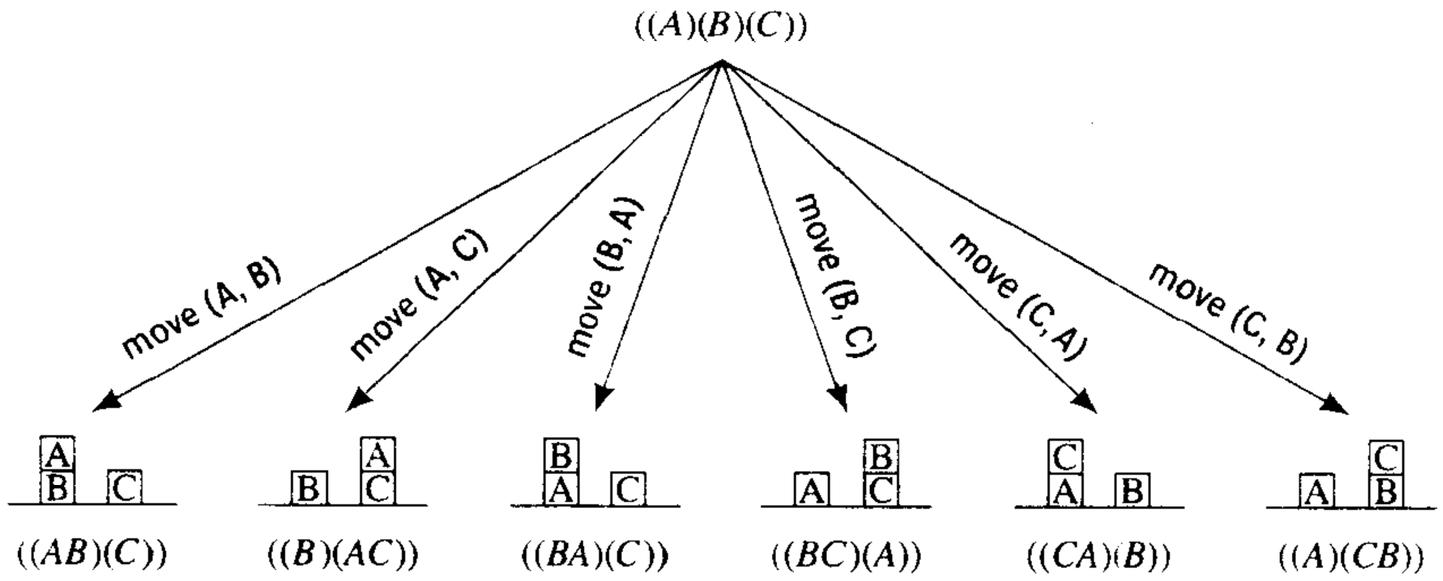


图7-1　移动一个积木的效果

如果大量可区分的环境状态足够小，那么一个代表所有可能动作和状态的图就能被显式地存储。例如，图7-2显示了所有的状态和相关移动来操作三个积木。这种环境模型和动作图被称为状态空间图(*state-space graph*)。需要注意的是每一个动作都是可逆的，为了使图不太混乱，仅仅标出了每对移动中的一个。从图中可以容易地看到，如果初始状态是（(*A*)(*B*)(*C*)），机器人的任务是获得（(*ABC*)）的状态，那么它应该执行{move(B,C),move(A,B)}的动作序列。

⇨　为简单起见，继续使用一种抽象的表示，在这里各个积木的水平位置是不相关的。因此，它们没被表示出来。

图结构表示可能世界的优点之一是图中的任何节点能代表一个目标状态——也许是由诸如人为的一些外部源指定的目标。这种任务灵活性可以和我们前面学习过的简单目的的agent形成对照。为了发现到达指定目标的一组动作，机器人只要能在图中发现一条代表初始状态节点到目标节点的路径就可以了。然后就能从该路径边上的标签读出到达目标的动作。例如，搭积木的机器人的任务是到达任务（（*CBA*））, 如果初始状态是（（*ABC*））, 动作序列将会是{move(A, floor), move(B,A), move(C, B)}。从图7-2中，可以通过视觉容易地找到路径，然而，为了发现路径，计算型agent要用各种图搜索过程。

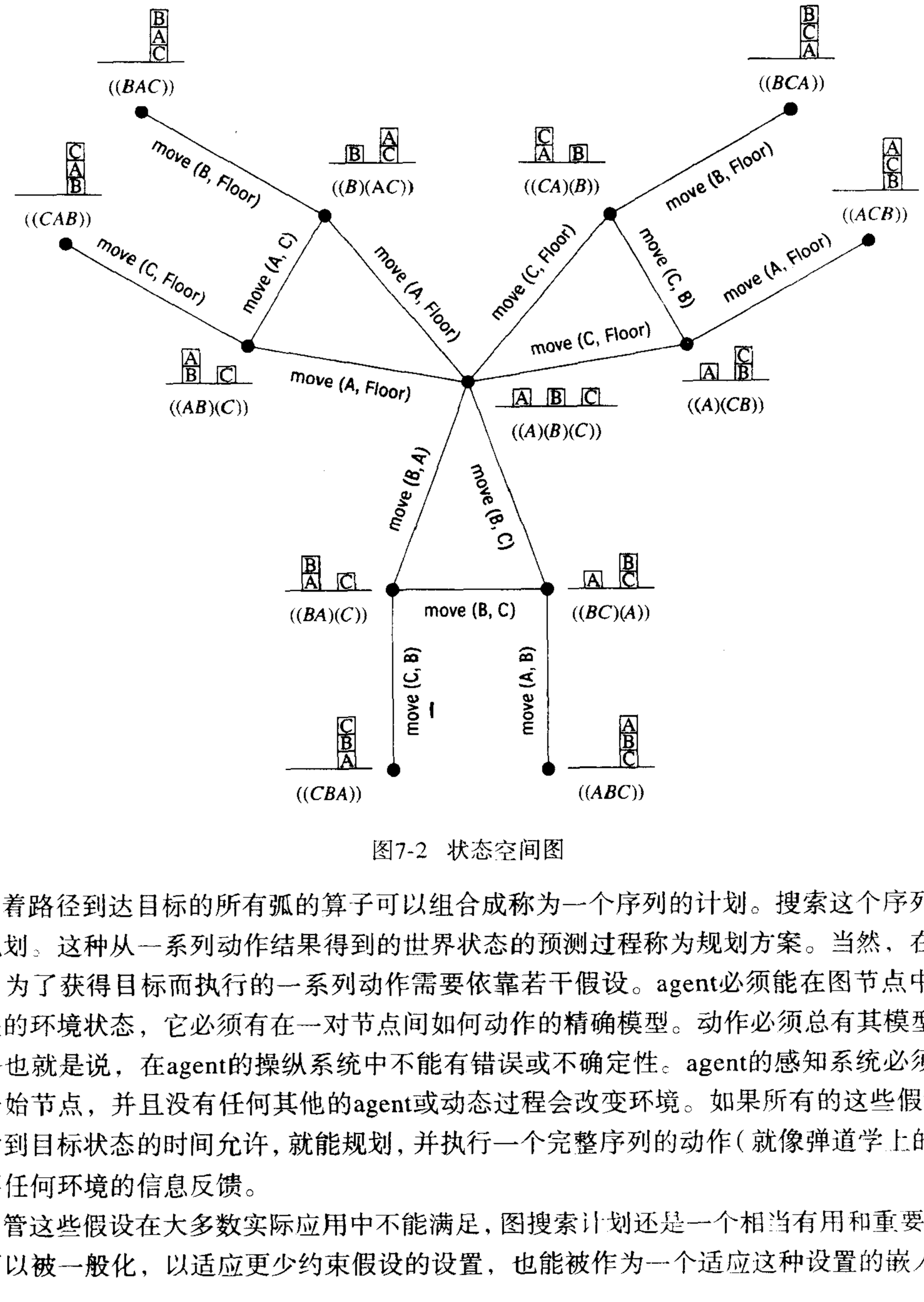


图7-2 状态空间图

顺着路径到达目标的所有弧的算子可以组合成称为一个序列的计划。搜索这个序列的过程称为规划。这种从一系列动作结果得到的世界状态的预测过程称为规划方案。当然，在这种方式下，为了获得目标而执行的一系列动作需要依靠若干假设。agent必须能在图节点中表示所有相关的环境状态，它必须有在一对节点间如何动作的精确模型。动作必须总有其模型化的结果——也就是说，在agent的操纵系统中不能有错误或不确定性。agent的感知系统必须精确地指定开始节点，并且没有任何其他的agent或动态过程会改变环境。如果所有的这些假设满足，且搜索到目标状态的时间允许，就能规划，并执行一个完整序列的动作（就像弹道学上的一样），不需要任何环境的信息反馈。

尽管这些假设在大多数实际应用中不能满足，图搜索计划还是一个相当有用和重要的思想。它既可以被一般化，以适应更少约束假设的设置，也能被作为一个适应这种设置的嵌入结构部

件。第10章会谈到这个主题。在本章中，通过描述一个相当简单的搜索过程来开始对图进行搜索处理，这适合于被搜索图能被显式存储的情况。在这种情况下，图一般都很小，因此不必太关心搜索方法的效率。在下面的两章中，将讨论应用到很大的图的图搜索方法，这些图只能被隐式地表示，而且必须被高效地搜索。

7.3　显式状态空间搜索

显式图搜索方法涉及到在图节点上传播“标记”。我们把开始节点标记为0,然后顺着图的边，连续传播更大的整数直至遇到目标节点。然后，顺着数字下降序列从目标点回溯到开始节点。顺着开始点到目标点路径上的动作序列就是获得目标应该采取的动作。这种方法需要$O(n)$步，n是图中的节点数目（如果只有一个目标节点，这个过程也可以从相反方向实现——从目标节点开始，直到某个整数遇到了开始节点）。搜索过程中放在节点上的数字可以作为该节点上的一种人工式势函数，并且开始节点有一个全局最小值。相反路径（从目标到开始）顺着这个函数的“梯度”下降。

从（(*BAC*)）转移到（(*ABC*)）的标记传播过程显示在图7-3中。这种方法和所谓的广度优先算法(*breadth-first search*)相一致，该算法由[Moore1959]首次提出。

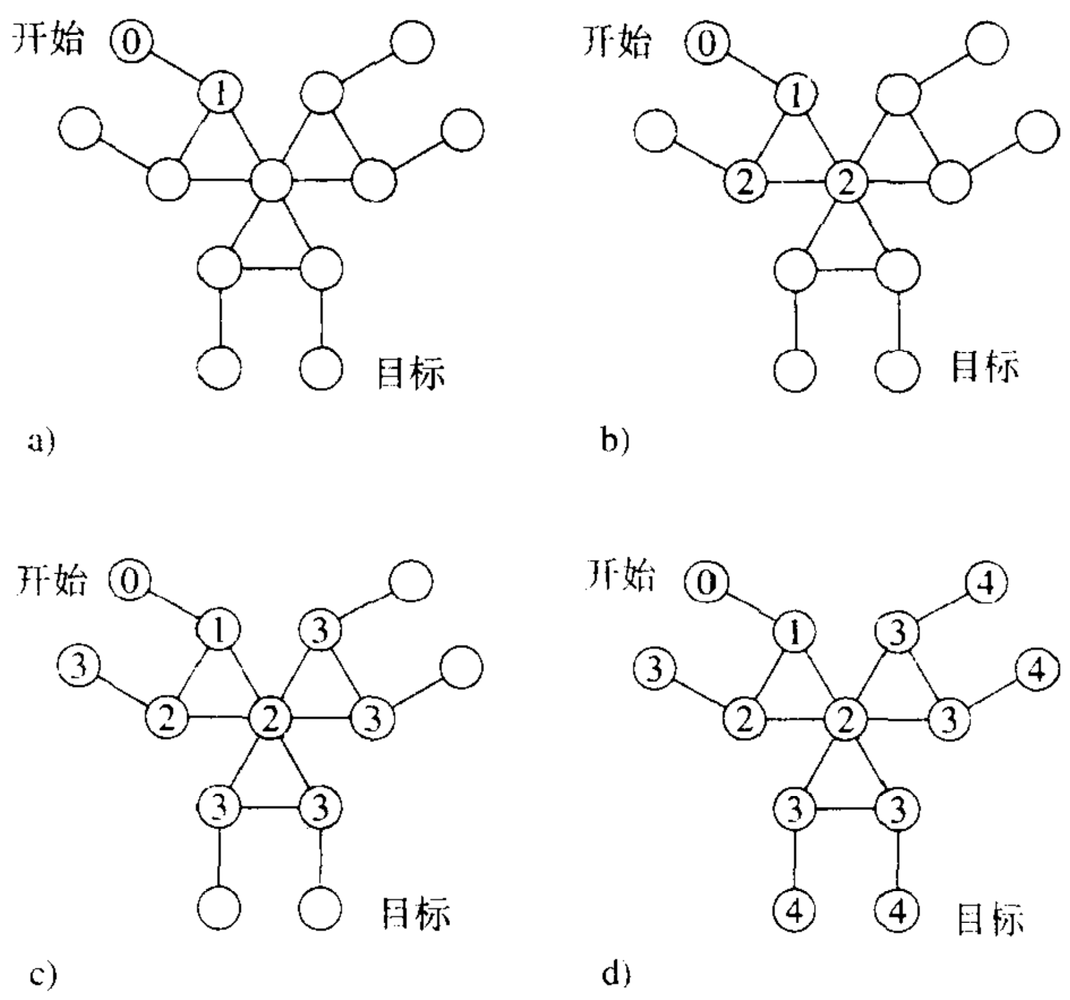


图7-3　搜索过程

把标记一个节点的后继节点的过程称为扩展(*expansion*)。扩展将标记放在所有已标记过的节点的未标记的相邻节点上。应将哪一个已标记但还没有扩展的节点作为下一个扩展点是一个很重要的效率问题。在广度优先搜索中，下一个要扩展的节点是其节点标识数不大于任何其他没有扩展的节点标识数的节点。也就是说，在扩展标识为j的节点之前，先要扩展标识为i的所有节点，条件是$i < j$。其他搜索算法有不同的节点扩展选择，这些将在后面详细讨论。

7.4　基于特征的状态空间

用图标模型标识节点来解释状态空间是相当直接的——可以很容易地使状态上的动作结果

形象化。可以定义一个有特征标识的节点图，但此时，我们需要一种方法来描述一个动作是如何影响特征的。STRIPS [Fikes & Nilsson 1971, Fikes, Hartn, & Nilsson 1972]系统使用一种这种技术。其基本思想是定义一个有三个列表的算子，第一个是前提列表，指定了值为1和0的特征，以便可以应用动作。第二个是删除列表，列出其值将从1变为0的特征。第三个是值从0变为1的加入列表。在删除和加入列表中没有明确提到的特征值是不变的。这三个列表构成了一个STRIPS算子——一个动作结果模型。关于基于空间的STRIPS算子的讨论将推迟到本书的后面，在那之前会介绍关于特征计算的更为有效的技术。

我们也能训练一个神经网络，从它在$t-1$时刻的值和该时刻采取的动作来学习预测一个特征向量在时刻t时的值[Jordan & Rumelhart 1992]。如图7-4所示。虽然只显示了一层网络，但也可以使用具有隐藏单元的中间层。在训练后，预测网络能被用来计算来源于各种动作的特征向量。这些向量反过来又能作为网络的新输入来预测两步以后的特征向量，等等。一个相似的过程（基于统计群技术）被用来控制一个移动机器人[Mahadevan 1992, Connell & Mahadevan 1993a]。因为在预测过程中不可避免的错误,用这种方法企图作出更大的移动都证实是徒劳的。

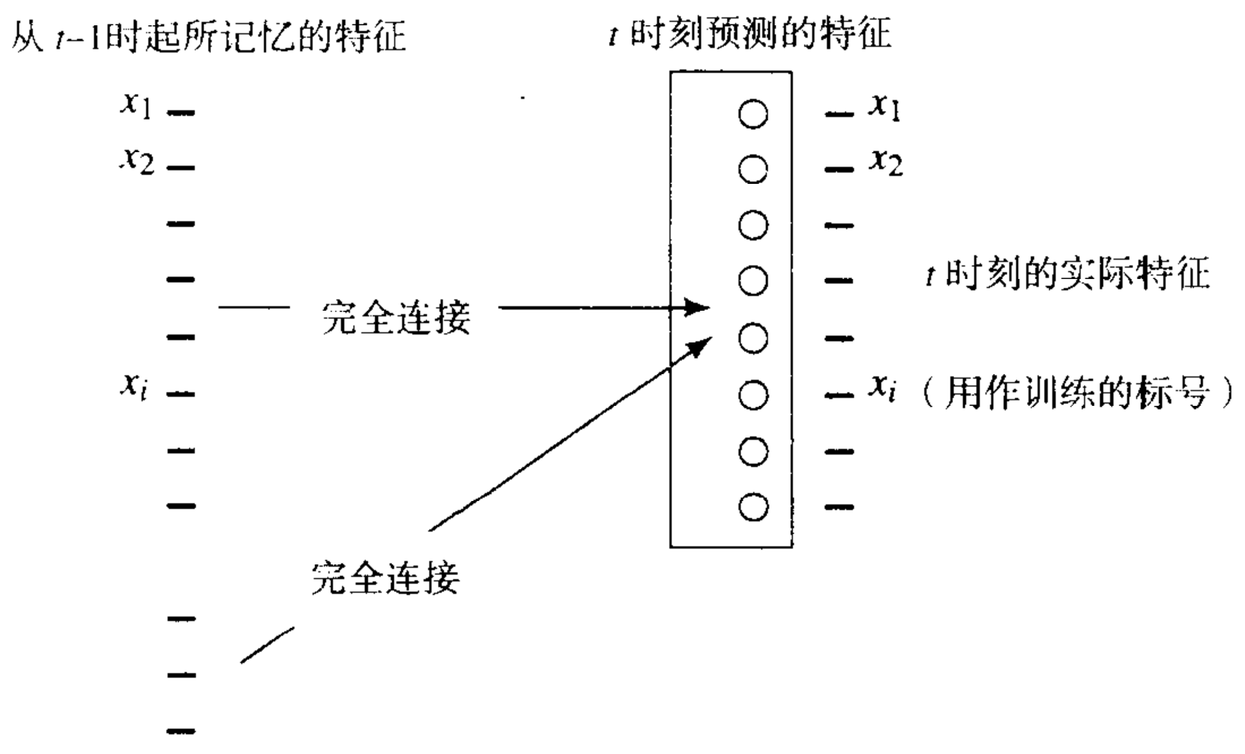


图7-4 在神经网络中预测一个特征向量

7.5 图记号

下一章我们将要处理大量的状态空间问题，在这之前，定义一些可以接受的、在讨论图和图搜索时要用到的术语是很有帮助的。在图7-5中，使用了一个图和树的示例来说明将要定义的术语。

一个图由一组节点（不一定是有限的）构成，节点对由弧连接，这些弧是从节点的一方指向另一方的有向弧，这样的图被称为有向图(*directed graph*)。为方便讨论，节点由环境状态模型标识，弧由动作名标识。如果弧是从节点n_i指向n_j，那么n_j就是n_i的后继（或者叫孩子），n_i是n_j的双亲。在我们研究的图中，一个节点只能有有限个后继。如果对节点中的每一个都可以是后继，在这种情况下，用边代替这一对节点。只包括边的图称为无向图。在随后有关图的表示中，弧带有箭头，而边没有。

一个（有限的）有向树是有向图的特殊情况。在有向树中（除了一个节点），每个节点只有一个父亲。没有父节点的节点被称为根节点。在树中没有后继的节点称为末端节点或叶节点。

我们称根节点的深度为0，树上任何其他节点的深度是其父节点的深度加1。一个（有根的）无向树（用边代替弧）是一个无向图，这里一对节点之间只有一条路径。

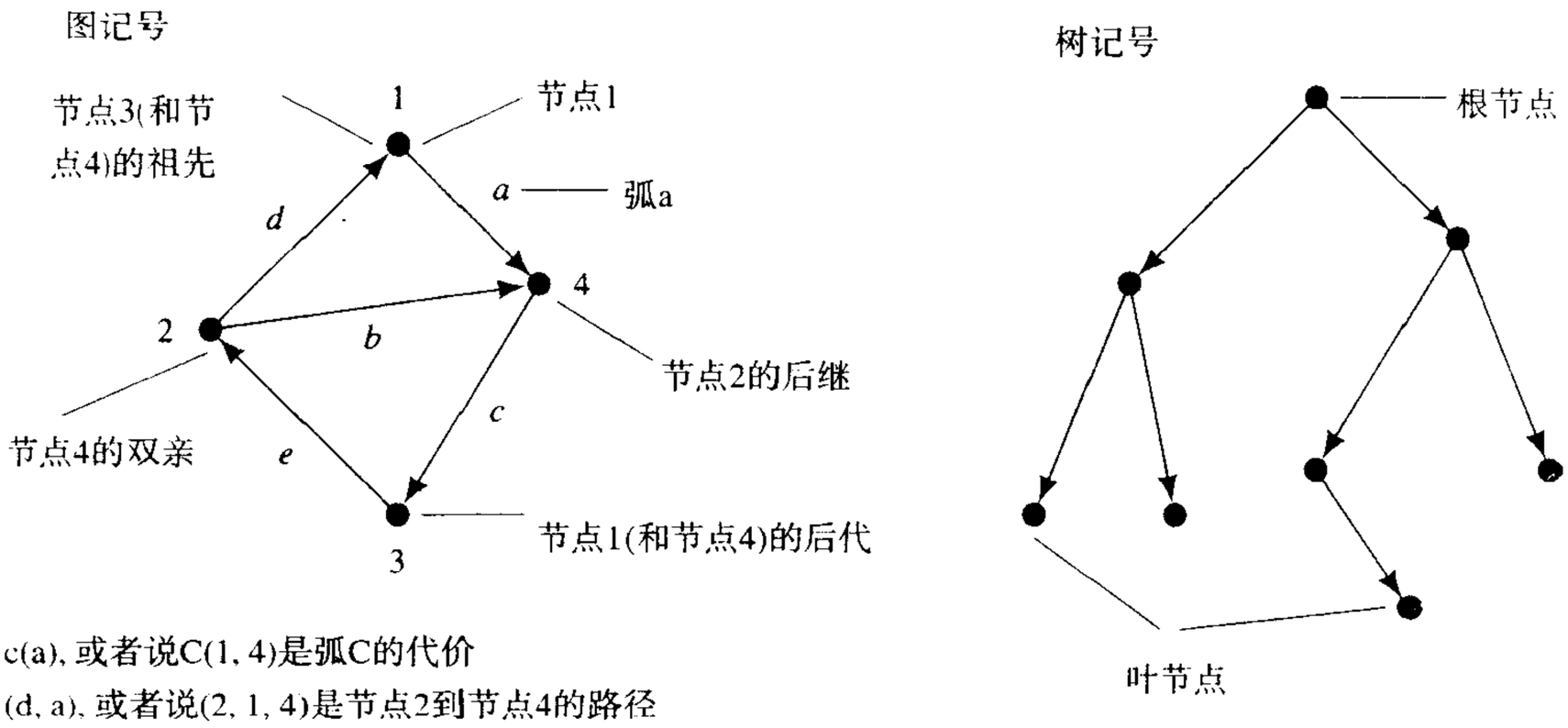


图7-5　图和树记号

在理论分析中，有一些树有这样的特征，即除了叶节点外，所有的节点都有相同数量b 个后继。在这种情况下，b被称为这个树的分枝因子。

一个节点序列（$n_1, n_2, \ldots, n_k$），n_{i+1}是n_i的后继，$i=1,2,\ldots,k-1$被称为从节点n_1到n_k的长度为k的一条路径（另外,我们可以把连接节点的弧序列定义为一条路径）。如果存在从节点n_i到n_j的路径，那么就说从n_i可以访问n_j。n_j就是n_i的后代，n_i是n_j的祖先。

通过分配一个正值给弧来代表执行相应的动作的代价常常是方便的。用符号$c(n_i, n_j)$（或者有时是$c(a)$）指示弧a（从n_i指向n_j）的代价。在后面的讨论中，假定这些代价比任意小的正数ϵ都大，这是很重要的。两个节点之间的路径代价是连接路径上节点间所有弧的代价的总和。在某些问题中，我们想找到两个节点之间的最小代价路径，这个路径被称为最佳路径。

最简单的问题是，在一个给定的代表初始状态的节点n_0和另一个代表其他状态的节点n_g之间寻找一条路径（也许有最小代价）。然而，经常遇到的问题是在节点n_0和一组节点中任意成员之间找到一条路径，这些节点都代表着满足一些目标条件的状态。称这个节点集为目标集，它里面的每个节点都是目标节点。

给定一个图，有时我们可能想找到从图中其他各个节点到某个节点n_0的一条路径，这样的路径集构成以n_0为根节点的图的一个生成树（注意，为了使一棵生成树是一棵树，对树的定义必须做一点改变，见习题7.2）。

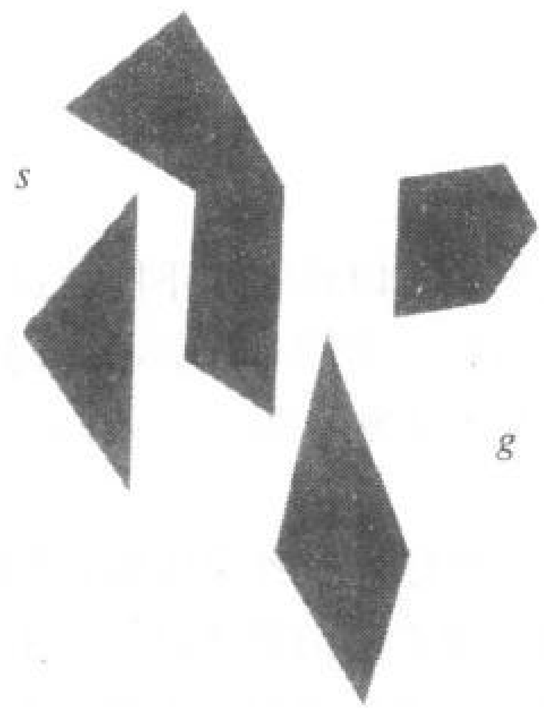
由于用图表示状态空间，因此有时可能交替地使用节点这个词，它既可以表示图中的一个节点，也可以代表环境状态，还可以表示那个节点的状态（数据结构或特征集）。同样，用单词“arc（弧）”代表图中的一条弧，或弧代表的动作，或对动作建模的算子（在状态描述中）。

7.6　补充读物和讨论

关于图的计算和算法的更多内容，可以参考[Cormen, Leiserson, & Rivest 1990, 第6章]。在图中寻找路径的任务可以引出很多问题，因此在这里和下面几章描述的方法还可应用在agent计划的研究范围之外的几个领域中。

习题

7.1 在下图由多边形障碍构成的二维空间中，一个机器人必须找到从开始点s到目标点g之间的最短路径。假定机器人是无穷小，路径可以与障碍相邻（或接触到）但不能交叉。



1) 复制上图，画出s和g之间的最短路径。
2) 在这个二维空间中，虽然有无限个点，但在任意的s和g点之间搜索一条最短路径时必须考虑的最小点集是什么？
3) 在刚找到的最小点集中给定一点，在相应的搜索图中描述产生该点后继的方法。图中s点的后继点是什么？

7.2 生成树：给定有向图S中的一个节点n，从S中的任何一个节点到n至少有一条路径，阐述一下以n为根的S的生成树的精确定义。然后，考虑一下什么是最小生成树。

7.3 “传教士和食人者”问题是人工智能中的一个著名问题。下面是该问题的描述：

三个传教士和三个食人者来到一条河边。河边只有一条每次最多可供两个人过河的小船。传教士如何用这条小船过河才能使河两边的食人者决不会超过传教士的数量？

指定状态描述的格式、开始状态和该问题的目标状态。画出整个状态描述图，标出其节点（只要画出“合法”的状态即可——也就是说，只要画出河两边食人者不超过传教士数量的状态就行了）。

7.4 参考习题5.3中的三圆盘汉诺塔问题。假定我们不知道移动圆盘的CW/CCW方法，但必须找到一个解决方法。用算子描述一下由方法move(x, y, z)给出的动作，x可以是圆盘D_1、D_2或D_3中的任何一个，y和 z可以是A、B、C三个桩中任何不同的一对。定义这个问题的状态描述，确定开始状态和目标状态，画出包含该问题的所有可能状态的完整搜索空间。用适当的算子标出弧（每个移动都可反向进行；你只要标出每对可逆移动中的一个即可）。

第8章 盲目搜索

8.1 用公式表示状态空间

很多实际问题的搜索空间是如此之大，以致它们不能通过显式图来表示。本章需要上一章所讲到的基本搜索过程的详细细节。首先，我们非常关心如何用公式来表示这些搜索问题；第二，我们必须找到隐式表示大的搜索图的方法；第三，我们需要用高效的算法来对这些大图进行搜索。

在一些规划的问题中，如堆积木，想像代表各种环境状态的数据结构和改变它们的动作是不难的，然而，找到可管理状态空间图的表示是很难的。为了做到这一点，必须仔细分析问题——考虑对称性，忽略不相关的细节，发现适当的抽象等。不幸的是，建立一个搜索问题的任务仍然是一个需要人工参与的大问题。

除了agent堆积木等问题外，数码问题也常被用来演示如何在状态空间中生成动作序列考虑到多样化，将在本章和下一章用到这种问题。一个典型的例子是15数码问题，它由放在个4×4的方阵中的15个数码构成，其中的一个单元是空的，它的周边单元中的数码可以移到该单元中。此问题的任务是找到一个数码移动序列使初始的无序数码转变为一些特殊的排列。8数码问题是一个简化版本，在一个3×3的方阵中有8个数码。假定该问题的目标是把数码从开始状态移动到目标状态，如图8-1所示。

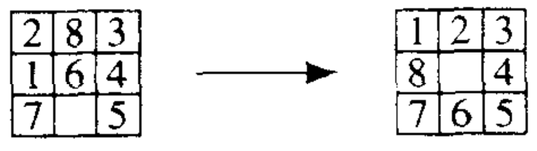


图8-1 8数码问题的开始和目标状态

在这个问题中，一个明显的图标状态描述是一个3×3的方阵，方阵的每个单元中包含1~8之间的数字或一个代表空格的符号。目标状态是图8-1中右边的方阵。状态间的移动就是把一个数码移到空的单元中。一般地讲，在构造一个问题的状态空间时我们有一些代表性的选择。在8数码问题中，我们可以想像有8×4个不同的移动，它们是：1上移、1下移、1左移、1右移、2上移、……，等等（当然在一个给定的状态中，并不是所有的这些移动都是可能的）。一个更精练的公式只有4个移动，即：空格左移、空格上移、空格右移和空格下移。一个给定的开始状态和一组可能的移动隐式地定义了从开始状态可到达的一个状态图。在8数码表示的状态空间中节点数是9！=362 880个（8数码状态空间刚好可以分成两个独立的图；一个图中的数码不能从其他图中的状态到达）。

8.2 隐式状态空间图的组成

从开始状态可以到达的状态空间图部分是通过开始状态描述和可能在任何状态下采用的动作结果描述来隐式表示的，因此，从理论上讲，可以把一个图的隐式表示转换为显式表示。为此，可以产生开始节点的所有后继节点（通过那个节点应用所有可能的算子），然后再生成所有后继节点的所有后继节点，等等。对那些太大以至不能显式表示的图，搜索过程只需要生成要求的状态空间，以发现到达目标的路径。过程在发现了一个可以接受的目标路径时终止。

有三个基本部分参与表示隐式状态空间图：

1) 一个标识开始节点的描述。这个描述是对环境初始状态建模的一些数据结构。

2) 把代表环境状态的状态描述转换成代表动作后状态描述的转换函数。这些函数也常被叫做算子。在agent问题中，它们是动作结果的模型。当一个算子应用到一个节点时，它产生该节点的一个后继节点。

3) 目标状态，可以是状态描述中的一个真假值函数，或者是和目标状态一致的状态描述的真实实例列表。

我们将学习两类主要的搜索过程。其中之一，我们没有指定问题的任何推理信息，例如要搜索这一部分而不是另一部分，就像到目前为止的只要发现一条到目标的路径即可。这种过程被称为是盲目的。另一种，我们指定了要解决问题的信息以帮助集中搜索。这个过程叫启发式(*heuristic*)搜索[⊖]。本章将讨论盲目搜索过程，下一章再讨论启发式搜索过程。

8.3 广度优先搜索

盲目搜索过程只把算子应用到节点，它没有使用问题领域的任何特殊知识（除了关于什么

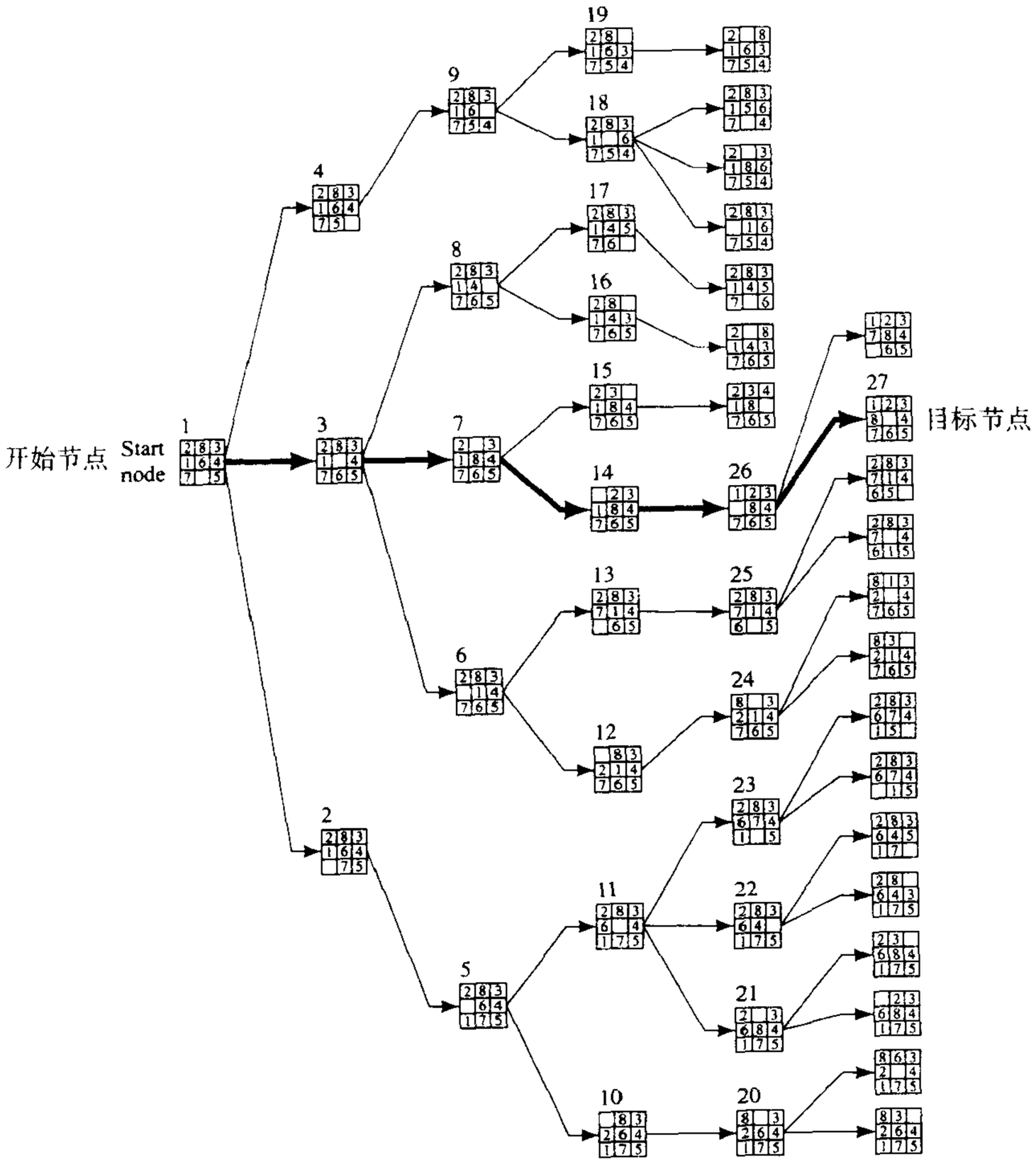


图8-2 8数码问题的广度优先搜索

⊖ heuristic 一词来源于希腊语heuriskein，意思是“发现”。外来语Eureka!（“我已发现了它!”）和heuristic同源。

动作是合法的知识外）。最简单的盲目搜索过程就是广度优先搜索(*breadth-first search*)。该过程把所有的算子应用到开始节点以产生一个显式的状态空间图，再把所有可能的算子应用到开始节点的所有直接后继，再到后继的后继，等等。搜索过程一律从开始节点向外扩展。由于每一步将所有可能的算子应用到一个节点，因此可把它们组成一个叫后继函数(*successor function*)的函数。当把后继函数应用到一个节点时，产生一个节点集，该节点集就是把所有能应用到那个节点的算子应用到该节点而产生的。一个节点的后继函数的每一次应用称为节点的扩展(*expanding*)。

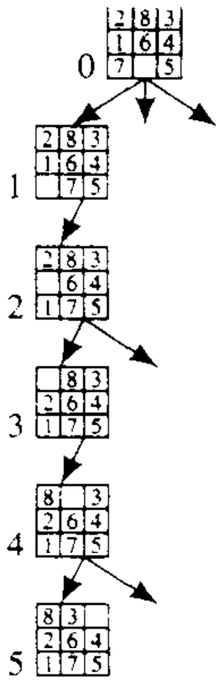
图8-2显示了8数码问题经广度优先搜索所产生的节点集。标出开始和目标节点，节点扩展顺序由一个数字表示在节点的旁边。相同深度的节点按照一些固定的顺序扩展。在扩展一个节点时，按空格左移、上移、右移和下移的顺序应用算子。尽管每个移动都是可逆的，但删去了从后继到双亲的弧。解答路径在图中用黑线表示。如我们将看到的一样，广度优先搜索的一个特征是：当发现目标节点时，我们已经找到了到达目标的一条最短路径。然而广度优先搜索的一个缺点是它要求产生和存储一个大小是最浅目标节点深度的指数的树。

相同代价搜索(*uniform-cost search*)[Dijkstra 1959]是广度优先搜索的一种变体，在该方法中，节点从开始节点顺着代价等高点向外扩展，而不是顺着相同深度等高线。如果图中所有弧的代价相同（比如说等于1），那么相同代价搜索就和广度优先搜索一致。反过来，相同代价搜索可以看作是下一章要讲的启发式搜索的一个特殊情况。广度优先和相同代价搜索方法的简要描述只给出了它们的主要思想，但是要解决其他复杂的情况则需要技术改进。比如一个节点扩展产生的节点在前面的搜索过程中已经到达过。将在下一章介绍了更一般的算法后再讨论这些方法。

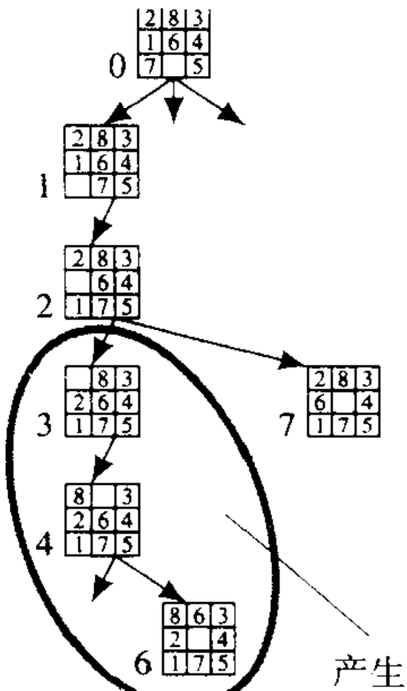
8.4　深度优先或回溯搜索

深度优先搜索一次对节点应用一个算子以产生该节点的一个后继。每一个节点都留下一个标记，用来指示如果需要时所必需的附加算子。对每一个节点，必须有一个决策来决定哪个算子先用，哪个次之等等。只要一个后继产生，它的下一个后继就会被生成，一直向下传下去，等等。为了防止从开始节点的搜索过程深度太深，需要一个深度约束(*depth bound*)。超过这个深度约束时不再产生后继（假定没有任何目标节点超过这个深度约束）。这种限制允许我们忽略搜索图的一部分，这些部分已经确定不能高效地到达目标节点。

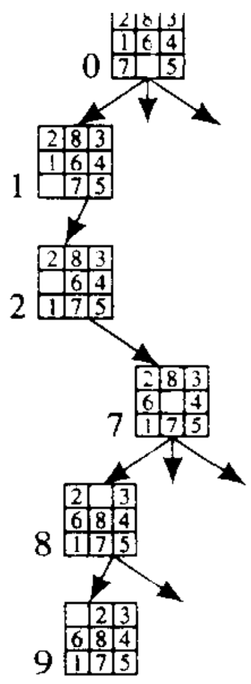
用8数码和深度约束为5为例说明这个过程。我们再次以下列顺序：空格左移、上移、右移和下移来应用算子，忽略从后继到双亲的弧。在图8-3a中给出了开始的几个节点，每个节点左边的数字是该节点产生的顺序。也留下了一些弧指示那些还没有完全展开的节点。在节点5，我们到达了深度约束点但还没有到达目标，因此，考虑下一个最近产生的但还没有完全展开的节点4，生成它的另一个后继节点6（见图8-3b）。这时，可以抛弃节点5，因为我们不会再在它下面产生节点。节点6也在深度约束点且不是目标节点，因此我们考虑下一个最近产生、但没有完全展开的节点2，所产生的节点7——抛弃节点3和它的所有后继，因为我们不再产生它们下面的任何节点。返回节点2是年历回溯(*chronological backtracking*)的一个例子。当再次到达深度约束时，产生了图8-3c。这时没有到达目标节点，我们就生成节点8的另一个后继，它也不是目标节点。因此，我们抛弃节点8和它的后继，回溯产生节点7的另一个后继。继续这个过程，最终产生图8-4。



a)



b)



c)

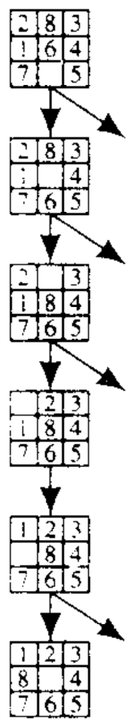


图8-3 深度优先搜索最初几个节点的产生

图8-4 深度优先搜索中到达目标节点时的图

深度优先算法使我们只保存搜索树的一部分，它由当前正在搜索的路径和指示该路径上还没有完全展开的节点标志构成。因此，深度优先搜索的存储器要求是深度约束的线性函数。深度优先搜索的一个缺点是当发现目标时，我们不能保证找到的路径是最短长度。另一个缺点是如果只有一个很浅的目标，且该目标位于搜索过程的后部时，也必须浏览大部分搜索空间。

8.5 迭代加深

一种叫做迭代加深(*iterative deepening*)[Korf 1985, Stickel & Tyson 1985]的技术既能满足深度优先搜索的线性存储要求，同时又能保证发现一个最小深度的目标节点（如果一个目标节点能被发现）。在迭代加深方法中，连续的深度优先搜索被引入，每一个的深度约束逐次加1，直到一个目标节点被发现。图8-5说明迭代加深是如何在一个简单树上工作的。令人惊奇的是，由迭代加深搜索扩展产生的节点数并不比广度优先搜索产生的多很多。我们可以计算一个有相同分枝因子的树在最坏搜索情况下所生产的节点数，这个树的最浅目标节点在深度d处，并且是该深度最后一个生产的节点。由广度优先搜索扩展产生的节点数是：

$$N_{\mathrm{bf}} = 1 + b + b^2 + \cdots + b^d = \frac{b^{d+1} - 1}{b - 1}$$

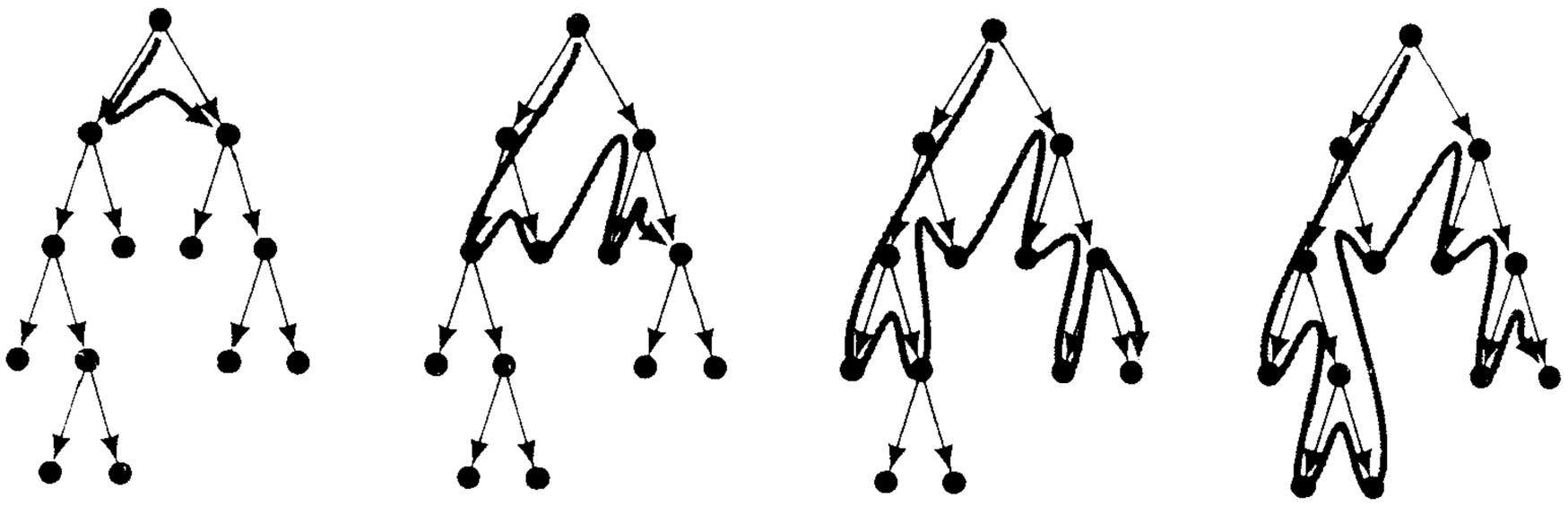


图8-5 迭代加深搜索过程

为了计算由迭代加深扩展来的节点数，我们首先指出，由一个完全的深度优先搜索搜索到第*j*级而扩展成的节点数是：

$$N_{\mathrm{df}_j} = \frac{b^{j+1} - 1}{b - 1}$$

在最坏的情况下，对深度为*d*的目标进行迭代加深搜索时必须实施分裂，完全进行深度优先搜索直到深度为*d*。由所有这些搜索扩展而来的节点数之和是：

$$\begin{aligned} N_{\mathrm{id}} &= \sum_{j=0}^{d} \frac{b^{j+1} - 1}{b - 1} \\ &= \frac{1}{b-1}\left[b\left(\sum_{j=0}^{d} b^j\right) - \sum_{j=0}^{d} 1\right] \\ &= \frac{1}{b-1}\left[b\left(\frac{b^{d+1} - 1}{b - 1}\right) - (d+1)\right] \end{aligned}$$

简化上式可得到

$$N_{\mathrm{id}} = \frac{b^{d+2} - 2b - bd + d + 1}{(b-1)^2}$$

对大的*d*而言，$N_{\mathrm{id}}/N_{\mathrm{bf}}$的比率是$b/(b-1)$。对一个分枝因子为10的深度目标，迭代加深搜索只比广度优先搜索多了大约11%的扩展节点。

另外，当有很多目标节点时，一种称为迭代加宽（*iterative broadening*）的技术也很有用。[Ginsberg & Harvey 1992, Harvey] 1994]描述了这个方法。

8.6　补充读物和讨论

对年代回溯方法有各种改进方法，如[Stallman & Sussman 1977]的Dependency-directed回溯，[Gaschnig 1979] 的backjumping和 [Ginsberg 1993]的dynamic backtracking。[Ginsberg 1993]比较了这些方法，并指出了动态回溯的优越性。这些扩展的回溯技术常用于约束满足问题中（第11章将讲到）。

习题

8.1 在水壶问题中，给定3公升和4公升的水壶各一个，分别叫做3和4 。开始时，3和4都是空的。两个壶都可以从水龙头*T*中灌水，也可以把水从两个壶中倒到排水沟*D*中。水可以从一个壶倒入另一个壶中。没有其他的度量器。我们要找到一套方法让4中的水刚好是2公升。[不要害怕！这里就有一个办法：（a）把3用水龙头灌满；(b)把3中的水倒入4中；(c)把3用水龙头灌满；(d)用3中的水把4倒满；(e)倒掉4中的水；(f)把3中的水倒入4中。]

1) 给水壶问题建一个状态空间搜索公式：

a) 把初始的图标状态描述作为一个数据结构。

b) 给定状态图的一个目标条件作为对数据结构的一些测试。

c) 命名状态算子，准确描述每个算子对状态图的操作。

2) 画出所有不同状态空间节点的图；这些节点在开始节点的三步移动之内，用状态描

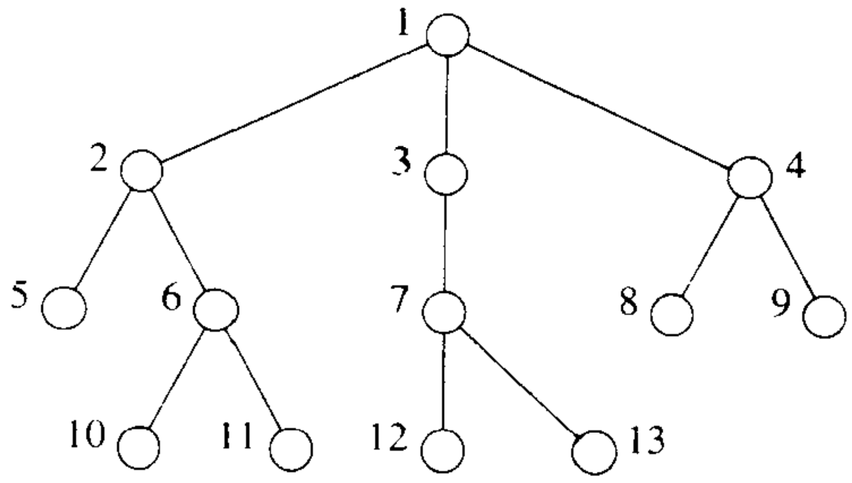
述标出每一个节点，画出图中到每个节点的至少一条路径。用合适的算子标出每一条弧。除了这些节点，画出解决方法中所有的节点和弧（被正确标识）。

8.2 列出下图中树的节点访问序列以满足下面的三个搜索策略(在所有情况中都选择最左分枝优先访问)：

1) 深度优先搜索；

2) 深度优先迭代加深搜索（每个迭代深度加1）；

3) 广度优先搜索。



8.3 考虑一棵有限树，深度为d，分枝因子为b(一棵只有一个根节点的树，其深度为0；一棵由一个根节点和它的b个后继组成的树，其深度为1；等等)。假设，最浅目标节点在深度g ($g \leqslant d$)。

1) 深度约束为d、深度优先搜索所产生的，节点的最大和最小数目分别是什么？

2) 广度优先搜索时产生的节点的最大和最小数目是什么？

3) 由深度优先迭代加深搜索产生的节点的最大和最小数目是什么？(假定初始深度限制是1，如果当前深度限制下没找到目标节点，将深度加1继续搜索)。

8.4 假设在广度优先搜索中每秒产生10 000个节点，存储每个节点需要100个字节。对一个深度为d、分枝因子为5的树进行完全广度优先搜索，需要多少空间和时间？把这些值放在一个表中(你可用一个大概的数字，时间最好表示为小时、天、月、年等等)。

8.5 假定我们正在对一个分枝因子为b的树进行搜索，然而我们不知道我们真的在搜索一个树，因此考虑检查每一个产生的状态描述看它是否匹配以前产生的状态描述。在一棵深度为d的树中进行搜索，将会有多少个这种检查？

第9章　启发式搜索

9.1　使用评估函数

除了搜索过程不是从开始节点统一向外扩展外，本章描述的搜索过程有点像广度优先搜索，不同的是，它会优先顺着有启发性和具有特定信息的节点搜索下去，这些节点可能是到达目标的最好路径。我们称这个过程为最优（*best-first*）或启发式搜索。下面是其基本思想：

1) 假定有一个启发式(评估)函数 $\hat{f}$，可以帮助确定下一个要扩展的最优节点(给*f*加一个“帽子”的原因后面就会清楚，称 $\hat{f}$ 为“f-hat”)。我们采用一个约定，即 $\hat{f}$ 的值小表示找到了好的节点。这个函数基于指定问题域的信息，它是状态描述的一个实数值函数。

2) 下一个要扩展的节点*n*是 $\hat{f}$ (*n*)值最小的节点(本章假定节点扩展产生一个节点的所有后继)。

3) 当下一个要扩展的节点是目标节点时过程终止。

我们常常可以为最优搜索指定好的评估函数。如在8数码问题中，可以用不正确位置的数字个数作为状态描述好坏的一个度量：

$\hat{f}$ (*n*) = 位置不正确的数字个数(和目标相比)

在搜索过程中采用这个启发式函数将产生图9-1所示的图，每个节点的数值是该节点的 $\hat{f}$ 值。

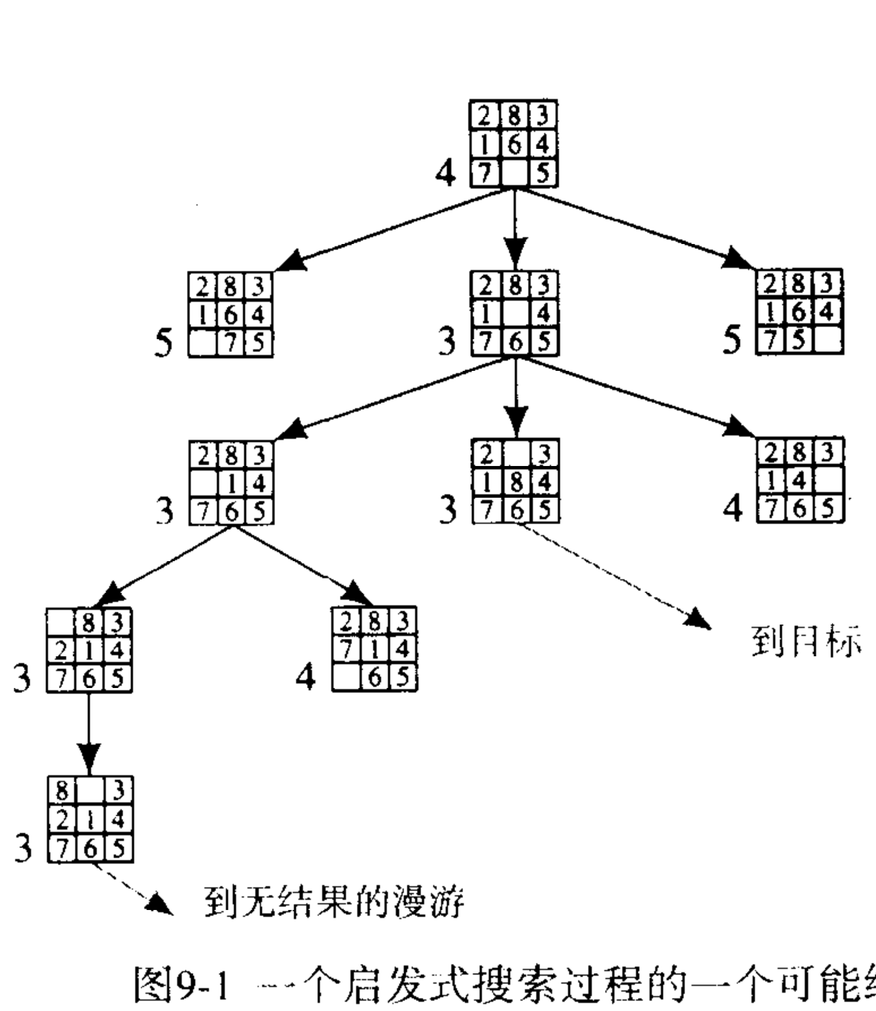


图9-1　一个启发式搜索过程的一个可能结果

这个例子表明，在搜索过程中我们需要偏向有利于回溯到早期路径的搜索(为了避免由于过分的优化试探而陷入“花园小径”)。因此我们加了一个“深度因子”给 $\hat{f}$：$\hat{f}(n) = \hat{g}(n) + \hat{h}(n)$，$\hat{g}(n)$ 是对图中节点*n*的“深度”估计(即从开始节点到*n*的最短路径长度)，$\hat{h}$ (*n*)是对节点*n*的启发或评估。像前面一样，如果 $\hat{h}(n)$ = 不正确位置的数字个数(和目标相比)，$\hat{g}(n)$= 搜索图中节点*n*的深度，我们可以得到图9-2。在这个图中，把 $\hat{g}(n)$ 和 $\hat{h}(n)$ 的值写在每个节点的旁边，在这种情况

下，可以看到搜索相当直接地朝着目标进行(除了用圆圈标注的节点外)。

这些例子提出了两个重要的问题。第一，我们如何为最优搜索决定评估函数？第二，最优搜索的特性是什么？它能找到到达目标节点的好路径吗？本章的结尾将会提到关于评估函数选择的一些指导，第10章将会解释关于自动学习评估函数的一些方法。本章主要讨论最优搜索的形式表示。作为特殊的例子，我开发了一个包括最优搜索版本的一般图搜索算法（为了更详细地了解启发式搜索，可以参考引用的文章和Pearl写的书[pearl 1984]）。

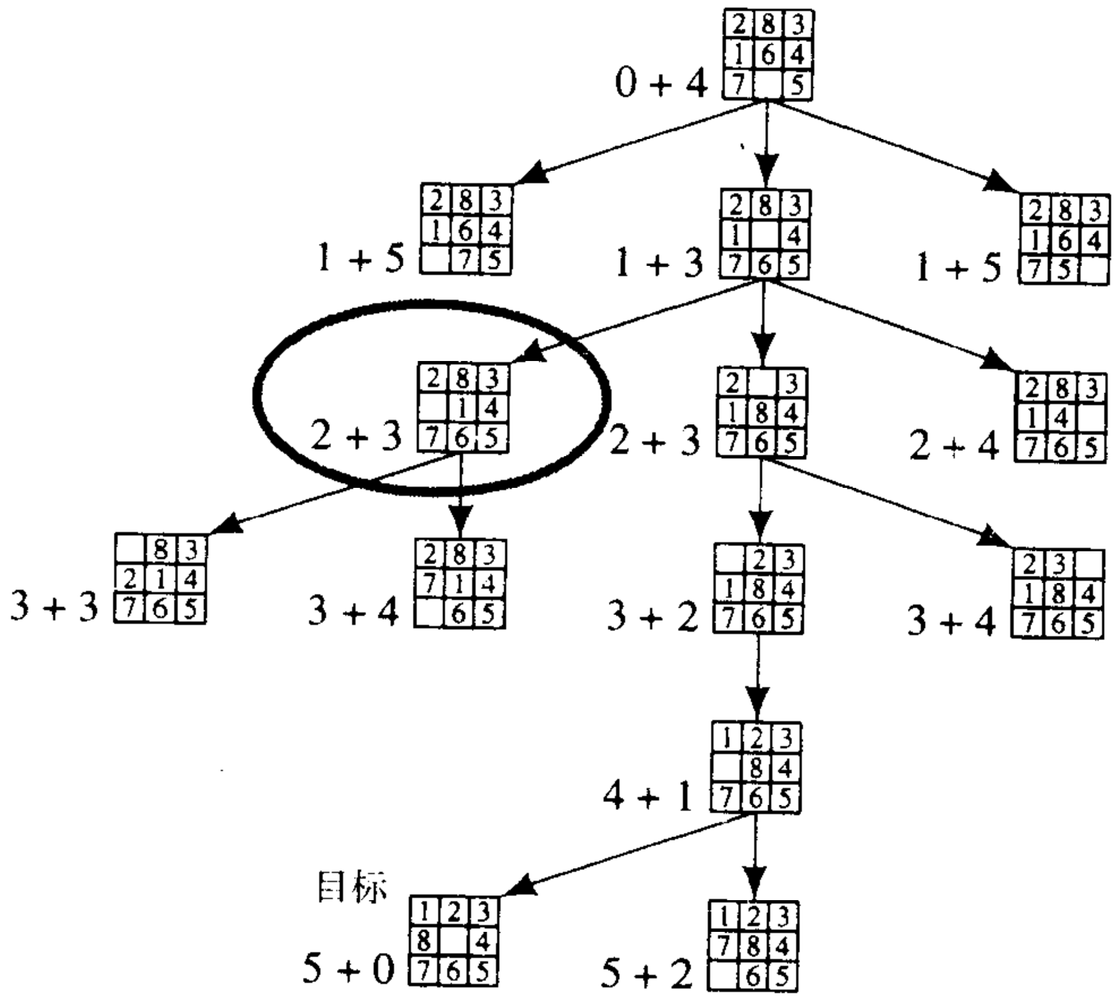


图9-2 使用 $\hat{f}(n)=\hat{g}(n)+\hat{h}(n)$ 的启发式搜索

9.2 一个通用的图搜索算法

为了更准确地解释本章的启发式搜索过程，这里提出一个通用的图搜索算法，它允许各种用户——偏爱启发式的或盲目的，进行定制。我把这个算法叫做图搜索（GRAPHSEARCH）。下面是（第一个版本）它的定义。

GRAPHSEACH

1) 生成一个仅包含开始节点n_0的搜索树Tr。把n_0放在一个称为*OPEN*的有序列表中。

2) 生成一个初始值为空的列表*CLOSED*。

3) 如果*OPEN*为空，则失败并退出。

4) 选出*OPEN*中的第一个节点，并将它从*OPEN*中移出，放入*CLOSED*中。称该节点为n。

5) 如果n是目标节点，顺着*Tr*中的弧从n回溯到n_0找到一条路径，获得解决方案，则成功退出（弧在第6步产生）。

6) 扩展节点n，生成n的后继节点集M。通过在*Tr*中建立从n到M中每个成员的弧生成n的后继。

7) 按照任意的模式或启发式方式对列表*OPEN*重新排序。

8) 返回步骤3。

这个算法可用来执行最优搜索、广度优先搜索或深度优先搜索。在广度优先搜索中，新节点只要放在*OPEN*的尾部即可（先进先出，FIFO），节点不用重排。在深度优先搜索中，新节

点放在*OPEN*的开始（后进先出，LIFO）。在最优（也叫启发式）搜索中，按照节点的启发式方式来重排*OPEN*。

9.2.1　算法A*

用最优搜索算法详细说明GRAPHSEARCH。最优搜索算法根据函数 $\hat{f}$ 的增加值，(在第7步)重排*OPEN*中的节点，如8数码问题。把GRAPHSEACH的这种算法称为算法A*。下面将会看到，定义使A*执行广度搜索或相同代价搜索的函数 $\hat{f}$ 是可行的。为了确定要用的函数 $\hat{f}$ 族，必须先介绍一些其他符号。

设$h(n)$ =节点n和目标节点（遍及所有可能的目标节点以及从n到它们的所有可能路径）之间的最小代价路径的实际代价。

设$g(n)$ = 从开始节点n_0到节点n的一个最小代价路径的代价。

那么$f(n) = g(n) + h(n)$就是从n_0到目标节点并且经过节点n的最小代价路径的代价。注意$f(n_0) = h(n_0)$是从n_0到目标节点的一个（不受限的）最小代价路径的代价。

对每个节点n，设 $\hat{h}(n)$（启发因子）是$h(n)$的某个估计，$\hat{g}(n)$（深度因子）是由A*发现的到节点n的最小代价路径的代价。在算法A*中，我们用 $\hat{f} = \hat{h} + \hat{g}$。注意，如果算法A*中的 $\hat{h}$ 恒等于0，就成为相同代价搜索。这些定义示例在图9-3中。

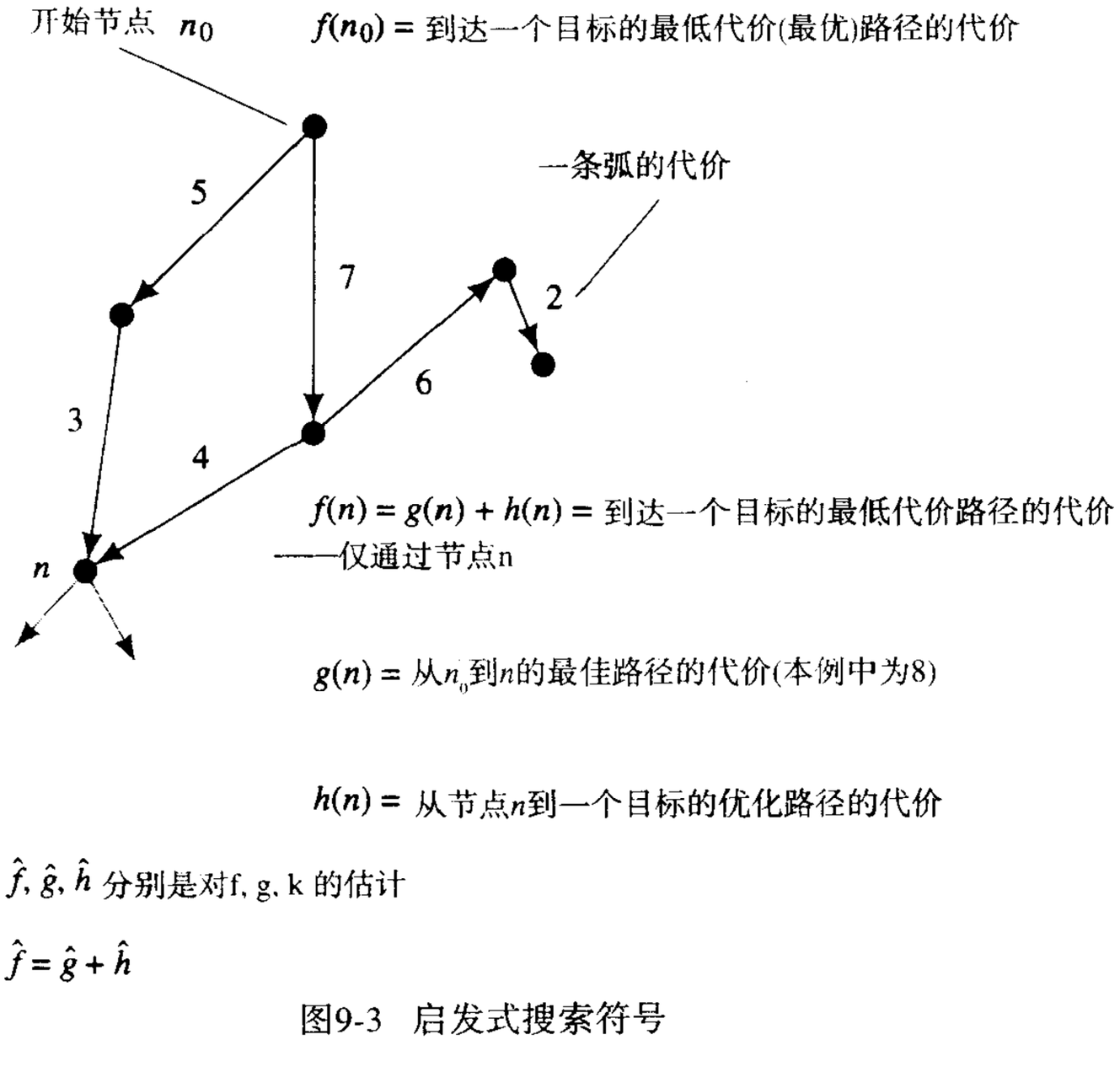


图9-3　启发式搜索符号

图9-2所示的8数码搜索过程是A*应用的一个例子。在这里，我们假定了单位弧代价，因此$g(n)$就是图中节点n的深度。尽管稍微有一点复杂，但在算法A*的定义中忽略了它们。如果要搜索的隐式图不是一棵树会怎样呢？也就是说，假如有超过一个动作序列能从开始状态到达相同的环境状态。例如，8数码隐式图显然就不是一棵树，因为动作是可逆的——即任何节点n的每一个后继都可以使n作为它的一个后继。在建立8数码搜索树中忽略了这些循环。在那种情况下，

它们容易被忽略，只要在节点的后继中不包括它的双亲就行了。把GRAPHSEARCH 中的第6步改为：

6) 扩展节点n，生成后继集合 $\mathcal{M}$，n的双亲不能在 $\mathcal{M}$中。通过在Tr中建立从n到 $\mathcal{M}$中每个成员的弧生成n的后继。

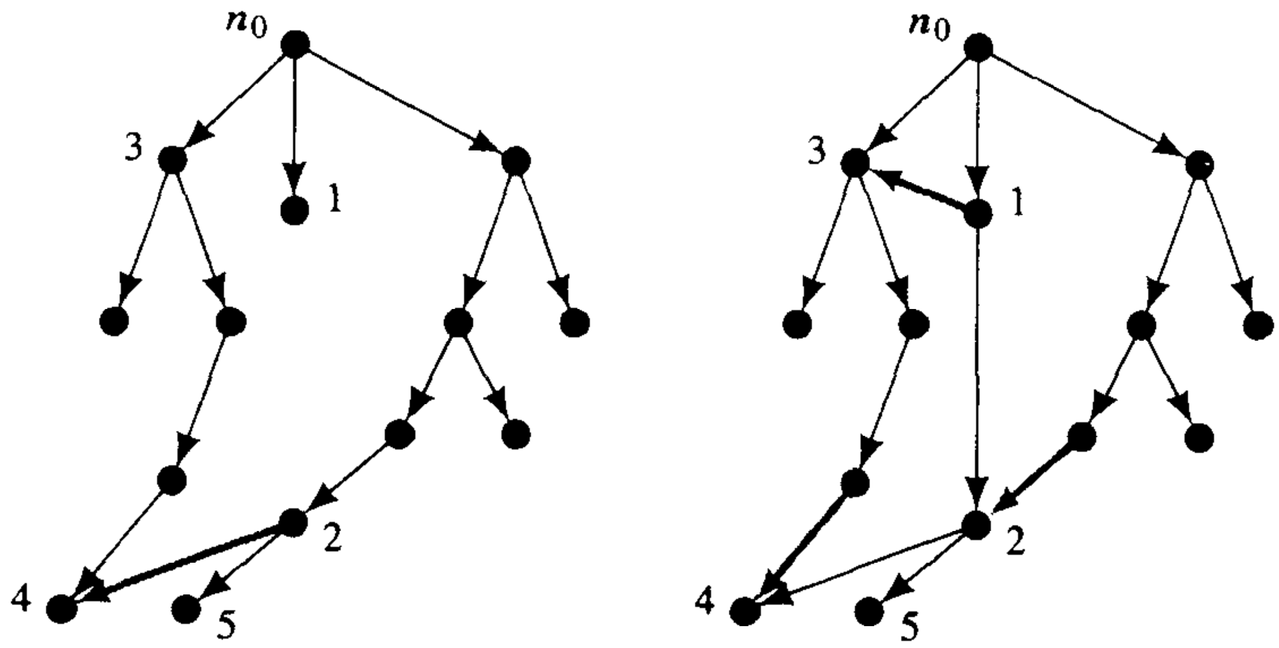
考虑到更长的循环，我们把6改为：

6) 扩展节点n，生成后继集合 $\mathcal{M}$，n的祖先不能在 $\mathcal{M}$中。通过在Tr中建立从n到 $\mathcal{M}$中每个成员的弧生成n的后继。

当然，为了检查这些更长的循环，我们要检查标识节点n的祖先的数据结构。对复杂的数据结构，这一步增加了算法的复杂度。

修改第6步的算法在搜索目标路径时避免了再循环，但是仍有可能通过不同的路径到达相同的环境状态。解决这种问题的一个方法就是简单地忽略它。也就是说，算法不检查 $\mathcal{M}$中的一个节点是否已经在*OPEN*或*CLOSED*中。因此算法对那种可能性就健忘了，以致它可能通过不同的路径到达相同的节点。这个“相同节点”在Tr中重复多次，重复的次数是算法发现到达该节点的不同路径数目。如果Tr中的两个节点被同样的数据结构标识，那么在它们下面有相同的子树。也就是说，算法会重复搜索结果。后面我们将看到，对 $\hat{f}$ 施加适当的条件，当**A***首次扩展Tr中的节点n时，它已经找到了到达节点n并且有最小值 $\hat{f}$ 的一条路径。

为了防止在前述条件不成立下 $\hat{f}$ 的重复搜索，需要对算法**A***进行一些修改。因为搜索可以顺着不同的路径到达相同的节点，因此算法**A***会产生一个搜索图G。G是**A***扩展开始节点、开始节点的后继等等而产生的节点和弧的结构。**A***也包含一个搜索树Tr。Tr是G的一个子图，它是到达搜索图中所有节点的最优（最小代价）路径生成树。图9-4显示了一个单位弧代价的图。图9-4a表示一个早期搜索阶段，搜索图的搜索树部分用黑色的弧表示，灰色的弧不在搜索树中。注意图9-4a中的节点4有两条路径可以到达；两条路径都在搜索图中，但只有一条在搜索树中。我们保留搜索图是因为后面的搜索过程可能会发现更短的路径，这些路径使用了在早期搜索图中、但不在早期搜索树中的弧。例如图9-4b，扩展节点1会发现到达节点2和它的子孙（包括节点4）的一条更短路径，因此搜索树会相应地改变。



a) b)

图9-4 搜索图和搜索过程产生的树

为了完整起见，阐述一下包含搜索图的**A***算法。然而，应该注意，很少需要这个算法，因为我们常常对 $\hat{f}$ 施加一定的条件以保证当算法**A***扩展一个节点时，它已经发现了到达该节

点的最小代价路径。

算法A*：

1) 生成一个只包含开始节点n_0的搜索图G，把n_0放在一个叫*OPEN*的列表上。
2) 生成一个列表*CLOSED*，它的初始值为空。
3) 如果*OPEN*为空，则失败退出。
4) 选择*OPEN*上的第一个节点，把它从*OPEN*中移入*CLOSED*，称该节点为n。
5) 如果n是目标节点，顺着G中，从n到n_0的指针找到一条路径，获得解决方案，成功退出（该指针定义了一个搜索树，在第7步建立）。
6) 扩展节点n，生成其后继节点集 $\mathcal{M}$，在G中，n的祖先不能在 $\mathcal{M}$ 中。在G中安置 $\mathcal{M}$ 的成员，使它们成为n的后继。
7) 从 $\mathcal{M}$ 的每一个不在G中的成员建立一个指向n的指针（例如，既不在*OPEN*中，也不在*CLOSED*中）。把 $\mathcal{M}$ 的这些成员加到*OPEN*中。对 $\mathcal{M}$ 的每一个已在*OPEN* 中或*CLOSED*中的成员m，如果到目前为止找到的到达m的最好路径通过n，就把它的指针指向n。对已在*CLOSE*中的 $\mathcal{M}$ 的每一个成员，重定向它在G中的每一个后继，以使它们顺着到目前为止发现的最好路径指向它们的祖先。
8) 按递增 $\hat{f}$ 值，重排*OPEN*（相同最小 $\hat{f}$ 值可根据搜索树中的最深节点来解决）。
9) 返回第3步。

在第7步中，如果搜索过程发现一条路径到达一个节点的代价比现存的路径代价低，我们就要重定向指向该节点的指针。已经在*CLOSED*中的节点子孙的重定向保存了后面的搜索结果，但是可能需要指数级的计算代价。因此，第7步常常不会实现。随着搜索的向前推进，其中有些指针最终将会被重定向。

9.2.2 A*的可接纳性

对图和 $\hat{h}$ 施加一些条件可以保证应用到图的算法A*总能找到最小代价路径。图的条件是：

1) 图中每个节点的后继是有限的（如果有的话）；
2) 图中所有弧的代价都大于某个正数ϵ。

$\hat{h}$ 的条件是：

3) 对搜索图中的所有节点n，$\hat{h}(n) \leqslant h(n)$。也就是说，$\hat{h}$ 决不会超过实际值h的估计。（这样的 $\hat{h}$ 函数有时被称为优化概算机）

一般来说，找到满足这个低约束条件的 $\hat{h}$ 并不困难。例如，在城市路由问题中，从城市n到目标的直线距离就是从n到目标节点的最佳路径距离的一个最低约束。在8数码问题中，不在正确位置的数码个数就是到达目标步数的一个最小约束。

用这三个约束条件，只要存在到达目标的路径，算法A*可以保证找到一条到达目标的最佳路径。把这个结果作为一个定理（下面给出该定理的一个新版证明，原始证明参见[Hart，Nilsson，& Raphael 1968]。

定理9.1　如果图和 $\hat{h}$ 具有上述的稳定条件，而且从开始节点n_0到目标节点有一条有限代价的路径，那么算法A*保证终止于到达目标的一条最小代价路径。

证明：证明的主要部分是下面的重要引理。为了获得关于A*为什么能保证发现最优路径

的直觉知识，完全理解这个引理是重要的。

引理9.1 在A*终止前的每一步，总是有一个节点n^*，它*OPEN*上有下面的特性：

1) n^*在到达目标的一条最佳路径上。

2) A*已经发现了到达n^*的一条最佳路径。

3) $\hat{f}(n^*) \leqslant f(n_0)$

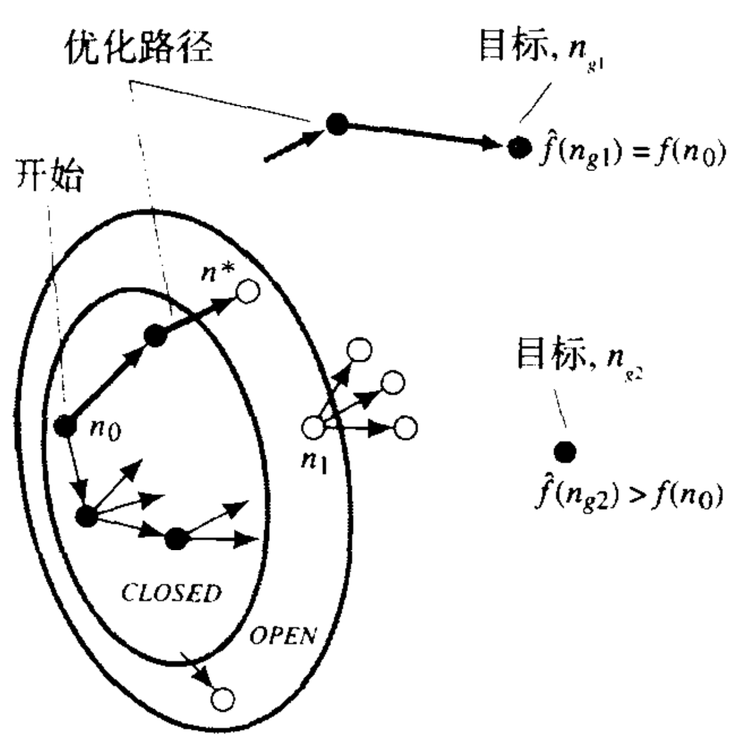
引理证明：为了证明A*每一步保证引理结论，只要证明（1）在算法开始时结论正确；（2）如果一个节点扩展前结论正确，那么节点扩展后结论继续正确。我们称这种证明方法为数学归纳法。下面是证明过程：

基本条件：在搜索开始时（当节点n_0准备扩展时），n_0在*OPEN*上，它是到达目标的一条最佳路径，A*已经发现了这条路径，而且，因为 $\hat{f}(n_0) = \hat{h}(n_0) \leqslant f(n_0)$，故 $\hat{f}(n_0) \leqslant f(n_0)$。

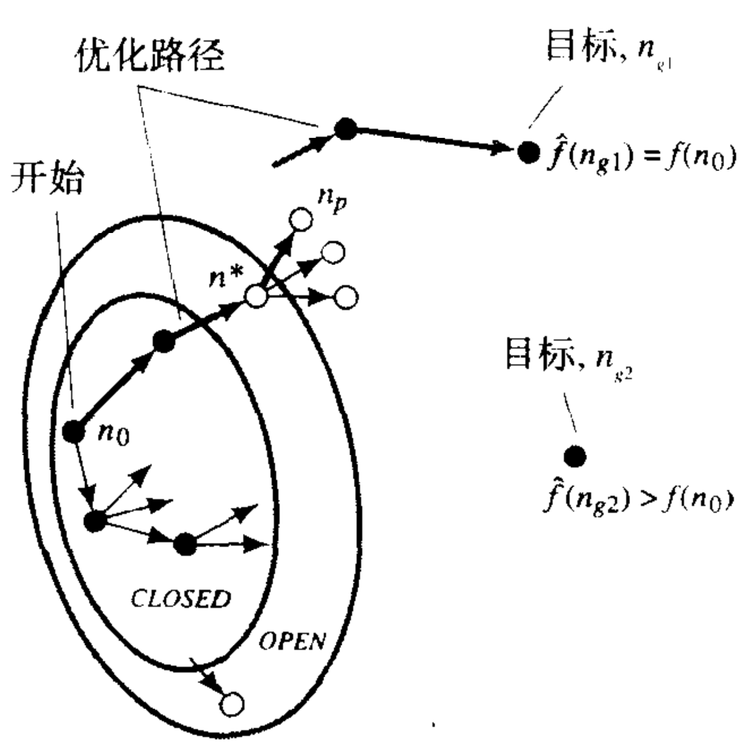
因此，在该阶段，节点n_0可以是引理中的节点n^*。

归纳步骤：假设在节点$m(m \geqslant 0)$扩展时引理的结论是正确的，（用这个假设）证明在节点$m+1$扩展时引理是正确的。参考图9-5将有助于我们证明归纳步骤。

设n^*是m个节点扩展后，A*发现的一个最佳路径上的假设节点，它在*OPEN*上。如果在第（$m+1$）步，n^*没有被选择扩展（图9-5a），n^*的属性和以前相同，因此证明了归纳步骤。如果n^*被选为扩展点（图9-5b），它的所有新后继将被放在*OPEN*上，它们中至少有一个n_p，将会在到达目标的最优路径上（由于假定一个最优路径通过n^*，它必须继续通过它的一个后继）。故A*找到了到达n_p的一条最佳路径，因为如果有到达n_p的一条更好路径，那么这条路径也是到达目标的更好路径（这和我们的假设没有比通过n^*到达目标的路径更好的路径相矛盾）。这样，让n_p成为第$(m+1)$步的新n^*，除了 $\hat{f}(n^*) \leqslant f(n_0)$外，我们已经证明了归纳步骤。



a) A* 选择节点n_1作为要扩展的第$(m+1)$个节点



b) A* 选择节点n^*作为要扩展的第$(m+1)$个节点

图9-5 当第$(m+1)$个节点要扩展时的情况

现在证明 $\hat{f}(n^*) \leqslant f(n_0)$。对在一条最佳路径上的节点$n^*$，A*已经找到了一条到达$n^*$的最佳路径，我们有：

$\hat{f}(n) = \hat{g}(n) + \hat{h}(n)$，根据定义

$\leqslant g(n^*) + h(n^*)$　　因为 $\hat{g}(n^*) = g(n^*)$，$\hat{h}(n^*) \leqslant h(n^*)$

$\leqslant f(n^*)$　　根据定义$g(n^*) + h(n^*) = f(n^*)$

$\leqslant f(n_0)$　　由于n^*在一条最佳路径上，故$f(n^*) = f(n)$

这就完成了引理的证明。

继续证明定理，我们首先证明如果存在可以到达的目标，**A***必须终止，然后证明**A***终止于找到一条到达目标的最佳路径。

- **A***必须终止：假如它不会终止。在这种情况下，**A***始终不断地扩展*OPEN*上的节点，最终它将在搜索树上扩展比任何预设的深度约束更深的节点，因为我们已经假设正在搜索的图有有限个分枝因子。由于每个弧的代价都比$\epsilon>0$大，故在*OPEN*中的所有节点的 $\hat{g}$ (因此 $\hat{f}$)值将最终超过$f(n_0)$。这将和引理产生矛盾。
- **A***终止于一条最优路径：**A***只能在第3步（*OPEN*为空）或第5步（在一个目标节点）终止。一个第3步的终止只能出现在不包含任何目标节点的有限图中，只要有一个目标节点，定理都声称找到到达目标的一条最佳路径。因此，**A***终止于找到一个目标节点。假如它终止于发现了一个非最佳目标n_{g2}，$f(n_{g2})>f(n_0)$，当有一个最佳目标n_{g1}时，$n_{g1}\neq n_{g2}$，$f(n_{g1})=f(n_0)$（见图9-5）。在n_{g2}终止时，$\hat{f}(n_{g2})\geq f(n_{g2})>f(n_0)$。但是在**A***选择$n_{g2}$进行扩展之前，根据引理在*OPEN*上有一个节点$n^*$，它在最优路径上，且 $\hat{f}(n^*)\leq f(n_0)$。因此，**A***不可能选择n_{g2}进行扩展，因为**A***总是选择有最小 $\hat{f}$ 值的节点进行扩展，而 $\hat{f}(n^*)\leq f(n_0)<f(n_{g2})$。

定理到此完全证明完毕。我们称任何保证能找到一条到达目标的最佳路径的算法是可接纳的。因此，在定理的三个条件下，**A***是可接纳的。延伸开来，任何没有高过h的函数 $\hat{h}$ 都是可接纳的。从现在起，当谈到关于**A***的应用时，都假设定理的三个条件是满足的。

如果**A***的两个版本$\mathbf{A}_1$*和$\mathbf{A}_2$*，其差别在于对所有的非目标节点，$\hat{h}_1<\hat{h}_2$，那么我们就说$\mathbf{A}_1$*比$\mathbf{A}_2$*更灵通(*informed*)。在叙述下面的结果时没有进行证明（参见[Hart，Nilsson，& Raphael 1968，Hart，Nilsson，& Raphael 1972，Nilsson 1980]）。

定理9.2　如果$\mathbf{A}_2$*比$\mathbf{A}_1$*更灵通，那么对任何有从n_0到目标节点的一条路径的图，在搜索终止时，被$\mathbf{A}_2$*扩展过的每一个节点也被$\mathbf{A}_1$*扩展过。

可以得出$\mathbf{A}_1$*扩展的节点至少和$\mathbf{A}_2$*的一样多，这个更灵通的算法$\mathbf{A}_2$*也就更有效。因此，我们要寻找尽可能接近真实函数h的函数 $\hat{h}$（为了搜索效率），同时又不能超过它们（为了可接纳性）。当然，在衡量总的搜索效率时，也应当考虑计算 $\hat{h}$ 的代价。最灵通的算法是 $\hat{h}\equiv h$，但是典型地讲，这样的 $\hat{f}$ 只能在很高的代价下，通过完成我们正在尝试的每一个搜索才能得到。

图9-6概括了我们已经讨论过的一些搜索算法的关系。当对所有的节点 $\hat{h}\equiv 0$时，我们得到相同代价算法（搜索顺着相同代价的边沿向外扩展）。当 $\hat{f}(n)=\hat{g}(n)$=深度（n）时，我们得到广度优先搜索法，它顺着相同深度的边沿向外扩展。相同代价和广度优先算法都是**A*** $\hat{h}\equiv 0$)的特殊情况，故它们也都是可接纳的。

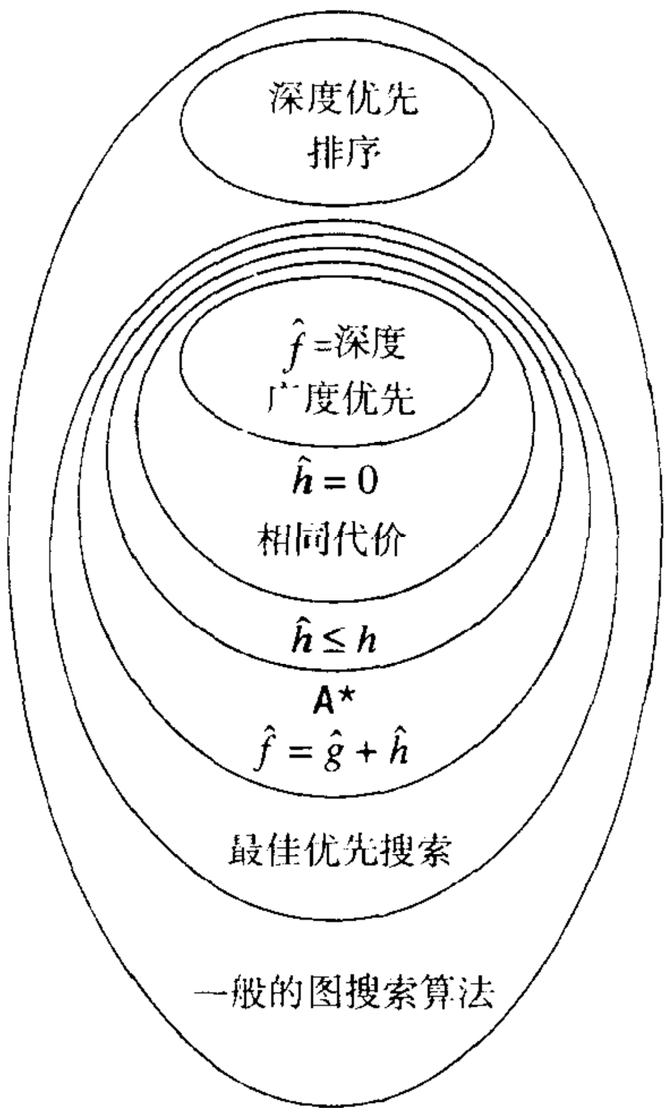


图9-6　搜索算法之间的关系

9.2.3 一致性（或单调）条件

考虑一对节点（n_i、n_j），n_j是n_i的一个后继。如果在搜索图中所有的这种节点对都满足下述条件：

$$\hat{h}(n_i) - \hat{h}(n_j) \leqslant c(n_i, n_j)$$

其中$c(n_i, n_j)$是从n_i到n_j的代价。

上式也写作：$\hat{h}(n_i) \leqslant \hat{h}(n_j) + c(n_i, n_j)$

和

$$\hat{h}(n_j) \geqslant \hat{h}(n_i) - c(n_i, n_j)$$

那么我们就说h服从一致性条件。这个条件陈述了顺着搜索图中的任何路径，到达目标的最优（剩余）代价的估价的减少不会大于该路径弧代价。也就是说，在考虑了一个弧的已知代价后，启发式函数在局部是一致的[⊖]。这种一致性条件可以表示为如图9-7所示的三角不等式。

一致性函数也暗示了当搜索树中结点的值远离开始节点时，它是单调非递减的。设n_i和n是由A*在搜索树上产生的两个节点，n_j是n_i的后继。如果满足一致性条件，就有$\hat{f}(n_j) \geqslant \hat{f}(n_i)$。为了证明这个事实，我们从一致性条件开始：

$$\hat{h}(n_j) \geqslant \hat{h}(n_i) - c(n_i, n_j)$$

给上式两边都加上$\hat{g}(n_i)$，有：

$$\hat{h}(n_j) + \hat{g}(n_j) \geqslant \hat{h}(n_i) + \hat{g}(n_i) - c(n_i, n_j)$$

但是$\hat{g}(n_j) = \hat{g}(n_i) + c(n_i, n_j)$（我们可能会担心$\hat{g}(n_j)$会比这个值小，因为我们可能会顺着其他的比通过$n_i$点代价更低的路径到达$n_j$。但这样一来，在搜索树上$n_j$就不是$n_i$的后继了）。

因此，$\hat{f}(n_j) \geqslant \hat{f}(n_i)$；

因为这个原因，一致性条件（对$\hat{h}$）也常被称为单调条件（对$\hat{f}$）。下面是一个涉及到一致性条件的重要定理（[Hart, Nilsson & Raphael 1968]）。

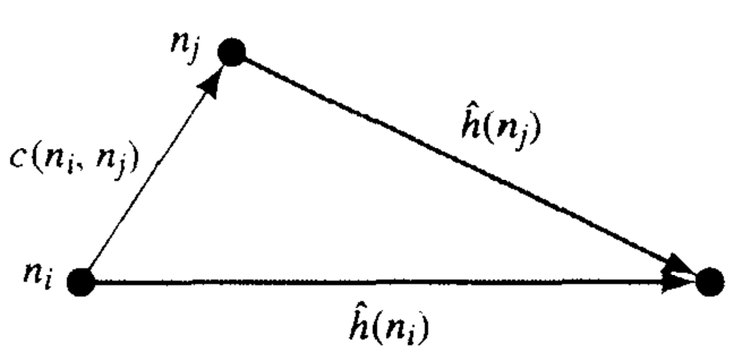


图9-7 一致性条件

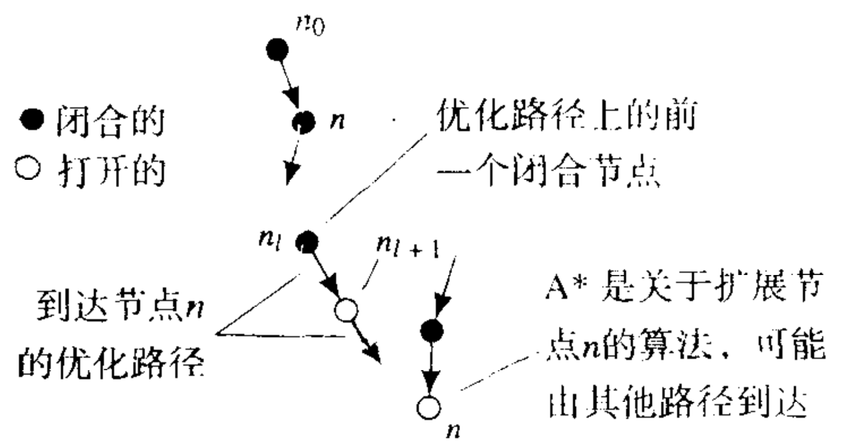


图9-8 用于证明定理9.3的图

定理9.3 如果$\hat{h}$上的一致性条件被满足，那么当A*扩展节点n时，它已经找到了到达节点n的一条最优路径。

证明：假设A*在隐式图G中正在搜索从开始节点n_0到目标节点的一条最佳路径，它准备扩展一个打开的节点n。设$\xi = (n_0, n_1, \ldots, n_l, n_{l+1}, \ldots, n = n_k)$是图$G$中从$n_0$到$n$的一条最佳路径的节点序列，$n_l$是$\xi$上被A*扩展的最后一个节点（参见图9-8）。因为$n_l$是$\xi$上最后一个没打开的

⊖ Pearl [Pearl 1984, pp.82-83] 定义了一个启发式函数的全局一致性概念，并证明它等同于我们关于局部一致性的定义

节点，因此n_{i+1}在*OPEN*上是一个扩展候选者。

对任何节点n_i和它的后继节点n_{i+1}，在ξ中（到达n的一条最优路径），我们有

$$g(n_{i+1}) + \hat{h}(n_{i+1}) = g(n_i) + c(n_i, n_{i+1}) + \hat{h}(n_{i+1}) \geqslant g(n_i) + \hat{h}(n_i);$$

当一致性条件满足时，由≥关系的传递性可以得到

$$g(n_j) + \hat{h}(n_j) \geqslant g(n_i) + \hat{h}(n_i)$$（对ξ上的任何n_i和n_j $(i<j)$）

在特殊情况下，

$$g(n) + \hat{h}(n) \geqslant g(n_{i+1}) + \hat{h}(n_{i+1}) = \hat{f}(n_{i+1})$$

因为**A***已经发现了到达n_{i+1}的一条最佳路径，使 $\hat{g}(n_{i+1}) = g(n_{i+1})$

但是由于**A***将要扩展节点n而不是n_{i+1}，故必然有

$$\hat{f}(n) = \hat{g}(n) + \hat{h}(n) \leqslant \hat{f}(n_{i+1})$$

但是我们已经有

$$\hat{f}(n_{i+1}) \leqslant g(n) + \hat{h}(n);$$

因此

$$\hat{g}(n) \leqslant g(n);$$

但是由于我们计算 $\hat{g}$ 的方法暗示了 $\hat{g}(n) \geqslant g(n)$，故必然有 $\hat{g}(n) = g(n)$;

这表示了或者$n_{i+1} = n$，或者我们必然已经找到了到达节点n的一些其他最佳路径。证明完毕。

一致性条件是很重要的，因为当它被满足时，**A***不再需要重定向指针（第7步），搜索一个图与搜索一个树就没有什么差别了。

满足一致性条件时，可以为**A***的可接纳性给出一个简单直观的论证。它是这样的：

1) $\hat{f}$ 的单调性暗示了搜索顺着 $\hat{f}$ 值增大的边缘向外扩展。

2) 因此，被选择的第一个目标节点就是有最小 $\hat{f}$ 值的一个目标节点。

3) 对任何目标节点n_g，$\hat{f}(n_g) = \hat{g}(n_g)$（这里，我们用了这样的事实，如果 $\hat{h}$ 函数是一致的，它也将绝不会比真正的h函数大[Pearl 1984，p. 83]）。

4) 因此，第一个被选择的目标节点将是有最小 $\hat{g}$ 值的目标节点。

5) 作为定理9.3的一个结论，无论何时（特别地）当一个目标节点n_g被选择扩展时，我们已找到了到达那个目标节点的一个最佳路径。即 $\hat{g}(n_g) = g(n_g)$。

6) 这样，第一个被选择的目标节点将是算法发现的最佳路径的目标节点。

很多启发式函数满足一致性条件。例如，8数码问题中的“不在正确位置的数码个数”函数就是一个例子。当一个启发式函数不满足一致性条件，但其他方面是可以接受的条件时，那么（用Mérõ[Merõ 1984]提出的思想）我们能（在搜索期间）调整该函数使它满足一致性条件。假如，在**A***的每一步，我们检查刚刚扩展的节点n的后继的 $\hat{h}$ 值。任何 $\hat{h}$ 值小于 $\hat{h}(n)$值的节点减去从n到这个节点的弧代价就会得到它们调整后（在搜索过程中）的 $\hat{h}$ 值，这样它们就刚好等于 $\hat{h}(n)$值减去那段弧代价的所得值（但是*CLOSED*中的某个节点必须移回到*OPEN*中。这种情况可能会发生，该节点的 $\hat{h}$ 值是用这种方式调整过的）。读者可以去验证一下这种调整方法，它将使算法是可以接纳的。

9.2.4　迭代加深的 A*

在第8章，讲过广度优先搜索的存储需求会随着搜索空间中目标深度的增加呈指数递增。

尽管好的启发式搜索减少了分枝因子，但启发式搜索还是有如上一样的缺点。在第8章介绍的迭代加深搜索，不但允许我们找到最小代价路径，而且存储需求随着深度增加呈线性增长。由[Korf 1985]提出的迭代加深A*(IDA*)方法能获得同启发式搜索相似的好处。通过使用IDA*的并行实现能获得更高的效率（见[Powley，Ferguson，& Korf 1993]）。

该方法同普通迭代加深方法工作相似，我们执行一系列深度优先搜索。在第一次搜索中，建立一个“截止代价”，它等于 $\hat{f}(n_0) = \hat{h}(n_0) + \hat{g}(n_0) = \hat{h}(n_0)$，$n_0$是开始节点。我们所知道的就是到达目标的最佳路径代价可能等于这个截断值（如果$h(n_0) = \hat{h}(n_0)$，上述语句就是肯定的；因为$h(n_0) \geq \hat{h}(n_0)$，它不可能较小）。我们在深度优先方式下扩展节点，无论何时，只要扩展节点的一个后继的 $\hat{f}$ 值超过截断值，就进行回溯。如果该深度优先搜索终止在一个目标节点，显然它已经找到了到达目标的一个最小代价路径。否则，一个最优路径的代价一定比这个截断值要大。因此，我们增大截断值，开始另一次深度优先搜索。一个最优路径的下一个可能值会是什么？它可能和上一次深度优先搜索中遍历过（但没有扩展）的节点的最小 $\hat{f}$ 值一样小。有这个最小 $\hat{f}$ 值的节点可能是在一条最优路径上（因为我们知道没有任何最佳路径的代价等于前一次的截断值）。这个最小 $\hat{f}$ 值可以作为下次深度优先搜索的新的截断值，等等，如此重复进行。显然，可以容易地看到，IDA*保证会找到一个到达目标的最小代价路径（[Korf 1985]提出了这个结果的一个证据，[Korf 1993]再次提供证明。后面还讨论了当单调条件不满足时迭代加深方法的一些限制，并且提出了一个称为递归最优搜索的变更算法）。

IDA* 确实重复节点扩展，但与普通迭代加深一样，存储需求和深度优先搜索实现效率之间存有折衷。当搜索空间的每个节点的 $\hat{f}$ 值都不同，迭代次数可能等于其 $\hat{f}$ 值比最优路径代价小的节点的数目，情况又怎样呢？（有些方法放弃可接纳性来处理这个问题）。

9.2.5 递归最优搜索

由Korf提出的递归最优搜索，RBFS([Korf 1993]方法比IDA*用到的存储稍微多一点，但是比IDA*产生的节点少。当扩展一个节点n时，RBFS计算n的后继的 $\hat{f}$ 值，并且重新计算搜索树上n和n的祖先的 $\hat{f}$ 的值。这个重新计算的过程叫做 $\hat{f}$ 值的回溯(*backing up*)。回溯是这样进行的：刚刚扩展的节点的后继的回溯值就是该后继的 $\hat{f}$ 值。搜索树上带有后继m_i的节点m的回溯值是

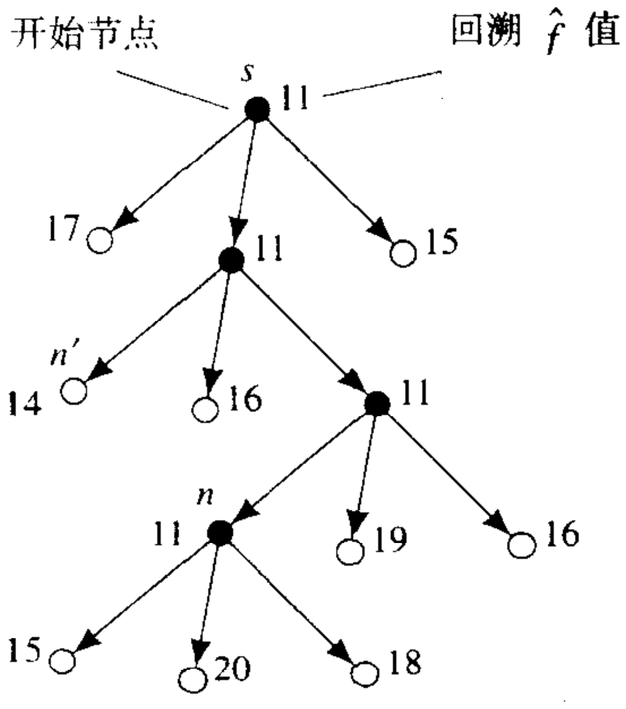
$$\hat{f}(m) = \min_{m_i} \hat{f}(m_i)$$

其中 $\hat{f}(m_i)$是节点m_i的回溯值。

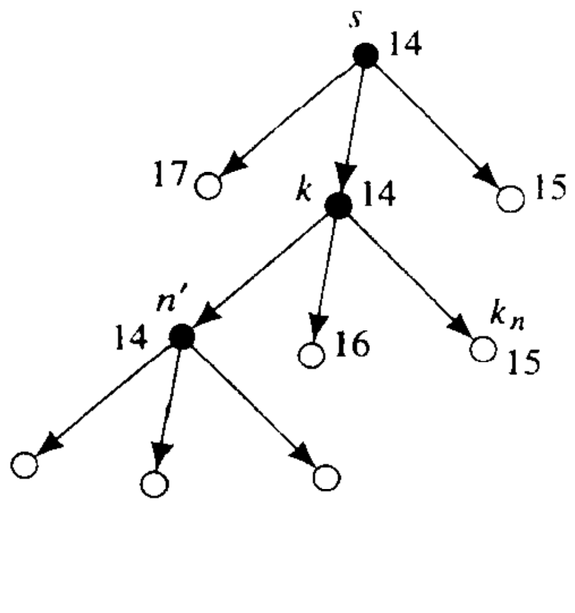
如果节点n(它刚被扩展)的后继之一在所有的*OPEN*节点中有最小 $\hat{f}$ 值，它被依次扩展一直进行下去。但是如果*OPEN* 中的其他节点n'——它不是n的后继，有最小的 $\hat{f}$ 值。在这种情况下，算法回朔到节点n'和n的最低共同祖先节点k。让节点k_n是到节点n的路径上节点k的后继。RBFS移去以k_n为根的子树，于是k_n变成了其 $\hat{f}$ 值等于它的回溯值的一个*OPEN* 节点，搜索继续在有最低 $\hat{f}$ 值的*OPEN*节点n'下进行。

图9-9表达了该算法的主要思想。搜索由一条节点路径和该路径上所有节点的兄弟构成。因此，存储要求仍是到目前为止所探索的最优路径长度的线性函数。顶部节点都在*OPEN*上，在搜索树上的所有节点都和回溯的 $\hat{f}$ 值有关。

若需了解其他有关空间效率的搜索算法，可以参考[korf 1996]。



a) RBFS 仅扩展节点n，并未回溯出它的后继的 $\hat{f}$ 值



b) 已回溯出 $\hat{f}$ 值，节点k_n的子树已丢弃，搜索从节点n'以下的节点继续

图9-9　递归最优搜索

9.3　启发式函数和搜索效率

在决定A*效率时，启发式函数的选择是至关重要的。用 $\hat{h} \equiv 0$保证了可接纳性，但生成了相同代价搜索，因而效率不高。$\hat{h}$ 等于h上较低约束的最高可能值，不但可以扩展较少的节点，还能维持可接纳性。如8数码问题中，函数 $\hat{h}(n) = W(n)$ (其中$W(n)$是不在正确位置的数码个数)就是$h(n)$上的一个较低约束，但是它没有提供对数码状态难度（按照到达目标的步数）的一个很好的估计。一个更好的估计是函数 $\hat{h}(n) = P(n)$，$P(n)$是每个数码离开“初始点”（忽略插入数码）的距离总和。

在AI历史的早期，[Newell，Shaw & Simon 1959，Newell 1964]建议用一个简化的模型来构成指导搜索的“计划”。同样的思想被应用于找到一个好的启发式函数的问题中，[Gaschnig 1979]和[Pearl 1984，第4章]描述过这种思想。例如，我们可以通过允许较少约束的数码移动来放宽8数码的约束条件。如果我们允许一个数码直接移到(一次)它的目标区中，那么到达目标的步数就是在错误位置的数码的个数之和$W(n)$。一个约束稍强(因此更好)的模型允许数码移动到相邻的位置，即使已经有一个数码在那里。在这种模型中，到达目标的步数就是每个数码离它的目标位置之和$P(n)$。这些模型表示了一个agent如何自动计算我们直觉猜想的 $\hat{f}$ 函数。Pearl指出，我们可能已经计算了一个函数 $\hat{h}$，它比$W(n)$稍微好一些，在$W(n)$中，任何数码在一步移动中可以和空格交换。在另一个约束稍强的模型中，数码只能顺着相邻单元(顺着该路径计算每一个位置)的一条路径移到空格的位置。

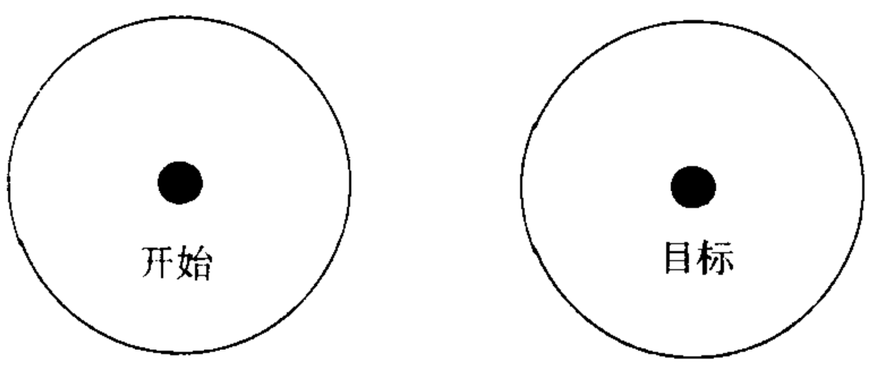
在road-map问题中，能够用不严格约束的模型来解释作为 $\hat{h}$ 函数使用的欧几里德(直线)距离。在一个非严格约束的模型中，不必穿过每一条现存的路，一个旅行者能“电话传输”，用直线距离直接到达任何城市。由于在非严格约束的模型中的方案决不会超过原始问题的代价，故所选择的 $\hat{h}$ 函数总是可接纳的！结果是它们也是一致的。因此使用它们时，算法不必再次遍历以前扩展过的节点。

在选择 $\hat{h}$ 函数时，我们必须考虑计算 $\hat{h}$ 本身的计算量。模型的非严格约束越少，启发式函数就越好——当然计算会越困难。如果一个人用的启发式函数等于h，那么可以达到扩展节点的绝对最小值，但是这个 $\hat{h}$ 的计算将要求解决最初的问题。我们常常是在精确 $\hat{h}$ 函数和其计算代价之间取折衷值。

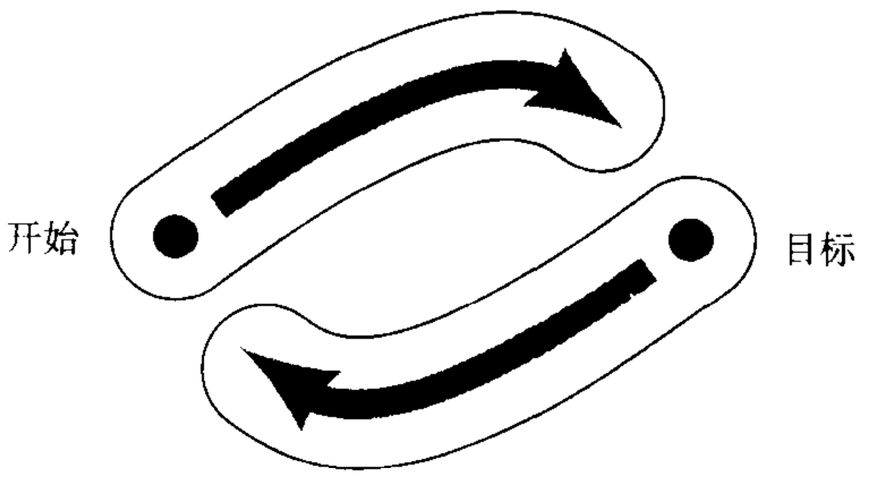
经常，通过使用一些非低约束的函数 $\hat{h}$ 以可接纳性为代价来换取搜索效率。也就是说，一个可能不是最佳的路径比最佳路径更容易找到，一个非较低约束的 $\hat{h}$ 函数可能比一个较低约束的函数容易计算。在这些情况下，效率可能会成倍地增加——因为被扩展的节点会被减少（虽然以可接纳性为代价）并且计算量也被减少。

另一种可能性是修改评估函数中 $\hat{g}$ 和 $\hat{h}$ 的权值，即用 $\hat{f}=\hat{g}+w\hat{h}$，w是一个正数。非常大的w值会过分强调启发式部分，而非常小的w会突出搜索的广度优先特性。实验结果表明如果w值与搜索树上节点深度成反比，常会提高搜索效率。在浅深度时，搜索主要依赖启发式部分，然而随着深度的增加，为了确保最终会发现一些到达目标的路径，搜索会逐渐以广度优先为主。

可能有人认为通过从开始点和目标点两边同时搜索，可以改善搜索效率。典型地讲，这种改善只能通过广度优先搜索获得。当广度优先搜索用于双向搜索时，我们可以保证搜索的前沿会在开始点和目标点之间相遇（见图9-10a），且由于已经建立终止条件，双向广度优先搜索可以保证找到一条最佳路径[Pohl 1971]。但是如果启发式搜索由双向开始，如果启发式倾向于在不适当的方向进行搜索，那么两个搜索前沿可能不会相遇，从而不能找到一条最佳路径（见图9-10b）。



a) 广度优先搜索



b) 启发式搜索

图9-10 双向搜索

搜索效率的一个度量是有效分枝因子 B，它描述了一个搜索过程朝着目标前进的激烈程度。假设搜索找到了一个长为d的路径，生成了N个节点，那么B就是有下列属性的树上的每个节点的后继个数：

- 树中每个非树叶节点都有B个后继。
- 树中的树叶节点的深度均为d。
- 树中的节点总数是N。

因此，B和路径长度d以及生成的总节点数N之间有下列关系：

$$B + B^2 + \cdots + B^d = N$$

$$\frac{(B^d - 1)B}{(B - 1)} = N$$

虽然B不能明确地表达为d和N的函数，但是B相对于N在不同d值下的图表被显示在图9-11中。一个朝着目标高聚集的搜索，其B值在其他方向是非常小的；另一方面，一个“灌木丛式”的搜索图将会有高的B值。

在有效搜索因子适当地独立于路径长度的范围内，可以用它来评价在各种长度的搜索中产生的节点数。例如，用评估函数 $\hat{f}(n) = \hat{g}(n) + W(n)$，对于图9-2中的8数码问题，我们可以用图9-11计算它产生的节点个数，B值等于1.2。假如我们想用同样的评估函数来解决一个更难的8数码问题，比如说要求30 步，来估计会产生多少个节点。从图9-11中，假定有效分枝因子保持常量，我们看到，30步问题产生大约2000个节点。

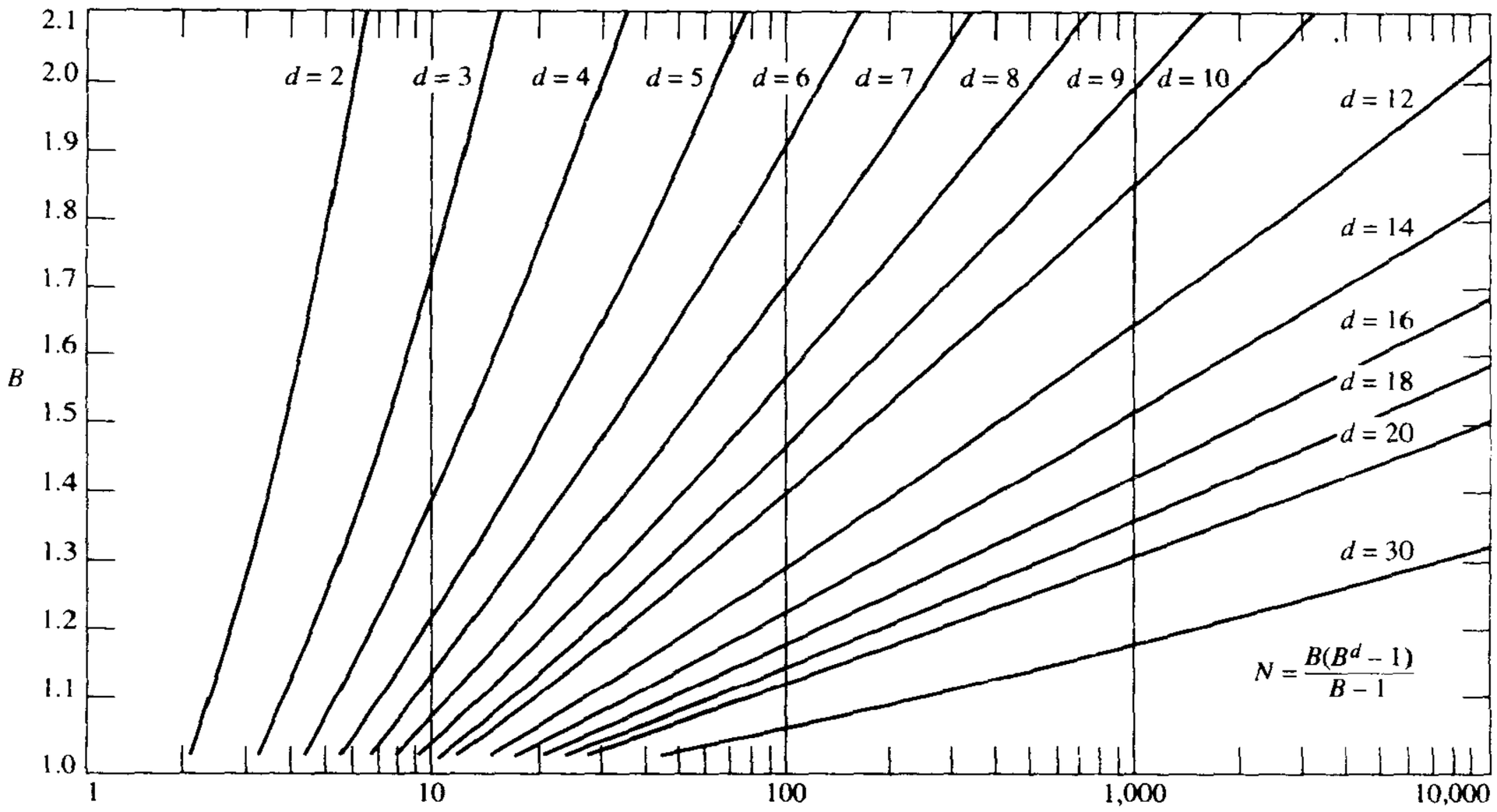


图9-11　不同d值下，NB随N的变化图

归纳一下，有三个重要的因素影响算法**A***的效率：

- 被发现路径的代价（长度）；
- 在发现路径中被扩展的节点数；
- 计算 $\hat{h}$ 的计算量。

选择适当的启发式函数可以让我们平衡这些因素以最大化搜索效率。

到目前为止讨论的所有搜索方法，包括启发式搜索，其时间复杂度都是$O(n)$，n是产生的节点数（假定启发式函数能在常量时间内被计算出）。特殊地，当有效分枝因子是B，弧代价都相等，并且从开始节点到目标的深度为d的情况下，广度优先搜索的复杂度为$O(B^d)$，对相同代价搜索（ $\hat{h} \equiv 0$）和不相等的弧代价，复杂度是$O(B^{C/c})$，其中C是最优方案的代价，c是最小代价弧的代价［korf 1992］。对很多实际的问题，由于d(或C/c)太 大，从而导致不能计算最优方案的搜索（甚至启发式搜索）。例如，我们用搜索算法计算一个目标是15步远的最佳下一步动作（比如有4个选择），搜索算法可能必须扩展$4^{15} \approx 10^9$个节点。这么大的计算量对一个要在数

秒内做出决策的agent可能是不实际的。由于所有被扩展的节点必须存在一个树结构中，A*的空间复杂度和时间复杂度的级数相同。

由于agent对它们的计算资源有时间和空间约束，因此不可能在很多任务中找到最佳方案。相反，做出非最优但是可以接受的方案(称为可以满足的方案)或者局部方案倒是必需的。下一章将提出各种能在agent的有限资源下可以使用的方法。

9.4 补充读物和讨论

启发式搜索涉及两种计算。第一是真正的扩展节点和产生路径的目标级计算。第二是决定下一个要扩展的节点的元级计算。目标级是环境中的物理动作；元级是图中的计算动作。在人工智能中，常常会提到目标级/元级的差别。[Russell & Wefald 1991]完整地讨论了它们，在下一章要讨论的简化型搜索方法中，它们扮演了一个特别重要的角色。

8数码问题被很多AI研究者作为启发式搜索方法的试验台。由[Doran & Michie 1966]写的一篇论文使用了8数码问题，开始了在图中寻找路径过程中使用评估函数的历史。

1990年，Korf阐述了“IDA*能解决15数码问题，但更大的数码问题[如24数码]在当前的机器上是很难处理的”[korf 1990，p.191]。然而，随着功能更强的机器(一台Sun Ultra Sparc 工作站每秒钟能产生一百万个节点)和更好的(自动发现的)启发式算法的出现，[Korf & taylor 1996]能在2.25小时到1个月的时间范围内找到随机产生的可解决的24数码问题的最优方案。

虽然这些数字问题对改进和测试搜索技术是有用的，但A*和相关的启发式搜索过程也已被成功地应用到了很多实际问题中。

关于更多的使用非约束模型发现启发式函数的内容，参见[Mostow & Prieditis 1989，Prieditis 1993]。[Pohl 1973]用不同的权值对 $\hat{f}$ 函数的启发式部分进行了实验。

关于启发式搜索的经典书籍是[Pearl 1984]。[Kanal & Kumar 1988]是关于搜索算法的论文集。后一本书的第一篇文章提出了一个由AI和运筹学人员分别独立开发的一致搜索方法。

习题

9.1 在“四皇后问题”中，我们把四个皇后放在一个4 × 4的棋盘上以便四个棋子不能彼此抓到对方(即，只有一个皇后能在棋盘的任何行列或对角线上)。假如我们想用下面的问题空间解决这个问题：开始节点由一个空的4 × 4数组表示；后继函数产生新的4 × 4数组，该数组包含一个皇后的附加合法放置，它可在数组的任何位置;预计目标是只要四个皇后在数组中的条件(合法位置) 都被满足。

1) 根据皇后到达目标的剩余步数，为这个问题设计一个可以接纳的函数(注意所有的目标节点离开始节点刚好四步!)。

2) 使用你的启发式函数在A*中搜索一个目标节点。画出包含搜索产生的所有4 × 4数组的搜索树，用g和 $\hat{h}$ 值标出每一个数组(注意：考虑到对称性，我们只要产生开始节点的三个后继就可以了)。

9.2 这个习题假定你已经完成了习题7.4中的汉诺塔问题。参考你对那个问题的答案，如果你还没有做它，那么现在解答。提出该问题的一个可接纳的函数 $\hat{h}$，它要比 $\hat{h} \equiv 0$更好。

9.3 算法A*直到一个目标节点被选择扩展才会终止。然而，到达目标节点的一条路径可能

在那个节点被选择扩展前早就找到了。一旦目标节点被发现，为什么不终止搜索呢?用一个例子说明你的答案。

9.4 启发式函数中的单调条件要求对所有的节点—后继对(n_i, n_j)，满足 $\hat{h}(n_i) \leqslant \hat{h}(n_j)+c(n_i, n_j)$，$c(n_i, n_j)$是从$n_i$到$n_j$的弧代价。如果单调条件不能满足，一种方法建议在搜索过程中可以调整 $\hat{h}$ 以使条件被满足。该思想是指无论何时，当后继为n_j的节点n_i被扩展时，我们可以给 $\hat{h}(n_j)$增加一个值使单调条件满足。构造一个例子说明即使使用这种方法，当一个节点被扩展时，我们不必找到它的一个最小代价路径。

9.5 说明在算法A*中，如果我们从*OPEN*中移去任何节点n，$\hat{f}(n) > F$，F是$f(n_0)$的一个上限约束，n_0是开始节点，这时的A*仍是可接纳的。

9.6 在这个习题中，我们考虑在两个可接纳的启发式函数中做出选择，其中一个总是“至少像另一个一样精确”。更准确地讲，对一个固定的目标，让 $\hat{h}_1$和 $\hat{h}_2$是两个可接纳的启发式函数，在搜索树中的每一个节点n有 $\hat{h}_1(n) \leqslant \hat{h}_2(n)$。

回想一下，A*按照 $\hat{f}$ 值扩展节点，但是相同 $\hat{f}$ 值的节点被扩展的次序会根据优先队列的实现发生变化。对一个给定的启发式函数 $\hat{h}$，如果扩展节点的顺序和A*使用 $\hat{h}$ 搜索的顺序一致，我们就说搜索树中的节点顺序是 $A_{\hat{h}}$*—合法。

让N为使用 $\hat{h}$ 的A*搜索扩展的节点集。证明总是有一些搜索顺序是 $A_{\hat{h}_1}$*—合法，它仅仅(不必严格)扩展了N中节点的一个子集。即证明 $\hat{h}_2$比 $\hat{h}_1$产生了一个更小的搜索空间，这种情况总是可能的(注意这个问题叫你找到一些有特殊属性的搜索顺序。在一般情况下，在任意的搜索顺序下拥有这些属性是不可能的，你知道为什么吗?)。

第10章　计划、动作和学习

我们已经探讨了在图中查找路径的几种技术,现在来研究这些方法如何被agent用于现实问题中。先回顾一下在第7章第一次考虑图搜索计划方法时所做的假设，并提出一个能满足这种理想假设的agent结构，然后，将讨论如何修改一些搜索算法以减少它们对时间和空间的要求——使它们在提出的体系结构中更有用。最后，讨论了如何学习启发式函数和动作模型。

10.1　感知/计划/动作循环

如第7章所述，基于搜索的规划方法的功效依赖于几个很强的假设。由于以下原因，这些假设常常得不到满足：

1) 知觉过程不可能总是提供环境状态的必需信息(由于噪声或者对重要的特性不敏感)。当两种不同的环境状态引起相同的传感输入时,我们称这种情况为感知混淆 (*perceptual aliasing*)。

2) 动作并不总有其模型效果(由于模型不够精确，或者受动器系统在执行动作时偶尔会产生错误)。

3) 可能在环境中有其他的物理过程或其他的agent(例如,在游戏中有对手)。这些过程可能会改变环境以致于干扰agent的动作。

4) 外部作用的存在会引起其他的问题：在构造一个计划期间，环境可能变得与原来的计划不相干。这种困难使得花费太多的时间为一个agent进行计划而变得毫无意义。

5) agent可能在完成一个到达目标状态的搜索之前被要求动作。

6) 即使agent有充分的计算时间，但是计算要求的空间资源不允许搜索进行到目标状态。

有两种主要方法可以用来解决这些困难，同时又能保留基于搜索的计划的主要特征。一种是用概率方法来形式化知觉、环境和受动器的不确定性；另一种办法是用各种附加的假设和近似来消除这些困难的影响。

处理动作的不确定效果的一种正式方法是假定对一定状态下的每一个可执行动作，结果状态由一个已知的概率分布给出。在这种情况下找到合适的动作被称为Markov决策问题（*Markov decision problem*, MDP）[*Puterman*, 1994]。通过假定agent的传感设备在它的状态集上提供一个概率分布，可以解决有缺陷知觉的其他问题。发现动作则被称为局部可见的Markov决策问题（*Partially observable Markov decision problem*, POMDP) [Lovejoy 1991, Monahan 1982, Cassandra, Kaelbling, & Littman 1994]。讨论MDP和POMDP超出了本书的范围，但在介绍了概率干扰后，会在第20章讨论与它们相关的一些技术。

在这里不讨论正式的、基于概率的方法，而是提出一个叫感知/计划/动作(*sense/plan/act*)的结构，在很多应用中它避开了上述的一些复杂性。该结构的基本原理是即使动作偶尔产生了没有预料的结果，或者agent有时不能决定它处于哪一种环境状态下，但是通过保证agent从它的执行环境中得到连续的反馈，这些困难可以被充分地解决。

确保连续反馈的一个方法是计划一个动作序列，只执行这个序列中的第一个动作，感知结果环境状态，重新计算开始节点，然后重复上述过程。这种方式，选择动作的agent被叫做感

知/计划/动作agent。然而为了使这个方法有效，计算一个计划的时间必须比每个动作执行时间要少。图10-1显示了一个感知/计划/动作agent的结构。在良性环境中（容忍几个错误步骤），感知和动作中的错误在感知/计划/动作循环序列中“达到平均数”。

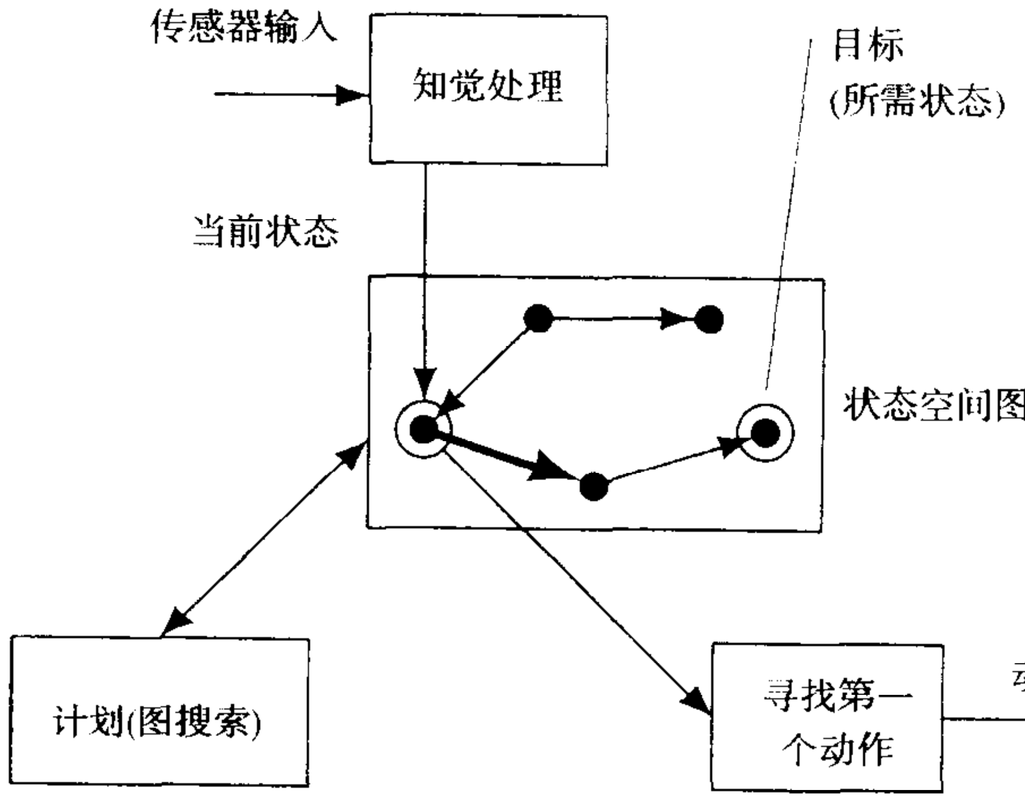


图10-1　一个感知/计划/动作agent的结构

在感知/计划/动作循环中的环境反馈允许解决感知、环境和受动器的一些不确定性。然而，为使反馈有效，必须保证感知和动作一般来说是精确的。在很多应用中，这种假设是现实的。毕竟，提供感觉、感知和受动器特征适合于任务要求是agent设计人员的任务。通常，agent通过比较即刻的感知数据和未展开状态的存储模型能够提高扩展感知精确度（回顾第5章演示的黑板系统的过滤的例子）。

10.2　逼近搜索

下面对以产生计划质量为代价的有限计算或时间资源的搜索算法进行修改，这些计划可能不是最佳的，或者可能不是总能可靠地到达目标状态。即使这个计划不是最优的（甚至也不正确），这些技术的应用也能被合并到感知/计划/动作循环中。定性地讲，只要第一个动作有缩短到达目标距离的趋势（平均情况），经感知/计划/动作循环的多次迭代将最终到达目标。

如上一章所讲，放宽产生最优计划的要求常会减少找到一个计划的计算代价。可以从两个方面来减少代价。一是能找到到达目标的一条完整路径但不必要求它是最优的；或者是能找到一条局部的路径，它不要求已达到目标节点。一个**A***类型的搜索可用于这两种方法。对前者，我们用一个不可接纳的启发式函数；对后者，在到达目标前（用可接纳的或不可接纳的启发式函数）退出搜索。在到达目标前退出搜索是任意时间算法（*anytime algorithm*）[Dean & Boddy 1988， Horvitz 1987]的一个例子。任意时间算法能在任何时刻停止，结果的质量会随着运行时间的增加而改善。在下面的几个部分，我们将详细讨论这些逼近搜索方法。

10.2.1　孤岛驱动搜索

在孤岛驱动(*island-driven*)搜索中，来自问题领域的启发性知识被用于在搜索空间中建立一个“岛节点”序列，假定有好的路径通过这个搜索空间。例如，在计划通过有障碍的地形时，这些岛就是相应的山。假如n_0是开始节点，n_g是目标节点，（n_1，n_2，…，n_k）是这些岛的一个

序列。我们用n_0作为开始节点，n_1作为目标节点，开始一个启发式搜索（用一个同那个目标相适应的启发式函数）。当搜索找到了一条到n_1的路径时，就用n_1作起始点，n_2作目标点开始另一个搜索，等等，直到我们发现了一条到达n_g的路。图10-2显示了孤岛驱动搜索。

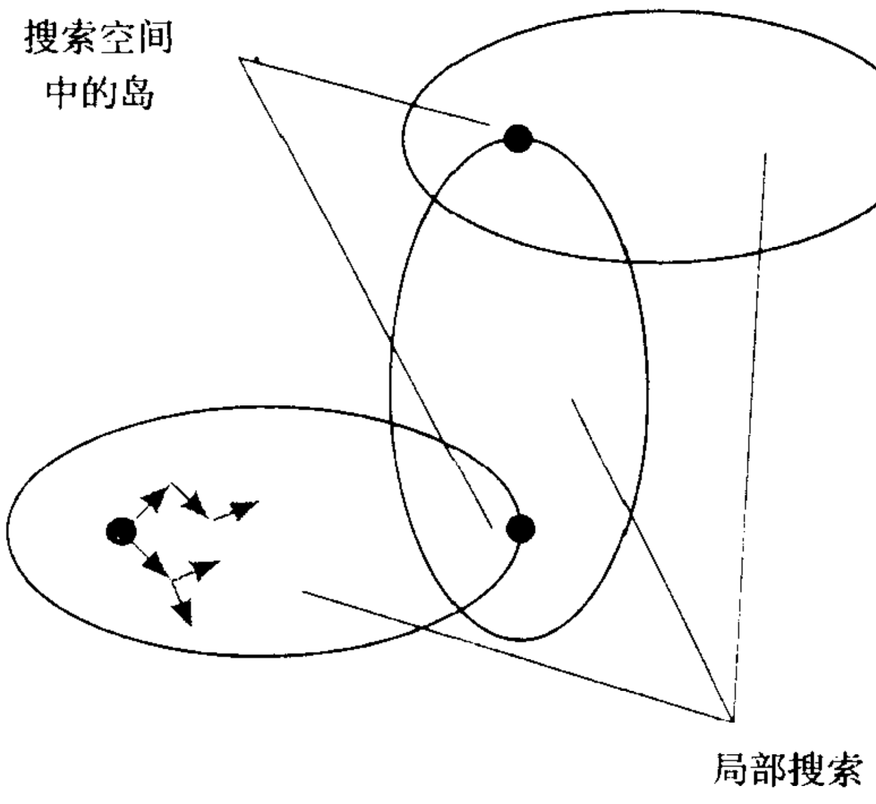


图10-2 孤岛驱动搜索

10.2.2 层次搜索

除了没有显式的岛集合外，层次搜索(*hierarchical search*)非常像孤岛搜索。假定有一些“宏算子”，它们能在一个隐式的岛搜索空间中采取大步骤。一个起始岛（在开始节点附近）和这些宏算子构成了岛的一个隐式的“元级”超大图。首先用一个元(*metalevel*)搜索来搜索这个超大图，直到找到一条宏算子路径，它可以让我们从基级开始节点附近的一个节点到达基级目标节点附近的一个节点。如果已经按照一个基级算子序列定义过宏算子，宏算子可扩展为一条基级算子路径，然后根据基级搜索，这条路径与开始和目标节点相连接。如果没有以基级算子定义的宏算子，我们必须顺着元级搜索中的岛节点路径进行基级搜索。后一种可能性如图10-3所示。

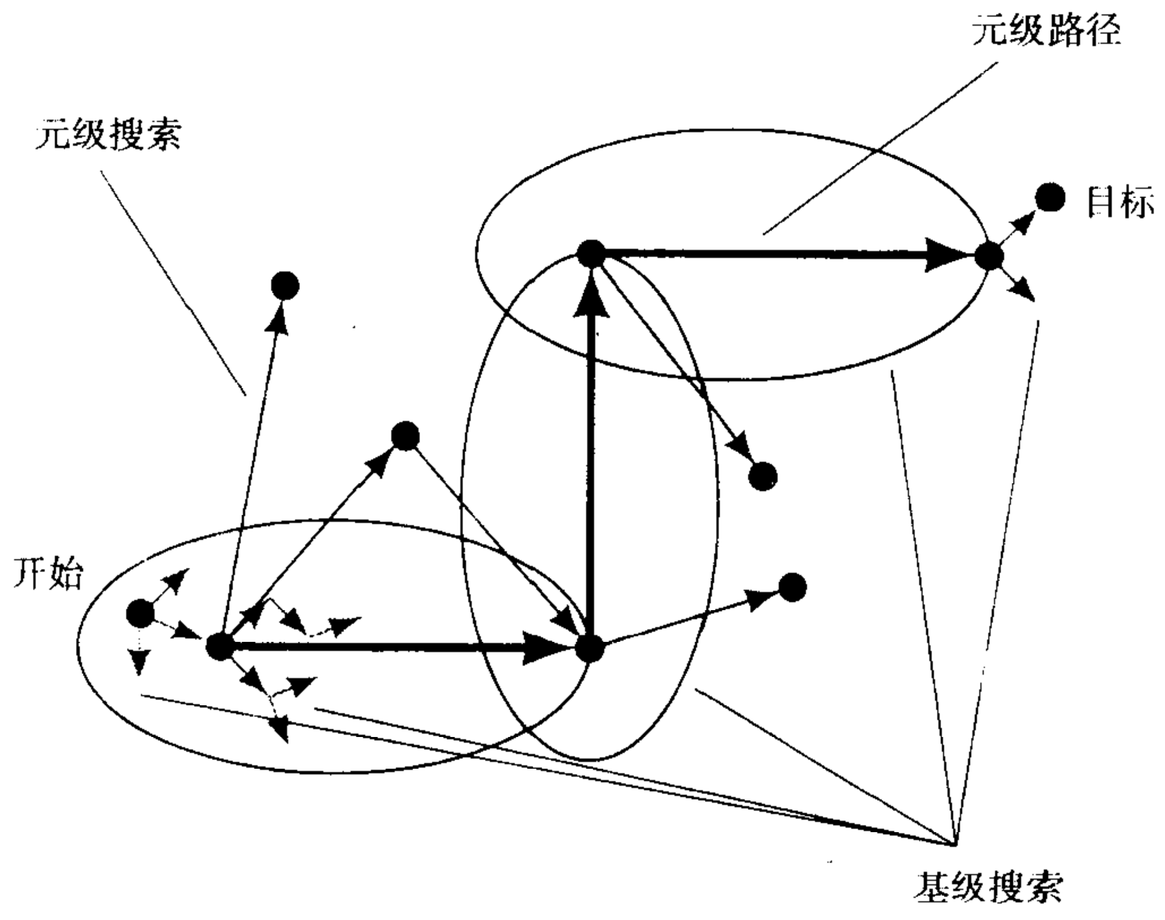


图10-3 分层搜索

作为层次计划的一个例子，考虑在一个网格空间中机器人要将一个方块推到一个给定的目标位置，如图10-4所示。一个可以推动的方块如图所示位于一个单元中，机器人的任务是把目标推到标有*G*的单元中。假定机器人能感知到它的8个相邻单元，能够判断被占据的相邻单元中是一个不可移动的障碍物还是一个可以移动的方块。像第2章一样，假定机器人在它的行列中可移到一个相邻的空白单元中，或者移入一个包含可移动方块的相邻单元中，在这种情况下，方块也向前移动一个单元（除非这种移动被一个不可移动的障碍挡住）。

假如机器人有一个所处环境的图标模型，该模型由一个与图10-4中的单元方格相似的数组表示，那么，机器人首先能做出一个关于方块如何移动的元级计划——假定方块的移动可以和机器人的移动方式相同。这个计划结果用灰箭头画在图10-4中。然后方块移动的每一步就能展开成一个基级计划。方块移动的第一步是要求机器人找到一条路径到达这样一个单元，它和方块相邻且在方块下一个目标位置的相反一侧。基级计划的结果用黑箭头画在图10-4中。随后的基级计划都是简单的，直到方块必须改变方向，这时，机器人必须找到一条路径到达与方块移动方向相反一侧的单元中，这样一直进行下去。完成在方向上变化的基级计划也在图10-4中[⊖]。

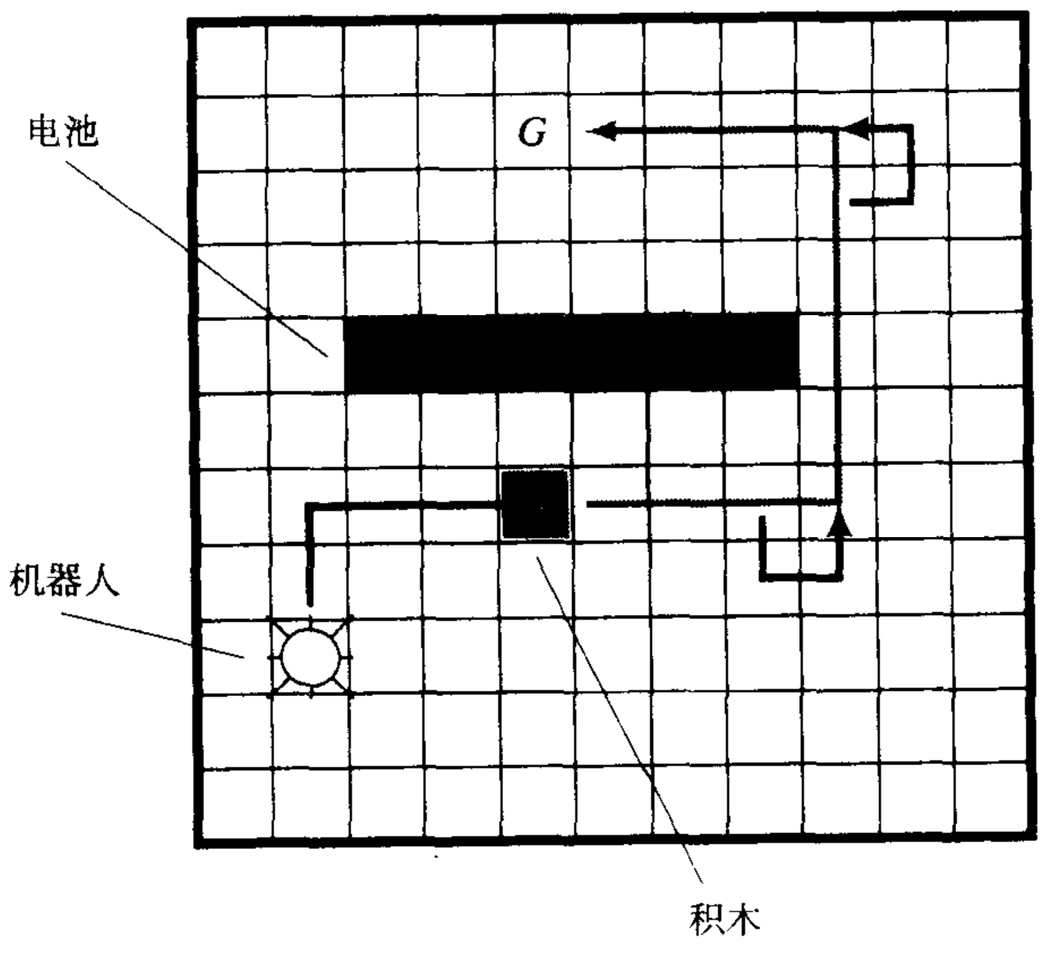


图10-4　推动一个方块

在层次计划中，如果在计划执行期间环境可能变化，仅仅展开元级计划的开始几步是明智的。仅仅展开第一个元级步可以让基级动作去执行，在它执行后，环境反馈可用来开发一个更新的元级计划。

10.2.3　有限范围搜索

在某些问题中，用任何方法搜索发现一条到达目标的路径从计算上讲都是不可能的；而在另外一些问题中，一个动作必须在一个限定的时间内作出选择，而不能在这个时间内搜索到所有到达目标的路径。在这些问题中，用有限的时间和计算量找到一条被认为是在到达目标的好

⊖　在一篇关于计划和执行的早期AI论文中谈到了一个非常像图10-4中模拟机器人的任务。[Nilsson & Raphael 1967]。

路径上的节点可能是有用的，尽管该节点并不是目标节点本身。当必须终止搜索时，这个替身节点 n^*在搜索前沿的所有节点中，有最小的 $\hat{f}$ 值。

假定在一个动作被选择前的可用搜索时间允许搜索到深度d，即所有深度为d或小于d的路径都能被搜索到；在该深度的节点$\mathcal{H}$将被称为范围节点。那么我们的搜索过程将搜索到深度d，然后选择

$$n^* = \underset{n \in \mathcal{H}}{\operatorname{argmin}} \hat{f}(n)$$

作为目标节点的替代。这个方法叫做有限范围搜索(*limited-horizon search*)。[Korf 1990]研究了这个算法并称它为最小搜索（*minimin search*）⊖。一个感知/计划/动作系统将在到达n^*的路径上采取第一个动作，感知结果状态，再迭代搜索，一遍一遍地进行下去。我们希望朝着一个拥有最优启发式指标的节点的第一步动作，正好是在朝着目标的路径上。通常，一个agent没有必要去搜索所有到达目标的路径；因为不确定性，远距离搜索可能是不相关的，不能提供比应用在搜索水平上的启发式函数更好的信息。

有限范围搜索能通过一个处理到深度d的深度优先搜索而高效地执行。使用单调函数 $\hat{f}$ 评估节点可以极大地减少搜索工作。一旦达到搜索范围的第一个节点n_1，当 $\hat{f}(n) > \hat{f}(n_1)$，就能在其他节点$n$下终止搜索。节点$n$不可能有一个子孙，它的 $\hat{f}$ 值比 $\hat{f}(n_1)$小，这样在它下面进行搜索就没有意义了（在单调假设下，顺着搜索树上的任何路径上的节点 $\hat{f}$ 值是单调非递减的）。$\hat{f}(n_1)$被称为α截断值(*alpha cut-off value*)，在节点n下终止搜索的过程是分支和约束（*branch and bound*）搜索过程的一个实例。并且无论何时当到达其他的搜索范围点n_2，$\hat{f}(n_2) < \hat{f}(n_1)$，α截断值就降低到 $\hat{f}(n_2)$，它放宽了截断条件。无论何时，当到达的范围节点 $\hat{f}$ 值比当前的α截断值小时，就能以这种方式降低α截断值。[Korf 1990]把有限范围搜索和各种其他方法进行了比较，包括**IDA*** 和受时间约束的**A***。

有限范围搜索的一种极端形式用1作为深度约束。开始节点的最近后继被评估，导致执行到达最低 $\hat{f}$ 值的后继的动作。这些 $\hat{f}$ 直和第5章讨论的机器人导航问题中的势函数的值相类似，事实上，一个势函数在机器人导航问题中也是一个合适的 $\hat{f}$ 选择。

形式上，设$\sigma(n_0, a)$是agent通过对节点n_0采取动作a期望到达的状态描述，应用与节点n_0上动作a相对应的算子产生状态描述。在节点n_0选择一个如下动作策略：

$$a = \underset{a}{\operatorname{argmin}} \hat{f}(\sigma(n_0, a)) = \underset{a}{\operatorname{argmin}}[c(a) + \hat{h}(\sigma(n_0, a))]$$

其中$c(a)$是动作的代价。在后面讨论学习 $\hat{f}(\sigma(n_0, a))$方法时，将会使用和上式相似的一个等式。

10.2.4 循环

在存在不确定性和agent依赖逼近计划的所有情况中，用感知/计划/动作循环可以产生重复的循环。即agent可能会回到前面遇到过的环境状态，重复在那里采用过的动作。当然，这种反复并不意味着agent永远不能达到目标状态。Korf提出了一个计划执行算法叫实时（*real-time*）A*（RTA*），它建立了所有已经遍历过的状态的一个显式图，同时调整这个图中节点的 $\hat{h}$ 值，使它们在到达前面已经遍历过的节点时不会采取动作[korf 1990]。

⊖ 我们可能会问为什么不搜索到一个给定代价的范围而要搜索到一个给定深度的范围。这是因为搜索需要的时间常常依赖深度而不是代价，因此我们要用深度。

10.2.5 建立反应过程

如第7章开始所述，在一个反应型机器中，设计者已为每一个可能的状态提前计算了合适的到达目标的动作。存储这些和环境状态相对应的动作可能需要大量的内存。另一方面，反应型agent常常比计划型agent反应更快。在某些情况下，提前计算（汇编）一些频繁使用的离线(*offline*)计划，把它们存储为反应例程以便可以在线(*online*)快速产生适当的动作，这样做是有益的。

例如，离线搜索能计划一个以状态空间图中的目标节点为根的生成树(*spanning tree*)，它包含从状态空间中所有（至少很多）节点到达目标的路径。能从目标节点向后搜索产生一个生成树。例如，一个获得目标（块A在块B上，块B在块C上）的积木问题的生成树如图10-5所示，它表示了从所有其他节点到达目标的路径。

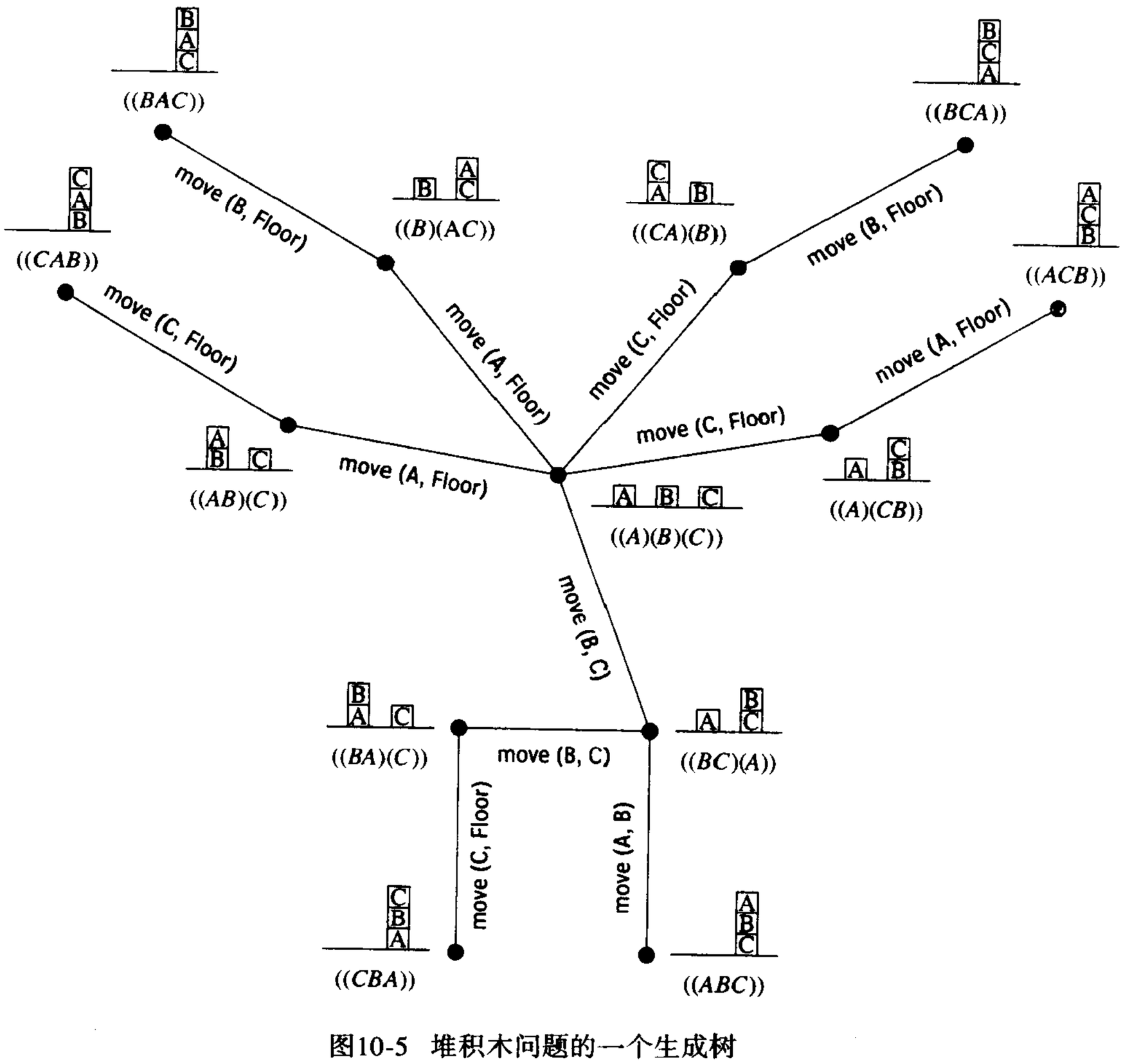


图10-5　堆积木问题的一个生成树

生成树和局部生成树能容易地转换成完全反应型的T-R程序。如果一个反应程序为每一个可能的状态指定了一个动作，就被称为通用计划（*universal plan*）[Schoppers 1987]或动作策略(*action policy*)。在本章的结尾，将讨论动作策略和学习动作策略的方法。即使在一个给定状态下采取的动作并不能总是产生预期的下一个状态，反应型机器仍能处理任何可能产生的状态。

10.3 学习启发式函数

连续的环境反馈是减少不确定性和弥补一个 agent缺乏对自己动作结果知识了解的一种方法。另外，有用的信息也能从环境中的搜索经验和动作经验中抽取出来。下面将讨论agent能够更有效地学会计划和动作的不同方法。

10.3.1 显式图

如果agent没有一个好的启发式函数来评估到达目标的代价，有时需要学习这样一个函数。首先解释一个非常简单的适合特定情况的学习过程，在这种情况下可能会存储所有可能节点的一个显式列表。

首先假定agent对它的动作结果有一个好的模型，并知道从任何节点移到它的后继节点的代价。通过把所有节点的 $\hat{h}$ 值初始化为0来开始这个学习过程，然后启动一个**A***搜索。在扩展 n_i产生后继 $S(n_i)$后，如下修改 $\hat{h}(n_i)$

$$\hat{h}(n_i) \leftarrow \min_{n_j \in S(n_i)} \left[\hat{h}(n_j) + c(n_i, n_j) \right]$$

$c(n_i, n_j)$是从n_i 到n_j的代价。可以将 $\hat{h}$ 值保存在一个节点表中（因为数量少，可以管理）还假定如果产生一个目标节点n_g，并知道 $\hat{h}(n_g)=0$。虽然这个学习过程不能提高我们在第一次到达一个目标节点的搜索时达到比相同代价搜索更快的效率，但以后到达相同目标时（从可能的不同开始状态），通过使用学到的 $\hat{h}$ 函数会加快搜索。在n次搜索后，对真正h函数越来越好的评估会从目标节点逐步向后传播（学习算法**LRTA***对 $\hat{h}$ 使用相同的更新规则[Korf 1990]）。

如果agent对其动作结果没有模型，通过一个相似的过程，agent能同时学会它们和一个 $\hat{h}$ 函数，虽然学习必须在真正的环境而不是状态空间模型中进行(当然，这种学习也可能是危险的!)。假定agent有办法区分它真正遍历过的状态，并能对它们进行命名，做一个显式的图或表来表示状态，它能做出它们的估计值和由动作所引起的迁移。也假定如果agent不知道它们的动作代价，它能学会实施动作的代价。这个过程从一个代表agent开始状态的单个节点开始。它采取一个动作，也许是一个随机的动作，转变到另一个状态。当它访问到一个状态时，就命名它们，并如下计算它们的 $\hat{h}$ 值：

$$\hat{h}(n_i) \leftarrow \left[\hat{h}(n_j) + c(n_i, n_j) \right]$$

其中，n_i是动作a发生的节点，n_j是结果节点，$c(n_i, n_j)$是动作实施的代价，$\hat{h}(n_j)$是对n_j值的一个评估。如果前面从没访问过n_j，则 $\hat{h}(n_j)$等于0，否则它被存在表中。

无论何时，当agent准备对在图中存有后继节点的节点n采取动作时，它根据下面的策略选择一个动作：

$$a = \underset{a}{\operatorname{argmin}} \left[\hat{h}(\sigma(n, a)) + c(n_i, \sigma(n, a)) \right]$$

$\sigma(n, a)$是对在节点n采取动作a后到达状态的描述。

这个特殊的学习过程从一个随机步开始，最终慢慢到达目标．在随后的试验中向后传播来自更好路径的更好的 $\hat{h}$ 值。因为在节点n选择的动作会到达一个节点，这个节点被估计是在从n到一个目标的最小代价路径上，因此当更新 $\hat{h}(n)$时没有必要评价n的所有后继。由于模型是逐

渐建立起来的，因此可以把“在环境学习”和“在模型中学习和计划”组合起来。这种组合在[Sutton 1990]的DYNA系统中探讨过。

由于在最佳路径上的节点可能从没被遍历过，因此这种技术会导致agent学到非最佳路径。允许偶然的随机动作(代替那些由学到的策略选择的动作)可以帮助agent学会到达目标的新路径(也许更好)。采用随机动作是一个agent解决“探测与开发权衡”(the exploration (of new paths)versus the exploitation(of already learned knowledge.tradeoff)的一种方式。也有一些其他的方法可以确保所有的节点都被遍历到。

10.3.2　隐式图

现在考虑生成所有节点及其迁移的显式图是不实际的情况。与前面一样，当有动作结果的模型时(即我们有算子，它能把一个状态描述转换成一个后继状态的描述)，我们能执行一个由评估函数导向的搜索过程。典型地讲，这种函数对几个节点可能会产生相同的评估值，因此可以把这样的函数应用到状态描述。然而在一个表中显式保存所有的节点和它们的值可能是不可行的。

使用和第3章的描述相似的方法，在执行搜索过程的同时学习启发式函数。我们先猜想一组子函数，认为它们是启发式函数的组成部分。例如，在8数码问题中，我们可用函数$W(n)$=在错误位置的数码个数，用$P(n)$=每一个数码离“家”的距离之和，还有其他的任何函数，这些函数可能与一个位置离目标的远近有关。然后可以把启发式函数作为这些函数的一个加权组合：

$$\hat{h}(n) = w_1 W(n) + w_2 P(n) + \cdots$$

这里简述两个学习权值的方法。第一种方法，把权值的初始值设为任何我们认为是最好的值，然后使用这些权值构成的 $\hat{h}$ 函数进行搜索。当我们到达一个目标节点n_g时，用最终已知的值 $\hat{h}(n_g) = 0$，顺着到达目标的路径备份(back up)所有节点n_i的 $\hat{h}$ 值。用这些值作为“训练的例子”，把权值调节为训练例子和加权组合的 $\hat{h}$ 函数的平方差之和的最小值。这个过程在n次搜索中迭代执行。

或者我们能用和第一种方法类似的办法——用每一个节点的扩展来调整 $\hat{h}$。在扩展节点n_i产生后继集合$S(n_i)$后，调整权值以使得：

$$\hat{h}(n_i) \leftarrow \hat{h}(n_i) + \beta \left(\min_{n_j \in S(n_i)} \left[\hat{h}(n_j) + c(n_i, n_j) \right] - \hat{h}(n_i) \right)$$

或重新整理为：

$$\hat{h}(n_i) \leftarrow (1 - \beta)\hat{h}(n_i) + \beta \min_{n_j \in S(n_i)} \left[\hat{h}(n_j) + c(n_i, n_j) \right]$$

$0<\beta<1$是一个学习率参数，它控制着 $\hat{h}(n_i)$向 $\min_{n_j \in S(n_i)} \left[\hat{h}(n_j) + c(n_i, n_j) \right]$ 逼近的快慢程度。当$\beta=0$时，没有任何变化。当$\beta=1$时 $\hat{h}(n_i)$等于 $\min_{n_j \in S(n_i)} \left[\hat{h}(n_j) + c(n_i, n_j) \right]$ 小的β值使得学习非常慢，然而β趋于1会使学习过程不稳定且不能收敛。

这种学习方法是时态差分(*temporal difference*)学习的一个实例，时态差分学习由[Sutton 1988]正式提出。权值调节只依赖一个函数的两个时态相邻值[⊖]。Sutton 指出并证明了在各种情

⊖ Sutton 认为[Samuel 1995]是这个思想的创始人。

况下和该方法的收敛性相关的一些属性。关于时态差分方法，令人感兴趣的是在到达一个目标之前它能在搜索过程中应用(但是直到到达目标时，学习得到的 $\hat{h}$ 值才能和目标相关)。这个过程也必须在几个搜索过程中迭代执行。

我们注意到这种学习技术也能应用于没有动作结果模型的情况，也就是说，这个学习能发生在前面讨论过的现实世界中。走一步(也许是一个随机步或者是根据形成的动作策略选择的步骤)，在这一步的前后评估 $\hat{h}$ 的值，记下这一步的代价，并对权值进行调整。

10.4 奖赏代替目标

在讨论状态空间的搜索策略中，假定agent有一个简单的短期任务，它由一个目标条件描述。目标改变着世界，直到它的图标模型(以数据结构的方式)满足给定的条件。在很多实际问题中，任务并不像刚才描述的那样简单。相反，任务可能是正在进行的。用户按照给agent一些偶然的正负奖赏(*reward*)来表达他对任务执行的满意程度。agent的任务是把它收到的奖赏数量最大化(一个简单到达目标的特例也可用这种框架来描述，在这个框架中，当agent到达目标时，给agent一个正的奖赏（只有一次），而每次当agent采取动作时给它一个负的奖赏(根据动作的代价))。

在这种任务环境下，我们要寻找使奖赏最大化的动作策略。对于正在进行还没有终止的任务，存在的一个问题是未来的奖赏可能是无限的，因此难以决定如何使它最大化。一种处理办法是通过一些因子对未来的奖赏打折扣，即agent宁愿要不久将来的奖赏而不愿遥远的未来奖赏。然后，假定agent在每一个时间步采用一个动作(所谓“采用”一个动作，是指在环境中真正地执行一个动作或者在环境的图搜索模型中应用一个算子)。每个动作使状态描述产生一个改变——这个改变或者是真的被agent感知到的，或者是在模型中应用一个算子计算得到的。

设n代表agent的状态空间图的一个节点，π是节点上的策略函数，节点的值是由节点上的策略描述的动作，设$r(n_i, a)$是agent在节点n_i采用了动作a时agent收到的奖赏。如果这个动作产生了节点n_j，一般地，我们会有$r(n_i, a)= -c(n_i, n_j)+\rho(n_j)$，$\rho(n_j)$ 是到达节点n_j的任何特定的奖赏值。有些策略会比其他的策略产生对未来奖赏更大的折扣，我们寻求一个最优策略π^*,它能使每个节点的未来折扣奖赏最大化。

给定一个策略π，我们能为在状态空间中的每个节点n估算一个值$V^{\pi}(n)$；如果agent从n开始，并采用策略π，$V^{\pi}(n)$是agent得到的总的折扣值。假如我们在节点n_i，采用动作$\pi(n_i)$，产生了节点n_j，就得到

$$V^{\pi}(n_i) = r[n_i, \pi(n_i)] + \gamma V^{\pi}(n_j)$$

$0<\gamma<1$是折扣因子，它通过在t_i时刻的奖赏值计算在$t=t_{i+1}$时刻的奖赏，$\pi(n_i)$是由节点n_i的策略π描述的动作。对最优策略π^*，我们有:

$$V^{\pi^*}(n_i) = \max_a \left(r[n_i, a] + \gamma V^{\pi^*}(n_j)\right)$$

也就是说，n_i在最优策略下的值是直接奖赏加上最优策略下n_j 的折扣值(乘以γ)的和的最大值(注意，n_j是动作a的函数，这使$V^{\pi^*}(n_j)$也成为a的函数)。如果知道一个最优策略下的节点值(所谓的最佳值(*optimal value*))，我们就能用下面的方式写最优策略:

$$\pi^*(n_i) = \operatorname{argmax}_a \left(r[n_i, a] + \gamma V^{\pi^*}(n_i)\right)$$

问题是我们一般不知道这些值。但是有一个叫做值迭代(*value iteration*)的学习过程，它将(在一定条件下)收敛到那些最佳值。

值迭代的工作过程如下：开始，我们给每个节点n分配一个随机的估计值(*estimated value*)$\hat{V}(n)$。假如在过程中的某一步到达节点n_i，节点n_i的估计值是$\hat{V}(n_i)$。然后我们选择动作a，它取直接奖赏和后继节点估计值之和的最大值(假定有动作结果模型上的算子，并且这些算子能应用到节点产生的后继节点)。假定动作a将我们带到节点n_j。那么如下更新节点n_i的估计值$\hat{V}(n_i)$：

$$\hat{V}(n_i) \leftarrow (1-\beta)\hat{V}(n_i) + \beta\left[r(n_i, a) + \gamma\hat{V}(n_j)\right]$$

所有其他节点的估计值保持不变。

我们看到这种调整给$\hat{V}(n_i)$一个接近$[r(n_i, a)+\gamma\hat{V}(n_j)]$的增量(依赖β)。在一定程度上，$\hat{V}(n_j)$是$\hat{V}^{\pi^*}(n_j)$的一个好的估计，这个调整有助于使$\hat{V}(n_i)$成为$\hat{V}^{\pi^*}(n_i)$的一个更好估计。

值迭代常常用于动作有随机结果和产生随机奖赏(两者都由概率函数描述)这样一些更普通的情况。尽管如此，倘若$0<\beta<1$，并且我们经常无限地遍历每一个节点，值迭代将收敛到最佳值——可能情况下的希望值(在确定性领域，我们总能用$\beta=1$)，这个结果是令人惊奇的，因为我们正在探索使用(也许一个坏的)最优策略评估的空间，同时我们正试图在一个最优策略下学习节点的值。对值迭代和有关的过程以及它们和动态编程的关系的完整讨论可以参见[Barto, Bradtke, & Singh1995]。

在奖赏依靠早期动作的序列框架中，学习动作策略被称为延迟加强学习(*delayed-reinforcement learning*)。当奖赏被延迟时会出现两个问题。第一，必须确信那些状态——动作对要对奖赏负责。完成此任务就是所谓的时态信用分配问题(*temporal credit assignment problem*)。值迭代是传播信用的一个方法，以便加强合适的动作。第二，当状态空间太大以致于不能存储整个图时，必须用相似的$\hat{V}$值聚集状态。完成此任务被称作结构信用分配问题（*structural credit assignment problem*）。神经网络和其他的学习方法对解决该问题也是有用的。在[Kaelbling, littman, & Moore 1996]中很好地讨论了延迟加强学习。

10.5　补充读物和讨论

感知/计划/动作循环是Agre和Chapman 所称的交叉计划(*interleaved planning* [Agre & Chapman 1990])的一个实例。他们比较了交叉计划和他们提出的即席创作。他们的话如下(Agre&Chapman 1990,p30)：

交叉计划和即席创作的不同在于它们对故障的理解。在交叉计划的领域中，一个人会假定事件的正常状态是事件按照计划应该到达的状态。也就是说故障是一个少量现象。在即席创作领域，一个人会假定事情不可能按照计划进行下去。从完全相反的结论中，一个人希望必须继续重新决定该做什么。

但是，负责将感知系统和电机系统设计(记着任务和环境)好，以便“故障”实际上成为次要的，这些难道不是agent设计者的事吗？在T-R 树[Nilsson 1994]形式中的计划对处理次要故障是相当健壮的。关于交叉计划和执行的更多内容，参见[Stentz 1995, Stentz & Hebert 1995, Nourbakhsh 1997]。

关于岛搜索问题参见[Chakrabarti, Ghose, & DeSarkar 1986]，对于层次计划建模和分析，分别参见[Korf'1987, Bacchus & Yang 1992]。[Stefik 1995, pp. 259~280]对层次计划提供了一个

非常清晰的说明。

在有限范围搜索中，一个人必须决定“范围”。这种决定必须考虑在附加计算值和已做计算推荐的动作值之间权衡。这个权衡受动作延迟的代价影响。评价更多计算和紧接动作的有关值是元级计算的一个实例。在[Russell & wefald 1991.第5章]中详细地介绍了这个主题。他们的DTA* 算法实现了关于这个主题的一些思想。

[lee & Mahajan 1988]描述了关于学习评估函数的一些方法。延迟加强和时态差分学习方法同随机的动态编程紧密相关。参考[Barto,Bradtke,&Singh 1995,Ross 1988]。基于奖赏学习动作策略的机器人系统的例子可以参考[MahadeVan & Connell 1992]和[Connell & Mahadevan 1993b]中的文章。[Moore &Atkeson 1993]介绍了有效的基于加强存储的方法来控制物理系统。[Montague, et al.1995]介绍了一个基于加强学习的蜜蜂找食物的模型，[Schultz, Dayan, & Montague 1997]描述了灵长类的神经系统是如何实现时态差分学习机制的。

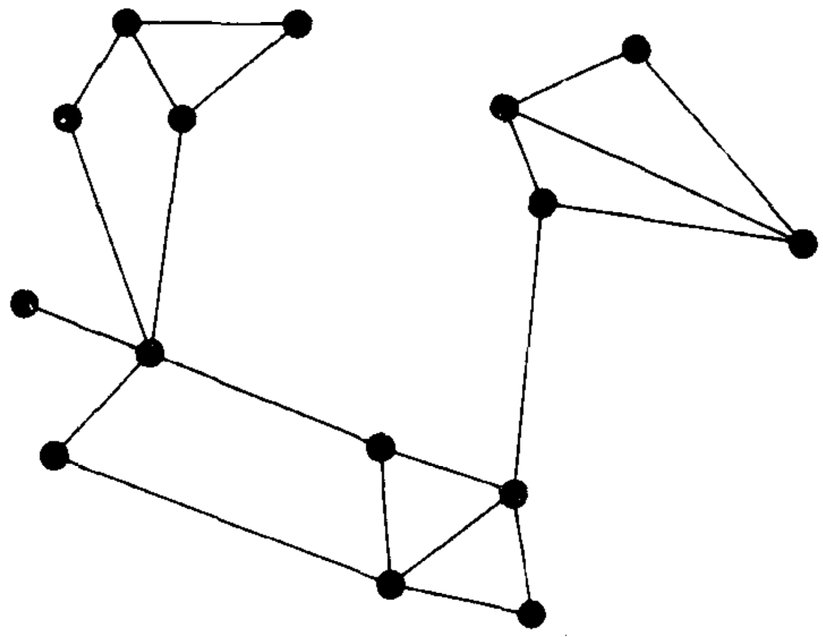
习题

10.1 在孤岛搜索中不是直接找到到达目标的路径，而且一个人首先确定一个岛，它接近于在初始节点和目标节点的中途上。我们首先要找到通过这个岛节点到达目标的一条可接受的路径(先找到从开始节点到该岛节点的一条路径，再找从该岛节点到达目标节点的一条路径)；如果不能找到可接受的途经岛节点的路径，我们只要解决原始问题就行了。假如搜索只能向前进行——即朝着目标节点进行。

- 假定对任何值d，搜索一棵分枝因子为b，深度为d的树需要的时间是kb^d，确定一个合适的岛的时间是c,岛在一条到达目标的可以接受的路径上的可能性是p。找到关于p和c的条件以便用孤岛方法需要的平均时间比普通的广度优先搜索的时间少。
- 给出一个孤岛搜索可能节约时间的搜索问题的例子。

10.2 考虑N个有名称的积木问题，这些积木可以有任意初始的配置，它们必须放在一个任意的目标配置中(忽略积木的水平位置，只考虑它们在堆中的垂直安排)。假如所有的N个积木都在地板上作为一个状态孤岛。根据搜索、存储器、时间要求和解答长度分别讨论这个方法的实现。

10.3 解释对下面的图如何建立一个层次搜索过程。希望从图中的任何节点到标为“目标”的节点找到一条路径(尽管层次搜索对这样小的一个图根本不必要，但在考虑大图中的层次搜索时，这个习题会给你一个锻炼的机会)。

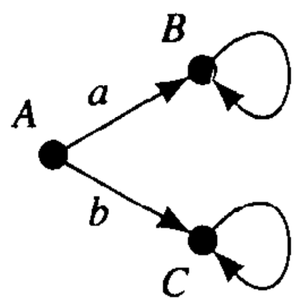


10.4 想像一个agent 面对着一个非常大的状态空间。假如状态空间图有b^d 个节点，b是平均分枝因子，d是状态空间图中随机选择的两个节点之间的平均路径长度。我们希望比较两种设计策略：一个agent 从任意的起始状态到达一个任意的目标状态。一种策略用**IDA***在运行时计划一条路径，即当给定目标和初始状态时，agent用**IDA***计算一条路径。另一个策略对所有的目标状态和起始状态提前计算和保存所有可能的路径。按照它们的时间和空间复杂度比较这两种策略(对很大的d值，你可以用适当的近似值)。

10.5 你在一个陌生的城市有了一份新工作，目前正和那个城市的一位朋友呆在一起。每天早晨，他驱车送你到城中的一个地铁站，你必须从那儿乘车去工作(那位朋友是城中的递送人员，在你们相处期间他会把你送到很多不同的车站)。地铁站(有有限个站)是一个方形网格布局。其中一个称为中心站，它是你去工作必须到达的一个站。你总能知道你到达了中心站。在每个站，你有四辆火车可以选择：北、东、南、西。每辆火车都是局部的，它只能将你带到网格中一个相邻的站，在那儿你必须下车，再搭乘另一辆车继续。一些相邻站点之间的连接永久地坏掉了，但是你知道从网格中的任何一个站到中心站仍然有一些其他的路径。每到一个站必须付1美元，你每天会从工作中得到100美元。你没有路线图，不知道各个站相对于中心站的任何位置信息。你决定用值迭代方法开发一个在每个站点要乘火车的策略。值迭代似乎是合适的，因为你总是知道你当前所在站点的名字，能从该站点乘坐哪些火车，还有这些火车到达的站点的名字。

- 描述一下这个问题中的值迭代如何工作。需要一个时态折扣因子吗？为什么需要或不需要？如果当你旅行时调整了站点值，如果你用这些值选择火车，在每次旅行中你将在中心站点结束，证明上述问题(提示：你的算法可以调整值，以便在任何试探中你从来不会通过一些不包括中心站点的站点子集进行一个无穷的旅行)。
- 如果你的学习算法不能保证产生最终会产生最佳(最短)路径的值，解释一下为什么不能，并应采取什么办法来保证最佳性。

10.6 考虑下面的状态转换图。从开始状态A有两个可能的动作。动作a到达状态B，并产生一个直接的奖赏0。动作b到达状态C产生直接奖赏1000。一旦到了状态B，只有一个可能的动作。它产生一个+1奖赏并回到B。在状态C，也只有一个可能的动作。它产生一个-1奖赏并回到C。为了采用动作a而不是b，需要什么样的时态折扣因子？



第11章　其他搜索公式及其应用

有一些搜索技术的应用超出了为一个agent选择动作的应用范围，这些应用包括为受限变量分配值和为解决最优问题等寻求解决方案。有一些专门的方法已经被开发用于这些应用中，虽然它们看上去并不直接与agent设计有关，但它们也是重要的AI技术，本章将讨论其中的一部分。

11.1　赋值问题

图搜索问题中的目标节点的条件可以定义为一个指定的数据结构或者标识它的状态描述，或者它可以根据对那个数据结构的条件和约束隐式地定义。在任一情况下，当问题是找到agent的一个动作序列时，标识目标节点的数据结构并不重要，重要的是到达目标的步骤序列。或者当目标节点不是由一个指定的数据结构定义而是由条件或约束定义时，问题可能就是要显示一些满足这些条件的数据结构，而用图搜索方法产生它的步骤可能是不相关的。我们称这类问题为约束满足(*constraint-satisfaction*)问题。这类问题中的一个著名例子是给受限变量赋值。它们被称为赋值问题(*assignment problem*)，下面将讨论这个问题。

我们能通过图搜索方法解决约束满足问题。一个目标节点是由满足约束的数据结构(或状态描述)标识的节点。算子将一个数据结构改变为另一个数据结构。开始节点是一些初始的数据结构。赋值问题的一个典型例子是8皇后问题(*Eight-Queens problem*)。问题是这样的：把8个皇后放在一个棋盘上，每行和每列只能有一个皇后，另外，一个皇后只能在任一行、列或对角线上(也就是说，按照国际象棋规则，没有皇后能被放在一个位置以便它能抓住任何其他的皇后)。这个问题的一个解法参见图11-1。由于这种问题具有从集合{第1行，第2行，...，第8行} 到变量{第1列的皇后位置，第2列的皇后位置，…，第8列的皇后位置}的赋值形式，所以称为赋值问题。

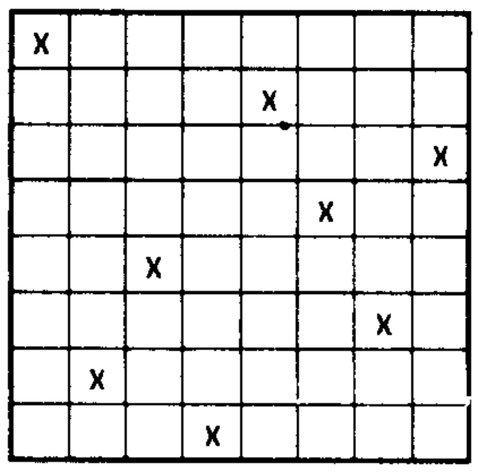


图11-1　8皇后问题的解决方案

把这个问题作为一个图搜索问题，一个明显的数据结构是一个8×8的数组，数组中的每个单元包含两个符号(1和0)中的一个，1代表皇后，0为空。一个根据条件定义的隐式目标状态是保证8个皇后都是安全的，不会被抓住。连接状态描述的算子和数组转换的方式相一致。例如，一个算子能把一个皇后加到还没有8个皇后的数组中，或者它能把一个皇后移到另一个单元中。在赋值问题中，由于到达目标的路径并不重要，因此关于开始状态和算子是什么我们常有很多选择。

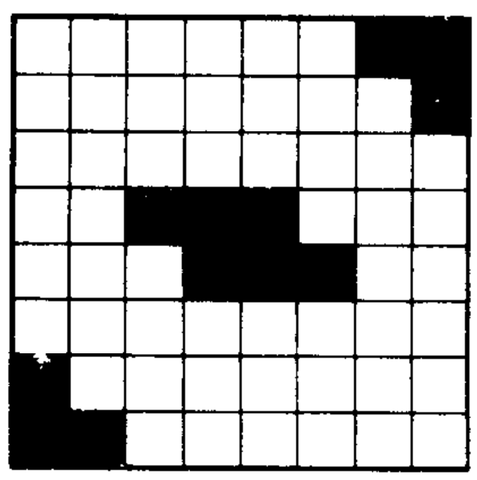


图11-2　纵横字谜的一个数组

作为赋值问题的另一个例子，考虑一下有很大状态空间的纵横字谜问题[⊖]。图11-2 显示了

⊖ 关于纵横字谜的产生问题，Matt, Ginsberg和他的学生已经测试了各种状态空间搜索技术 [Ginsberg et al. 1990。]

一个纵横字谜的数组。问题是根据纵横字谜规则用字母填充数组中的所有空白块，以使所有的行和列都是英语单词(没考虑这个问题的另一方面，即对企图解决纵横字谜的人创建提示)。在这个问题中，一个状态描述是字母和空格(和已涂黑的单元一起)组成的任何数组，一个目标状态是一个合理纵横字谜解法的任何数组。连接状态描述的算子能将字母和空格的组合转变为另一种数组。例如，一个算子能把一个单词加到一个空行或空列上，或者它能把一个字母变为另一个字母。

11.2　构造性方法

我们可以用上一章讲的搜索方法解决赋值问题。最直接的方法是一步一步地构造所要求的赋值——尽管我们对那些步骤不是太感兴趣。开始（开始节点）不进行任何赋值，对8皇后问题，相应的数据结构是全为0的一个数组。每个算子加一个皇后到数组中，但要以满足约束的方式加入。因为在每一列必须有一个皇后，不失一般性，我们能保证在深度为0的节点应用算子产生第1列皇后的放置，应用到深度为1的节点的算子是第2列皇后的放置，等等。因为这个解法是一步步构成的，我们称这个方法为构造方法(*constructive method*)（后面将提到一个与之相对应的方法）。

图11-3中给出了搜索树的一部分，它能用于8皇后问题和纵横字谜问题的构造方法（为了易读性，用*X*而不是1和0来表示一个皇后的位置）。特别注意，算子用来从前一个节点产生节点。尤其对纵横字谜，很明显状态空间是巨大的，在每个节点我们有成千上万的算子可以应用。

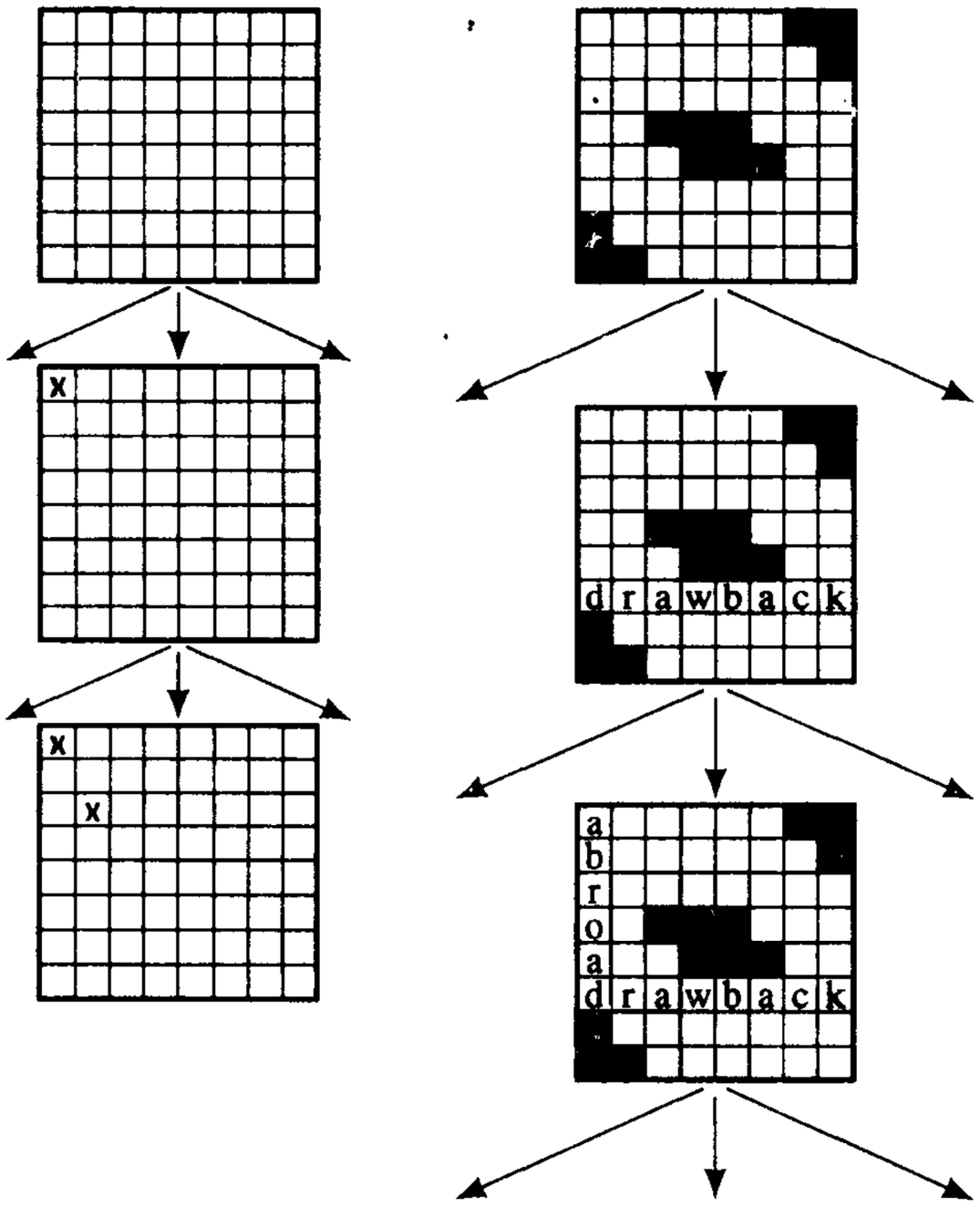


图11-3　构造性公式表示的状态空间

一种称为约束传播(*constraint propagation*)的计算技术有助于显著地减少搜索空间的大小。它和构造技术组合使用——依次给每一个变量赋值。我在如何应用8皇后问题的简化版本中描述了这个技术。考虑4个皇后放在一个4×4棋盘中，彼此不能互相抓住。在4皇后问题中，我们有4个变量q_1、q_2、q_3和q_4，分别代表4列中的某一列，一个皇后可以放在它们的其中之一上。每个变量可以是1、2、3和4这4个和行数相对应的值中的一个。例如，当q_3等于2时，一个皇后被放在第2行第3列。4皇后问题对这些变量的值提出了约束，因此如果q_1等于1，q_2就不能等于1或2。

约束被表示在一个叫约束图(*constraint graph*)的有向图中，在这个图中的每一个节点有一个变量和该变量的一组可能值标识。一个有向约束弧(*constraint arc*)（i，j），连结着节点i和j，条件是节点i的变量值受节点j的变量约束。图11-4表示了针对4皇后问题的这样一个图的例子。在这个问题中，每个变量约束着所有其他的变量，因此所有的节点之间都有弧存在。如果对弧尾的每一个变量值至少有一个弧头的变量值没有违反约束，我们就说有向弧（i，j）是一致的。图11-4中的弧是一致的，因为对每对q_i和q_j(i不等于j)，对q_i的每一个值，有一个q_j值没有违反约束。

给变量中的一个或几个赋值后，我们能用弧一致性概念排除其他变量的一些值。约束传播过程在图中的弧上进行迭代，试用加强弧一致性来排除弧尾上的变量值。当没有更好的值能被删除时这个过程终止。现用一个例子阐明这个过程。假如一个深度优先搜索过程通过给变量q_1赋值1开始（即我们把一个皇后放在第1行第1列）。应用到这个赋值的约束传播如下向前推进：

1) 考虑弧（q_2，q_1）：排除q_2=1和q_2=2，因为q_1值（我们刚分配的值）和q_2的这些值不一致。

2) 考虑弧（q_3，q_1）：排除q_3=1和q_3=3，因为q_1的值和q_3的这些值不一致。

3) 考虑弧（q_4，q_1）：排除q_4=1和q_4=4，因为q_1的值和q_4的这些值不一致。

在约束传播的这一阶段（不完全），我们有图11-5。在图随后的约束传播过程中，排除q_3的所有值，因此看到q_1=1时这个问题没有解法。在q_1=1的情况下不能再进行进一步的搜索，而是回溯到q_1=2的情况。

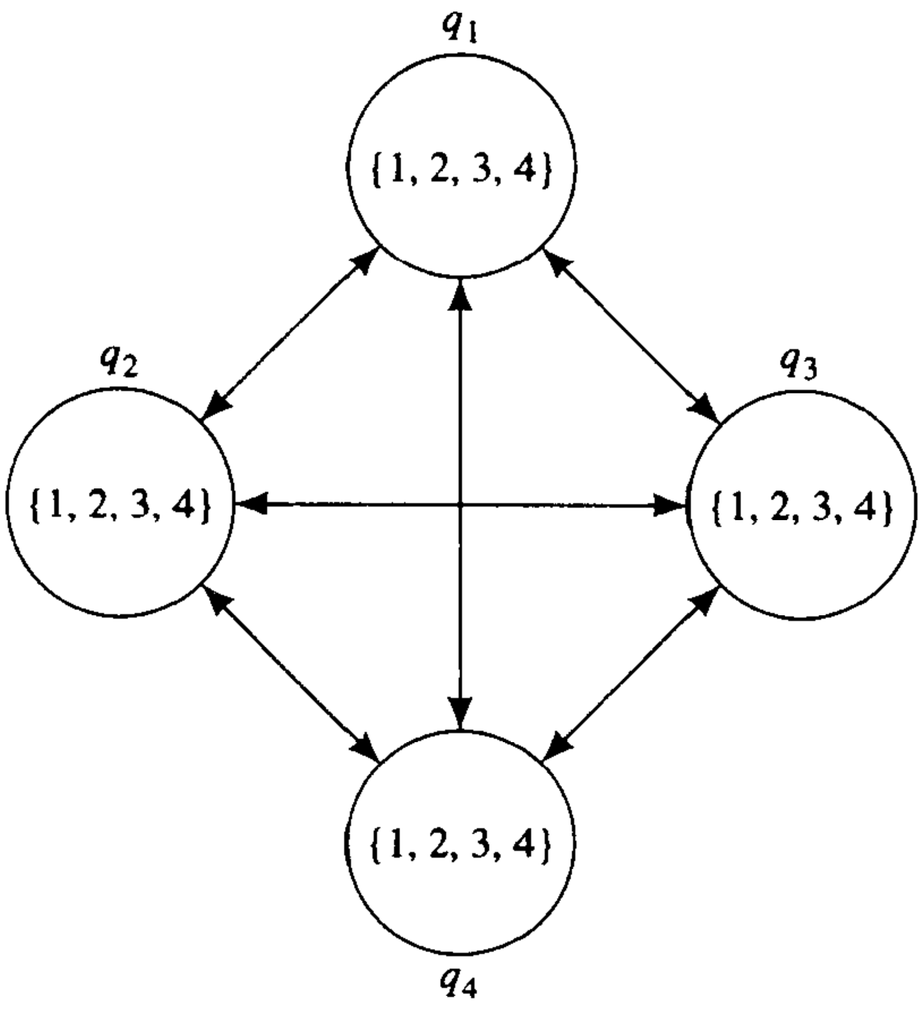


图11-4 4皇后问题的一个约束图

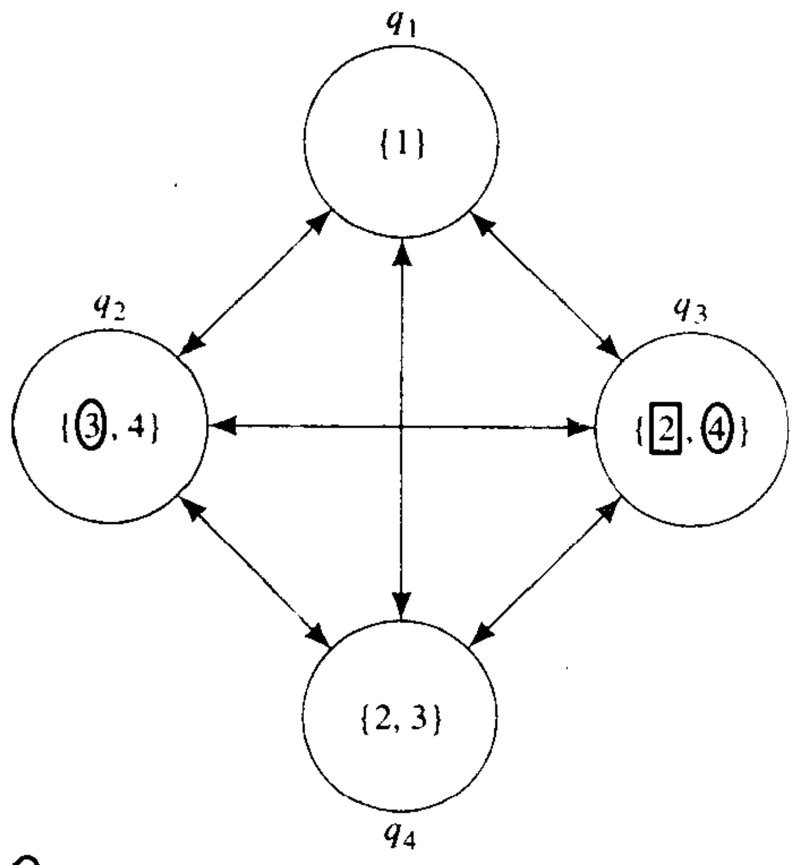


图11-5 q_1 = 1时的约束图

下面，我们假定q_1 =2进行约束传播。开始几步如下：

1) 考虑弧（q_2, q_1）：排除q_2 =1, q_2=2和q_2=3。

2) 考虑弧（q_3, q_1）：排除q_3 =2和q_3 = 4。

3) 考虑弧（q_4, q_1）：排除q_4 =2。

我们得到图11-6。连续的约束传播使每个节点变量只有一个值，使所有的弧保持一致。因此，我们看到在完成搜索前只有一个解法，约束传播已经找到了那个解法。

就像例子中一样，如果采用前面赋给变量的值，约束传播有时会排除几个变量的所有值——使得一个搜索问题没有解法。有时用先前给变量的赋值，可能只有一个一致的解法。在这些情况下，约束传播取消进一步搜索。约束传播的基本思想已扩展到包括对一致性更复杂的测试，但是它的大部分经济型应用可能涉及到刚才示例的一致性检查。约束传播已经应用到各种有趣的问题中，包括在可视地形分析中用 +、 −、或→标识线和本书后面讨论的命题满足问题。为了更好地研究这个方法及其扩展和应用，可参考[Kumar 1992]。

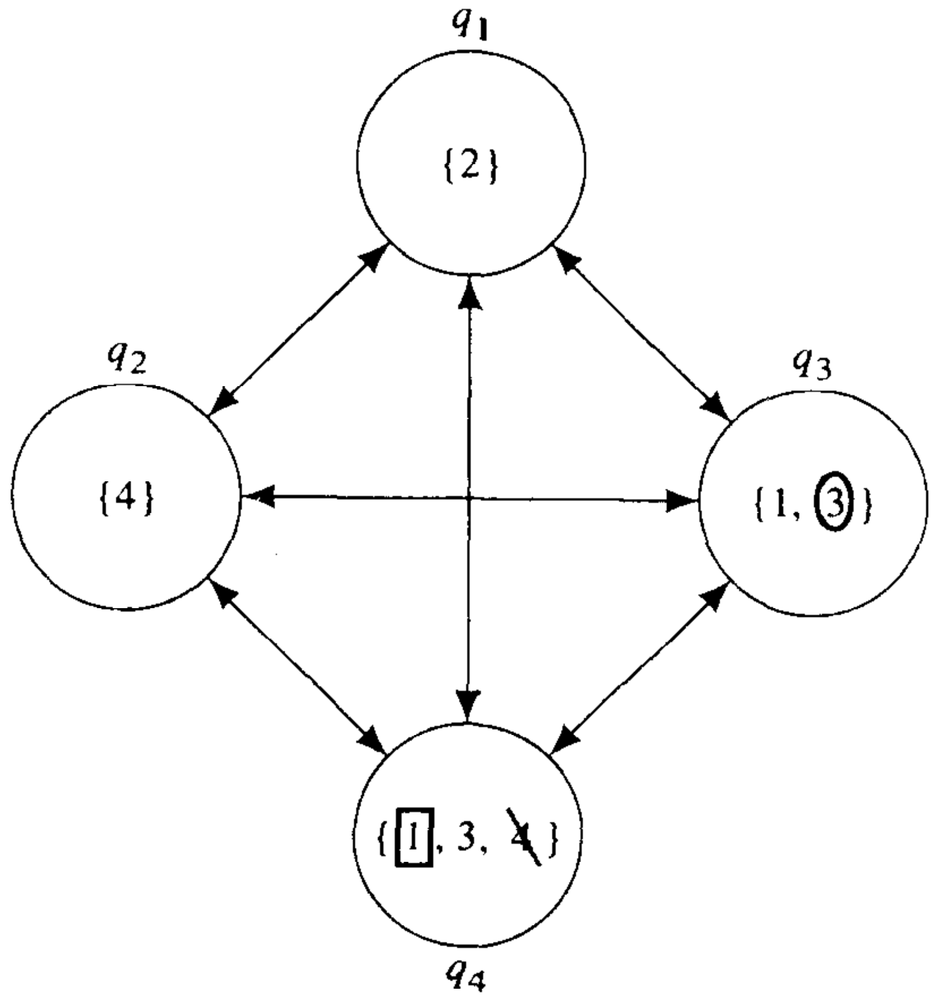


图11-6　q_1 = 2的约束图

11.3　启发式修补

存在另外的方法来为图搜索方法的解法设置一个问题。这个方法叫修补方法(*repair approach*)，因为它用一个提议的解法开始，这个解法一般不满足约束条件，修补这个解法直到它满足约束条件。因此，初始节点一般由不满足所有约束的一个数据结构表示。算子产生一个新的数据结构，它相对应于一个不同的提议解法。

例如在8皇后问题中，开始时8个皇后每列一个，行位置是任意的，也许是随机的。我们通过移动一个皇后以违反更少的约束来修补有缺陷的解法。在所谓的最小冲突(*min-conflict*)[Gu 1989, Minton, et al. 1992]修补方法中，我们依次考虑每一列（从第一列开始），用进攻那个单元的皇后数（在那一列的外面）标识出那一列的每个单元。然后，将那一列的皇后移到有最小进攻皇后数的单元（最小冲突数）。彼此间的联系被随机打破。这个算子产生一个后继节点，它由被提议算法的一个轻微修改所标识。这样依次通过每个列。在图11-7中，用最小冲突示例了8皇后问题深度优先搜索的一部分。再一次，皇后位置用*X*表示，单元中的数字表示进攻该单元的皇后数。类似的基于修补的方法[Minton, et. al 1990]已被用于解决更大的问题，象百万皇后问题[⊖]。

当把修补方法应用于纵横字谜问题中时，开始节点可以是完全充满字母的任何组合。我们可以把一个组合中的某个字母换成其他的字母来修补一个有缺陷的解法，以使最终每行和每列

⊖ 尽管图标约束扩展和基于修补的方法是有趣的，但对*N*皇后问题碰巧也是一个线性时间算法[Abramson & Yung, 1989]

都是一个单词。因为在每个节点，有成千上万的其他算子可以应用，因此纵横字谜的搜索空间是巨大的。

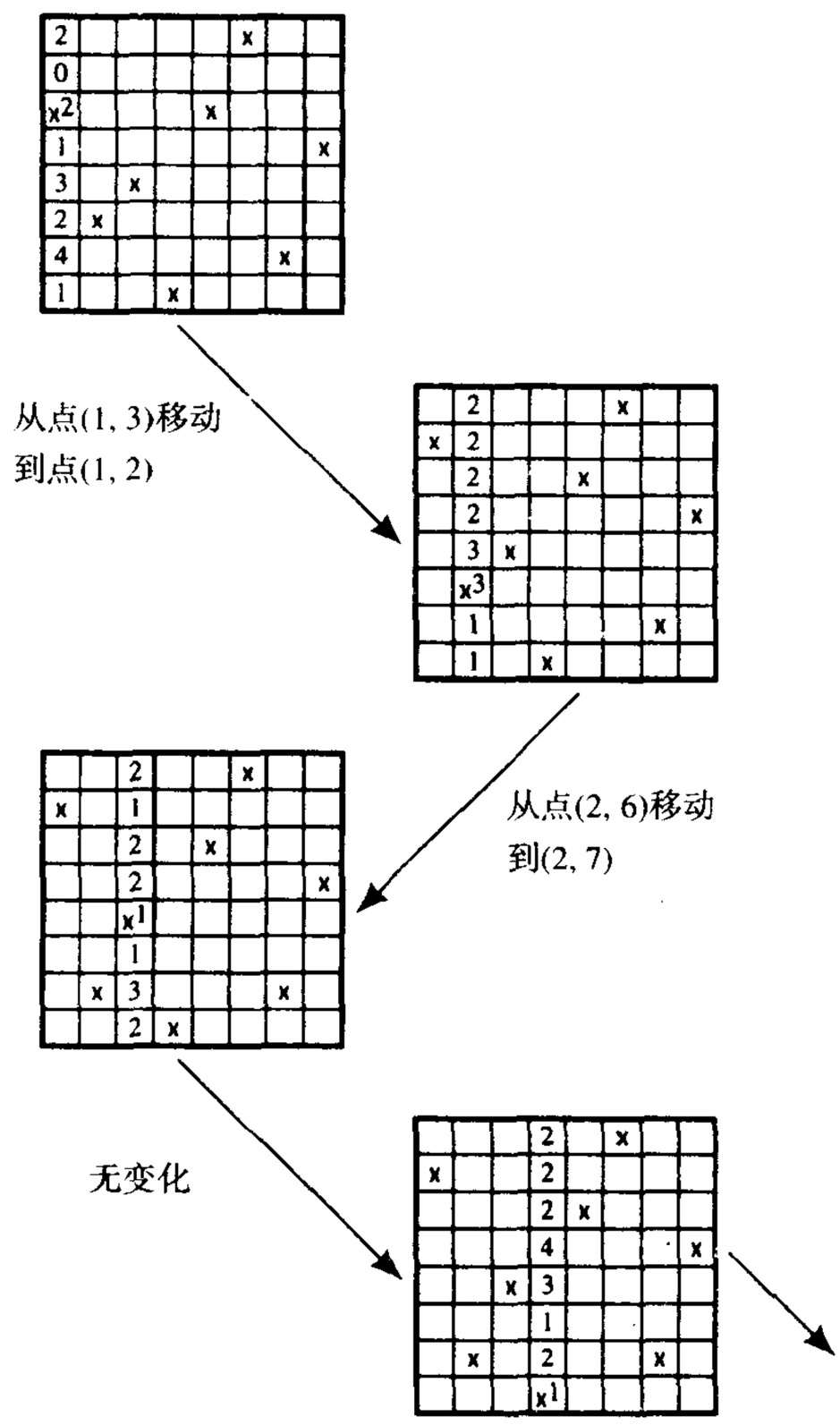


图11-7 8皇后问题的修补步骤

我们应该用构造方法还是修补方法，能使用的状态和算子是什么，所有这些对搜索的困难度都会有很大的影响。

11.4 函数优化

在某些问题中，我们没有一个显式的目标条件，而可能只有建立在一些数据结构上的函数 v，我们要找到一个结构，它有这个函数的最大（或最小）值。如果把这些数据结构当做空间中的一些“点”，这个函数就能被作为空间上的一个“地形”。一类方法就是遍历这个地形，在它当中寻找最高的点。由于不知道全局最大值，因此我们可能已找到了最大值时却不能肯定。

遍历一个空间的技术叫爬山方法(*hill-climbing*)，它从一个点遍历到“相邻”的有最高高度的点[⊖]。当相邻节点没有比当前点更高的高度时，爬山方法终止——因此，它们可以得到一个

⊖ 当寻找一个函数的最小值时，严格地讲，应该称其为下山方法。但一般地讲，将用爬山方法描述这两种活动。在任何情况下，通过乘一个因子 -1，我们能把一个问题转化为另一个问题。

局部最大值。

我们能用图搜索方法解决爬山问题。同样，节点用数据结构标识。算子能把一个给定的数据结构改变为一个相邻的结构。爬山顺着一条路径（更像没有回溯的深度优先搜索）估计高度，它决不会下降到一个更低的点。下面是找到一个有（局部的）最大值的节点的爬山算法。

爬山算法：

1) 把当前节点n设为随机选择的节点n_0；

2) 产生n的后继（用给这个问题定义的算子），选择后继n_b，它的值$v(n_b) = v_b$是这些后继中最大值。（任意断开联系）；

3) 如果$v_b < v(n)$，将n作为到目前发现的最好节点，并退出；

4) 否则，将n设为nb，转到第2步。

注意，除了我们总是扩展有最高值的后继（只要它的值不比它的父节点小）和没有提供回溯外，这个算法非常像深度优先搜索。爬山搜索中的移动是不能取消的。

在一个3×3的网格中用给单元填色示例了爬山方法（在这个问题中，山的高度其实在下降）。给出一个由任意的红色和兰色网格单元构成的网格，要找到一种配色以使得一个单元的颜色和其相邻单元颜色相同的数量达到最小。也就是说，在状态空间中我们要找到一个节点n^*，对所有的n, ,$v(n^*) \le v(n)$, $v(n)$是节点n中有相同颜色的相邻单元数。对算子而言，允许任何单元的颜色可以从红色变为蓝色，反之也行。图11-8表示了该问题的一部分图和在下山过程中采用的路线。

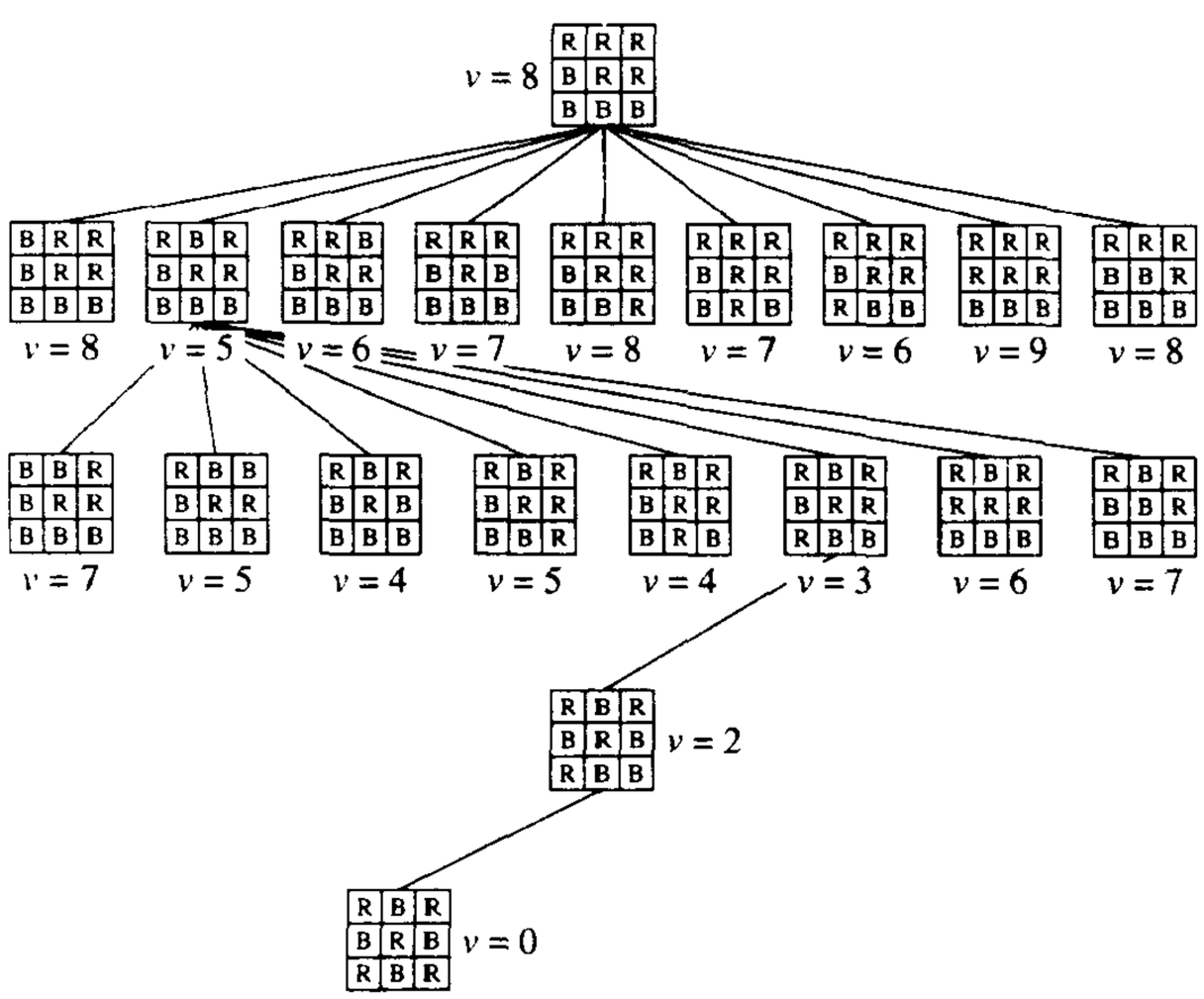


图11-8　两颜色问题解法

如果一次移动不能使v的值改变，我们就说在搜索空间中移到了一个高地。对于爬山算法，在高地上无休止地徘徊是可能的，在这种情况下，算法重复地遍历前面访问过的节点而不能爬到一个更高的地带。通过增加一个跟踪v值保持不变的次数计数器可以限制这种徘徊。以附加存储为代价，可以通过拒绝把以前访问过的有相同值的节点作为当前节点来改进高地问题。

在搜索空间中，另一个困难由“山脊”（或它的反义词“山沟”）引起。在这种情况下，任何存在的移动都会把我们带到一个更低的节点，但是顺序执行的两次移动将会增加高度。该问题如图11-9所示。每一个可能的移动都使我们离开山脊到达一个更低的高度，但是两个移动序列会使

我们移动到一个更高的山脊。山脊问题有时可通过将两个或更多的移动组合成一个“宏移动”来避免，或者通过一个有限数量的“向前”搜索来避免。

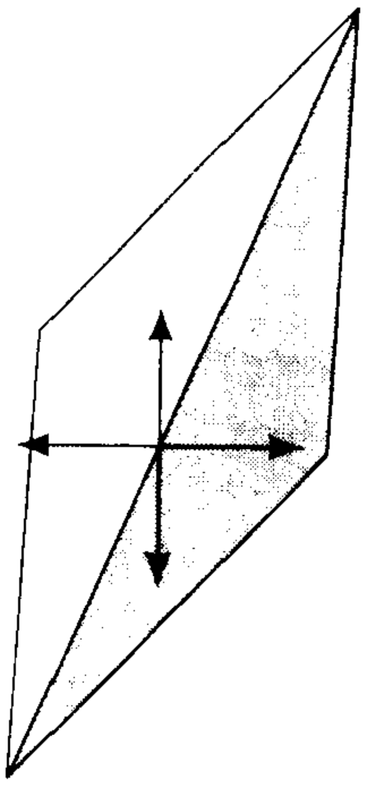


图11-9 山脊问题

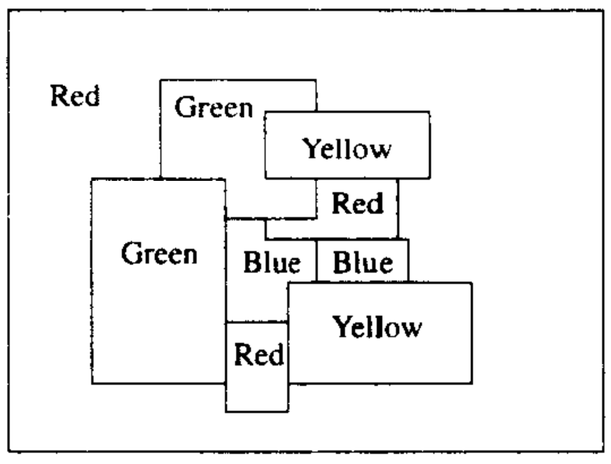
我们可以通过从不同的位置开始执行几个独立的爬山搜索（并行或顺序的）来解决这个问题，以找到局部最大值。这些搜索中的每一个都可能结束在不同的局部最大值，我们可以用这些值中的最大值。第4章讨论的GP方法是一种随机的爬山方法。在这个方法中，几个“爬山者”并行地工作，通过建立后代爬山者进行移动，被优化的函数是适应性函数。可以用各种普通爬山方法[Juel & Wattenberg 1996，O' Reilly & Oppacher 1994]来比较GA和GP方法的效率。

一个叫做模拟退火(*simulated annealing*)的过程[Kirkpatrick，Gelatt，& Vecchi 1993]也有助于解决极端的局部问题。这个过程有多种版本。在一种版本中，根据可用移动的概率分布选择移动，分布促成向较低高度（下山）的节点移动。用一个几乎可以忽略偏差的分布开始这个过程，使它向着这些偏好的节点移动，逐步增加偏差，直到最后移动可能以极大的概率向着一个较低高度的节点进行。最终的结果是：在过程的开始，我们在这个地形上随机地移动。最终，如过程保证的话，我们会下降到一个山谷之中。如果这个山谷不是非常深，也不是非常宽，不久一个后继的随机移动将把我们从这个山谷中推出。不可能从一个宽阔的（因此可能很深）山谷移出，在这个过程结束时（没有任何随机移动），我们下降到它的最深点。这个过程因类似于冶金中的退火过程而得名，在冶金过程中一个材料的温度被逐渐降低以允许它的晶体结构达到一个最小能量状态。在模拟退火过程中，在概率分布中控制宽度的参数常叫做温度。

习题

11.1 参照第6章习题6.4中的图。解释一下如何用一个约束图和约束传播为这个图找到一个一致的标识。

11.2 用启发式修补方法解决下图中有4种颜色的着色问题，以便没有任何相邻的两个区域有相同的颜色。用下面的图开始。



11.3 用最小冲突方法解决4皇后问题。开始时，4个皇后都在主对角线上。随机地打破连接！（手工完成这个问题可能是冗长乏味的，但是这很值得做）。

11.4 如果你还没有做习题7.3（传教士和食人者问题），现在就做它。假设函数f = 在河的开始一侧的总人数。这个问题能用基于f的下山方法解决吗？能或不能，为什么？

第12章　敌对搜索

12.1　双agent博弈

在第10章中提到的问题之一是在其他主动agent参与的环境中如何计划并行动。如果不了解其他agent如何行动，就只能用感知/计划/动作体系结构，而这种结构不能更加深入地考虑到不可预测的将来。但是，当条件允许时，一个agent建立的计划可以考虑到其他agent的行为影响。以双agent的特殊情况为例，在理想的情况下，如这两个agent的行为是互相交替的，它们可以考虑到对方的行为。其中之一先行动，然后是另一个，接着如此反复。

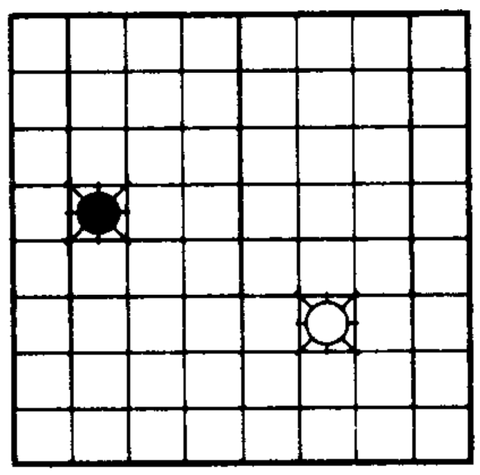


图12-1　在网格环境中的两个机器人

以图12-1中的网格为例，两个机器人，分别命名为“black”和“white”。它们可以向其所在的行或列中的相邻一格交替地移动（比如说，white先移动），而且轮到其中一个时，它必须移动。假设white的目标是与black在同一格，而black的目标是避免发生这种情况。white就可建立一棵搜索树，在交替的级别上，black可能的行动也被考虑进去。图12-2画出了这棵搜索树的一部分。

为了选取最佳的起始动作，white需要分析这棵树来决定可能的结果——即：考虑black会阻止white实现此目标。在这种冲突的情况下，一个agent可能发现一步移动，使得无论对方如何移动，它都可达到目标。不过更常见的情况是，由于计算和时间的限制，无论哪方都不能找

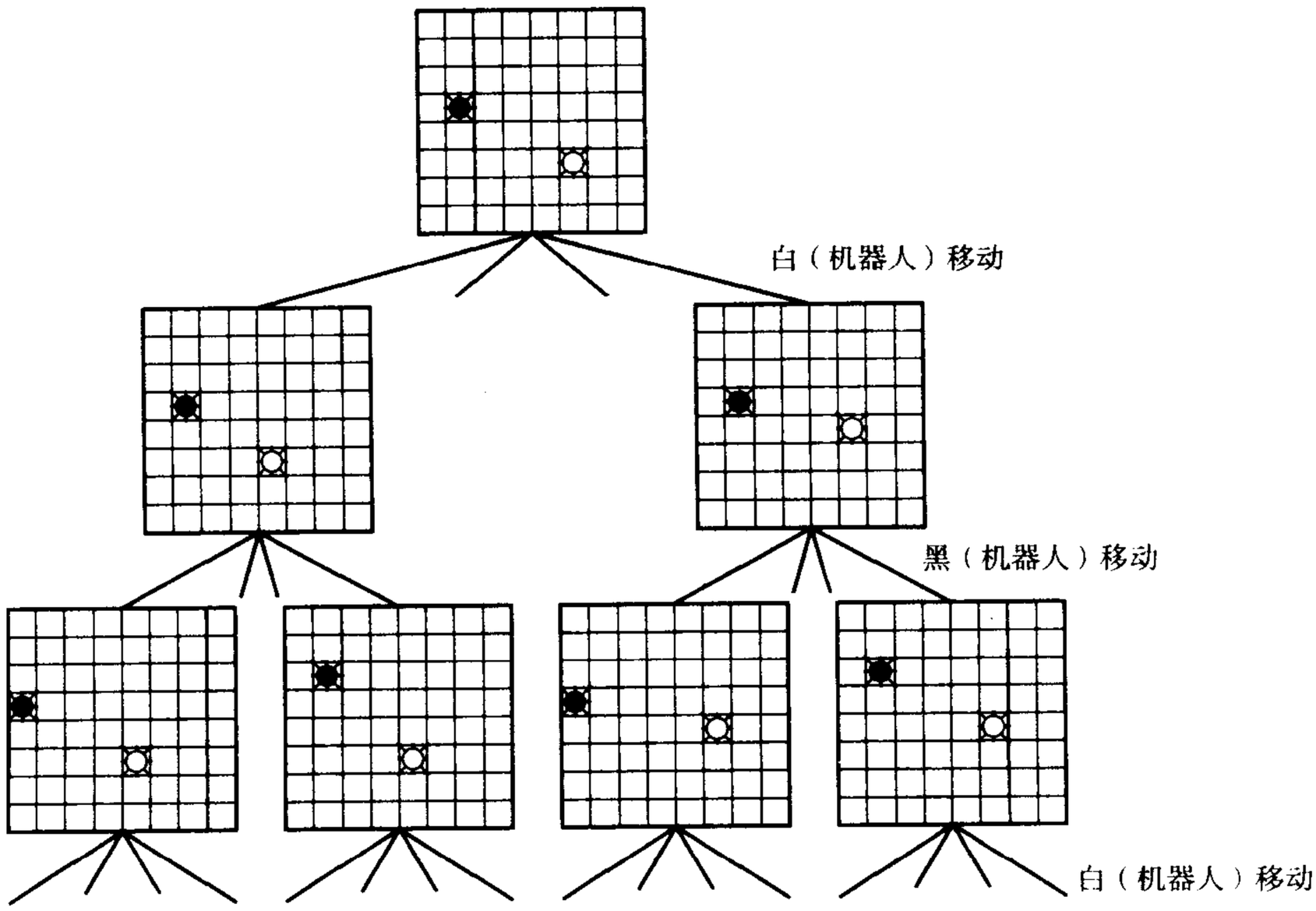


图12-2　两个机器人移动的搜索树

到一个动作以保证取得成功。本书将提供有限范围搜索方法，可用于在这种情况下找到合理的移动方式。在任何情况下，在决定了第一步之后，执行移动，考虑另一方的可能移动，然后在感知/计划/动作方式中重复计划过程。

这个网格例子是双agent、信息完全，零和（*zero-sum*）博奕的一个实例。此处所讨论的是，两个agent（称为博弈者）轮流移动，直到其中任何一方获胜（另一方因此失败），或双方和局。每个博弈者完全熟悉环境及自己和对方可能的移动方式和影响（尽管每个博弈者都不知道另一方在任何情况下究竟会怎样移动）。研究这种博弈可使我们深入了解当有多个agent时，计划过程可能出现的更多的普遍问题——即使在这些agent的目标并不互相冲突时。

可以看出，有许多常见的博弈，包括国际象棋、西洋跳棋(dranght)和围棋，都属于这种类型。而且它们的程序已经编写得相当好——有的甚至可以达到参赛的水平。但此处，以并不十分有趣的井字博弈（*Tic-Tac-Toe*）为例，因为它简单，有利于分析搜索技巧。有些博弈（例如西洋双陆棋，*Backgammon*）由于包含了概率因素而导致难以对它们进行分析。

对许多博弈，特别像国际象棋这一类，通常都用图标来描述自己的状态空间，我们用8×8数组来记录黑白机器人在8×8网格中的不同位置[⊖]；用算子表示博弈的移动，算子把一种状态描述转换为另一种描述，由一个开始节点和每个博弈者的算子隐式定义博弈图，照前面章节的方法建立搜索树，但在选择第一步时要使用一些不同方法。下面将讨论这些方法。

12.2 最小最大化过程

在以下讨论中，命名两个博弈者*MAX*和*MIN*。我们的任务是为*MAX*找最佳的移动。假设MAX先移动，然后两个博弈者轮流移动。因此，深度为偶数的节点，对应于*MAX*下一步移动的位置，称为*MAX*节点；深度为奇数的节点对应于*MIN*下一步移动的位置，称为*MIN*节点（博弈树的顶节点深度为0）。*k*层包括深度为$2k$和$2k+1$的节点。通常用层数给出博弈树的搜索程度，它可以表示出向前预测的*MAX*和*MIN*交替运动的回合数。

正如我所说，完全的搜索（赢、输或和局）对于大多数博弈来说是不可行的。据估计完全的国际象棋博弈图解大约有10^{40}个节点。即使假设一个后继节点可在1/3 ns[⊖]内产生，也需要10^{24}年才能产生国际象棋博弈完全搜索图解（据推测，宇宙也只有大约10^{10}年的历史）。而且，启发式搜索方法并不会减少起作用的有效分枝因子。因此，对于复杂的博弈，必须认识到搜索到终点是不可能的（除了在博弈快结束时），所以，应该使用类似于在第10章中描述的有限范围搜索方法。

我们可使用广度优先搜索、深度优先搜索或启发式搜索，除非必须马上修改终止条件。几个人为的终止条件为时间限制、存储空间限制以及在搜索树中最深节点的深度。例如，通常在国际象棋中，如果有任何叶节点代表可继续的好位置，即存在可以继续移动的好位置，就不终止。

搜索结束后，需从搜索树中选取一个最佳首次移动，这个选取方法可以对搜索树的叶节点

⊖ 然而，在这种情况下，当轮到White移动时，一个更好的表示是基于机器人之间的距离是奇数还是偶数，这个留给读者自己思考。奇偶校验在一对移动后进行。

⊖ 相反，Schaeffer 估计西洋跳棋的完整游戏图只有大约10^{18}个节点[Schaeffer & Lake 1996]。因为*MAX*只要在每一层出示到达 解法的一个移动就行了，这样，一个解法树将只有这个数的平方根即10^9个节点。Schaeffer 和 Lake 认为“在不久的将来”，完美的机器西洋跳棋游戏将是可得到的。

采用静态评估函数。此评估函数衡量每一个叶节点位置的“值”。这种衡量基于影响这个值的许多不同特性；例如，在西洋跳棋中，一些有用的特性衡量相关部分优势、中心控制、王的中心控制等等。通常分析博弈树时，对*MAX*有利的位置，评估函数将赋于正值；对*MIN*有利的位置赋于负值，接近零的值表示该位置对*MAX*和*MIN*都一样。

一个最佳首步可以由一个最小最大化过程产生（为简单起见，在描述这个过程和基于它的其他过程时，把博弈图当作一棵树）。假设轮到*MAX*从搜索树的叶节点中选取，他肯定选择拥有最大值的节点。因此，*MIN*叶节点的一个*MAX*节点双亲的倒推值就等于叶节点的静态评估值中的最大值。另一方面，*MIN*从叶节点中选取时，必然选值最小的节点（即最负的值）。既然如此，*MAX*叶节点的*MIN*双亲节点被分配一个倒推值，它等于叶节点静态评估值的最小值。在所有叶节点的父节点被赋于倒推值后，开始倒推另一层，假定*MAX*将选择有最大倒推值的*MIN*后继节点，而*MIN*会选择有最小倒推值的*MAX*后继节点。

我们继续逐层对节点评估，直到最后开始节点的后继者被赋于倒推值。假定*MAX*首先移动，*MAX*将选择有最大倒推值的节点作为它的首步。

整个过程的有效性基于这样的假设：开始节点的后继的倒推值比直接从静态评估函数中得到的值更可靠。由于倒推值基于在博弈树中的预先推算，并且取决于在博弈结束时发生的一些特性，这些值往往更加切合实际。

井字博弈的简单例子阐述了最小最大化方法（在井字博弈中，博弈者在3×3数组中轮流标记，一个标记（X），一个标记（O）。先用标记填满一行、一列或一条对角线者便赢得博弈）。假设*MAX*标记（X），*MIN*标记（O），*MAX*先开始。在深度为2的范围内进行广度优先搜索，直到第二级节点全部产生，然后在这些节点代表的位置采用静态评估函数。位置p的静态评估函数$e(p)$可如下给出：

假如对任何一方，p位置都不是取胜位置

$e(p)$=（对*MAX*开放的完整的行、列或对角线数）-（对*MIN*开放的完整的行、列或对角线数）

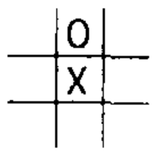
假如对*MAX*来说，p是取胜位置，

$e(p)=\infty$(用∞来表示一个非常大的正数)

假如对*MIN*来说，p是取胜位置，

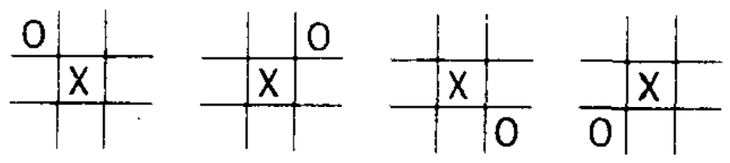
$e(p)=-\infty$

所以，假如p为：



我们就得到$e(p)=6-4=2$。

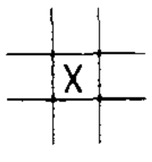
在产生后继者位置时，采用对称法；因此，以下状态是相同的：



（博弈初期，井字博弈的分枝因子由于对称而很小；在后期，由于可用开放空间的数量而仍很小）。

图12-3说明了深度为2的搜索产生的树，静态评估表示在叶节点的右边，倒推值被圈起，

既然在



中有最大倒推值，它被选为首次移动（假如进行完全搜索，这同样也是*MAX*的最佳首步）。

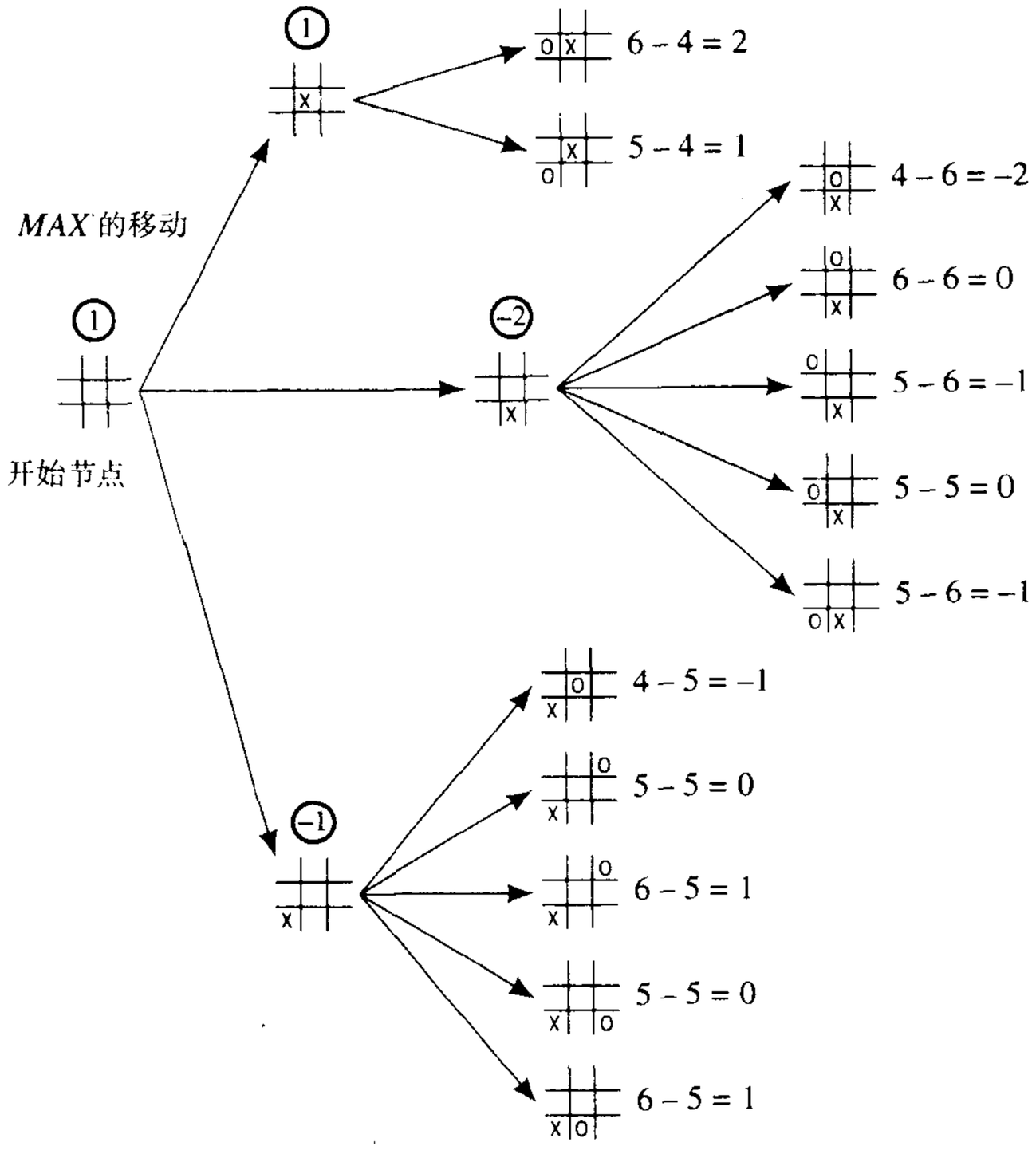
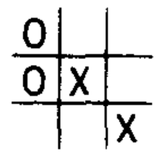


图12-3 井字博弈搜索的开始阶段

现在，按照感知/计划/动作的循环，假设*MAX*走了这一步而且*MIN*在（X）的左边做标记（O）（对*MIN*来说，这不是一个好的走法，它没有用一个好的搜索策略）。在这种布局下，*MAX*进行深度为2的搜索，产生如图12-4所示的树。现在有两个可能的最佳走法，假定*MAX*走其中之一。*MIN*移动以免自己马上失败，形成



*MAX*再次搜索，产生如图12-5所示的树。

该树的一些叶节点（如*A*）代表*MIN*的胜利，估计值为 $-\infty$。当倒推这些估计值时，可以看出*MAX*的最佳移动也是惟一可以避免马上失败的一步。现在*MIN*明白*MAX*下一步将取胜，于是只好投降。

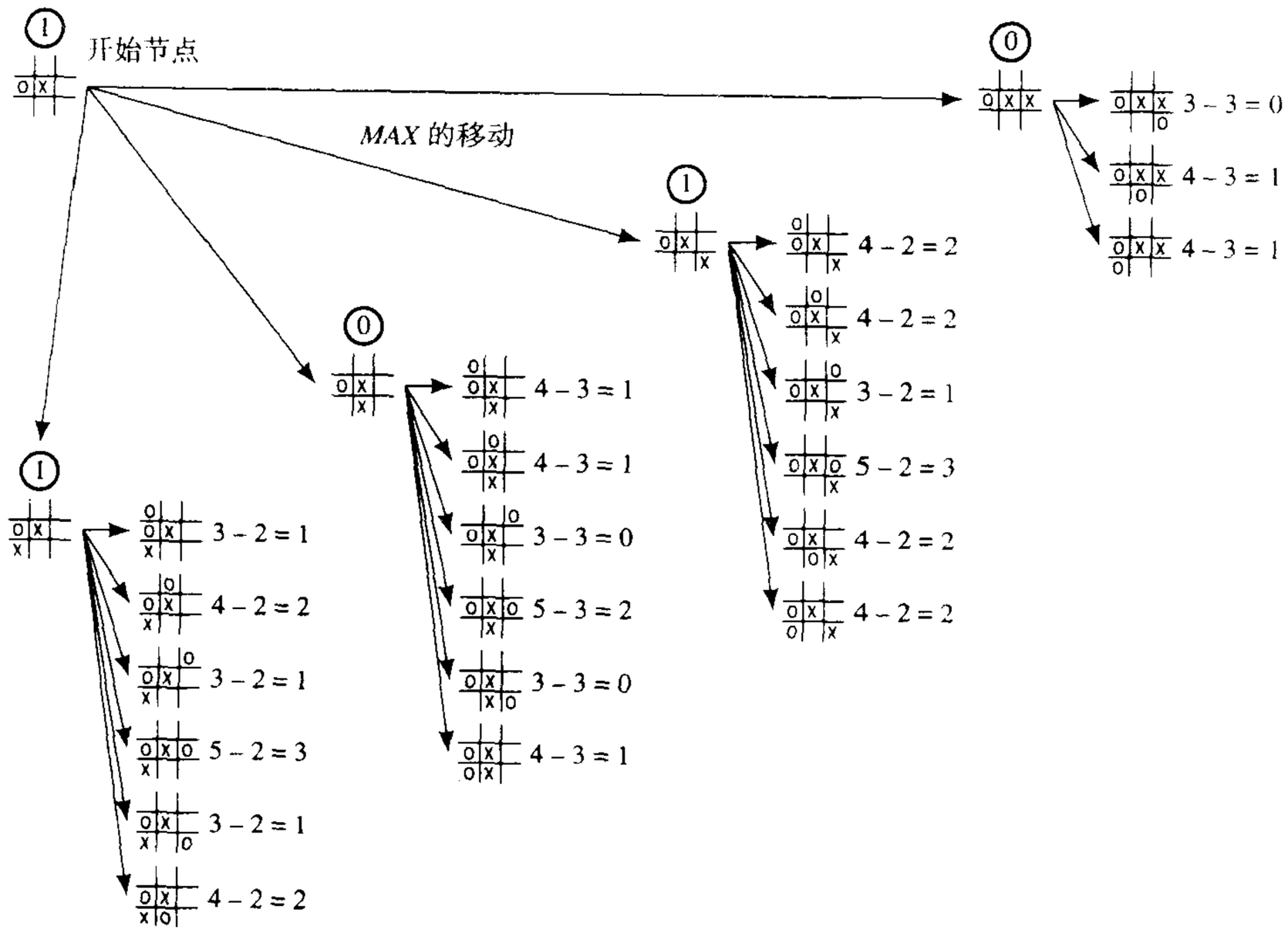


图12-4　井字博弈搜索的第二阶段

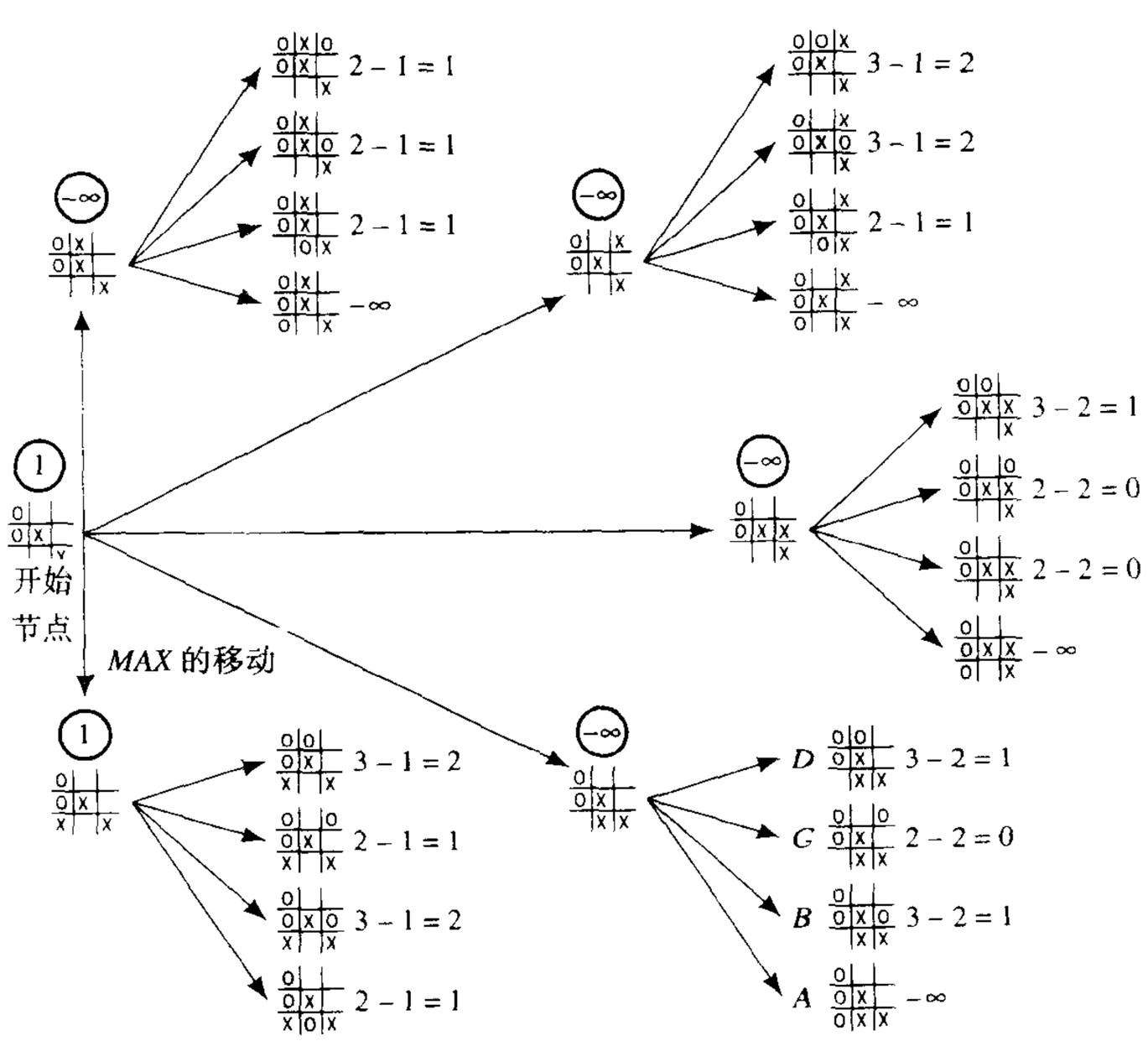


图12-5　井字博弈搜索的最后阶段

12.3　α-β过程

刚才叙述的搜索过程将搜索树的产生和位置评估完全分开，只有在树完全产生后才能对位

置评估。这种分离导致算法粗糙和效率低。假如将叶节点的评估、计算倒推值与树的产生同时进行，就可能大量减少所需搜索的数目（有时非常大），这个过程类似于第10章提到的α截断。

考虑井字博弈搜索中最后阶段的搜索树（图12-5）。假定一个叶节点一产生便被评估，那么在产生节点A并评估后，便没有必要产生（并评估）节点*B*、*C*和*D*；即由于*MIN*可以选*A*，并且只能选*A*，我们立即知道*MIN*肯定选择*A*。于是将*A*的父节点的倒推值赋于-∞，继续搜索，不再去费力地产生节点*B*、*C*、*D*并评估（注意，当做更深的搜索时，这种节省尤为明显，因为同样也不需产生节点*B*、*C*和*D*的后代）。重要的是要注意到不产生*B*、*C*和*D*，决不会影响*MAX*的最佳第一步的产生。

在这个例子中，搜索节约取决于代表了*MIN*获胜的节点*A*，然而，在搜索树中没有表示任何一方胜利的节点时，也能同样节省搜索操作。

考虑前面表示的井字博弈树的开始阶段（图12-3），在图12-6中重复了这棵树的一部分。假定采用深度优先搜索，而且节点一产生，便被静态评估，并假定一旦一个位置可给于倒推值时，这个值便被计算出来。设想这样的情况：*A*节点及其所有后继都已产生，而*B*节点还未产生，*A*节点的倒推值为－1。这时可知开始节点的倒推值范围大于等于－1。根据开始节点的其他后继的倒推值，开始节点的最终倒推值可能大于－1，但决不会小于－1，我们把这个下界称为开始节点的α值。

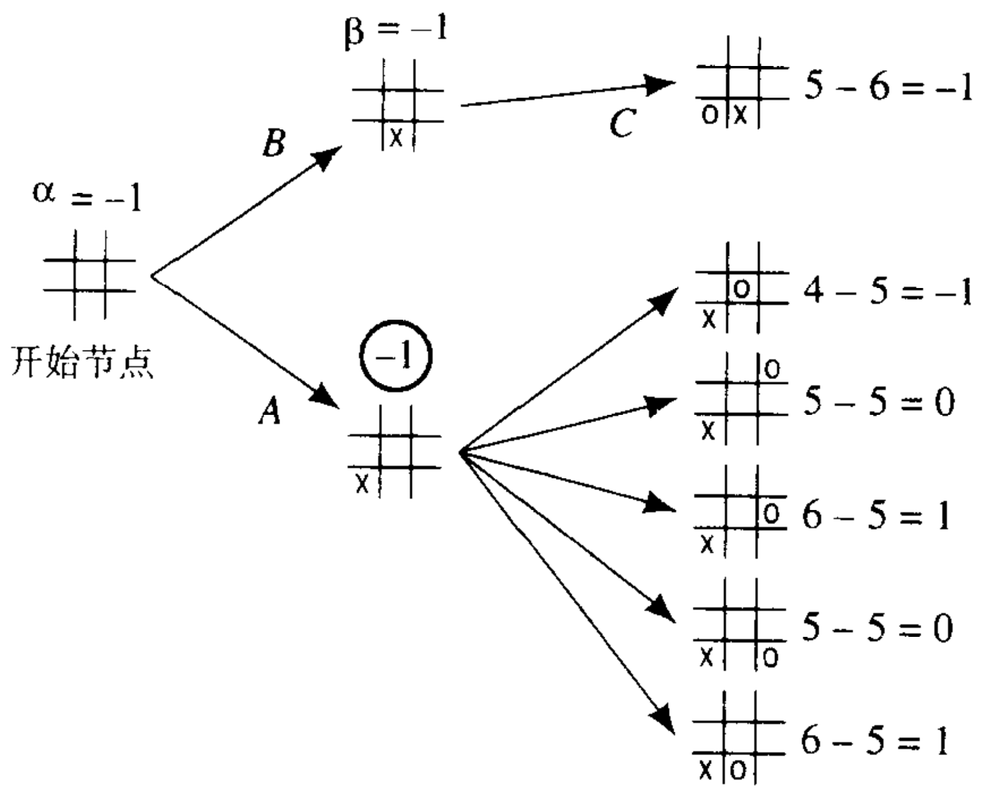


图12-6 井字博弈搜索开始阶段的一部分

继续深度优先搜索，直到节点*B*及它的第一个后继节点*C*生成，*C*节点的静态值为－1，现在可知*B*节点的倒推值限制在－1之下，*B*节点的其他后继节点的静态值可能会使*B*的最终倒推值小于－1，但决不会大于－1。我们称这个上界为*B*节点的*β*值，这时可以看出*B*的最终倒推值不会超过开始节点的*α*值。因此，可以不再继续*B*节点以下的搜索，从而可以肯定*A*节点不会选择*B*节点为后继节点。

通过记录倒推值的界限可减少搜索。通常，当一个节点的后继节点有了倒推值之后，倒推值的界限就会被修改，但我们注意到：

- *MAX*节点的*α*值（包括开始节点）决不会减小。
- *MIN*节点的*β*值决不会增加。

由于有这些限制，可以用以下规则来减少搜索量：

1）当任何*MIN*节点的*β*值小于等于它的*MAX*的祖先节点的*α*值，则可中止该*MIN*节点以下的

搜索。该*MIN*节点的最终倒推值即为它的β值。该值与完全最小化搜索得到的值不等，但同样可以选择到相同的最佳移动方式。

2) 当任何*MAX*节点的α值大于等于它的*MIN*的祖先节点的β值时，则可中止该*MAX*节点以下的搜索，该*MAX*节点的最终倒推值等于它的α值。

在搜索中，如下计算α和β值：

- *MAX*节点的α值等于它的后继节点中的最大最终倒推值。
- *MIN*节点的β值等于它的后继节点中的最小最终倒推值。

当按规则1停止搜索时，称产生α剪枝；当按规则2停止搜索时，称产生β剪枝。保存α、β值、并且当可能时进行剪枝的整个过程通常称为α-β过程。当开始节点的所有后继节点都得到倒推值后，过程终止。最佳首步移动是产生有最大倒推值的后继节点的移动。该过程产生的移动方案与简单使用极大极小过程搜索相同深度得到的方案是相似的。惟一的不同是，α-β过程通常用少得多的搜索就可找到最佳首步移动。

上述α-β过程可用一个简洁的伪码递归算法表示。下面的例子评估了节点n相对于α和β截断值的最大最小值，摘自[Pearl 1984，P. 234]:

AB（n; α, β）

1) 如果n在深度约束上，返回AB(n) = n的静态值。否则，令n_1，…，n_k，…，n_b为n（按顺序）的后继节点，令k←1，如果n是一个*MAX*节点，转第2步：否则，到第2'步。

2) 令α← max[α，AB(n_k; α，β)]。

3) 如果$\alpha \geq \beta$，返回β;否则继续。

4) 如果$k = b$，返回α；否则前进到n_{k+1}，例如，让$k \leftarrow k+1$，转到第2步。

2') β← min[β，AB(n_k; α，β)].

3') 如果$\beta \leq \alpha$，返回α;否则继续。

4') 如果$k = b$，返回β; 否则前进到n_{k+1}例如，让$k \leftarrow k+1$，转到第2'步

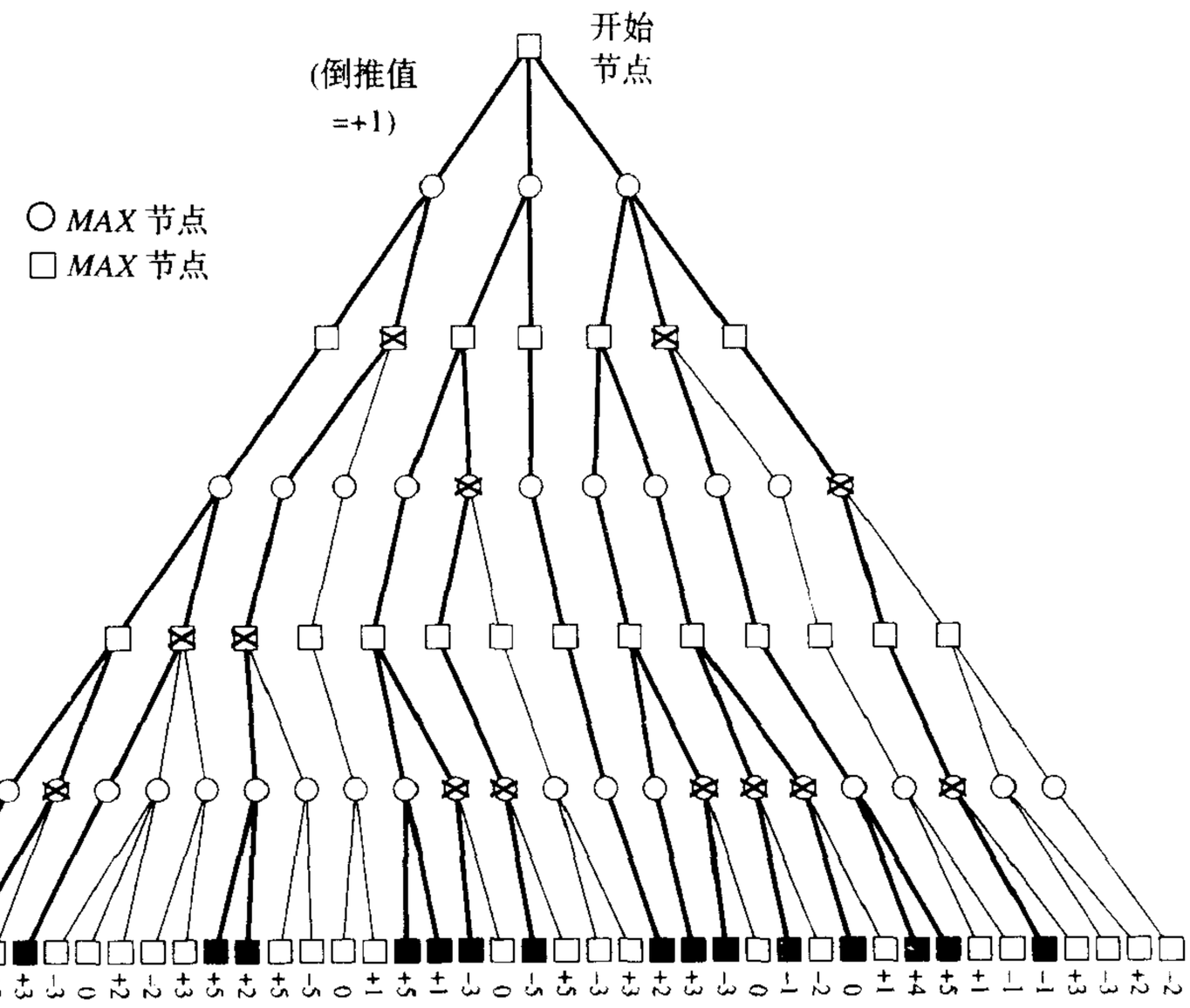


图12-7　α–β搜索的一个例子

通过调用AB（s；$-\infty$，$+\infty$）来开始α-β过程，s为开始节点。在整个算法中，$\alpha<\beta$。正如我们将会看到的，算法第1步中节点的顺序对算法的效率有很大影响。

图12-7说明了α-β过程的应用之一。画出一棵6层的搜索树，方形表示*MAX*节点，圆圈表示*MIN*节点，叶节点注明静态值。假定用α–β过程进行深度优先搜索（我习惯先产生最底层节点）。α–β过程产生的子树用黑色分枝标明，剪枝节点中画*X*，原先41个叶节点中只有18个需评估（可以试着重复此例中α–β搜索来检测对该过程的理解）。

12.4 α-β过程的搜索效率

为了完成α和β剪枝，至少部分搜索树要产生至最大深度值，因为α、β值需要基于叶节点的静态值。因此，使用α-β过程时通常用到深度优先搜索，而且，搜索中可以得到的剪枝数取决于早期的α、β值与最终倒推值的近似程度。

假设一棵树深度为d，并且每个节点（除了叶节点）都有b个后继节点，这样一棵树大约有b^d 个叶节点。假定α-β过程产生的后继节点按其倒推值大小排列——对*MIN*节点按从小到大顺序，对*MAX*节点按从大到小顺序（当然，它们的倒推值是不可能在后继节点产生前知道的，因此，除非偶然，否则该顺序不可能得到）。

这种排序可以使剪枝的数目最大化，并使产生的末端节点数最小化。我们用N_d表示叶节点的最小值，可如下表示[Slagle & Dixon 1969, Knuth & Moore 1975]：

$$N_d = \begin{cases} 2b^{d/2} - 1 & d \text{ 为偶数} \\ b^{(d+1)/2} + b^{(d-1)/2} - 1 & d \text{ 为奇数} \end{cases}$$

也就是说，优化的α-β搜索产生的深度为d的树，其叶节点的数目等同于不用α-β搜索时深度为$d/2$ 产生的叶节点数。因此，当不用α-β的搜索进行到$d/2$ 深度时，α-β搜索（最佳排序）将进行到d 深度，换而言之，用最佳排序的α-β搜索，减少了大约从b到 $\sqrt{b}$ 的有效分枝因子。

当然，最佳节点排序是不可实现的（假如可实现，就完全不需要搜索过程了）。最糟糕的情况下，α-β搜索不会产生剪枝，有效分枝因子不变化。Pearl指出，假如后继节点随机排列，α-β搜索深度会增加约4/3倍，即平均分枝因子减少到大约 $\sqrt[4]{b^3}$ [Pearl 1982b]（见[pearl 1984，第9章]详细分析及注解）。实际上，假如将好的试探方法用于排列后继节点，α-β过程通常可以最大程度地减少有效分枝因子。

排列后继节点最直接的方法是使用静态评估函数。节点排序的另一种技巧来自迭代加深的副效应——迭代加深首次运用在chess博弈程序CHESS4.5中[slate & Atkin 1977]。程序首先搜索一层，然后评价最佳移动；然后又搜索两层，评价最佳移动；如此继续。这种多重搜索看似浪费的主要原因是博弈程序通常有时间限制，由于可用时间资源的关系，向更深层次的搜索可能随时终止，搜索得到被认为最佳的移动。这种副效应在节点排序中体现为，在*K*层搜索中被认为最佳的节点，可用于在*K*+1层搜索中排列节点。

12.5 其他重要问题

对于大多数博弈，必须在找到终止点之前结束搜索（有限范围内），这就导致了许多困难。其一，搜索可能终止在使*MAX*（或*MIN*）能实现大的移动的位置。因此，许多博弈搜索方法确保一个位置为静止，直到在这个位置上搜索结束。如果一个位置的静态值与它前面一两步之内的倒推值相差不大时，称之为静止的。

即使沿一定分枝在终止搜索前保持静止，在搜索范围外，还存在许多情况难以预料。有些博弈树存在一些情形，会不可避免地产生*MAX*获胜，也就是说，无论*MIN*如何走，一定范围内都不会影响*MAX*的胜利。这种所谓的范围效应就是博弈树面临的基本困难之一。

最大最小法和α-β方法都假设对手会选最佳位置移动。在有些情况下这是不合适的。例如，假设*MAX*看起来要输了——假定*MIN*走最佳步，然而，*MAX*依然可以摆脱困境，只要*MIN*犯个错误。最大最小法不会推荐这种走法，然而，由于博弈看来无论如何都会输，那么猜测*MIN*会出错也不会带来任何损失。假如博弈一方有另一方的策略模型，最大最小法也不合适，也许，另一方并非是对手而只是另一个变化的agent。

12.6　概率博弈

有些博弈，如西洋双陆棋（*Backgammon*），包含了概率因素。例如，一方允许的移动取决于骰子的投掷结果，图12-8表示了这种博弈的博弈树（为了使这棵树简单些，只表示出投掷模型的6种可能的不同结果）。*MAX*和*MIN*的每次移动都牵涉到投掷一次骰子。假想在每一次投掷后，一个想像的第三方博弈者，骰子，也做了一次移动。这种移动由概率决定，投掷时，6种结果都是等可能的，但概率因素也可能涉及任意的一种概率分布。

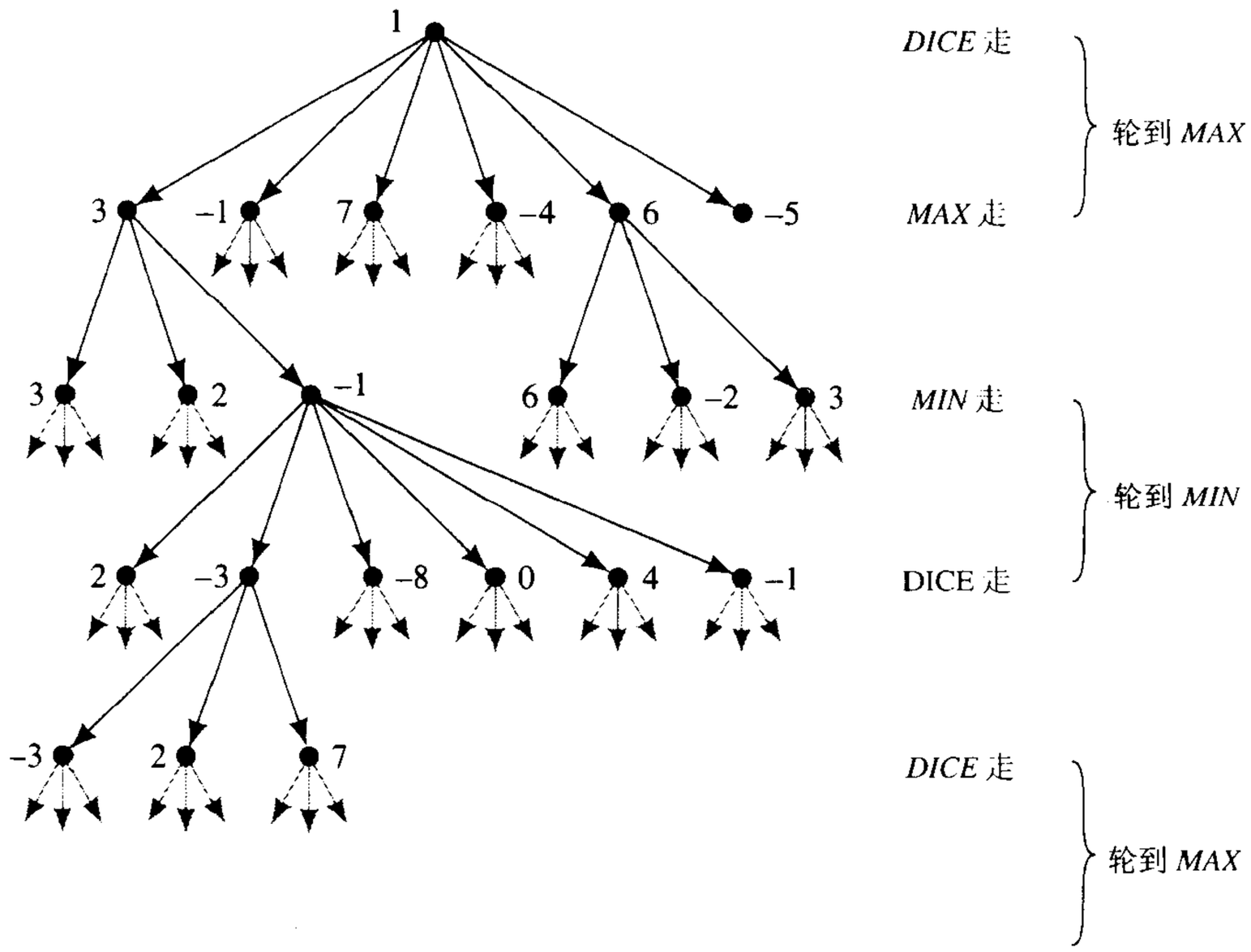


图12-8　一个掷骰子的博弈树

在涉及到概率的博弈树中，也可以倒推值，除了有时在有概率移动的节点，我们取后继节点值的平均值而非最大值或最小值[⊖]。图12-8中节点边上的数值为倒推值。我们倒推以下值：代表*MIN*移动的节点的后继节点值的最小值；代表*MAX*移动的节点的后继节点值的最大值；代表骰子移动的节点的后继节点的期望值。这种修改过的倒推程序有时称为期望的最大值(*expectimaxing*)。

⊖　当然，如果我们对概率移动的结果非常保守，无论什么时候只要骰子移动，我们就倒推*MAX*的最坏值。

引入概率移动会使博弈树搜索产生过多的分枝，此时，一个好的静态评估函数对限制搜索的深度是相当重要的。假如可找到一个足够好的静态评估函数，*MAX*在一些位置选取移动时只需向前预测一步，然后选择可使这些位置的静态值最大的移动方式。下面我们将讨论学习动作策略的技术，以及在有很大的分枝因子博弈中学习静态评估函数以允许搜索的技术。

12.7 学习评估函数

有些博弈（如围棋）的每一步都存在着多种移动，以至于不能进行深层搜索[⊖]。在不进行搜索的情况下，可通过后继节点的静态值或各种模式识别技术对当前布局作出反应来选择移动。例如对围棋，就曾经试图寻找一些对位置评估并做出反应的方法[Zobrist 1970，Reitman & Wilcox 1979，Kierulf，Chen，& Nievergelt 1990]。

有时，可通过神经网络来了解好的静态评估函数，西洋双陆棋博弈就是这方面的一个好例子。一个叫TD-GAMMON的程序[Tesauro 1992，Tesauro 1995]通过一个具有层次结构且向前反馈的神经网络来操作西洋双陆棋博弈，神经网络的结构和编码如图12-9所示。198种输入（代表西洋双陆棋格局）被全部连接到隐藏单元，隐藏单元全部连接到输出单元。隐藏单元与输出单元都是Sigmoid。输出单元根据输入格局产生的结果的可能值产生4个估计值p_1、p_2、p_3和p_4。布局位置的全局值由一个估算的性能指标给出：$v = p_1+2p_2-p_3-2p_4$。在用网络操作西洋双陆棋时，投掷骰子，现有布局的任何可能的移动产生新的布局，并由网络来评估骰子的变化。选取含有最好v值的布局，进行产生这种格局的移动（假如是白棋移动，最大的v值最好；假如是黑棋移动，最小的v值最好）。

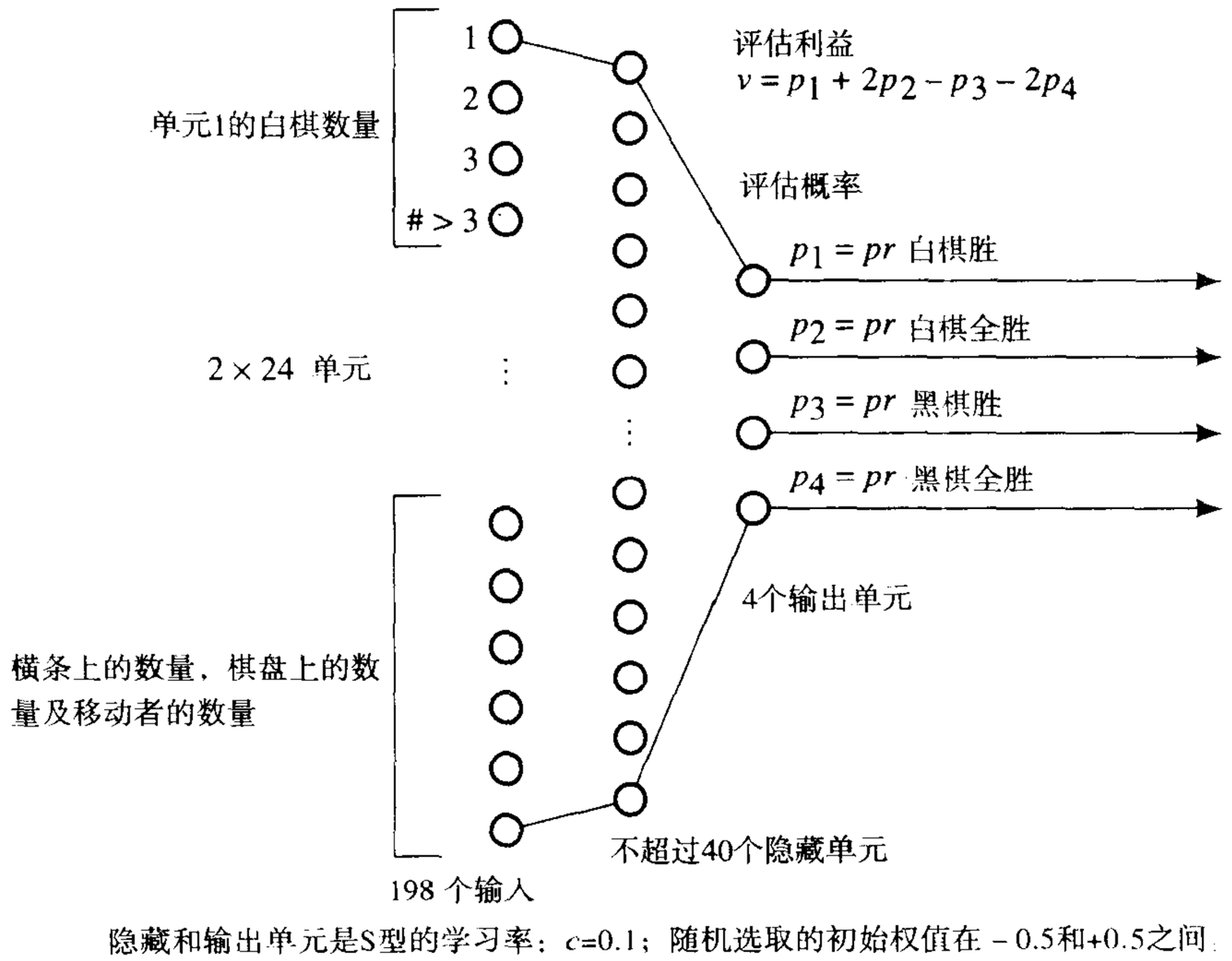


图12-9 TD-GAMMON 网络

在实际使用网络操作博弈时，会有时间差异。每移动一步，通过反向传播调整网络权值，

⊖ 下围棋的人显然可以对布局的孤立子集进行搜索，一些有能力的程序下围棋也是可能的。

使预期指标从初始位置更接近结果位置。为简单起见，解释这种方法时，假定它只有时间差（实际上用到一个变量，但这里该变量与我们无关），假如v_t是t时刻网络博奕结果的估计值（一步移动之前），v_{t+1}是时间$t+1$时的估计值（一步移动之后），标准的时间差的权修正值规则为：

$$\Delta \mathbf{W}_t = c(v_{t+1} - v_t)\frac{\partial v_t}{\partial \mathbf{W}}$$

$\mathbf{W}_t$是t时刻网络中所有权值的向量，$\frac{\partial v_t}{\partial \mathbf{W}}$是在权空间中对$v_t$的梯度（对一个具有多层结构并向前的网络，如TDGAMMON中的网络，每一层权向量中分量的变化都可用第3章中的方式描述）。训练网络以便对所有的t，一步移动前的输入产生的输出v_t总是趋向于一步移动之后输入产生的输出值v_{t+1}（就像在数值迭代中）。在用这种训练方法的测试实验中，使用网络对成千上万种博弈进行了操作（大约包括了150万种博弈）。

经过完善训练，网络的性能几乎是一流的。在用向前搜索程序操作了40个博弈后，Bill Robertie（前西洋双陆棋世界冠军）估计TD-GAMMON 2.1的水平处于顶级高手——极其接近（相差只有几个百分点）世界上最好的人类选手[Tesauro 1995]。

12.8　补充读物和讨论

如同谜宫一样，博弈对完善和测试人工智能技术也很重要。例如，[Russell & Wefald 1991，第4章]提出并评估了博弈树搜索算法（MGSS*和MGSS2），它使用了元级计算（涉及继续搜索的期望值）来减小博弈树，比α-β过程更有效。Berliner's B*算法使用间隔范围[Berliner 1979]，可进行更有效的剪枝。[korf 1991]把α-β过程扩展到多人博弈。

有些书中没有使用数值评估函数，而利用模式识别技巧来判断位置的好坏，这种技术已被用于国际象棋残局博弈的程序中[Huberman 1968，Bratko & Michie 1980]。

博弈操作方面早期成功的著作来自于Arthur Samuel，他开发了西洋跳棋博弈的机器学习方法[Samuel 1959，Samuel 1967]。Samuel的西洋跳棋操作程序达到冠军级水平，Alberta大学的Jonathan Schaeffer的CHINOOK程序[Schaeffer，et al. 1992，Schaeffer 1997]现在被认为是世界西洋跳棋冠军。1997年，IBM程序“深蓝”在一场冠军赛中击败了国际象棋冠军Garry Kasparov（盖利·卡斯帕洛夫）。

[Newborn 1996]是一本关于计算机下棋的书，它将历史追溯到Garry kasparov战败“深蓝”的1996年。要了解这本书及关于象棋博弈作为人工智能研究平台方面的评论，见[McCarthy 1997]。McCarthy认为如果象棋程序采用类似人的推理方法，用较少的搜索就能有更好的性能。他认为假如研究者们用它测试基于知识的推理方法，象棋又会成为人工智能的“果蝇”（*Drosophila.*）

[Michie 1966]提出了期望最大(*expectimax*)一词，并进行了实验。

[Lee & Mahajan 1988] 对奥赛罗博弈（Othello）采用加强学习方法，[Schraudolph，Dayan & Sejnowski 1994]将时间差方法运用于围棋博弈。

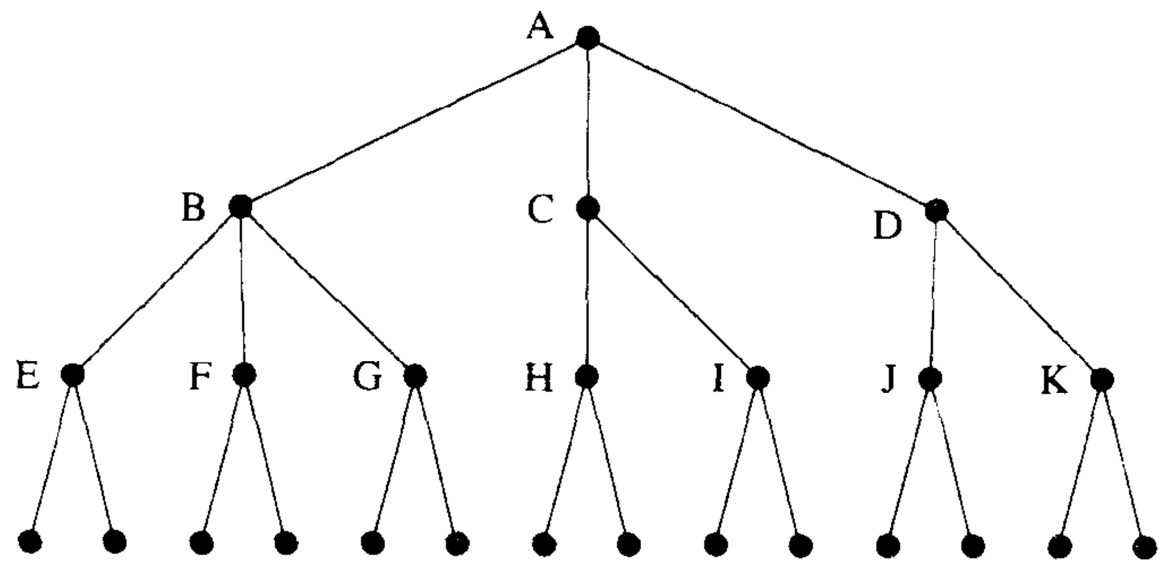
要更多地了解博弈搜索方法，见[Pearl 1984，第9章]

习题

12.1 为什么在博弈的程序中搜索总是从当前位置向前而不是从目标向回搜索？

12.2 考虑下面的博弈树，静态值（在叶节点的圆括号中）都是从第一个博弈者的角度得出

的，假设第一个博弈者为*MAX*一方。



L	M	N	O	P	Q	R	S	T	U	V	W	X	Y
(2)	(3)	(8)	(5)	(7)	(6)	(0)	(1)	(5)	(2)	(8)	(4)	(10)	(2)

1) 第一个博弈者将选择什么移动?

2) 假如使用α-β算法，哪些节点无须检验？——假设节点按从左到右顺序检验。

12.3 假设用α-β截断来决定博弈树中的移动值，而且已断定节点n及其子女可被截断。假如博弈树实际上是一个图，并且有另一条路可到达节点n，你是否仍认为应删除该节点？如果应该删除，证明它，否则举出一个反例。

12.4 许多博弈的程序在跳向下一步时不存储搜索结果，而是每次轮到机器移动时，它都完全重新开始，为什么?

12.5 谈谈（限于一页半以内）你将如何改进最小最大化策略，使之可用于在三人的完全信息博弈中为其中一方选择移动（一个完全信息博弈指任何一个博弈者都完全了解博弈状况，象棋即此类博弈，而桥牌不是）。假设有三个博弈者*A*、*B*和*C*，轮流开始。假设博弈可以平局结束或以任何一方胜利结束。还可假设博弈树不可搜索至终点，但为了选择移动，非终点位置的评估需用某个类似最小最大化策略的过程来倒推。

第三部分　知识的表示和推理

第13章　命题演算

13.1　对特征值加以约束

前面已经论述了两种截然不同的、为一个agent世界建模的方法：基于图标的和基于特征的方法。二元值特征是对这个世界的描述——什么是真的，什么是假的。而图标表示则是对这个世界的某些方面的模拟。虽然模拟比描述更直接，因而也常常更有效，但是描述有它自身的一些优势。特征值容易与别的agent进行通信，而图标模型就很难分解为独立的部分来满足增量的通信。特征值的计算与图标模型的构建和修改相比，常常只需要较少精细的知觉处理。并且，当不能直接感知某些特征值时，可以利用加在某个agent所处的世界的约束，从其他的特征值推断出。

进一步说，一个agent环境的某些信息是很难或者说是不可能由图标来表述的。例如：

- 普遍规律，如“所有的蓝色盒子都是可推动的”。
- 否定信息，如“积木A不在地板上（没有说积木A在哪儿）”。
- 不确定信息，如“或者积木A在积木B上面，或者积木A在积木C上面”。

有些这种难以表述的信息可以用公式表示为对特征值的约束，这些表示某个agent所处世界的重要知识的约束常常被用来推断那些不能被直接感知的特征值。推断有关一个agent当前（*present*）状态的信息（使用基于特征中约束的计算），可以与从一个agent行为的将来（*future*）状态的计算（使用本书第二部分讨论过的搜索方法）作对照。第一个方法称为“推理（*reasoning*）”，第二个称为“投影（*projecting*）”。在以下几章中，先不考虑投影的问题，而集中讨论推理。

包含有关特征值的推理有几项应用。可以确信，当agent（甚至是反应型agent）决定行动的时候，推理能增强它们的效力。但是，也存在许多别的应用。例如：已经可以验证用合适的特征集来表述各种物理系统的功能，包括生物的和电子机械的等。这些特征中的约束把物理的或别的与有机体或器械相关的规律进行编码，然后可以在其他的目标中进行推理，用来诊断这些系统中的故障。例如，与“原因”相联系的特征可以从与“症状”相联系的特征推断出。这种方法是人工智能应用中的重要一类——专家系统（*expert system*）的基础。

用一个富有启发性的例子来开始推理技术的讨论。设想一个能举起一块积木的机器人，假如那个木块是可举的（即不是太重的），并且假设这个机器人有足够的电池能源。假如以上条件都满足，那么当机器人试着举起这个它所握住的木块时，它的手臂就移动。可以用二元值特征来表述这些不同的条件：

x_1 (*BAT_OK*)（电池能源合适）

x_2 (*LIFTABLE*)（可举的）

x_3 (*MOVES*)（移动）

用这些便于记忆的特征名称（放在括号中）可有助于讨论。假设机器人能感知*BAT_OK*（通过读量表）和*MOVES*（通过联角传感器），但不能感知*LIFTABLE*。但知道*LIFTABLE*的值对机器人完成其必须完成的任务来说是很重要的。从上面的描述，我们知道假如*BAT_OK*和*LIFTABLE*的值都为1，那么*MOVES*也如此。所以，假如当这个机器人试着要移动这块积木时*MOVES*值为0，那么我们知道或者*BAT_OK*或者*LILFTABLE*（或者两者）肯定值为0。如果*BAT_OK*被感知到值为1，那么*LIFTABLE*的值一定为0。既然我们能如此推理，那么机器人也可如此。所需要的是一种能用于表达特征中的约束和特征值的语言(*language*)和能进行必要推理的推理（*inference*）机制。而命题演算（*propositional calculus*）作为布尔代数的一种延伸，为此提供了必要的工具。

上面例子中的约束可用命题演算语言如下表示：

BAT_OK ∧ LIFTABLE ⊃ MOVES

其中，∧的意思是“合取”，而⊃的意思是“蕴含”。

与这种语言相联系的机制可以用来从这种语句（*statement*）中推出结论（*consequence*）。既然逻辑语言在人工智能中是如此重要，那么，必须在表述基于使用这些语言的更复杂和更具智能的agent之前，对这些语言作更详细的说明。

首先是一些定义。逻辑包含

- 一种语言（具有一个句法用以规定在这种语言中什么是合法的表达式）。
- 推理规则　用以操作这种语言中的语句。
- 语义　用以把这种语言中的要素和某些主题（*subject matter*）中的要素联系起来。

我们要学习两种逻辑语言：第1种是两种语言中较为简单的，叫做命题演算；第2种，也是更有用的，叫做一阶谓词演算（*first-order predicate calculus*，　*FOPC*）。由于许多在一阶谓词演算中重要的概念可以更为简单地在命题演算中作介绍，因此首先介绍命题演算。

13.2　语言

下面从形式上描述命题演算中的组成元素。此刻最好不要把某种意义与这种语言的组成元素联系起来，把我们现在正在做的想像为对一个无意义的游戏规则的描述，以后我们再谈论意义。以下是这种语言的组成元素：

原子（*atom*）：两个最明显的原子是T和F，还有可数的以大写字母开头的那些字符串的无限集合，如：P, Q, R, ..., P_1, P_2, ON_A_B, 等等。

联结词（*connective*）：∨、∧、⊃和¬，分别称为：“析取（*disjunction*）”、“合取（*conjunction*）”、“蕴含（*implication*）”和“非（*negation*）”（以后我们会给出与这些名称相关的联结词的意义，而现在它们只是无意义的符号）。

合式公式（*well-formed formulas*，*wff*）（也称为句子*sentence*）的语法：

- 任何原子都是一个合式公式。

例如：P, R, p3

- 假如ω_1和ω_2是合式公式，那么以下这些也是：

$\omega_1 \vee \omega_2$（ω_1和ω_2的析取）

$\omega_1 \wedge \omega_2$（ω_1和ω_2的合取）

$\omega_1 \supset \omega_2$（蕴含）

$\neg\omega_1$（ω_1的非）

原子以及前面带有 ¬ 符号的原子叫做文字（*literal*）。在$\omega_1 \supset \omega_2$中，$\omega_1$被称为蕴含的前项（*antecedent*），而$\omega_2$被称为蕴含的后项(*consequent*)。

合式公式的例子：

$(P \wedge Q) \supset \neg P$

$P \supset \neg P$

$P \vee P \supset P$

$(P \supset Q) \supset (\neg Q \supset \neg P)$

$\neg\neg P$

- 不再有别的合式公式。

例如：$P \supset \neg\neg$ 不是一个合式公式。

注意以上例子中有语言外的分隔符“(”和“)”的使用。它们根据递归定义把合式公式组成次级（sub）合式公式。有些逻辑处理把这两个分隔符公式化为语言的组成部分，然而在此用得有点宽松并且较直观。合式公式可以通过递归地使用它们的定义来识别。例如，$(P \wedge Q) \supset \neg R$是一个合式公式。首先，P和Q是合式公式，所以（P ∧ Q）是合式公式；并且由于R是一个合式公式，¬R也是一个合式公式，因而，$(P \wedge Q) \supset \neg R$是一个合式公式。

13.3 推理规则

从一些合式公式推出另一些合式公式可以有许多方法，称它们为推理规则（*rule of inference*）。一条推理规则的典型形式是：γ可以从α（或从α和β）推出。下面是一些通用的推理规则：

- 合式公式ω_2可以根据合式公式ω_1和$\omega_1 \supset \omega_2$推出（称之为假言推理或演绎推理（*modus ponens*））。
- 合式公式$\omega_1 \wedge \omega_2$可以根据两个合式公式ω_1和ω_2推出（∧引入）
- 合式公式$\omega_2 \wedge \omega_1$可以根据合式公式$\omega_1 \wedge \omega_2$推出（∧交换）
- 合式公式ω_1可以根据合式公式$\omega_1 \wedge \omega_2$推出（∧消除）
- 合式公式$\omega_1 \vee \omega_2$可以根据单个合式公式ω_1或单个合式公式ω_2推出（∨引入）
- 合式公式ω_1可以根据合式公式$\neg(\neg\omega_1)$推出（¬消除）

13.4 验证定义

合式公式序列 $\{\omega_1, \omega_2, \cdots, \omega_n\}$ 被称为是从一个合式公式集合Δ得出的验证（*proof*）（或演绎，*deduction*），当且仅当在序列中的每个ω_i或者是在Δ中，或者可以从处于这个序列中的较前的一个（或多个）合式公式运用若干推理规则中的一条推出。假如有一个从Δ推出ω_n的验证，就说ω_n是集合Δ的一个定理（*theorem*）。用下面的标记来表示ω_n可以从Δ得到验证：

$$\Delta \vdash \omega_n$$

验证和定理的概念是与一个所使用的特定推理规则集合相关的。如果我们用字母 $\mathfrak{R}$ 来表示推理规则集合，那么我们可以用如下符号写出这样的事实：ω_n可以用 $\mathfrak{R}$ 中的推理规则从Δ中得到验证：

$$\Delta \vdash_{\mathcal{R}} \omega_n$$

例如：给定一个合式公式的集合Δ：{P，R，P ⊃ Q}，下面的序列是一个Q ∧ R 验证，它表达了上面所述的推理规则：

{P，P ⊃ Q，Q，R，Q ∧ R}

验证的概念可以基于一个偏序，也可以基于一个序列。这种偏序可以由一个树结构来表示。验证树中的每个节点都标上一个合式公式，并且必须或者与Δ中的一个合式公式对应，或者用若干推理规则中的一条从树的父节点推出。被标记的树是一个根节点标记的验证。图13-1 是一个验证树的例子。

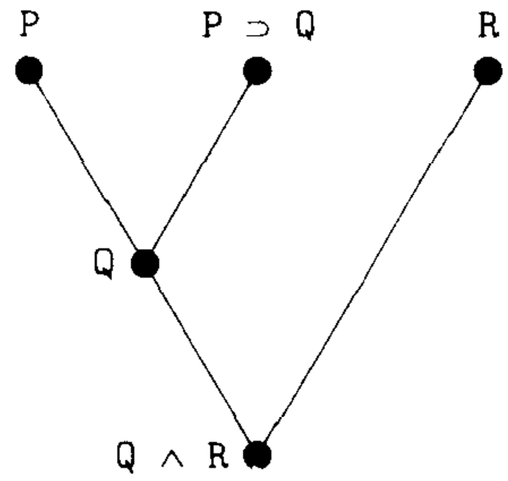


图13-1　验证树举例

13.5　语义

13.5.1　解释

现在我们来讨论一下“意义”。语义（*semantics*）必须把逻辑语言的要素与论域（*domain of discourse*）的要素联系起来。这种联系就是我们用“意义”（*meaning*）这一词所指的。就命题逻辑来说，我们把原子与关于这个世界的命题（*proposition*）联系起来（因而有命题逻辑的称谓）。所以，举例说，我们可以把原子*BAT_OK*与命题”电池是充了电的”联系起来（我们并没有被迫要用助记原子串来作出这种联系，也可以用另外的代替）。原子与命题的联系被称为解释（*interpretation*）。在给定的解释中，与一个原子相联系的命题被称为这个原子的指称（*denotation*）。

给定一个解释，原子为真或假值。假如原子α与命题P相联系，那么对这个世界来说，只有P为真时我们才说α为真；反之它就为假。特殊原子T总是为真，而特殊原子F总是为假。既然关于这个世界的命题必须是真或者是假（在此我们愿意有这么一种理想化），我们可以通过用一种语言直接把值指派给原子来规定一个解释，而不管每个原子所指称的关于这个世界的命题。

假如一个agent有传感装置，这个装置可以用来决定各种有关这个世界的命题的真假，那么当感觉特征*x*1值为1时，与此相应的有关这个世界的命题也为真，并且与此相联系的命题逻辑原子，也许是X1，也为真。因此，我们不用通过在某种输入“线”上的一个值1或0来为agent表述感觉到的信息，而是能用一个在agent的记忆结构（我们称它为知识库，*knowledge base*）中的命题演算原子来表述它。一个原子X1在agent的知识库中出现就意味着这个agent把与此相联系的命题在它的世界中当做真。我们很快会看到一个agent怎样在它的知识库中使用合式公式。

13.5.2　命题真值表

在某种解释下给定原子的值，我们可以用一个真值表（*truth table*）来计算在同样解释下的任何合式公式的值。这个真值表建立了命题联结词的语义（意义）。设ω_1和ω_2是合式公式，那么真值表规则就是：

- 假如ω_1为真并且ω_2也为真，那么（$\omega_1 \wedge \omega_2$）就为真；否则，它就为假。
- 假如ω_1为真或者ω_2为真，或两者都为真，那么（$\omega_1 \vee \omega_2$）就为真；否则，它就为假。
- 假如ω_1为假，那么$\neg\omega_1$就为真；否则，它就为假。

- ⊃的语义是以∨和¬来定义的。具体地说，（$\omega_1 \supset \omega_2$）是（$\neg \omega_1 \vee \omega_2$）的替换形式和等价形式。

这些之所以被称为真值表规则是因为它们常以列表的形式来表述，如在表13-1中那样：

表13-1 命题的真值表

ω_1	ω_2	$\omega_1 \wedge \omega_2$	$\omega_1 \vee \omega_2$	$\neg \omega_1$	$\omega_1 \supset \omega_2$
真	真	真	真	假	真
真	假	假	真	假	假
假	真	假	真	真	真
假	假	假	假	真	真

给定了组成原子的值，我们就可以用真值表来计算合式公式的值。举一个使用真值表来计算一个合式公式的例子，假设P为假，Q为假，R为真。根据这种解释，[（P⊃Q）⊃R]⊃P的值是什么？按“从内到外”计算，我们首先算出P⊃Q的值为真；然后算出（P⊃Q）⊃R也为真；最后，既然P为假，那么整个表达式的值一定为假。

假如一个agent使用*n*种特征（与命题相对应）来描述它的世界，并且这些特征是用一个相应的*n*个原子的集合表述在这个agent世界模型中，那么它的世界就会有2^n种不同的情形——只要这个agent能辨别，因为*n*个原子都可以有真值或假值，故存在着2^n种不同的情形。这个世界所能有的每一种情形都对应于一个解释。给定*n*个原子的值（解释），那么这个agent就可以用真值表找到任何合式公式的值。相反的过程又是怎样的呢？假设在一个合式公式的集合中已给定了这些合式公式的值，那么这些值能导出一个惟一的解释吗？这个问题很重要，因为agent常常被给定一个在诸多特征（表达式为真的合式公式）中的约束集合，那么这个agent能否用它的语言导出一个值的指派（一个解释）给原子，并且因此决定有关这个世界的命题是否为真或为假呢？也就是说，这个公式能说明这是这个世界的2^n种情形之一吗？这个答案通常是否定的。相反，在一个合式公式集合中也许会有许多给其中每个合式公式以真值的解释。为进一步探索这个主题，现在介绍模型的概念。

13.5.3 可满足性与模型

一个合式公式在一种解释下被指派为真值，那么这种解释满足这个合式公式。一种满足一个合式公式的解释被称为这个合式公式的一个模型。一种解释满足在一个合式公式集合中的每一个合式公式被称为这个合式公式集合的模型。因为一种解释给每一个原子指派一个值，所以我们可以判定一种解释是否满足任何一个原子。我们可以用真值表来判定一种解释是否满足一个包含原子的合式公式。

在举积木的机器人的例子里，我用下面的合式公式表达了一个在某些特征中的约束：

BAT_OK ∧ LIFTABLE⊃MOVES

假如这个合式公式如我们所期望的那样值为真，那么我们必须排除所有这样的解释。在这些解释中BAT_OK 和LIFTABLE为真，而MOVES为假。每个“约束合式公式”都告诉我们一些有关这个世界可能的情形，并且排除一些可能的模型（每个模型都与这个世界的一种可能情形相对应）。一般说来，描述这个世界的合式公式越多，模型就越少！没有必要为这个事实感到吃惊。我们知道有关这个世界的东西越多，那么有关这个世界可能情形的不确定性就越小。

可能没有任何解释可以满足一个合式公式（或一个合式公式的集合），在这种情况下，这个合式公式（或合式公式的集合）被说成是不一致的（*inconsistent*）或不可满足的（*unsatisfiable*）。不可满足的合式公式的例子如F和P ∧ ¬P（没有一种解释可以使这些合式公式为真）。一个不可满足的合式公式集合的例子是{P ∨ Q，P ∨ ¬Q，¬P ∨ Q，¬P ∨ ¬Q}（用真值表可证实没有一种解释可以使所有这些合式公式为真）。

13.5.4　永真性

假如一个合式公式在它的组成原子的所有解释下都为真，那么它是永真的（因而，一个永真的合式公式是空的，它没有告诉我们任何有关这个世界可以是怎样的情形）。下面是一些永真的合式公式的例子：

- P ⊃ P（这与 ¬P ∨ P 相同）
- T
- ¬(P ∧ ¬P)
- Q ∨ T
- [(P ⊃ Q) ⊃ P] ⊃ P
- P ⊃ (Q ⊃ P)

用真值表来确定一个合式公式的永真性要化费大量的时间，因为这个合式公式必须要针对所有原子值的组合来计算。

13.5.5　等价

当且仅当两个合式公式的真假值在所有的解释中都是相同的，那么我们可以说它们是等价的。用符号“≡”表示等价。可以用真值表来验证下面的等价：

- 德·摩根定律：

  $\neg(\omega_1 \vee \omega_2) \equiv \neg\omega_1 \wedge \neg\omega_2$

  $\neg(\omega_1 \wedge \omega_2) \equiv \neg\omega_1 \vee \neg\omega_2$

- 换质换位定律：

  $(\omega_1 \supset \omega_2) \equiv (\neg\omega_2 \supset \neg\omega_1)$

- 假如ω_1和ω_2是等价的，那么下面的公式是永真的：

  $(\omega_1 \supset \omega_2) \wedge (\omega_2 \supset \omega_1)$

由于这个事实，$\omega_1 \equiv \omega_2$这个标记常常被用作为$(\omega_1 \supset \omega_2) \wedge (\omega_2 \supset \omega_1)$的缩写。

13.5.6　涵蕴

如果在集合Δ中，每一个合式公式都为真的所有解释下合式公式ω值为真，那么我们说Δ逻辑涵蕴（*logically entail*）ω，并且ω从Δ中逻辑地派生（*logically follow*），ω 是Δ的一个逻辑推论（*logical consequence*）。用符号⊨来指称逻辑涵蕴，并且写为Δ⊨ω。下面是一些例子。

- {P}⊨ P
- {P，P ⊃ Q}⊨ Q
- F⊨ ω（ω是任何合式公式！）
- P ∧ Q ⊨ P

在最后的两个例子中，把标记稍微作了简缩。当Δ为单个时，常常这么做。

逻辑涵蕴在人工智能中是很重要的，因为它提供了一种强有力的方法来说明，如果有关一个世界的命题是真的，那么另一些所关注的命题（也许是不能被感觉到的）也必须是真的。例如，假设我们感觉一些特征，把它们与公式BAT_OK和¬ MOVES相联系，并且用公式BAT_OK ∧ LIFTABLE ⊃ MOVES来表述有关这个世界的知识。就是说，我们有三个公式，其中两个描述一个特定的世界的情景，其中另一个描述有关这个世界的一般知识。读者可根据真值表知道¬LIFTABLE由这三个公式逻辑涵蕴。既然根据这种涵蕴，¬LIFTABLE在所有这三个公式都为真的解释中为真，那么它在我们预期的解释（*intended interpretation*）（即我们把它与机器人世界相联系的）中一定为真。所以，这个作为我们预期的解释的一部分命题，即“积木是不可举的”必须是真的！

因为涵蕴对于决定有关这个世界的命题是真还是假是一个强劲的工具，所以对我们来说，研究怎样把信息表述为合式公式，并且怎样有效地产生出涵蕴的合式公式是至关重要的。可以总用真值表的来做这件事，但是我们要寻求更简便的方法。一个富有吸引力的可以替代涵蕴的是推理（*inference*）。虽然它们是根本不同的概念，但是它们可以由合理性（*soundness*）和完备性（*completeness*）的概念联结起来。

13.6 合理性和完备性

把推理与涵蕴联系起来有两个重要的定义（现在给出定理和验证的直觉含义）：

1) 假如对任意合式公式的集合Δ和合式公式ω，$\Delta \vdash_{\Re} \omega$ 蕴含着 $\Delta \models \omega$，那么我们说推理规则集合ℜ是合理的（*sound*）。

2) 假如对任意合式公式的集合Δ和合式公式ω，当有$\Delta \models \omega$时，存在用推理规则集合ℜ从Δ推出ω的验证，那么就说ℜ是完备的（*complete*）。

对命题演算还没有给出一个完备的推理规则，下一章会讨论这个问题。

当推理规则是合理和完备的时候，可以通过寻找一个验证而不是用真值表来确定一个合式公式是否由一个合式公式的集合导出。当推理规则合理时，假如找到了一个ω 从 Δ 导出的验证，那么我们知道ω是从Δ逻辑地导出的。当推理规则完备时，我们知道最终能够通过一个完备的搜索过程搜寻一个验证来确定ω从Δ导出（当它如此做的时候）。不管是在命题演算还是在谓词演算（将在后面研究）中，用验证的方法来替换真值表法通常节省了巨大的计算量。

要确定一个合式公式是否由一个合式公式的集合逻辑地导出或能否由此得到验证，这是一个NP难题[Cook 1971]（即其复杂性不会比原子数的指数少）。尽管如此，还是有些易处理的特殊情况，所以，了解逻辑推理的过程是重要的。

13.7 命题可满足性问题

在大多数情况下，我们都想尝试确定一个集合Δ的所有模型都是某些合式公式ω的模型。不过有时也会试图至少找出一个Δ的模型，就是说，我们也许要说明集合中的合式公式根据同样的解释每个都是可满足的。换句话说，我们要为一个包含了Δ中所有合式公式的合取公式找到一个模型。

为一个公式找一个模型的问题是一个命题可满足性问题（*propositional satisfiability*, *PSAT*）。在下一章中要说明任何公式可以被写为一些文字析取的合取。一些文字的析取被称为一个子句，

一个写成子句合取的公式叫做合取范式（*conjunctive normal form*，*CNF*）。所以它足以解决CNF公式的命题可满足性问题。许多有趣的问题，包括那些涉及约束满足、电路综合、电路诊断和规划的问题，可以编码为CNF PSAT问题（[Selman, Kautz, & Cohen 1994]）来解决。

穷尽解决CNF PSAT问题的方法就是试着系统地给公式中的原子指派真和假，然后检查每一个指派，看所有这些子句是否在这种指派下为真。假如公式中有n个原子，那么就有2^n个不同的指派，所以对大的n来说，这种指派过程从计算上来说是不可行的。

命题可满足性（PSAT）问题的一些特例如2SAT和3SAT问题。kSAT问题就是要为一个子句的合取找到一个模型，最长的合取式恰好包含k个文字。2SAT问题具有多项式复杂性，3SAT则是NP完备的。因而，一般的PSAT问题是NP完备的。但是即使是所有为解决问题的已知算法，象PSAT，在最坏的情况下，也要花费指数量级的时间，许多这样的问题却只要花费多项式的期待（*expected*）时间。事实上，就许多自然概率分布而言，PSAT问题只需要多项式的平均时间[Goldberg 1979，Purdom 和Brown 1987]。

GSAT是一个非穷尽的、贪婪的、爬山型搜索过程 [Selman， Levesque 和 Mitchell 1992，Selman 和 Kautz 1993]。这个过程通过为公式中的所有原子选择一个随机的真假值的集合开始。这个真假值的集合就是一种解释。在这种解释下值为真的子句的数量可以被标识。接下来，遍历原子的列表，并当原子的值改变时，对其中每一个原子计算值为真的子句数量的增加量。改变那些给出最大增量的原子的值，并继续。当然最大增量可能会是0或负数，但是GSAT在任何情况下都作出改变。因为这个过程可以无止境地在一个“高地”上漫游，它就应在一定次数之后中止。既然这个过程可以在局部极大值（一种满足某些但不是所有子句的解释）上中止，它就得由另一个随机的解释重新开始去搜索一个更大的极大值。GSAT已经用来为2000个可变的、随机产生的3SAT问题找到模型，并且已被用来解决大型N皇后问题的命题编码。GSAT上的随机行走变化（即WALKSAT）已被用来改进它的效率 [Selman，Kautz, & Cohen 1994，Selman，Kautz, & Cohen 1996]。

13.8　另一些重要的问题

13.8.1　语言差异

命题演算是一种形式语言，它是人工agent用来描述它的世界的。这里总是存在着一种把非形式的数学语言和英语（在这本书中用此来谈论命题演算）与形式的命题演算语言相混淆的可能性。例如，当我们说 {P，P ⊃ Q} ⊢ Q 时，我们用符号 ⊢。这个符号不是一个命题演算语言中的符号，而是一个我们用来谈论命题演算用的语言中的符号。例如，元语言（*metalinguistic*）的符号 ⊢ 和 ⊨ 永远不应该与符号 ⊃ 相混淆。

13.8.2　元定理

除了在命题演算中的定理（由推理规则链导出），我们还有关于命题演算的定理（这些常常被叫做元定理）。这里有两个重要的关于命题演算的定理。

- 演绎定理：如果 $\{\omega_1, \omega_2, \cdots, \omega_n\} \models \omega$，那么 $\{\omega_1 \wedge \omega_2 \wedge \cdots \wedge \omega_n\} \supset \omega$ 是永真的，反之亦然。
- 反证法：如果集合Δ有一个模型，但是 $\Delta \cup \{\neg \omega\}$ 没有模型（即，没有什么解释可满足所

有在这个组成的集合中的合取范式），那么 $\Delta \models \omega$。

13.8.3 结合律

二元联结词 $\wedge$ 和 $\vee$ 是可结合的，即

$$(\omega_1 \wedge \omega_2) \wedge \omega_3 \equiv \omega_1 \wedge (\omega_2 \wedge \omega_3)$$

$$(\omega_1 \vee \omega_2) \vee \omega_3 \equiv \omega_1 \vee (\omega_2 \vee \omega_3)$$

因此，在这些合取范式中，我们可以去掉括号并写成

$$\omega_1 \wedge \omega_2 \wedge \omega_3$$

$$\omega_1 \vee \omega_2 \vee \omega_3$$

上面的第一个式子叫做合取式（*conjunction*），第二个式子叫做析取式（*disjunction*）。第一个式子中的每一个 ω_i 叫做合取项（*conjunct*），第二个式子中的每一个ω_i叫做析取项（*disjunct*）。

13.8.4 分配律

$$\omega_1 \wedge (\omega_2 \vee \omega_3) \equiv (\omega_1 \wedge \omega_2) \vee (\omega_1 \wedge \omega_3)$$

$$\omega_1 \vee (\omega_2 \wedge \omega_3) \equiv (\omega_1 \vee \omega_2) \wedge (\omega_1 \vee \omega_3)$$

不要被在本章中表述的某些问题表面的简明性所误导；不能恰当地掌握和记住这些概念将给以后的学习带来困难！

习题

13.1 用真值表的方法说明$\neg(P \wedge Q) \equiv (\neg P \vee \neg Q)$。

13.2 验证如果 Δ 是前后矛盾的（即它没有模型），那么 $\Delta \models \omega$，ω可以是任何合式公式。

13.3 怎样用真值表来验证假言推理是正确的？

13.4 考虑下面7个子句：

$\neg A \vee \neg B \vee \neg C$

$\neg A \vee B$

$\neg A \vee C$

$\neg B \vee C$

$\neg B \vee A$

$\neg C \vee A$

$\neg C \vee B$

为满足一个子句的集合而建立一个模型的GSAT过程有时候会在一个局部的最大值内结束。说明这里有一个对A、B和C的真值指派，这个指派是一个由它所满足的子句数量的局部（不是全局）的最大值。

13.5 说明N皇后问题可以被表述为一个PSAT问题（提示：为$N \times N$方块图的每一个方块（k，l）引入一个原子$q_{k,l}$。假如$q_{k,l}$为真，那么在方块（k，l）上有一个皇后；假如它为假，那么该方块是空的。现在根据原子来陈述这个问题的约束。）

第14章　命题演算中的归结

14.1　一种新的推理规则：归结

14.1.1　作为合式公式的子句

在上一章中提到了几种推理规则，包括假言推理。许多这样的规则可以组合成一个规则，叫做归结（*resolution*）。本书中使用的归结应用于代表合式公式的一个特殊的形式，叫做子句（*clause*），现在来定义它。

首先，回想一下，一个文字或者是一个原子（在这种情况下，它叫做肯定文字（*positive literal*）），或者是一个原子的否定（在这种情况下，它叫做否定文字（*negative literal*））。一个子句是一个文字的集合。这个集合是对这个集合中的所有文字的析取式的一个缩写。因此，一个子句是一种特殊的合式公式。通常把子句写为析取式，但是当它们用集合概念来表达时，有些包含归结的定义就会比较简单。例如，子句{P，Q，¬R}（相当于P ∨ Q ∨ ¬R）是一个合式公式。空子句{}（有时候写为Nil）相当于F（它的值是假）。

14.1.2　子句上的归结

命题演算的归结规则可以陈述如下：从 $\{\lambda\}\cup\Sigma_1$ 和 $\{\neg\lambda\}\cup\Sigma_2$（其中 Σ_1 和 Σ_2 是文字的集合，而 λ 是一个原子），我们可以推出 $\Sigma_1\cup\Sigma_2$，它被称为两个子句的归结式（*resolvent*）。原子 λ 是被归结的原子（*atom resolved upon*），这个过程叫做归结。

下面是一些例子：

- 归结R ∨ P和 ¬P ∨ Q 产生R ∨ Q。这两个被归结的子句可以写成隐含式 ¬R ⊃ P和P ⊃ Q，一条应用于这些隐含的、叫做链式（*chaining*）的推理规则产生¬R ⊃ Q，它等价于归结式R ∨ Q。因此我们知道链是归结的一个特例。
- 归结R和 ¬R ∨ P 产生P。既然第二个子句与R ⊃ P等价，我们知道假言推理也是归结的一个特例。
- 在P上归结 ¬P ∨ Q ∨ W 和P ∨ Q ∨ R ∨ S 产生 Q ∨ R ∨ S ∨ W。注意只有一个Q的个例出现在归结式中——这个毕竟被定义为一个集合。
- 在Q上归结P ∨ W ∨ ¬Q ∨ R 和P ∨ Q ∨ ¬R 产生P ∨ ¬R ∨ R ∨ W。在R上归结它们产生 P ∨ Q ∨ 人Q ∨ W。在这种情况下，既然 ¬R ∨ R和Q ∨ ¬Q有真值，那么这些归结式中的每一个的值也是真的。在这个例子中，我们必须在Q上或者在R上归结，但不能两者同时！也就是说，P ∨ W不是这两个子句的归结式！

用 ¬λ 来归结一个肯定文字λ 产生空子句。因为λ 和 ¬λ 是互补的，从 λ 和 ¬λ 可以推出F。任何包含 λ 和 ¬λ 的合式公式的集合都是不可满足的。另一方面，一个包含一个原子和它的否定的子句（如 λ ∨ ¬λ），不管 λ 的真假值，总是为真值。

14.1.3 归结的合理性

刚才描述的归结规则是一种合理的推理规则。就是说，假如子句$\{\lambda\}\cup\Sigma_1$和$\{\neg\lambda\}\cup\Sigma_2$都有真值，那么它们的归结式$\Sigma_1\cup\Sigma_2$也有真值。验证这一事实的一种方法是“用实例来推论”。我们知道原子λ或者有真值或者为假值。假如（情形1）λ为真值，那么¬λ就有假值，故要使子句$\{\neg\lambda\}\cup\Sigma_2$为真，子句$\Sigma_2$必须为真值。假如（情形2）λ为假值，那么要使子句$\{\lambda\}\cup\Sigma_1$为真，子句$\Sigma_1$必须有真值。结合这两种情形，可以看到或者$\Sigma_1$或者$\Sigma_2$必须有真值；因此$\Sigma_1\cup\Sigma_2$就有真值。一种相似的论证可以用真值表来进行。

14.2 转换任意的合式公式为子句的合取式

命题演算中的任何合式公式都可以被转换为一个等价的子句的合取式。一个表示为子句的合取式的合式公式叫做合取范式。（一个表示为文字合取式的析取式的合式公式叫做析取范式）。用一个例子来说明转换一个任意的合式公式为合取范式的过程的每一步，用来说明这个过程的合式公式是¬（P⊃Q）∨（R⊃P）。

1) 用∨符号，用等价的形式来消除蕴含符号：

¬(¬P ∨ Q) ∨ (¬R ∨ P)

2) 用德·摩根定律和用消除双¬符号的方法来缩小¬符号的辖域：

(P ∧ ¬Q) ∨ (¬R ∨ P)

3) 首先，用结合律和分配律把它转换为CNF。

(P ∨ ¬R ∨ P) ∧ (¬Q ∨ ¬R ∨ P)

然后

(P ∨ ¬R) ∧ (¬Q ∨ ¬R ∨ P)

一个子句的合取式（即一个合式公式的CNF形式）常常（用蕴含子句的合取式）表示为一个子句的集合；因而是

{(P ∨ ¬R), (¬Q ∨ ¬R ∨ P)}

在第3步中把合式公式（或合式公式的部分）从DNF形式转换到CNF形式，下面的过程也许是有用的。首先，把DNF合式公式写成一个矩阵，它的行元素是每个合取项中的文字；我们有蕴含行的析取式。例如，DNF形式（P∧Q¬R）∨（S∧R ∧¬P）∨（Q∧S∧P）可以表示为如下的矩阵：

P Q ¬R

S R ¬P

Q S P

现在，在每一行中选一个文字并生成这些文字的析取式。在例子中，这样的一个选择可以是P∨R∨P。作出所有可能的这种选择，每一个选择对应于一个子句，并且把所有这些子句的合取式当作原始合式公式的CNF形式。我们可以把其中某些子句简化，例如，P∨R∨P可简化为P∨R，还可以把某些子句消除，例如，P∨¬P∨Q可以被消除，因为它是永真的。也可以消除被包容（*subsumed*）在另外一个子句中的一些子句（一个子句γ_1包容在子句γ_2中，如果γ_1中的文字是γ_2中文字的子集）。例如，P∨R包容在P∨R∨Q和P∨R∨S中。

14.3　归结反驳

归结是一种合理的推理规则，也就是说，$\Delta \vdash_{res} \gamma$ 蕴含 $\Delta \models \gamma$，其中γ是一个子句。但是归结并不完备。例如，P ∧ R ⊨ P ∨ R，但是不能在子句的集合{P，R}上用归结推出P ∨ R（因为这里没有什么可以被归结的）。所以不能直接用归结来决定所有的逻辑涵蕴。尽管如此，我们还是能用归结来说明P ∨ R的否定是与集合{P，R}不一致的，因而，使用及证法可以验证P ∧ R ⊨ P ∨ R。

为验证这个过程，P ∨ R的否定是 ¬P ∧ ¬R。表示为子句的（合取的）集合，我们所要验证的否定是{¬P，¬R}。为说明这些子句与P ∧ R的不一致性，将P和R加到这个集合中，从而得到{¬P、¬R，P，R}。在后面这个集合的成员上归结产生出空子句，这是一个矛盾式，所以我们间接地从P ∧ R中得到P ∨ R。

一般说来，为从一个合式公式的集合Δ中证得一个任意的合式公式ω，一个归结反驳（*resolution refutation*）有以下过程：

1) 把Δ中的合式公式转换成子句形式——一个（合取的）子句集合。

2) 把待验证的合式公式ω的否定转换成子句形式。

3) 把从第1步和第2步得到的子句转换成一个单一的集合Γ。

4) 反复地对Γ中的子句应用归结，并且把结果加到Γ中，直到再也没有更多的归结项可被加上去，或者产生一个空子句。

不需验证，列出以下的结论[⊖]：

- 归结反驳的完备性：假如 Δ ⊨ ω，那么归结反驳过程将导出空子句。因而，我们说命题归结是反驳完备的（*refutation complete*）。
- 由归结反驳作命题演算的可决定性：假如Δ是一个子句的有限集合，并且，假如Δ ⊭ ω，那么归结反驳过程会在未导出空子句的情况下终止。

在上一章的举积木例子中所用的推理可以用归结反驳来推导。我们有合式公式的集合Δ：

1) BAT_OK

2) ¬MOVES

3) BAT_OK ∧ LIFTABLE ⊃ MOVES

第3个合式公式的子句形式是：

4) ¬BAT_OK ∨ ¬LIFTABLE ∨ MOVES

待证的合式公式的否定产生另一个子句：

5) LIFTABLE

现在我们用归结来导出下列子句：

6) ¬BAT_OK ∨ MOVES（用4归结5得出）

7) ¬BAT_OK（用2归结6得出）

8) Nil（用1归结6得出）

我们也可以表述这个反驳为一个反驳树（*refutation tree*），如图14-1所示。

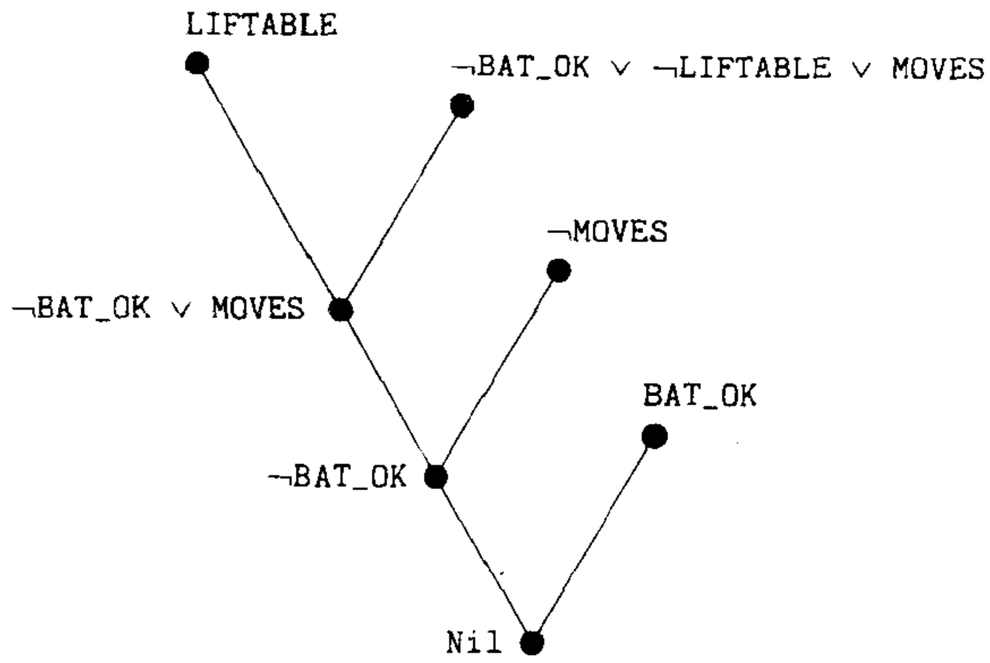


图14-1　归结反驳树

14.4　归结反驳搜索策略

虽然归结反驳过程可以很容易地被描述为“运用归结直到产生空子句为止”，但是这里有

⊖ 有多个关于命题归结反驳完备性的证明，参见[Gensereth & Nillson 1987, P.87](命题完备性也是第16章所述一阶谓词归结反驳完备性的一个特例)。可确定性成立是由于对一个限子句集合，只有有限数目的可能归结。

一个非常重要的问题，就是应该首先运用哪个归结。同样，有些归结根本就无需运用。本节中将讨论这些问题。

14.4.1 排序策略

首先，将按什么序列执行归结？这个问题与在状态空间中下一步将扩展哪个节点的问题类似。前面已介绍了各种排序策略。例如，我们可以定义广度优先和深度优先策略。但首先作一些定义：把原始的子句（包括那些待证合式公式的否定的子句形式）叫做0层归结项（*0-th level resolvent*）。（$i+1$）层的归结项是一个i层归结项和一个j层（$j \leqslant i$）归结项归结的统一归结项。广度优先策略就是生成所有第1层的归结项，然后是所有第2层的归结项，依此类推。

深度优先策略将首先产生一个第1层的归结项，然后用某些第1层或0层的子句来归结这个子句以导出第2层的归结项，依此类推。关于深度的限制，可应用标准的回溯策略。在此书的后面，将继续讨论深度优先归结的应用。

排序归结的一个通用策略是单元优先（*unit-preference*）策略。使用这个策略，我们优先用这样的一种归结，在这种归结中至少有一个子句由一个单一的文字构成。这样的一个子句叫做单元子句（*unit clause*）。注意：图14-1中的例子并不违反单元优先策略（即在反驳树中没有一个归结是处在两个非单元子句之间）。

14.4.2 精确策略

精确策略不涉及被归结子句的排序，它们只允许某些归结发生。其中某些限制（而不是全部！）使归结反驳完备。

1. 支持集

子句γ_2是另一个子句γ_1的后裔（*descendant*），当且仅当：（a）γ_2是γ_1和另一些子句的一个归结项；或者（b）γ_2是γ_1的后裔与另一些子句的一个归结项。假如γ_2是γ_1的一个后裔，那么γ_1是γ_2的一个祖先。支持集（*set of support*）定义为由一些子句组成，这些子句或者来源于待证定理的否定，或者是这些子句的后裔。

支持集策略只允许这样一些归结：在其中正在被归结的子句中的一个在支持集中。注意图14-1中的归结反驳服从一个支持集的限制。

支持集策略是反驳完备的[Chang 和 Lee 1973，P.110]。也就是说，假如我们只对一个不可满足的子句集合运用支持集归结，那么最终会导出空子句。

2. 线性输入

线性输入（*linear-input*）策略只允许这样的归结：其中至少有一个正被归结的子句是原始子句集的一个成员（包括那些待证合式公式的否定）。图14-1中的归结反驳也服从一个支持线性输入策略。

线性输入策略不是反驳完备的，这从下列不一致的子句的集合就可以看出：

{ P ∨ Q，P ∨ ¬Q，¬P ∨ Q，¬P ∨ ¬Q }

这些子句没有模型，所以对它们来说不存在一个归结反驳。这里，把演示对这些子句来说不存在一个线性输入反驳而存在一个归结反驳留作一个练习。

3. 祖先过滤

祖先过滤（*ancestry-filtering*）策略仅允许这样一些归结：在其中至少有一个正被归结的子

句的一个或者是原始子句集的一个，或者是正被归结的别的子句的一个祖先。祖先过滤策略是反驳完备的 [Luckham 1970]。

14.5 Horn 子句

Horn子句是一种特殊类型的子句，它在人工智能和计算机科学的其他领域中都很重要。一个Horn子句就是至多只有一个肯定文字的子句。

下面是一些Horn子句的例子：

P，$\neg P \vee Q$，$\neg P \vee \neg Q \vee R$，$\neg P \vee \neg R$

这种类型的子句首先是由逻辑学家Alfred Horn 研究的 [Horn 1951]。

有三种类型的Horn子句。

- 一个单一原子——常被称为一个“事实”。
- 一个蕴含——常被称为一个“规则”——它的前件由一个肯定文字的合取组成，而它的后件由一个肯定的文字组成。
- 一个否定文字的集合——写成带有一个由肯定文字的合取组成的前件和一个空后件的蕴含形式。例如，当人们否定一个由肯定文字的合取组成的待证的合式公式时，这种形式即可获得。所以这类子句常称为一个“目标”。

这三类Horn 子句的例子分别是P，$P \wedge Q \supset R$，$P \wedge Q \supset$。

一个有关命题的Horn子句的重要结果是，存在着线性时间的演绎算法 [Dowling 和 Gallier 1984]。直观地说，用非Horn子句作NP难题推理的原因是，在试图验证一个肯定文字的析取式如$P \vee Q$ 时，我们必须考虑多种情况——验证P或者Q。在Horn子句中，没有肯定文字的析取式。

为验证源于Horn 子句的规则和事实的目标证明系统，通常以下面的方式提供排序信息给系统：用一定的序列写出事实和规则；用一定的序列写出在每个规则和目标的前件中的文字；然后在基于这些序列的深度优先的样式中搜索一个验证。我们将在讨论了谓词演算之后更详细地研究这个过程。

习题

14.1 验证从DNF转换到CNF的矩阵程序保留了等价性。

14.2 头，我赢；尾，你输。在命题演算中表达这些陈述（加上别的你可能需要的陈述），然后用归结验证我赢（换一种方法，你可以改变这个问题，假如你喜欢：头，你赢；尾，我输）。

14.3 下面的合式公式有时被用在命题演算中作为原子的实例：

1) 蕴含引入：$P \supset (Q \supset P)$

2) 蕴含分配：$(P \supset (Q \supset R)) \supset ((P \supset Q) \supset (P \supset R))$

3) 矛盾体现：$(Q \supset \neg P) \supset ((Q \supset P) \supset \neg Q)$

用归结反驳来验证这些公式中的每一个。

14.4 考虑下列不可满足的子句集合：

$P \vee Q$

$P \vee \neg Q$

$\neg P \vee Q$

$\neg P \vee \neg Q$

1) 为下列的每一种策略导出归结反驳：

(a) 支持集归结（其中支持集是在上述的子句列表中的最后一个子句）。

(b) 祖先过滤形式归结。

(c) 一种既违反支持集也违反祖先过滤的策略。

2) 验证不存在这种不可满足的子句集合的线性输入归结反驳。

14.5 转换下列命题演算合式公式为子句：

$\neg[((P \vee \neg Q) \supset R) \supset (P \wedge R)]$

第15章 谓词演算

15.1 动机

命题演算有一些局限性。例如，我们不能表达这样的事实：当移动木块*B*时，说它就是ON_B_C所断言的木块*C*上的木块。在命题演算中，原子是没有内部结构的串。在关于木块的命题中，ON_A_B和ON_B_C是完全不同的。尽管给这些原子使用了助记名称（帮助我们记住它们代表什么），但也可以使用其他命题字母，如P124和Q23。

一种更有用的语言应该是既能指称在这个世界中的事物（像木块），也能指称有关这个世界的命题。我们需要一种语言，它既有对此进行命题陈述的事物的名称，又有我们要进行陈述的命题的名称。在玩积木的世界（在此之后，称之为“积木世界”）中，也许应该有像ON_B_C ⊃ ¬CLEAR_C这样的命题，其中，CLEAR_C表示木块C上是空的。要为每个木块都表达一个这样的事实将需要几个命题公式。假如我们能够用On(x, y)⊃¬Clear(y) 这样简单的陈述就好了，这里*x*和*y*是能够指称任何木块的变量。

在本章中，将介绍一种叫做一阶谓词演算的语言，它有这些需要的特征。谓词演算具有一些被称为对象常量（*object constant*）、关系常量（*relation constants*）和函数常量（*function constants*）的符号，以及另外我们后面要介绍的一些结构。这些语言实体（当我们讨论语义学时）将被用于指称这个世界中的事物和有关这个世界的命题。

15.2 谓词演算语言和它的句法

首先介绍一种受限的谓词演算，后面再介绍完整的谓词演算。同样，就目前来说，不要去设法对语言的结构加上意思，这样，你将与必须操作这些结构的计算机程序处在同样的地位！

组成：

- 我们有一个对象常量的无限集合，它们是字母数字组成的字符串（常常是助记的，但是只是有助于我们而不是计算机）。本书的对象常量用一个大写字母开始或者用一个数字开始。
  例如：Aa，125，13B，Q，John，EiffelTower
- 一个所有“目（*arity*）”[⊖]函数常量的无限集合。它们是字母数字组成的字符串，总是以小写字母开头，并且总是以它们的目作为上标。
  例如：$\mathtt{fatherOf}^1$，$\mathtt{distanceBetween}^2$，$\mathtt{times}^2$
- 一个所有目的关系常量的无限集合。这些是以大写字母开头的字母数字组成的字符串，并且以它们的目作为上标（有时称一个关系常量为谓词）。例如：$\mathtt{B17}^3$，$\mathtt{Parent}^2$，$\mathtt{Large}^1$，$\mathtt{Clear}^1$，$\mathtt{X11}^4$
- 命题联结词 ∨ 、∧ 、¬ 和 ⊃，还有分隔符（、）、[、] 和分隔符，。

项

⊖ arity 是一个合成词，源于如 binary (arity=2)、tertiary (arity=3)，等等中的后缀。

- 一个对象常量是一个项（*term*）。
- 一个*n*目的函数常量，后面跟着处于括号中、由逗号分隔的项，是一个词项。这类词项被称为函数表达式。在表示这么一个项的时候，当它的值明显可从上下文得知时，通常省略它的目上标。我们可以把对象常量当作目为0 的函数常项。例如：fatherOf（John，Bill），times（4，plus（3，6）），Sam

合式公式

- 原子：一个*n*目的、处于括号中由逗号分隔的*n* 个词项所跟随的关系常量是一个原子（也被称为原子公式，*atomic formula*）。一个 0 目的关系常量省略括号。另外，当目上标的值明显可从上下文得知时，常常把它省略。

  一个原子是一个合式公式。

  例如：Greaterthan（7，2），P（A，B，C，D），Q
- 命题合式公式：任何由谓词演算构成的表达式，构造方式与命题演算从别的合式公式构成一个合式公式的方式一样，它是一个合式公式，称为命题合式公式(*propositional wff*)。

  例如：[Greaterthan(7，2)∧ Lessthan(15，4)] ∨ ¬Brother(John，Sam)∨ P

  （请记住用在这些例子中的合式公式并不一定有含意！）

把命题演算中所作的扩充（具有两个合取项以上的合取式、具有两个析取项以上的析取式、子句、子句合取的集合，等等）作为谓词演算中的合式公式。目前，我们只考虑项与合式公式，后面会介绍其他的（可以允许承诺的变量）。

15.3 语义

15.3.1 世界

现在我们有了一种语言可以用来像指称命题一样指称在这个世界中的对象。我们可以如下表达：

- 这个世界可以有无限多的对象，也叫做个体（*individual*）。这些对象可以是相当具体的，如“木块*A*”、“Whitney先生”、“Julius Caesar”，等等；或者它们也可以是抽象的东西，如“数字7”、“π”、“所有整数的集合”，等等；它们甚至可以是虚构的或者创造的东西（对它们是否实际存在，人们可能有争议），如“美”、“圣诞老人”、“麒麟”、“诚实”，等等。只要愿意给它一个名称，并且对它说些什么，就可以把它当作我们要谈论的这个世界中的一个实际的个体。
- 个体上的函数。我们可以有数目无限的多目函数，能映射*n*元个体到个体。例如：可以由有一个函数映射一个人到他或她的父亲，或者可以有一个函数映射数字10和2到商数5。
- 个体上的关系。个体可以参与任意数目的关系。这些关系中每一个都有目。（具有1目的关系被称为一种属性（*property*））。所以，个体也许会有像“重”、“大”、“蓝”等等这样一些性质，它们也许会参与在如“比……大”、“在……之间”等等这样的关系中。要从外延上指明一个*n*元（*n-ary*）关系，我们就要显式地列出所有参与这种关系的*n*元个体。

15.3.2 解释

对谓词演算中的一个表达式的解释就是一种指派，这种指派把对象常量映射到世界中的对

象，把*n*元函数常项映射到世界中的*n*元函数，把*n*元关系常量映射到世界中的*n*元关系。这些指派被称为它们相应的谓词演算表达式的指称（*denotation*）。受对象常量指派的对象的集合被称为这种解释的域。

给定一个原子的组成部分的一种解释，当原子的词项指称的那些个体的指称关系能成立时，这个原子就为真。假如这种关系不能成立，这个原子就为假。

非原子合式公式的真值与假值可用与命题演算中同样的真值表来决定。

例如：再来考虑积木世界。这是一个存在实体为A、B、C和Floor (地板)[⊖]的世界。我们最终要建立一种指派，把谓词演算的要素映射到这些世界的对象中。因为我们实际上没有办法让世界的对象本身成为指派的值，所以我们想象这个世界是一个数学结构，它（在别的事物中）包括数学对象A、B、C和Floor（尽量不要被“世界是个数学结构”这样的陈述所烦扰。不管这个世界实际是什么，都不妨碍我们的目标，即我们把它考虑为一个数学结构）。

我们也可以想像处于这些对象中的关系。例如，我们可以有关系On（在……上）和Clear（清除）。这些关系可以被参与其中的对象的*n*元来作出外延上的定义。例如，设想我们有一个如图15-1所示这样的积木配置。在这个世界中，关系On由对偶<B，A>、<A，C>和<C，Floor>给出。关系Clear由单项<B>给出。

所以，在这个例子中，个体A、B、C和Floor，关系On和Clear构成了这个积木世界。为了用谓词演算来描述这个世界，使用了对象常量A、B、C和Fl，二元关系常量On，和一元关系常量Clear。为方便起见，在这里用有助于记忆的符号；也许用关系常量G0045、G123、……等等来强调用在谓词演算中的符号表达没有事先指定的意义是比较好的方法。

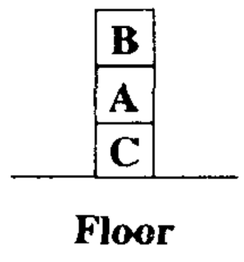


图15-1　一种积木的配置

表15-1　谓词演算与世界之间的映射

谓词演算	世　界
A	A
B	B
C	C
Fl	Floor
On	On = {<B，A>，<A，C>，<C，Floor>}
Clear	Clear = {<B>}

这些谓词演算表达式要作的解释（这恰恰是许多可能解释的一种）在表15-1中已给出。我们可以用这些指派来决定某些谓词演算合式公式的值：

On（A，B）为假，因为<A，B>不在关系On中。

Clear（B）为真，因为<B>在关系Clear中。

On（C，Fl）为真，因为<C，Floor>在关系On中。

On（C，Fl）∧ ¬On（A，B）为真，因为On（C，Fl）和 ¬ On（A，B）都为真。

15.3.3　模型及其相关的概念

有几种语义概念在谓词演算中有与在命题演算中相同的定义。让我们回顾一下：

⊖ 当我想强调语言的元素和这些元素的指称的区别时，有时会用黑体字来表示在环境中的对象、函数及关系，而与谓词演算所用的为等宽字体。

- 如果一个合式公式在某种解释下为真，则这个解释就满足这个合式公式。
- 一个满足一个合式公式的解释就是这个合式公式的模型。
- 任何一个合式公式在所有的解释下都有真值就是永真（*valid*）。
- 任何没有模型的合式公式是不一致的或不可满足的。
- 如果一个合式公式 ω 在所有能使每一个在集合 Δ 中的合式公式都有真值的解释上为真，那么 Δ 逻辑涵蕴（*logically entails*）ω（$\Delta \models \omega$）。
- 当且仅当在所有的解释下两个合式公式都有相同值（即，当且仅当两个合式公式互相逻辑地涵蕴时），它们是等价的。

15.3.4 知识

谓词演算公式可以被用来表达一个agent所具有的有关这个世界的知识。这种公式构成的一个集合 Δ 被称为这个agent的知识库，假如公式 ω 被包括在 Δ 中，我们（一般）说这个agent“知道”ω。（比较准确地说应该是这个agent“相信”ω）。

更严密地检查一下当我们说一个公式的集合包含有关这个世界的知识时可能意味着什么。例如，下面的公式是否包含了有关一个可能的积木世界的知识?

1) On(A, Fl) ⊃ Clear(B)

2) Clear(B) ∧ Clear(C) ⊃ On(A, Fl)

3) Clear(B) ∨ Clear(A)

4) Clear(B)

5) Clear(C)

使用助记法作为提示，可以很容易地构造出适合这些公式、因而也是它们的模型的积木世界的解释。图15-2中给出了我们所能想到的所有情形。在所有这三种情形中，模型使用了与用在表15-1中的把对象常量映射到对象的相同的映射。尽管如此，在这个世界中，关系常量和关系之间的映射在三个模型中是不同的。在第一个中，关系常量On被映射到关系On = {<B，A>，<A，Floor>，<C，Floor>}，关系常量Clear被映射到关系Clear = {<C><B>}。可以很容易地建立起与另外两个积木世界情形相对应的模型。

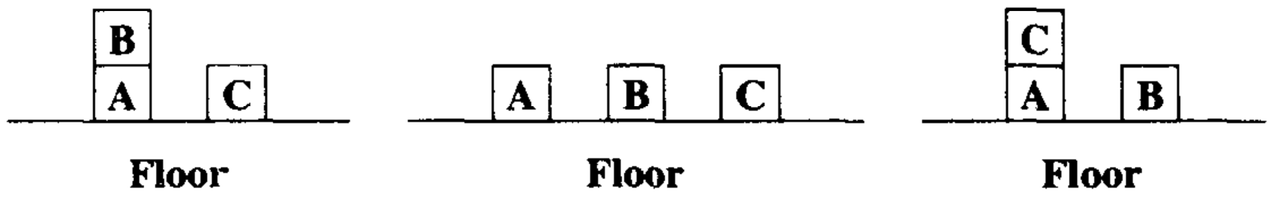


图15-2 三种积木世界情形

除了谓词演算记法提出的模型外，还可以有另外有关这些公式的模型。例如，有这样一个模型，在这个模型中，B和C是在这个世界中的同一个对象比如B的不同的谓词演算的名称 还有一些模型，在其中，A、B和C根本没有被指派给积木，而是指派给了数字⊖。这样一些可选的模型可以通过提供附加的公式来排除。例如，我们可以通过提供附加的公式Clear(A)，再假定对象常量对对象的指派已在表15-1中给出，排除与图15-2中的两种情形相对应的模型。如在命题演算时提到的，我们拥有的公式越多，它们的模型就越少。所以，如果要约束一个公式集合的意义，以便这些公式构成有关一个特定世界的“知识”，我们必须有足够的公式使得

⊖ Löwenheim 定理指出，所有谓词演算合式公式的一致集都有一个模型，它的域是整数[Löwenheim 1915]。

它们的模型不仅包括我们的世界，而且排除那些我们不想让它与我们的世界相混淆的世界。

15.4 量化

假定要使在域中的每一个对象有一定属性（或参与在一定的关系中）。对有限的论域，可以通过一个Clear（B1）∧ Clear（B2）∧ Clear（B3）∧ Clear（B4）这样的合取式来获得。但是长的合取式是麻烦的，而且我们不能写下无限长的合取式，虽然当论域是无限时，它也许是需要的。

或者假定在域中至少有一个对象有一定的属性。对于有限的域，这样的陈述可以通过一个析取式Clear（B1）∨ Clear（B2）∨ Clear（B3）∨ Clear（B4）来作出。然而对于大的或无限的域，问题依然存在。

为达到这样的目的，要引入新的句法实体，即变量符号（*variable symbol*）和量词符号（*quantifier symbol*）。除了先前引入的一些句法实体外，还有：

1) 我们有一个变量符号的无限集合，变量符号由靠近字母表末尾的小写字母开头的串组成，如p、q、r、s、t、...、p1、p2、...，等等（这些由它们在上下文中的使用与函数常项区分开来）。一个变量符号是一个项（除了先前定义过的）。所以，如f（x，Bob，C17）是一个三元函数表达式。

2) 我们有量词符号 ∀ 和 ∃。∀ 称为全称量词（*universal quantifier*），而∃称为存在量词（*existential quantifier*）。

3) 假如 ω 是一个合式公式而 ξ 是一个变量符号，那么（∀ξ）ω 和（∃ξ）ω 都是合式公式。ξ 称为被量化的（*quantified over*）的变量，而 ω 被认为是在量词的辖域内（*within the scope*）。在形如（$Q\xi$）ω 的合式公式中，ξ 是一个变量符号而Q 或者是 ∀ 或者是 ∃。通常，ξ 作为一个项的变量符号嵌入在ω中。假如除了ω 中的 ξ，所有的变量符号都已在 ω 中被量化的，那么（$Q\xi$）ω 被称为封闭的合式公式（*closed wff*）或封闭的语句（*closed sentence*）。在所有的应用中，合式公式是封闭的。例如：

(∀x) [P(x) ⊃ R(x)]，(∃x) [P(x) ⊃ (∃y) [R(x，y) ⊃ S(f(x))]]

（在下面描述了量词语义学之后）可以说明（∀x）[（∀y）ω] 是与（∀y）[（∀x）ω]等价的，所以我们也可以把受全称量词量化的变量组成一个串放在 ω 前：（∀x，y）ω。在这样的一个公式中，ω 被称为矩阵。对存在量词来说也是同样的。但是全称量词与存在量词的混合则必须保持它们的次序！（∀x）[（∃y）ω] 与（∃y）[（∀x）ω]并不等价。

也可以（在描述了下面的语义学之后）说明量词的变量是一种“哑变量”，因而可以不改变合式公式的值而被重命名。因此，假如所有出现在 ω 中的x都被y代替，（∀x）ω 是与（∀y）ω等价的。对存在量词来说也是如此。

既然量化变量符号给予谓词演算如此强劲的表达力，你也许会问我们是否也能量化关系符号和函数符号。在谓词演算的例子中，这样的量化是不允许的。为此，它被称为一阶谓词演算。二阶和高阶谓词演算允许量化关系和函数符号，但是要使用极其复杂的推理机制。

15.5 量词语义学

15.5.1 全称量词

在所有变量符号 ξ 对在论域中的对象的指派来说 ω（ξ）都有真值的情况下，（∀ξ）ω（∀ξ），

才有真值（在给定的对象常量、函数常项和关系常量对对象、函数和关系的指派下）。

例如：假设我们对Clear和On使用两个由图15-2的配置所提示的解释，在每一种这样的解释下，什么是（∀ x）[On（x，C）⊃ ¬ Clear（C）]的真值？在这些解释中，变量*x*可以被指派给A、B、C或者Floor。我们必须为每一个解释弄清每一个这样的指派。例如，把*x*指派给A，假如On（x，C）⊃ ¬ Clear（C）要有真值，我们必须让<A，C>不在关系On中，或者必须让<C>不在关系Clear中。事实上，在每一种解释中，<C>在关系Clear中，但是<A，C>不在关系On中，所以，第一个（x对A的）指派产生真值。弄清余下的指派留给你自己去做。

15.5.2 存在量词

（∃ ξ ）ω（ξ），只是在就至少一个变量符号ξ 对在论域中的对象的指派来说 ω（ξ）都有真值的情况下，才有真值（在给定的对对象、函数和关系的指派下）。

15.5.3 有用的等价式

按照约定的量词的语义学，与德·摩根定律相类似，可以建立下列等价式：

¬(∀ξ) ω（ξ）≡（∃ξ）¬ ω（ξ）

¬(∃ξ) ω（ξ）≡（∀ξ）¬ ω（ξ）

以及前面已经提到的：

（∀ξ）ω（ξ）≡（∀η）ω（η）

15.5.4 推理规则

适当地一般化后，命题演算的推理规则可以用于谓词演算。这些包括假言推理、∧的引入和消除、∨ 的引入、¬ 的消解以及归结。将在下一章介绍归结。除了命题演算规则，还有两个重要的问题：

- 全称例化（*universal instantiation*，*UI*）。从（∀ξ）ω（ξ）推出 ω（α），其中，ω（ξ）是具有变量 ξ 的任何合式公式，α是任何常量符号，ω（α）是 ω（ξ）用 α 在 ω 中替换 ξ 得到的。

  例如：从（∀x）P（x，f（x），B）推出P（A，f（A），B）。
- 存在一般化（*existential generalization*，*EG*）。从 ω（α）推出（∃ξ）ω（ξ），其中，ω（α）是一个包含常项符号 α 的合式公式，而 ω(ξ) 是一个用 ξ 在 ω 中代替每一个 α 出现的形式。

  例如：从（∀x）Q（A，g（A），x）推出（∃y）（∀x）Q（y，g（y），x）。

  容易看出UI和EG是合理的推理规则。

15.6 谓词演算作为一种表示知识的语言

15.6.1 概念化

对人工智能来说重要的是说什么，而不是怎样说它。谓词演算不过是提供一种形式统一的语言。用这种语言，有关这个世界的知识可以被表示和推断。如此表示知识的结果可以通过下一章描述的逻辑演绎方法来探索。

表示关于一个世界的知识的第一步是用对象、函数和关系把它概念化。概念化通常包含一种对概念化对象的部分进行创造的行为。常常会有种种关于我们认为什么样的对象会存在于我们的世界中的选择。我们可以自由运用任何我们所希望的方式来概念化这个世界；尽管如此，有些概念化会比另外一些更有用（未必更“精确”）。

下一步，我们构造谓词演算表达式，它要表示的意义包含对象、函数和关系。最后，我们写出在概念化了的世界中得到满足的合式公式。这些合式公式也同样可由别的解释得到满足；我们只是需要关注在某些解释下它们不会得到满足，这些解释是我们关于这个世界的知识水平所能排除的。

当设计能够推断现实世界（而不是想像的）并与其进行相互作用的agent时，概念化要有基础是十分重要的。就是说，通过与这个世界相联系的感知机制，某些被用在知识库中的原子的真值必须是可估价的。另一些原子可以用这些初始的、感知的原子来定义，但是，假如由逻辑方法产生的结论必须与机器人存在的世界相关联，整个结构就必须依赖于某种感知。另一方面，无需用数学作基础，因为数学的论述并不一定是物质世界。

John McCarthy [McCarthy 1958]第一次正式提出在人工智能领域使用谓词演算来表示有关这个世界的知识。[Guha & Lenat 1990，Lenat 1995，Lenat & Guha 1990] 描述了一个表示数以百万计的、关于这个世界的常识的大型工程（CYC 工程）。对人工智能中的逻辑的更完整的讨论，可参考[Nilsson 1991]。而基于逻辑的人工智能的教材，可参见[Genesereth & Nilsson 1987]。

15.6.2　举例

用一些例子来说明有关一个世界的知识概念化的过程。假设我们设计了一个agent来发送一幢大楼里的包裹。另外，我们还需要有一个关系常量，它指称某些是包裹的特性，此处用Package（x）表示；特别地，还需要一个关系常量，它指称某一个对象在某一个房间里，用Inroom（x，y）表示。另外还需要有一个关系常量，它指称某一个对象小于另一个对象。用这些，我们可以在谓词演算中表示下列关于这个机器人世界的例子：

- “All of the packages in room 27 are smaller than any of the packages in room 28.”

  (∀x, y) {[Package (x) ∧ Package (y) ∧ Inroom (x, 27) ∧ Inroom (y, 28)] ⊃ Smaller (x, y)}

- “Every package in room27 is smaller than one of the packages in room 29。”

  这一叙述是有歧义的（它是日常用语中的一种常见现象）。它可表示为下列两个公式的任意一个（它们不是等价的）：

  (∃y)(∀x) {[Package (x) ∧ Package (y) ∧ Inroom (x, 27) ∧ Inroom (y, 29)] ⊃ Smaller (x, y)}

  (∀x)(∃y) {[Package (x) ∧ Package (y) ∧ Inroom (x, 27) ∧ Inroom (y, 29)] ⊃ Smaller (x, y)}

一些关于机器人世界的知识也许要求我们概念化时间观念。例如，考虑这样的叙述“包裹A在包裹B前到达”。一个可推断的概念化应具有时间间隔和关于这种间隔的序列关系。我们必须有一种方法陈述一个对象的到达时间。用Arrived（x，z）表示，其中*x*指一个到达的对象，*z*指*x*到达的时间间隔。对此，我们可以写为

(∃z1, z2)[Arrived (A, z1) ∧ Arrived (B, z2) ∧ Before (z1, z2)]

我们在第18章中更详细地讨论了一个为时间间隔的概念化过程。也有一些别的方法，有些包含已被用于计算机科学和人工智能去处理时间的时态逻辑（*temporal logic*）[Emerson 1989, Shoham 1987]。

创造概念化会常常涉及相当棘手的问题。例如，要决定怎样处理像在陈述句“在28号房间中的包裹包括一夸脱牛奶”中的“牛奶”这样的物质名词是很难的。牛奶是一个具有我们所说的白色这一特性的对象吗？如果是，当我们把一夸脱牛奶分成两品脱时会发生什么呢？它会变成两个对象（两个都是白的），还是仍然是一个对象？许多像这样的问题依然是留待继续研究的课题。

在本书的后面，我们要使用概念化，把无形的东西，如情形和行为，处理为对象。并将引入外延到谓词演算，使它可以允许一个agent叙述有关另一个agent的知识，如“机器人A知道包裹B在28号房间里”。

15.7 补充读物和讨论

如前所述，用逻辑语句来表示知识和用逻辑推理过程来推理已在人工智能领域引起争论，相关的讨论可参见[McDermott & Doyle 1980]。某些争议必须处理一些不匹配，它存在于归因于[Tarski 1935]（英语译本见[Tarski 1956]）的逻辑语言的精确语义学与表述真实世界知识（与数学知识相对的）所需的更易变的和更依赖上下文的观念之间。

Tarski的语义把确定世界的个体与在语言中的对象常量联系起来。另一种观点强调索引—功能(*indexical-functional*)的表述。用Agre 和Chapman [Agre & Chapman 1990, P. 21]的话说就是：

> 尽管传统上将语义表示为一个agent头脑中的符号与这个世界中具有个性化的对象之间的对应关系，我们的理论描述这个agent与这个世界上具有索引个性化和功能个性化的实体之间的因果关系。例如，Pengi（在[Agre 和Chapman 1987]中描述）作用的实体之一是the-bee-I-am-chasing。这个实体，就它与agent（“I”）关系的定义而言，它在索引上被赋予个性。它也在功能上被赋予个性，用一个agent正在进行的工程之一（追一只蜜蜂）来定义。与传统的表述相对照，符号BEE-34 和BEE-35将总是指称同样的两只蜜蜂，不同的蜜蜂也许是在不同时间的the-bee-I-am-chasing。

然而，如我们在以下的章节中将看到的，研究者们已经能够应用逻辑语言（带有某些扩展）对人工智能进行表述和推理。

[Enderton 1972, Pospesel 1976]是两本论述逻辑的书——前一本更具数学性，而后一本则少点形式上的东西。从可读性来看，包括示例性的计算机程序，可参见[Barwise 和 Etchemendy 1993]。

习题

15.1 给出一个下列域是整数的公式的模型：

$On(A, Fl) \supset Clear(B)$

$Clear(B) \wedge Clear(C) \supset On(A, Fl)$

$Clear(B) \vee Clear(A)$

$Clear(B)$

$Clear(C)$

15.2 说明合式公式（∃x）On（x，A）⊃ ¬ Clear（A）在每一种由图15-2的配置所提出的解释中有真值。

15.3 下列合式公式有一个明显的“积木世界”的解释：

On(C, A)

On(A, Fl)

On(B, Fl)

Clear(C)

Clear(B)

(∀x)[Clear(x) ⊃ ¬(∃y) On(y, x)]

作出另一种满足这些合式公式的合取式的解释（对象、关系和一个映射）。

15.4 就下列每一个谓词演算表达式，说出它是否在句法上能成为一个合法的句子（合式公式），并且，如果不能，为什么？就每一个合法的句子，说出它是否永真（必然为真）、不可满足（必然为假）或是偶然的（依赖于解释）。就每一个偶然的句子，给符号一种解释使句子为真。

1) Won(Election, Clinton) ∧ Won(Election, Bush)

2) ¬

3) (∀s)[Attends(s, (CS221 ∨ CS157))]

4) (∀x)[Set(x) ⊃ (Subset(x, x) ∨ ¬Subset(x, x))]

5) (∀x)[P(x) ∨ Q(x)] ⊃ ¬(∃x)[(¬P(x) ∨ Q(x))]

6) (∀x)[IsTrue(x)]

7) (∃x y z)[Equal(add(x, y), add(x, y, z))]

8) P(Igor) ∧ Q(Bertha) ∧ (∀x)[(¬P(x) ∨ Q(x))]

9) (¬(∃x)) ⊃ NonExistent(x)

15.5 一个做成尖塔形的机器人寻找两块上面都是空的积木，并把一块放在另一块的上面（假如被移动的积木可以被置于别的积木和地板上）。请写出一个一阶谓词演算表达式，它可以用于决定这里是否存在两块这样的积木。

15.6 一个在网格世界中的机器人可以移到一个单元中，假如这个单元是四个相邻单元之一，并且是没有障碍的。在这种情况下，我们说这个单元是可用的。同时，网格世界没有“稠密空间”（如第2章中的定义）。作出谓词演算表达式，它可以用来定义何时一个单元是可用的，并描述这个“非稠密空间”的条件。

第16章　谓词演算中的归结

16.1　合一

合式公式（$\forall \xi_1, \xi_2, \cdots, \xi_n)(\lambda_1 \vee \lambda_2 \vee \cdots \vee \lambda_k$）可缩写为$\lambda_1 \vee \lambda_2 \vee \cdots \vee \lambda_k$，其中$\lambda_1, \lambda_2, \cdots, \lambda_i$是可能包含变量$\xi_1, \xi_2, \cdots, \xi_n$的文字。也就是说，仅仅去掉了全称量词，并假定$\lambda_i$中任何变量全称量化（后面将说明如何能首先消除任何存在的量化变量）。这种缩写形式的合式公式叫做子句。有时，用集合符号$\{\lambda_1, \lambda_2, \cdots, \lambda_k\}$表达一个子句，并假定集合中的元素是析取的。

如果两个子句的文字相匹配但是互补，我们能归结它们——就像在命题演算中一样。如果一个子句中有一个文字$\lambda(\xi)$（ξ是一个变量），而另一个子句中有一个互补文字$\neg\lambda(\tau)$，τ是不包含ξ的某个项，我们能把第一个子句中的所有ξ用τ代替，然后对互补文字进行命题归结以产生那两个子句的归结式。

举例：考虑两个子句P(f(y), A) ∨Q(B, C)和¬P(x, A) ∨R(x, C) ∨S(A, B)。用f(y)代替第二个子句中的x产生¬P(f(y), A) ∨R(f(y), C) ∨S(A, B)。现在，两个子句中的第一个文字刚好互补，因此我们能对文字（f(y), A）执行一次归结，产生归结式R(f(y), C) ∨S(A, B) ∨ Q(B, C)。

用一个被称为合一的方法计算适当的置换。合一在AI中是一个极其重要的方法。为了描述它，必须先讨论一下置换。

一个表达式项可能是变量符号、对象常量或者函数表达式，后者包含函数常量和表达式项。一个表达式的置换实例通过置换那个表达式中的变量项而得到。因此，P[x, f(y), B]的四个置换实例是：

```
P[z, f(w), B]
P[x, f(A), B]
P[g(z), f(A), B]
P[C, f(A), B]
```

上面第一个实例称为原始文字的字母变种（*alphabetic variant*），因为我们仅仅用另外的变量代替了P[x, f(y), B]中出现的变量。第4个叫基例（*ground instance*），因为文字中没有一项包含变量（一个基项是一个不包含任何变量的项）。

我们能通过一组有序对$s = \{\tau_1/\xi_1, \tau_2/\xi_2, \ldots, \tau_n/\xi_n\}$来表达任何置换。$\tau_i/\xi_i$对意思是说$\tau_i$项替换在整个置换范围内的$\xi$的每次出现。而且，变量不能被一个包含相同变量的项代替。使用前面P[x, f(y), B]的四个实例的置换是：

```
s1 = {z/x, w/y}
s2 = {A/y}
s3 = {g(z)/x, A/y}
s4 = {c/x, A/y}
```

用ωs来指称一个使用置换s的表达式ω的一个置换实例。因此，P[z, f(w), B] = P[x, f(y),

B]s1。两个置换*s*1和*s*2的组合用*s*1*s*2指称，它指的是这个置换通过先把*s*2应用到*s*1各项，再加上不含出现在*s*1中变量的所有*s*2对而得到，因此：

{g(x, y)/z} {A/x, B/y, C/w, D/z} = {g(A, B)/z, A/x, B/y, C/w}

可以看出，把*s*1和*s*2连续地应用到ω表达式和把*s*1*s*2应用到ω 是相同的，即：(ω s1) s2 = ω (s1s2)。也能看出，置换组合是符合结合律的。即：(s1s2) s3 = s1 (s2s3)。

举例 ω是P (x, y)，*s*1是{f(y) / x}，*s*2是{A/y}。那么

(ω*s*1) *s*2 = [p(f(y), y)]{A/y} = P(f (A), A)

和

ω(*s*1*s*2) = [P(x, y)]{f(A) / x, A/y} = P(f(A), A)

一般地讲，置换不符合交换律，即*s*1*s*2 = *s*2*s*1是不成立的。因此，改变应用置换顺序会产生差异。

例如（使用前面例子中的ω、*s*1和*s*2）

ω (*s*1*s*2) = P (f(A), A)

ω (*s*2*s*1) = [P(x, y)]{ A/y, f(y) / x } = P(f(y), A)

当一个置换*s*被应用到一个表达式集合{ω_i}的每一个成员时，用{ω_i}*s*表示置换实例集合。如果存在一个置换*s*，它使$\omega_1 s = \omega_2 s = \omega_3 s = \cdots$，我们说表达式集合{$\omega_i$}是可以合一的（*unifiable*）。在这种情况下，*s*被称为{ω_i}的一个合一式(*unifier*)，因为它的使用把集合压缩成为一个单元素集合。例如：*s* = {A/x, B/y}把集合{p[x, f(y), B], P[x, f(B), B]}合一产生{p[A, f(B), B]}。虽然*s* = {A/x, B/y}是集合{p[x, f(y), B], p[x, f(B), B]}的一个合一式，但在某种意义上它不是最简单的合一式。我们注意到确实不必用A置换x来获得合一。最一般（或最简单）的合一式（mgu），{ω_i}的*g*有下面的特性：如果*s*是产生 {ω_i}*s*的{ω_i}的任意合一式，那么存在一个置换s'以使{ω_i}*s* = {ω_i}*gs'*。而且，经一个最一般的合一式产生的通用实例除了字母变化外是惟一的。

有很多算法可以用来找到一个可以合一的表达式的有限集合的mgu，并且当那个集合不能被合一时能返回失败。这里给出的算法UNIFY是从[Chang & Lee 1973，P. 77]给出的一个算法改编的。它工作在一个列表结构的表达式集合上，在这些表达式中，每个文字和项作为一个列表项。例如：¬P(x, f(A, y)写为（¬P x （f A y））列表结构形式，表达式 ¬P是列表中的第一个顶级表达式，（f A y）是第三个顶级表达式。

UNIFY的基础是分歧集（*disagreement set*）的思想。一个非空的表达式集合W的分歧集由下面的方法得到：首先定位第一个符号（从左边计数），在这个位置不是W中的所有表达式有完全相同的符号，然后从W的每个表达式中提取从占据那个位置的符号开始的子表达式，各个子表达式集合构成W的分歧集。例如，两个列表{（¬P x (f A y)），(¬p x (f z B))}集合的分歧集是{A，z}。分歧集能用置换A/z产生协调。

UNIFY（Γ）（Γ是一个列表表达式集合）

1) k ← 0，Γ_k ←Γ，σ_k ← ε (初始化步骤；ε是空的置换)。

2) 如果Γ_k是一个单元素集合，用Γ 的mgu σ_k 退出；否则继续。

3) D_k ←Γ_k的分歧集。

4) 如果在D_k中存在元素v_k和t_k，v_k是一个不会出现在t_k中的变量，则继续；否则，失败退出，Γ是不可以合一的。

5) $\sigma_{k+1} \leftarrow \sigma_k\{t_k/v_k\}$，$\Gamma_{k+1} \leftarrow \Gamma_k\{t_k/v_k\}$(注意$\Gamma_{k+1} = \Gamma_k\sigma_{k+1}$)。

6) $k \leftarrow k+1$

7) 转到第2步。

[Chang & Lee 1973, p.79]证明了UNIFY或者能找到一个可以合一表达式集合的一个最一般的合一式；或者当表达式不能合一时，能报告失败。算法产生mgu作为一个列表结构表达式对，但这些能被方便地转换为一阶逻辑中使用的形式。有几个算法可以执行合一化过程，包括一个以线性时间运行的算法[Paterson & Wegman 1978]。

作为例子，列出了几个文字集合的最一般置换实例（它们由mgu获得）。可以对文字集合中的每一个逐步执行UNIFY算法。

文字集合	最一般置换实例
{P(x), P(A)}	P(A)
{P[f(x), y, g(y)], P[f(x), z, g(x)]}	P[f(x), x, g(x)]
{P[f(x, g(A, y), g(A, y))], P[f(x, z), z]}	p[f(x, g(A, y)), g(A, y)]

在UNIFY的第4步，检查是否一个变量出现在我们将要置换的一个项中。这样的置换会导致一个无限循环，因此不允许。作为一个例子，假如我们企图合一P(x, x)和P(f(z), z)。UNIFY首先用f(z)置换这些表达式中的x，产生P(f(z), f (z))和P(f(z), z)。如果不进行检查，它将用f(z)置换z产生P(f(f(z)), f(f(z)))和P(f(f(z)), f(z))，等等。

16.2 谓词演算归结

假如γ_1和γ_2是两个子句（表示为文字集合）。如果γ_1中有一个原子ϕ，γ_2中有一个文字$\neg\ \psi$，并使ϕ和ψ有一个最一般合一式，那么这两个子句有一个归结式ρ，它通过把置换μ与γ_1和γ_2减去互补其文字的并集而得到。

即：$\rho = [(\gamma_1 - \{\phi\}) \cup (\gamma 2 - \{\neg\ \psi\})]\mu$

在两个子句被归结前，为了避免变量混淆，我们对每个子句中的变量重命名以使一个子句中的变量不会出现在另一个中。例如：假如我们正在归结P(x) ∨ Q(f(x))与R(g(x)) ∨ ¬ Q(f(A))，首先重写第二个子句，比如说为R(g(y)) ∨ ¬ Q(f(A))，然后执行归结获得P(A) ∨ R(g(y))。变量重命名被称为对变量进行标准化(*standardizing the variables apart*)。

下面是一些例子。

{P(x), Q(x, y)}和{¬P(A), R(B, z)}归结产生{Q(A, y), R(B, z)}。

{P(x, x), Q(x), R(x)}和{¬P(A, z), ¬Q(B)}可用两种不同的方式归结，分别产生{Q(A), R(A), ¬ Q(B)}和{P(B, B), R(B), ¬P(A, z)}。

有时需要对谓词演算归结有一个稍强的定义。例如，考虑两个子句{P(u), P(v)}和{¬P(x), P(y)}。这两个子句各自有基例P(A)和 ¬P(A)（由置换A/u, A/v, A/x, A/y获得）。从这些基例中，能推断出空子句，因此我们应该能从初始子句推断它，但是刚刚给定的归结规则不能做到这些。更强的规则如下。

假定γ_1和γ_2是两个子句（再次表示为文字集合）。如果有γ_1的一个子集γ_1'和γ_2的一个子集γ_2'使得γ_1'的文字能与γ_2'的文字的否定式用最一般合一式μ合一，那么这两个子句有一个归结式ρ，它通过把置换μ与γ_1和γ_2减去其互补子集的并集而得到。即：

$\rho = [(\gamma_1 - \gamma_i) \cup (\gamma_2 - \gamma_i)]\mu$

用这个归结定义，归结子句{P(u)，P(v)}和{¬p(x)，¬P(y)}产生空子句。

16.3　完备性和合理性

谓词演算的归结是合理的。也就是说，如果ρ是两个子句ϕ和ψ的归结式，那么$\{\phi, \psi\} \models \rho$这个事实的证明不比命题归结的合理性证明难。就如同在命题演算中一样，归结的完备性相对更难一些。我们不能只根据归结就推论出被一个给定集合逻辑涵蕴的所有公式。例如：我们不能从P(Ac中推断P(A)∨ P(B)。在命题归结中，我们用归结反驳克服这个困难，在谓词演算中也能如此。然而，为了保证反驳的完备性，我们必须用刚刚给定的更强的归结规则。

16.4　把任意的合式公式转化为子句形式

就像在命题演算中一样，任何合式公式能被转化为子句形式。步骤如下：

1) 消除蕴含符号（如在命题演算中一样）。

2) 减少否定符号的范围（如在命题演算中一样）。

3) 变量标准化，由于量词范围内的变量像“哑元”，因此它们能被更名，以使每个量词有它自己的变量符号。

举例：(∀x)[¬P(x) ∨ (∃x)Q(x)]可以改写为(∀x)[¬P(x) ∨ (∃y)Q(y)]

4) 取消存在量词。

举例：在(∀x)[(∃y)Height(x, y)]中，存在量词在一个全称量词范围内，因此，y的“存在”可能依赖x的值。例如，如果(∀x)[(∃y)Height(x, y)]的意思是“每个人x有身高y”，那么很明显身高与人有关。用某个未知的函数h(x)显式定义这个依赖关系，h(x)把x的每个值映射为存在的y值。这样的一个函数叫做Skolem函数[⊖]。如果我们把Skolem函数用在y存在的位置，就能取消存在量词，并写为(∀x)Height[x, h(x)]。

从一个合式公式中消除一个存在量词的一般规则是用一个Skolem函数代替存在量词作用范围内变量的每一次出现，Skolem函数的参数是那些被全称量词约束的变量，全称量词的范围包括要被消除的存在量词的范围。用在Skolem函数中的函数符号必须是“新的”，不能是已经出现在任何合式公式中被用在归结反驳中的符号。

因此，我们能从

$$[(\forall w)Q(w)] \supset (\forall x)\{(\forall y)\{(\exists z)[P(x, y, z) \supset (\forall u)R(x, y, u, z)]\}\}$$

经消除(∃z)，产生

$$[(\forall w)Q(w)] \supset (\forall x)\{(\forall y)[P(x, y, g(x, y)) \supset (\forall u)R(x, y, u, g(x, y))]$$

我们能从

$$(\forall x)\{\neg P(x) \vee \{(\forall y)[\neg P(y) \vee P(f(x, y))] \wedge (\exists w)[Q(x, w) \wedge \neg P(w)]\}\}$$

经消除(∃w)，产生

$$*(\forall x)\{\neg P(x) \vee \{\forall y[\neg P(y) \vee P(f(x, y))] \wedge [Q(x, h(x)) \wedge \neg P(h(x))]\}\}$$

⊖ Skolem 函数是根据逻辑学家Thoralf Skolem [Skolem 1920] 的名字命名的。

其中g(x, y)和h(x)都是Skolem函数。

如果要被消除的存在量词不在任何全称量词的范围内，那么我们使用一个没有参数的Skolem函数，它只是一个常量。因此，(∃x) P(x)成为P(Sk)，常量符号Sk用来指向我们知道存在的那个项。另外，Sk是一个新的符号常量且没有用在其他函数中，这是必要的。

为了从一个合式公式中消除所有的存在量化变量，依次对每个子公式使用前述的过程。从一组合式公式中消除存在量词产生公式集合的Skolem范式。

注意，一个合式公式的Skolem范式并不等价于原始的合式公式！公式(∃x) P(x)被它的Skoem范式P(Sk)逻辑涵蕴，但反过来就不行。作为另一个例子，注意[P(A) ∨ P(B)]⊨(∃x) P(x)，但是[P(A) ∨ P(B)]⊭P(Sk)。公式集合Δ是可以满足的，当且仅当Δ的Skolem范式是可以满足的。或者，对归结反驳更有用的是，Δ是不可满足的，当且仅当Δ的 Skolem范式是不可满足 [Loveland 1978，PP.41 以后]。

5) 转化为前束范式。在这个阶段，没有任何残留的存在量词，每个全称量词有它自己的变量符号。现在，我们可以把所有的全称量词移到合式公式的前面，并且让每个量词的范围包括跟随它的合式公式的全部。这样的合式公式被称为是在前束范式中。前束范式的一个合式公式包括一个称为前束范式的量词，它后面跟随一个称为矩阵（*matrix*）的没有量词的公式。前面标有 * 的合式公式例子的前束范式是：

$$(\forall x)(\forall y)\{\neg P(x) \vee \{[\neg P(y) \vee P(f(x,y))] \wedge [Q(x,h(x)) \wedge \neg P(h(x))]\}\}$$

6) 将矩阵写成一个合取范式。像命题演算一样，我们可以通过重复使用分配律，用（$\omega_1 \vee \omega_2$）∧（$\omega_1 \vee \omega_3$）代替$\omega_1 \vee (\omega_2 \wedge \omega_3)$，把任何矩阵改写成合取范式。

将前述合式公式例子的矩阵改写成合取范式，它变为

$$(\forall x)(\forall y)\{[\neg P(x) \vee \neg P(y) \vee P(f(x,y))] \wedge [\neg P(x) \vee Q(x,h(x))] \wedge [\neg P(\vee \neg P(h(x))]\}$$

7) 消除全称量词。由于用在合式公式中的所有变量必须是在一个量词范围内，保证在这一步保留的所有变量都被全称量化。而且，全称量化的顺序并不重要，因此可以删去明确出现的全称量词，并且根据约定可以保证矩阵中的所有变量都被全称量化。现在，只留下了一个合取范式的矩阵。

8) 消除∧符号。可以用合式公式集合$\{\omega_1, \omega_2\}$代替表达式（$\omega_1 \wedge \omega_2$），删除显式出现的∧符号。重复代替的结果是获得一个有限的合式公式集合，合式公式中的每一项是一个文字析取项。只包含文字析取项的任何合式公式被称为一个子句，我们的合式公式例子被转化为下面的子句集合：

$$\neg P(x) \vee \neg P(y) \vee P[f(x,y)]$$

$$\neg P(x) \vee Q[x, h(x)]$$

$$\neg P(x) \vee \neg P[h(x)]$$

9) 变量更名。变量符号可以被更名，以使没有任何变量符号出现在多于一个的子句中。注意到(∀x)[P(x) ∧Q (x)]等价于[(∀x)P(x) ∧((∀y)Q (y))。现在子句是：

$$\neg P(x1) \vee \neg P(y) \vee P[f(x1,y)]$$

$\neg P(x2) \vee Q[x2, h(x2)]$

$\neg P(x3) \vee \neg P[h(x3)]$

一个子句的文字可以包含变量，但这些变量总是被理解为是全称量化的。

16.5　用归结证明定理

当归结用作一个定理证明系统的推论规则时，可以将要从中证明一个定理的合式公式集合首先转化成子句。可以证明如果合式公式ω从一个合式公式集合Δ中逻辑地产生，那么同样也从Δ中的合式公式转换成的子句集合中逻辑地产生。反之亦然[Davis & Putnam 1960]。因此，子句是一个表达谓词演算合式公式的完备的通用形式。

归结反驳能被证明既是合理的也是完备的[Robinson 1965]（也见[Chang & Lee 1973，p.85]）。因此，为了证明来自Δ 的一个合式公式ω，我们像在命题演算中一样进行。对ω取反，把这个否定转换为子句形式，并把它加到Δ的子句形式中；然后应用归结直到推出空子句。我们能用任何在命题演算中讨论的与归结有关的排序和精炼策略。

回到上一章提到的搬箱子机器人的一个例子。假如这个机器人知道27号房间中的所有箱子都比28号房间中的小。即：

1) $(\forall x, y)\{[Package(x) \wedge Package(y) \wedge Inroom(x, 27) \wedge Inroom(y, 28)] \supset Smaller(x), y\}$

缩写谓词符号使公式更紧凑。转换为子句形式，产生：

2) $\neg P(x) \vee \neg P(y) \vee \neg I(x, 27) \vee \neg I(y, 28) \vee S(x, y)$

假定机器人知道箱子A在27号或28号房间中（但不知道是哪一个）。它知道箱子B在房间27中且B不比A小。

3) $P(A)$
4) $P(B)$
5) $I(A, 27) \vee I(A, 28)$
6) $I(B, 27)$
7) $\neg S(B, A)$

用归结反驳，机器人能证明箱子A在27号房间中。图16-1是一棵证明树。待证明的合式公式的否定被标在左上角；和机器人明确知道的相对应的合式公式被标在图中沿右手方向。归结期间的置换在图中也已表示出来。

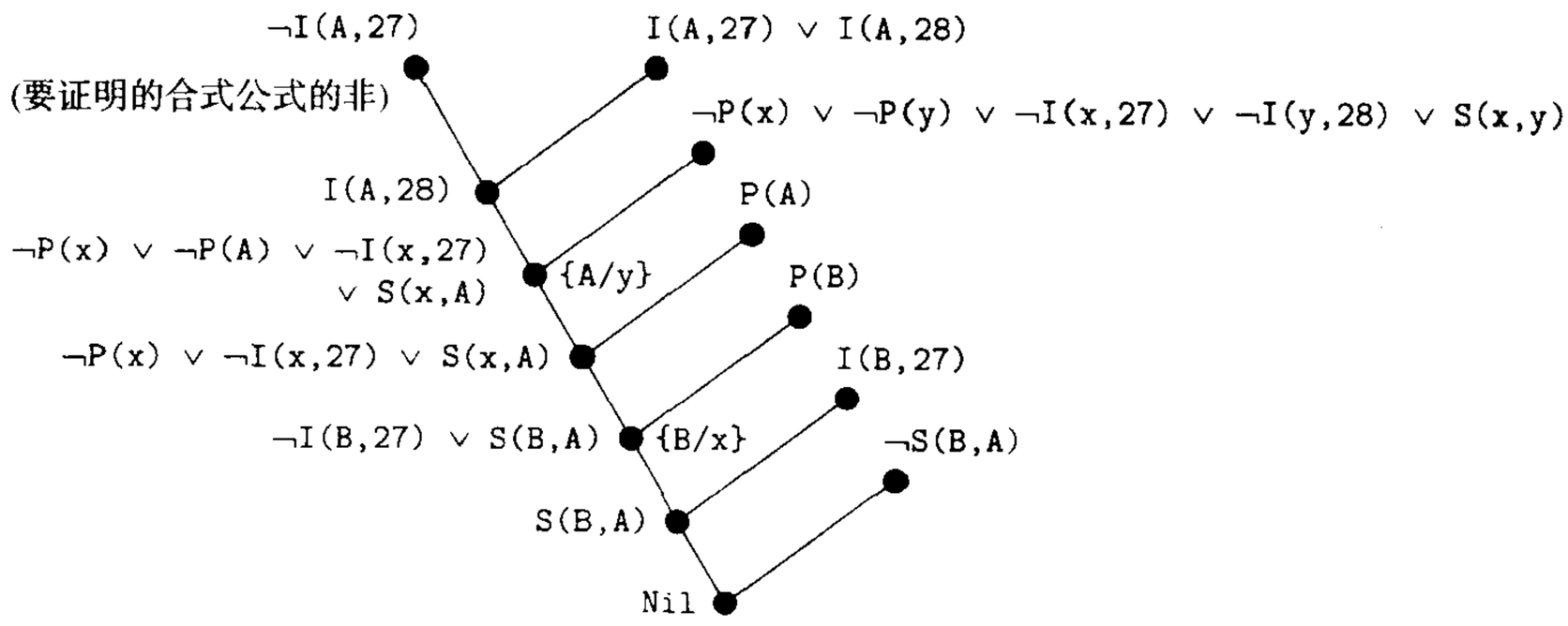


图16-1　一个归结反驳

16.6 回答提取

为了用归结回答由谓词演算合式公式表示一个领域知识的问题，通常要做的不只是证明一个定理。例如，假定必须证明形如（∃ξ）ω（ξ）的一个定理。我们可能也想得到已经证明存在的ξ。为了做到这些，必须跟踪在反驳过程中所做的置换，可以用一个回答文字策略跟踪那个置换。将文字Ans(ξ_1，ξ_2，...)加到要证明的定理的否定子句，执行归结直到只剩下一个回答文字。在Ans文字中的变量是出现在待证明定理的否定子句形式中的所有变量。在证明期间代替Ans文字中变量的项，将成为被证明合式公式中存在量化变量的实例。回答提取由[Green 1969b]发明，它后来被[Luckham & Nilsson 1971]分析并扩展。也可参见[Nilsson 1980，pp 175以后]。

在图16-2中，给出了如何使用Ans文字的一个例子，现在要证明合式公式(∃u)I(A，u)，它可以看作是机器人问它自己："A究竟在哪个房间？"。

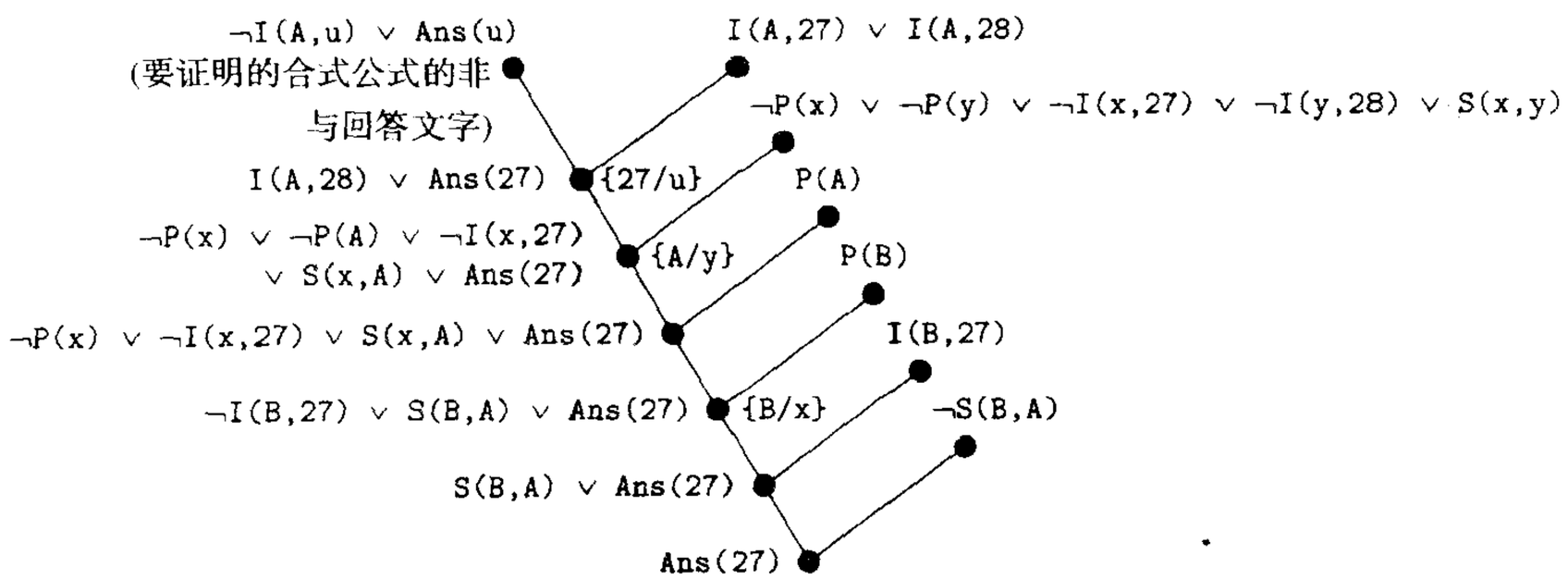


图16-2 回答提取

16.7 等式谓词

用在一个知识库公式中的关系常量常常有意向含意（即，关系），但是这些关系仅仅被知识库的模型集合所约束，根本不会被用在关系常量中的特殊符号所约束。归结反驳的结果将和意图含意相一致，只要知识库适当地约束实际的关系。

然而，有一些重要且经常出现的关系，我们可能想让所有的知识库描述它们。例如等式关系，我们可以使用前缀形式Equals(A，B)或者中缀形式A = B来表示一个关系常量。仅仅因为在知识库中包括了公式Equals(A，B)，并不意味着能够从P（A）中推出P（B），正如我们也不能归结¬Q(B，A)和Q(A，B)一样。没有附加的公式，知识库无法知道Equals是什么。

有一些著名的等式属性能用公式表达，我们能把知识库加到这些公式中。这些等式关系是：

- 自反性(∀x) Equals(x，x)
- 对称性(∀x，y)[Equals(x，y) ⊃ Equals(y，x)]
- 传递性（∀x，y，z）[Equals(x，y) ∧ Equals(y，z) ⊃ Equals(x，z)]

即使有这些公理，我们也不能从P（A）和Equals（A，B）证明P（B）。需要的是能把相等项置换为任何表达式的相等项。为了支持这种置换，知识库将另外需要显式列出所有允许的置换（对每个谓词常量和函数常量）——这是一个不切实际的要求。

实际上有几个可能的选择。最强的一个叫调解（*Paramodulation*）[Wos & Robinson 1968]，[Chang & Lee 1973，pp.168~170]。调解是一个特定的等式推论规则，在知识库包含等式谓词的

情况下它和归结混合使用。规则定义如下：

假如γ_1和γ_2是两个子句（表示为文字集合）。如果$\gamma_1 = \{\lambda(\tau) \cup \gamma_1'\}$，$\gamma_2 = \{\text{Equals}(\alpha, \beta) \cup \gamma_2'\}$ τ、α和β是项，γ_1' 和 γ_2' 是子句，$\lambda(\tau)$是一个包含项τ的文字（可能在其他条目之中）。如果τ和α有一个最一般合一式σ，那么推导出γ_1和γ_2的二元调解式（*binary Paramodulant*）：

$$\pi = \{\lambda\sigma[(\beta\sigma)] \cup \gamma_1'\sigma \cup \gamma_2'\sigma\}$$

其中 $\lambda\sigma[(\beta\sigma)]$表示用$\beta\sigma$代替在$\lambda\sigma$中单个$\tau\sigma$出现的结果（等式的对称性被一对倒置$\alpha$和$\beta$角色的规则处理）。

在操作过程中，规则不像它的正式定义显示的那样复杂。作为第一个例子（用大锤砸坚果），从P（A）和（A = B）证明P（B）（为简短，用中缀符号）。对一个反驳类型的证明，我们必须从子句 P（B）、P（A）和（A = B）推导出空子句。对上面两个子句使用调解法，$\lambda(\tau)$是P（A），τ是A，α是A，β是B。由于A（在τ中）和A（在α中）普通地合一而没有置换，二元调解式是P（B），它是用β（即B）代替τ(即A)产生的一个结果。用 ¬ P（B）归结这个二元调解式产生了空子句。

把下面的例子（来自[Chang & Lee 1973，p.170]）留给读者去思考：

P(g(f(x))) ∨ Q(x)　和　[f(g(B)) = A] ∨ R(g(C))的二元调解式是

$$\text{P(g(A))} \vee \text{Q(g(B))} \vee \text{R(g(C))}$$

对这种允许的调解式作一个轻微的扩展（超过二元的），能够证明与归结反驳组合的调解对包含等式谓词的知识库是完备的。

对那些不需要用相等置换相等的问题，就不再需要调解的能力了。在很多情况下，我们用一个叫做谓词评估的技术来处理。如果一个外部处理能返回一个等式谓词的真值，我们可以用适当的T或F代替那个谓词（当它出现在一个公式中时）。在归结反驳中，包含文字T的子句能被取消。任何子句中的文字F能被消除（它总能被一个假定的无所不在的T所归结。）

例如，考虑问题（它可能是关于一个推东西的机器人）：证明如果一个箱子A在一个特定的屋子*R*1中，那么它不能在一个不同的屋子*R*2中。机器人在它的知识库中将有如下的语句：

(∀x, y, u, v) [In(x, u) ∧ (u ≠ v)] ⊃ ¬ In(x , v)

In(A, R1)

⋮

它试图证明¬In (A, R2)。把第一个公式转化为子句形式，产生

¬In (x, u) ∨ (u = v) ∨ ¬ In(x , v)

该策略延迟处理等式谓词，直到它们只包含基本项。用待证明的合式公式的否定归结子句，产生

(R2 = v) ∨¬In(A , v)

用给定的合式公式 In(A, R1)归结这个结果，产生：

(R2 = R1)

现在，我们可以想像知识库事实上包含合式公式 ¬ (R2 = R1)。如果这样，我们能用它产生空子句，完成反驳。但是在有*M*个房间的大建筑物中，我们将需要$\frac{M(M-1)}{2}$个这种不等式。如果机器人必须做涉及到数字的推理，它可能需要一个大得无法管理的合式公式集合，如 ¬（3724 = 4861），等等。并非将所有的这些合式公式显式地放在知识库中，而是提供一个例程（即一个程序），它能够对所有的（基本）α和β的表达式（$\alpha = \beta$）进行评估，这样将会更好。在例子中，

程序对表达式（R2= R1）将返回F（或Nil），反驳将完成。

几个其他常用的关系（大于，小于，……）和函数（加法，减数，除法，……）能被直接评估而不必用公式推理。因此，在自动推理系统中表达式评估是一个强大而高效的工具。

16.8 补充读物和讨论

有些人并不愿意寻求归结推论规则，而宁愿使用自然演绎方法[Prawitz 1965]。这些被称为是“自然的”，因为推论是在那些多少“似乎”没有转化为规范形式的句子上进行。[Bledsoe 1977]讨论了一些非归结方法。

在把归结和其他推理方法应用到数学定理证明中，Larry wos和后来的Woody Bledsoe一直是领先者。这个工作产生了新的归结策略，像支持集[Wos，Carson，& Robinson 1965]和单元首选。强大的定理证明系统（其中有些已经解决了数学中的开放问题）的例子能在[Wos & Winker 1983，Boyer & Moore 1979，Stickel 1988，McCune 1992，McCune 1994，Wos 1993]中发现。[Wos，et al 1992]是一本有关计算机定理证明和它在问题解决应用中的教科书。

定理证明也已被应用于验证和综合给定规范说明的计算机程序中。[Manna & Waldinger 1992]是一本处理自动程序综合的教程。[Manna & Waldinger 1985，Manna waldinger 1990]是处理逻辑和编程的书。[Lowry & McCartney 1991]是有关程序合成的文集。

谓词评估是一个叫语义连接的更一般过程的一个实例。在语义连接中，数据结构和程序与谓词演算语言的元素关联（即连接到）。然后，被连接的结构和过程能被用来在一种与它们的意图解释对应的方式下评估语言中的表达式[Weyhrauch 1980，Myers 1994]。

《Journal of Automated Reasoning》中包含了定理证明技术的理论和应用。

习题

16.1 判断下面的表达式对是否能够合一，并给出每个可能的合一对的最一般合一式：

1) P(x, B, B)和P（A, y, z）

2) P(g(f(v)), g(u))和P（x, x）

3) P(x, f(x))和P(y, y)

4) P(y, y, B)和P(z, x, z)

5) 2 +3 = x 和 x = 3+3

16.2 解释一下为什么P（f(x, x), A）与P(f(y), f(y, A)), A）不能合一。

16.3 把下面的表达式转换成子句形式：

1) ((∃x）[P(x)]∨(∃x)[Q(x)]⊃(∃x)[P(x)∨Q(x)]

2) (∀x)[P(x)⊃ (∀y)[(∀z)[Q(x, y)]⊃ ¬(∀z)[R(y, x)]]]

3) (∀x)[P(x)]⊃ (∃x)[(∀z)[Q(x, z)]∨(∀z)[R(x, y, z)]]

4) (∀x)[P(x)⊃Q(x, y)]⊃((∃y)[P(y)]∩(∃z)[Q(y, z)])

16.4 给定下面的一段话：

Tony，Mike，and John belong to the Alpine Club. Every member of the Alpine Club is either a skier or a mountain Climber or both. No mountain Climber likes rain，and all skiers like snow. Mike dislike whatever Tony likes and likes whatever Tony dislikes .Tony likes rain and snow.

用一种方式通过谓词演算语句表达这个信息，在这种方式中，把问题："Who is a member of the Alpine Club who is a mountain climber but not a skier?" 表达为一个谓词演算表达式。用归结反驳与回答提取回答它。

16.5 证明（∀x）λ(x)⊨ λ(τ)，τ是一个基本项。

16.6 通过一个归结反驳，证明合式公式(∃ x) P(x)逻辑上产生于合式公式 [P(A1) ∨ P(A2)]。存在一个证明(∃x) P(x)的Skolem范式P（Sk）逻辑上产生于[P(A1) P(A2)的归结反驳吗?

16.7 Sam、Clyde和Oscar是大象。关于它们，我们知道下面的事实：

1) Sam是粉红色的；

2) Clyde是灰色的且喜欢Oscar；

3) Oscar 是粉红色或者是灰色（但不是两种颜色）且喜欢Sam。

用归结反驳证明一个灰色大象喜欢一个粉红色大象，即，证明：

(∃x, y) [Gray(x) ∧ Pink(y) ∧ likes(x,y)]

16.8 对下面的每个公式，或者证明它是永真的，或者给出它的永真性的一个反例（提示：不要立即进行归结证明）。

1) {[(∃x) P(x)] ⊃ Q(A)} ⊃ {(∀x) [P(x) ⊃ Q (A)]}

2) {[(∀x) P(x)] ⊃ Q(A)} ⊃ {(∃ x) [P(x) ⊃ Q (A)]}

3) (∃x) [P(x) ⊃ Q(A)] ⊃ (∀x) [P(x) ⊃ Q (A)]

16.9 把下面的公式转化为既没有存在量词也没有Skolem函数的形式。

(∀ x, y){(∃z) [On(x, z) ∧ Above(z, y)] ⊃ Above(x, y)}

16.10 如果给定下面的条件，用一组子句上的归结反驳证明有一个绿色对象：

- 如果可推动的对象是蓝色的，那么不可推的对象是绿的。
- 所有的对象或者是蓝色的或者是绿色的，但不能是两色的。
- 如果有一个不可推动的对象，那么所有可推动的对象都是蓝色的。
- 对象*O*1是可推动的。
- 对象*O*2是不可推动的。

1) 把这些句子转换成一阶谓词演算中的表达式。

2) 把上述谓词演算表达式转换为子句形式。

3) 把上述子句形式的表达式与要证明的语句的否定子句形式组合起来，然后给出用在获得一个归结反驳中的步骤。

16.11 函数cons(x, y)表示由把元素x插在列表y的头部形成的列表。我们用Nil表示空列表；列表（2）由cons（2，nil）表示；列表（1、2）由cons(1,cons(2, Nil))表示；等等。公式Last(L, e)意指e是列表L的最后一个元素。我们有下面的公理：

- (" u) [Last(cons(u , Nil), u)]
- (" x, y, z)[Last(y, z) … Last(cons(x, y), z)]

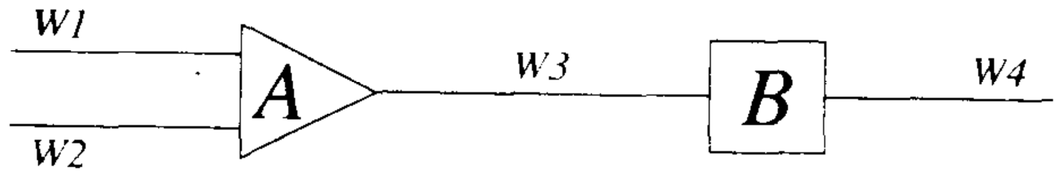
1) 从这些公理中用归结反驳证明下面的定理：

（∃v）[Last(cons(2，cons(1，Nil))，v)]

2) 用回答提取找到v，列表（2，1）的最后一个元素。

16.12 下面的逻辑电路有4条线，*W*1，*W*2，*W*3和*W*4。它有一个"与门"*A*和一个"反相器"

B。输入线*W*1和*W*2或者是“on”或者是“not”。与门*A*运行“OK”，则仅当*W*1和*W*2都是“on”时*W*3才是“on”。如果反相器*B*运行“OK”，仅当*W*3是“on”时*W*4才是“on”。



1) 使用形如OK(A)、ON(W1)的表达式描述所定义的电路功能。

2) 使用描述电路功能的公式，假定所有的元件都是运行正确的，并且*W*1和*W*2是“on”，用归结法证明*W*4不是“on”。

3) 再次使用描述电路功能的公式，给定*W*1和*W*2是“on”，且W_4也是“on”，用归结法证明或者与门或者反相器不是运行正确的。

第17章　基于知识的系统

17.1　面对现实世界

既然我们已经学习了用逻辑来进行表示和推理，并且用一些比较简单的例子演示了它的使用，那么我们要问它是否能被应用到“现实世界”问题中呢？AI研究已经发现诸如医疗诊断、税金咨询和设备设计等方面的应用典型地需要大量的随手可得的主题知识。这些应用强调知识的重要性促使我们使用基于知识的系统来描述对大量知识库进行推理的程序。已经描述的方法能“按比例增大”以使其在实际应用中表现好吗？与这个问题相关的推理方法的一些理论特性是什么？在本章中，要解决其中一些问题，并讨论在现实应用中已经证明对实际推理是有用的一些方法。

逻辑推理系统有三个主要的理论特性：合理性、完备性和易处理性。为了确信一个推导的结论是“真的”，需要合理性。为了确信推论最终将产生真的结论，需要完备性。为了确信推论是可行的，需要易处理性。关于谓词演算，归结反驳是合理的和完备的。归结反驳能被用来证明，如果一个合式公式ω被一合式公式集Δ逻辑涵蕴，那么它是完备的；否则，归结反驳过程可能永不能终止。因此，我们不能用归结作为一个完全的决策过程(*decision procedure*)。进一步讲，可以证明没有其他方法总能告诉我们什么时候一个合式公式ω不是由合式公式集合Δ逻辑地派生，什么时候是。因此，我们说谓词演算是不完全决策的。当然，不完全决策性导致了谓词演算固有的不易于处理性。

但是情形变得更糟。即使对那些归结反驳终止的问题，那个过程也是NP难题——就像对一阶谓词演算的任何合理和完备推理过程一样[Börger 1989]。因此，虽然很多推理问题能被公式化为归结反驳问题，但对非常大的问题，该方法是不易处理的，这个事实导致很多人对在大规模推理问题中使用正式的逻辑方法感到绝望（例如，见[Schwartz 1987, McDermott 1987]）。然而，由于人类自己进行复杂推理，一定有启发性的和特定的公式允许易处理的计算。

研究人员已经探索了各种方式以使推理更有效。首先，我们能在坚持推理规则的合理性前，使用那些可能偶尔（我们很少希望）会“证明”一个不正确的公式过程。第二，在坚持完备性前，使用不能保证找到正确公式证据的过程。这两种修改可以使推理更有效。第三，我们能使用一种比完全谓词演算表达力弱的语言。下面要讨论的一个表达力弱的语言的例子仅仅使用了Horn子句。使用Horn子句的推理典型地讲更有效，它们可以满足很多应用。

17.2　用Horn子句进行推理

前面，我把Horn子句定义为至多有一个正文字的子句。如果有至少一个负文字和一个正文字，Horn子句就能被写为一个蕴含，它的前项是正文字的一个合取，它的后项是单个正文字，这样一个子句称为一个规则。在子句中可以没有负文字，在这种情况下，我们把它记为一个蕴含，它的前项为空，后项是一个单一正文字，这样一个子句被叫做一个事实。或者在子句中可能没有正文字，在这种情况下，我们记作后项为空、前项是一个正文字列表的蕴含，这样的子

句称为一个目标。Horn子句形成PROLOG编程语言的基础[Colmerauer 1978,Clocksin & Mellish 1987, Sterling & shapiro 1986,Bratko 1990]。在PROLOG语言中，这些子句用作该语言的语句，写成如下格式：

• 规则：λ_h :- λ_{b1}, ..., λ_{bn}

（这是表示蕴含$\lambda_{b1} \wedge ..., \wedge \lambda_{bn} \supset \lambda_h$的一种特殊方式），每个$\lambda_i$是一个正文字。文字$\lambda_h$叫做子句的头，文字$\lambda_{b1}$,..., λ_{bn}的有序列表叫做子句的体。

• 事实：λ_h :-

• 目标：- λ_{b1}, ..., λ_{bn}

目标和规则体中的文字是有序列表，这个顺序在一个PROLOG程序的执行中起到重要的作用。

对PROLOG子句的推理由试图“证明”一个目标组成，它通过执行一个PROLOG程序来完成。这样一个证明通过对PROLOG事实、目标和规则执行类似于归结的操作而获得。每个归结在一个目标与一个事实或规则之间执行。

- 一个目标能与一个事实归结——通过将该事实和目标中的一个文字合一。我们称这个文字是被归结的。归结式是一个由初始目标中其他文字的所有置换实例组成的一个新目标（例如，那些没有被归结的）——用同初始目标相同的顺序记录。置换实例通过把合一的mgu应用到所有其他的文字完成。
- 一个目标能和一个规则归结——通过合一规则的头与目标中的一个文字。归结式是一个新目标，它通过把规则体中的所有文字的置换实例列表追加到目标中所有其他（没有归结的）文字置换实例表的前面而形成。

在PROLOG程序中，子句常常用下面的方式排序：第一个子句是目标子句，接着是事实，最后是规则。在执行程序时，解释器依次通过顺序中的目标文字，检查排序的每个子句，执行第一个可能的归结来检查与目标的归结，然后从新的目标子句开始。当应用一个归结产生的子句为空时，一个目标子句的证明成功。当一个仅有一个文字的目标子句和一个事实规则归结时，上述情况才会发生（因此，我们把事实放在首位以便只要可能就会成功）。如果解释器已经试过一个目标文字的所有归结，且在能被证明的新目标中没有任何结果时，一个目标子句宣布失败（不能被证明）。在这种情况下，解释器回溯到前面的目标子句，对它尝试其他的归结。这个过程容易被看成与对子句使用深度优先、回溯搜索过程的普通线性归结（用子句中的文字和子句的排序控制的归结顺序）等价。如果能很好地选择归结的子句顺序和要归结的子句中的文字，使用这些顺序的深度优先搜索是能发现一个证明的有效方式。PROLOG程序员用有序地表达子句和有序地表达每个子句中的文字来表示深度优先搜索的归结顺序。

用一些简单的例子演示PROLOG推理。第一个不涉及变量，给出了一个过程的高级视图。回想前面用命题演算演示推理的举木块的例子。假如木块是可以举起的，电池也已被充电，那么一个证明手臂移动的PROLOG程序如下：

```
1) :- MOVES
2) BAT_OK :-
3) LIFTABLE :-
4) MOVES :- BAT_OK, LIFTABLE
```

第一个归结在目标子句和句子4之间进行，产生新目标子句::-BAK_OK, LIFTABLE。新

目标子句与语句2归结（在新目标子句中的第一个文字上归结），产生新目标子句：-LIFTABLE，然后再与语句3归结，产生空目标子句，成功终止程序。在这种情况下，不需要回溯。

上面的证明可用图17-1中的树结构表示，在方框中的节点集对应程序语句（目标、规则或事实）。节点用程序语句中的文字标识。有两种弧：匹配弧（用双线表示）和规则弧（用单线表示）。规则弧把一个规则的体节点同该规则的头节点相连。匹配弧把一个规则（或一个目标节点）的一个体节点连向一个等价的另一个规则的标识头节点（或到一个事实）。在这样一棵树中的一个目标节点或一个体节点被证明——如果它被一个匹配弧连向一个事实节点或者连向一个其体节点均已证明的头节点。给规则弧加了一个圆形符号（∪）以强调为了证明头节点，必须证明所有的体节点。图17-1中的树是“与/或（AND/OR）”证明树的一个实例。在这样一棵树中，体节点被称为AND节点，因为它们必须全被证明（如果有另一个可能的归结，通过另一个匹配弧对一个证明树的搜索会产生另外的节点，称为OR节点。随后我们将看到带有嵌入证明树的“与/或”树的一个例子）。

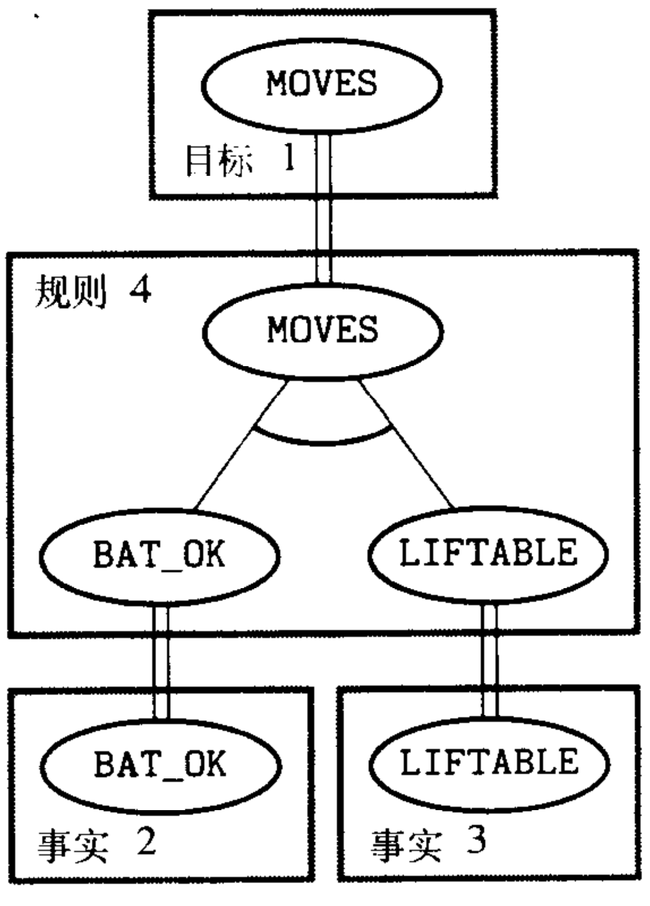


图17-1　“与/或”证明树

作为使用变量的一个例子[⊖]，回到积木问题。假如我们想使用一个积木在另一个积木上（above）的概念。用谓词演算，我们能用下面的两个公式用项on递归地定义above：

$(\forall x, y, z)[On(x, y) \supset Above(x, y)]$

$(\forall x, y)\{(\exists z)[On(x, z) \wedge Above(z, y)] \supset Above(x, y)\}$

用这些定义，下面的PROLOG程序可用来证明当*A*在*B*上，*B*在*C*上时，A在C上：

```
1) :- Above(A, C)
2) On(A, B) :-
3) On(B, C) :-
4) Above(x, y) :- On(x, y)
5) Above(x, y) :- On(x, z), Above(z, y)
```

第一个可能的目标归结（用规则4）产生新目标 :- On(A, C)。目标失败，因此我们回溯到初始目标试下一个归结（用规则5）产生一个新目标 :- On(A, z), Above(z, C)。归结这个目标中的第一个文字（用事实2）产生新目标 :- Above(B, C)。这个目标与规则4归结产生 :- On(B, C)，它和事实3归结产生空目标，过程终止。目标 :- Above(B, C) 的产生可以被认为是对同一个程序的递归调用。

被PROLOG解释器执行的“与/或”演示搜索树显示在图17-2中。证明树中的节点用粗体椭圆表示。被中止的搜索部分的节点用阴影节点表示。一些匹配弧由置换标识（置换是对体节点和头节点合一时得到的），方框结构代表规则的置换实例。为了使树代表一个合法的证明，置

⊖ PROLOG 印刷约定与我们相反；在PROLOG中，变量是大写的，常量是小写的字母和数字。为了方便与FOPC 比较，本书坚持我的常用约定。

换必须是一致的。这里的一致性要求在整个证明树中用相同的项被置换为变量z。当然，在PROLOG中，一致性通过在建立一个新目标时用项度量的立即置换得到。注意在证明树的规则4中实例化变量部分的使用。

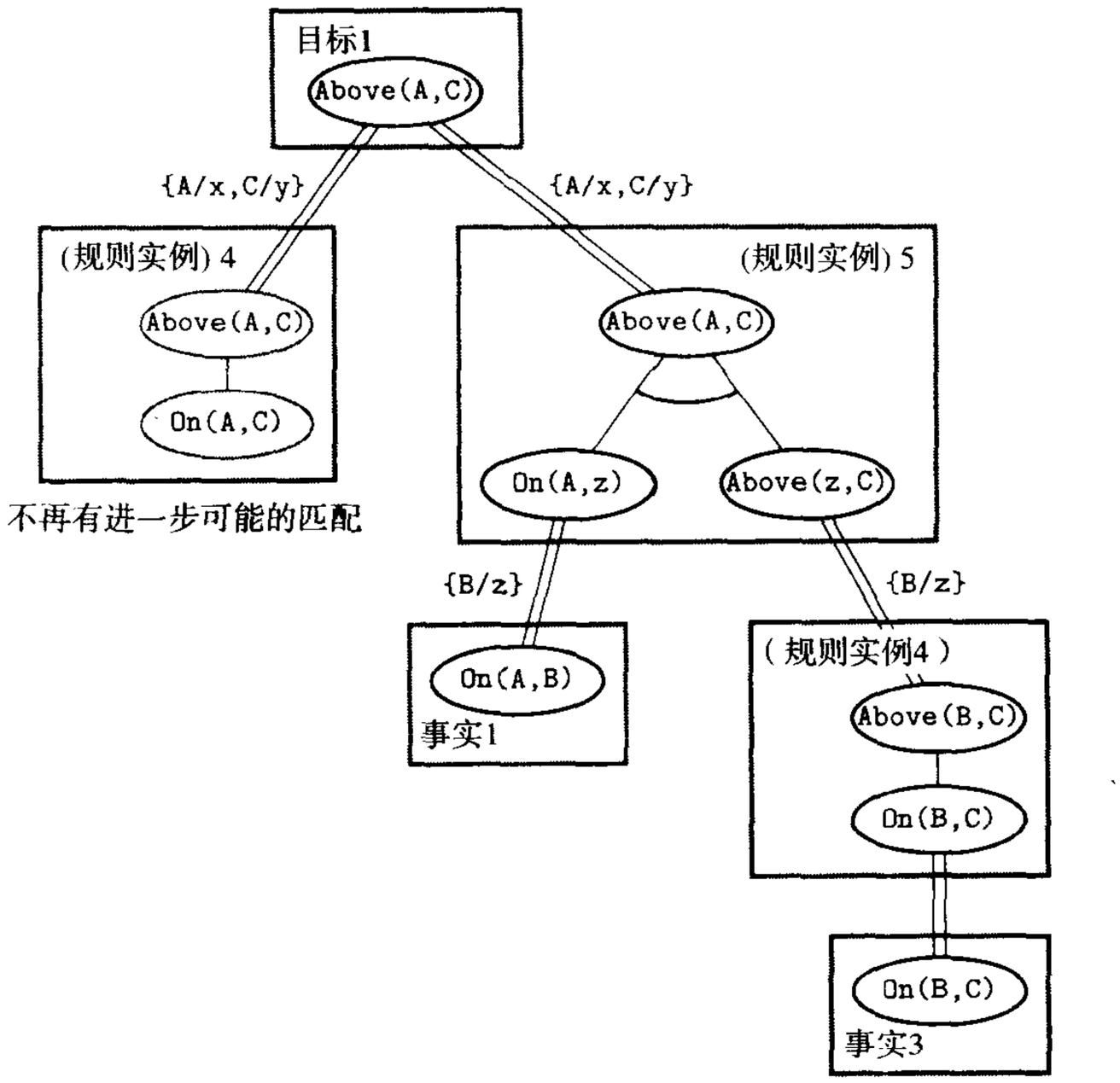


图17-2 一个“与/或”树

PROLOG语言是一门通用的编程语言，已经证明它在各种AI应用中都是有用的，尤其在自然语言处理中[Pereira & Shieber 1987]。由[Shoham 1994]写的一本课本讲解了如何用PROLOG来实现几项AI技术。和PROLOG有关的逻辑语言（如DATALOG）被用来定义“内涵关系”，用来扩展在一个关系数据库中的显式出现（外延关系）。要全面了解逻辑方法在数据库中的使用，可参见[Ullman 1988, Ullman 1989]。[Minker 1997]是一篇值得阅读的综述性文章。

这些基本的PROLOG程序说明了如何将Horn子句规则用于推理问题。在PROLOG语言中，推理“向后”进行，从要建立的目标开始，顺着规则链，最后锚定在事实上。使用目标规则和事实执行这类向后推理的AI系统被称为执行反向链。也可用正向链，在这种情况下，推理向前进行，从事实开始，顺着规则链，最后产生目标。正向链系统常常用标准的蕴含形式表示规则，它的前项由正文字构成，后项由一个正文字构成。就像PROLOG是一个基于反向链的系统一样，有一种计算机语言OPS5[Brownston, et al. 1985]是基于正向链的。已经用OPS5写出了几个实际有效的推理系统。

在正向链系统中，如果一个规则的前项文字的每一个都能与一个相应的事实（用一致的置换）合一，那么它是可应用的。由于事实是基本原子，变量仅出现在可被合一的一对表达式中。这种受限的合一形式常叫做模式匹配。一个规则应用就是将规则后项的相应实例加到事实列表中（规则的前项文字被一致地匹配到事实）。当可应用多于一个规则时，某种外部冲突归结被用来决定将应用哪个规则。当有大量的规则和事实时，必须被尝试的合一数量成为障碍。

OPS5使用一个叫RETE 的算法过程，它把规则编译成一个网络，用来通过规则的一致匹配有效地指导事实[Brownston, et al.1985,Forgy 1982]。

正向链——事实触发规则的过程，它加入新事实等等——非常类似第5章讨论的黑板系统的工作过程。在一个特定的形式中，它也可以作为人类认知过程的一个模型。这个事实相应于所谓的工作或短期记忆。规则相应于长期记忆。在AI中，基于这个认知模型的最完备的开发系统是那些基于SOAR形式方法的系统[Laird,Newell, & Rosenbloom 1987,Rosenbloom,Laird, & Newell 1993]。

17.3　动态知识库的维持

正向链生成规则必需的明显事实，但这些事实直到被推断出才能显式地应用到知识库中。一旦可用，这些被推断的事实就能用来演绎另外的事实，或者可以引发agent动作。因此，直接向前推理被看作是对事实知识库做改变。这里详细讨论这个观点。

考虑一个命题演算原子知识库**KB**。用原子“前提（*premiss*）”代替事实，因为我们想能够推理各种“假设分析”的可能性。为具体起见，考虑一个例子，它的前提是P和Q，规则是P∧Q ⊃ R和P∨R ⊃ S。注意，像前面一样，规则是带有单文字后项的蕴含，但现在它们没被限制为Horn子句。

一个使用规则的正向推理链允许我们把R和S作为新原子加到**KB**中。考虑这个知识库的方式之一是它像一个只有一行和几列的一维电子表格，我们可以称之为一个“扩展行”。对P、Q、R和S都有一列，在每列有一个单元放这些原子的值。在P和 Q的每个单元中，我们能输入1（真）或0（假），而在R的值单元中，我们可以输入公式P∧Q，在S的值单元中输入公式P∨R。这些公式是每个规则的前项。一些扩展行实例显示在图17-3中。上面的实例表示一个公式，底部显示了两个例子的单元推断值。像在电子表格中一样，一个公式的任意单元立即得到它的值，这个值从元件值计算（用正向链）而来。如果一个公式的任何元件值碰巧改变了，公式的值也自动改变。

P	Q	R	S
1	1	= P ∧ Q	= P ∨ R

P	Q	R	S
1	1	1	1

P	Q	R	S
1	0	0	1

图17-3　一个Spreadline推理系统

我们能延伸扩展行以包含任何数量的前提和规则，这样一个结构可作为一种动态知识库。推理过程被机制化，与显示的单元值无关。用这种方式实现的知识库可用于各种目的。就像电

子表格一样，它们能用来回答“假设分析”问题。某个人（或一个agent）可能想知道，如果前提在一定的方式下改变，各种原子值将怎么改变。在AI中，这些扩展行叫做推理维持（*reason maintenance*）系统。它们和后面要讨论的结构是所谓的真值维持系统（TMS）的实例，在规则单元值中输入的公式称为“理由（*justification*）”。它们常被表示为原子合取的析取——析取范式。各种有效的过程（像RETE算法[Forgy 1982]）已经被设计来进行正向链后台计算，以计算所有单元的值。

一个有趣的扩展（超过传统的电子表格）使我们能够省略某些前提单元的值。不再把每个标识为1或0，而可能就是不给它们值。如果缺少前提单元的值，那么某个规则单元的值也可如此。当一个传统的电子表格面对一个带有省略值的元件公式时，它可以假定那个值是0，或者它可以警告用户那个省略值。但是当一个动态知识库不能计算一个公式的值时（因为元件有省略值），它把公式的值也报告为省略的——留下该单元为空。在图17-4中，展示了一个例子，对R和S的公式与图17-3中的相同。注意，在一种情况下即使缺少S的一个元件，也能计算它的值。

P	Q	R	S
		= P ∧ Q	= P ∨ R

KB 实例

P	Q	R	S
1			1

P	Q	R	S
	1		

图17-4 一个带有缺少值的知识库

在TMS的词汇中，一个其值既不是1也不是0的原子被称为OUT。用OUT作为OUT原子的值，而不把它们的值单元留为空。当一个单元的值是1或0时，它是IN。特别注意，把单元P的值从1改为OUT与P是False是截然不同的。这样一个改变仅仅是说知识库不再知道P是否为真（或为假）。

允许单元值是OUT准许对规则的类型一般化。我们可以有规则，它的前项包括其值必须是OUT而不是0或1的原子。用一个例子来说明，如果R为真，S是OUT，则我们能推论U。可以将它表示为一个类似蕴含的形式：

$$R \wedge \bar{S} \supset U$$

如果一个原子必须是OUT，用($\bar{}$)号作为它的上标（在TMS蕴含中的后项从来没有上标符号。即，当前项被满足时，后项是1，因此是IN）。

TMS应该被认为并非通用目的推理系统（像我们在第13章到第16章学习的一样），而是知识库维持系统，当原子的真假改变时，它执行自动更新。IN和OUT的使用允许一种基本的“非单调”或可废止的推理，它不被普通的推理系统支持。普通的逻辑推理是单调的，以致当一个新公式被加到知识库上时，在公式被加入前允许的所有推理仍然是有效的，加入新公式不会减

少可以证明的定理集合。但是如果对一个依赖某个别的为OUT的原子的原子ϕ做推理，那么当把那个别的原子加到IN后面时，我们将必须取消ϕ。当然，它可能导致进一步的取消（或演绎）。

然而规则的循环带来了一些困难，一是相互证明（*mutual justification*）。考虑合式公式P、P ⊃ Q、Q ⊃ R和R ⊃ Q，如果P有真值，那么Q和R也有真值。但是如果我们不知道P的值，就不能对Q和R做出任何结论。但是在动态知识库计算中，Q和R能证明它们自己的值。考虑图17-5中的计算阶段。开始时，假设P为1，产生图中的开始**KB**。如果P变成OUT，扩展行计算能如图所示进行——导致R和Q证明自己而不用P的支持。另外，当一个公式中有OUT时，原子的值或者是不确定的或者过于确定。在第一种情况下，可能有不只一种方式使原子是IN或OUT；在第二种情况下，它们可能没有一致的值（更详细的内容参见[Shoham 1994, 第5章]。

一个**TMS**系统的一些应用要求值为1和IN的原子的某个子集的所有成员是不一致的。这些子集被叫做nogoods。一个一致的维持系统（CMS）通过改变一些IN为OUT，强迫这些约束。由于一致性能用几种不同的方式维持，故CMS是不确定的。

公式

P	Q	R
	= P ∨ R	= Q

开始知识库

P	Q	R
1	1	1

当P变为 OUT 时的计算阶段

阶段1：用公式及 R 的先前的值，计算新的Q值

P	Q	R
OUT	1	1

阶段2：给定 P 和Q 的先前的值，计算新的R值

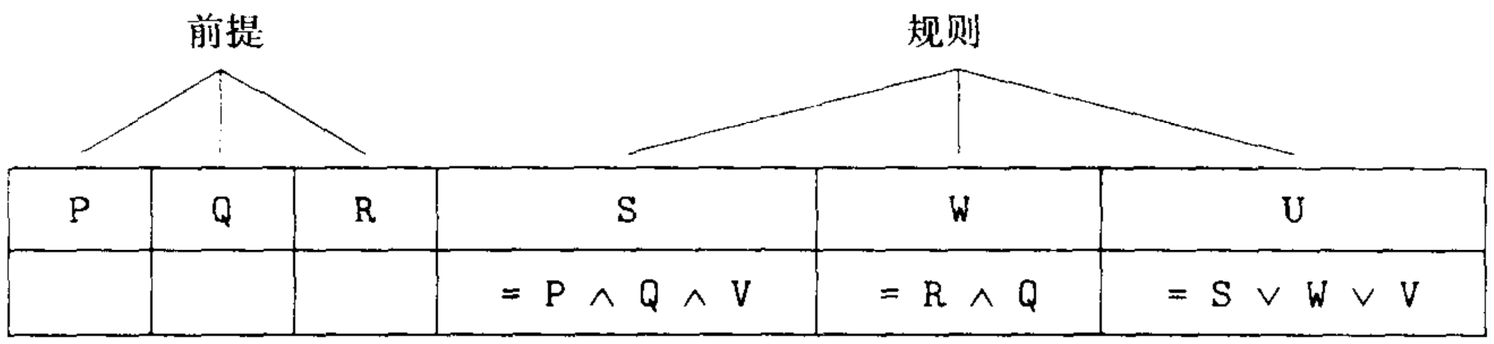
P	Q	R
OUT	1	1

图17-5　相互证明

继续使用扩展行来表示关于动态知识库中的讨论，其中的扩展行被向后而不是向前使用，即，不是从某些前提开始，用公式更新所有单元的值，而是以一个特殊单元的公式开始，寻找哪些前提值（它和某些其他公式一起被称为后台理论（*background theory*））将使那个单元的公式为真。在这些应用中，将单元的值作为公式——按照前提原子用DNF形式表达。这些公式被称为单元的标记。具有这些连向原子的标记的知识库被称为基于假设的真值维持系统（ATMS）。每个原子的标记由各种前提值集（假设的）构成，这些前提值和后台理论一起使原子有真值。

例如，考虑图17-6顶部的扩展行。我们重新按照前提表达公式，并得到图底部的扩展行以获得ATM标记。通常，集合符号被用作标记。一个原子合取用一个集合表示，这些合取的析取用多个集合的一个列表表示。例如，在图17-6中，为了使U为真，或者P必须为真（使V为真）或者R和Q必须为真（使W为真）。在U的一个最小标记中，我们不必包括集合{P，Q}（使S为真），因为P在标记中的出现包容了{P，Q}。

ATM能用于各种目的。一个是执行诊断，例如图17-6中的公式可以对某个电子设备的功能建模。如果观察到U表示的部分有问题（即U有假值），可以用ATM确定P和P ∧ Q必定都为假，因为否则U将为真。因此我们能确定由R和Q代表的功能部分之一有问题，由P表示的部分也必然有问题。



后台理论：P ⊃ V

ATMS 标记

P	Q	R	S	W	U
P	Q	R	{P,Q}	{R,Q}	{P} {R,Q}

图17-6 一个TMS到一个ATMS的转换

ATM有时有其他的特征和限制。有些前提总是产生真值，因此去除标记（即，这些前提是后台理论的一部分）。标记也可能被一个假设（前提的一些子集是nogoods）所修改，因此不能总有真值。有关TMS，ATMS和它们的应用的详细情况，参见[de Kleer 1986a,de Kleer 1986b, de Kleer 1986c, Forbus & de Kleer 1993]。

17.4 基于规则的专家系统

使用事实和规则的AI推理技术的最成功应用之一是建立专家系统，专家系统包含了人类努力探索的一个专门领域的知识，如医疗、工程和商业。在[Feigenbaum,McCorduck & Nii 1988]中给出了在用的很多专家系统的调查（circa 1988）。[Dym & Levitt 1991]写了一本关于专家系统工程的教课书。严格地讲，任何具有专家功能的程序都被称为是一个专家系统。[Feigenbaum,McCorduck & Nii 1988]中有如下的定义：

在解决问题中，通过运用知识体达到专家级水平的AI程序叫做知识库系统或专家系统。经常，术语“专家系统”被约定用于知识库中包含人类专家使用的知识而不是从课本或非专家那里收集来的知识的程序。两个术语——专家系统和知识库系统更经常被同义使用。

在本书中，主要讨论基于规则的专家系统。

一个在使用环境中的专家系统的基本结构如图17-7所示（[摘自[Feigenbaum,McCorduck, & Nii 1988]）。系统的主要部分是知识库和推理引擎。根据到目前为止讨论的推理系统，知识库由谓词演算事实和有关讨论主题的规则构成（虽然它也可能包含用谓词演算的语法变种表示的知识。下一章将看到这样一个变种的例子）。推理引擎由所有操纵知识库来演绎用户要求的信息的过程构成——如归结、前向链或反向链（在某些实例中，知识库和推理机制可能被紧紧地连结在一起，就像它们在TMS中一样）。用户接口可能包括某种自然语言处理系统，它允许用户用一个有限的自然语言形式与系统交互（见第24章）。也可使用带有菜单的图形接口界面。解释子系统分析被系统执行的推理结构，并把它解释给用户（后面我们将看这种解释的一个简单例子）。

在实际应用中，这四个部分构成了一个系统。在一个专家系统结构中，一个“知识工程师”（经常是一个训练过的AI计算机科学家）与应用领域的一个专家（或几个专家）共同工作以便

把专家的相关知识表示成一种形式，以使它能被输入到知识库。这个过程经常由一个知识采集子系统协助。和其他情况一样，这个子系统检查正在增长的知识库的可能不一致和不完备信息，然后将它们表示给专家以做出决定。

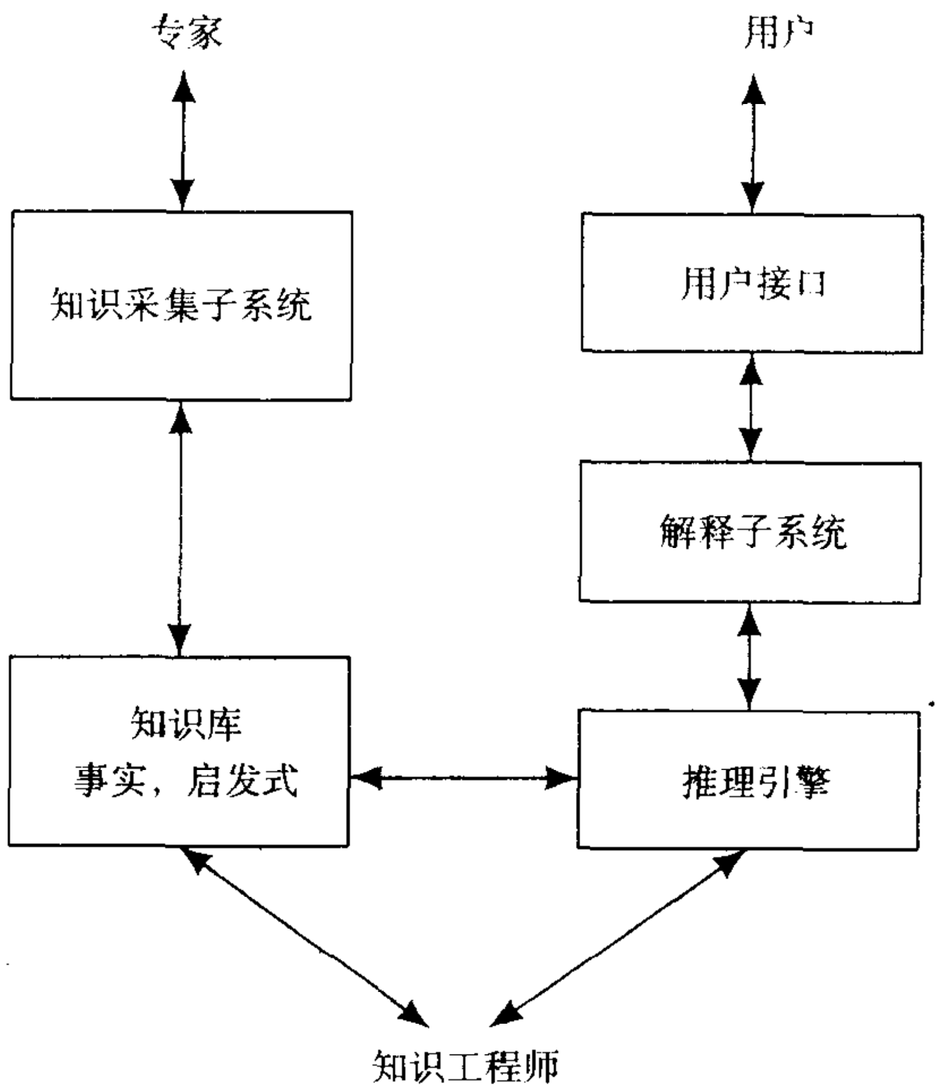


图17-7　一个专家系统的基本结构

建立系统的过程常常通过很多循环迭代进行，在循环的每一步，知识工程师和专家测试一个原型系统，看它是否能做出与专家对用户提出的典型问题相同的推理。如果系统与专家的反应不同，用解释子系统来帮助开发小组决定分歧由哪些信息和推理引起。专家可能需要更详细地说明某个信息或者提供附加的信息来包含当前的情况。这个过程继续下去直到开发小组对系统操作满意为止。建立专家系统的过程要求经验和判断。关于这个主题已经有了各种书籍；[Stefik 1995]是一本优秀的书籍。这里仅给出一些基本思想。

基于规则的专家系统常常建立在关于命题逻辑Horn子句的推理之上（也许还有某种其他的机制解决不确定性）。知识库由从专家收集来的规则构成。这些系统已被应用到很多环境中。例如，想像银行的一个贷款员使用这样一个系统帮助决定是否把一个私人贷款授于一个人。一个实际的贷款表决系统可能要考虑很多因素，远远超过我们在一个演示例子中所能想到的。但是能通过描述一个极大地简化了的版本来给出这样一个系统如何工作的大致思想。假如下面的原子将指称相关的命题：

- OK（贷款应该被批准）
- COLLAT（贷款抵押被满足）
- PYMT（申请人能做出贷款偿还）
- REP（申请人有良好的金融声誉）
- APP（抵押评估充分大于贷款额）
- RATING（申请人有良好的信贷等级）
- INC（申请人的收入超过他/她的消费）

• BAL（申请人有良好的资产负债表）

那么，下面的规则可被用来做决定：

1) COLLAT ∧ PYMT ∧ REP ⊃ OK

2) APP ⊃ COLLAT

3) RATING ⊃ REP

4) INC ⊃ PYMT

5) BAL ∧ REP ⊃ OK

假定贷款员想确定一个申请人的OK值是否为真。为了证明OK，推理引擎用一个“与/或”证明树使用反向或正向链（或双向）进行搜索。“与/或”证明树（如果存在一个）将用OK节点做为根节点，事实（或者由用户决定是真的，或者列在一些事实数据库中）做为叶节点。根结点和叶结点通过规则连接（常常通过中间节点）。

用前述反向链中的规则，一个OK证明搜索树的上面部分如图17-8所示。用户建立OK的目标或者通过证明BAL和REP，或者证明COLLAT、PYMT和REP中的每一个来实现。证明OK的两种可选方式被两个OK节点表示在根节点下面。回忆在一个“与/或”树中被称为“或”的节点。在一个“与/或”树中，一个有“或”后继节点的节点能通过证明这些后继中的任何一个来证明。应用图示的其他规则，以证明其他的节点集。

为了本例的目的，我们假定通过询问数据库或直接让推理系统询问贷款用户，能建立BAL、RATING、APP和INC的真值（或假值）。另外，OK、PYMT、COLLAT和REP指称对推理有用的概念，但是关于这些用户和数据库却没有明确的信息（如果用户已经有这些概念的知识，他或她将不需要专家系统！）。在全面的专家系统中，可以尝试几个可选的规则来连接中间节点和叶节点。

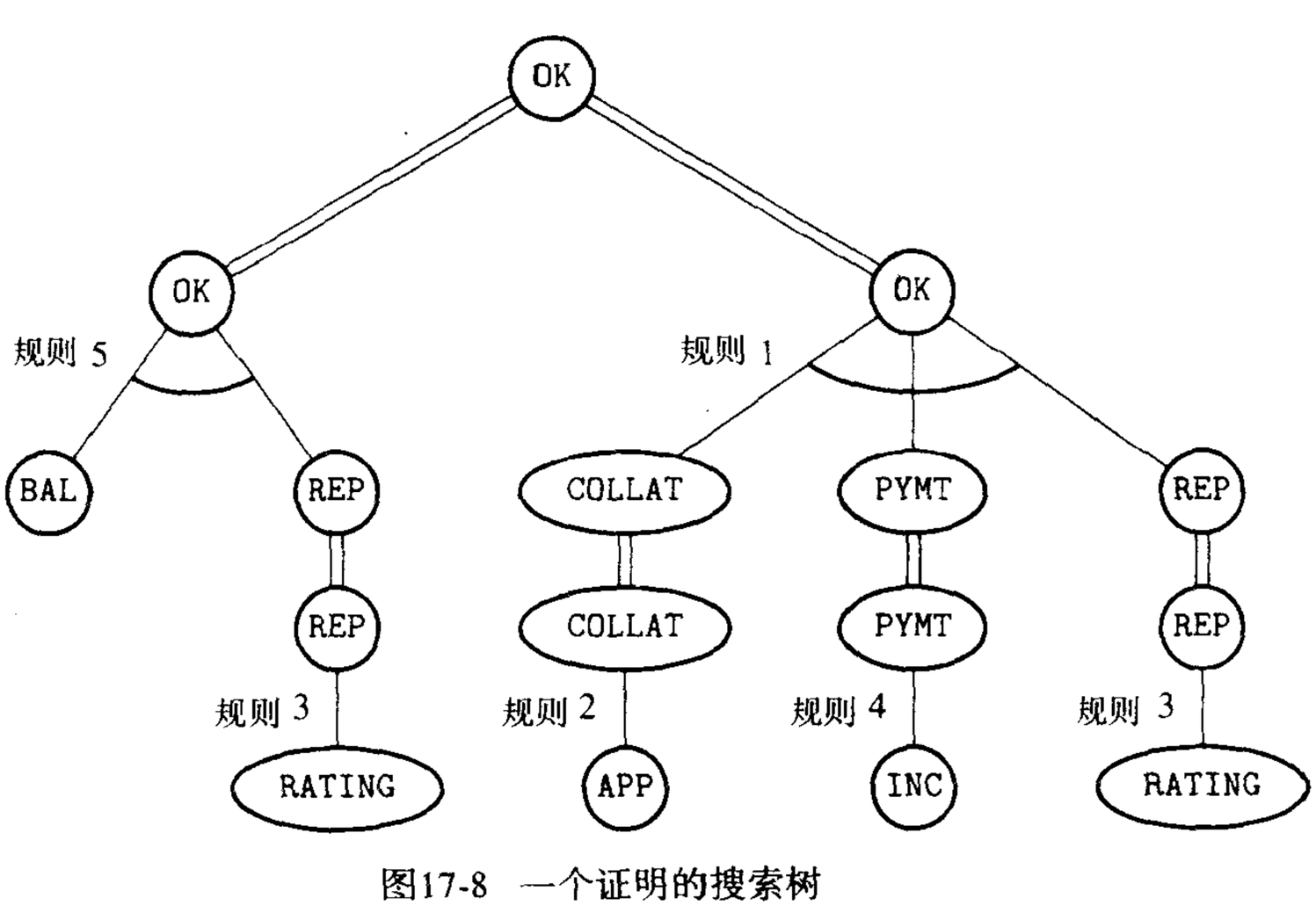


图17-8 一个证明的搜索树

如果按深度优先方式进行证明，对用户或数据库系统的查询能与搜索交替进行。例如，如果在搜索过程的早期，节点BAL被发现不为真，那么搜索将用规则1继续。顺着规则1、2、3和4向前进行，搜索过程最终产生请求查明APP、INC和RATING。如果所有这些节点的真值能被建立（从数据库或用户），那么我们证明了OK。当然，在实际应用中，证明树将更深更宽。

咨询系统通过问一些用户或数据库可能知道的命题的真值来设法回答用户的询问。通过规

则的反向链得到要回答的问题。如果一个用户想要“无偿”的信息，自底而上，通过规则的正向链能用来试图连接已经建立的证明树。

可以用同样的例子来描述系统的推理如何解释给用户。给出一个结论的解释能力对接受专家系统建议是非常重要的。证明树用来指导解释生成过程。再次引用图17-8，在下面的对话中可以直接看到系统的反应是如何产生的（被推理系统使用的每个原子有一些与它相关的适当句子）。

用户：为什么你相信贷款应该被批准？

系统：因为**贷款抵押被满足**，并且**申请人能做出贷款偿还，申请人有好的金融信誉**。

用户：为什么你相信**申请人有良好的金融信誉**？

系统：因为信贷等级服务说那个**申请人有良好的信贷等级**。

用户：为什么你问我是否**申请人的收入超过他/她的消费**？

系统：因为我正试图建立是否**申请人能够做出贷款偿还**。

用户：为什么你要建立是否**申请人能够做出贷款偿还**？

系统：因为这是建立是否**贷款应该被批准**的标准之一。

用户：为什么你要建立**贷款应该被批准**？

系统：因为你问我。

在很多应用中，系统仅对不确定规则有访问取权，用户也不能确定地回答问题。例如，适于在计算机可理解的形式下编码的以下规则，被用在MYCIN系统中[Shortliffe 1976]，该系统是诊断细菌感染的：

规则300

如果：

1) The infection which requires therapy is meningitis, and

2) The patient does have evidence of serious skin or soft tissue infection and

3) Organisms were not seen on the stain of the culture ,and

4) The type of the infection is bacterial

那么：

There is evidence that the organism (other than those seen on cultures or smears) which might be causing the infection is staphylococcus-coag-pos (.75); streptococcus-group-a (.5).

同样，下面的规则被用在PROSPECTOR系统[Duda Gaschnig, & Hart 1979 ,Campbell,et al. 1982]，它是对矿石沉淀进行推理的：

If there is a pre-intrusive, thorough-going fault system ,then there is (5, 0.7) a regional environment favorable for a porphyry copper deposit.

数字（在MYCIN中的0.75和0.5，在PROSPECTOR中的5和0.7）是表达一个规则的可信度和强度的一种方式。它们被用来计算结论的可信度。被MYCIN与PROSPECTOR使用的相似技术已被应用，并产生了良好的效果，第19章将介绍更复杂更深奥的技术以解决不确定性问题。

17.5　规则学习

既然已经知道基于规则的系统的重要性和从专家那里提取好的规则的艰难，那么很自然地，我们会问专家系统规则能否被自动学习。有两种类型的学习方法，归纳法和演绎法。两种类型

都能用于学习规则。例如，神经网络学习就是归纳法，因为学习的函数是对潜在的、未知的函数的假设。在成功的学习中，一般假设会给出大部分输入的正确输出，但是它们也可能出错。归纳规则学习方法建立一个领域的新规则——不能从任何以前的规则推导出。将介绍用于在命题和谓词演算中介绍归纳规则学习的方法。

演绎规则学习通过从以前知道的领域规则和事实推理附加的规则来扩展一个系统的性能效率。不用附加规则，系统也能推出使用附加规则推出的结论。但是使用附加规则，系统可以更有效地执行。对演绎附加规则，将解释一个叫做基于解释的广义化（EBG）技术。

17.5.1 学习命题演算规则

已经提出了几种归纳规则学习的方法，这里将描述其中一种[㊀]。首先描述Horn子句命题逻辑的一般思想，然后给出一个相似的、可以用来学习一阶逻辑Horn子句规则的技术。

再次使用批准银行贷款的简单例子。不再提供该问题的规则，而是假设给出了一个训练集合，它由大量的个人属性值构成。为了说明，考虑表17-1中给出的数据（用1代表T，0代表F）[㊁]。例如，这个表可以从贷款应用记录和贷款员所做结论来收集。训练集中OK值为1的成员称为正例，OK值为0的叫反例。从训练集中，我们希望归纳出如下规则：

$$\alpha_1 \wedge \alpha_2 \wedge \ldots \wedge \alpha_n \supset \mathrm{OK}$$

其中α_i是集合{APP，RATING，INC，BAL}中的命题原子。如果一规则的前项在训练集中有实例T值，我们说那个规则覆盖（*cover*）那个实例。我们能通过给规则的前项加一个原子来改变任何现存的规则，使之覆盖更少的实例。此改变使那个规则更特殊。两个规则比单个规则能覆盖更多的实例。增加一个规则使得使用这些规则的系统更为通用。我们寻找一个规则集合，它仅覆盖训练集中的所有正实例。

表17-1 银行数据

例子	APP	RATING	INC	BAL	OK
1	1	0	0	1	0
2	0	0	1	0	0
3	1	1	0	1	1
4	0	1	1	1	1
5	0	1	1	0	0
6	1	1	1	0	1
7	1	1	1	1	1
8	1	0	1	0	0
9	1	1	0	0	0

对一个规则集合的搜索在计算上是困难的。这里描述了一个“贪婪”方法，亦称为分治法（*separate and conquer*）[㊂]。首先设法找到一个规则，它仅仅覆盖正例——尽管它不覆盖所有的正例。我们从一个覆盖所有实例（正例和反例）的规则开始来对这样一个规则进行搜索，通过加原子到它的前项，逐渐使它更特殊。由于一个单一规则不可能覆盖所有的正例，我们逐渐

㊀ 规则也能从决策树(见Quinlan 1993，第5章) 和神经网络(见[Towell & Shavlik 1992，Towell，Shavlik & Noordweier 1990])中提取。

㊁ 当然，在实际应用中，我们需要比这里给出的多得多的数据。

㊂ 该方法由[Michalski 1969]提出，由一系列基于Michalski 的AQ 算法检查过。

增加规则（使它们像我们需要的一样特殊）直到整个规则集仅覆盖所有的正例。

下面说明如何使用该方法。首先从覆盖所有实例的临时规则

T ⊃ OK

开始。现在，我们必须加一个原子使它包含较少的反例——朝着仅覆盖正例的方向前进。我们应该加哪一个原子（从集合{APP，RATING，INC，BAL}中）呢？有几个标准可用来做这个选择。为简单起见，决定基于一个容易计算的比率：

$$r_\alpha = n_\alpha^+ / n_\alpha$$

其中，n_α是将α原子加到规则的前项后，被（新的）前项覆盖的（正的和反的）实例的总量，n_α^+是将α原子加到规则的前项后，被（新的）前项覆盖的正例的总量。

我们选取产生最大r_α值的α值。在例子中，该值是：

$r_{\text{APP}} = 3/6 = 0.5$

$r_{\text{RATING}} = 4/6 = 0.667$

$r_{\text{INC}} = 3/6 = 0.5$

$r_{\text{BAL}} = 3/4 = 0.75$

因此，我们选BAL，产生临时规则：

BAL ⊃ OK

这个规则覆盖了正例3、4和7，但也覆盖反例1，因此必须把它进一步特殊化。用同样的技术选择另一个原子。r_α的计算必须考虑到我们已经决定了前项中的第一个部件是BAL：

$r_{\text{APP}} = 2/3 = 0.667$

$r_{\text{RATING}} = 3/3 = 1.0$

$r_{\text{INC}} = 2/2 = 1.0$

其中，在RATING和INC之间有一个平局。我们可能选择RATING，因为r_{RATING}基于一个更大的取样（你应该研究一下选择INC的结果）。

规则BAL ∧ RATING ⊃ OK仅覆盖正例，故不必再加原子到这个规则的前项。但是这个规则没有覆盖所有的正例，尤其他不覆盖正例6。因此，我们必须增加另一个规则。

为了学习下一个规则，首先从表中删去被第一个规则覆盖的所有正例，获得表17-2中所示的数据。我们用规则T ⊃ OK，从这个已精减过的表开始，重复上述的所有过程。这个规则覆盖一些反例1、2、5、8和9。为了选择一个原子加到前项，我们计算

$r_{\text{APP}} = 1/4 = 0.25$

$r_{\text{RATING}} = 0/3 = 0.0$

$r_{\text{INC}} = 1/4 = 0.25$

$r_{\text{BAL}} = 0/1 = 0.0$

表17-2　被减的数据

个例	APP	RATTNG	INC	BAL	OK
1	1	0	0	1	0
2	0	0	1	0	0
5	0	1	1	0	0
6	1	1	1	0	1
8	1	0	1	0	0
9	1	1	0	0	0

又有一个平局。任意选择APP给出规则APP ⊃ OK。这个规则覆盖反例1、8和9，因此我们必须加其他的原子到前项。r的值为：

$r_{\text{RATING}} = 1/2 = 0.5$

$r_{\text{INC}} = 1/2 = 0.5$

$r_{\text{BAL}} = 0/1 = 0.0$

选择RATING给出规则APP∧RATING ⊃ OK，这个规则覆盖反例9。使这个规则更特殊（用通常的方式），最终产生规则APP∧RATING∧INC ⊃ OK。这两个规则，BAL∧RATING ⊃ OK和APP∧RATING∧INC ⊃ OK，覆盖了所有正例，且仅覆盖正例。因此我们完成了任务。

由于这个发现规则的过程使用了贪婪搜索，学习的规则有时能被简化就不惊奇了。对每个规则，通过测试来看在不改变其余规则在训练集上做出的决定的情况下，规则是否能被遗弃。如果没有任何影响（或者数据中有噪声时，对精度有很小的影响），那个规则可以被删除。同样，对规则中的每个原子，我们测试那个原子能否被移走而只有最小的影响。确实，如果数据有噪声，我们宁愿修改学到的规则仅覆盖所有正例的标准。我们可能允许每个规则覆盖“主要的”正例——允许（当必须考虑噪声数据时）每个规则包含少量的反例。同样，学到的规则集可以允许未能覆盖少量的正例。这些“修剪”操作和噪声容限修改有助于把过高的冒险最小化。

用伪代码简洁地描述这个规则学习过程，称之为通用分治算法（GSCA）。在算法中：

Ξ 是初始的二值化特征训练实例集，每个特征用一个原子γ的值标识；

π 是将要学习的规则集；

ρ 是规则之一；γ是它的后项，（原子的合取）Γ是它的前项；

α 是从 Ξ 中的特征之一提取的原子。

通用分治算法GSCA：

1) 初始化 $\Xi_{cur} \leftarrow \Xi$。

2) 初始化π← 空的规则集。

3) repeat 外部循环加入规则直到π覆盖所有（或大部分）正例。

4) 初始化 Γ← T。

5) 初始化ρ←Γ ⊃ γ。

6) repeat 内部循环把原子加到Γ中，直到ρ仅覆盖（或主要）正例。

7) 选择一个原子α加到Γ，这是一个用于回溯的非确定性选择点。

8) Γ←Γ∧α。

9) until ρ仅包含（或主要）Ξ_{cur}中的正例。

10) π←π, ρ。把规则ρ加到规则集。

11) $\Xi_{cur} \leftarrow \Xi_{cur}$——($\Xi_{cur}$中被π包含的正例)。

12) until π覆盖所有（或大部分）Ξ中的正例。

在银行贷款例子中，学到的规则与最初“专家”为这个问题给出的规则是一致的。然而，用这样一个小的训练集学到的特殊规则可能与一个专家对那个问题的直觉判断不一致（例如，如果平局中我们选用不同的项，就会学到不同的规则）。对学到的规则和专家给出的规则进行比较，揭示了学到的规则不会提到专家没有估量或问到的任何谓词（COLLAT、PYMT和REP），它们没有被数据中的特征表示。中间谓词通过封装更原始谓词的一般出现组简化了知识表示和推理。学习这些中间谓词是一个重要的研究课题。关于“谓词发明”已经做了一些初步的工作

（例如，见[Muggleton & Buntine 1998]。即使没有产生涉及中间谓词的规则，规则学习常常也能被用来加速专家系统的建立过程。

17.5.2　学习一阶逻辑规则

上面描述的规则学习技术产生命题逻辑中的规则。虽然很多专家系统建立在命题逻辑的基础之上，但是更全面的专家系统是通过使用带有通用量化变量的规则而得到的。归纳逻辑编程（ILP）的子领域集中于FOPC（因此与PROLOG程序一样）中Horn子句的归纳学习方法。已经开发了学习Horn子句的几个方法——有些对能产生的Horn子句的类型有一定的限制。对这个主题的全面解决远远超过了这本书的范围，但是本书简短地构画出使用分治方法的一个有代表性的ILP技术。描述的方法是基于一个叫FOIL[Quinlan 1990.]的系统（关于本领域的全面和易读的论述，参见[Lavrač & Džeroski 1994]。[Muggleton 1992]是一本ILP论文集，[Muggleton & De Raedt 1994]是一个标准的参考）。

为了使讨论符合大部分归纳逻辑编程（ILP）著作习惯，本书使用类似PROLOG的符号，然而为了与FOPC约定相一致，继续用大写字母表示常量，小写字母表示变量符号。在归纳逻辑编程中，我们的目标是学习由Horn子句ρ构成的程序π，每个ρ的形式是$\rho: -\alpha_1, \alpha_2, ..., \alpha_i$，其中$\alpha_i$是与基本原子事实合一的原子公式。其思想是当$\pi$的变量被限制到一些值集（已知它们是在我们试图学习的关系中）时，它的值应该评估为真；这些值是训练集合的正例Ξ^+。当π的变量被约束到不在关系中的值时，π的值应该评估为假，这些值集是反例Ξ^-。就像在学习命题规则中一样，我们想让π覆盖正例而不覆盖反例。将与α合一的基本原子事实构成后台知识（*background knowledge*），假定它们已被给出——以能被运行和评估的补充PROLOG程序的方式，或者显式地用事实列表的形式。

做为一个例子，假设在一个大楼里运送东西的一个机器人通过经验发现，在某些位置对之间走动是容易的，但要走到其他位置对去就不是那么容易了，我们在图17-9中显示了这样一个大楼的部分地图。在这个大楼中，位置*A*、*B*和*C*是交叉点（*junction*），所有其他的位置是商店。

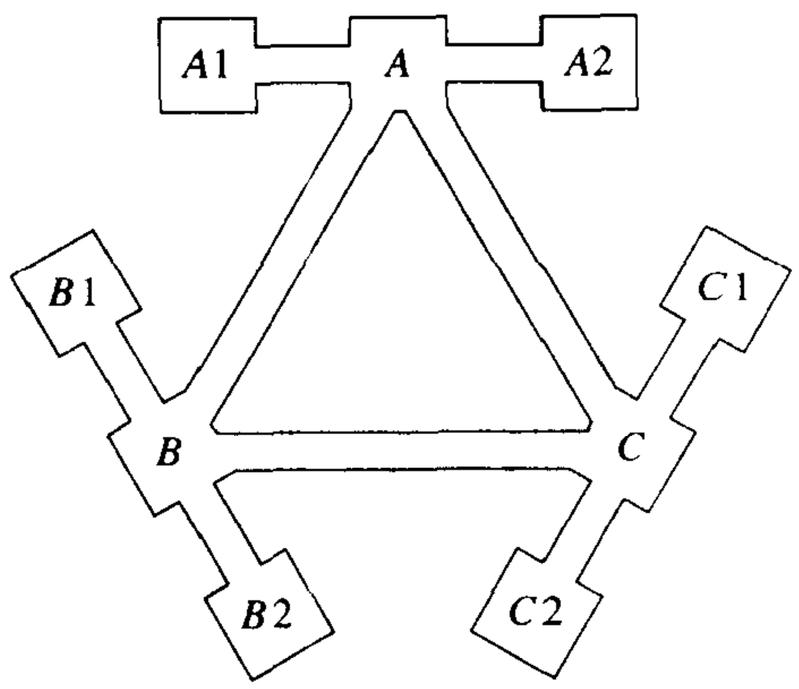


图17-9　一个建筑的部分图

假定机器人已经汇编了位置对的一个训练集Ξ，每对位置都被标识了，不管它是否容易导航。假定机器人有关于这些位置属性和位置之间关系的一些信息。特别地，对任何位置，我们知道它是不是一个交叉点；对任何位置对，知道它的一个成员是不是连向一个交叉点的商店。这些关系用表达式（Junction(x)和Shop(x, y)）表示。我们想让一个学习程序学习一个程序Easy (x, y)，它覆盖Ξ中的正例但不覆盖反例。Easy (x, y)能使用后台子表达式Junction (x)和Shop(x, y)。

为了使我们例子具体化，假定训练集由下面的Easy (Ξ^+)正例：

$$\{<A,B>, <A,C>, <B,C>, <B,A>, <C,A>, <C,B>,$$
$$<A,A1>, <A,A2>, <A1,A>, <A2,A>, <B,B1>, <B,B2>,$$
$$<B1,B>, <B2,B>, <C,C1>, <C,C2>, <C1,C>, <C2,C>\}$$

和下面的Easy(Ξ)反例：

$$\{<A,B1>,<A,B2>,<A,C1>,<A,C2>,<B,C1>,<B,C2>,$$
$$<B,A1>,<B,A2>,<C,A1>,<C,A2>,<C,B1>,<C,B2>,$$
$$<B1,A>,<B2,A>,<C1,A>,<C2,A>,<C1,B>,<C2,B>,$$
$$<A1,B>,<A2,B>,<A1,C>,<A2,C>,<B1,C>,<B2,C>\}$$

构成。

假定机器人能对Junction(x)和Shop(x, y)的任何变量值进行一段评估。特别地，对Ξ中命名的所有位置，只有集合{A，B，C}会给Junction一个真值。合并成一段在Ξ中提到的所有其他位置都给Junction一个假值。

对Ξ中命名的所有位置，只有下面（排序的）位置对给Shop真值：

$$\{<A1,A,>,<A2,A>,<B1,B>,<B2,B>,<C1,C>,<C2,C>\}$$

Ξ中提到的所有其他位置对Shop都有假值。

下面的PROLOG程序正好覆盖了训练集中的所有正例，且不包含反例：

```
Easy(x, y) :- Junction(x), Junction(y)
              :- Shop(x, y)
              :- Shop(y, x)
```

就像其他的学习问题一样，我们希望推理程序能具有一般性，即，如果表达用到的一些参数没有在训练集中出现（但对需要的后台关系，我们有它们的值），我们想让程序做出好的猜测（想像大楼里未在训练集中表达的其他地面和侧楼具有重复的结构）。

待解释的学习过程使用了一种GSCA算法。从一个没有体的单个规则的程序开始，给规则体增加文字直到它仅（主要）覆盖正例，然后用同样的方式增加其他规则，直到程序覆盖所有（或大多数）（有很少的例外）正实例（附加的放松办法可应用于数据有噪声的情况）。

我们有无限个可能文字能加到一个子句体中。实际的ILP系统用各种方式约束文字。典型的允许增加的条件是：

1) 文字要用在后台知识中。

2) 文字的参数是子句头中参数的子集。

3) 文字要引入一个新的独特的变量，它不同于子句头中的变量。

4) 一个文字使子句头中的一个变量等同于另外一个这样的变量或者一个在后台知识中提到的项。（这种可能性等价于通过一个置换形成一个特例）。

5) 一个文字要和子句头中的文字一样（除了它的参数）（这种可能性允许递归程序，在一些系统中不允许这种情况，在此我也不谈论它）。

在机器人导航的例子中，考虑加到一个子句中的文字是：

```
Junction(x)
Junction(y)
Junction(z)
shop(x,y)
shop(y,x)
shop(x,z)
shop(z,y)
```

(x=y)

遵从这里讨论的方法（通过加文字使子句具体化）的ILP系统会有很好定义的方法，以便计算可能加到一个子句的文字。选择加入哪个文字是更复杂的事情；大多数系统使用类似于在命题逻辑规则学习例子中的记分方法。

使用机器人导航的例子，演示了GSCA的ILP方法如何工作。知道谓词Easy是一个二位谓词，算法的内部循环把第一个子句初始化为Easy(x, y):-。这个子句包含所有的训练实例（正的和反的），因此我们必须加一个文字到它的（空）体。为了演算算法而不考虑如何选择一个要加入的原子公式，假定算法加入Junction(x)。Ξ中下面的正例被Easy(x,y):-Junction(x)覆盖：

$$\{<A,B>,<A,C>,<B,C>,<B,A>,<C,A>,<C,B>,$$
$$<A,A1>,<A,A2>,<B,B1>,<B,B2>,<C,C1>,<C,C2>\}$$

为了计算这个覆盖，使用后台关系Junction给出的城市作为基本事实，为Ξ中所有的城市对解释逻辑程序Easy(x,y):-Juntion(x)。下面的反例也被覆盖：

$$\{<A,B1>,<A,B2>,<A,C1>,<A,C2>,<C,A1>,<C,A2>,$$
$$<C,B1>,<C,B2>,<B,A1>,<B,A2>,<B,C1>,<B,C2>\}$$

因此，必须加入另一个文字。假定接下来加入Junction(y)。下面的正例被Easy(x,y):- Junction(x),Junction(y)覆盖：

$$\{<A,B>,<A,C>,<B,C>,<B,A>,<C,A>,<C,B>\}$$

不再有任何Ξ中的反例被覆盖，因此我们用子句Easy(x,y): - Junction(x),Juntion(y)终止内部循环的第一次循环。

但是仅由这个子句组成的程序π不能覆盖下面的正例：

$$\{<A,A1>,<A,A2>,<A1,A>,<A2,A>,<B,B1>,<B,B2>,$$
$$<B1,B>,<B2,B>,<C,C1>,<C,C2>,<C1,C>,<C2,C>\}$$

被Easy(x, y):- Junction(x), Junction(y) 覆盖的正例从Ξ中删除，形成下次内部循环中要用的Ξ_{cur}。Ξ_{cur}由Ξ中所有的反例以及还没被覆盖的正例（前面列出的）构成。为了设法覆盖它们，内部循环建立另一个子句，首先赋予Easy(x, y):-。这个子句覆盖所有的反例，因此必须加文字，假定我们加入Shop(x,y)。子句Easy(x,y):- Shop(x,y)不覆盖任何反例，因此我们完成了另一个内部循环。

Easy(x,y):- Shop(x,y) 覆盖Ξ_{cur}中的下述正例：

$$\{<A1,A>,<A2,A>,<B1,B>,<B2,B>,<C1,C>,<C2,C>\}$$

将这些实例从Ξ_{cur}中移走，并为下一次通过内部循环做准备。程序现在包含两个子句：

```
Easy(x, y) :- Junction(x), Junction(y)
           :- Shop(x, y)
```

由于没有覆盖下述的正例，该程序仍然不合适：

$$\{<A,A1>,<A,A2>,<B,B1>,<B,B2>,<C,C1>,<C,C2>\}$$

在下一次通过内部循环后，我们加入子句Easy(x, y):- Shop(y, x)。现在程序包含所有

的、且都是正的实例。因此过程终止于

```
Easy(x, y) :- Junction(x), Junction(y)
           :- Shop(x, y)
           :- Shop(y, x)
```

这个程序能应用于（也许以好的一般化）除了在Ξ中命名以外的其他位置——只要我们能对这些其他的位置评估Junction和Shop的关系就行。

17.5.3 基于解释的一般化

现在转向演绎学习，一种从以前的规则和事实推导规则的方法。用一个积木世界的例子解释这个方法。假定我们关于这个世界的一般知识包括下述规则：

$$Green(x) \supset Heavy(x)$$

$$Heavy(x) \supset \neg Pushable(x)$$

关于这个世界的事实之一是：

$$Green(A)$$

假定要求我们证明 ¬ Pushable(A)。在这个演示例子中，证明是简单的。然而在实际问题中它可能更复杂。在图17-10中显示了归结反驳。

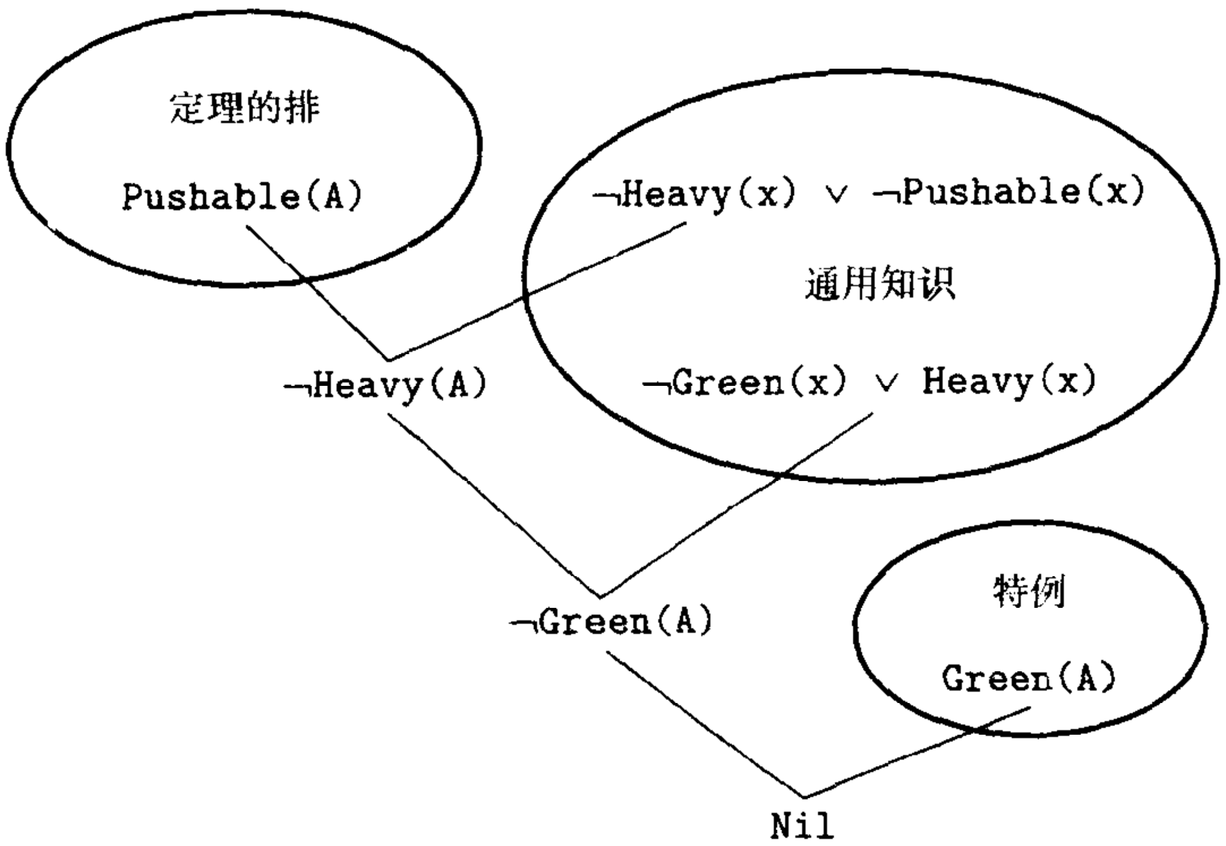


图17-10 一个用在EBG中的归结反驳

根据证明，我们能对结论构造一个解释。一个解释是用在证明中的事实集合。在这种情况下，对¬Pushable(A)的解释是Green(A)。用这个解释，我们能构造一个新规则：

$$Green(A) \supset \neg Pushable(A)$$

但是这个规则的应用十分有限，因为它只适用于积木A。然而，我们注意到证明能被一般化——A可以是任何东西，证明将仍然成立。把常量A用变量x代替使解释一般化，产生Green(x)。仿照 ¬Pushable(A)的证明结构，不用附加的搜索，我们看到用Green(x)代替Greeen(A)，能证明 ¬Pushable(x)。那么，这个过程的结果，即我们称为基于解释的一般化(EBG)的结果是生成规则

$$Green(x) \supset \neg Pushable(x)$$

这个规则可以单独由一个一般规则建立，因此我们并没有学到任何新东西。但是EBG过程可以指导我们从基于需要的规则去建立适用于特殊情况的规则。对更全面的EBG讨论，参见[Minton, et al. 1989]。

在执行EBG的过程中，我们假定一般规则对其他相似的情况也是有用的,因此值得推导和保存。当然，更多的规则会随着分枝因子的增加而减慢推理过程。因此EBG必须小心使用——可能要通过保持有关学到的规则的实用（*utility*）信息而获得。Minton按照由规则提供的搜索工作的平均节省、规则被使用的频繁程度及决定规则是否可应用的代价来定义实用[Minton 1988, Minton 1990]。他的PRODIGY系统只保留高实用的规则。

17.6　补充读物和讨论

[Levesque & Brachman 1987]讨论了在逻辑表示的可表达性和逻辑推理方法的易处理性之间的权衡。[levesque 1986]指出推理的难处理是由于在逻辑语言中存在析取和否定。他建议用鲜明的表示省却这些结构。如果不包含函数符号，只使用Horn子句的表示（称为Horn 理论）是易处理的。这些约束在一个叫DATALOG的语言中可以看到，DATALOG用来增加数据库的信息[Ullman 1989]。[Gogic et al. 1995]对各种命题逻辑和它的非单调扩展性的可表达性 、易处理性和简洁性进行了比较。

[Selman & Kautz 1991]提出使用“近似理论”加速某些推理过程。它们是Horn最大较低约束（GLB）和Horn最低较高约束（LUB）理论。如果一个公式不遵循一个LUB理论，它就不遵循原始理论。另一方面，如果遵循一个GLB理论，它也遵循原始理论。[Kautz , Kearns , & Selman 1993]建议使用特征模型作为一个可行的方式对Horn理论进行推理。

下一章将阐述术语逻辑（*terminological logic*）——通过约束表达语言获得易处理性的另一种方式。

作为对约束语言的一个可选方法，我们可以使用近似推理方法——它可能不是合理的和完备的。这方面的例子有随机模型采样[Khardon & Roth 1998]。

使用一阶逻辑（和归结方法）作为一门编程语言（和它的解释器）的第一个建议要归功于[Green 1969a]。[Kowals ki 1974]独自详细研究了这个思想，并激发了PROLOG的发明。Alain Colmerauer开发出了PROLOG的第一个解释器[Roussel 1975,Colmerauer 1973]。一个有效的实现归功于[Warren,Pereira,& Pereira 1977]。关于这个主题的主要论文期刊是《Journal of Logic》。

“与/或”图结构是求解问题方法的基础，该方法把问题简化为子问题的可选集合（就像在通过规则向后推理中一样）。[Nilsson 1980 第3章]介绍了搜索“与/或”图的一个算法AO*，它基于 [Nilsson 1969,Martelli & Montanari 1973] 的早期工作。[Davis 1980]描述了使用元规则控制基于规则的专家系统“与/或”树结构上的搜索，也可参考[smith,Genesereth，& Ginsberg 1986, Smith 1989]。

决定最小的ATMS标记是NP完全问题[Selman & Levesque 1990],即使在后台理论仅包含Horn子句的情况下。然而，[Kautz, Kearns, & Selman 1993]证明了这样的标记能在多项式时间内被计算——如果计算是基于特征模型的。其他有关TMS的著作，参见[Doyle 1979,de Klear 1986a, de Kleer 1986b, de Kleer 1986c, Forbus & de Kleer 1993，Shoham 1994]。

作为对专家系统的建立过程的一个概述，可参见[Bobrow,Mittal, & Stefik 1986]。[Stefik 1995]是一本涉及知识库系统建立的很多方面的课本（它也包含有关AI的很多其他材料）。尤其

有用的是Stefik的注释参考文献，它里面包含了该领域重要论文的概括短文（附录A）。

专家系统的一个例子是配置DEC公司的VAX-11/780系统，参见[McDermott 1982]。[Leonard-Barton 1987]介绍了专家系统在DEC公司的开发历史和使用。有关应用方面正在进行的工作在人工智能创新应用（IAAI）的年会论文集中有报告，有关专家系统的论文定期出现在IEEE的专家系统中。

关于谓词（概念）创新的更多情况，参见[Kautz & Selman 1992]和[Muggleton & Buntine 1988]。ILP应用的一个给人印象深刻的例子是在GOLEM系统中[Muggleton, King, & Sternberg 1992]蛋白质二次结构的预测。国际归纳逻辑编程研究组的学报包含该主题的当前论文。

知识表示和推理（KRR）国际会议论文集包含了描述当前研究成果的论文。[Brachman & Levesque 1985]是关于知识表示的重要论文集。

习题

17.1 下面的问题与PROLOG有关：

1) 解释一下为什么一个运行PROLOG程序的PROLOG解释器不能用来证明一个原子的否定。

2) 有些推理系统（如PROLOG），当它们不能证明原子ϕ时，就得出结论$\neg\phi$。这种否定类型被称为失败否定。解释一下为什么一般地我们不能因证明P为假而得出$\neg P$的结论。

17.2 如本章所描述的，PROLOG解释器在每一步执行一个指定的归结。如果一个PROLGO程序中的所有语句能被转化为文字的析取，描述一个归结反驳系统的控制策略，该系统能像PROLOG解释器一样执行同样的归结。

17.3 Tom声称是Paul Revere的后代。检验Tom声明的一个较容易方式是：通过说明Revere是Tom的祖先之一（向后搜索）或说明Tom是Revere的后代之一（向前搜索）？为什么？你的答案有例外吗？

17.4 在习题16.12描述的电路中，假如$W1$、$W2$和$W4$是“on”。描述一下如何用一个ATMS系统来诊断这个电路的可能故障。

17.5 一个归纳学习系统观察到每次Q为真时，下面的公式之一也是真：

P(A, B)

P(C, B)

P(D, B)

这个学习系统决定建立规则$(\forall x)(P(x, B) \supset Q)$

这个归纳一般化的合理性是什么？为什么系统不建立规则$(\forall x, y)(P(x, y) \supset Q)$？

17.6 一个专家系统被用来评估人们的信用风险，它基于大量事实，这些事实组合在一起产生一个数字分数。为了被判断为一个好的信用风险，分数必须比某个N值还要大。专家系统的设计者没有说N值是什么，但我们已经知道Pete被判断为是一个好的信用风险，Joe的分数比Pete的高。通过用下面的领域规则：

$\forall(x, y, z)[\text{Greater}(x, y) \wedge \text{Greater}(y, z) \supset \text{Greater}(x, z)]$

$\forall(x)[C(x) \supset \text{Greater}(\text{score}(x), N)]$

$\forall(x)[\text{Greater}(\text{score}(x), N) \supset C(x)]$

和下面的数据：

C(Pete)

Greater(score(Joe), score(Pete))

我们能证明Joe也会被判断为是一个好的信用风险。

首先，用这些公理和数据证明Joe是一个好的信用风险，然后用基于解释的一般化技术对你的证明建立规则

$[\forall(x, y)\{[\text{Greater}(\text{score}(x), \text{score}(y)) \wedge C(y)] \supset C(x)\}$

第18章　表示常识知识

18.1　常识世界

18.1.1 什么是常识知识

前面已经利用数码难题、博奕以及关于位置、墙、门、电池和积木等的网格世界来举例说明了几个AI技术。在上一章用一个非常简单的确定贷款申请人的专家系统解释了正向和反向推理时，稍微扩展了我们的讨论范围。当然，真正的专家系统比演示的例子有更多的事实和规则，但即使这样，它们也只包含相当有限领域的有限知识。

真正全面的、人类级的AI系统将需要更广泛的知识。这些系统将需要知道很多主题，已经证明对这些主题概念形式化是困难的。AI科学家发现对一个10岁的孩子是很容易的事情用专家们研究了很多年的AI方法也难驾驭，也许这比较荒谬。物理学家能够通过波动方程、相对理论和其他数学结构描述详细的、确切的物理现象，但是AI研究者仍在争论简单事实（但非常有用）——用液体填充一个杯状物，如果杯子倒过来液体将流出的最好表达方式。由物理学家和数学家发明的高等理论描述似乎比思想更容易形式化，毕竟，思想使人类甚至在亚里士多德以前就能很好地行使职责。

只考虑一些一个10岁的孩子就能知道的事情：

如果扔下一个物体，它将下落（今天，一个10岁的孩子也知道如果物体从一个轨道卫星上掉下，它将不会“下落”）。

人在出生前不存在。

鱼生活在水中，如果从水中拿出它会死掉。

人们在杂货店购面包和牛奶。

人一般在晚上睡觉。

这种知识常叫做常识知识(*commonsense knowledge*)。典型地讲，任何学科的知识可覆盖包括各种层次——从街上的行人到专家。据说，人类现存的公元前500年左右的最高级科学理论，与人们从日常观察结果所做的精心表述几乎没有什么差别。对某些人来说地球是平的，物体掉到地球上，因为那是“它们的归宿”，人类的疾病是由各种形形色色的“影响”引起的。当知识缺乏时，日常的常识知识很少同科学知识分离。常识知识对人们想做的很多事情是（仍然是）足够的。随着人们寻求对自己世界更精确的描述，科学知识逐渐和常识知识分开了。

甚至今天，我们想让机器人做的很多事情，如果不是基于史前的，可能也是中世纪的科学的。下雨时到棚子下面去（不需要有关低压系统的知识）；突然停下或转弯时，小心不要把咖啡溢出杯子（不需要高级的流体力学）；电池电压变低时要插上电源（不需要电化学理论）；及时付账单以便债权人不要采取法律行动（不需要复杂的债权人行为的心理学）。

在此并不是争辩当需要高级知识时，不要去用这样的知识。天气预报系统需要知道气象学；控制核电站热能交换机的系统需要知道热力学。这只是说，对很多任务仅仅10岁的孩童知

道的知识就够了。使用这些知识（一旦我们勾画出如何表示它）应该比使用过于详细的科学知识更方便。

AI研究人员把常识世界的知识叫纯朴（*naive*）知识。因此我们设法构造有关纯朴物理、纯朴经济学、纯朴心理学、纯朴统计学和纯朴社会学等理论。虽然我们已对纯朴知识和专家知识进行了比较，但应该强调的是这些术语略微涉及连续区的范围。不同的任务要求不同的关于这个连续区的知识级别。拿物理学做一个例子，为了把一个积木放在桌子上，一个机器人只需要知道对象的简单属性，对这个机器人的物理学常识知识将有：没有两个积木能放在相同的位置，一个积木必须被桌子或另一个积木支撑，等等。一个能把台球打到球洞中的机器人需要知道得稍多一点。对这个任务，机器人必须知道滚动磨擦、非弹性碰撞及动量转换等等。我们能明显地想到一些任务，它们将要求更复杂的知识直到我们逼近现代科学的前沿。因此，当我们说我们对形式化常识知识感兴趣时，不必清楚地定义这个连续区的区域，而是含糊地思考一个相当宽广的接近直觉的区域。

18.1.2　表示常识知识的困难

难以形式化常识知识的原因是什么呢？一个可能的原因是它的纯粹大量性。很多专家知识能用一种方式划分以便几百或几千个事实就足以建立有用的专家系统。一个具有普通人类级智能的系统需要多少知识呢？没有人知道确切数字。Doug Lenat设法建立这样的事实的知识库——称为CYC，他认为需要100万到1000万的事实 [Guha & Lenat 1990,Lenat & Guha 1990,Lenat 1995]。在1990年，CYC的作者说：

> 34年以来，AI一直在设法摆脱这样一个事实：可能并没有任何优雅的、不费力的方法去获得这个巨大的知识库，而是要花大量的力气（至少刚开始时）来人工输入每一个断言 [Guha & Lenat 1990, p.33]。

常识知识的另一个困难是没有很好的定义使我们能够控制独立于其他部分的部分的边界。常识世界的概念化将可能涉及到很多实体、功能和关系，它们可能会扩散到整个概念化中。因此，当设法开发一个概念化时，我们不知道是否已经“使它正确了”，直到我们几乎完成了全部的、大量的工作。

形式化常识知识的另一个障碍是关于某些主题的知识很难通过声明语句捕获。用语句难以描述形状和其他复杂的物理对象。例如，一个人的脸能用单词描述以便另一个以前没见过他的人能识别出他吗？我们怎样用文字来捕获一棵树、一个山景和一个热带的落日呢？如果某个东西不能用英语或某些其他的自然语言描述，我们有理由相信将不会找到用逻辑描述的一个概念化。

与捕获一个声明语句中的知识相关的困难是我们用来描述世界的很多语句仅仅是一个大概。特别地，全称语句（即形如所有x的都是y的）很少有效，除非它们仅仅是一种定义。已有各种对普通逻辑的修改用来处理很多知识是近似的事实。在本章的后面介绍了这样的一种方法，在第19章和第20章提到另一个基于概率论的方法。

除了由相互依赖的知识引起的困难，我们应该如何概念化一些主题也并不总是明确的。例如，我们应该把时间当作实数集呢（即作为瞬时连续），还是仅仅是整数，还是实数轴上的区间，还是某些其他的东西呢？我们可能想把过去想像为一个单一的时间线，那我们怎么想像未来呢？尽管我们可能不想承认有超过一个未来（Qué, Será, Será），但我们都习惯于想像我们的

动作能影响的可选未来。我们怎么概念化世界以使形如“如果我不上法律课，我就不会遇到你”这样的句子有意义呢？我们想让一个意图成为怎样一种实体，以便我们的概念化能处理像“我没打算迟到”这样的语句？

18.1.3 常识知识的重要性

我们承认用常识来构造机器是困难的——为什么它重要呢？难道大多数有用的AI应用不在知识并不难形式化的专家系统中吗？有几个答案可以回答这个问题。首先，对有常识的机器将发现很多商业价值的应用。没有几个人对“家用机器人”有争论，这样一个机器能做像保持房间干净、洗衣、做饭、进行日常的家庭维护（如更换烧掉的灯泡）及洗盘子等等的日常家庭工作。但是想像一下这样一个机器人必须拥有的知识以足以回答如下的问题：烧掉的灯泡扔到哪里去？怎么拿起一个盘子、一个杯子？从洗碗机出来的东西放到哪儿去？你怎么告诉机器人吸尘器需要打扫？放在冰箱中的生菜一般保存多久？等等。

对常识知识是为了让专家系统更有用也需要讨论。专家系统仅仅在专门知识的非常有限的领域内执行良好。一般的常识知识至少使他们能认识到用户什么时候想要那个区域以外的信息，它也允许系统更准确地预测什么时候它的知识和手头的任务相关，什么时候不相关。

扩展一个专家系统的知识中常识知识的结构也是重要的。我们都熟悉在推理中使用类推和比喻。空间比喻尤其普遍。例如，我们说量子电动力学超过了我们的知识范围；洁癖紧挨着信仰；Mary的薪水高于John的，等等。有充分的理由怀疑比喻不只是简单的语言巧合。事实上，很多好的主题的概念化基础是基于空间和其他常识思想的。因此，一个拥有世界的基本常识概念化的专家系统可以几乎不再需要扩大和修改来扩展它的知识库。

最后，为了理解自然语言——在第24章提出的一个主题，常识知识毫无疑问是需要的。

18.1.4 研究领域

尽管我们还没有具有常识知识的系统（CYC是可能的例外，它也是一个正在进行的工作），AI研究人员已经在几个前沿对常识知识的表示展开了研究：

1) 对象和材料。世界是由对象构成的。有些对象如网格世界中的积木是离散的、固态的东西这些相对容易讨论和描述。有些对象是有层次的，即是由各部分（其他对象）按某种方式放在一起构成的。也有流体、气体和汇集，像沙堆、面粉袋和星系。描述材料及其属性（尤其是流体）的著名成果是[Hayes 1978，Hayes 1985a，Hayes 1985b]的研究。
2) 空间。物理世界有空间范围。对象存在于空间中，在空间中的位置是相对于其他对象的。因此，谈论一个东西是在另一个的里面、上面和紧挨着这些情况，例如，我们必须能描述东西是多么大，它们的形状如何等等。形式化有关空间的各种符号的一个早期AI成果是[Kautz 1985]。在各种机器人任务中有关空间推理的论文，参见[Chen 1990]
3) 物理属性。AI系统也应该能推理这些物理属性，如质量、温度、体积、压力、放射性级别、波长和它们之间的任何关系。
4) 物理过程和事件。物体下落，球被扔出，草长出了，杯子被倒满又倒空，蜡烛燃烧，热东西变凉。在物理学中，很多这样的过程是用不同的方程式描述的，能在AI中使用这些方程式。然而，经常我们不需要由物理过程提供这些确切的（且昂贵的）求解。相反，AI研究者已经开发了一个定性物理，利用它推理普通趋势而不需要确切的计算[Weld & de Kleer 1990]。

5) 时间。过程（包括计算）发生在时间里，计算机科学家和AI研究者已经开发了各种技术来描述和推理时间。用在计算机程序分析中的特殊时态逻辑[Emerson 1989]对建立在它们中的时间有一定的重要特征。AI人员已打算用两种其他的方式解决时间问题（但是[Shoham 1987]讨论了在AI中使用时态逻辑）。首先，通过参照状态，即一个未指定的时刻世界的“快照”，能够忽略时间的明确阐述。状态由动作连接，动作将一个状态转化为另一个状态。在第21章讨论这个问题。第二，时间和时间间隔能被包括在被明确推理的实体之中。作为常识推理需要的一个概念形式化的例子，下一节将描述用这个方法来对时间形式化。

18.2　时间

我们如何考虑时间呢？这像一个数字的“实数轴”延伸到了无限的过去和将来之中吗？或者它像可数的整数，在宇宙形成的“大爆炸”中从0开始，用离散的时间单位滴滴嗒嗒地向前走。一些早期的科学家认为时间是圆形的——在一个无休止的循环中重复运行。在我们能陈述关于时间的有用事实前我们需要决定将使用哪种描述。

AI中最常用的一个描述方法是由James Allen[Allen 1983, Allen 1984]形式化的（也见[Allen 1991a]，它讨论了很多表达时间的方式）。在这个描述中，时间是事件和过程在它里面出现的某个东西（在一个完全静态的世界，没有任何东西在继续，将会不需要时间；事实上，在这样一个世界上也难以理解如何定义时间）。这些包含事件和过程的“容器”被称为区间，时间区间就像实数轴上的区间一样。那么在这个概念化中，时间区间是在“存在”的实体之中。

为了描述时间区间，需要给它们命名，我们将用谓词演算对象常量，如I1，I2,...，来指称它们。为了说由E指称的某个事件或过程（完全地）占据区间I，我们记为Occurs（E, I）（留给读者去思考如何对事件和过程符号形式化；对当前的目的，它们也是“存在”的实体）。

时间间隔有开始和结束时间点。时间点被当作实数。一个区间的开始由一个函数start给定，结束用函数end给定。关于间隔的一个基本事实是（∀x）[start(x) ≤ end(x)]

（当区间的开始和结束相同时，间隔则成为退化的情况）。

如下定义区间之间的一个基本关系：

$$(\forall x, y)[\mathbf{Meets}(x, y) \equiv (\mathbf{end}(x) = \mathbf{start}(y)]$$

（如果第一个的结束是第二个的开始，两个区间相接）。我们能按照相接或区间的开始和结束时间定义区间之间的其他6种关系。它们被表示为Before, Overlaps, Starts, Ends During和Equals。我们也有相反的表示Met_by After, Overlapped_by, Started_by, Ended_by和Contains(Equals的逆是它自己)。例如

$$(\forall x, y)\{\mathbf{Before}(x, y) \equiv (\exists z)[\mathbf{Meets}(x, z) \wedge \mathbf{Meets}(z, y)]\}$$

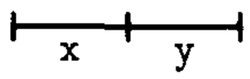
$$(\forall x, y)\{\mathbf{Before}(x, y) \equiv [(\mathbf{end}(x) < \mathbf{start}(y)]\}$$

在图18-1中给出了这些关系的一个图形表示，并留给读者去完成用Before开始的定义。

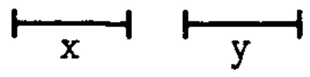
区间关系能被用来表示一些时间中的事件的常识事实。例如，为了说水从水龙头流出是首先反时针旋转一个阀门然后顺时针旋转阀门这样一个事件，我们可以写出

$$(\forall y)\{\mathbf{Occurs}(\mathbf{Flow}, y) \supset (\exists x, z)[\mathbf{Occurs}(\mathbf{Turn\_ccw}, x) \wedge \mathbf{Occurs}(\mathbf{Turn\_cw}, z) \wedge \mathbf{Overlaps}(x, y) \wedge \mathbf{Overlaps}(y, z)]\}$$

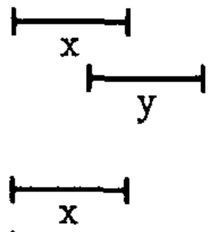
也有一些基本公理可用来表达Before的传递性。这种特殊的时间形式化已应用于各种时间推理问题中。



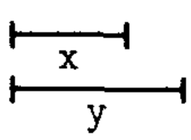
Meets(x,y) Met_by(y,x)



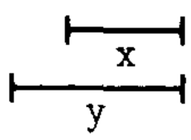
Before(x,y) After(y,x)



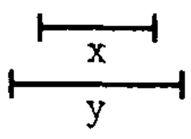
Overlaps(x,y) Overlapped_by(y,x)



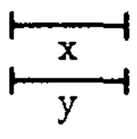
Starts(x,y) Started_by(y,x)



Ends(x,y) Ended_by(y,x)



During(x,y) Contains(y,x)



Equals(x,y) Equals(y,x)

图18-1 区间之间的关系

18.3 用网络表示知识

18.3.1 分类的知识

通常，常识领域和专家领域的实体都能被安排在层次结构中，用以组织和简化推理。例如，在CYC的常识知识表示中，最基本的实体是用对象常量Thing指称的。在CYC中，有几类thing: 世界中的对象、数学对象、事件和过程等等。这些可以安排在分类或层次中，它们可以隐式地对形如“X是一个P，所有P的都是Q的，所有Q的都R的”、等等事实进行编码(讨论CYC的层次性见[Guha & Lenat 1990])。分类层次能用网络或称为框架的数据结构编码。首先讨论网络表示，使用有关办公机器的信息作为一个演示例子。

假定我们想表达如下的事实：Snoopy是一台激光打印机，所有的激光打印机都是打印机、所有的打印机都是机器，附加一些相关的信息。用谓词演算语句，我们可能有

Laser_printer(Snoopy)

(∀x)[Laser_printer(x) ⊃ Printer(x)]

(∀x)[Printer(x) ⊃ Office_machine(x)]

谓词Laser_printer, Printer和Office_machine用一个分类法表达分类。使用分类知识的一个重要推理涉及到分类传递。例如，给定前面的事实，我们能推导出：

(∀x)Laser Printer(x)⊃office_machine(x)]和office_machine(Snoppy)

每个分类类别的成员可能都有一定的属性，如所有办公机器的电源是墙上的一个电源插座。这

些属性能用一个函数和一个等式谓词表达：

$$(\forall x)[\mathtt{Office\_machine}(x) \supset [\mathtt{energy\_source}(x)=\mathtt{wall\_outlet}]]$$

注意子类别的成员通常继承了它们父类别的一般属性：

$$(\forall x)[\mathtt{Laser\_printer}(x) \supset [\mathtt{energy\_source}(x)=\mathtt{wall\_outlet}]]$$

在一个分类层次中实体的这些一般推理可方便地通过一个叫语义网络的图形来表示。

18.3.2　语义网络

语义网络是对对象及其属性分类知识编码的图形结构。下面讨论一个体现其主要思想的简单例子。其中有两类节点：

- 由关系常量标识的节点，对应分类类别或属性。
- 由对象常量标识的节点，对应领域对象。

有三类连接节点的弧：

- 子集弧（有时叫isa连接）。
- 集合从属关系弧（有时叫实例连接）。
- 函数弧。

图18-2中给出了一个有各种节点和弧的网络例子。

关于属性和集合从属关系的推理在一个语义网络中比在对应的合式公式上使用非导向的归结更容易、更有效[⊖]。为了确定由节点*A*表示的一个对象是不是由节点*B*表示的集合的一个成员，我们从*A*顺着弧向上看能否遇到节点*B*。参照图18-2，可以方便地确定R2D2 是一个办公机器。为了确定由*A*表示的节点的一些属性值，我们从*A*顺着弧向上寻找有相同属性的一个节点。例如，为了确定Snoopy的电源，我们顺着从Snoopy到Office_machines的弧，将在那里找到答案Wall_outlet。

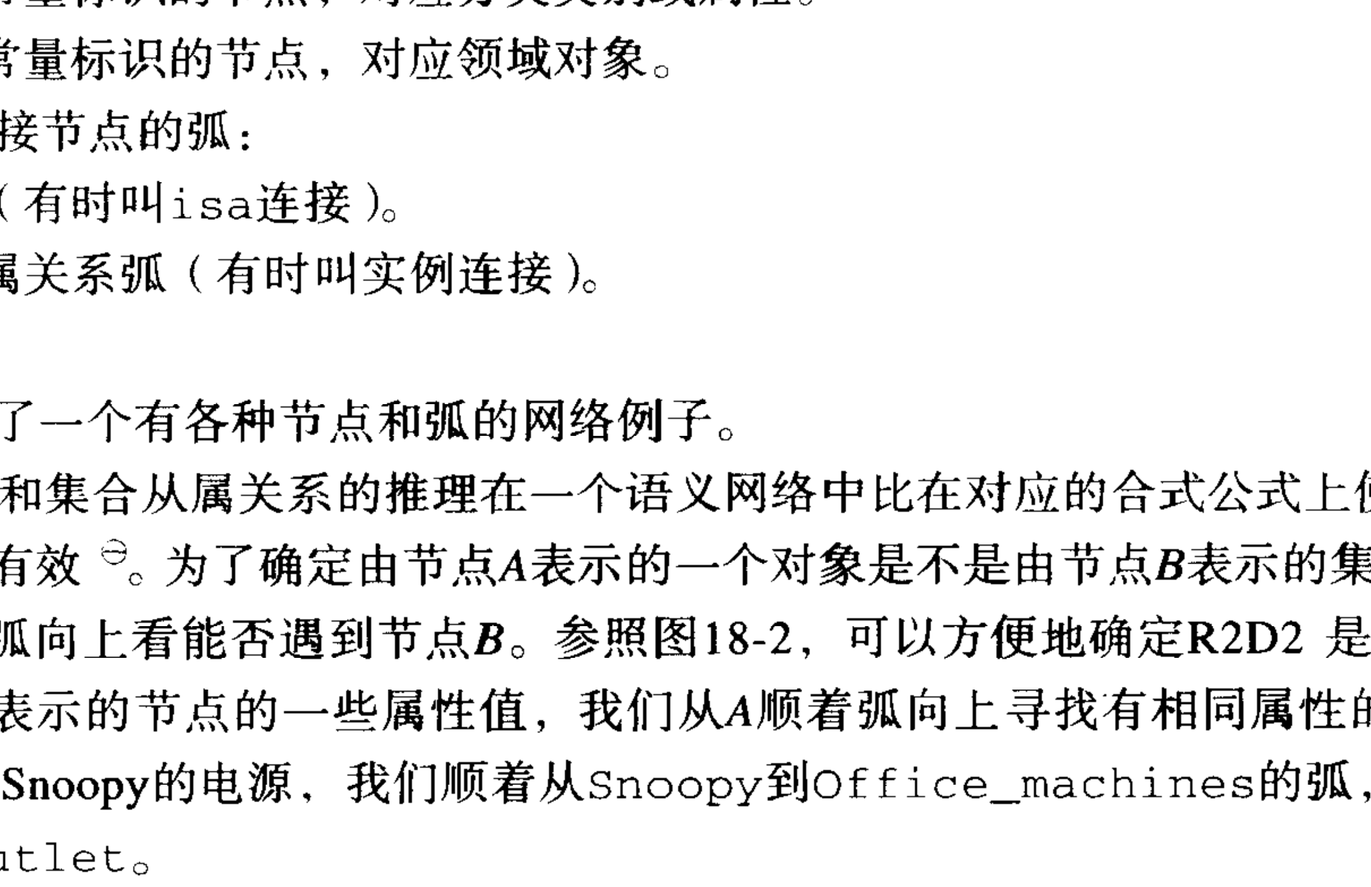


图18-2　一个语义网络

⊖ 尽管考虑了变量的“类型信息”和其他启发式的合一过程的修改，产生的归结系统基本上也是有效的[Sticke 1985]。

18.3.3 语义网络的非单调推理

如上所述，通用逻辑中的推理是单调的，因为将公理添加到一个逻辑系统不会减少可以证明的定理集合。即，如果Δ'是Δ的一个超集，那么对任何ω，如果Δ ⊢ω, 则Δ' ⊢ ω。但是人类使用的很多常识推理的例子是非单调的。我们经常做缺省推理——除反面的知识以外，我们宁愿假定它们为真。但是新的、矛盾的知识出现时，我们必须收回缺省推理。已经提出了很多系统和逻辑机制来捕获这种非单调现象——前面已经描述过TMS如何对之进行处理。其他的方法有缺省逻辑[Reiter 1980]、自动认识逻辑[Moore 1985a]、非单调逻辑[McDermott & Doyle 1980]和界限[McCarthy 1980，McCarthy 1986]。

下面描述一个简单的（虽然不完全充分）、叫继承抵销的非单调推理的机理，该机理用语义网络可得到最好解释。假如我们想说办公机器的电源缺省是墙上的电源插座，但作为一个例外，一个机器人的电源是电池。我们能像图18-3那样加入另外的函数弧到语义网络中来表示这个知识。

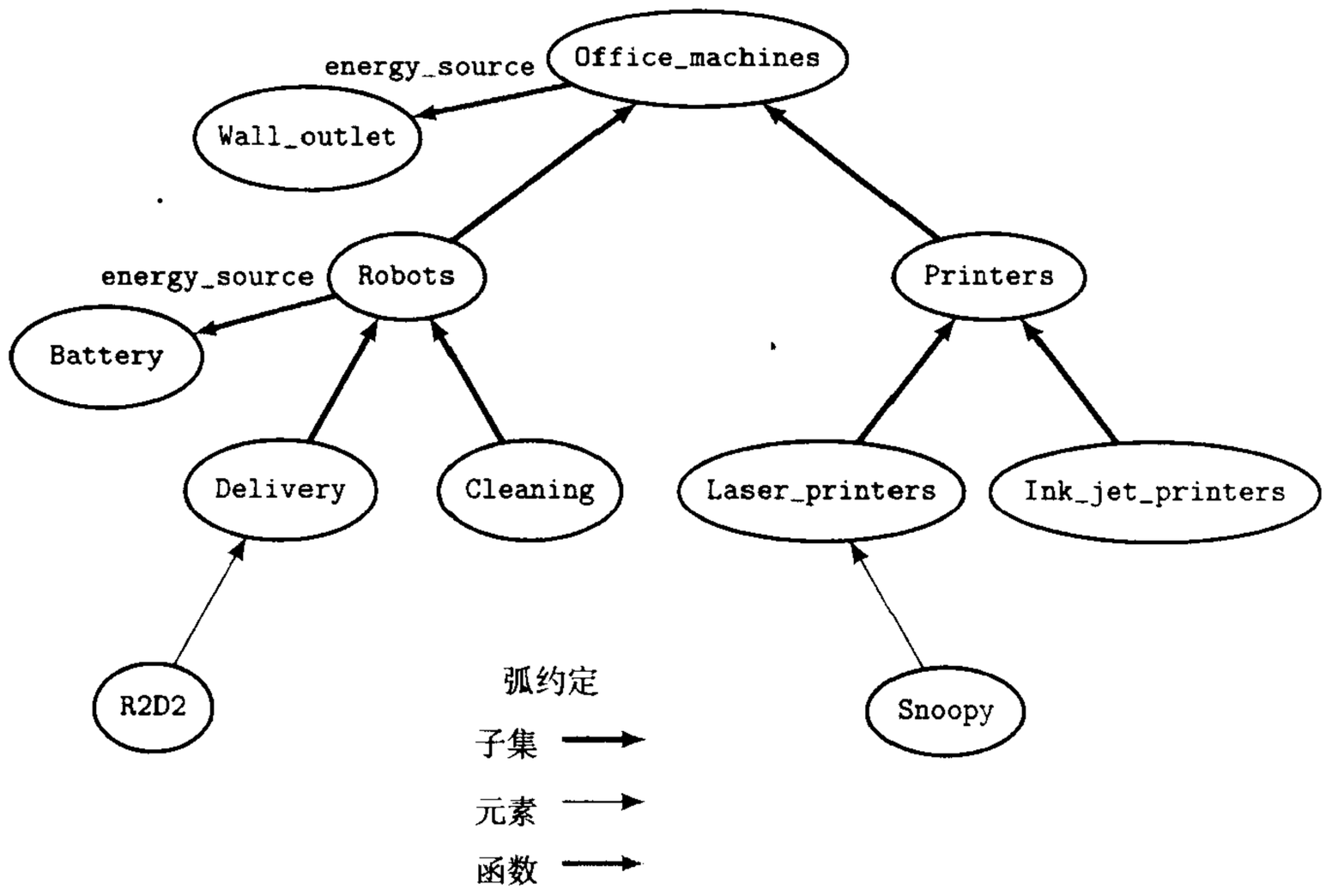


图18-3 一个缺省推理语义网络

新网络中的属性继承机制会导致一个矛盾。虽然属性继承允许我们推导出（正确地）打印机的电源是一个墙上插座，但它也同明确表达的事实——机器人的电源是电池相矛盾（假定电池和墙头插座不同）。我们能通过使用网络来避免这个矛盾。最特殊分类（根据子集或实例弧排序）优于不太特殊的分类。因此，如果想查询R2D2 的电源，首先要在网络中定位R2D2节点，查询该节点是否有energy_source函数弧。如果有，我们接受那个弧的头节点给出的答案作为最终回答。如果没有，我们顺着层次结构跟随弧向上直到到达第一个有energy_source函数弧的节点，然后用弧的头节点回答这个查询。在分类层次中与上层节点相关的信息是一般的、缺省的信息，会被层次结构中低层节点相关的特殊信息抵销。

有各种与属性继承机制相关的困难。问题之一是从不同双亲节点继承的属性可能冲突。图18-4中的子网络展示了这个问题。C3PO既是一个搬运机器人又是一个清洁机器人；它在晚上工作还是白天呢？在冲突情况下，不能做出任何结论。在更复杂的非单调系统中，缺省知识被

区分优先次序，以便在多继承情况下一个缺省能优于另一个。

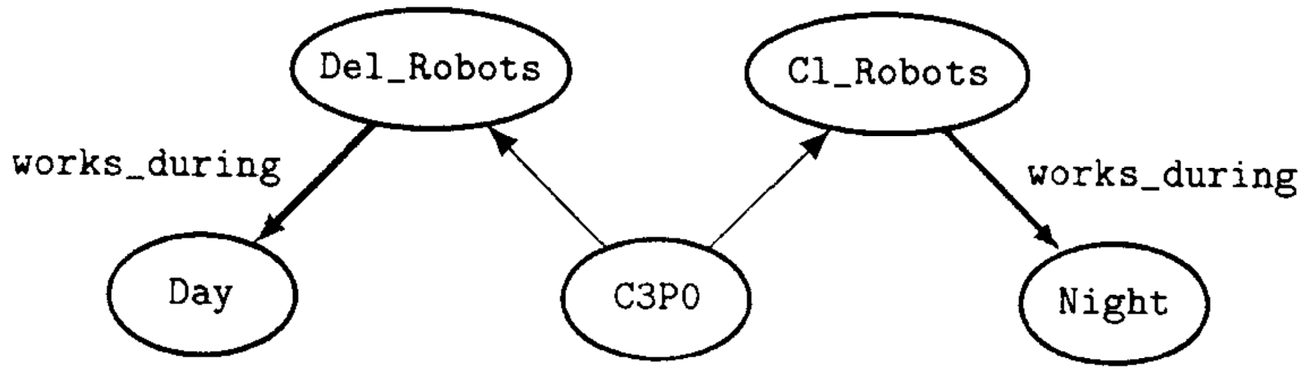


图18-4　冲突缺省

18.3.4　框架

分类知识也能用一个叫框架（*frame*）的数据结构进行编码。一个框架有一个名称和一个属性—值对集合。框架的名称对应语义网络中的节点，属性对应于与这个节点相连接的弧的名字，值对应这个弧的另一终端的节点。属性—值对常被叫做槽(*slot*)，属性叫做槽的名字，值叫做槽的填充符。在图18-5中给出了一个例子。注意我们也能用这些表达方式表达叫做元知识的知识；例如，创建框架的日期和框架的创建者是有关框架的知识，不是有关打印机的。

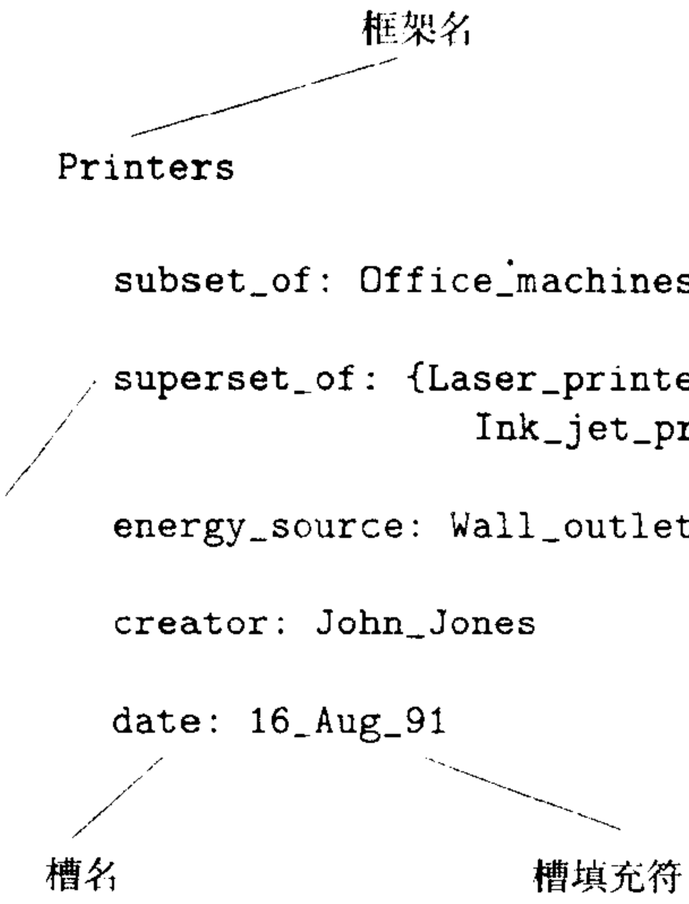


图18-5　一个框架

然而，语义网络和框架在表达某些知识时会有困难。例如，表示析取（及蕴含）、否定和一般的、没有分类的知识是困难的（但并非不可能[Hendrix 1979]）。这些问题导致出现混合系统，如KRYPTON[Brachman,Gilbert, & Levesque 1985]。CLASSIC是另一个混合系统[Borgida,et al. 1989]。它们使用所谓的术语逻辑——利用层次结构表示实体、类别和属性，以及关于其他信息的逻辑表达。

18.4　补充读物和讨论

关于常识表示和推理方法的更多知识，参见[Davis 1990,Hobbs & Moore 1985]。在1993年第1期（总第61期）的《Artificial Intelligence》的第41~104页中，有[Lenat & Guha 1990]的五篇评论是关于CYC的。

[Sowa 1991]是经过编辑的有关语义网络的一本论文集。在语义网络中为了有效推理，被AI研究者开发的属性继承机制已被融入作为面向对象编程语言的类继承的一个基本特征。各种术语逻辑系统的一个实际比较，参见[Heinsohn, et al. 1992]。

语义网络中的继承抵消是一个非单调的推理技术。我已经引用了更复杂的缺省逻辑和界限机制。[Brewka,Dix, & Konolige 1997]有关于这方面的综述。另一个非单调推理约定称为封闭世界假设（CWA）。CWA通过在Δ中包括不能从Δ推理的所有基本原子的否定而扩大了谓词演算公式的知识库Δ。因为如果一个原子α被加到Δ，被CWA加入的 $\neg\ \alpha$必须被收回，因此上述过

程是非单调的。关于非单调推理的一本论文集，参见[Ginsberg 1987]。

关于知识表示的一些工作集中在允许不同知识密集度的程序共享信息的语言和形式化中。例如，知识互换格式（knowledge interchange format KIF）[Genesereth & Fikes 1992]和KQML[Finin,Labrou, & Mayfield 1997]。这些系统依赖于建立叫做存在论的常识概念化。[Gruber 1997]描述了这些设计的原则。

人类常识的表示和推理涉及到很多推理，它们与这本书中讲到的逻辑推理很不一样。类比和比喻的使用是基础。已经有几个AI和心理学方面的研究人员研究类比推理，其中包括[Gentner 1983]。[Lakoff 1987,Lakoff & Johnson 1980]描述了类比在人类思考和语言方面所扮的角色。基于事例的推理是另一个推理方法，它利用了前面解决问题的类比和相似性[Kolodner 1993](1993年第3期(总第10期)《Machine Learning》的第1~5页有基于事件推理的专门讨论)。

习题

18.1 假定知识能用由构成知识库的谓词演算公式编码。例如，考虑下面的公式，它们是对ACME公司的员工和组织知识的编码。

$$\vdots$$

$$\mathtt{Salary(Joe, 20000)}$$

$$\mathtt{Dept(Joe, Sales)}$$

$$\mathtt{Boss\_of(Joe, Henry)}$$

$$(\forall x, y)[[\mathtt{Dept}(x, \mathtt{Sales}) \wedge \mathtt{Salary}(x, y)] \supset (y \geq 20000)]$$

$$\vdots$$

编码知识的这些公式的任何解释当然也必须应用在这样一种情况下，其中，所有助记谓词常量（如salary）和对象常量（如Henry）被提示性知识较少的“gensym”常量代替。从何种技术意义上讲，使公式（假定用gensym常量）能对ACME公司的知识进行编码？

18.2 考虑下面的论点：

Wellington听到了Napoleon的死讯。因此，Napoleon不能听到Wellington的死讯。

给出这个论点的一个形式化的一阶逻辑推理（提示：当进行知识表示时，这常常是一个好的想法：用一个临时词汇表和一个临时的公理集开始，当你看到对证明需要的东西时修改它们）。

1) 列出这个领域的一阶逻辑知识表示的一个关系常量、函数常量和对象常量的集合，领域的元素应该是人、时间和文件（如一个人的死亡）（提示：在有意义的情况下使用函数常常是容易的。对一个函数来讲，最好不要在整个空间上定义（例如，一个事件可以有时间，而一个人不可能），只要它总是被一致地使用）。

2) 用这个词汇表，写出基本的事实和用在这个论点中的任何常识公理。

3) 使用一个正式的逻辑证据，证明你写出的事实的结论。

18.3 假如要证明一只叫Tweety的鸟位于一个动物园中。如果我们有一个知识库，则我们的目标有点类似于

$$\mathtt{Bird(x)} \wedge \mathtt{Name(x,Tweety)} \wedge \mathtt{Located\_in(x,y)} \wedge \mathtt{Zoo(y)}$$

假如我们知道有500只鸟，5只叫Tweety，10^{10}个对象/位置对，100个动物园。而且，每个对象有惟一的位置，每个位置有大约1000个对象。

1) 在通过证明顺序中的每个合取项的求解方案中（考虑在求解合取项中所做的替换），什么是合取项的最佳排序？（对一个给定的问题，最佳合取项排序是能够产生最小搜索空间的顺序，换句话说，就是给出最少数量的要检查的可能方案的顺序。）

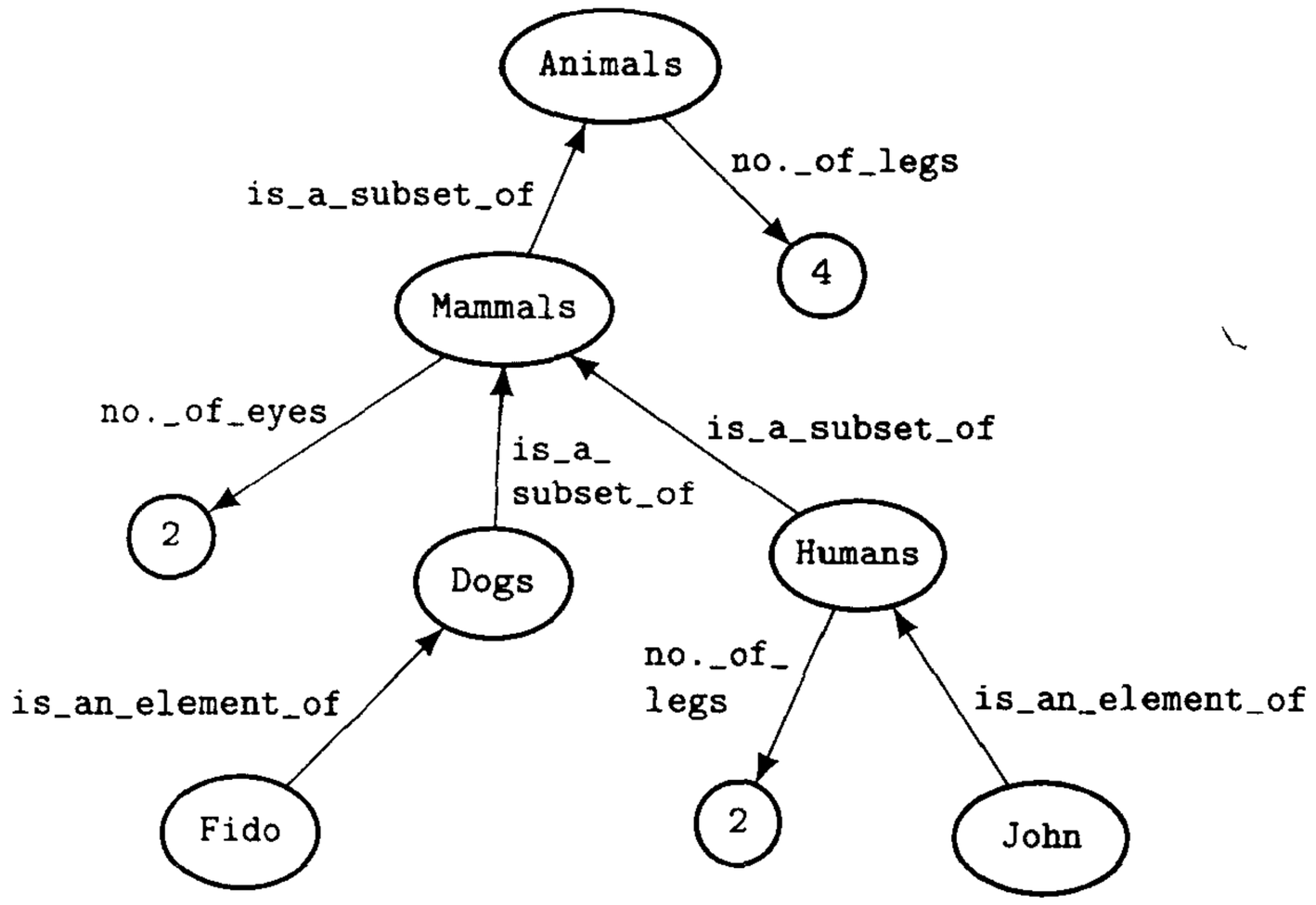
2) 一个常用的启发就是按照有最少数量答案的顺序排列合取项（在它尝试的时间）。这种启发有时叫最小的优先代价。使用这个启发的合取项顺序是什么？

3) 证明代价最小的优先启发总是最优的，或者修改例子建立一个反例。

18.4 构造一个例子，证明封闭世界假设（CWA）应用到一个公式集合会导致不一致性。

18.5 1) 把这里显示的语义网络中的信息表达为一个谓词演算合式公式，使这个合式公式能显式描述在继承层次中隐式的继承抵销信息。

2) 从谓词演算合式公式中能证明Fido有4条腿吗？如果不能，为了完成这个证明需要加入什么信息？



18.6 对下面的两个例子，挑选最合适的知识表示系统，用选择的表示对例子的陈述编码。用你的表示和推理系统回答每个例子中的问题。详细解释推理系统是如何到达答案的——例如，如果使用一个基于归结的推理系统，证明归结反驳。如果在你的解释中图有帮助，可以使用它：

1) 一般地讲，鸟会飞。驼鸟是鸟。Oliver是一只驼鸟。驼鸟不会飞。

问题：Oliver会飞吗？

2) Seymore是一头大象且是John的朋友。所有的大象都是灰色的，或者它们是Bert的一个朋友。Bert没有任何朋友是John的朋友。

问题：Seymore是灰色的吗？

第19章　用不确定信息进行推理

一个agent关于它的任务和环境常常只有不确定的信息。到目前为止所讨论的技术对不确定知识的表示和推理能力十分有限。一条语句如P∨Q允许我们表示不确定性——P和Q哪一个是真的，但是还没有描述如何表示关于P或Q不确定的程度。

在一般逻辑中，我们能从P和P ⊃ Q中推导出Q，即，如果一个agent知道P⊃Q，它随后学会了P，那么它也能推导出Q。当信息不确定时有类似的推理过程吗？各种形式化已被用来对不确定信息进行表示和推理。前面已经提到过被MYCIN和PROSPECTOR使用的形式化。开发得最好的形式化（一些人对“最适合的”有争议）是基于概率的。在本章的开始，先回顾一下概率论的基础知识。

19.1　概率论简介

19.1.1　基本思想

假定有随机变量V_1,V_2，…，V_k的一个集合。当我们想谈论V_i的值而没有说这个值是什么时，我们使用符号v_i。在例子中，随机变量代表一个所讨论领域的特征。随机变量的值可以是不同类型。如果变量代表命题，它们的值为True或False(或者，1或0)；如果变量代表物理度量（如高度、密度和速度等），则值是数字；如果变量代表分类（如颜色、字母表中的字母等），则值是范畴。例如，投一个硬币的结果可以用简单变量C表示，它的值c可以是分类值H（头）或T（尾）。如果我们正在谈论把一个硬币投k次的结果值，我们需要k个变量（$C_1,\cdots,C_k$），它们中的每一个有值H或T。

我们用表达式$p(V_1=v_1,V_2=v_2,\cdots,V_k=v_k)$指称一个联合概率，即变量$V_1$，$V_2$，…，$V_k$的值分别是$v_1$，$v_2$，…，$v_k$时的概率。表达式$p(V_1, V_2,\cdots, V_k)$叫做变量$V_1$，$V_2$，…，$V_k$的联合概率函数。它把变量集合映射为一个在0和1之间的实数。把$p(V_1, V_2,\cdots, V_k)$中的变量替换为特定的值，以给我们一个表达式$p(v_1, v_2,\cdots v_k)$——$p(V_1=v_1,V_2=v_2,\cdots,V_k=v_k)$的缩写形式。因此，对一个公平的硬币投掷，我们有$p(H)=1/2$。如果我们将一个硬币公平地投五次，就可能有$p(H, T, T, H, T)=1/32$。$p(H, T, T, H, T)$是第一投结果为头、第二投为尾、第三投为尾、第四投为头和第五投为尾的一个联合概率。

概率函数必须满足一定的属性，它们是：

(a)　$0 \le p(V_1, V_2, \ldots, V_k) \le 1$

上式适用于变量的任何分配值，且

(b)　$\sum p(V_1, V_2, \ldots, V_k) = 1$

其中的和是建立在变量的所有值的基础之上。因此，在投硬币的例子中，$p(H)=1/2$是和（a）一致的，然而$p(H)=1/2$和属性（b）一起限制$p(T)$等于1/2。这里不过多谈论如何给随机变量值分配概率,就像在命题演算中由合式公式指称的各种命题的真假是基于应用领域的专家主观判断（或者传感数据的知觉处理），随机变量的概率值也同样依赖专家判断或知觉处理。相反，

我们主要关心的是怎么执行计算，以让它告诉我们所感兴趣的变量的概率。

在本章的应用例子中，变量对应一个域的命题。这些命题或为真或为假，相应的命题变量将有True或False值。我们可能不确定关于这些命题的一个或多个的事实，这种不确定性能用相应变量的值的概率表示。因此，本章描述的技术可以认为是第13章和第14章讨论的使用谓词逻辑进行表示和推理的概率方案（正在开发的一阶逻辑概率方案是一个前沿的研究问题，可参见[Nilsson 1986,Glesner & Koller 1995]）。

用一个特定的例子将有助于表达重要的概念。用一个和前面演示命题演算中的推理相同的例子。回想命题原子BAT_OK、MOVES和LIFTABLE，它们的意图分别是电池被充满了电、手臂可以移动（当拿积木时）和积木是可以举起的。除此之外，我们加入原子GAUGE，它指称电池状态的量度，表示电池是被充满电的。为了使图和公式不太繁杂，用简单的字母B、M、L和G重新命名这些原子。现在，假定我们不能确定这些原子是*True*还是*False*。在提取任何传感器的读数前，有一个对这些值的各种组合的先验概率。例如，当其他都是*True*时，我们认为M为*False*是不太可能的。

因为有4个二值变量，故对这些变量的每种形式（B=b, M=m, L=l, G=g），其中b, m, l和g的值是*True*或*False*，有16种联合概率。一个agent设计者可以指定这16个值，同时要遵守约束：每个值在0和1之间，它们的和是1。作为一个例子，在下面的表中列出了这些联合概率中的一部分：

(B, M, L, G)	联合概率
(*True*, *True*, *True*, *True*)	0.5686
(*True*, *True*, *True*, *False*)	0.0299
(*True*, *True*, *False*, *True*)	0.0135
(*True*, *True*, *False*, *False*)	0.0007
…	

（当然，一个设计者不可能用表中给定的精确程度指定概率。这样做是为了使这些值和本章后面给出的这个例子的其他有关概率相一致）。

当我们知道一个随机变量集合的联合概率的所有值时，就能计算这些随机变量之一的边缘概率(*marginal probability*)。例如，边缘概率$p(B = b)$被定义为是16个联合概率中B= b 的8个概率之和：

$$p(B = b) = \sum_{B=b} p(B, M, L, G)$$

用这个公式，边缘概率p（B=*True*）= 0.95,这是B值为*True*的8个联合概率之和。

更低阶数的联合概率也能通过对所有联合概率的合适项相加而计算得到。例如，对于$B = b$，$M = m$，联合概率$p(B = b, M = m)$是所有联合概率中4项之和。

$$p(B = b, M = m) = \sum_{B=b,M=m} p(B, M, L, G)$$

当已知更低阶数的联合概率时，我们也能用它们计算边界和其他更低阶的联合概率。因此，例如

$$p(B = b) = \sum_{B=b} p(B, M)$$

和

$$p(B = b, M = m) = \sum_{B=b,M=m} p(B, M, L)$$

当处理命题变量(有*True*或*False* 值)时，常常利用一个简洁符号,例如，不必再写为p（B =

True、M = *False*）的形式，而将它记为p（B，¬M）——假定没有取反的变量已被实例化为*True*、取反变量被实例化为*False*。只有当上下文清楚地表明正指示一个实例化变量的概率值，而不是那些变量上的概率函数时，才能使用这种缩写符号。

因此，给定一个随机变量集合的完全联合概率函数（如一个表），从理论上讲，就能计算所有的边缘概率和所有的更低阶的联合概率。然而，当我们有一个极大的随机变量集合时，指定所有的联合概率的任务就变得不可处理，更不用说低阶概率了。幸运的是，在大多数应用中，联合概率要满足一定的特殊条件，这些条件使得对它们的说明和计算变得可行。本章的后面将描述这些条件。

19.1.2 条件概率

我们想能够用一些变量值的信息来获得其他变量值的概率。例如，如果搬积木的机器人感知到自己手臂不能移动，它可能想计算（给定那个事实）电池要被充电的概率。和逻辑推理方法相似，这样的计算叫做概率推理。在解释如何执行概率推理前，必须先定义什么是条件概率。

给定V_j，V_i的条件概率函数由$p(V_i|V_j)$表示。对变量V_i和V_j的任何值，可以给出

$$p(V_i|V_j)=\frac{p(V_i,V_j)}{p(V_j)}$$

其中$p(V_i,V_j)$是V_i和V_j的联合概率，$p(V_j)$是V_j的边缘概率。从这个表达式，我们也能按照条件概率表示一个联合概率

$$p(V_i,V_j)=p(V_i|V_j)p(V_j)$$

回到搬积木的例子，给定条件手臂不能移动，我们能计算电池被充电的概率

$$p(\mathrm{B}=True|\mathrm{M}=False)=\frac{p(\mathrm{B}=True,\mathrm{M}=False)}{p(\mathrm{M}=False)}$$

这个表达式的分子分母都能用前面解释的联合概率的求和计算得到。

用一个概率的频率的解释，可以帮助我们比较容易地理解条件概率。在这样一个解释中，例如p（M=*False*）是手臂不能移动的次数和总的尝试次数的比率（在某个想像的实验中执行无限次）。因此，给定手臂不能移动，电池被充电的概率就是手臂不动、电池充电的次数除以手臂不动的次数的结果。因此，一个条件概率是对一个联合概率的规范化。

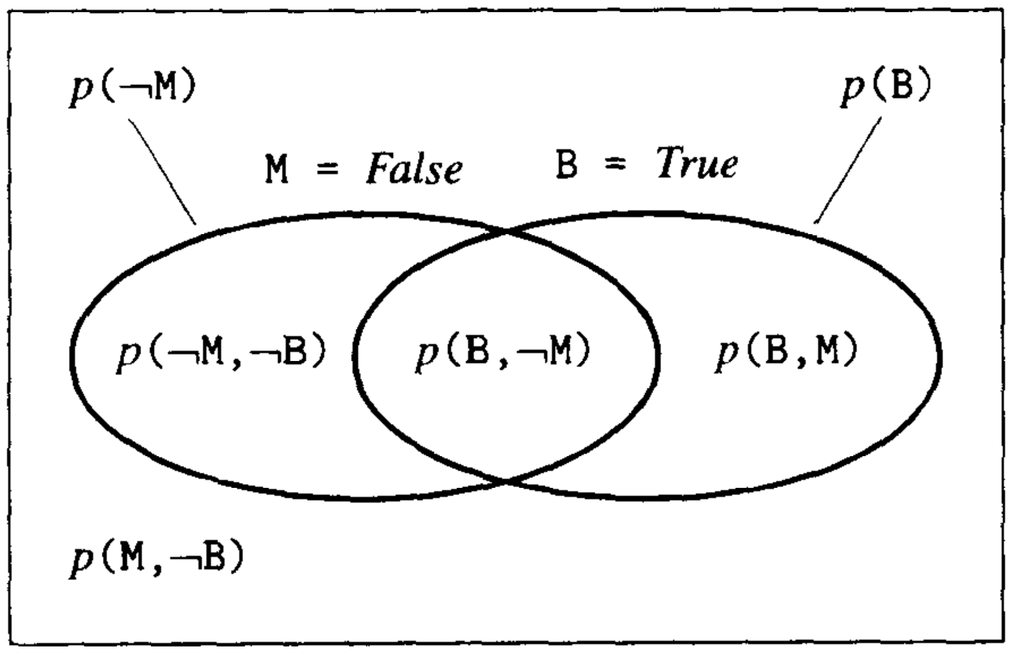
如图19-1所示的Venn[⊖]图有助于说明联合概率和条件概率（对少量的变量）。在那个图中，显示了两个重叠的椭圆区域，一个表示手臂不能移动的几率（M=*False*），一个表示电池被充电的几率（B=*True*）。每一个区域都和相应的（边缘）概率成比例，它们用简化符号标在图中，两个椭圆外部的区域对应手臂能动、电池没充电（p(M=*True*, B=*False*)）的几率。

尤其要注意椭圆的三个独立的不相交的部分，它们分别对应手臂不动电池没电、手臂不动电池有电和手臂能动电池有电的组合几率。这些分开的部分的每个区域与图中相应的联合概率成比例。我们从联合概率计算边缘概率的方法显然是源于图中p（B）=p（B，M）+p（B，¬M）的事实。

我们也有几个变量基于另外几个变量的组合条件概率　例如，（用简写符号）

⊖ John Venn 是一个英国逻辑学家[Venn 1880]。

$$p(\neg G, B|\neg M, L) = \frac{p(\neg G, B, \neg M, L)}{p(\neg M, L)}$$



$$p(B|\neg M) = p(B,\neg M)/p(\neg M)$$

图19-1　一个Venn图

在计算任何条件概率时，出现在计算中的联合概率和边缘概率能从前面描述的包含所有必需变量的任何完全联合概率集中计算得到。

我们也能按照一个条件概率链表达一个联合概率。例如

$$p(B, L, G, M) = p(B|L, G, M)p(L|G, M)p(G|M)p(M)$$

这个链规则的一般形式是

$$p(V_1, V_2, \ldots, V_k) = \prod_{i=1}^{k} p(V_i|V_{i-1}, \ldots, V_1)$$

链规则表达式依赖于我们选择对V_i排序的方式。不同的排序给出不同的表达式，但对变量值的相同集合它们都有相同的值。

由于在一个联合概率函数中变量排序的方式并不重要（只要跟踪谁是谁就行了），我们能写出：

$$p(V_i, V_j) = p(V_i|V_j)p(V_j) = p(V_j|V_i)p(V_i) = p(V_j, V_i)$$

注意到

$$p(V_i|V_j) = \frac{p(V_j|V_i)p(V_i)}{p(V_j)}$$

后面这个等式是非常重要的，叫做贝叶斯法则[⊖]。

下面介绍一个最后的符号约定。当有一个变量集合的联合概率或一个变量集合的条件概率时，使用集合符号将很方便。因此$p(\mathcal{V})$有时被用作$p(V_1, V_2, \cdots, V_k)$的一个缩写，其中$\mathcal{V}=\{V_1, V_2, \cdots, V_k\}$。同样地，我们可以用$p(\mathcal{V}|\mathcal{V}_j)$，其中$\mathcal{V}_j$也是一个变量集合。如果变量$(V_1, V_2, \cdots, V_k)$分别有值$v_1, v_2, \cdots, v_k$，我们用表达式$\mathcal{V}=\boldsymbol{v}$表示这个事实，$\mathcal{V}$和$\boldsymbol{v}$都是有序列表。

⊖ 贝叶斯法则是Reverend Thomas Bayas [Bayes 1763]首次用公式表示的。

19.2 概率推理

19.2.1 一个一般的方法

概率推理的一般情景置是：我们有命题变量V_1, V_2,…, V_k的一个集合 $\mathcal{V}$，并给定了 $\mathcal{V}$的子集ε中的变量的某些值ε=e（*True*或*False*）作为证据。在agent应用中，这些“给定”的变量通常有由感知过程决定的值。我们希望计算条件概率p（V_i=v_i|ε=e），即给定证据时变量V_i的值为v_i条件概率。我们把这个过程叫概率推理。

由于V_i有值*True*或*False*,故我们对两个条件概率感兴趣，它们是p（V_i=*True*|ε=e）和p(V_i=*False*|ε = e)。当然，我们只要计算它们中的一个就行了，因为p（V_i=*True*|ε =e）+p(V_i=*False*|ε=e) =1，不管ε为何值。用“笨”方法说明p（V_i=*True*|ε =e）的计算。用条件概率的定义，我们有

$$p(V_i = True|\mathcal{E} = \mathbf{e}) = \frac{p(V_i = True, \mathcal{E} = \mathbf{e})}{p(\mathcal{E} = \mathbf{e})}$$

其中，p（V_i=*True*，ε= e）通过使用从高阶联合概率计算低阶联合概率的方法获得：

$$p(V_i = True, \mathcal{E} = \mathbf{e}) = \sum_{V_i=True, \mathcal{E}=\mathbf{e}} p(V_1, \ldots, V_k)$$

其中V_i,i =1,…，k构成了命题变量集合。即，对 V_i=*True*，证据变量有它们的给定值。对所有的联合概率值求和。p（ε=e）的计算能用同样的方式进行，然而像下一个例子演示的一样，它不需要明确地计算。

作为一个例子，假如我们有下面给出的联合概率:

p(P,Q,R) = 0.3

p(P,Q,¬R) = 0.2

p(P,¬Q,R) = 0.2

p(P,¬Q,¬R) = 0.1

p(¬P,Q,R) = 0.05

p(¬P,Q,¬R) = 0.1

p(¬P,¬Q,R) = 0.05

p(¬P,¬Q,¬R) = 0.0

¬R作为证据，我们希望计算p（Q|¬R）。用刚刚给定的过程，我们计算

$$p(Q|\neg R) = \frac{p(Q, \neg R)}{p(\neg R)} = \frac{[p(P, Q, \neg R) + p(\neg P, Q, \neg R)]}{p(\neg R)}$$

$$= \frac{(0.2 + 0.1)}{p(\neg R)} = \frac{0.3}{p(\neg R)}$$

现在我们既可直接计算边缘p（¬R），也可（像通常所做的一样）用刚刚使用的相同方法计算p(¬Q|¬R)——通过利用p(Q|¬R) + p(¬Q|¬R) = 1来避免计算p(¬R)。

下面用后一种方式进行：

$$p(\neg Q|\neg R) = \frac{p(\neg Q, \neg R)}{p(\neg R)} = \frac{[p(P, \neg Q, \neg R) + p(\neg P, \neg Q, \neg R)]}{p(\neg R)}$$

$$= \frac{(0.1 + 0.0)}{p(\neg R)} = \frac{0.1}{p(\neg R)}$$

由于这两个量的和为1，我们得到$p(Q \mid \neg R)=0.75$。

一般地讲，使用这个方法的概率推理是难处理的，因为在有k个变量的情况下执行它，我们需要联合概率$p(V_1, V_2, \cdots, V_k)$的2^k个值的一个显式列表。对很多问题，即使我们知道这样一个列表也不能写出它（一般不这样做）。

考虑到这种难处理性，我们可能要问："人是如何对不确定信息有效地推理的？"Pearl[Pearl1986,Pearl 1988,Pearl 1990]推测人类通过一个特殊的方式把一个领域的知识公式化来做推理。这种方式能极大地简化一定变量在给定证据下的条件概率的计算。这些有效的知识公式化涉及到各种变量中的条件独立性——马上要讲的一个主题。

19.2.2　条件独立

给定变量集合$\mathcal{V}$，如果$p(V \mid \mathcal{V}_i, \mathcal{V}_j) = p(V \mid \mathcal{V}_j)$，那么我们就说变量$V$条件独立于变量集$\mathcal{V}_i$，用符号$I(V, \mathcal{V}_i \mid \mathcal{V}_j)$阐述这个事实。条件独立后面的直觉知识是如果$I(V, \mathcal{V}_i \mid \mathcal{V})$，那么$\mathcal{V}_i$不会告诉我们比我们通过$\mathcal{V}_j$知道的任何更多的东西。对$V$而言，如果我们知道$\mathcal{V}_j$，可以忽略$\mathcal{V}_i$。在举积木的例子中，这应该是合理的：如果我们已知（通过其他的一些方式）电池有电（B=*True*），那么在手臂移动的范围内，我们不需要有关G的（量规指示电池有电）明确知识。即$p(M \mid B, G) = p(M \mid B)$。

给定一个集合$\mathcal{V}$，如果一个变量V_i是条件独立于另一个变量V_j，则有（按照定义）$p(V_i \mid V_j, \mathcal{V}) = p(V_i \mid \mathcal{V})$。根据条件概率的定义，有$p(V_i \mid V_j, \mathcal{V})p(V_j \mid \mathcal{V}) = p(T_i, V_j \mid \mathcal{V})$。对$I(V_i, V_j \mid \mathcal{V})$。组合这两个结果产生：

$$p(V_i, V_j \mid \mathcal{V}) = p(V_i \mid \mathcal{V})p(V_j \mid \mathcal{V})$$

注意到V_i和V_j对称地出现。因此，给定$\mathcal{V}$，说V_i条件独立于V_j，也就是说V_j条件独立于V_i。它足以说明给定$\mathcal{V}$，V_i和V_j是条件独立的。相同的结果可用于集合，即给定$\mathcal{V}$，如果$\mathcal{V}_i$和$\mathcal{V}_j$是条件独立的，那么$p(\mathcal{V}_i, \mathcal{V}_j \mid \mathcal{V}) = p(T_i \mid \mathcal{V})\, p(V_j \mid \mathcal{V})$。如果$\mathcal{V}$是空集，我们简单地说$\mathcal{V}_i$和$\mathcal{V}_j$是独立的。

不失一般性，我们说给定集合$\mathcal{V}$，如果每个变量条件独立于所有其他的变量，那么变量$V_1, \ldots, V_k$是相互条件独立的。由于

$$p(V_1, V_2, \ldots, V_k \mid \mathcal{V}) = \prod_{i=1}^{k} p(V_i \mid V_{i-1}, \ldots, V_1, \mathcal{V})$$

且每个V_i是条件独立于其他给定的　，于是我们有

$$p(V_1, V_2, \ldots, V_k \mid \mathcal{V}) = \prod_{i=1}^{k} p(V_i \mid \mathcal{V})$$

当$\mathcal{V}$为空时，我们有

$$p(V_1, V_2, \ldots, V_k) = p(V_1)p(V_2) \cdots p(V_k)$$

故我们说变量是元条件独立的。

条件独立性能用贝叶斯网（也叫信念网）结构方便地表示。这些结构对概率推理是非常有

用的。用贝叶斯网表示的条件独立能大量地节约概率推理计算。

19.3 贝叶斯网

一个贝叶斯网是一个有向无环图（DAG），它的节点用随机变量标识。一个贝叶斯网规定图中的每个节点V_i条件独立于由V_i的父节点给定的V_i的非后代节点构成的任何节点子集。也就是说，假设$A(V_i)$是图中非V_i后代节点的任何节点集合，设$\mathcal{P}(V_i)$是图中V_i的直接双亲。图仅仅是陈述对图中的所有V_i，$I(V_i, A(V_i)|\mathcal{P}(V_i))$的一种方式。$I(V_i, A(V_i)|\mathcal{P}(V_i))$的意思是$p(V_i|A(V_i), \mathcal{P}(V_i)) = p(V_i|\mathcal{P}(V_i))$。

假设$V_1, V_2, \cdots, V_k$是贝叶斯网中的节点，给定由网络假设的条件独立性，我们能写出网中所有节点的联合概率如下：

$$p(V_1, V_2, \ldots, V_k) = \prod_{i=1}^{k} p(V_i|\mathcal{P}(V_i))$$

这个表达式能用一个直接的方式推导出，利用与贝叶斯网DAG蕴含的部分序一致的链规则序，将条件独立性应用于链规则表达式，可计算所有变量的联合概率。

贝叶斯网有时叫做因果网，因为可以将连接节点的弧认为是表达了直接的因果关系。人类专家常常能把原因和结果用一种方式联系起来，这种方式显示了继承的条件独立性，这也能用得到的贝叶斯网描绘。使用直觉因果概念来构建贝叶斯网，通常可以使其能实现内含的条件独立性假设更为合适。一位研究者说过[Heckerman 1996,p.14]："……为了构造一个给定变量集的贝叶斯网，我们从原因变量向直接结果画弧。在几乎所有的情况下，这样做会生成一个贝叶斯网[它的条件独立性蕴含是精确的]。"

用举积木的例子演示一个贝叶斯网的构造。我们首先想像这个例子的"第一个原因"，即和命题"电池被充电"（B）和"积木是可举起来的"（L）相对应的变量。B和L对M（"手臂移动"）有一个因果影响，B对G（"量规指示电池被充电了"）也有因果影响。因此，我们将为这个问题画一个如图19-2所示的贝叶斯网。注意，除了其他的，网络陈述了$p(M|G, B, L) = p(M|B, L)$。如果在网络中有另外的节点U（意指积木被举起），将不会有$p(M|G, B, L, U) = p(M|B, L)$，因为U是M的一个后继（积木被拿起影响了手臂移动的概率，除此之外，还有其他能影响吗？）。网中所有节点的联合概率函数的表达式都在图中给出。

我们看到，为了计算给定贝叶斯网的联合概率值，如图19-2所示，我们需要知道恰恰以它的父节点为条件的每个节点的条件概率函数。对没有父节点的节点，概率不以其他节点为条件，这些叫做这些变量的先验概率。因此，一个随机变量集合的概率的一个完整说明涉及到这些变量的一个贝叶斯网及网中每个变量的条件概率表（CPT）。

举积木例子中联合概率的贝叶斯网公式应该与由使用链规则：

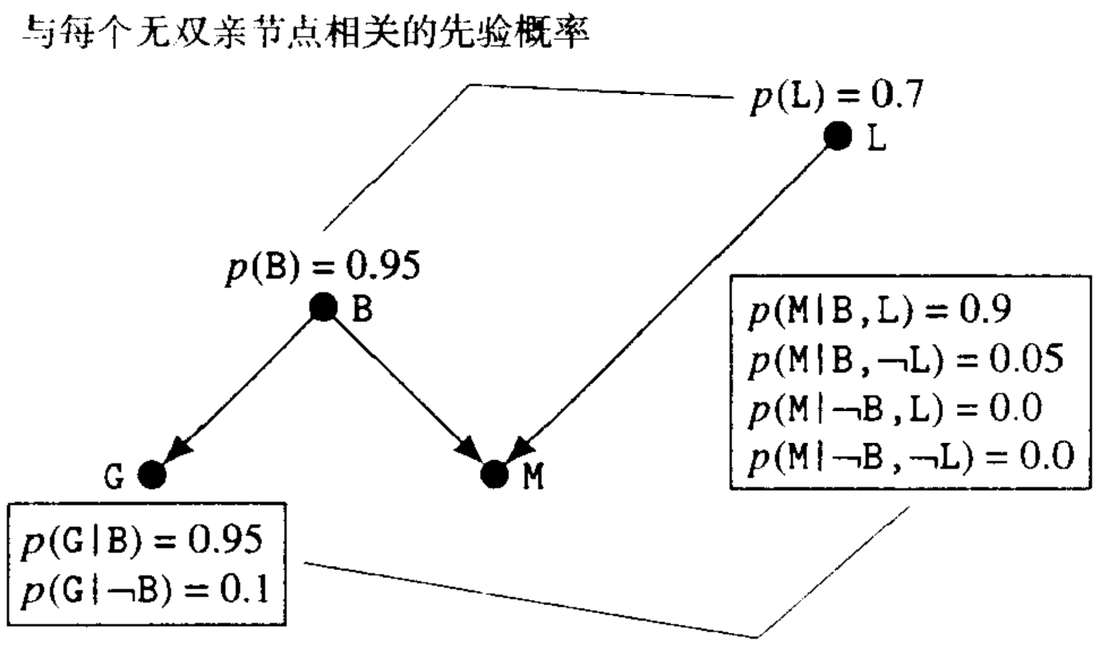


图19-2 一个贝叶斯网

$$p(G, M, B, L) = p(G|B, M, L)p(M|B, L)p(B|L)p(L)$$

获得的一个相似公式（假定没有任何条件独立）进行比较。注意到贝叶斯网公式更简单。没有贝叶斯网规定的条件独立性，对这个例子的所有4个变量的一个联合概率的规范涉及到指定16个独立的联合概率（实际仅需要15个，因为它们的和必须为1）。从图19-2可以明显看到，由贝叶斯网所做的假设要求我们仅指定8个概率。当在领域中的变量中有几个条件独立时，从贝叶斯网计算的联合概率表达式要求的概率规范比没有这些独立时的还要少。这种减少有时会使难处理的问题变得可处理。

19.4　贝叶斯网的推理模式

在贝叶斯网中有三种重要的推理模式。为了解释它们，继续我们的例子。

- 因果推理或由上向下推理。给定积木是可举起的，假如我们想计算手臂能移动的概率 $p(M|L)$。由于积木可举起是手臂能移动的原因之一。我们说这个计算是因果推理的一个例子。L称做用于推理的证据，M叫询问节点。下面说明我们在这种情况下如何执行推理：首先，我们把$p(M|L)$（一个边缘概率）扩展成两个联合概率的和（因为我们想提及M的另一个双亲B）：

$$p(M|L) = p(M, B|L) + p(M, \neg B|L)$$

接下来，我们希望M以另一双亲为条件，因此我们用一个链规则形式得到

$$p(M|L) = p(M|B, L)p(B|L) + p(M|\neg B, L)p(\neg B|L)$$

但是$p(B|L) = p(B)$（从网结构而来，注意B没有任何双亲）。同样地，

$p(\neg B|L) = p(\neg B)$

因此，$p(M|L) = p(M|B, L)p(B) + p(M|\neg B, L)p(\neg B)$。由于所有的这些数值与网络一同给出，我们能计算

$$p(M|L) = 0.855$$

我们在这个例子中执行的操作值得注意，因为它们能被一般化为如我们后面看到的更复杂的因果推理。主要操作如下：

a) 按照给定证据的V和它的所有双亲（它们不是证据）的联合概率，重新表达给定证据的询问节点V的所求条件概率。

b) 回到以所有双亲为条件的 V的概率，重新表达这个联合概率。

- 诊断推理或自底向上推理。现在我们计算$p(\neg L|\neg M)$，即给定手臂未移动时积木不可举起的概率。其中询问和证据的角色刚好和它们在上一个例子中的相反。由于我们用一个结果（或症状）来推理一个起因，我们称这类推理叫诊断推理。

$$p(\neg L|\neg M) = \frac{p(\neg M|\neg L)p(\neg L)}{p(\neg M)} \text{（贝叶斯规则）}$$

现在我们计算$p(\neg M|\neg L) = 0.9525$（用因果推理），并计算

$$p(\neg L|\neg M) = \frac{0.9525 \times 0.3}{p(\neg M)} = \frac{0.28575}{p(\neg M)}$$

同样地，$p(L|\neg M) = \frac{p(\neg M|L)p(L)}{p(\neg M)} = \frac{0.0595 \times 0.7}{p(\neg M)} = \frac{0.03665}{p(\neg M)}$。

由于这两个表达的和必须为1，得到$p(\neg L|\neg M)=0.88632$。

用在这个简单例子中的诊断推理计算也能被一般化。主要步骤是使用贝叶斯规则把问题转化成因果推理。

- 辩解。如果我们的证据仅仅是¬ M（手臂不能移动），像刚做的那样，我们能计算积木不能举起的概率。但是如果也给定¬ B（电池没被充电），那么¬ L应该变得不确定。在这种情况下，我们说¬ B解释¬ M，使¬ L不确定。这类推理使用嵌入在一个自底而上或诊断推理中的自顶而下或因果推理。

$$p(\neg L|\neg B,\neg M)=\frac{p(\neg M,\neg B|\neg L)p(\neg L)}{p(\neg B,\neg M)}\text{（贝叶斯规则）}$$

$$=\frac{p(\neg M|\neg B,\neg L)p(\neg B|\neg L)p(\neg L)}{p(\neg B,\neg M)}\text{（条件概率定义）}$$

$$=\frac{p(\neg M|\neg B,\neg L)p(\neg B)p(\neg L)}{p(\neg B,\neg M)}\text{（贝叶斯网结构）}$$

从这个表达式，使用网中给定的概率，用普通方式求解$p(\neg B,\neg M)$，我们可以计算$p(\neg L|\neg B,\neg M)=0.030$。像预期的一样，它比前面计算的$p(\neg L|\neg M)$更小。再次注意贝叶斯法则的使用，它是辩解过程中的一个重要步骤。

19.5 不确定证据

当证据ε本身不确定时，表达式$p(V|\varepsilon)$，V是一个询问节点，不会给出正确的概率。在贝叶斯网计算中，为了“给定”证据节点，必须确定它们表示的命题的真假。我们能通过让每个证据节点（我们不能确定的）有一个确定的子节点来获得那个要求。因此，在上一个例子中（辩解），假定机器人对它的手臂不能移动不确定，它可能有一个有点不可靠的关节传感器。在这种情况下，证据能被一个节点M'提供，M'代表命题“手臂传感器说手臂可以移动”。依赖它的读数，我们能确定命题是真还是假。然后用贝叶斯网计算$p(\neg L|\neg B,\neg M')$，而不是$p(\neg L|\neg B,\neg M)$。当然，网络将需要$p(M'|M)$和$p(M'|\neg M)$的值，它们描述了传感器的可靠性。

注意图19-2中的网络已提供了事实——关于电池是否被充电我们是不确定的。节点B有一个子节点G，我们通过概率$p(G|B)$和$p(G|\neg B)$表示量规的可信度，留给你（也许在进一步阅读后）去计算$p(\neg L|\neg G,\neg M')$。

即使通过一个贝叶斯网给出的简化，用于从一个联合概率计算各种条件概率的强制方法对大的网络来讲，一般仍是难处理的。它的最坏时间复杂度是命题变量的指数。幸运的是，有几个简便方法可用于计算特殊网络的条件概率。下面，先表达贝叶斯网中关于条件独立的另一个结果，然后再考虑上述简便方法。

19.6 D分离

一个贝叶斯网比那些仅涉及个节点双亲的网蕴含了更多的条件独立，例如，在图19-2中，$p(M|G,B)=p(M|B)$，即，给定B，M条件独立于G（即使没有给出M的双亲）。从直觉上讲，在图19-2的网中，G的知识（结果）能影响B的知识（起因），B会影响M的知识（另一个结果）。但是如果给定原因B，G并不能告诉我们有关M更多的事情。在这种情况下，我们说B d分离（依赖方向的分离）G和M。

其他的这种条件独立存在于贝叶斯网中。在这里仅介绍它们，你可以参考[Pearl 1988, pp. 117~122]寻找证明。

如果对贝叶斯网中的节点V_i和V_j之间的每个无向路径，在路径上有某个节点V_b，它有如下的三个属性之一（见图19-3），就说节点V_i和V_j条件独立于给定的节点集ε（即$I(V_i, V_j|\varepsilon)$）。三个属性是：

1) V_b在ε中，且路径上的两条弧都从V_b开始。

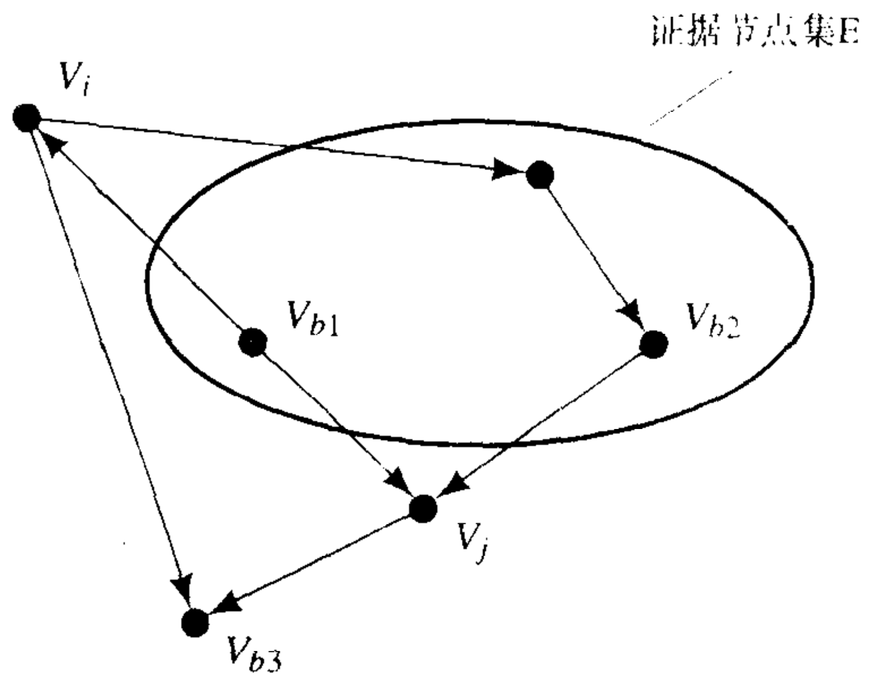
2) V_b在ε中，路径上的一条弧以V_b为头，另一个以V_b为尾。

3) V_b和它的任何后继都不在ε中，路径上的两条弧都以V_b为头。

给定ε，当这些条件中的任何一个占据一条路径时，我们说V_b阻塞那条路径。注意，在这个结果中引用的路径是无向路径，即路径忽略了弧方向。如果V_i和V_j之间的所有路径被阻塞，我们说ε d分离V_i和V_j（依赖方向的分离）且得出结论：V_i和V_j条件独立于给定的ε。图19-2中d分离产生的其他条件独立的例子如下：

- I(G,L|B)，因为根据规则1，给定B阻塞了G和L之间的惟一的路径。根据规则3，给定B，M也阻塞了这个路径，因为M不是证据集的一个成员。
- I(G, L)和I(B, L)，因为按规则了3，给定空的证据集合，M阻塞了G和L之间、B和L之间的（惟一）路径（M不是空的证据集的一个成员）。

然而，注意，B和L不是条件独立于给定的M的。因为虽然M在B和L的路径上，但这个路径上的两条弧都指向M，且M在证据集合中——因此在这种情况下，M没有阻塞路径。



由于V_i到V_j之间的所有三条路径都被阻塞，给定证据节点，Vi 独立于V_j。阻塞的节点为：

a) V_{b1}是一证据节点，两条弧都用V_{b1}开始。

b)V_{b2}是一证据节点，一条弧以V_{b2}为头，另一条弧以V_{b2}为尾。

c) V_{b3}及其任一后代都不是证据节点，两条弧都以V_{b3}为头。

图19-3　通过阻塞节点的条件独立

*d*分离的概念也能应用到集合。给定ε，如果两个节点集V_i和V_j被ε d分离，则它们是条件独立的。给定ε，如果V_i中的所有节点和V_j中的所有节点之间的每条无向路径被阻塞，则V_i和V_j被ε d分离。

即使使用了*d*分离，一般地讲，在贝叶斯网中，概率推理仍是NP难题[Copper 1990]。然而，有些简化能在一个叫polytree的重要网络分类中使用。一个polytree网是一个DAG，在该DAG的任何两个节点之间，顺着弧的每一个方向只有一条路径。例如，图19-2的网络就是一个polytree。用一个扩展的符号例子来说明在polytree中，是如何执行概率推理的（说明的例子基于由[Russell & Norvig 1995, pp. 447以后]提出的一个算法）。

19.7　在polytree中的概率推理

图19-4的网络是polytree的一个典型例子。在这个网络中，给定一些其他的节点，我们想计算Q的概率。注意，有些节点仅通过Q的双亲连到Q，我们说这些节点在Q的上方。其他的节

点仅通过Q的直接后继（它的孩子）连到Q，我们说这些节点在Q的下方。我们也注意到没有任何路径（除了通过Q连接Q上的一个节点和Q下的一个节点外——因为不这样的话，网络将不是一个polytree）将这些定义和连接属性应用到polytree中的每一个节点！我们的例子将有三种类型：

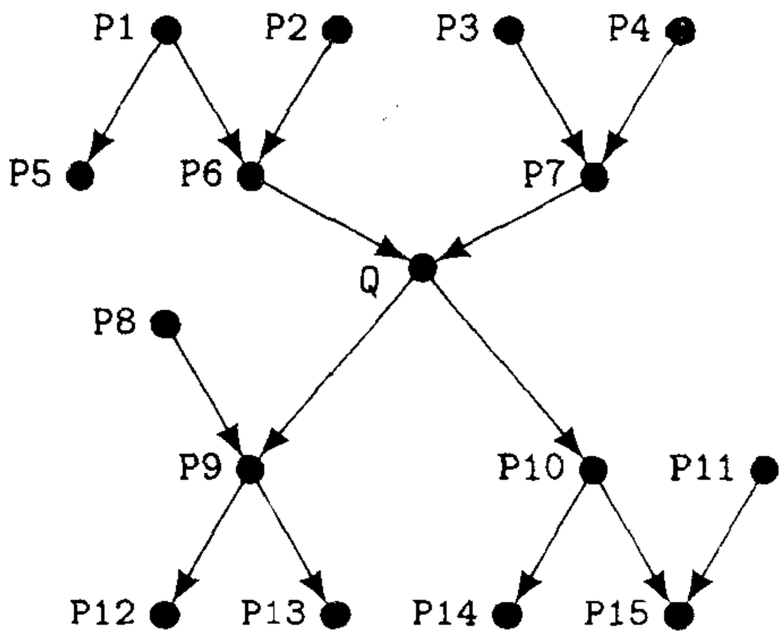


图19-4 一个典型的Polytree

1) 所有的证据节点在Q的上方。作为这种类型的一个典型例子，我们将计算p（Q|P5，P4）。

2) 所有的证据节点在Q的下方。作为这种类型的一个典型例子，我们将计算p（Q|P12，P13，P14，P11）。

3) 在Q的上方和下方均有证据节点。

19.7.1 证据在上方

我们先计算p（Q|P5，P4），其中所有的证据节点均在Q的上方。我们的计算是沿着一个“自底向上”的递归算法执行的。这个算法在给定证据的情况下，计算Q的每个祖先的概率，直到我们到达证据节点或证据节点在那个祖先的下面。算法过程如下：

首先，我们“包括双亲”（Q的）：

$$p(Q|P5, P4) = \sum_{P6,P7} p(Q, P6, P7|P5, P4)$$

(这个求和特殊符号的意思是相加p（Q，P6，P7|P5，P4）的4个形式——原始的、用¬ P6代替P6、用¬ P7代替P7以及两个都代替。)

下面，我们用条件独立的定义产生Q的双亲部分的证据，记为

$$p(Q, P6, P7|P5, P4) = p(Q|P6, P7, P5, P4)p(P6, P7|P5, P4)$$

替换产生

$$p(Q|P5, P4) = \sum_{P6,P7} p(Q|P6, P7, P5, P4)p(P6, P7|P5, P4)$$

现在，因为一个节点条件独立于它的双亲给定的非后继节点，

$$p(Q|P5, P4) = \sum_{P6,P7} p(Q|P6, P7)p(P6, P7|P5, P4)$$

那么，d分离允许我们分割双亲：

$$p(Q|P5, P4) = \sum_{P6,P7} p(Q|P6, P7)p(P6|P5, P4)p(P7|P5, P4)$$

最后，在计算其他的概率中，d分离允许我们忽略一个双亲上的证据：

$$p(Q|P5, P4) = \sum_{P6,P7} p(Q|P6, P7)p(P6|P5)p(P7|P4)$$

它是非常重要的——注意到正被求和的项是：（a）给定的父节点的各个值，询问节点的概率（父节点的概率与贝叶斯网一同给出）；（b）只要给出那个父节点上的部分证据，它是每个父

节点的概率（递归调用我们正在执行的算法）。这些结果直接利用了我们用一个polytree产生的事实。

这个相同的过程被递归应用，直到最终到达一个有一证据节点做为父节点的节点，或到达没有父节点的节点（不是自身证据节点的节点）。在计算p（P7 | P4）中，我们有这两种情况的第一种，证据节点P4是询问节点P7的一个父节点。此时，“包括双亲”的步骤比较简单，因为其中之一已经被包括了。因为p（P7，P3 | P4）=p（P7 | P3，P4）p（P3 | P4），我们能写出

$$p(\mathrm{P7}|\mathrm{P4}) = \sum_{\mathrm{P3}} p(\mathrm{P7}|\mathrm{P3}, \mathrm{P4})p(\mathrm{P3}|\mathrm{P4}) = \sum_{\mathrm{P3}} p(\mathrm{P7}|\mathrm{P3}, \mathrm{P4})p(\mathrm{P3})$$

（因为I（P3，P4），故有最后一步。）这个求和中的所有项由贝叶斯网给出，因此过程连同这个例子的分枝一同结束。

在计算p（P6 | P5）时，得到

$$p(\mathrm{P6}|\mathrm{P5}) = \sum_{\mathrm{P1},\mathrm{P2}} p(\mathrm{P6}|\mathrm{P1}, \mathrm{P2})p(\mathrm{P1}|\mathrm{P5})p(\mathrm{P2})$$

下一步必须计算p（P1 | P5），注意到证据节点不在询问节点的“上方”，而是在“下面”，我们不能再用这个递归过程，而必须用“证据在下方”的过程——将要描述。在这个例子中，我们仅仅用贝叶斯法则获得 $p(\mathrm{P1}|\mathrm{P5}) = \frac{p(\mathrm{P5}|\mathrm{P1})p(\mathrm{P1})}{p(\mathrm{P5})}$。现在计算$p$（P6 | P5）需要的所有量都由贝叶斯网给出了。我们能集成所有的这些结果（执行所有的求和）得到p（Q | P5，P4）的最终结果

19.7.2　证据在下方

接着，我们计算p（Q | P12，P13，P14，P11），其中，所有的证据节点都在Q的下方，我们的计算再次沿着一个递归算法执行。它按如下进行：在顶级，我们用贝叶斯规则写出

$$\begin{aligned} p(\mathrm{Q}|\mathrm{P12}, \mathrm{P13}, \mathrm{P14}, \mathrm{P11}) &= \frac{p(\mathrm{P12}, \mathrm{P13}, \mathrm{P14}, \mathrm{P11}|\mathrm{Q})p(\mathrm{Q})}{p(\mathrm{P12}, \mathrm{P13}, \mathrm{P14}, \mathrm{P11})} \\ &= kp(\mathrm{P12}, \mathrm{P13}, \mathrm{P14}, \mathrm{P11}|\mathrm{Q})p(\mathrm{Q}) \end{aligned}$$

其中 $k = \frac{1}{p(\mathrm{P12},\mathrm{P13},\mathrm{P14},\mathrm{P11})}$ 是一个在后面要按与前面的例子同样的方式计算的标准化因子。用d分离，I（{P12，P13}，{P14，P11} | Q），产生

$$p(\mathrm{Q}|\mathrm{P12}, \mathrm{P13}, \mathrm{P14}, \mathrm{P11}) = kp(\mathrm{P12}, \mathrm{P13}|\mathrm{Q})p(\mathrm{P14}, \mathrm{P11}|\mathrm{Q})p(\mathrm{Q})$$

注意我们已把集合{P12，P13，P14，P11}分割成对应Q的两个孩子的子集。p（P12，P13 | Q）和p（P14，P11 | Q）的每一项包括给定其上的一个单一证据节点，计算一个询问节点集合的概率，因此，我们能使用类似于前面的算法。因为只有一个证据节点，故使用一个自顶而下的递归算法，而不用前面所用的自底而上算法。

通过首先计算p（P12，P13 | Q），说明自顶而下方案是如何进行的。关键步骤是包括Q的单一孩子P9，它在询问节点集{P12，P13}的上方。注意，根据条件独立的定义，p（P12，P13，P9 | Q）= p（P12，P13 | P9，Q）p（P9 | Q）。

那么，

$$p(\mathrm{P12}, \mathrm{P13}|\mathrm{Q}) = \sum_{\mathrm{P9}} p(\mathrm{P12}, \mathrm{P13}|\mathrm{P9}, \mathrm{Q})p(\mathrm{P9}|\mathrm{Q})$$

现在，用d分离，I({P12,P13},Q|P9)，故

$$p(\mathrm{P12},\mathrm{P13}|\mathrm{Q})=\sum_{\mathrm{P9}}p(\mathrm{P12},\mathrm{P13}|\mathrm{P9})p(\mathrm{P9}|\mathrm{Q})$$

这个求和中的项p(P9|Q)的计算涉及到P9的所有父节点：

$$p(\mathrm{P9}|\mathrm{Q})=\sum_{\mathrm{P8}}p(\mathrm{P9}|\mathrm{P8},\mathrm{Q})p(\mathrm{P8})$$

p(P9|P8，Q)由网络给出。另一项p(P12，P13|P9)是对相同的自顶而下的过程的递归调用，该过程在给定询问节点集上的一个单一证据节点的情况下，计算它们的概率。在这种情况下，递归调用在一步后终止，因为P9的孩子是证据节点。由于P12和P13独立于给定的P9，因此有p(P12，P13|P9) = p(P12|P9)p(P13|P9)。这两个概率都由网络给出。

把自顶而下过程应用到p(P14，P11|Q)，产生

$$p(\mathrm{P14},\mathrm{P11}|\mathrm{Q})=\sum_{\mathrm{P10}}p(\mathrm{P14},\mathrm{P11}|\mathrm{P10})p(\mathrm{P10}|\mathrm{Q})$$

那么由于I(P14，P11|P10)，

$$p(\mathrm{P14},\mathrm{P11}|\mathrm{Q})=\sum_{\mathrm{P10}}p(\mathrm{P14}|\mathrm{P10}),p(\mathrm{P11}|\mathrm{P10})p(\mathrm{P10}|\mathrm{Q})$$

仅仅这个结果的中间项不是由网络直接给出。我们再次用自顶向下过程计算该项：

$$p(\mathrm{P11}|\mathrm{P10})=\sum_{\mathrm{P15}}p(\mathrm{P11}|\mathrm{P15},\mathrm{P10})p(\mathrm{P15}|\mathrm{P10})$$

其中

$$p(\mathrm{P15}|\mathrm{P10})=\sum_{\mathrm{P11}}p(\mathrm{P15}|\mathrm{P10},\mathrm{P11})p(\mathrm{P11})$$（为什么）？

但在p(P11|P15，P10)中，询问节点P11在证据节点的上方，因此我们必须再次应用这个过程的顶级（用贝叶斯规则）：

$$p(\mathrm{P11}|\mathrm{P15},\mathrm{P10})=\frac{p(\mathrm{P15},\mathrm{P10}|\mathrm{P11})p(\mathrm{P11})}{p(\mathrm{P15},\mathrm{P10})}=k_1p(\mathrm{P15},\mathrm{P10}|\mathrm{P11})p(\mathrm{P11})$$

其中$k_1=\frac{1}{p(\mathrm{P15},\mathrm{P10})}$，p(P11)由网络直接给出；因为P10和P11是独立的，算法终止于

$$p(\mathrm{P15},\mathrm{P10}|\mathrm{P11})=p(\mathrm{P15}|\mathrm{P10},\mathrm{P11})p(\mathrm{P10}|\mathrm{P11})=p(\mathrm{P15}|\mathrm{P10},\mathrm{P11})p(\mathrm{P10})$$

现在，所有的结果能被收集，可以计算出总和以及k和k_1，以得到p(Q|P12，P13，P14，P11)的最终结果。

证据在上和证据在下算法的复杂度都和网络中节点数（对polytree而言）成线性关系。

19.7.3 证据在上下两方

如果在Q的上方和下方均有证据，如在p(Q|{P5,P4},{P12,P13,P14,P11})中，我们把证据分成上部ε^-和下部ε^-两部分，用一种贝叶斯规则

$$p(\mathrm{Q}|\varepsilon^+,\varepsilon^-)=\frac{p(\varepsilon^-|\mathrm{Q},\varepsilon^+)p(\mathrm{Q}|\varepsilon^+)}{p(\varepsilon^-|\varepsilon^+)}$$

像往常一样，我们把$\frac{1}{p(\varepsilon^-|\varepsilon^+)}=k_2$作为一个标准化因子，并表达为：

$$p(Q|\mathcal{E}^+, \mathcal{E}^-) = k_2 p(\mathcal{E}^-|Q, \mathcal{E}^+) p(Q|\mathcal{E}^+)$$

注意，Q从$\mathcal{E}^+$中d分离$\mathcal{E}^-$，故

$$p(Q|\mathcal{E}^+, \mathcal{E}^-) = k_2 p(\mathcal{E}^-|Q) p(Q|\mathcal{E}^+)$$

注意到在这个结果中，我们计算的第一个概率已作为计算p(Q|ε)的自顶而下的一部分。第二个概率直接用自底而上过程计算。

19.7.4　一个数值例子

在图19-5中，用一个更小的抽象polytree数值化地说明这些方法的使用。我们的目的是计算p(Q|U)。

像平常的诊断推理一样，首先应用贝叶斯法则得到

$$p(Q|U) = kp(U|Q)p(Q), \quad 其中 \quad k = \frac{1}{p(U)}。$$

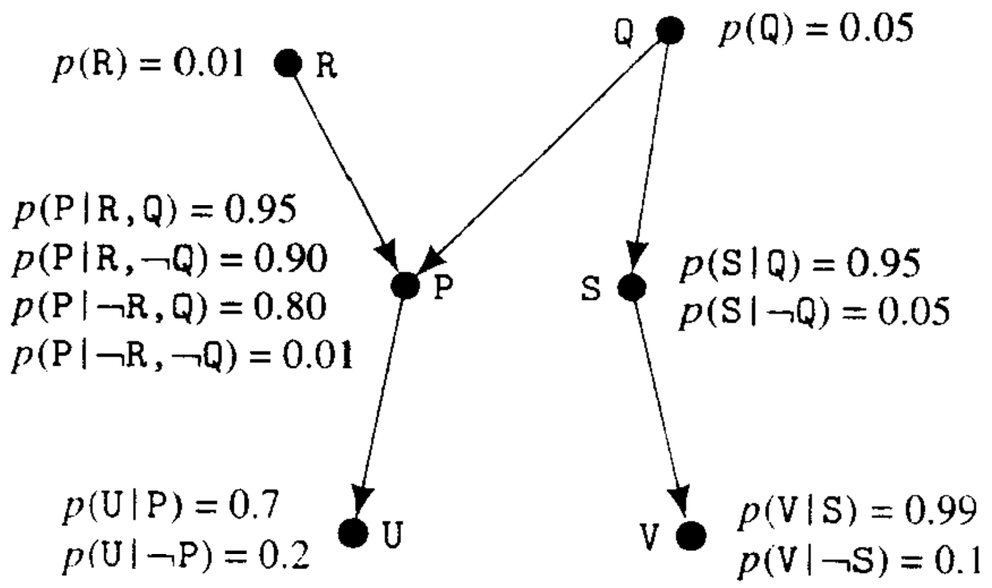


图19-5　一个小polytree

自顶而下算法连续计算

$$p(U|Q) = \sum_P p(U|P)p(P|Q)$$

$$p(P|Q) = \sum_R p(P|R, Q)p(R)$$

$$= p(P|R, Q)p(R) + p(P|\neg R, Q)p(\neg R)$$

$$= 0.95 \times 0.01 + 0.8 \times 0.99 = 0.80, \text{ 因此}$$

$$p(\neg P|Q) = 0.20$$

$$p(U|Q) = p(U|P) \times 0.8 + p(U|\neg P) \times 0.2$$

$$= 0.7 \times 0.8 + 0.2 \times 0.2 = 0.60, \text{ 这样}$$

$$p(Q|U) = k \times 0.6 \times 0.05 = k \times 0.03$$

$$p(\neg Q|U) = kp(U|\neg Q)p(\neg Q)$$

$$p(U|\neg Q) = \sum_P p(U|P)p(P|\neg Q)$$

$$p(P|\neg Q) = \sum_R p(P|R, \neg Q)p(R)$$

$$= p(P|R, \neg Q)p(R) + p(P|\neg R, \neg Q)p(\neg R)$$

$$= 0.90 \times 0.01 + 0.01 \times 0.99 = 0.019, \text{这样}$$

$$p(\neg P|\neg Q) = 0.98$$

$$p(U|\neg Q) = p(U|P) \times 0.019 + p(U|\neg P) \times 0.98$$

$$= 0.7 \times 0.019 + 0.2 \times 0.98 = 0.21, \text{这样}$$

$$p(\neg Q|U) = k \times 0.21 \times 0.95 = k \times 0.20$$

因此，k=4.35，最终

$$p(Q|U) = 4.35 \times 0.03 = 0.13$$

像这个例子的这些计算能被组织以避免重复的子计算，这样做的一个方法是所谓的桶排除法（*bucket elimination*）[Dechter 1996]。

当网络不是一个polytree时，由于在节点之间有多条路径，上述的递归过程将不会终止。已有一些其他的技术被提出来解决这些更复杂的网络。其中之一是Monte Carlo方法(也叫逻辑采样[Henrion 1988])。在这个技术中，用无双亲节点的边缘概率来给那些节点分配随机值（如True或False）。用那些值，它们的后继的CPT被用来分配随机值给这些后继，等等，顺着网络向下。最终，网络中的每个节点都有一个值。这个过程被重复很多次，我们跟踪被分配给节点的所有值。在无限次试验的限制中，节点值将和被网络和它的CPT规定的节点的联合概率相一致。经过大量的试验后，我们能用P和E被分配真值的次数除以E被分配真值的次数评估$p(Q|E)$的值。显然，给定一个证据节点集，能用同样的方法来计算一个询问节点集的联合概率。

另一个方法叫群集（*clustering*）[Lauritzen & Spiegelhalter 1988]，它把网络中的节点用一种方式分组成“超节点”以便超节点图是一个polytree。超节点的可能值是它们的构成节点值的所有组合。然后可以使用polytree算法，但现在对每个超节点有很多CPT——给定所有超节点值的条件概率，以所有父节点值为条件（父节点自己可以是超节点）。

19.8 补充读物和讨论

有几本关于概率的课本可以用来补充本章内容；[Feller 1968]是其中的一本。

一些研究者认为非单调推理能用概率方法最好地处理。例如，参见[Goldszmidt,Morris & Pearl 1990]。

在AI中，关于使用贝叶斯网的概率推理工作开始于[Pearl 1982a, Kim & Pearl 1983]，他们为树和polytree网分别开发了“消息传递”算法。在本章描述的polytree方法是基于[Russell & Norvig 1995,pp.447以后]的。对贝叶斯网的处理只限于离散变量，对连续随机变量也已做了一些工作，参见[Shachter & Kenley 1989]。[Wellman 1990]研究了“定性”网络。

引用了由[Pearl 1984]写的关于概率推理的书。[Neapolitan 1990]是一本关于概率方法在专家系统中应用的课本。[Henrion 1990]是一篇有关贝叶斯网中概率推理的文章。[Jensen 1996]是一本关于贝叶斯网的课本，它以HUGIN系统为特征。[Neal 1991]调查了贝叶斯网和神经网络之间的联系。《Communicaions of the ACM》中关于“AI的不确定性”是由David Heckerman、Michael Wellman和Abe Mamdani编辑的（1995年3月，第38卷，第3期）。

贝叶斯网已用在很多专家系统中。一个典型的例子是PATHFINDER，它帮助病理学者诊断淋巴节疾病[Heckerman 1991, Heckerman & Nathwani 1992]。另一个是用于内科医学的CPCSBN[Pradhan,et al. 1994]，它有448个节点和908个弧，可以和世界上内科医学的一流的诊

断专家相媲美。

除了贝叶斯网之外，还有几个可选的方法可对不确定信息进行推理。用于医疗诊断和治疗建议的MYCIN专家系统使用确定性因素[Shortliffe 1976,Buchanan & Shortliffe 1984]。[Duda,Hart,& Nilsson 1976]在他们的PROSPECTOR专家系统中使用充分性和必要性索引帮助矿物勘探。

其他的方法基于模糊逻辑和“概率理论：[Zadeh 1975,Zadeh 1978,Elkan 1993]以及组合Dempster-Shafer规则[Dempster 1968, Shafer 1979]。[Nilsson 1986]开发了一个“概率逻辑”，引用了概率论和多值逻辑中的相关工作。我认为贝叶斯网的概率推理主宰着用于大多数专家系统应用的其他方法，但是这个说法是有争议的。

当面对不确定性时，人类的行为可能是相当的不一致[Tversky & Kahneman 1982]。因此它不会对工程学提供有用的模型。

[Shafer & Pearl 1990]是关于不确定推理的一本论文集。在关于AI不确定性（UAI）的年会论文集中包含当前的研究进展。《International Journal of Approximate Reasoning》以及其他的AI期刊和AI会议学报也发表重要的论文。

习题

19.1 假定有颜色的小球分别在三个不能区分的盒子$B1$、$B2$和$B3$中，如下所示：

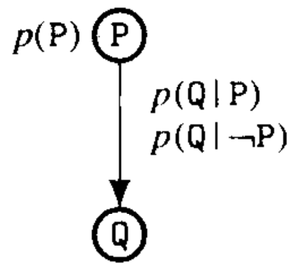
随机选择一个盒子，再从该盒子随机地选一个球，球是红色。被选择的盒子是$B1$、$B2$和$B3$的概率各是什么？解释你的推理。

	B1	B2	B3
红色	2	4	3
白色	3	2	4
蓝色	6	3	3

19.2 考虑下面显示的信念网络。

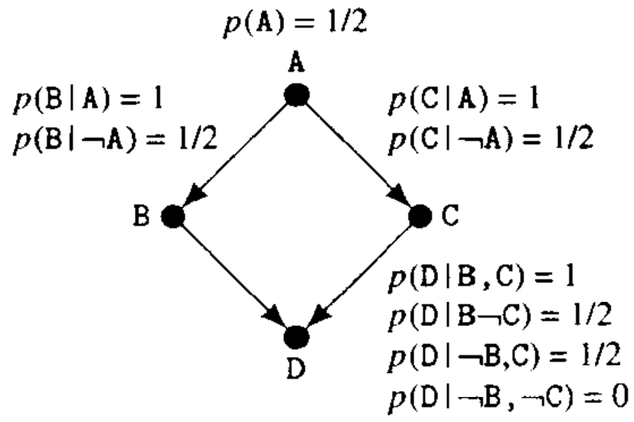
1) 推导出$P \supset Q$的一个概率表达式。

2) $p(P \supset Q)$和$p(Q \mid P)$什么时候相等？



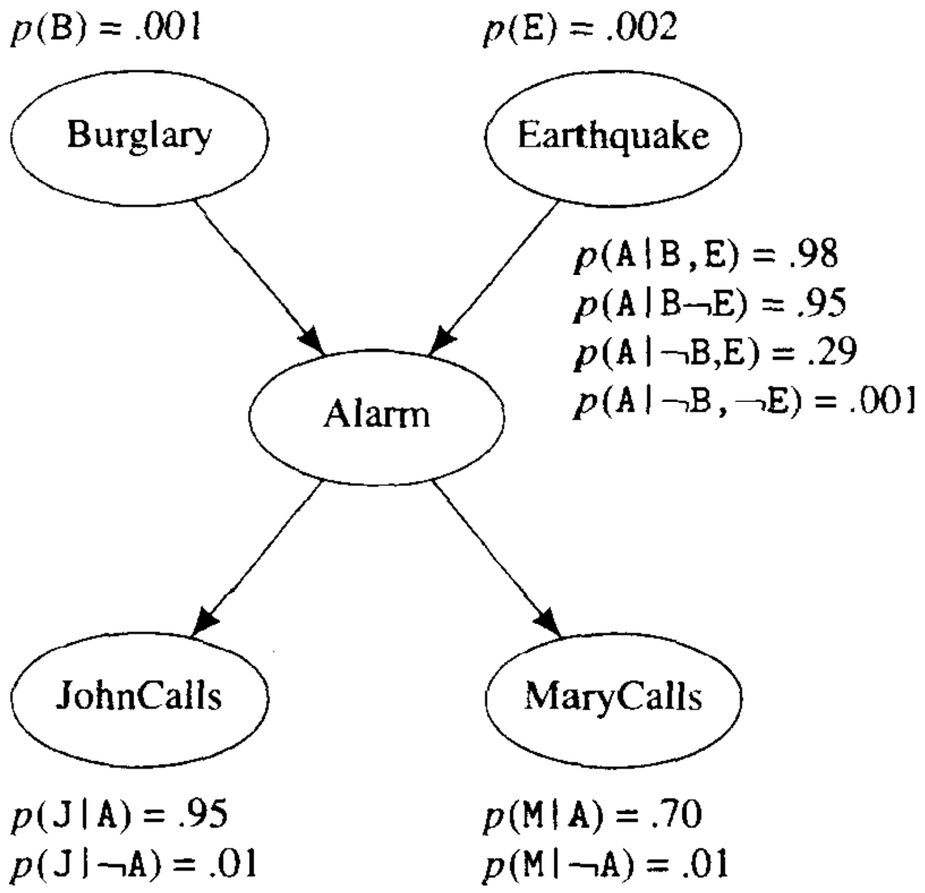
3) 假定不知道该网络的条件概率表，而是知道$p(P)$和$p(P \supset Q)$的值。关于$p(Q)$的值能说些什么？

19.3 一个大学的招生委员会试图决定一个接受的申请人真正合格的概率。相关的概率给在如下的贝叶斯网中。计算$p(A \mid D)$



A=申请人合格
B=申请人有高平均分
C=申请人有优秀的推荐
D=申请通过

19.4 下面显示的信念网络形式化如下的情形：你有一个新的夜盗警铃按装在家里。它预防盗贼相当可靠，且偶尔会对小地震做出反应。你有两个邻居：John和Mary，他们都答应当听到警铃时会叫你。当John听到警铃时，他相当相信它，但有时却会混淆电话铃和警铃而叫你。另一方面，Mary喜欢大声的音乐，结果有时错过了警铃。



练习一下使用信念网络定义的联合概率工作的能力，计算在同时有地震和盗贼的情况下，John和Mary都没有呼叫的概率。即：计算$p(\neg J, \neg M, B, E)$。

19.5 在一个遥远的星球中。90%的出租车是绿色的，10%的是蓝色的。一个与一辆出租车有关的车祸发生了；我们假定绿车和蓝车的事故率相等。一个法庭处理这个事故，现场的一个记者说："出事的出租车是蓝色的"。记者常常是可信的；实际上，他的陈述在80%的时间内是正确的。即，如果出事的出租车确实是蓝色（或绿色的），我们的目击者说：蓝色（或绿色）的概率是0.8。给定记者的陈述，出事出租车是蓝色的概率是多少？

19.6 变戏法机器人Orville，当它的电池电压低时，掉球是很常见的。在以前的测试中，已经决定了当电池低压时，它掉球的概率是0.9。然而当电池电压不低时，掉球的概率仅仅是0.01。电池不久前刚被充电，我们最好的估计（在看Orville的最新戏法记录之前）是电池低压时的几率是10比1。一个有不太可靠视觉系统的机器人观察者，记录orville的掉球。观察者的可信度由下面的概率给出：

p（观察者说Orville掉球了|Orville掉球了）= 0.9

p（观察者说Orville掉球了|Orville没有掉球）= 0.2

画出贝叶斯网，给定观察者的记录以计算电池是低压的概率。

第20章　用贝叶斯网学习和动作

20.1　学习贝叶斯网

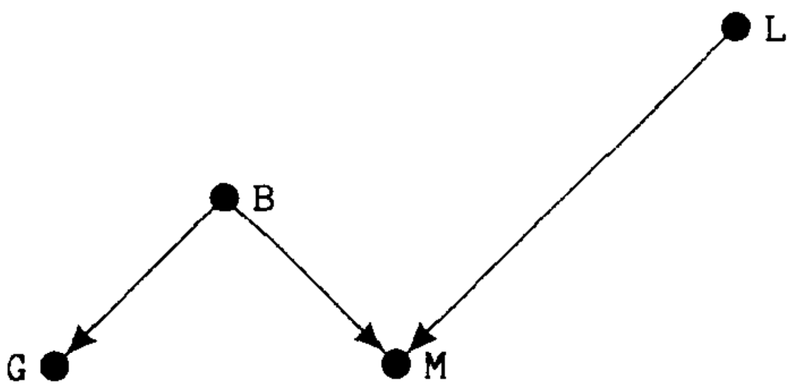
学习一个贝叶斯网的问题是寻找一个网络，它能最好地匹配（按照某个记分度量）一个数据训练集（*training set*）Ξ，Ξ是所有（至少有一些）变量值的实例集合。说“寻找一个网”，我们的意思是既要找到DAG结构，也要找到与DAG中每个节点相关的条件概率表（CPT）。

20.1.1　已知网络结构

如果知道网络的结构，那么只需找到CPT，我们首先描述这种情况。通常，人类专家能对一个问题领域提出适当的结构但不能做出CPT。在我们必须学习网络结构的情况下，学习CPT仍是必需的。学习CPT有一个比较容易的和一个比较难的两种情况。在容易的情况下，没有缺失的数据，即训练集合Ξ的每个成员对网中表达的每个变量有一个值。然而在更实际的设置中，情况常常是一些训练记录的变量值缺失了；缺失的数据导致更难以学习CPT。

1. 无缺失数据

首先假定没有缺失任何数据。这里，如果有充足的训练样本，我们只要计算每个节点和它的双亲的采样统计信息。假如给定双亲 $\mathcal{P}(V_i)$，我们想得到某个节点V_i的CPT。遵守前面的约定，我们用v_i指称V_i的值。V_i 的表和它具有的不同值（小于1）一样多。在布尔表达式中，再次假定，对每个节点仅有一个CPT。设V_i有k_i个父节点。因为每个双亲有两个可能值，那么在表中有 2^{k_i} 项（行）。我们用向量变量$\boldsymbol{P}_i$指称与 V_i 的双亲有关的变量，用向量值$\boldsymbol{P}_i$指称这些变量的值。采样统计结果 $\hat{p}(V_i = v_i \mid \boldsymbol{P}_i = \boldsymbol{p}_i)$，由Ξ中有$V_i = v_i$ 和$\boldsymbol{P}_i = \boldsymbol{p}_i$ 的采样数除以有$\boldsymbol{P}_i = \boldsymbol{p}_i$的采样数得到。为了学习CPT，我们仅仅将实际的这些采样统计结果用于网中的所有节点。



G	M	B	L	实例数
True	*True*	*True*	*True*	54
True	*True*	*True*	*False*	1
True	*False*	*True*	*True*	7
True	*False*	*True*	*False*	27
False	*True*	*True*	*True*	3
False	*False*	*True*	*False*	2
False	*False*	*False*	*True*	4
False	*False*	*False*	*False*	2
				100

图20-1　一个网络和一些示例值

用一个例子可以使这个计算更清楚。考虑一个同图19-2有相同结构的贝叶斯网，在图20-1中重复它，但没有CPT。假如我们观察了图中G、M、B和L的100组值（注意到有些组合没有出现，有些比其他出现得更频繁）。为了计算采样概率$\hat{p}$(B = *True*)，我们只要计算在所有的采样中B为*True*出现的次数。得到$\hat{p}$(B = *True*)= 0.94。同样，$\hat{p}$(L = *True*) = 0.68。对节点B和L，这些概率正是它们的CPT所需要的。

我们用下面解释的典型计算方式计算节点M的CPT行：为了计算$\hat{p}$(M = *True* | B = *True*，L = *False*) (简写为$\hat{p}$(M | B，¬L))，计算M为*True*、B为*True*，L为*False*的次数，并除以B为*True*、L为

*False*的次数。我们得到$\hat{p}$（M | B, ¬L）= 0.03。对节点G我们进行相似的计算。可以计算整个采样统计结果集，把它们与图19-2中给出的CPT比较。

注意，在例子中，有些采样统计是基于很小的采样的——导致相应的基础概率的可能不精确评估。一般地讲，一个CPT的指数级数量的大量参数可能无法使训练集对这些参数产生良好评估的能力。如果很多参数有相同（或接近相同）的值，可能会减轻这个问题。[Friedman & Goldszmidt 1996a]已经探索过如何使用这个冗余，减少在一个CPT中必须评估的参数数量的技术。

同样，在观察采样前，我们可以有CPT中项的先验概率。给定一个训练集，给先验概率合适的权值。Bayesian更新了CPT，当然这个过程有点复杂（见[Heckerman, Geiger, & Chickering 1995]。当有一个非常大的训练集时，先验的作用被极大地减小了。

2. 缺失数据

在收集被一个学习过程使用的训练数据中，常常发生数据缺失。有时，要被捕获的数据不经意地缺失了，有时数据缺失本身是重要的。这里处理第一种情况。一个简单、收敛的迭代计算采样统计过程已被证明是对之有效的[Lauritzen 1991]。用刚刚描述的例子来介绍该方法的主要思想。假如，不用图20-1中的数据，而用如下数据：

G	M	B	L	实例数
True	*True*	*True*	*True*	54
True	*True*	*True*	*False*	1
*	*	*True*	*True*	7
True	*False*	*True*	*False*	27
False	*True*	*	*True*	3
False	*False*	*True*	*False*	2
False	*False*	*False*	*True*	4
False	*False*	*False*	*False*	2

星号*表示变量值组中与那个位置相关的变量值缺失了。问题是当试图评估这个网络的CPT时，我们如何处理这些缺失值？先考虑三次采样，其中G = *False*，M = *True*，L = *True*，B的值缺失的情况。这三次采样中的每一次可能有B = *True*或B = *False*，我们不知道是哪一个。但对这些采样，我们知道G、M和L的值。因此，虽然不知道B的值，但给定了G、M和L的值，我们能计算B的概率p(B | ¬G，M，L)。这个概率能用前面讲述的概率推理方法计算——工作在网络结构（图20-1）和网络的CPT上，条件是我们有这些CPT（当然，我们还没有它们，但是将简要讨论这个问题）。因此，为了计算采样统计以估计该网络的CPT，三次采样中的每一次能用两个加权采样代替——一个是B = *True*，用p(B | ¬G，M，L）加权，另一个是B = *False*，权值为p（¬B | ¬G，M，L）= 1 − p（B | ¬G，M，L）（顺便提一下，从图20-1有p(B | ¬G，M，L）= p(B | ¬G，M)，因为给定G和M，B条件独立于L。）

我们把相同的过程应用到7次采样B = *True*，L = *True*，G和M的值缺失的情形。这些采样中的每一个可由对应组合（G，M）、（G，¬M）、（¬G，M）和（¬G，¬M）的4个加权采样代替，权值分别是概率p(G，M | B，L)、p(G，¬M | B，L)，p(¬G，M | B，L）和p(¬G，¬M | B，L)。我们可以再次用网络结构和CPT计算这些概率（当在任何一个采样中的缺失值数量很大时，存在指数爆炸的采样危险）。

现在，我们能用加权采样（其中，缺失值已被填充——用所有可能的方法）和其他采样（它们中没有缺失值）一起进行频率统计以计算CPT的估计。这个过程与在没有缺失值中描述的过程相同，除了一些计数现在不是整个数量（因为加权）外。但是正如前面提到的，为了通过概率推理计算权值，我们需要CPT，然而我们没有它。一个叫期望最大化（EM）[Dempster, Laird, & Rubin 1977]的方法可以用来对一个CPT集合调零。首先，我们为整个网络的CPT中的参数选择随机值，用这些随机值计算需要的权值（给定观察要数据的值时缺失数据值的条件概率），然后用这些权值反过来估计新的CPT。我们迭代这个过程，直到CPT收敛，保证收敛能达到。在大部应用中，收敛是快速的。

即使这个有缺失值的小例子也需要冗长的计算，应用EM方法时最好让计算机程序去执行概率推理和频率统计。

20.1.2　学习网络结构

如果不知道网络结构，那么我们必须设法去找出这个结构以及相关的CPT，以使它能最好地符合训练数据。为此，需要一个尺度对候选网络打分，我们需要指定一个过程以在可能的结构中搜索。在本小节将讨论这两个问题。

1. 记分制

有几个度量方法可用来对网络打分。一种方法是基于描述长度的。它的思想是：假如我们想发送训练集Ξ给某个人，为此，把变量值编码成一个位串，然后发送位。我们需要多少位呢？即，消息的长度是多少？有效的编码利用要发送数据的统计属性，这些统计属性正是我们企图在贝叶斯网中建模的。如果找到一个适当的贝叶斯网，我们能基于它使用哈夫曼编码对要传输的数据编码。根据信息理论[Cover & Thomas 1991]，一组数据Ξ，按照一个给定贝叶斯网 $\mathcal{B}$ 的组合概率分布，其最好编码需要$L(\Xi, \mathcal{B})$比特：

$$L(\Xi, \mathcal{B}) = -\log p[\Xi]$$

其中$P[\Xi]$是被发送的特殊数据的概率（按照 $\mathcal{B}$ 规定的联合概率）。给定一些特殊数据Ξ，我们企图找到网络 $\mathcal{B}$，它使$L(\Xi, B)$最小化。在了解这个方法之前，我们先计算$\log P[\Xi]$。假定数据Ξ包含m个采样值：$\boldsymbol{v}_1,\cdots,\boldsymbol{v}_m$，每个$\boldsymbol{v}_i$是$n$个变量值的一个$n$维向量，$p(\Xi)$是联合概率$p[v_1, \cdots\cdots, v_m]$。假定每个数据按照 $\mathcal{B}$ 指定的概率分布被独立提供，我们有

$$p(\Xi) = \prod_{i=1}^{m} p(\mathbf{v}_i)$$

和

$$-\log p(\Xi) = -\sum_{i=1}^{m} \log p(\mathbf{v}_i)$$

其中每个$p(\boldsymbol{v}_i)$（由$\boldsymbol{v}_i$指称变量值的联合概率）从贝叶斯网 $\mathcal{B}$ 计算。这些计算虽然是冗长的，但确实被用来对各种试验贝叶斯网计分。当然，每个网络不但包括网络图结构，而且有CPT。可以证明：给定一个网络结构和一个训练集Ξ，使$L(\Xi, \mathcal{B})$最小的CPT是从Ξ计算的采样统计结果获得的[Friedman & Goldszmidt 1996a]。

但是单独的$L(\Xi, \mathcal{B})$不是一个非常好的度量，因为它的使用促成了有很多弧的大网络。这样一个网络对Ξ过于特殊化，即它过于适合数据。对记分度量的适当调整能这样实现；为了

用一个基于 $\mathcal{B}$ 的有效编码把Ξ传送给某个人，我们也必须传递 $\mathcal{B}$ 的描述以便接收者能够对消息解码。这样，我们必须加一项到$L(\Xi, \mathcal{B})$，这个项是传输 $\mathcal{B}$ 需要的消息长度。粗略地讲，传送 $\mathcal{B}$ 要求的比特数是 $\frac{|\mathcal{B}| \log m}{2}$，其中$|\mathcal{B}|$ 是 $\mathcal{B}$ 中的参数个数，$\frac{\log m}{2}$ 一般被认为是适合表示每个数字参数的比特数。因此调整后的记分制$L(\Xi, \mathcal{B})$ 为：

$$L'(\Xi, \mathcal{B}) = -\sum_{i=1}^{m} \log p(\mathbf{v}_i) + \frac{|\mathcal{B}| \log m}{2}$$

现在，我们搜索一个给出数据和网络编码的最小描述长度的网络。使用两个因子允许我们对发送数据和网络做出更好的权衡。

作为一个例子，对图19-2中显示的网络和图20-1中所示的发送数据，我们计算$L'(\Xi, \mathcal{B})$。首先，我们计算$L(\Xi, \mathcal{B})$，即发送图20-1中的数据要求的比特数——假定数据由图19-2中的贝叶斯网给定的一个概率分布提供。图20-1中表的第1项的概率是

$$p(\text{第1项}) = p(G|B)p(M|B, L)p(B)p(L)$$
$$= 0.95 \times 0.9 \times 0.95 \times 0.7 = 0.569$$

采用负对数（对以2为底的），产生

$$-\log p(\text{第1项}) = 0.814$$

在数据中有54个这种“第1项”，因此54个第1项对$L(\Xi, \mathcal{B})$的作用结果是$54 \times 0.814 = 43.9$。表中其他项的总结果分别是6.16、27.9、52.92、16.33、12.32、24.83和12.32。把这些结果加起来得到：

$$L(\Xi, \mathcal{B}) = 196.68 \text{ bit}$$

下面，我们计算 $\frac{|\mathcal{B}| \log m}{2}$，发送图19-2的网络需要的位数。在这个网中有8个参数。因此

$$\frac{|\mathcal{B}| \log 100}{2} = 4 \times 6.64 = 26.58 \text{ bit}$$

因此这个网络的记分度量是

$$L'(\Xi, \mathcal{B}) = 196.68 + 26.58 = 223.26 \text{ bit}$$

其他的网络可用同样的方式评估，而图19-2中的网络大概是最理想的。

2. 搜索网络空间

当然，所有可能的贝叶斯网集合是如此之大，以致我们不可能企求一种详尽的搜索以发现一个使$L'(\Xi, \mathcal{B})$最小的网络。一种可能是利用下山搜索或“贪婪”搜索。即我们从一个给定的网络开始（比如一个没有弧的网络，假定它独立于所有的变量），估计该网络的$L'(\Xi, \mathcal{B})$，然后对它做一些小改变，来看这些改变产生的网络是否减少了$L'(\Xi, \mathcal{B})$。这些小改变可以是加一条弧，或减一条弧，或者掉转一条弧的方向。每发生一个改变，我们用从Ξ中导出的采样统计计算改变网络的CPT。然后这些CPT被用来计算$L'(\Xi, \mathcal{B})$的部分 $-\sum_{i=1}^{m} \log p(V_i)$。新网络中的参数数量用来计算部分$(|\mathcal{B}| \log m)/2$。这些计算可以简化，由于描述长度计算可分解成网络中每个CPT上的计算。当记分度量可分解时，总度量是局部度量的和[Friedman & Goldszmidt 1996a]。因此，当一个弧被加、被减或被反向时，我们只需计算在采样统计中的变

化和改变涉及到的节点 V_i的p（ $V_i \mid \mathcal{P}(V_i)$ ），其他的 $p(V_i \mid \mathcal{P}(V_i))$ 保持不变。除了描述长度，可分解度量也被用来评价一个网络适合数据的程度（参见[Heckerman, Geiger & Chickering 1995]。

对一些非试验问题，贝叶斯网学习方法已用来学习网络结构和CPT。作为一个例子，考虑图20-2中的网络．给出三个网络，第一个是对一个问题中37个变量的关系编码，这个问题是一个用于医院特护单元通风设备管理的警报系统。这个已知网络被用来产生37个节点随机值的、大小为10 000的训练集。用这个随机采样，从第二个网络开始（相互之间无任何依赖性），第三个网络用类似于刚刚描述的方法来学习（详见[Spirtes & Meek 1995]。注意在结构上非常相似——仅去掉了一个弧。

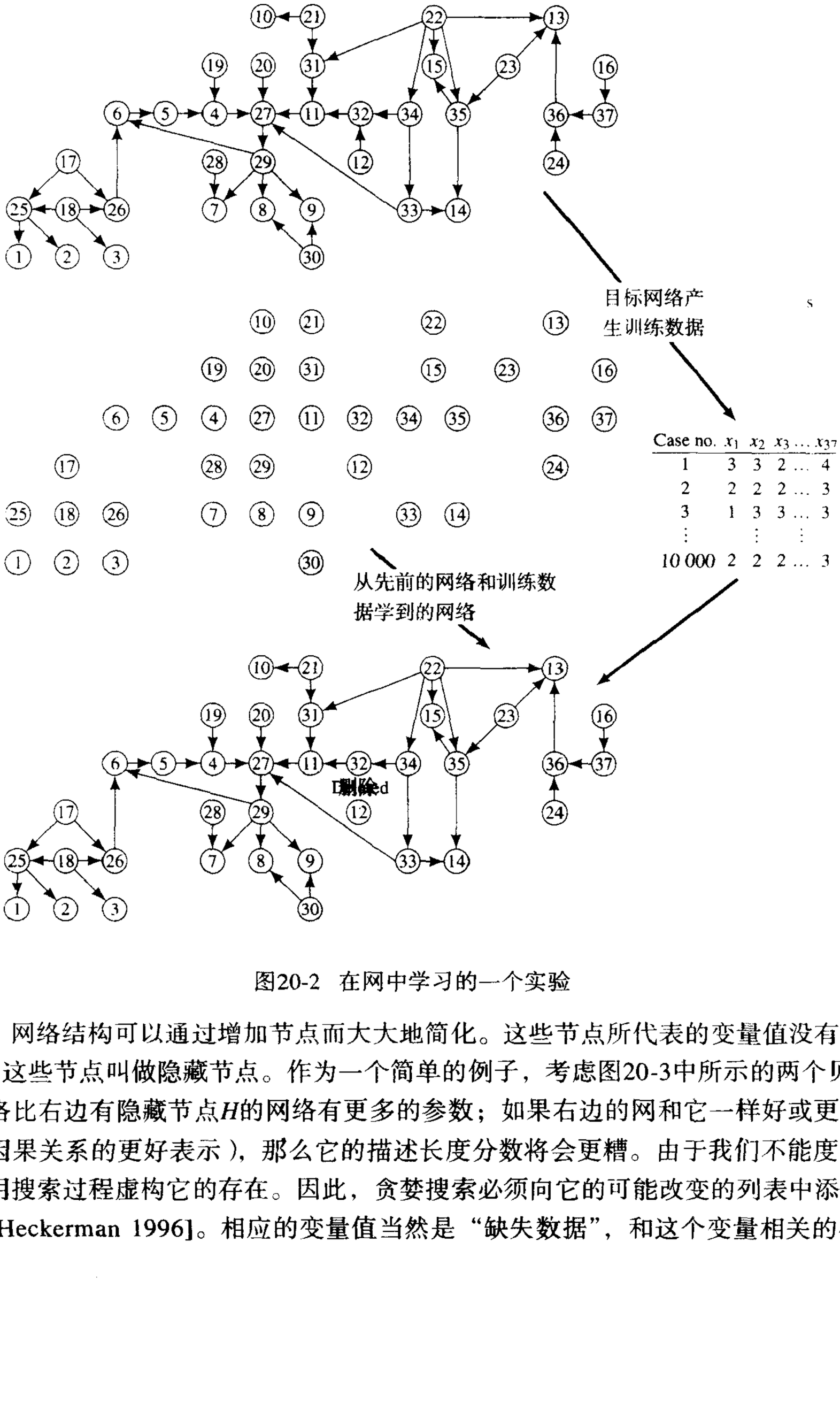


图20-2　在网中学习的一个实验

有时，网络结构可以通过增加节点而大大地简化。这些节点所代表的变量值没有在训练集三中给定。这些节点叫做隐藏节点。作为一个简单的例子，考虑图20-3中所示的两个贝叶斯网。左边的网络比右边有隐藏节点H的网络有更多的参数；如果右边的网和它一样好或更好（如果它是变量因果关系的更好表示），那么它的描述长度分数将会更糟。由于我们不能度量隐藏变量，必须用搜索过程虚构它的存在。因此，贪婪搜索必须向它的可能改变的列表中添加一个新节点（见[Heckerman 1996]。相应的变量值当然是“缺失数据”，和这个变量相关的概率必须

通过前面描述的EM方法引证[⊖]。

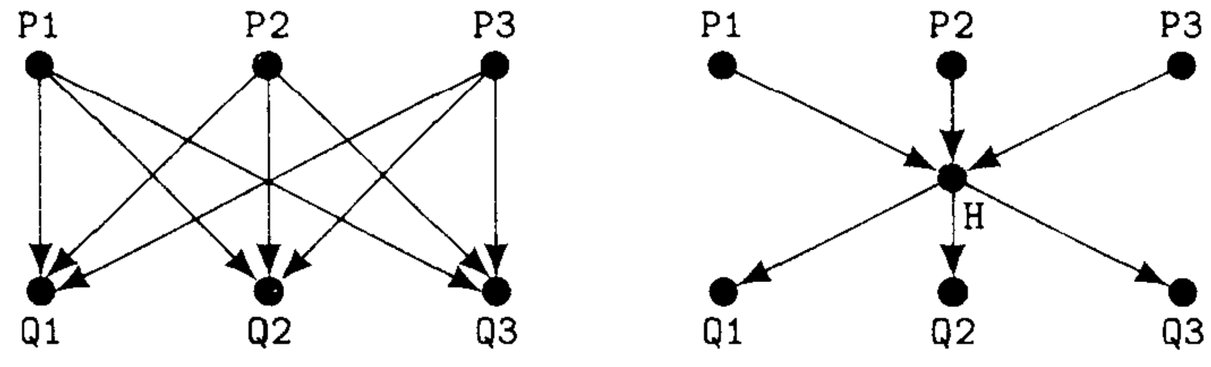


图20-3 两个网络——其中一个有一个隐藏变量节点

20.2 概率推理与动作

20.2.1 一般设置

有很多概率推理的应用。一个著名的应用是在专家系统中根据应用到特殊数据的一般知识做出最好的猜测。诊断给定症状的疾病就是一个例子。下面描述使用贝叶斯网的概率推理如何能被一个agent利用，这个agent必须根据给定感知信息和关于环境状态的记分尺度决定下一个最好动作（我的处理基于[Russell & Norvig 1995, 第17章]，它利用了[Dean & Wellman 1991, 第17章]描述的一个方法）。

待研究的问题是第10章所处理问题的一个一般化。回忆一下，在那里描述了一个agent使用一个感知/计划/动作循环，计划阶段通过朝着目标方向预测在当前环境状态下计算要被采用的下一个最好动作（有时仅仅到达一个有限的范围），动作阶段执行由计划者推荐的第一个动作，感知阶段设法辨别下一次循环的结果环境状态。在第10章，也已将一个“目标”符号一般化为在一定的环境状态下给出（或得到）的一个“奖赏”目录。这个奖赏反过来按照总计的打折未来奖赏为每个状态导出一个“值”，其中的未来奖赏被一个执行动作以使它的奖赏最大化的agent实现。这里保留这个一般化的目标符号，把一个应用归因于每个环境状态。

前面，对感知/计划/动作的结构采用了关于环境和感知动作可靠性的有些不一致的假设集合。假定一个agent通过感知能准确决定它的当前状态，能精确预测它的所有动作结果。然而，倘若这些假设是不合理的（它们常常是不合理的），我们的agent采用一个简单动作，并立即用它的传感器去发现实现产生的环境状态。用下面的说法使这个方法合理化：即使采用的动作并不总会有它的预期效果，传感器有时会出错，但传感器（平均地）将使agent通过状态空间知道它的进展，并且重复计划将再次引导agent朝着需要的目标（或奖赏）前进。

现在，有工具允许我们更适当地处理不确定性，就能明确地采用更现实的假设。新的agent不知道它在哪个状态，它仅仅知道它在各种状态的概率。agent的传感器不能给出环境状态的确切知识，它最多只能加深这些状态的概率。动作结果仅仅被大概地了解：在一个给定状态下采取的动作可能导致一组新状态中的任何一个——每一个都有相应的概率。通过计划和感知，我们想让agent选择使它的预期应用最大化的那个动作。一个agent能在它的完全一般化中处理这个问题需要的计算量非常大，并需要再一次近似和进一步的限制假设。在下面的一个特殊例子中讨论这些问题。

⊖ 在学习贝叶斯网中虚构隐藏节点类似于在学习命题规则中虚构新原子或在学习逻辑程序中虚构新的关系常量（谓词）。

20.2.2　一个扩展的例子

由于各种假设极大地增加了计算难度，下面的例子使用一个简单agent和环境，以免模糊了我们的主要思想。考虑一个机器人，它处于图20-4所示的一个有5个单元的一维网格中。环境状态只涉及到机器人的位置，我们用一个状态变量E表示它，E有5个可能值{-2, -1,0,1,2 }，每个位置对机器人有一个应用U。中间单元有一个应用0，它和其他的应用值显示在图中。为了开始过程，我们假定机器人（精确地）知道当$t = 0$时,它在标志为0的单元中，即$E = 0$。

机器人在第i个时间步骤采取的动作用符号A_i指称。它试图向左移一个单元（$A = L$）或向右移一个单元（$A_i = R$）。无论哪种情况，一个移动有预期结果的概率为0.5；每个动作没有任何结果的概率是0.25，一个动作使机器人以与预期相反的方向移动到相邻单元的概率为0.25。因此，在几次移动后，机器人只有它实际位置的概率知识。

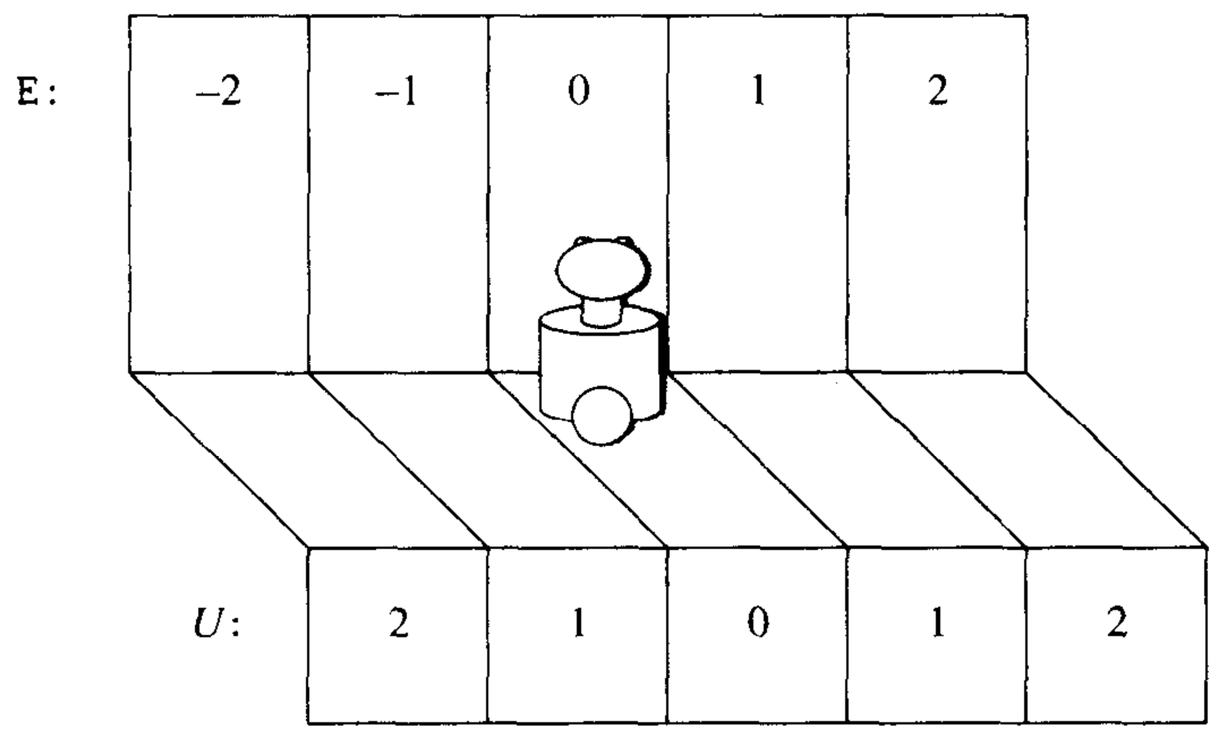


图20-4　限制在5个单元的机器人

机器人在第i个时间步骤通过一个感知信号S_i感知它的位置。但是我们假定那个传感器有点不可靠。假如E_i有确定值，S_i有同样值的概率是0.9。S_i有每个其他值的概率是0.025。

面对各种障碍，机器人的问题是做出使它的应用的期望值提前几步最大化的移动。如果我们使预期的应用仅提前一步最大化，用来选择机器人的下一步动作的决策技术最容易解释。假定当$t = 0$时机器人企图向右移动，即$A_0 = R$。所得到的环境状态由E_1给出。为了继续感知/计划/动作循环，机器人通过观察$S_1 = 1$感知它的位置。它下一步采取什么移动呢？确实，在做了那个移动后，它如何基于感知数据、对当前状态所做推理和动作和效果来？

使用一个叫做动态决策网的特殊信念网的概率推理，来选择最大化应用动作。在图20-5中显示了适合这个问题的网络。这个网络允许机器人在采用动作时和新感知的信息变得可用时迭代地进行推理，给定$E_0 = 0$、$A_0 = R$和$S_0 = 1$后，我们能用普通的概率推理计算期望的应用值U_1，它先由$A_1 = R$产生，再由$A_1 = L$产生。然后机器人选择给出更大值的动作（这种情况

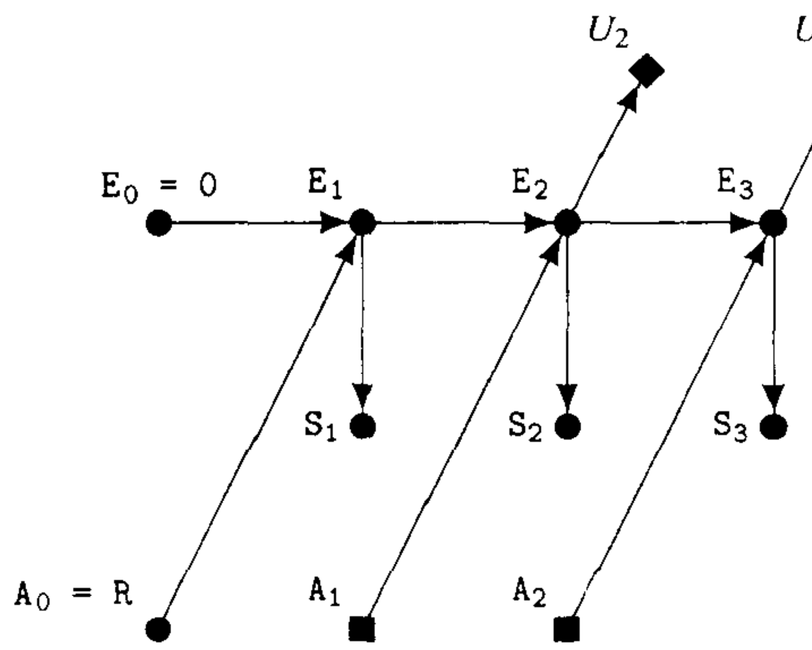


图20-5　一个动态决策网

下的动作选择基于机器人仅向前看一步。进一步的向前看将涉及到对可选动作序列计算更远的应用）。在动态决策网中使用不同形状的节点表示这些节点的不同假设。方形节点表示变量的值仍在agent的完全控制之下，它们被称为决策点。在一个动态决策网中，在agent已经做了决策之后，它们成为（有已知值）普通信念网节点。棱形节点表示应用值的变量。应用变量的期望值是它们双亲的各种值的概率函数。

由网络结构注意到该环境是马尔可夫模型，也就是说，在$t+1$时刻，环境状态所能知道的东西（没有感知的）完全取决于t时刻的环境状态和动作（概率地）。在这种意义下，大多数agent环境能被假设是马尔可夫模型（或者能引入附加的变量以使它们成为那样）。也要注意网络结构包含了假设——给定时刻t的环境，在时刻t感知的东西条件独立于以前的每个事情。

在$t = 1$时，通过采取一个计划的动作把时间前沿向前移动一步（通过使预期应用U_2最大化的过程进行计划），并在感知了下一个状态E_2时能被感知的东西S_2后，整个处理用另一个感知/计划/动作循环重复。执行一些计算将有助于理解。

首先，给定时刻$t = 0$时已知的东西、和假定A_0的两个不同概率，我们需要计算两个预期应用：

$$Ex[U_2 \mid E_0 = 0, A_0 = R, S_1 = 1, A_1 = R]$$

和

$$Ex[U_2 \mid E_0 = 0, A_0 = R, S_1 = 1, A_1 = L]$$

为了计算这两个预期值，对A_1的两个值中的每一个的E_2的不同值，我们要计算$p(E_2 \mid E_1 = 0, A_0 = R, S_1 = 1, A_1)$。这些计算形式在后面的步骤中将被重复，但为了具体，在此问题的步骤中都写出它。这个形式对A_1的每个值也是相同的，因此仅写出$A_1 = R$的情形。我们用上一章解释的polytree算法计算$p(E_2 \mid E_0 = 0, A_0 = R, S_1 = 1, A_1 = R)$。

首先，我们“包含没有给定的双亲”，得到：

$$p(E_2 \mid E_0 = 0, A_0 = R, S_1 = 1, A_1 = R)$$
$$= \sum_{E_1} p(E_2 \mid E_0 = 0, A_0 = R, S_1 = 1, A_1 = R, E_1) p(E_1 \mid E_0 = 0, A_0 = R, S_1 = 1, A_1 = R)$$

考虑条件独立性，这个等式能被简化：

$$p(E_2 \mid E_0 = 0, A_0 = R, S_1 = 1, A_1 = R)$$
$$= \sum_{E_1} p(E_2 \mid A_1 = R, E_1)\, p(E_1 \mid E_0 = 0, A_0 = R, S_1 = 1)$$

在用于机器人动作选择的决策网中，上述求和的第一项叫做马尔可夫过程的动作模型。对一个给定的前一个状态和动作，它给出各种可能的随后状态的概率。根据polytree算法，求和中的第二项用贝叶斯法则重写为：

$$p(E_1 \mid E_0 = 0, A_0 = R, S_1 = 1) = kp(S_1 = 1 \mid E_0 = 0, A_0 = R, E_1) p(E_1 \mid E_0 = 0, A_0 = R)$$

其中k是一个标准化因子，它被选择（后面）以使概率和为1。再使用条件独立，我们有

$$p(E_1 \mid E_0 = 0, A_0 = R, S_1 = 1) = kp(S_1 = 1 \mid E_1) p(E_1 \mid E_0 = 0, A_0 = R)$$

上述结果中的第1项叫感知模型。对很多环境状态，它给出了传感器将有各种值的概率（对每个环境状态，一个完全可靠和提供最多信息的传感器，将把所有的概率集中到一个传感器值）。

第2项又是一个动作模型。

汇集我们的结果，产生：

$$p(E_2 \mid E_0 = 0, A_0 = R, S_1 = 1, A_1 = R)$$

$$= k \sum_{E_1} p(E_2 \mid A_1 = R, E_1)\, p(S_1 = 1 \mid E_1) p(E_1 \mid E_0 = 0, A_0 = R)$$

为了评估这个表达式（为E_2的各种值），使用我们已经做出的关于动作结果和传感器可信度的概率假设。为了说明，我们仅计算$p(E_2 = 1 \mid E_0 = 0, A_0 = R, S_1 = 1, A_1 = R)$。计算涉及到下面的概率，它们是图20-5中网络的CPT条目（我们必须对所有的E_1值求和）。

$p(E_2 = 1 \mid A_1 = R, E_1 = 0) = 0.5$

$p(E_2 = 1 \mid A_1 = R, E_1 = 1) = 0.25$

$p(E_2 = 1 \mid A_1 = R, E_1 = 2) = 0.25$

$p(E_2 = 1 \mid A_1 = R, E_1 = -1)$ 和 $p(E_2 = 1 \mid A_1 = R, E_1 = -2)$ 都等于0。

$p(S_1 = 1 \mid E_1 = -2) = 0.025$

$p(S_1 = 1 \mid E_1 = -1) = 0.025$

$p(S_1 = 1 \mid E_1 = 0) = 0.025$

$p(S_1 = 1 \mid E_1 = 1) = 0.9$

$p(S_1 = 1 \mid E_1 = 2) = 0.025$

$p(E_1 = -1 \mid E_0 = 0, A_0 = R) = 0.25$

$p(E_1 = 0 \mid E_0 = 0, A_0 = R) = 0.25$

$p(E_1 = 1 \mid E_0 = 0, A_0 = R) = 0.5$

$p(E_1 = -2 \mid E_0 = 0, A_0 = R)$ 和 $p(E_1 = 2 \mid E_0 = 0, A_0 = R)$ 都等于0。

执行求和产生

$$p(E_2 = 1 \mid E_0 = 0, A_0 = R, S_1 = 1, A_1 = R)$$

$$= k \times [(0.5 \times 0.025 \times 0.25) + (0.25 \times 0.9 \times 0.5)]$$

$$= k \times 0.14375$$

我们对E_2的其他值执行类似的计算以得到$p(E_2 \mid E_0 = 0, A_0 = R, S_1 = 1, A_1 = R)$。由于所有这些的和为1，我们能求出$k$。用$E_2$的这些概率，给定$A_1 = R$，我们计算$U_2$的预期值。重复这个过程来计算给定$A_1 = L$的$U_2$的预期值，并选择产生更大值的动作（从问题的结构我们知道它将是$A = R$。但是，用这个例子仅仅说明这个方法——没有什么惊奇的）。

20.2.3 一般化举例

分析方程$p(E_2 \mid E_0 = 0, A_0 = R, S_1 = 1, A_1 = R)$，允许我们把它扩展到随后的时间步。这个方程在$t = 1$时被估计，就像$S_1 = 1$已被感知和动作$A_1 = R$被采取前一样。其他“给定”的值成为过去。因此，我们能改写为$p(E_2 \mid <values\ before\ t = 1>, S_1 = 1, A_1 = R)$。同样，在求和中的表达式$p(E_1 \mid E_0 = 0, A_0 = R)$能被写为$p(E_1 \mid <values\ before\ t = 1>)$。用这些改变，我们有：

$$p(E_2 \mid <values\ before\ t = 1>, S_1 = 1, A_1)$$

$$= k \sum_{E_1} p(E_2 \mid A_1, E_1)\, p(S_1 = 1 \mid E_1) p(E_1 \mid <values\ before\ t = 1>)$$

将等式一般化为：

$$p(E_{i+1} \mid < values\ before\ t = i >, S_i = s_i, A_i)$$

$$= k \sum_{E_i} p(E_{i+1} \mid A_i, E_i) p(S_i = s_i \mid E_i) p(E_i \mid < values\ before\ t = i >)$$

为了决定动作，当我们处理时，用下面的方式使用这个等式。为了计算在$t = i$时要采取的动作A_i：

1) 从上一个时间步（$i-1$）（以及感知到$S_{i-1} = s_{i-1}$后），我们已经对E_{i-1}的所有值计算了$p(E_{i-1} \mid < values\ before\ t = i >)$。

2) 在$t = i$时，感知$S_i = s_i$，并且用感知模型计算E_i所有值的$p(S_i = s_i \mid E_i)$。

3) 根据动作模型，计算E_{i+1}和A_i所有值的$p(E_{i+1} \mid A_i, E_i)$。

4) 对A_i的每个值和E_{i-1}的一个特定值，我们对E_i所有值上的$p(E_{i+1} \mid A_i, E_i)\, p(S_i = s_i \mid E_i)\, p(E_i \mid < values\ before\ t = i >)$求和，并乘以一个常量$k$，产生和$p(E_{i-1} \mid < values\ before\ t = i >, S_i = s_i, A_i)$成比例的值。

5) 对E_{i-1}的所有其他值，重复上述过程，计算常量k得到对每个E_{i-1}和A_i值的$p(E_{i-1} \mid < values\ before\ t = i >, S_i = s_i, A_i)$的实际值。

6) 用这些概率值，计算对A_i的每个值U_{i+1}的预期值，选择使预期值最大的那个A_i。

7) 采取在上一步选择的动作，i加1，循环。

这就是使用一个动态决策网络选择动作的要素。该方法可扩展到其他应用。动作影响环境的方式和环境影响传感刺激的方式一般通过类似图20-5中的网络来很好地建模。代替有一个单一环境变量E_i，在每个时间步，我们可以有一个值向量$\mathbf{E}_i = (E_{i1}, \cdots, E_{in})$。同样，代替有一个单一的传感变量$S_i$，我们可以有一个值向量$\mathbf{S}_i = (S_{i1}, \cdots, S_{im})$。当然动态决策网在每个时间步必须扩展到包括所有的这些变量节点和它们的依赖，但是计算的一般形式和这个简单例子中的一样。虽然网络对一个给定的时间步本质上会复杂一些，但马尔可夫假设简化了时间步之间的依赖。

向前看超过一步涉及到向前传播概率到要计算一个预期应用的点。显然，这样一个扩展迅速地变得不实际。同时也变得不是非常有用，因为$\mathbf{E}_i$的概率分布变得相当扩散。因此，在表达这个概率扩展之前所做的有点不一致的假设保留了一个合理的选择。

20.3 补充读物和讨论

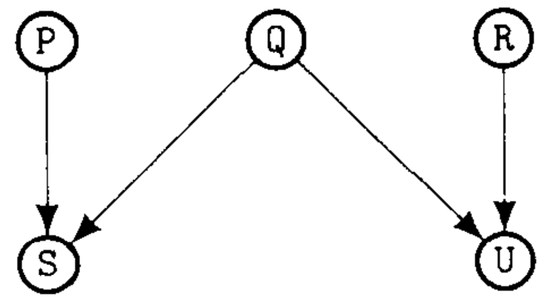
贝叶斯网学习是一个每年有很多重要新论文出现的活跃的研究领域。[Neal 1991]描述了用神经网络学习贝叶斯网的方法。这里描述的用于评估建议的贝叶斯网的技术使用了最小描述长度的概念[Rissanen 1984]。[Friedman 1997]描述了当不知道一个网络结构且有缺失数据时学习贝叶斯网的技术。[Cooper & Herskovitz 1992]做了关于学习贝叶斯网的早期工作。

[Forbes, et al. 1995]提出了一个使用动态决策网驱动一个自动设备的系统。

在随机状态下的应用评估构成了决策理论的主旨。在AI中使用决策理论的一个事例见[Horvtiz, Breese, & Henrion 1988]。Markov决策问题（MDP）[Puterman 1994]和部分可观察Markov决策问题（POMDP）[Cassandra, Kaelbling , & Littman 1994]理论提供了在随机状态下动作结果的基本理论模型。

习题

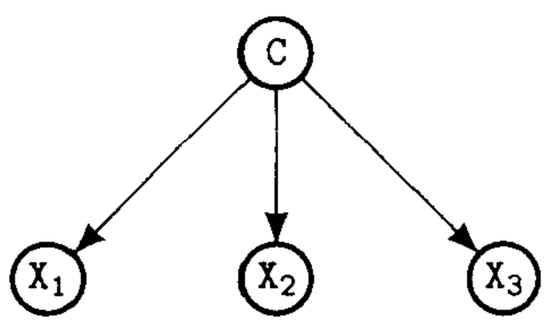
20.1 考虑有下面结构的贝叶斯网：



假如我们能控制“原因变量”P、Q和R，即，我们能安排实验以让这些变量可以取值（*True*或*False*）的任意组合，然后观察S和U的结果。为了学习S和U的CPT，将需要这些变量的什么组合？从这个实验，我们能学到P、Q和R的先验概率吗？

20.2 计算图20-3中每个网络的参数数量。

20.3 一个贝叶斯网有下图所示的结构（注意，这个结构暗示X_1条件独立于给定的C）。给定X_1、X_2、X_3值的一个训练集，每一对有一个C值，给出如何应用采样统计以计算$p(C \mid X_1, X_2, X_3)$的一个估计。



20.4 用上面习题的假设和结果，假定C代表两个动作之一的名字，这两个动作是一个机器人在它的二值传感器输入有由X_1、X_2、X_3给出的值时采取的。假定选择一个动作的策略是：如果$p(C = 1 \mid X_1, X_2, X_3) \geqslant p(C = 2 \mid X_1, X_2, X_3)$，那么选择$C = 1$。证明这样一个策略能被一个输入为$X_1$，$X_2$，$X_3$的TLU实现，把你的结果解释为一个训练该TLU的方法（在这个问题中，信念网没有一个因果解释；即应该采取的动作不是传感器输入的一个“原因”）。

20.5 一个机器人生活在一个如下图所示的5×5网格中。网格中的数字代表相对温度值。假设机器人开始在一个温度 = 2的单元中。它不知道从哪一个温度 = 2的单元开始，但它有一个网格地图，显示了每个单元的温度。

4	5	6	7	8
3	4	5	6	7
2	3	4	5	6
1	2	3	4	5
0	1	2	3	4

假定机器人能精确感知它所占据单元的温度值。它有4个动作，用north、east、south和west表示，为了使它的预期温度最大化，它必须选择其中之一。一股强风正从西南方向吹来。通常，每个动作将按指示的方向移动机器人一个单元，但由于有风，无论何时当机器人在一个温度 = 2的单元中时，动作有下述结果：

south → 没影响

west → 没影响

north → 向北移动一个单元的概率为0.5

north → 向北移动两个单元的概率为0.25

north → 向东移动一个单元的概率为0.25

east → 向东移动一个单元的概率为0.5

east → 向东移动二个单元的概率为0.5

1) 画出对应一个动作步的一个动态决策网。给出在CPT中的相关项。

2) 对4个动作中的每一个，计算期望的温度Ex[T1|A]。

3) 现在假定在时刻$t = t_0$机器人不能确信它的温度。相反，它感知到一个信号$S_0 = 2$。它告诉机器人关于它的温度。S_0的传感模型包括

$p(S_0 = 2|T_0 = 2) = 0.9$

$p(S_0 = 2|T_0 = 3) = 0.3$

$p(S_0 = 2|T_0 = i) = 0$ 对所有的i不等于2或3的值。

为以上情况画一个一步动态决策网。

4) 假定在一个温度 = 3的单元中采取的动作和温度 = 2的单元中有相同的结果。在这种情况下，来自4个动作的每一个的期望温度是什么?

第四部分　基于逻辑的规划方法

第21章　状 态 演 算

21.1　状态和动作推理

第7章介绍了状态空间的概念及如何搜索来计算动作计划以到达目标。当时还将讨论了基于图标或基于特征的状态空间搜索。现在，我们有更丰富的语言来描述特征和特征之间的约束，能更全面地研究基于特征的规划方法。

我们能删掉世界状态的一些属性，它们与当前的问题不相关或是未知的，这个事实是使用基于特征方法的强有力的方面。这在描述我们想让agent通过其动作达到的目标条件方面尤其重要。从图21-1中的配置开始，我们想让agent生成一个计划以让某个积木在*B*上面（不必关心哪个积木在*B*上或者*B*在那里）。这个目标能用公式（∃x）On(x, B)简单地描述。一般地讲，在谓词演算中，一个目标条件能用任何合式公式描述，我们能通过试图从这些公式证明目标合式公式来确定一个目标能否在一个由这些公式描述的状态空间中被满足。

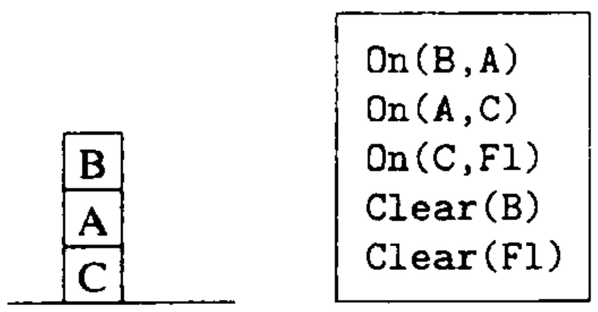


图21-1　积木的一个配置

在本章和下一章，将介绍一些用来寻找一组动作以获得由合式公式描述的目标状态的技术。这里用谓词演算机制直接推理状态和动作。就像在所有的谓词演算推理中一样，搜索仍然是必要的，但现在搜索是在一个逻辑表达式空间上进行，而不是在一个世界状态的模型空间上进行。在下一章，描述了另一个方法，在那里算子被用来改变状态描述，搜索将在一个状态描述空间上进行。

状态演算(*situation calculus*)[McCarthy & Hayes 1969, Green 1969a]是一种状态、动作和动作作用于状态的结果的谓词演算的形式化。在一阶谓词演算中我们把知识表达为关于状态和动作的公式，然后用一个演绎系统来问这样的问题："存在满足一定（目标）属性的状态吗？如果存在，现在的状态如何通过动作才能被转换为那种状态呢？"。对这样一个询问的回答是构造一个到达希望状态的计划。虽然状态演算在一些早期的AI规划系统中有显著的地位，但现在很大程度上已被下一章讲到的方法所替代。然而，这种方法对研究和帮助澄清有关动作结果的概念问题仍有重要作用。

通过一个积木世界的例子引入状态演算。假如把图21-1中的状态标识为*S0*。用一阶谓词演算，我们可以用下面的公式描述*S0*：

$$On(B, A) \wedge On(A, C) \wedge On(C, Fl) \wedge Clear(B) \wedge \cdots$$

为了描述状态演算中的这个状态和其他状态，我们把状态具体化[⊖]，即将它们作为现存实

⊖ 动词reify 意思是说把某个抽象的东西看作是一个实在的或具体的东西。

体包括进我们的世界概念中。可能有很多这种状态，它们能用常量符号（如S0，S1，S1，…）、变量或者函数表达式表示。改变原子合式公式以包括一个指称包含预期关系的状态项。要对这些原子合式公式的预期解释进行一个相应的变化，这些原子合式公式现在指称状态间的关系，它们被称为流（*fluent*）。用公式构造一个在状态*S0*中为真的句子：

$$On(B, A, S0) \wedge On(A, C, S0) \wedge On(C, Fl, S0) \wedge Clear(B, S0)$$

我们也可以有所有状态的命题真值。例如：

$$(\forall x, y, s)[On(x, y, s) \wedge \neg(y = Fl) \supset \neg Clear(y, s)]$$

和

$$(\forall s)Clear(Fl, s)$$

(Clear（x，s）的意思是x上可以放某个东西。)用这些一般公理，我们能证明*S0*的各种命题。例如，我们能证明¬ clear(A, S0)和Clear(Fl，S0)。

为了表示动作及其结果，我们采取下面的步骤：

1) 把动作具体化（即我们想像有这样一个东西来做为动作）。这样，动作能用常量、变量或函数表达式表示。在状态演算形式中，一个动作被看作为动作涉及的实体的函数。在例子中，动作是积木的函数。例如，考虑把一个积木从一个地方移到另一个地方的动作。用表达式move（B，A，Fl）表示把*B*从*A*移到地板上的动作[⊖]（把一个动作看作为由那个函数值给定的一个“对象”）。

一般地讲，我们能用模式(*schema*)move (x, y, z)表达一个move动作系列，其中x、y和z是模式变量。利用常量，这些变量的实例化可以产生指称真正动作实例的表达式。

2) 接下来，想像一个函数常量do，它指称一个可以把动作和状态映射到状态的函数。如果α代表一个动作，σ代表一个状态，那么do（α，σ）指称把状态—动作对映射到通过在σ指称的状态中执行由α指称的动作获得的状态的一个函数。

3) 用合式公式表达动作的结果。在某些形式中，对每个动作-流对有两个这种合式公式（目前，我们忽略动作对流没有影响的对）。对{On, move}对，合式公式是

$$[On(x, y, s) \wedge Clear(x, s) \wedge Clear(z, s) \wedge (x \neq z) \supset On(x, z, do(move(x, y, z), s))]$$

和

$$[On(x, y, s) \wedge Clear(x, s) \wedge Clear(z, s) \wedge (x \neq z) \supset \neg On(x, y, do(move(x, y, z), s))]$$

这里我们假定公式中提到的所有变量都被全称量化。这些公式中的第一个被称为正效应公理(*positive effect axiom*)，它描述了一个动作怎么使一个流为真。第二个叫负效应公理(*negative effect axiom*)，它描述一个动作如何使一个流为假。在这个例子中，一个效应公理的先行条件描述了为了应用那个动作必须满足的前提条件(*precondition*)，结果描述了在应用动作后流是如何被改变的。

即使已根据On定义了Clear，我们也能为{Clear, move}对写出效应公理。当然，它们必须和定义以及{On ,move}的效应公理相一致。对{Clear, move}的效应公理是：

⊖ 注意，和前面用到的move算子相比，这里移动一个积木块的表达式包括那个积木被移走的地方。

$$[On(x, y, s) \wedge Clear(x, s) \wedge Clear(z, s) \wedge (x \neq z) \wedge (y \neq z) \supset Clear(y, do(move(x, y, z), s))]$$

和

$$[On(x, y, s) \wedge Clear(x, s) \wedge Clear(z, s) \wedge (x \neq z) \wedge (z \neq Fl) \supset \neg Clear(z, do(move(x, y, z), s))]$$

在这个例子中，前提由两部分组成。一部分表达了执行动作的前提条件；另一部分表达了条件，在这个条件下，如果动作被执行，动作将产生表达在公理结果中的效果。在正效应公理中，第二部分是（$y \neq z$）（即使$y=z$，动作也能被执行——把一个积木从一个位置向后移到相同的位置，但它没有在结果中声明的效果）。在负效应公理中，第二部分是（$z \neq Fl$）。（由于我们假定地板总是空的，没有任何动作可使它不空的）。

为了说明如何使用效应公理，考虑图21-2顶部的积木世界状态。这个状态满足使用置换{B/x, A/y, S0/s, Fl/z}的效应公理的前提条件。因此，我们能应用由move(B, A, Fl)指称的动作推出如下结果：

$On(B, Fl, do(move(B, A, Fl), S0))$,

$\neg On(B, A, do(move(B, A, Fl), S0))$, 和

$Clear(A, do(move(B, A, Fl), S0))$

这些表达式中的每一个都有do(move(B,A,Fl),S0)）作为它的动作后的状态项（为简短起见，用S1 指称这个动作后状态）。在图21-2中，显示了应用动作后的结果状态和描述那个状态的公式（除了从效应公理推出的公式，还有一些其他的公式也是S1 的真值公式；下面将简要描述这些其他的公式是如何被推导的。）

即使在一个动作后，在执行动作允许的推导之前我们拥有的所有公式仍是真的！意识到这一点是很重要的（在把*B*移到地面后，在状态*S0*中，*B*仍在*A*的上面。同样，在把*B*移到地板之前，在由do(move(B, A, Fl), S0)指称的状态中，B在地板上面。）状态演算中的公式在感觉上是“无状态的”，它们总是真值（它们谈论的状态）。

21.2 存在的一些困难

21.2.1 框架公理

图21-2明显地表明，并不是所有关于状态do(move (B, A, Fl), S0)为真的语句都能从效应公理中推导出。例如，在移动之后，很明显在移动之前的状态为真的事实，如*C*在地板上和*B*是空的，在移动之后状态仍为真值。典型地讲，动作只有“局部”影响，故留下很多流没有改变。为了对这些恒久不变的状态做推理，我们为每个动作和每个流提供一对所谓的框架公理(*frame axiom*)，它不会因动作的结果而改变。例如，{move, On}的框架公理是：

$$[On(x, y, s) \wedge (x \neq u)] \supset On(x, y, do(move(u, v, z), s))$$

和

$$(\neg On(x, y, s) \wedge [(x \neq u) \vee (y \neq z)]) \supset \neg On(x, y, do(move(u, v, z), s))$$

(如果在一个动作前一个积木在第二个积木上，那么在动作后它仍在第二个的上面，条件是那个动作没有把它从第二个积木上移走。如果动作前一个积木不在第二个积木上面，那么动作后它仍不在第二个上面，条件是该动作没有把它放在第二个上面)。

和效应公理类似，框架公理的第一个叫正框架公理(*positive frame axiom*)，第二个叫负框架公理(*negative frame axiom*)。

{*move, Clear*}的框架公理是

$$\mathtt{Clear(u,s)} \wedge (\mathtt{u} \neq \mathtt{z}) \supset \mathtt{Clear(u, do(move(x,y,z),s))}$$

$$\neg\mathtt{Clear(u,s)} \wedge (\mathtt{u} \neq \mathtt{y}) \supset \neg\mathtt{Clear(u, do(move(x,y,z),s))}$$

(假如动作前一个积木是 clear，如果那个动作没有把另一个积木放在它上面，那么该动作后那个积木仍是clear。如果动作前一个积木不是clear，而该动作没有从它上面移去另一个积木，那么该动作后它仍不是clear)。

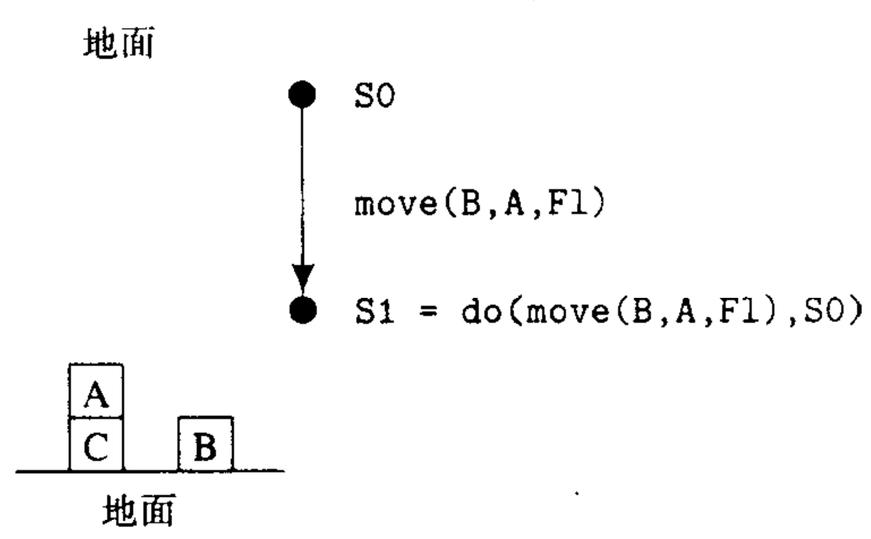


图21-2　将状态-动作对映射为一个状态

框架公理被用来证明如果状态由一个不影响某个属性的动作改变，那个属性保持为真。例如，{move, Clear}框架公理之一能用来推理图21-2中的clear (B, do(move (B, A, Fl)), S0))。典型地讲，由于对每一个流和动作的组合我们将有一对框架公理，在现实问题中用状态演算方式表达动作如何改变世界变得相当的难管理。

有几个作者已经探索了如何减少大量的框架公理，以及如何从效应公理自动导出框架公理。这里不想讨论这些技术，但是它们涉及到了那个假设——能施加于一个流的变化仅仅是那些被效应公理明确指定的变化（你可以参考[Pednault 1986, Schubert 1990, Reiter 1991, Elkan 1992]）。即使能够减少大量的框架公理，用它们对关于在几个动作序列上什么流不会改变做推理的计算仍是笨重的。各种与解决不受动作影响的流有关的困难被称为框架困难。下一章将讨论解决框架问题某些方面的一个方法。

21.2.2　条件

描述一个形如move的动作的转换公式的前提为一个相当理想的事件给出了一个前提条件。假如我们想通过确认被移动的对象不是太重而更准确一些，那么我们必须给前提条件加上另一个合取式¬ Too_heavy (x, s)。这个特别关注能被无穷无尽地进行，导致要加上其他的条件像 ¬ Glued_down (x, s)、¬ Armbroken (s), …，和那个明确的预期解释。指定所有的重要条件的困难称为条件问题。已进行使用非单调推理的很多尝试以解决条件问题。这个思想是效应公理允许缺省的结论，如果要加入进一步的条件，这个结论能被撤消（例如，参见[Dean & Wellman 1991, pp, 63以后]），这些尝试还没有完全成功。

21.2.3　分枝

还有其他的问题。在复杂的领域，我们常常用基于领域的一般知识演绎有关对象的语句。

例如，一个机器人可能推导出，如果它自己在一个房间里，那么它正在推的一个包也是在那个房间里。不用效应和框架公理推断包的位置，我们可能宁愿推理机器人的位置，然后用我们的普通知识去推理包的位置。例如，假如在状态S0我们有事实In(PA，R1)，其中PA指称某个包，R1指称某个房间。用它的效应公理，在移进某个不同的房间后（用R2表示），机器人能推断它（机器人）确实是在一个新的状态下。它还能推断出那个包也在一个新的状态下。但是现在我们如何才能阻止在新的状态下，框架公理断定那个包不是仍在由**R1**指示的房间里呢？跟踪导出的公式哪一个在随后的状态转换中幸存被称为分枝问题。已经提出各种机制（与真值维护过程有关）以解决分枝问题。

21.3 生成计划

让我们暂时不考虑框架、条件和分枝问题，从原则上说明状态演算如何用来规划一个使用推理方法的动作序列。为了生成到达某个目标$\gamma(s)$的一个计划，我们设法证明$(\exists s)\ \gamma(s)$，并用应答谓词提取状态作为嵌套动作的一个函数，那个动作产生了状态。

例如，假定我们想生成一个计划，它把积木*B*从图21-1中的初始状态*S0*放到地面上。为了计算这个计划，我们必须证明（∃s）On（B, Fl, s）。我们将用归结反驳法证明（∃s）On(B, Fl, s)的否定句（∀S）¬ On (B, Fl, s)与描述*S0*和移动结果的公式不一致。用一个应答谓词来捕获证明期间所做的置换。对这个问题的公式是：

$$\neg On(B, Fl, s) \vee Ans(s)$$

$$On(B, A, S0)$$

$$On(A, C, S0)$$

$$On(C, Fl, S0)$$

$$Clear(B, S0)$$

$$Clear(Fl, S0)$$

$$[On(x, y, s) \wedge Clear(x, s) \wedge Clear(z, s) \wedge (x \neq z)$$

$$\supset On(x, z, do(move(x, y, z), s))]$$

最后一个公式是{On,move}的正效应公理。在执行一个归结反驳法之前，它必须被转换成子句形式。像我们在归结反驳法中做的那样，单独留下等式谓词$(x \neq z)$（这是表达¬(x=z)的一种方式），直到它的所有变量被约束。然后，像前面讨论的那样，假定我们有无穷多的像（A=A）、¬(A=B）等等这样的公理可用在逆着它们的归结法中。将归结反驳法会产生Ans (do (move (B, A, Fl), S0)留给你去验证。

如果到达一个目标需要几个动作，动作函数将是嵌套的。例如，如果问题是从图21-1中的初始状态把*A*放在*B*上，我们将得到Ans (do (move（A, C, B）,do (move (B, A, Fl), S0))）。但是你将注意到，如果试图用效应和框架公理解决这个问题，证明工作对这个简单的计划是太多了。用状态演算产生计划的大部分努力都受到这类框架问题的困挠。

21.4 补充读物和讨论

条件问题首先由[McCarthy 1977]提出。使用非单调推理方法的各种努力试图解决框架和条件问题。[Hanks & McDermott 1986]发现，非单调结论关于哪些流改变了和哪些流保持没变是模糊的。以后由[Baker 1991]和[Shoham 1988]所做的工作讨论了这个问题（参见[Kartha 1994]

关于Baker方法的评论）。框架问题的另一个非单调方法使用一个“后继状态公理”限制和描述了能被任何动作施加于任何流的所有变化[Reiter 1991]。[Shanahan 1997]是一本关于框架问题的书。

虽然大多数当前有关agent计划的研究都使用了下一章描述的另一类方法，但状态演算和相关语言GOLOG（alGOl in LOGic）[Levesque, et al. 1997]在Toronto大学的认知机器人小组仍是机器人研究的基础[Scherl & Levesque 1993]。

习题

21.1 想像一个机器人*R*，在一个有两个盒子*B*1和*B*2的房子中。在初始状态，机器人在位置*PR*，*B*1在*P*1，*B*2在*P*2。在这个问题中，所有的三个实体（*R*、*B*1和*B*2）在同一位置是可能的。实际上，我们想让*B*1和*B*2在位置*P*3。机器人有两个动作，用goto(x)和push(y, w, z)表示。Goto(x)表示把机器人从任何地方移到x指示的位置；它没有任何前提条件。它的惟一结果是在执行那个动作后，机器人将处在x位置处。动作push(y, w, z)把机器人和由y表示的盒子移到由z指示的位置。只有当机器人和盒子都在相同的位置w时，该动作才能执行。它的结果是机器人和对象y都在z位置。

1) 用状态演算中的合式公式表达初始状态和目标条件。把初始状态公式和目标公式的否定形式写成子句形式。
2) 把两个动作的前提条件和结果表达为状态演算中的合式公式。把这些合式公式转换为子句形式。
3) 写出所有的框架公理，把它们转换成子句形式。
4) 勾画出一个步骤策略，导出一个从初始状态到达目标的计划。你实际上不必进行证明；仅仅用语言解释一下如何进行就行了。

21.2 想像一个屋中有一个机器人、一本书、一个桌子和三个不同的地方，分别叫做中间、壁橱和门口。机器人在门口，书在在桌子上，桌子在中间。机器人能在屋子的任何地方之间移动（即使那里已有东西）。机器人能把桌子从一个地方推到另一个地方，只要机器人和桌子在同一地方。

1) 用状态演算描述这个初始状态。
2) 用状态演算写出描述机器人移动结果的公式。
3) 用状态演算写出描述机器人推桌子的结果公式。
4) 写出需要的框架公理显示当机器人移动时书和桌子的位置不会改变。
5) 写出需要的框架公理显示当机器人推桌子时，有关书的位置的结果。
6) 按照初始状态和动作算子表达状态：书在桌子上，桌子在壁橱里。

21.3 建立Tower-of-Hanoi问题的一个状态演算公式（见习题7.4）。

1) 指定描述初始状态*S0*的公式。
2) 提出一个law-of-motion公式，描述一个移动结果，而且至少写一个与移动效果有关的框架公理。
3) 写出指定目标的公式。

21.4 考虑下面的积木世界问题：有两个积木（*A*和*B*）和三个位置（*L*1、*L*2和*L*3）。公式At(x, y, s)的预期含意是指在状态s，积木*x*在位置*y*。公式Empty(z, s)的预期含意是指

在状态s，位置z是空的。我们有一个简单的动作：move（x, y, z），它指的是把积木x从y移到z。我们假定move有一个前提条件是一个积木正被移过去的位置必须是空的。做下面的工作：

1) 写出move的算子描述和框架公理。
2) 描述一个初始状态：*A*在*L*1，*B*在*L*3。
3) 描述目标：*A*移到*L*2，*B*在*L*3不动。
4) 用归结和Ans谓词生成一个动作序列，当在初始状态执行时，会满足刚刚描述的目标。对不相等的事情可以做任何假设，如（B ≠ A），你可能在证明中需要它们

第22章　规　　划

22.1　STRIPS规划系统

22.1.1　描述状态和目标

解决框架问题的一个比较好的办法是混合状态空间和状态演算方法。这种混合涉及到把描述一组世界状态的谓词演算公式想像成一种“状态”——就像在第7章讨论基于特征的状态空间时所做的一样。例如，在积木世界中，*B*在*A*上，*A*在*C*上及*C*在地面上（见图22-1）的世界状态可以描述为：

```
On(B, A)
On(A, C)
On(C, Fl)
Clear(B)
Clear(Fl)
```

这些公式描述了一组世界状态，因为它们被所有的那些状态满足，在这些状态中由公式意指的关系都是真的。如果有其他的积木，积木有其他的属性，如颜色，那么正被描述的状态将包括所有的那些东西，在它们中有这些其他的积木，积木有所有的可能颜色等等。

在本章描述的第一种规划方法中，把描述一组世界状态的公式作为一个能够用相应的agent动作改变的数据结构。将描述一些对这些数据结构的空间进行搜索的方法，以便发现一个结构，它描述了一组满足一个目标的世界状态。在一个对这些数据结构上的状态空间进行的搜索中，数据结构是规划过程的状态。为了区分规划状态和世界状态，一般把前者称为状态描述。

为了避免各种技术困难——在此我们不必考虑它们，把动作前后的状态描述限制为基本的文字合取（或集）。对这种限制允许一个例外，即状态描述能包含所有状态中的任何公式真值——像$On(x, y) \supset (y=Fl) \vee \neg Clear(y)$。把目标约束为文字合取（可能存在的量化）。下面把（$\exists x1, x2,...,xn$）$\gamma(x1, x2,..., xn)$形式的目标合式公式写成简单的$\gamma(x1, x2, ..., xn)$——假定存在所有变量的量化。尽管在多种方法中都有这些约束，但几个有趣的规划问题能被形成，并适当地求解。

给定一个目标合式公式γ，我们的搜索方法企图发现一个动作序例，它产生一个由某些状态描述S，如$S \vdash \gamma$描述的世界状态，那么我们就说那个状态描述满足那个目标。对受限状态和目标合式公式$S \vdash \gamma$，当存在一个S替换时，这样γS就是一个基本的文字合取，它的每一个文字都出现在S中。通过试图合一γ中的第一个文字和S中的一个文字，不管能否建立$S \vdash \gamma$，都把合一置换应用到γ中的其他文字并继续对γ中的所有文字进行（这个过程与被PROLOG解释器使

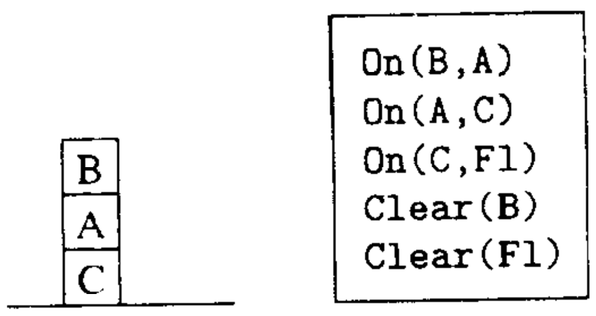


图22-1　一个状态描述

用的相同，PROLOG 解释器检查一个子句体是否与事实统一）。

我们可以用从开始到目标的向前搜索，或者用从目标回到开始的向后搜索来对一个状态描述空间进行搜索。有些作者把使用向前搜索的规划方法称为进步规划（*progression planning*），把使用向后搜索的方法叫回归规划(*regression planning*)。下面先讲向前搜索。

22.1.2 向前搜索方法

为了在状态描述空间上进行向前搜索,我们将需要与动作对应的算子，它把一个动作前状态描述改变成一个动作后状态描述。当它产生一个状态描述 S，使得对目标合式公式 γ，有 $S \vDash \gamma$，则搜索成功完成。我们的算子是基于一个叫做STRIPS的系统[Fikes & Nilsson 1971, Fikes,Hart & Nilsson 1972]。一个STRIPS算子由三部分构成：

1) 集合*PC*，称为算子的前提条件（*preconditions*）的基本文字。相应于一个算子的动作只有*PC*中的所有文字都在动作前状态描述中时才能被执行。

2) 集合*D*，称为删除列表(*delete list*)的基本文字。

3) 集合*A*，称为加入列表(*add list*)的基本文字。

为了生成动作后状态描述，我们首先从动作前状态描述中删除*D*中的所有文字，加入*A*中的所有文字。在*D*中没有提到的所有文字延续至动作后状态描述中。这种延续被叫做STRIPS假设，它是解决框架问题的一种方法。

STRIPS算子常常用与动作模式相符的模式定义。一个称为STRIPS规则的算子模式有自由变量，它是那些应于实际算子规则的基例。下面是STRIPS规则的一个例子，它带有自由变量x、y和z：

```
move(x,y,z)
PC: On(x,y) ∧Clear(x) ∧Clear(z)
D: Clear(z) On(x,y)
A: On(x,z),Clear(y), Clear(Fl)
```

(把公式Clear(Fl)显式地包括在加入列表中，因为如果z是Fl，Clear (Fl)将被删除，而我们希望它总是真的)。这个STRIPS规则的一个实例（把积木B从A上移到地面）显示在图22-2中。

注意，一个STRIPS规则本身的前提条件是在一个叫做文字合取的目标形式中。这不是一个巧合且将被利用。当所解释的是一个目标合式公式时，*PC*中的自由变量被假定是存在量化的。一个STRIPS规则的实例能被应用到一个状态描述 S ——如果*PC*（被作为一个目标）的一个基例被状态描述满足。如前面所讲，这样一个实例能用合一发现；反过来，*PC*中的每个文字在状态描述中有一个文字——把合一置换应用到*PC*中的其余文字。然后通过把结果置换应用到STRIPS规则的所有元素，从而获得可用的算子实例。规则必须用这样一种方式书写，在这种方式中，应用这样一个置换总是产生规则的一个基例。即，在*D*或*A*中不可有任何不在*PC*中出现的自由变量。图22-2是一个STRIPS算子应用的示例。

在向前搜索方法中，通过应用STRIPS规则实例直到产生一个满足目标合式公式的状态描述，生成一个新的状态描述。在图22-3中，显示了由向前搜索生成搜索图的一部分。由于一个状态描述中的大部分文字不会被STRIPS算子改变，通过只跟踪变化的部分，在向前搜索的实

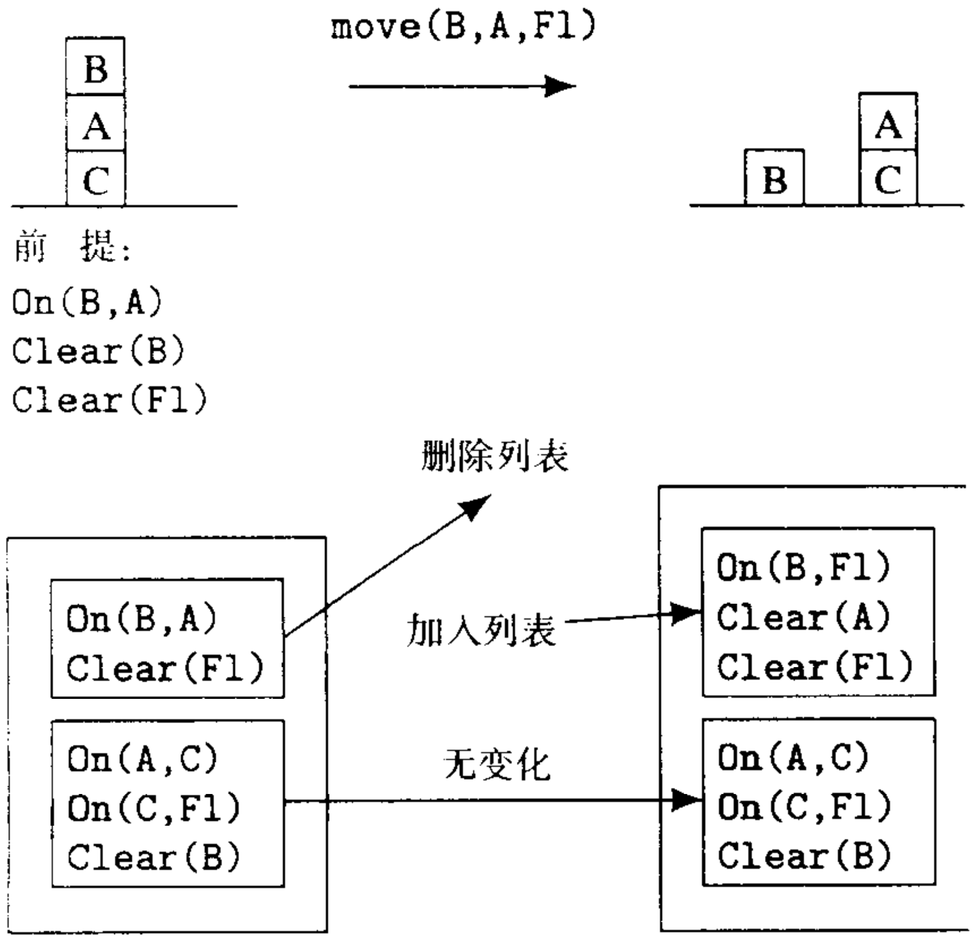


图22-2　一个STRIPS算子

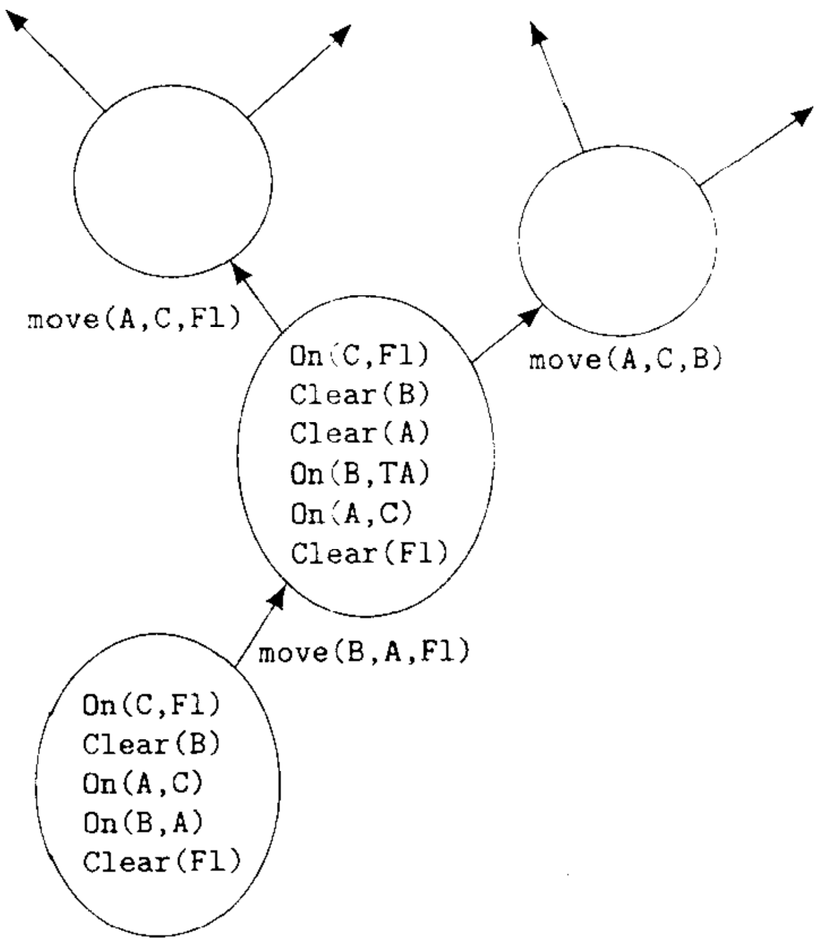


图22-3　向前搜索

能使用A* 搜索）。注意，和状态演算不同，我们不必证明动作留下了一些没有改变的关系，STRIPS假设已经考虑了这些问题。

没有关于应用哪个规则到哪个实例的启发性，向前搜索在那些状态描述和规则数量巨大的实际应用中是不现实的。使向前搜索更有效的一个方法是在搜索空间中确定岛，把到达目标状态描述的搜索路径集中地朝向岛。在下面的小节中，描述了一个标识和利用岛的方法。

22.1.3　递归STRIPS

当一个目标条件包含一个文字合取项时，像这里假定的一样，我们能通过一个分治启发式方法集中向前搜索。首先，获得一个合取项，然后另一个，等等继续下去。一个合取项被满足的状态描述表示搜索空间中的一个岛。我们先朝着那个岛前进，应用算子产生一个对应合取项被满足的状态描述，然后继续朝着另一个岛前进，等等。当然，可能会在获得随后的一个合取项时不能获得前面的合取项。

AI历史的早期开发了一个叫做通用问题求解程序（*General Problem Solver*，GPS）[Newell, Shaw, & Simon 1959, Newell & Simon 1963]系统，它利用了这种方式下的搜索空间岛。到达这些岛的算子由一个叫做“中间结局分析（*means-ends analysis*）”的过程确定。当应用到使用STRIPS规则的系统时，这个技术涉及到选择目标条件中合取项之一的一个实例和选择到达目标的一个STRIPS规则的一个实例。接下来，我们为应用那个规则的前提条件建立一个子目标，用确定孤岛的同样方法递归地朝着那个目标前进直到它被满足；然后，应用算子建立孤岛状态描述并继续标识另一个孤岛，等等。这个过程是我们称做的STRIPS递归程序的基础。下面是STRIPS如何求解一个合取目标公式γ:

STRIPS（γ）：这个过程使用包含一个基本文字集合的一个全局数据结构S。这个数据结构开始时设置为初始状态描述，并在过程进行中或之前改变。

1) repeat　STRIPS的主要循环是迭代和继续直到产生一个满足目标γ的状态描述。第9步

的终止测试产生一个置换σ（可能是空的），以便$\gamma\sigma$的一些合取项（可能没有）出现在第8步中。在执行测试中可能有几个试验置换，因此测试是一个可能的回溯点。

2) $g \leftarrow \gamma\sigma$的一个元素，以使$S_1 \neq g$。这是另一个选择，因此也是一个回溯点。在“中间结局分析”方法中，g被作为一个“差别”，它必须被“缩小”以到达目标。

3) $f \leftarrow$ 一个STRIPS规则，它的加入列表包含一个文字λ，它把g和mgu s统一起来。由于可能有几个这样的规则，这是另一个回溯点。f是一个与减少差别“有关”的算子。

4) f' $\leftarrow fs$，使用置换s的f实例。注意，f'不必是一个基例，因此它的前提条件可以包含变量。

5) $p \leftarrow$ f'的前提条件公式（用置换s实例化）。

6) STRIPS（p），一个递归调用以产生一个满足子目标的状态描述。这个调用通常会改变S。

7) f' $\leftarrow$ 可以在8中应用的f'的一个基例。

8) $S \leftarrow$把f″应用到S的结果。注意S总是包含一个基本文字合取。

9) until $d \models \gamma$

注意，在第9步，我们总是相对整个目标γ进行测试。如果γ的一个合取项在到达另一个合取项的过程中没有到达，那么在过程终止前第一个将必须被重新到达，或者回溯该过程，以寻找一条能到达第一个合取项的一条路径。

虽然在算法描述中没有明确提及，但构成一个计划的目标算子序列能从那个算法的成功执行记录中提取出来。它们依次对应agent能执行的动作，或者是“发射导弹似地（*ballistically*）”（在适当的情况下），或者在一个感知/计划/动作循环中。

下面的例子演示了算法是如何工作的。假如开始有图21-1中描述的状态，我们想获得目标On (A, Fl) ∧ On(B, Fl)∧On (C, B)。第9步的目标测试没有发现任何被初始状态描述满足的合取项。假定STRIPS选择On（A，Fl）作为g，规则实例move(A, x, Fl)在它的加入列表中有On(A,Fl)，因此我们递归地调用STRIPS获得规则的前提条件Clear (A) ∧Clear (Fl) ∧ On(A, x)。递归调用中第9步的测试产生置换C/x，它留下了除满足初始状态的Clear(A)以外的所有合取项。选择这个文字，并选择规则实例move（y，A，u）以获取该文字。

再次递归调用STRIPS以获得规则的前提条件Clear (y) ∧ Clear (u) ∧ On(y, A)。这个第二次递归调用中的第9步测试产生置换B/y和Fl/u，它使前提条件中的每个文字成为初始状态中出现的文字。因此，我们能退出第二个递归调用，并应用一个算子move (B, A, Fl)，这把初始状态改变成：

On(B, Fl)

On(A, C)

On(C, Fl)

Clear(A)

Clear(B)

Clear(Fl)

现在，回到第一次递归调用，再次执行第9步的测试（即Clear (A) ∧ Clear (Fl) ∧On (A, x)）。这个测试被我们用置换C/x改变的状态描述所满足，因此退出第一个递归调用应用算子move(A, C, Fl)，产生第三个状态描述：

On(B, Fl)

On(A, Fl)

On(C, Fl)

Clear(A)

Clear(B)

Clear(C)

Clear(Fl)

现在再次执行主程序中的第9步测试。在初始状态中惟一没有出现在新状态描述中的合取项是On(C, B)。再次从主程序递归，我们注意到move(C, Fl, B)的前提条件已被满足，因此应用它生成一个满足主目标的状态描述，过程终止。

22.1.4　带有运行时条件的计划

如果在计划执行期间能通过知觉处理精炼那些状态描述，我们就能让状态描述中允许的各种公式稍稍一般化。考虑在一个状态描述公式中允许文字析取意味着什么。为了清楚地说明，假定在一个状态描述中有合式公式 On(B, A) ∨On(B, C)。这个合式公式的预期含意是*B*在*A*上，或者*B*在*C*上，但是系统不知道当时正在产生哪个计划。一些STRIPS算子对这样一个状态描述的应用及这些算子的结果可能并不依赖于On(B, A)被满足还是On(B, C)被满足。对这些算子而言，在状态描述中有一个析取式是无所谓的，析取式将仅传给动作后的状态描述。

但是有的算子可能把On(B, A)或On(B, C)作为一个前提条件。为了执行和这种算子对应的动作，系统必须知道在执行动作时（即在运行时）哪一个析取项被满足。也许知觉过程能做出这个决定。如果这样，一个运行时条件能在这样一个算子的应用点被插入那个计划中。在构造随后的运行时条件计划中，计划过程分割成与满足算子前提条件的析取项一样多的分枝，每个分枝保持状态描述和算子的一个独立线路。在例子中，一个分枝将假定当执行随后的动作时，On(B, A)是（或将是）世界的真值，另一个将假定On(B, C)是（或将是）世界的真值，这些分枝中的每一个被称为一个上下文。在制定计划时，系统可能不知道在上下文分离时哪个状态描述描述了世界的真正状态，但它们中肯定有一个是。对每个上下文构造可选的完成计划，然后，在运行时当系统碰到分裂成两个或更多上下文时，知觉过程决定哪个析取项是真的，执行过程就顺着合适的路径向前推进。在复杂问题中，可能有几个分枝上下文。

有时一个知觉过程能够决定在需要信息时哪个析取项是真的，这种情况仅作为前面动作的结果发生。更典型地讲，系统必须通过使用与信息收集动作相应的算子进行计划以获得那个信息。在例子中，为了决定On(B, A)与On(B, C)哪一个是真的，系统可能必须执行能读出积木上字母的动作。这种动作能用STRIPS算子描述，算子的结果包含一个正式指定“知道哪一个”的方法。然后运行时条件让这些动作知道哪一个是前提条件。

22.1.5　Sussman异常

考虑图22-4中显示的将STRIPS规则递归地应用到积木世界的问题。目标条件是On(A, B)

∧On(B, C)，STRIPS必须首先获得一个合取项。假定它选择了On(A, B)，并把*C*从*A*上移到地板上，然后将*A*移到*B*上。但在获得另一个合取项On(B, C)时，它将取消On(A, B)且必须再次到达它。假定它首先选择了On(B, C)。它把*B*移到*C*上，但在获得另一个合取项On(A, B)时，它将取消On(B, C)且必须重新到达它。在一个计划中没有一个可选结果有最少数量的算子。这个特殊的积木问题由Sussman提出 [Sussman 1975]，之后被叫做Sussman异常。

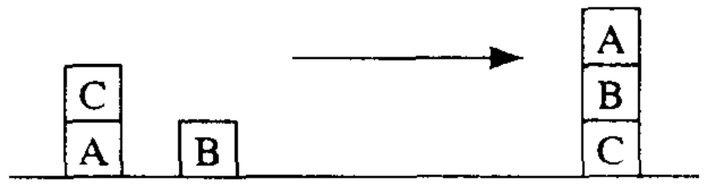


图22-4 Sussman异常

这个问题中递归STRIPS的困难是它用一种深度优先方式，每次仅仅集中在一个合取项上。如果向前搜索使用一个广度优先策略，它将发现最短计划。但是如已指出的那样，在实际问题中广度优先向前搜索在计算上是不可行的。

一种可选方案是尝试一个广度优先的向后搜索，也许从计算上讲，它比向前搜索效率更高，因为目标合式公式中的合取项比初始状态描述中的基本文字要少一些。下面将讨论向后搜索。

22.1.6 向后搜索方法

我们能用STRIPS规则从目标状态向后搜索。为此，必须通过STRIPS规则倒退目标合式公式，产生子目标合式公式。把公式γ通过一个STRIPS规则α倒退到最弱公式γ'，以便如果γ'在应用一个α实例前被一个状态描述满足（且γ'满足α的那个实例的前提条件），那么γ被应用α的那个实例后的状态描述所满足（如果$\phi_2 \models \phi_1$，则公式ϕ_1比ϕ_2弱。因此P∨Q比P弱，P比P∧Q弱）。

如果用一个基本文字合取式开始，且回溯仅通过STRIPS算子（STRIPS规则的基例），那么倒退计算是直接的。在这种情况下，所有的倒退将也是基本文字的合取式。在图22-5中，显示了一个例子，它用该方法从一个目标描述向后搜索。此处的问题是从状态*B*在*A*上、*A*在*C*上、*C*在地面上，到达状态*A*在*B*上、*B*在*C*上、*C*在地面上。在这个问题中，通过得到一个目标合取项的任何算子倒退目标On(C, Fl) ∧ On(B, C) ∧ On(A, B)。假设通过move(A, Fl, B)倒退产生一个子目标合式公式，算子获得一个合取项On(A, B)，因此On(A, B)不必在子目标中，但是不在目标描述中的任何算子的前提条件必须在子目标中。在这种情况中，有Clear(B)、Clear(A)和On(A, Fl)。在目标合式公式中的另两个合取项On(C, Fl) 和 On(B, C)既没被算子加入也没被破坏，因此它们仅仅通过算子传递到子目标中，如图22-5中所示。另一种向后搜索方法是通过move（B，Fl、C）倒退目标合式公式。向后搜索继续进行（也许使用某个类似A*的方法）直到产生一个被当前状态描述满足的子目标。

如果一组文字通过一个删除其中有这些文字之一的算子倒退，注意会发生什么。如果一个文字λ被一个算子删除，则该算子没有任何方法可以获得文字λ！因此，任何包含λ的合取式通过一个删除了λ的算子进行倒退都是*F*，当然，搜索没有必要从一个包含F的状态描述（这样一个状态是不可能到达的）向后跟踪。

通过一个没有完全实例化的规则倒退一个目标常常是明智的——也就是说，通过一个仍然包含着一些模式变量的规则。在图22-5中，我们考虑用move（A，Fl，B）作为到达On(A, B)的一个方法。为什么决定从地面移走*A*呢？的确，我们必须从某个地方移动它，但是为什么确定要从地面移动它呢？也许有更好的计划将它从其他某个地方移动。延迟这个决定的一种方法

是把“从……地方”留给移动算子暂时不指定。这样做是所谓的最小承诺规划（*least commitment Planning*）的一个实例。通过部分实例化的算子move(A, x, B) 倒退目标，留下“从……地方”不去指定。在图22-6中，给出了这个倒退的结果。除了现在的一个前提条件On(A, x)有一个变量外，这个结果更像早期得到的结果。如果这个文字与一个状态描述中的一个基本文字相一致，它就能被满足，可以解释为一个子目标。通过一个包含变量的子目标空间的向后搜索必须允许附加的算子对变量进行实例化——用其他变量或者用常量。一个启发式搜索将延迟实例化直到包含它们的文字能与初始状态描述中的文字合一。

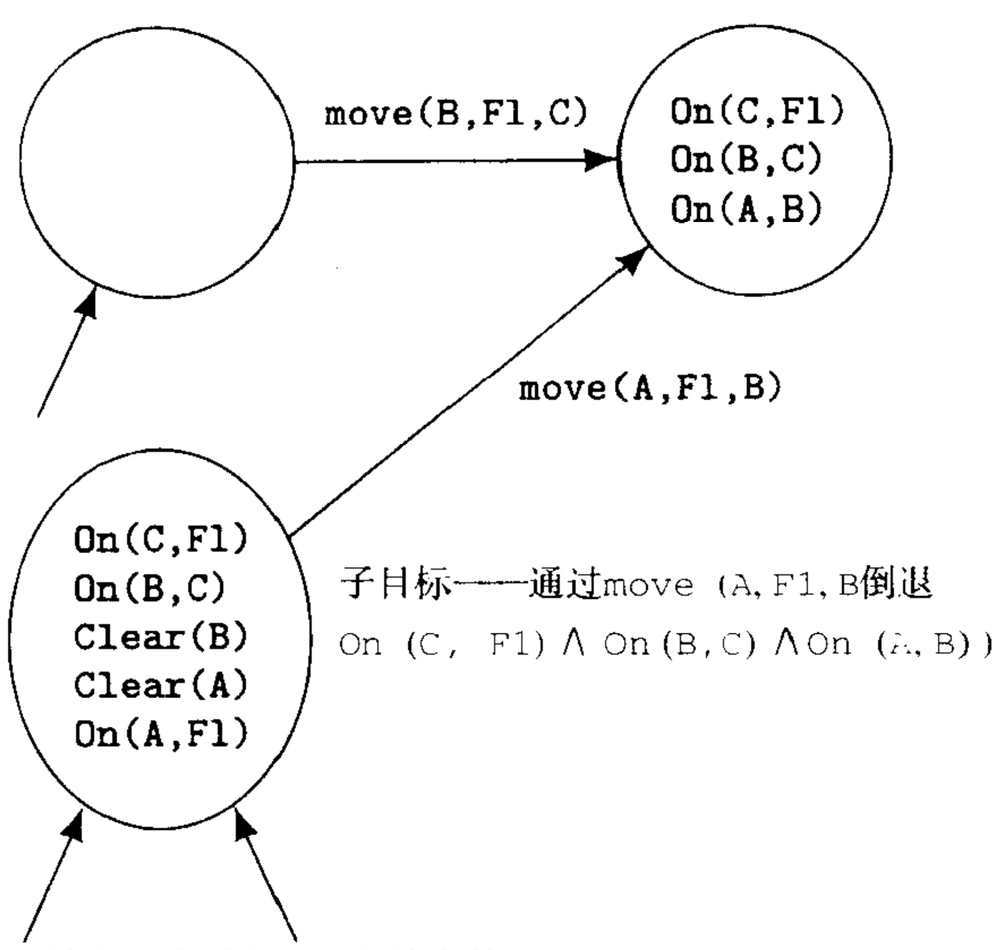


图22-5　通过一个STRIPS算子倒退一个合取式

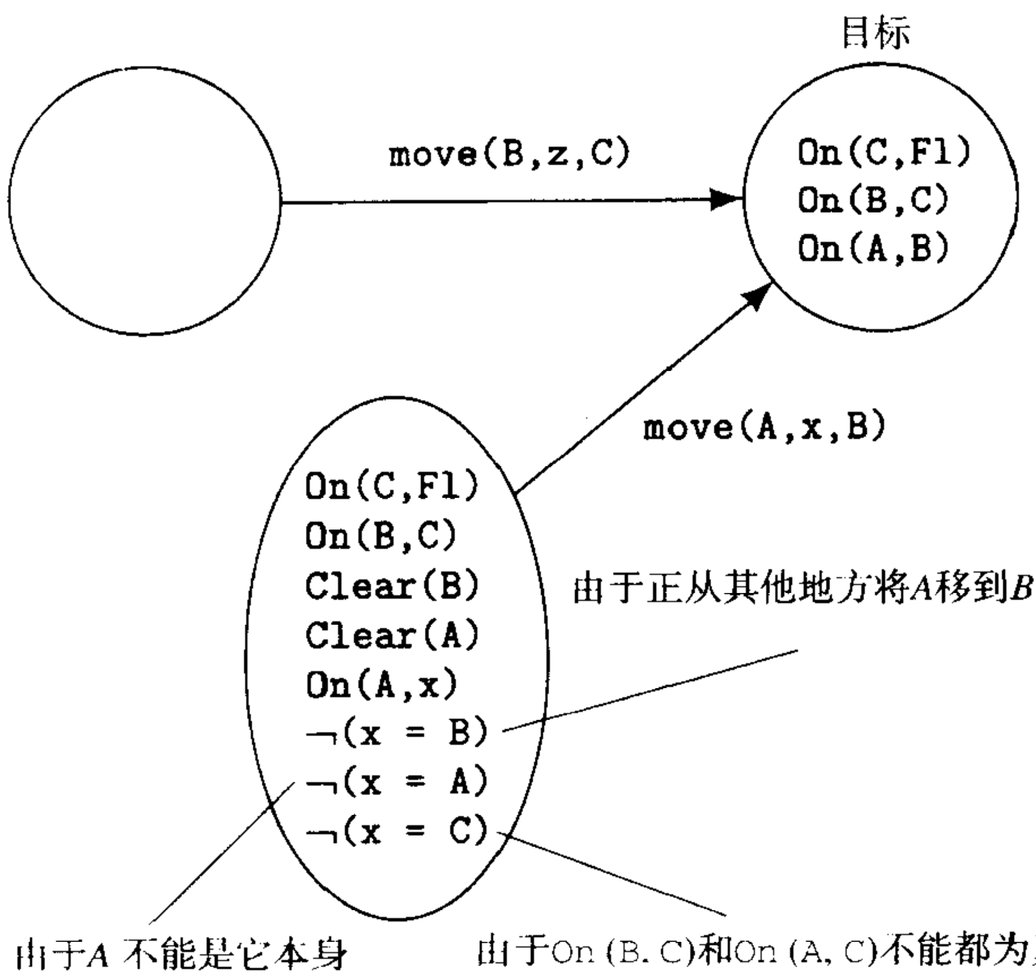


图22-6　通过一个包含一个变量的STRIPS规则倒退一个合取式

有时可以用各种推理过程来约束变量。在图22-6所示的例子中，On(A, x)中的变量x不能等于A，因为它是我们正在移动的积木*A*。x也不能是B，因为我们正把*A*从另外的地方移向*B* x也不能是C，但推理就有点微妙了。如果x是C，我们把On(A, C)作为子目标一个的合取项。但是，目标中的On(B, C)不能倒退子目标的On(B, C)，因为*B*和*A*不能同时在*C*上（注意，用在递归调用STRIPS中的子目标不同于由倒退产生的子目标！）。

倒退目标的全过程和通过没有完全实例化的STRIPS规则产生子目标是相当复杂的——在[Nilsson 1980 pp.288以后]中解释得更详细一些。图22-7是一个向后搜索的例子。在这个例子中，变量通过推理被实例化为F1，因为它们的约束排除了惟一可能的选择实例。当产生一个满足初始状态描述的子目标时搜索终止。连接主目标和满足初始状态的子目标的序列能被读出来并被实例化为一个获得目标的计划算子，该计划由执行搜索中发现的实例化生成。[Nilsson 1980, pp 292 ~ 296]给出了一个更复杂的例子。

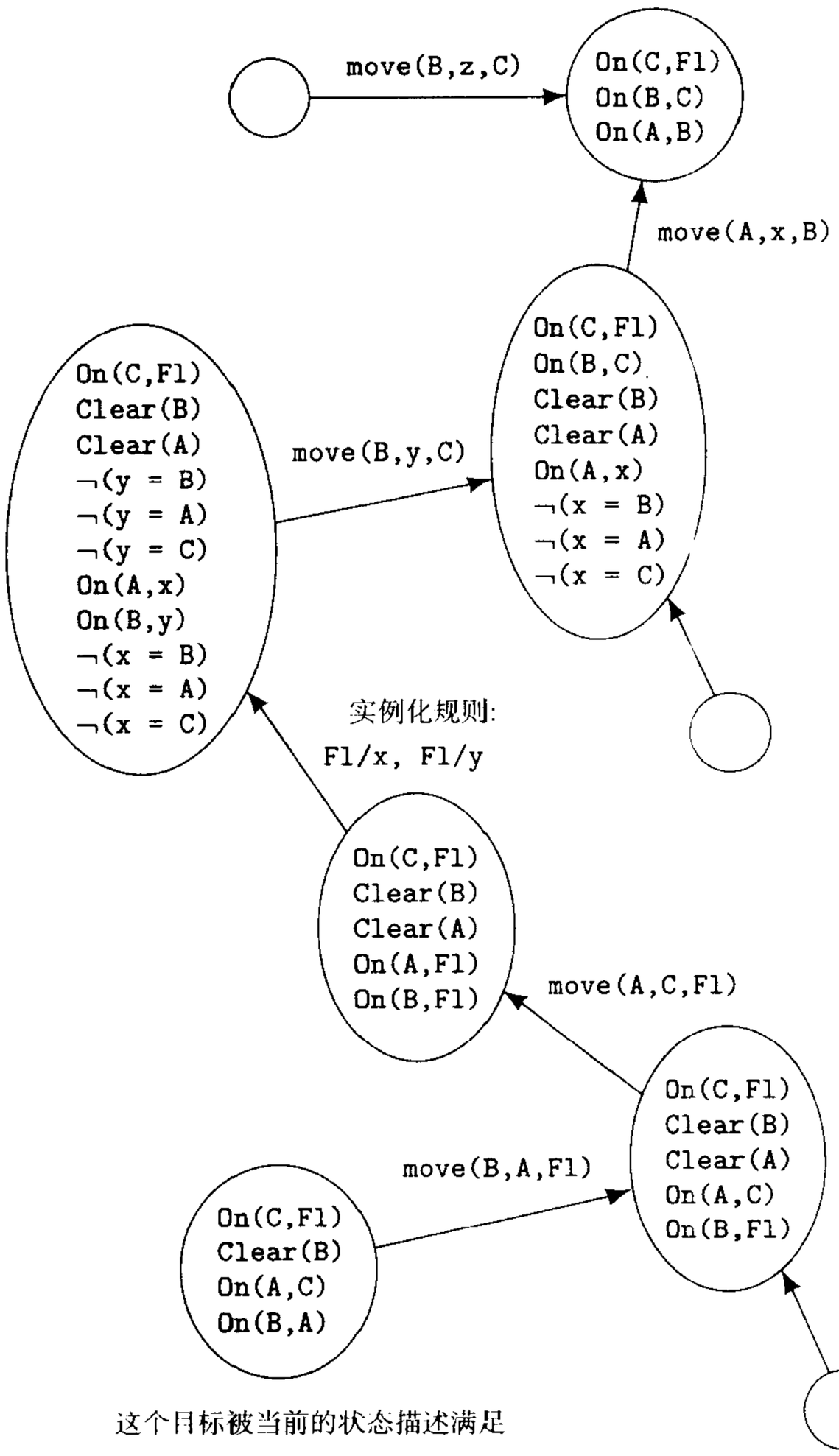


图22-7 向后搜索

使用倒退方法的向后搜索，虽然可能比向前搜索更有效，但由于出现变量而复杂化了。在小的积木问题中，变量能被实例化而不会有太多的困难，但是在很大的、没有使用指定领域启发的问题中，该方法在计算上是否可行是值得怀疑的。

在广度优先向前或向后搜索中，在任何特殊顺序中，没有承诺保证到达目标合取项。Sussman异常是一种问题的一个例子，在这种问题中，最好把解决每个合取项需要的步骤进行交叉。延迟提交最终规划中的步骤顺序是最小承诺规划的另一个实例，它能通过在一个规划空间而不是公式空间中进行搜索而获得最佳。在“计划空间”中，一个计划的步骤可以部分有序。在下一部分讨论这个叫做部分有序规划(*partial-order planning*，*POP*)的计划策略（部分有序计划有时也叫做非线性计划）。

22.2　计划空间和部分有序规划

图22-8中给出了两种不同的计划产生方法，当在一个公式空间中搜索（向前）时，STRIPS规则被应用到公式集合以产生它们的后继，这样一直进行，直到产生一个满足目标公式的状态描述。但是在计划空间中搜索时，后继算子不是STRIPS规则，而是能把不完全的、未实例化的或其他不合适的计划转化为表达力更强的计划的算子，等等，直到产生一个能把初始状态描述转换为一个满足目标条件的状态描述的执行计划。在计划空间搜索中的算子能通过各种方式把计划转换为其他的计划。这些包括：(a)给计划加一些步骤；（b）对已在计划中的步骤重新排序；（c）把一个部分有序的计划改变成一个完全有序的计划；（d）把一个计划模式（带有未实例化的变量）改变成那个模式的某个实例。图22-9给出了这些计划转换算子的一些例子。

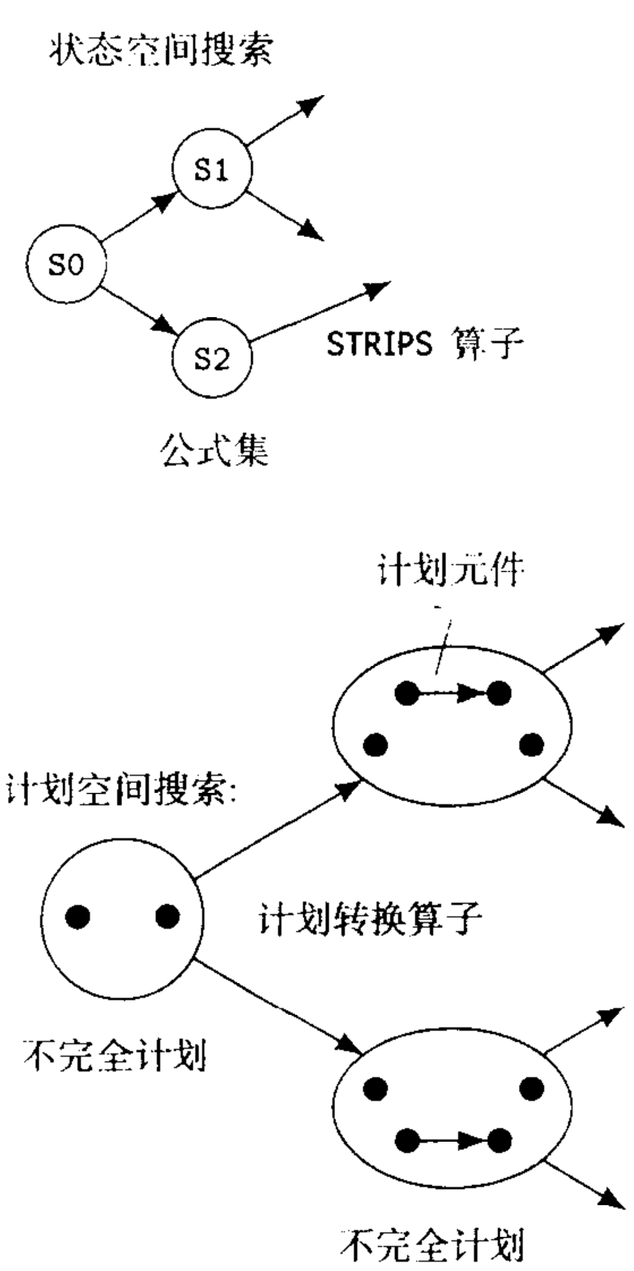


图22-8　状态空间与计划空间搜索

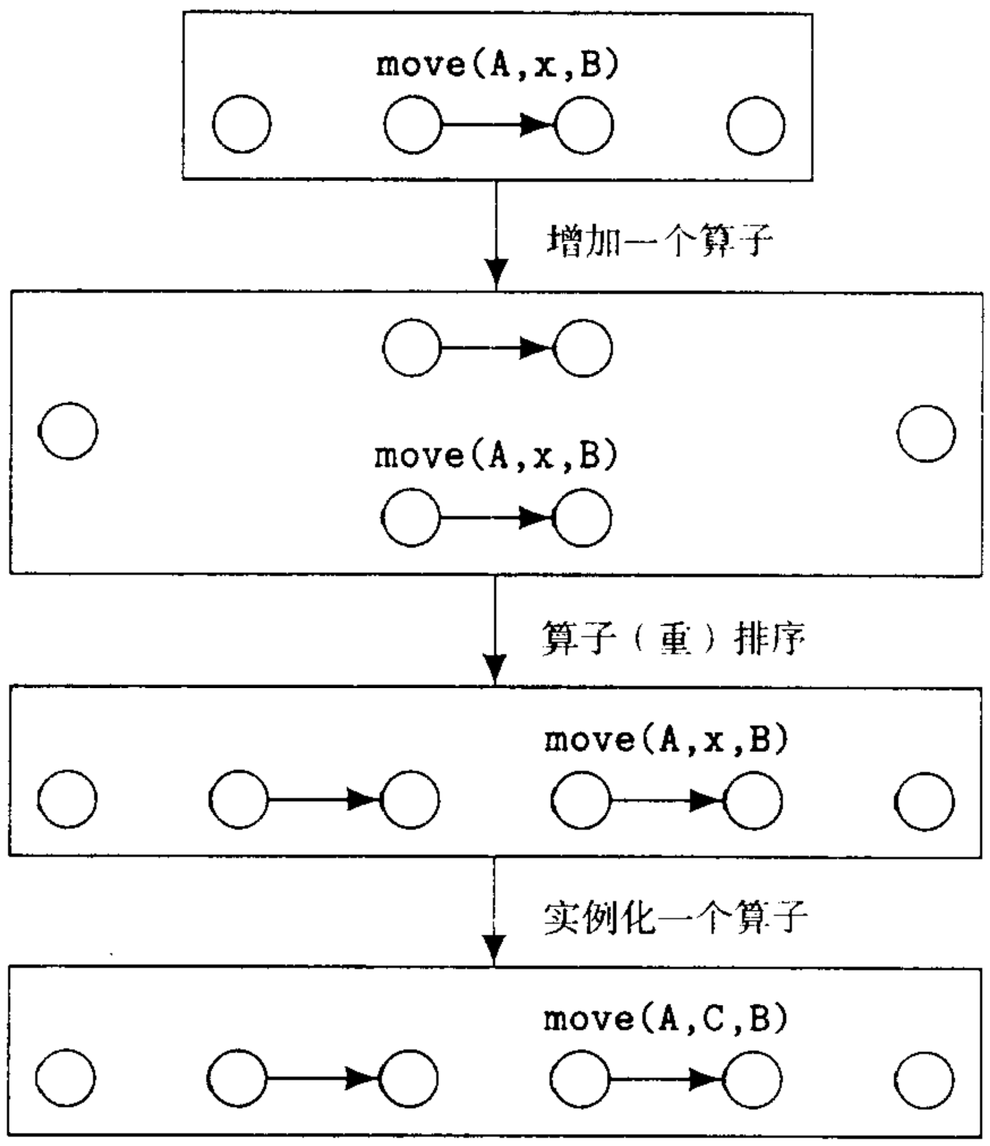


图22-9　一些计划转换算子

让我们说明一下应用到Sussman异常（图22-4）的计划空间搜索方法。描述基于[McAllester & Rosenblitt 1991]系统的非线性计划（SNLP）技术和[Tate 1977]的NONLIN。一个计划的基本构成是STRIPS规则，它们被有两类节点的图结构表示：随圆型节点用STRIPS规则的名字标识，长方形节点用那些规则的前提条件和结果标识。这样一个图结构显示在图22-10中。注意，图顶部的长方形是结果。底部的长方形是前提条件。

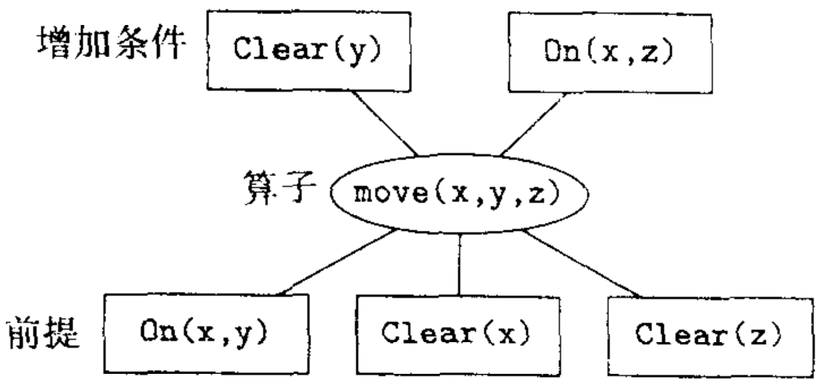


图22-10 一个STRIPS规则的图形表示

我们通过假想的、称为finish和start的规则图表示目标合式公式和初始状态的文字。规则finish用总目标作为它的前提条件，它的结果是nil。算子start用初始状态描述中的所有文字作为它的结果；它的前提条件是T。在图22-11中示例了Sussman异常的这两个图。

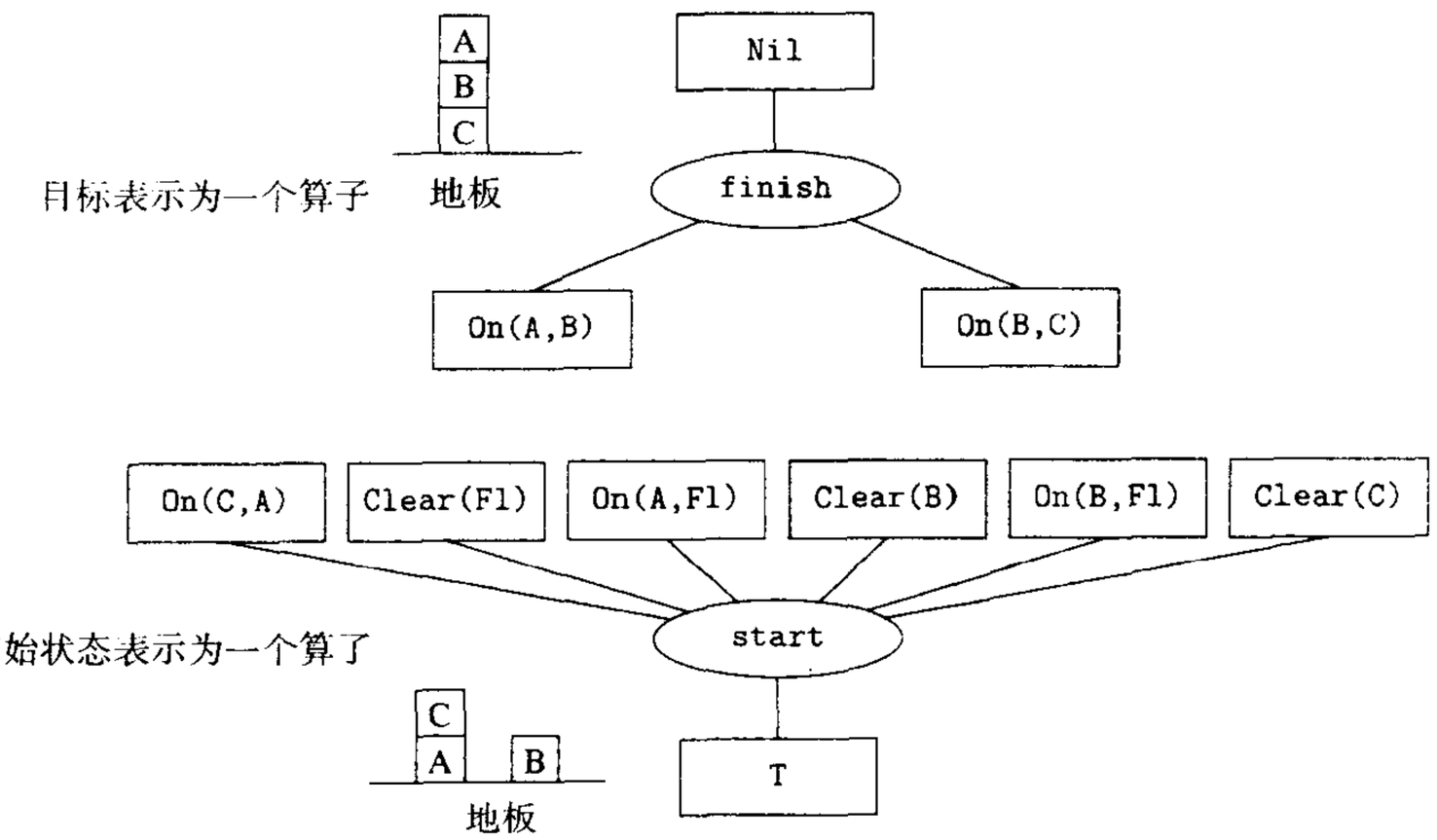


图22-11 finish和start规则的图形表示

初始计划结构（在我们应用任何计划转换算子之前）仅包括图22-10所示的（没有连接的）start和finish规则。当然它是相当不完全的（将定义它对一个后面要完成的计划的意义）。

现在一个计划的搜索通过把一个计划转换算子应用到初始计划结构开始，最小承诺计划者用各种方法选择计划转换算子。在这个阶段可能被选择的一个转换通过增加一个规则以到达目标的一个合取项来扩展初始计划结构。假定我们决定通过增加规则实例move(A, y, B)以到达On(A, B)。把这个规则的图结构加到初始计划结构中，由于已决定这个规则的加入会到达On(A, B)，我们把这个规则的结果框与finish规则相应的前提条件框相连接，其结果如图22-12所示。注意我们还没有承诺A从那个位置移走。

在这个阶段可能有几个计划转换。我们能把y实例化为Fl，并把它放在一个在实例化的move算子前提条件On(A, Fl)和在初始状态为真的On(A, Fl)之间的连接中。我们也能放进一个在两个Clear (B)之间的连接中，或者设法建立move(A. y, B)的Clear (A)前提条件，这个条件能通过插入规则move (u, A, v)（移动A上的某个东西（u的一个实例）到另外某个地方（v的一个实例）建立。然后把u实例化为C，v为Fl，并建立与初始状态中的公式相对应的连接。这一系列计划变更步骤的结果将产生图22-13中的计划结构。

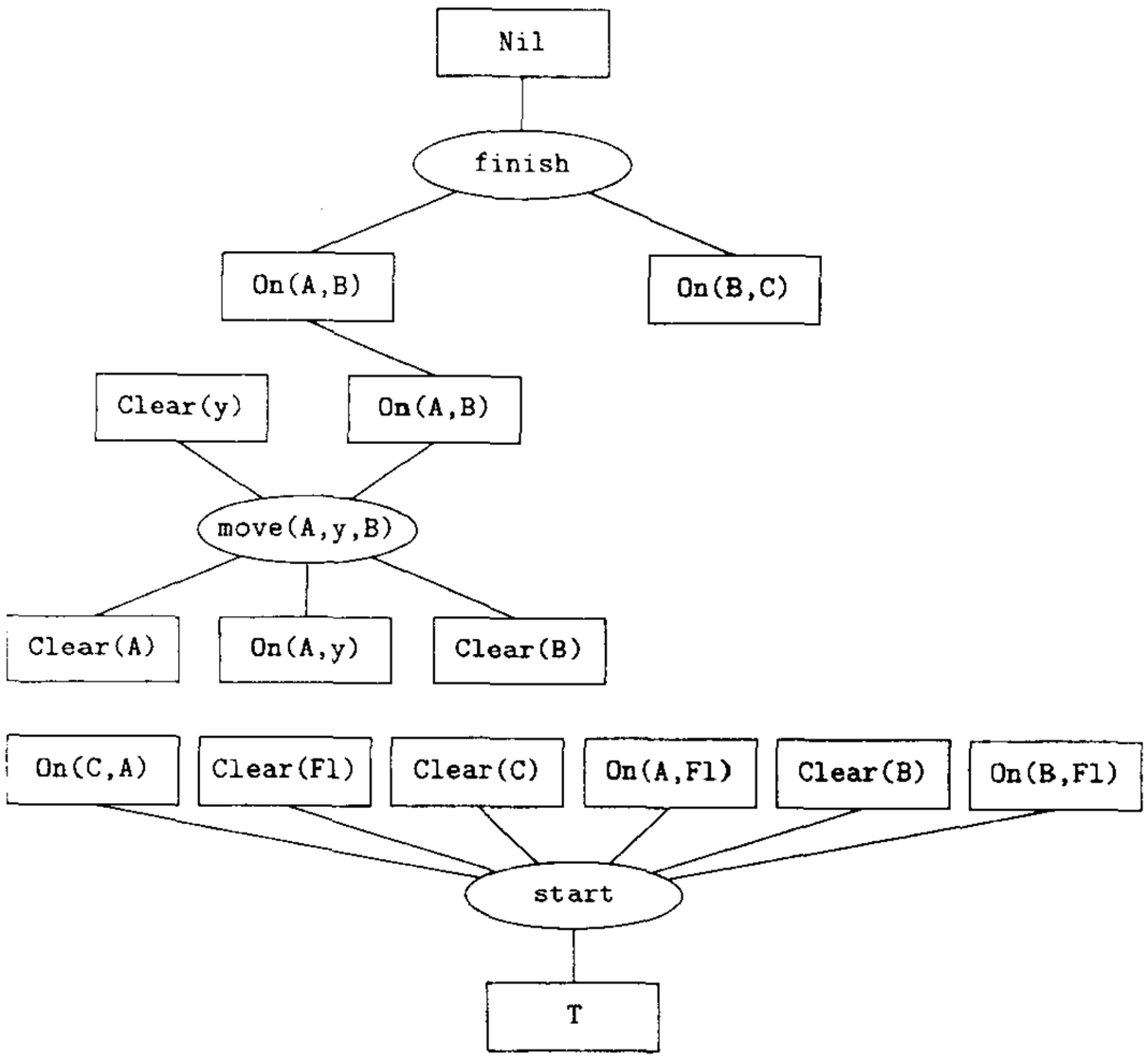


图22-12　下一个计划结构

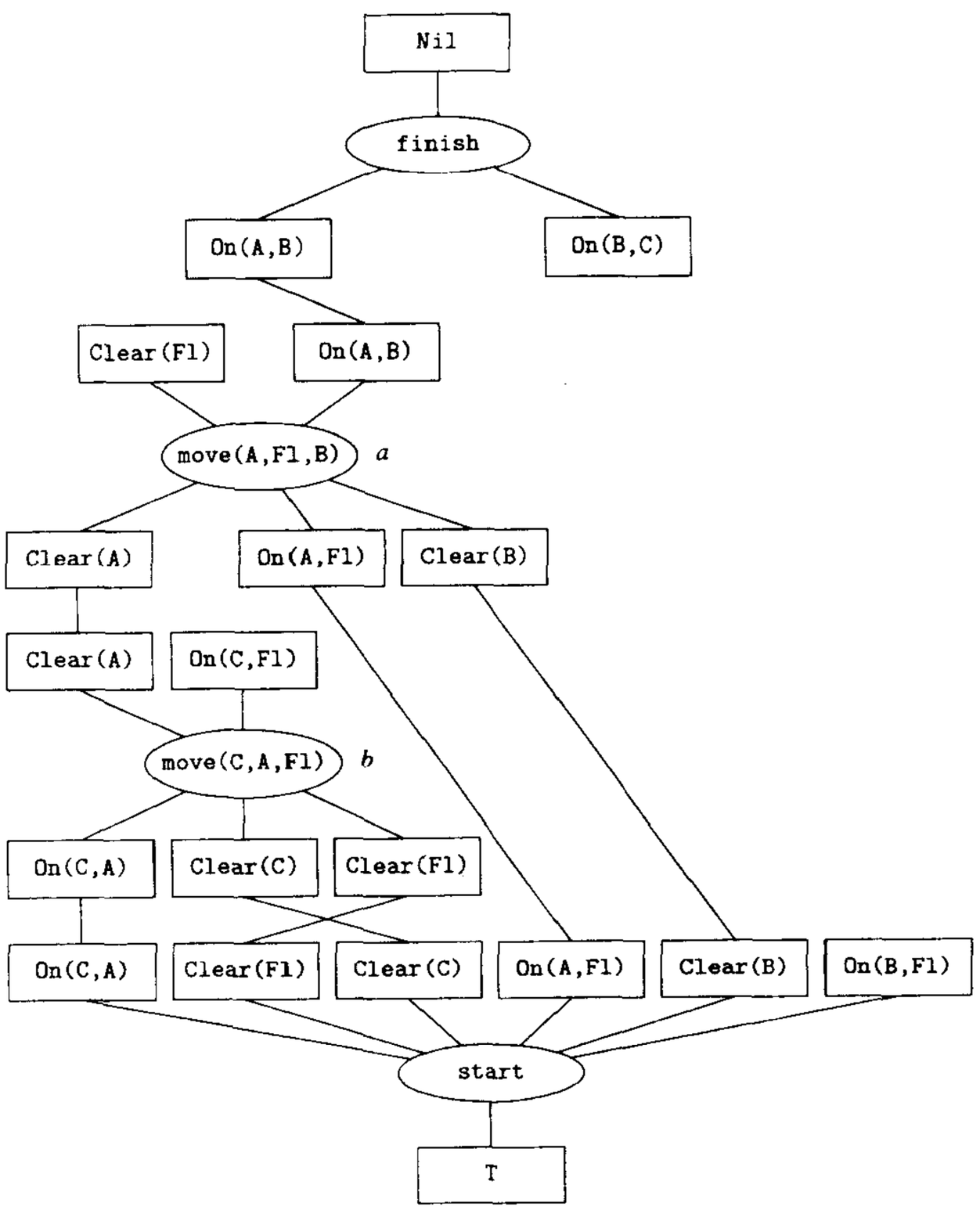


图22-13　一个接下来的计划结构

在这个阶段，我们看到两个move算子的前提条件在算子将被应用时得到满足。在表示不完全计划的图结构中有那两个move算子的一个隐含排序。在接下来关于计划过程的描述中，会引用这两个move算子，因此，在图22-13中用*a*和*b*标识它们。用符号*b*<*a* 、< 代表“在……之前”，明确表达了*b*必须出现在*a*的前面。注意两个长方形框On(C, Fl)和Clear(Fl)，它们是move算子的“产品”，但没有被随后的算子“消耗掉”。只要它们和后面的计划中必须为真的条件不矛盾，它们就不会妨碍计划。

假定下面我们考虑如何到达另一个主要目标合取项On(B, C)。这个条件能用规则实例move(B, z, C)实现，move(B, z, C)是指把*B*从某个位置移到*C*。因此我们能把这一步加到计划结构中，然后假定把z实例化为Fl，以便我们能连接到初始条件。产生的计划结构如图22-14所示。

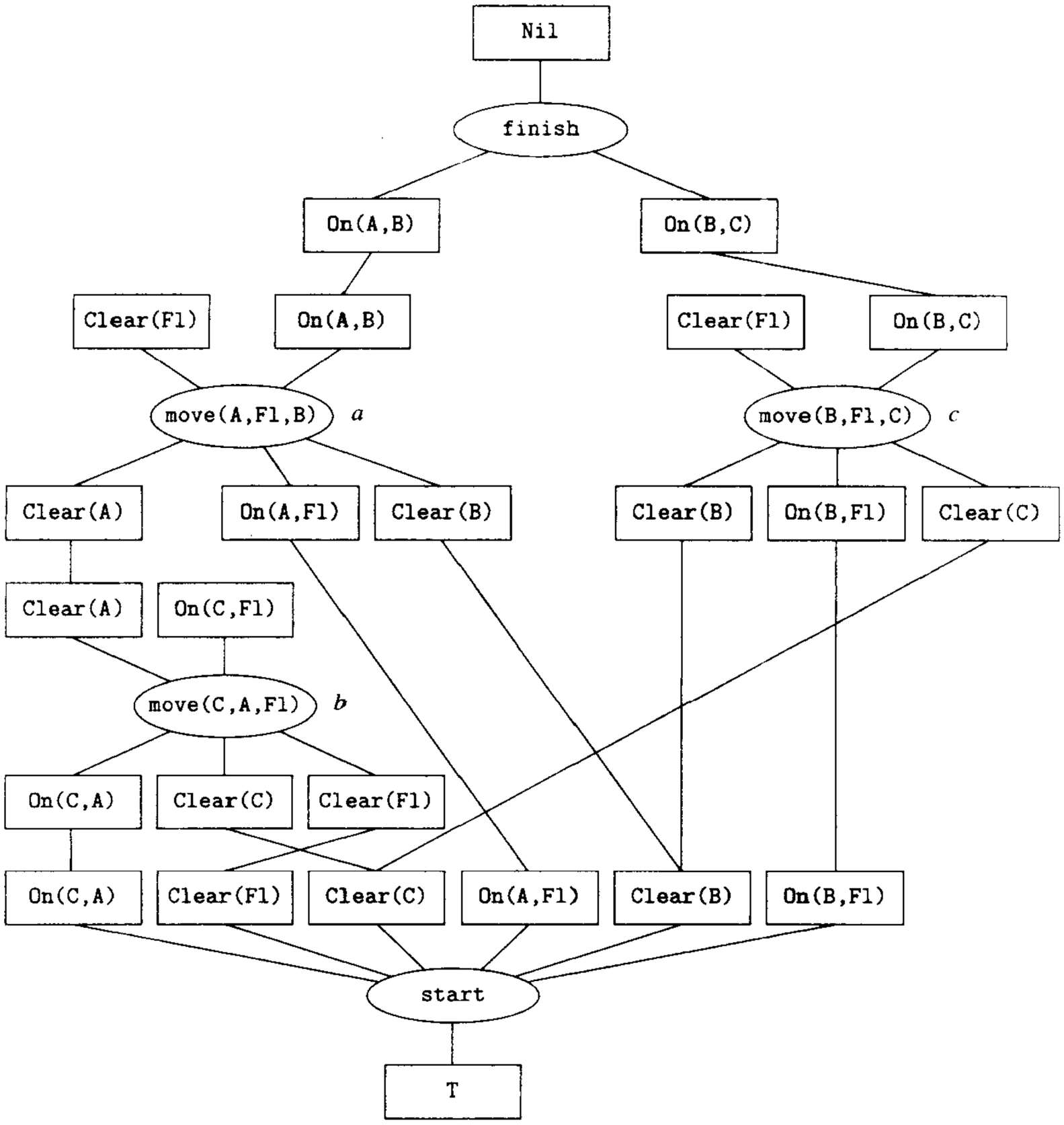


图22-14 计划结构的一个后期阶段

现在我们有一个涉及到算子*a*、*b*和*c*的计划结构，它仅仅是部分有序的。到目前为止，我们对*c*在计划步骤序列中的什么地方发生没有做出任何承诺。在部分有序计划中，一个算子如果在错误的时间被执行了，它可以取消另一个算子需要的前提条件，我们需要考虑由上述事实引起的问题。这种可能性在计划结构图中用威胁弧（*threat arc*）表示。考虑图22-15中的图结构。那些粗灰色弧线是威胁弧。威胁弧从算子（椭圆形的）节点画向那些前提条件（长方形的）

节点，它们是（a）在那个算子的删除列表上；（b）不是那个算子节点的后继。因此，在图22-15中，算子move（A，Fl，B）删除了move(B, Fl, C)的一个前提条件Clear(B)。在这种意义上，move(A, Fl, B)威胁move (B, Fl, C)，因为如果前者被首先执行，我们将不能执行后者。威胁弧必须通过给算子排序加上一些约束才能被消除。一个计划直到我们能发现一个一致的、能抛弃所有威胁弧的排序约束时才能完成。在例子中，这样的一组一致排序约束是（$c< a$、$b < c$）。当然，图结构本身暗示了约束（$b< a$）——它和其他的排序约束一致。因此，图22-15中的计划结构产生了总的排序：（$b< c< a$）。因为所有的威胁弧被消除了，这个计划也就完成了，所有被算子（包括finish算子）要求的条件在算子被应用时得到满足。最终的计划是{move(C, A, Fl), move (B, Fl, C), move (A, Fl, B)}。

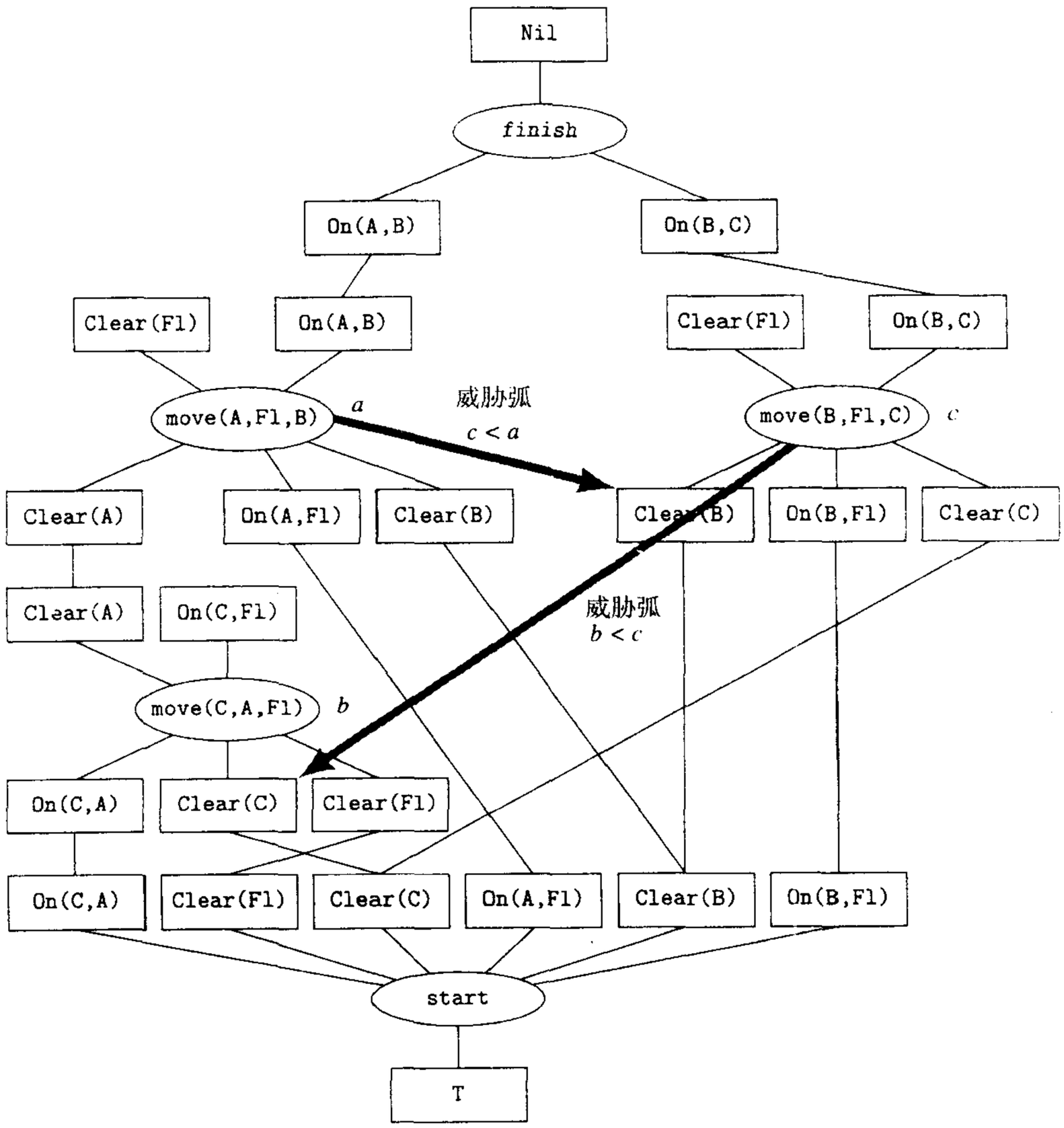


图22-15　插入威胁弧

22.3　层次规划

22.3.1　ABSTRIPS

第10章已提到过层次搜索过程。几个研究人员已经开发了基于STRIPS规则的一些层次计

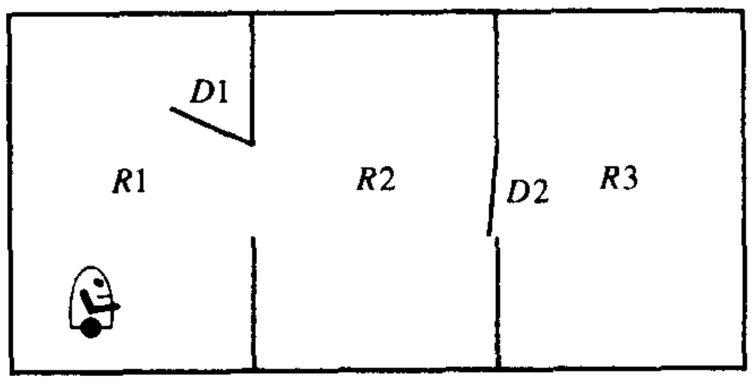
划者。一个是ABSTRIPS系统[Sacerdoti 1974]，它给一个STRIPS规则中每个前提条件的每个合取项分配关键度编号。到达一个合取项越容易（所有其他事情都相等），它的关键度编号越低。在ABSTRIPS中，规划在使用这些关键度编号的级别中进行。

1) 假定关键程度小于某个阈值的所有前提条件都已为真，生成一个基于该假设的计划。这里，除了最难的合取项，我们基本延迟了所有合取项的到达。

2) 把阈值减1，用第1步中产生的计划做向导，假定关键度比阈值低的前提条件都为真，生成一个计划。

3) 如此继续。

用一个例子解释ABSTRIPS的思想，这次用一个必须从一个房间移到另一个房间的机器人作例子。在图22-16中定义了初始状态、目标和要用的规则。规则goto（r1, d, r2）是机器人从房间*r*1，通过门*d*到达房间*r*2的动作模式的模型。规则open(d)打开门*d*，关键度编号用圆圈显示在这些规则的前提条件的相应文字上。特别地，假定打开一个门相对于通过一个门要容易一些，open(d)的关键度值是1。



初始状态：In(R1)∧Open(D1)∧Closed(D2)

目标：In(R3)

STRIPS 规则

goto(r1,d,r2)

②　①　②

PC: In(r1)∧Open(d)∧Connects(r1,d,r2)
D: In(r1)
A: In(r2)

open(d)

①

PC: Closed(d)
D: Closed(d)
A: Open(d)

图22-16 ABSTRIPS的一个规划问题

首先，我们构造一个抽象计划（用已经讨论的任何规划生成方法）——假定关键度为1的所有前提条件已被满足。在这个抽象级别到达In(R3)的计划是{goto (R1, D1, R2), goto (R2, D2, R3)}。接着，我们构造一个更详细的计划来首先到达抽象计划中第一个算子的前提条件，然后应用那个算子，再到达抽象计划中第二个算子的前提条件，这样一直进行下去。这些算子的前提条件能被看作为搜索空间中的孤岛——这些岛被第一级的抽象计划过程所发现。但现在，当我们到达这些前提条件时，将关键度阈值减1以要求我们也到达所有关键度为1的前提条件。结果是产生计划{goto (R1, D1, R2), open(D2), goto (R2, D2, R3)}。一般地讲，我们有多于两个的关键度值，规划过程将随着几个抽象级别下降。

ABSTRIPS使用的计划生成过程是所谓的长度优先（*length first*）。即在下降到一个更详细的级别以生成一个完全的计划之前，我们在每个抽象级别生成一个完整的计划。对长度优先计划生成有另一种选择。例如，我们能在顶级生成一个完整的计划标识搜索空间中的一个孤岛序列。这些孤岛将是各种第一级算子的前提条件。然后，我们能在下一级生成一个计划的第一部分，它仅仅完成到第一个孤岛，依此类推，直到我们在最低一级有了一个完全的计划，它到达了那个级别的第一个孤岛。假定在详细的最低级别，计划中的第一个算子和一个能在初始状态下执行的动作相对应。深度优先规划过程适合于使用感知/计划/动作循环的情况。在执行第一个动作且感知到结果状态后，我们通过重新规划重复那个过程——也许用前一个计划的一些更抽象级别作为指导。

层次规划是第9章描述的技术的一个实例，为了获得一个用在实际问题中的启发式函数值，首先要求解一个问题的简化版本。一个抽象规划过程的每一级能被看做下一级的一个简化版本（[Pearl 1984, pp.131~132]谈到了他的简化模型建议和ABSTRIPS之间的关系）。

22.3.2　层次规划和部分有序规划的组合

规划系统NOAH、SIPE和O-PLAN[Sacerdoti 1977, Wilkins 1988, Currie & Tate 1991]，组合了部分有序、计划空间规划和层次规划。除了已经提及的计划空间算子（实例化变量、加入STRIPS规则等等），这些系统还包括一些算子，可以把抽象计划清楚地表达为更低级的详细计划，这些清楚表达能被连接到前面考虑过的计划空间算子中。例如，用在积木问题中的move规则可能由更原始的规则pickup和putdown 序列构成，这些更低级的规则通常有更详细的前提条件，它们将要求插入其他更低级的规则以使计划在那个级别完成（见图22-17）。这个过程将继续，直到所有的算子（或至少第一个算子）与能被执行的原始动作相对应。

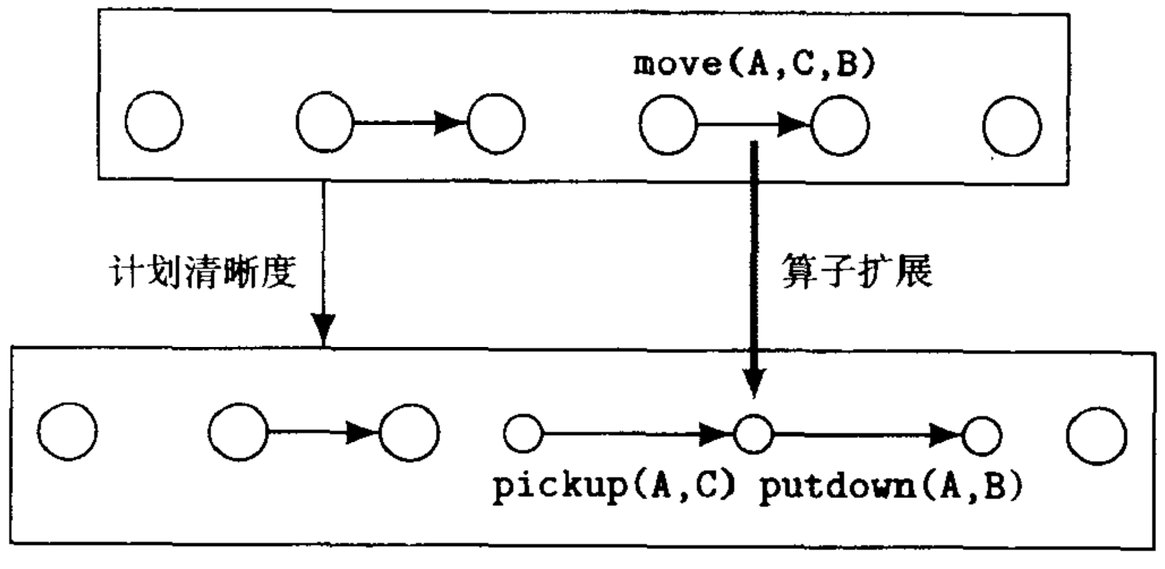


图22-17　清楚表达一个计划

22.4　学习计划

如同在第17章中用EBG来学习一个推理系统的新规则一样，也能用它学习包括一系列已经存在的STRIPS规则的新规则。然后用这些学习规则来建立更复杂的计划。当然，我们面临着前面提到过的相同应用问题。一个学到的规则不值得保存，除非它经常使用，节省规划的工作用起来不能是昂贵的。通过一个演示例子解释学习新规则的技术。

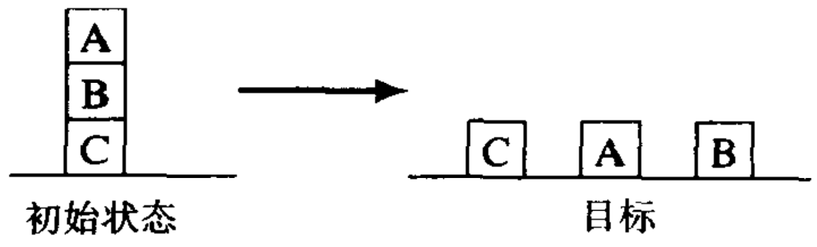


图22-18　分拆两个积木

考虑改变图22-18中积木配置的问题。开始时，*A*在*B*上，*B*在*C*上，我们想让*A*和*B*都在地板上。到目前为止讨论的任何规划方法都将产生计划{move(A, B, Fl), move(B, C, Fl)}。在一个新的STRIPS规则形式中，这个计划值得保存吗？如果我们的agent经常必须解决这个指定的问题，它就值得。如果这个计划能被一般化，以便它从一个积木堆上把顶部的两个积木（不管它们的名字）移到相同的目标地方，那么这个计划会有更多的应用。由于计划的构造不依赖积木的名字，将会呈现出一个象EBG一样的过程能产生一个带有变量符号的计划模式，变量符号能用对象常量A、B和C代替。但是由于这个两步计划依赖目标位置地板，故对象常量Fl不能被一般化。

计划一般化过程使用有关计划中算子的前提条件和结果的信息。表达这个信息的一个便利方法是用三角表形式[Fikes ,Hart & Nilsson 1972]。图22-19是一个拆积木问题的表。表的每一列用计划中的一个算子做标题——以这些算子在计划中出现的次序为序。就像在部分有序规划

中，很容易想像假想的start和finish算子。该表显示了每个算子的前提条件和每个算子的目的，每个算子下的单元包含一些由算子加入的文字（可能是重复的）。每个算子左边的单元包含那个算子的前提条件文字。我们用单元在表中的行位置*i*和列位置*j*对它们进行索引，表顶部的行值为1,表左部的列值为1。特别地，单元（*i*, *j*）(*i* ≥ *j*)包含了由算子*i*加入的文字，算子*i*被用做算子*j* +1的前提条件，它也是在*i*和*j*+1之间幸存的算子应用。因此，start算子下面有一些单元，它们包含了出现在初始状态描述中的文字以及被随后的算子需要的或满足目标条件中的合取项的文字。finish算子在它的左边有一些单元，它们包含了满足目标条件的文字。

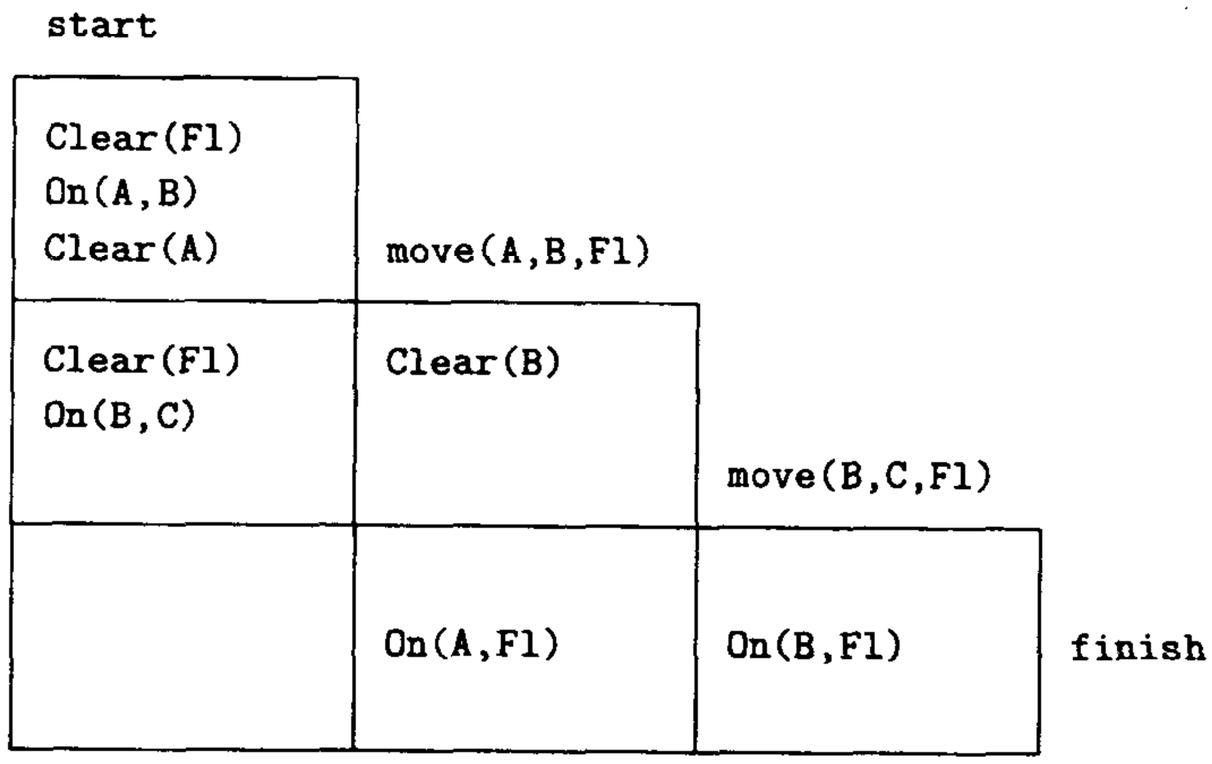


图22-19 拆积木的一个三角表

计划一般化过程的下一步是用变量符号代替所有的三角表中的对象常量。一个对象常量的每次出现被它自己的变量符号所代替。然后过程设法找到结果计划模式最一般的实例，以便所有规则的前提条件在应用一个规则实例时被满足。从第一个规则开始检查每个规则的前提条件，为了满足前提条件，还要做一些置换工作。图22-20给出了这个问题的结果三角表模式。

在三角表模式中，一般化的计划模式被称为一个MACROPS（为宏算子），它能被用做一个STRIPS规则来建立更长的计划。第1列的文字合取式是MACROPS的前提条件，第3行的文字是加入列表中的文字（为了计算删除列表，需要分析每个规则的删除列表）。[Fikes, Hart & Nilsson 1972]中讨论了一般化过程本身和使用三角表监视计划的执行。

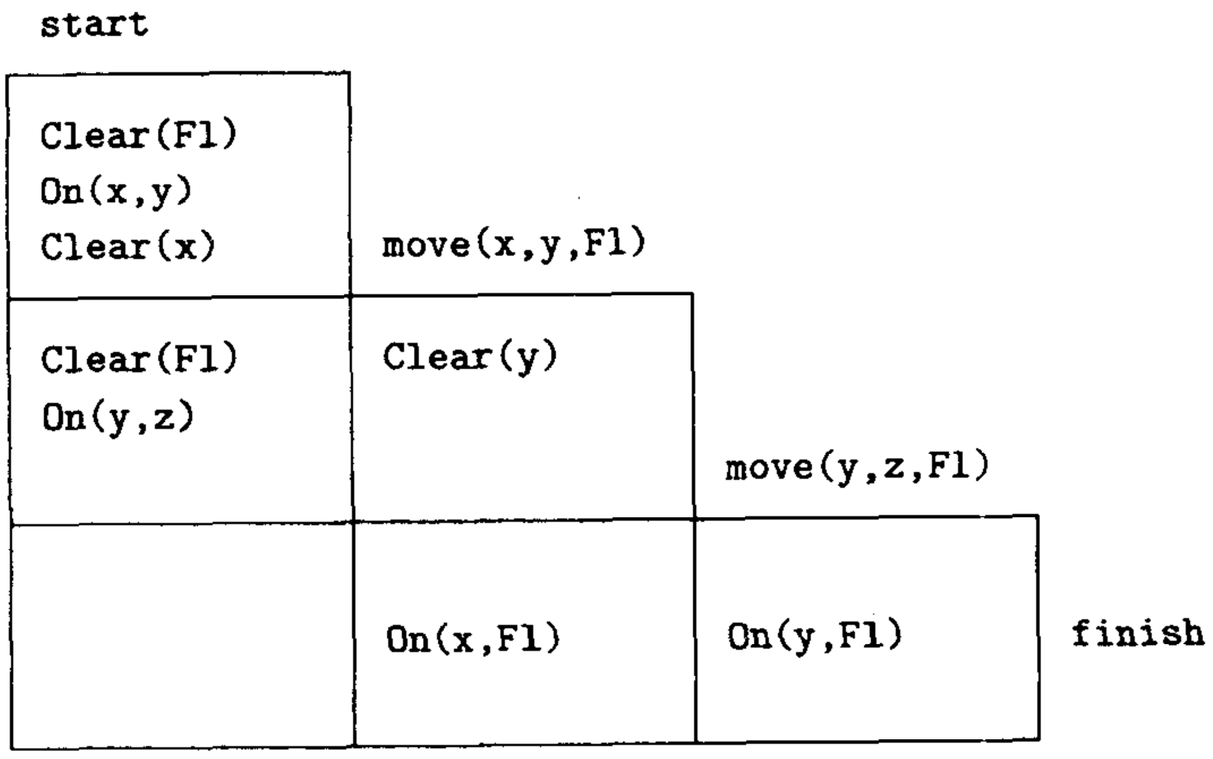


图22-20 拆积木的一个三角表模式

除了学习MACROPS，构造特定计划也能产生控制策略信息，这些信息可以减少将来的规划工作。例如，在上一个问题中，为试图获得 On(A, F1)∧On (B, F1)的合取式，首先获得合取项On(A, F1)是重要的（首先获得On(B, F1)可能要把*A*移到另一个积木的顶部，为了使*B*是空的，*A*必须被再次移到地面）。Minton的PRODIGY系统能学习这种使用EBG的控制信息[Minton 1988]（一个基于部分评估的可选方法，见[Etzioni 1993]）。

另一类在规划中有用的学习是学习规则自身的结果。一个agent可能有大量可用的动作，但它可能不知道这些动作的结果，或在什么条件下它们能被执行。有些在困难，但重要的学习动作模型问题中的结果已经被[shen 1994,Gil 1992,Wang 1995,Benson 1997]得到。[Benson 1997]使用一个T-R（远程反应）程序表示用学到的算子构造的计划。如第2章提到的，T-R表示法会给计划执行增加健壮性。

22.5　补充读物和讨论

[Lifschitz 1986]指出STRIPS的原始描述缺乏精确度和必要的约束以防止出现荒谬的公式。他讨论了STRIPS使用的语言约束，它们被用来避免这些问题的发生（在谈到公式时会提到这些约束）。[Pednault 1986，Pednault 1989]介绍了一个STRIPS版本，它既避免了Lifschitz提到的问题，又能用公式表示使用带有条件影响的规则的多agent计划 。[Bylander 1994]证明了带有命题STRIPS算子的规划在最坏情况下是不可处理的（见[Bylander 1993]的平均事件分析）。[Erol, Nau, & Subrahmanian 1992]给出一个STRIPS型规划系统复杂度的完整分析。

[Gupta & Nau 1992]证明了在世界状态和目标是由基原子合取式描述的积木世界问题中，找到一个最优（最短）计划，一般地讲是一个NP难题（然而，[Chapman 1989]指出一个简单的反应系统能快速地执行堆积木任务，如果它具有适当的感官谓词）。

[Waldinger 1975]介绍了通过STRIPS规则使用目标倒退，这个过程需要向后搜索。

[Blum & Furst 1995]把按照STRIPS规则形成的规划问题翻译成了能用路径发现方法求解的规划图结构。[Kautz & Selman 1996, Kautz, McAllester, & Selman 1996]提出了把规划图转化为能用WALKSAT求解的命题逻辑PSAT问题的方法。他们也提出了把规划问题直接写成PSAT问题的新技术。见[Ernst，Millstein，& Weld 1997]。这项工作是非常重要的，因为它可以导致利用有效的随机性和爬山搜索技术的可升级规划方法。

Sacerdoti的NOAH系统[Sacerdoti 1975, Sacerdoti 1977]和Tate的INTERPLAN[Tate 1977]是最早的部分有序规划者。Chapman的TWEAK系统[Chapman 1987]是第一个部分有序规划者的形式化，它介绍了几个重要的概念。他指出找到一个到达原子目标条件合取式的部分有序计划的问题是NP难题——在一定的合理公式表示下。

MCAllester和Rosenblitt的一个部分有序规划者的SNLP公式表示由[Soderland & Weld 1991]实现。UCPOP[Penberthy & Weld 1992]后来利用了这个实现。[Ephrati, Pollack, & Milshtein 1996]用一个A*型搜索去选择规划空间算子。[Minton, Bresina, & Drummond 1994]给出了一个部分有序和完全有序规划者的比较。对最小承诺和部分有序规划的一个综述见[Weld 1994]。

[Christensen 1990, Knoblock 1990]开发的方法可以自动学习算子层次。[Tenenberg 1991]可以研究了层次规划者需要的抽象观念。[Erol, Hendler, & Nau 1994]是另一个层次的部分有序规划者。

[Dean & Wellman 1991]是一本将AI规划和时态推理方法与现代控制理论很好集成的书。从其他方向逼近中间是由[Ramadge & Wonham 1989]写的关于离散事件系统的著作。有关时态推

理和它在规划中的作用的讨论，见[Allen, et al. 1990]。[Zweben & Fox 1994]介绍了应用于调度问题的各种规划和搜索方法。[Wilkins, et al. 1995]把规划方法扩展到包含不确定性的应用领域。

在[Allen, et al. 1990]中有很多有关规划的重要论文。[Minton 1993]中有关于学习和规划的论文。一本叫Practical Planning的书描述了SIPE系统和它的应用[Wilkins 1988]。

有关规划的论文定期出现在主要的AI期刊和会议论文集中。关于AI规划系统（AIPS）也有一个国际会议。规划和调度技术的一个最近应用的例子见[Tate 1996]。

习题

22.1 考虑为一个厨房清洁机器人设计一个计划的问题。

1) 写出一组可能要用的STRIPS型算子。当你描述这些算子时，做如下的考虑：

（a）打扫炉子或冰箱会弄脏地板。

（b）炉子必须在用铝箔盖住滴水的面板之前打扫。

（c）打扫冰箱会产生垃圾且会弄乱柜台。

（d）清洗柜台或地板会把水槽弄脏。

2) 写出一个橱房的初始状态描述，开始时厨房中有一个脏的炉子、冰箱、柜台和地板（水槽是干净的，垃圾已被清扫掉）。再写出一个目标状态描述。这时每件东西都被打扫过，没有任何废物。炉子的滴水面板已用铝箔盖住。

22.2 构思一个参数化的STRIPS规则move (x, y)，它给出了在8 数码问题中一个从位置x到y的移动（空白方格）的前提条件和结果。

22.3 解释递归STRIPS规则如何解决Sussman异常。

22.4 解释基于STRIPS规则利用倒退的向后搜索如何解决Sussman异常。

22.5 建立汉诺塔问题的一个STRIPS公式表示（参见习题5.3、7.4和9.2）。

即：

1) 指定描述一个移动结果的STRIPS规则，并指定目标条件。

2) 写出描述初始状态的公理，显示如何应用这个规则以产生一个后继状态（不要忘了包括在所有的状态中都为真的公式，可能需要它们来证明前提条件和目标条件）。

22.6 一个可消耗的火星机器人要计划到离它的大本营20公里远的一个火星山脉旅游路线。（它不必返回到大本营）机器人加一箱燃料刚好能旅行10公里。它也能带一箱燃料作为货物。大本营叫B，山脉叫M，一个临时台架站点（离山和大本营各10公里）是S。有三箱燃料，$P1$、$P2$和$P3$在B，机器人开始时没有加燃料。机器人有4个动作：

1) goto (x, y), 其中x是初始位置（B、M或S之一），y是目标位置（B、M和S之一）。为了执行这个动作，机器人在它的燃料箱中必须有一箱燃料，一箱燃料只能在B和S与S和M之间移动。

2) pickup (u, x)，u是在位置x的一箱燃料，这个动作的结果是机器人将u做为运输的货物。

3) putdown (u, x)，u是存放在位置x的一箱燃料。为了执行putdown，机器人必须正携带着u。

4) refuel (u, x)，u是位置x的一箱燃料。它将被加进机器人的燃料箱中。

（假定执行pickup、putdown和refuel动作不消耗任何燃料）。

用下面的谓词范式（带有明显的预期意思）：

Atrobot (x), 其中x是位置B、M或S之一。

At (r, x), 其中x是位置B、M或S之一，r是燃料箱之一即使机器人在x位置，机器人本身的一箱油也不认为是在x处。

Carrying (u)指出机器人正携带着燃料箱u（作为货物）。

Fueled 指出机器人的燃料箱中有一箱燃料（准备燃烧）。

Cango (x, y)给出了机器人在一箱燃料下的移动范围。它描述了机器人能从x去y（如果它被加了燃料）。

我们想建立起这个问题，以便一个STRIPS型的问题求解者能生成一个计划以使机器人能到达山脉。

1) 初始状态描述和目标条件是什么？

2) 解决这个问题需要的算子和它们的描述（前提条件、删除列表和加入列表）是什么？为了使问题简单一些，在公式中不必包括像Place (x)或Pellet (x)这些谓词。

3) 给出一个STRIPS系统（从初始状态向目标状态前进）可能产生的上述问题的一个求解计划。用下面的格式写出你的方案：

在初始状态S0中的文字

第一个算子

在随后的状态S1中的文字

第二个算子

依次类推，直至达到一个满足目标的状态。

22.7 发明一个“图标化的”数据结构模式（不是谓词演算公式）来表字上述练习题中的火星机器人和燃料箱的可能状态。开始状态的数据结构是什么？

为了表明你明白如何构造和使用这个问题的算子，显示把算子refuel(P1, B)和Pickup(P1, B)应用到开始节点产生的状态的数据结构。

22.8 假定我们的火星机器人有传感器，它使机器人能决定下面谓词的真假：

Atfuelpellet(S)

Atfuelpellet(B)

Fueled

Carrying

Atrobot(B)

Atrobot(S)

Atrobot(M)

它们有明显的预期含意。

机器人有如下的动作：

refuel　用一箱燃料给机器人加燃料（如果机器人的位置有一箱燃料）。

pickup　把一箱燃料做为货物加在机器人身上（如果机器人的位置有一箱燃料）。

putdown　在机器人的当前位置卸下一箱燃料。

goto(x)　使机器人到达位置x（如果机器人有燃料且x在10公里范围内）。

设计一个产生系统（第2章描述的类型），它将对所有传感输入的组合执行相应的

动作。

22.9 考虑使用一个部分有序规划系统（像本章描述的一样）来规划一个软件产品的开发和发布。假定在这个领域存在下面的4个STRIPS算子：

算 子	前提条件	加入列表	删除列表
optimize	HaveProgram	Optimized	BugFree
debug	HaveProgram	BugFree	
ship	HaveProgram BugFree Optimized HavePackaging	HappyCustomer	
designPackaging	HaveProgram	HavePackaging	

规划问题是从HaveProgram为真的一个状态到达HappyCustomer为真的一个状态。像我们知道的，部分有序规划方法将加入下面两个“虚构的”算子：

算 子	前提条件	加入列表	删除列表
start	T	HaveProgram	Nil
finish	HappyCustomer	Nil	HappyCustomer

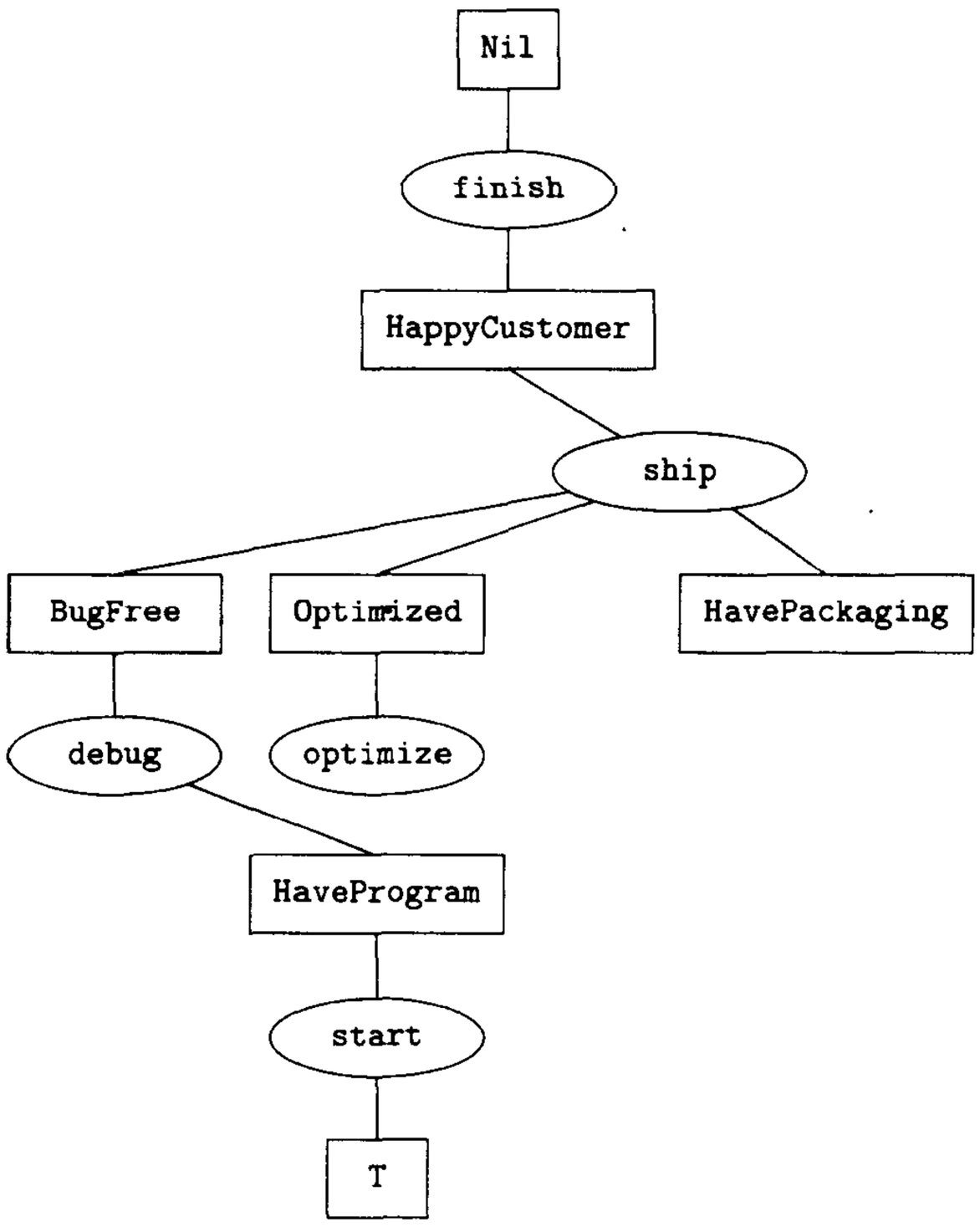


图22-21 习题22.9的部分计划

假定我们从图22-21中的部分规划开始。这个部分规划中当前没有包含任何排序约束，除了那些由部分规划的部分顺序暗示的约束。

1) 在这个部分规划中，如果有的话,现存的威胁弧是什么?

2) 列出这个部分规划中不被因果连接支持的任何算子的前提条件。

3) 出示一个部分有序规划过程可以产生的可能计划（即，用附加算子扩展显示的图，列出排除约束）。不必显示产生计划的所有步骤——显示完成的计划就行了。

22.10 描述如何用ABSTRIPS系统来产生一个“任何时间”的计划。简短评论一下为什么你希望随着计划所花时间的增多，计划的质量和可信度会改善?

第五部分　通信与集成

第23章　多 agent

23.1　交互agent

除了第12章讨论的游戏外，到目前为止我们关心的都是单个agent在一个和它的能力与目标多少相适应的环境中的反应、计划、推理和学习。假定能通过适当的agent反应来减轻或忽略任何其他agent或过程的作用。现在开始考虑如何在每个agent自己的计划中预测其他agent的作用，一个agent在它的目标服务中如何影响其他agent的动作。为了预测另一个agent将做什么，我们需要一些方法以使一个agent能对另一个agent建模；为了影响另一个agent将要做的事情，我们也需要一些方法以使一个agent能同另一个agent通信。本章和下一章将讨论这些主题。

采用一个“以agent为中心(*agent-centric*)”的观点，根据这种观点，我们用结构和目标来标识单个agent（称为我们的agent），这个agent在一个包含其他agent或过程的环境中活动。与大多数前面的假设不同，其他agent和过程对我们的agent的影响可以是非常大的，它们对实现我们的agent的目标可能是有帮助的、中立的或者甚至是敌意的。处理的方法是限定于一种情况：几个agent协调它们的活动以达到共同的目的——所谓的分布式AI（DAI）。

在进入本章和下一章的中心主题之前，要指出，最简单的一类agent交互既不需要其他agent的模型，也不需要显式通信。当这些影响发生或被感知时，我们的agent只要对其他agent的环境影响作出反应就行了。一种隐式的、非有意的agent之间的通信能通过这些环境影响发生。例如，另一个agent能够改变积木的环境配置，这样我们的agent就可以执行一个它前面不能执行的动作。其他agent的动作既实现了对其改变环境的目标，也给我们的agent发出了一个信号——即环境已准备好，你可以动作了；或者另一个agent可以发出某种我们的agent能够感知的听觉信号。对其他agent而言，这种“吱吱声”可能仅仅是对感知的环境状态的一种内在反应。但这种声音被我们的agent感知到后会引发一个动作，这个动作被一个外部观测者解释为对其他agent有意和计划的通信的反应。agent的设计者能够通过一个反应的信号和响应系统来协调agent之间的动作——就像一个分布式系统的设计者安排协调系统的部件之间的行为一样。

就像第2章到第6章的反应型机器拥有令人惊奇的复杂个人行为一样，这种隐式通信也能支持相当复杂的交互作用。一些动物的群体行为，如召集、群集和筑巢都是此类行为（参见[Mataric 1996, Theraulaz & Bonabeau 1995]）。

23.2　其他agent模型

23.2.1　模型种类

当我们的agent制定和执行自己的计划时，如果它要考虑其他agent和过程的活动，它将需

要一些能用来预测这些其他agent和过程如何动作的模型。由于agent和过程都是物理现象,我们可以用工程学和物理学的语言来描述它们的操作。像机器人一样的物理agent的行为能用各种详细级别的模型来描述，这些详细的范围从agent元件的电气机械图到控制它们的计算机程序。物理过程的行为常常可以用微分方程来描述。

一个著名的AI研究团体已经考虑了物理过程的建模问题，这个工作常被称为质朴物理或定性物理（*naive physics or qualitative physics*）。[Weld & De Kleer 1990]。定性物理寻求用更近似的、在动态定量处理模型上操作的AI类型的推理方法来代替用来解决基于各种物理现象的微分方程的确切计算。然而在此不讨论这种类型的建模，而是把重心放在其他agent的高级模型上（事实上，能够对很多物理过程建模，好象它们是反应型agent——也许被T-R程序支配着[⊖]）。

尽管agent本身也是物理过程，能像任何其他的工程设备一样建模，我们将发现用专门的技术对agent建模是方便的。我们在这本书中研究的逐渐复杂的agent领域提出了对其他agent建模各种方法。例如，我们的agent可以假定另一个agent是一个T-R程序，它能评估条件和选择动作。不管其他agent的程序是什么形式，我们的agent模型都不必像真正的事物那样复杂，只要模型能在各种环境中对其他agent的行为给出合理有用的预测就行了。或者我们的agent可以假定其他agent用一组STRIPS规则产生计划，并执行这些计划以达到其目标。在任何情况下，假定其他agent保持和使用它自己的一个内部环境模型以便选择动作常常是有用的。一个agent模型和用那个模型选择动作的装置一起被称为认知结构（*cognitive structure*）。通常，认知结构也包括一个agent的目标和意图（当适合于做这些由计划给定的动作时，把意图称为agent已决定执行的动作）。我们的agent需要对其他agent的假定认知结构在充分详细了解和精确条件下进行建模，以便用这些描述来推理以预测其他agent的行为。

这里集中讨论的agent认知结构是它的环境模型和其他agent的认知结构。我们已经研究了由agent使用的两种模型:图标的和基于特征的。回忆一下，一个环境图标模型企图模拟环境的相关部分，一个基于特征的模型试图描述环境——也许用谓词演算公式。当考虑我们的agent如何对另一个agent建模时，我们再次有这个选择。事实上，我们现在有4个选择。我们的agent能用其他认知结构的图标或基于特征的模型，另一个agent本身也可被假定正在使用图标的或基于特征的模型。表23-1给出了四种可能性以及建模策略。在下面的几小节中将讨论这些建模策略中的三个（结果是当我们的agent使用一个基于特征的模型策略时，不需关心另一个agent被假定是使用基于特征的还是图标的模型。为了现在的目的，当我们的agent用一个基于特征的策略时，假定另一个agent也使用这种策略）。

表23-1　建模策略

我们的agent	other agent	Modeling Strategy
图标的	图标的	模拟
图标的	基于特征的	模拟的数据库
基于特征的	图标的	描述性的
基于特征的	基于特征的	有意的

23.2.2　模拟策略

在第12章，我们已经见过一个agent使用图标模型、另一个agent也假定使用图标模型的简

⊖ 一些原始人类把这些现象看作是极端的事情——认为是一些故意的agent 引起了这些物理现象，像闪电、飓风、日蚀和地震。

单例子。计算一个对手的可能博弈位置的过程要求我们的agent能够推算出对手动作的可能结果。图标模型常常是有用的，但也受到已经阐述过的重大限制，即它们在表达肯目性和不确定性时有困难，这个困难也困扰着我们下面将要介绍的模拟数据库方法。

23.2.3 模拟数据库

我们的agent对另一个agent的公式（它们描述了另一个agent的世界）建模的方法是建立一个假定的公式数据库，假定这些公式和我们的agent 实际上要置入另一个agent的世界模型中的公式相同。也就是说，我们的agent只要设法复制它认为的另一个agent的模型。例如，如果我们的agent认为另一个agent在它的世界模型中有公式On(A, B)，我们的agent可能有一个“其他agent”数据库，里面包含着公式On(A, B)。在其他agent模型数据库有那个公式决不排除我们的agent在它的世界模型中有公式On(A, C)。当然，我们的agent也可能不知道关于另--个agent描述的世界词典中的任何东西。例如，另一个agent可能用字符串$#4(QW,V)表示我们的agent用On(A, B)表示的命题。只要agent之间不必通过传送这些没有翻译的公式来通信，我们就不必关心这些词汇差别。这种特殊的方法对一些相对简单类型的关于另一个agent的推理也许是可行的，但是它和图标模型一样有同样的缺陷。一个最明显的例子就是对我们的agent来说，表达另一个agent 在它的模型中有On（A，B）或On（A，C）这个事实比较困难。注意把On（A，B）∨On（A，C）放在模拟数据库中将表示一些不同的事情。--种情况是我们的agent是不确定的；另一种情况是另一个agent是不确定的。我们的agent要表达它的不确定性，必须有两个模型数据库——每种情况一个。而且，如果还有另外的这种不确定类型，不同模型数据库的数量将随着处理大量的不确定性的组合而呈指数增长。

为了给这个方法一个相关难度，假定我们的agent不知道另一个agent是否在它的世界模型中有On（A，B）。我们不能简单地从模型库中省略公式On（A，B）。因为这样做暗示了另一个agent在它的世界模型中没有这个公式（这也可能）。这个问题又是我们的agent缺乏知识和另一个agent缺乏知识的差别之一。

23.2.4 有意思维方式

为了能够区分这些差别，我们的agent需要某种语言（也许是一个谓词演算的变形），它能够描述另一个agent关于世界的知识和信念而不是模仿它们。当我们的agent描述另一个agent的知识和信念时，它正在进行着被Dennett称为朝着另一个agent的有意思维方式[Dennett 1971]。人类常常对自己使用的机器采用一种有意思维方式。例如，我们可能说：“这台计算机知道今年是润年”（见[McCarthy 1979a]）。

已经有三种可能性被建议用来构造另一个agent的有意思维方式模型。首先，我们能具体化另一个agent的信念（就像我们在环境计算中具体化动作一样）。[McCarthy 1979b]）提出了这样一个方法。即，我们可以给这些信念命名，用一个相关常量Bel来指称一个agent和它的（一个）信念之间的关系。但是我们需要把这些名字（另一个agent的信念）和我们的agent的相似信念（由逻辑公式指称）连接起来，一个适当的连接机制已证明是具有一定难度的。

第二，我们的agent能够假定另一个agent实际上通过谓词演算公式在它描述世界的事实数据库中表示它关于世界的信念。那么我们的agent可以利用一个关系常量Bel，这次它指称另--

个agent与在其他agent的数据库中的一个字符串之间的关系。例如，我们的agent可能用公式Bel（Sam，'On(A, B)'）来表示命题：一个叫做Sam 的agent在它的世界知识数据库中有字符串 'On(A, B)'（指称公式On(A, B)）。这种方法涉及到agent和公式之间的关系，因此被称为元语言（*metalinguistic*）方法[Perlis 1985; Perlis 1988; Konolige 1982; Kowalski & Kim 1991; 和 Genesereth Nilsson 1987, 第10章]。

第三，我们能用一个涉及到模式算子的谓词演算的细节。这种算子能被用来构造指称关于谁知道或相信更简单的、指称真正的知识或信念的命题。在下一节将对之进行详细介绍。

23.3　知识模式逻辑

23.3.1　模式算子

我们对在命题和一阶逻辑中使用连接词 ∧ 和 ∨ 都很熟悉。可以把这些连接词作为算子来从简单元件构造更复杂的公式。当这些算子构造一个新公式时，新公式的真值依赖于操作对象的真值，也依赖算子的特殊性。这里，我们想构造一个公式，它的预期含意是一个确定的agent知道一个确定的命题。元件由一个表示那个agent的术语和指称那个agent知道的一个命题的公式构成。为了实现这个构造，引入模式算子(*modal operator*)**K**。例如，为了说Sam(一个agent的名字)知道积木*A*在*B*的上面，我们写下：

K(Sam, On(A, B))

由**K**和术语Sam以及公式On(A, B)组合形成了一个新公式，它的期望含意是："Sam知道积木*A*在*B*上"（在本章中，偶尔会用缩写$\mathbf{X}_{\alpha}(\phi)$，代替$\mathbf{X}(\alpha, \phi)$，$\alpha$指称一个agent，$\phi$是一个公式）。

注意，我已经开始用"知道"代替"相信"。哲学家长期以来一直在讨论知识和信念的差别。一个差别是尽管一个agent能相信一个错误的命题，但它不知道它是错误的东西。定义和使用一个知识逻辑比定义和使用一个适当的信念逻辑简单一些。因此，在这一部分，把讨论限制在知识——把它作为一个有用的理想化事物，它能用各种方式调整以使它适于处理信念。

使用算子**K**的语言称为一种一阶模式语言。这个语言中的公式句法如下：

1) 所有的普通一阶谓词演算合式公式也是模式语言的合式公式。

2) 如果ϕ是模式语言的一个封闭合式公式（具有没有未量化的自由变量集），α是一个基本项，那么$\mathbf{X}(\alpha, \phi)$就是模式语言的一个合式公式。

3) 同样地，如果ϕ和ψ是合式公式，那么由ϕ和ψ通过常用的命题连接词构造的任何表达式也是合式公式。

作为例子，下面是一些合式公式：

- **X**[Agent1, **X**(Agent2, On (A, B))]，它意指Agent1知道Agent2知道*A*在*B*上。
- **X**(Agent1, On(A, B) ∨ K(Agent1, On(A, C))，它意指Agent1知道*A*在*B*上，或者Agent1知道*A*在*C*上。
- **X**[Agent1, On(A, B) ∨ On(A, C)]，它意指Agent1知道*A*在*B*上，或者*A*在*C*上。
- **X**(Agent1, On(A, B) ∨ **X**(Agent1, ¬ On(A, B))，它意指Agent1知道*A*是否在*B*上。
- ¬ **X**(Agent1, On(A, B)，它意指Agent1不知道*A*在*B*上。

注意，根据我们的句法，表达式（∃ x）**X**（Agent1, On(x, B)）不是一个合法的合式公式。表达式On(X, B)在模式算子内不是封闭的，它包含一个在模式算子外量化的自由变量。

一些模式逻辑允许这种类型的“量化”，但它带来的困难超出了我们这里的研究范围。

23.3.2 知识公理

算子∧和∨具有组合语义——意思是说一个由使用这些算子的元件公式构造的公式的真值依赖算子和元件公式的真值。然而**K**的语义并不是组合的。例如，**X**(Agent1, On(A, B))的真值不能由**X**的属性、Agent1的指称和On(A, B)的真值来决定。当然，如果On(A, B)有F值，那么对任何α值，$\mathbf{X}_{\alpha}$(On(A, B))的所有值也是F（因为一个agent不会知道错误的东西）。但是即使On(A, B)有T值，不是$\mathbf{X}_{\alpha}$（On(A, B)）的任何实例都必然有T值（因为由α指称的agent可能不知道由On(A, B)指称的命题）。同样地，即使两个公式ϕ和ψ是等价的，这种情况——如果$\mathbf{X}_{\alpha}(\phi)$有T值。那么也并非必须对任何$\alpha$值$\mathbf{K}_{\alpha}(\psi)$也有T值（由$\alpha$指称的agent可能不知道那些等价的公式[⊖]）。模式公式的语义必须考虑到这些结果。

这种模式语言语句的一个语义涉及到可能的领域（作为一个可选方法，参见[Konolige 1986]）。虽然不讨论这里考虑的可能领域，但概括地讲当一个命题在一个agent的所有可能领域内是真的时，那个agent知道这个命题。反过来，如果（对那个agent知道的所有领域）在某些领域那个命题是真的而在另一些领域它是假的，那么一个agent不知道该命题。即使未研究知识逻辑语义，仍能提出一些方案允许我们的agent推理另一个agent的信念。这些方案将使用公理，它声明了带有一个新的推理规则和假言推理的知识的特殊属性。

一些常用的公理模式是：

$$[\mathbf{K}_{\alpha}(\phi) \wedge \mathbf{K}_{\alpha}(\phi \supset \psi)] \supset \mathbf{K}_{\alpha}(\psi) \tag{23-1}$$

它的意图是如果一个agent知道ϕ，并且也知道$\phi \supset \psi$，那么该agent也知道ψ（因为我们假定它能执行假言推理）。这个公理模式有时用等价形式写为：

$$\mathbf{K}_{\alpha}(\phi \supset \psi) \supset [\mathbf{K}_{\alpha}(\phi) \supset \mathbf{K}_{\alpha}(\psi)] \tag{23-2}$$

这个公理被称为分布公理，因为它同意把K算子分布蕴涵。

另一个公理模式是知识公理，它指出一个agent不可能知道一些错误的事情。

$$\mathbf{K}_{\alpha}(\phi) \supset \phi \tag{23-3}$$

这个知识公理暗示了一个agent不知道矛盾：$\neg\mathbf{X}(\alpha, \mathrm{F})$。

作为知识的第三个特性，假定如果一个agent知道某件事，那么它知道它知道这件事是合理的。这个特性用肯定自省公理（positive-introspection axiom）表示：

$$\mathbf{K}_{\alpha}(\phi) \supset \mathbf{K}_{\alpha}(\mathbf{K}_{\alpha}(\phi)) \tag{23-4}$$

在一些知识公理中，如果一个agent不知道某事，那么它知道它不知道它——否定内省公理：

$$\neg\mathbf{K}_{\alpha}(\phi) \supset \mathbf{K}_{\alpha}(\neg\mathbf{K}_{\alpha}(\phi)) \tag{23-5}$$

下一个特性是任一个agent知道所有的这些公理（以及所有有效的公式）。我们能通过给逻辑加上其他的推理规则表达这个属性。这个规则叫认识必要性（*epistemic necessitation*），它允许我们（即我们的agent）推断$\mathbf{X}_{\alpha}(\phi)$，如果$\phi$是一个有效的公式。我们能把这个推理规则写为：

⊖ **X** 算子因此被称为是引用不透明的。

从 $\vdash \phi$ 推断 $\mathbf{X}_{\alpha}(\phi)$　　(23-6)

由于假言推理是命题逻辑中惟一需要的推理规则，公理23-2和规则23-6使我们可以断定一个agent知道所有它的知识的命题结果，即从逻辑上讲它是无所不知的。我们能把这个事实表达为如下的一个推理规则：

从 $\phi \vdash \psi$ 和 $\mathbf{X}_{\alpha}(\phi)$ 推断 $\mathbf{X}_{\alpha}(\psi)$　　(23-7)

这个规则的一个等价形式是

从 $\phi \vdash (\phi \supset \psi)$ 推断 $\mathbf{X}_{\alpha}(\phi) \supset \mathbf{X}_{\alpha}(\psi)$　　(23-8)

逻辑全知对有限的agent似乎是不实际的，这些agent毕竟不能导出它们清楚知道的东西的所有结果。如果一个agent不能导出一个命题（即使它跟随它知道的其他命题），能说它知道那个命题吗？一个知道数论公理的人能知道所有的理论吗？这依赖于我们如何看待“知道”。例如，我们可能有一个柏拉图式的知识观点，在这种观点中，按照定义，一个agent知道它的知识的所有结果——尽管明确地相信它们可能不必要。虽然逻辑全知似乎太强了，但由于智能agent将要做一些推理，因此作为一种近似它是有用的。

从逻辑全知（规则23-7），我们能导出：

$$\mathbf{K}(\alpha, (\phi \wedge \psi)) \equiv \mathbf{K}(\alpha, \phi) \wedge \mathbf{K}(\alpha, \psi) \qquad (23\text{-}9)$$

即K算子可以分布合取。然而它不能分布析取，因为（例如，像我已经讨论的）$\mathbf{X}(\alpha, (\phi \vee \psi)) \equiv \mathbf{X}(\alpha, \phi) \vee \mathbf{X}(\alpha, \psi)$ 是不成立的。

已经指出即使我们的agent不知道ϕ是否是真的，我们的agent也能表达另一个agent知道ϕ是否是真的。它用了表达式$\mathbf{K}(\alpha, \phi) \vee \mathbf{K}(\alpha, \neg \phi)$。

23.3.3　关于其他agent知识的推理

我们的agent能对其他agent的知识语句进行验证，这些agent只使用了知识公理、知识必要性和它自己的推理能力（假言推理和归结）。用所谓的聪明人（Wise-Man）问题示例这个过程。假定有三个聪明人，国王说他们中至少有一个人的前额上有一个白点；事实上，所有的三个人前额上都有白点。假定每个聪明人都能看到其他两个人的前额但不能看到自己的。因此每个人知道其他两人是否有白点。这个问题有各种版本，但是假如我们的agent被告知：第一个聪明人说“我不知道我是否有白点”，然后第二个聪明人说“我也不知道我是否有白点”。可以在我们的知识逻辑中用公式表达这个问题，以表明我们的agent能证明第三个聪明人由此知道他有一个白点。

用一个更简单的两个聪明人(two-wise-man)版本来说明这个问题的推理过程。我们把聪明人称为*A*和*B*，所需的信息包含在下面的假设中，它们来自于下述语句：

1) *A*和*B*都知道每个人能看到另一个的前额。因此，例如，

1a. 如果*A*没有白点，*B*将知道*A*没有白点。

1b. *A*知道(1a)。

2) *A*和*B*都知道他们中至少有一人有一白点，而且他们都知道对方知道这个事实。

2a. *A*知道*B*知道或者*A*或者*B*有一个白点。

3) B说他不知道他是否有白点，*A*因此知道*B*不知道。

语句（1b）、(2a)和（3）分别是下面$\mathbf{X}(\mathrm{A}, \mathrm{white}(\mathrm{A}))$证据的前三行：

1) $\mathbf{X}_A[\neg \mathrm{White}(A) \supset \mathbf{X}_B(\neg \mathrm{White}(A))]$ (给定)

2) $\mathbf{X}_A[\mathbf{X}_B(\neg \mathrm{White}(A) \supset \mathrm{White}(B))]$ (给定)

3) $\mathbf{X}_A(\neg \mathbf{X}_B(\mathrm{White}(B)))$ (给定)

4) $\neg \mathrm{White}(A) \supset \mathbf{X}_B(\neg \mathrm{White}(A))$ (1和公理23-3)

5) $\mathbf{X}_B[\neg \mathrm{White}(A) \supset \mathrm{White}(B)]$ (2和公理23-3)

6) $\mathbf{X}_B(\neg \mathrm{White}(A)) \supset \mathbf{X}_B(\mathrm{White}(B))$ (5和公理23-2)

7) $\neg \mathrm{White}(A) \supset \mathbf{X}_B(\mathrm{White}(B))$ (对子句4和6的归结)

8) $\neg \mathbf{X}_B(\mathrm{White}(B)) \supset \mathrm{White}(A)$ (7的对换句)

9) $\mathbf{K}_A[\neg \mathbf{X}_B(\mathrm{White}(B)) \supset \mathrm{White}(A)]$ (1-5, 8, 规则23-7)

10) $\mathbf{X}_A(\neg \mathbf{X}_B(\mathrm{White}(B))) \supset \mathbf{X}_A(\mathrm{White}(A))$ (公理23-2)

11) $\mathbf{X}_A(\mathrm{White}(A))$ (使用3和10的假言推理)

为了导出证据中的第9行，我们用规则23-7调整声明：当A相信前提（行1 和行2 ）时，它也相信来自前提（行4和行5）的一个证据（行8）结果。

在three-wise-man版本中，有另一级嵌套推理，但基本的策略是一样的。事实上，假定开始的（k-1）个聪明人声称他不知道他是否有白点，我们能解决k-wise-man问题（对任意的k）。

把公理23-2到23-5、普通的命题逻辑公理、普通的推理规则和规则23-6组合起来可构成我们的知识模式逻辑（*modal logic of knowledge*）。逻辑学家已经给各种模式系统（每一个都有一些不同的公理模式）命名了特殊的名字。对一个固定的agent A，公理23-2到23-5是一个称为S5的模式逻辑系统的公理。如果我们去掉公理23-5，有系统$S4$。如果我们去掉公理23-4和23-5，有系统T。如果去掉公理23-3, 23-4和23-5，有系统**K**。

23.3.4 预测其他agent的动作

为了预测另一个agent $A1$可能做什么，我们的agent必须有一个$A1$的模型。如果$A1$不是太复杂，那么对我们的agent来讲一种可能性就是假定$A1$的动作由一个T-R程序控制着（也许我们的agent用一个学习过程来导出一个T-R程序，它很好地考虑了可以接受的$A1$的过去动作）。假如在那个程序中的条件是$\gamma_i, i=1,...,k$，现在来预测$A1$的未来动作，我们的agent需要能够对$A1$如何计算这些条件进行推理。在下一章我们将看到对我们的agent来讲，对$A1$采取一个有意的思维方式和试图建立$\mathbf{X}_{A1}(\gamma_i), i=1,\cdots,k$是合适的。

23.4 补充读物和讨论

[Shoham 1993]已把对其他agent的目标和计划进行推理的思想合并进他的分布式系统高级方法中，系统使用了“面向agent的编程”。

Minsky的《Society of Mind》[Minsky 1986]认为甚至那些我们认为是单元（像我们自己一样）的agent实际上也是由成千上万的简单agent按层次构成（作为对这本书的评论和Minsky作出的答复，请见《Artificial Intelligence》, vol. 59,1991）。

知识模式逻辑首先由[Hintikka 1992]提出。对模式逻辑的可能世界语义由[Kripke 1963]首创。[Moore 1985b]在一阶逻辑里对可能的世界语义进行了公理化，并用这种形式研究了知识和动作之间的交互。知识逻辑叫做认识逻辑。也有信念逻辑（*doxastic logic*）、义务逻辑（*deontic logic*）和其他形式。[Levesque 1984b, Fagin & Halpern 1985, Konolige 1986]研究了处

理信念（与知识相对）的专用方法。[Cohen & Levesque 1990]用一个模式逻辑研究了意图和承诺之间的关系。

[Moore 1985a, Moore 1993]指出关于自己知识推理（自认识推理，*autoepistemic reasoning*）的元级活动是一种非单调推理方式。

很多重要的DAI论文被收集在[Bond & Gasser 1988]中，《多agent系统的国际会议论文集》收录了当前的研究成果。有关知识推理的论文收录在《知识推理理论方面（TARK）的研究论文集》中。

习题

23.1 证明：用知识公理证明一个人不可能既知道P又知道¬ P。

23.2 公式¬ $\mathbf{X}_\alpha$ ¬（$\mathbf{X}_\alpha(\phi)$）⊃ ϕ 被称为Brouwer公理。证明它（用归结反驳法和知识公理）。

23.3 假定给出下面的句子：

1) $\mathbf{X}_j(\mathbf{K}_s(P) \vee \mathbf{K}_s(Q))$

(John知道Sam知道P或者Sam知道Q。)

2) $\mathbf{X}_j(\mathbf{K}_s(P \supset Q))$

(John知道Sam知道P ⊃ Q)

3) $\mathbf{X}_j(\mathbf{X}_s(\neg R)$。)

(John知道Sam知道¬ R。)

证明$\mathbf{X}_j$（$\mathbf{X}_s(Q)$）

23.4 23.3.2节给出了关于知识的各种公理。检查每一个公理，讨论一下它们是否也适合于信念，以及在什么条件下适合。例如，下面的公理模式合适吗？

$[\mathbf{B}_\alpha(\phi) \wedge \mathbf{B}_\alpha(\phi \supset \psi)] \supset \mathbf{B}_\alpha(\psi)$

提出任何其他你认为适合于信念（和知识相对）的公理。信念能按照知识定义吗？（反过来呢）？

23.5 在23.3.1节指出表达式（∃ x）**X**（Agent1, On(X, B)）不是一个合法的合式公式。那么，我们怎么说Agent1知道一个确定的对象在*B*上呢（我们不知道那个对象是什么）？（注意表达式**X**（Agent1,（∃ x）On (X, B)）的预期意思是Agent1知道某个对象在*B*上但Agent1可能不知道那个对象是什么）。描述一下允许像（∃ x）**X**（Agent1, On(X, B)）一样的表达式是一个合式公式的一些困难之处。

第24章　agent 之间的通信

24.1　交谈

我们的agent可以通过两类方法来有意地影响另一个agent的动作。假如我们的agent知道另一个agent如何对其环境的变化做出反应，那么，我们的agent就可以通过改变环境来达到所需的效果。

或者我们的agent可以试着改变另一个agent的目标、知识（或信念）或者动作选择机制，它可以通过直接“写”另一个agent的认知结构中的这些元素来达到目的。例如，这个方法能被一个人（我们的“agent”）使用以控制一个拥有这种接口通道的agent。更有趣的是，我们的agent能用这种方法和另一个agent通信，以使另一个agent对它的信念和目标（最终使另一个agent采取所希望的动作）做出改变。通信媒体将依靠另一个agent的感觉和知觉装置。例如，可能包括“写”（基于另一个agent的视觉能力），“听”（基于另一个agent的听觉能力）或者“广播“（基于另一个agent的电磁接收能力）设备。当一个agent采取这样一个动作以企图影响另一个agent的认知结构时，我们说agent已经参与了一个通信动作(*communicative act*)。

人类之间的交谈常涉及到说话采用的语言（用声波作为介质）。因此，语言学家把各种类型的交流动作称为交谈(*speech act*)。他们把讲话一方称为讲话方(*speaker*)，把受话方称为听众(*hearer*)。根据哲学家John Searle [Searle 1969]的理论（他是第一个研究这种基础理论的人），交谈可分为几类：表示型（*representatives*，它陈述了一个主张或建议）、指示型（*directives*，要求或命令）、委托型（*commissives*，许诺或威胁）、表达型（*expressives*，感谢或道歉）和声明型（*declarations*，事实上它改变了世界的状态，如：“我现在宣布你们成为夫妻”）。

交谈可能有各种不同的物理表现形式，它们可以是一个动作序列（如在手势语言中）、一串符号（在文本中）、一个声音扰动（尖叫、讲话）或闪光。无论何种表现形式，交谈的表现都被称为讲话（*utterance*）。如Searle 所说的，讲话不但要表达讲话的内容还要表达讲话的类型。例如，如果所写的英语文字是媒体，那么把积木A放在B上既表达了自然的要求也说明了它的要求内容，叫做On (A, B)（这种讲话方式被讲话方用来改变听众的目标结构）。

假定交谈被认为会对听众的知识产生影响。如果我们的agent *A*1通过一个表示型交谈来通知一个听众*A*2，一个由ϕ指称的建议为真，那么*A*1认为这个交谈的结果是*A*2知道*A*1打算通知*A*2关于ϕ。没有描述*A*2是如何表达这个意图的，也没有描述*A*1如何表达以让*A*2知道。目标表示和意图表达涉及到的装置比在这本书中提到的还要多一些。幸运的是，通过谈话（*talking*）代替*A*1打算施加于*A*2的影响巧妙地解决这个问题，*A*1想让它的交谈使*A*2相信ϕ。讲话方想施加给听众的影响被称为交谈的言语表达效果（*perlocutionary*）的影响——和它的illocutionary相对照，后者是交谈的实际效果。当然，言语表达效果影响的实现完全受控于听众，只有听众是极端地轻信或者听众相信讲话方是值得信赖的情况下，才能假定言语表达的效果。回到例子中，可以通过假设*A*1假定它的动作（通知*A*2关于ϕ）有言语表达效果来简化我们的讨论。我们假定*A*1用公式$\mathbf{K}(A2,\phi)$表达它对*A*2的影响。

当人类使用语言时，一个语句的言语表达（打算的）效果有时会产生另外的效果。例如，在句子You left the refrigerator door open中，讲话方的真正意图是要求受话方关上冰箱。交谈的言语表达效果不同于所谓的间接交谈（*indirect speech act*）。一个使用间接交谈的讲话方假定一个受话方能从交谈的环境中推断出讲话方的意图，并用这个意图去决定交谈的言语表达效果。因此，Do you have the time?是一个间接的问话方式（因此被认为更有礼貌），它要求受话方告诉讲话方当前的时间，而并非受话方是否知道时间。

24.1.1　计划交谈

我们能像对待其他的agent动作一样对待交谈。我们的agent能用一个计划产生系统产生由交谈和其他动作构成的计划。为此，需要一个这些动作的结果模型。例如，考虑一个表示型的交谈Tell(α, ϕ),在这里，我们的agent通知agent α，ϕ 是真的。通过STRIPS规则，对该动作的结果建模：

Tell(α, ϕ)

PC: Next_to (α) $\wedge \phi \wedge \neg$**K**(α, ϕ)

D: $\neg$**K** (α, ϕ)

A: **K** (α, ϕ)

前提Next_to (α)确保我们的agent α是充分的接近agent α，能够进行可靠的通信；强加前提ϕ以确保我们的agent在通知另一个agent那个事实前确实相信ϕ；而前提条件¬ K(α, ϕ)确保我们的agent不会传达多余的信息。假定动作有其言语表达影响，即α知道ϕ。假定在环境On(A, B) ∧On (B, C) ∧On(C, Fl)中，我们的agent的目标是On (B, Fl)，还假定我们的agent 知道无论何时当*B*在*C*上、且*B*上无任何东西时，agent *A*1能相应地将*B*移到地板上。用普通的积木世界的STRIPS规则和前述的Tell规则，我们的agent可构造计划{Move(A, B, Fl), Tell (A1, Clear (B) ∧On (B, C))}。

24.1.2　实现交谈

交谈的实现或物理表现是讲话，现在必须讨论一下通信动作，如Tell(α, ϕ)，是如何像讲话方和受话方之间的交谈一样被传输的。考虑两种可能性：（a）从讲话方到受话方的一个逻辑公式的直接传输；（b）讲话方讲到一些符号字符串，然后受话方将它们翻译成它的认知结构（也许翻译成一个逻辑公式）[⊖]。

如果讲话方和受话方共享同一类基于特征的世界模型，使用相同符号的逻辑公式，那么一个交谈就能通过传播一个逻辑公式来实现（加上交谈的类型信息）。在这种情况下，例如，交谈Tell ((A1, Clear (B) ∧On (B, C))就能通过让我们的agent向*A*1发送公式Clear (B) ∧On (B, C) 和一个表示类型的指称来实现。注意，这样做要假定公式Clear (B) ∧On (B, C)对*A*1表达的意思与它对我们的agent 表达的意思相同（短语“means the same thing”暗示了数据库中的所有公式对我们的agent 和*A*1的逻辑模型是充分的相同）。甚至在我们正在对所有的agent（我们的和其他的agent）构建和编程的情况下，相同的知识表示词汇的假设，对一些有趣的环境来说也是非常难以令人相信的。即使两个agent的知识表示词汇和模型在启动时是一致的，如

⊖　符号字符串只是可以被传输的东西的一个例子，人和人之间也可以通过图表、图片和其他的媒体传输信息。

果遇到任何新的对象，这些agent也几乎不可能会给这些对象相同的内部名字。如果agent能够发明新的谓词，这些谓词按照更原始的方式定义，而这些等价的谓词也可能会有不同的符号。因为我们的agent刚好要对另一个agent 采取一个有意的姿态，这样另一个agent就不一定要用逻辑公式对它的世界模型编码。

在这些条件限制下，agent之间的通信如何发生呢？一种答案涉及到早期阐述过的两种选择中的第二个：用一个一致的、通用的通信语言，通过设计、使用和指令，通信agent能学会在这种通用语言方式下传输的符号字符串是如何改变其他 agent的认知结构的。采用一个有意的立场，我们的agent 能够预测，例如，如果传输给agent *A*1的字符串是`block B is on block C and block B is clear`，那么对*A*1的认知结构的言语表示效果影响能用我们的agent 的知识库中的公式**K**(A1, On(B, C)∧Clear (B))描述（不管*A*1实际上是如何表示那个知识的）。在这个例子中，我们的agent通过发送讲话`block B is on block c` 和`block B is clear` 来执行交谈Tell (A1, On (B, C)∧Clear (B))。听众把讲话翻译成它用来表示这个知识的任何内部形式。当然，由于假定正在设计这些agent，我们能选择我们喜爱的任何通用通信语言。如果我们的agent也需要用一些像英语一样的语言与人通信，就可以发明一个如例子中所说的类似于英语的语言。

使用基于符号字符串的语言预示了两个问题的解决方案：给定一个交谈，如何生成一个符号字符串；如何把一个符号字符串翻译成对一个认知结构的影响。虽然鼓励在人工agent中使用符号字符串作为通信媒体，但是机器生成和自然语言（人类使用的语言，像英语、法语和汉语等）的理解是依各自的兴趣的。通信使用的符号字符串的生成和理解主要在自然语言的领域学习研究。因此，对agent之间通过字符串的通信，处理将主要集中在类似于英语的语句讲话之中。自然语言的处理、生成和理解极其困难，到目前为止只取得了有限的进展。人类级的能力毫无疑问是解决大量AI问题的先决条件，在下一章我们将看到这个，不过已有一些应用已经是可行的。在下一部分对使用语言理解系统的一些重要的技术进行了归纳（省略了讨论语言生成，对这方面感兴趣的读者可以参见[Appelt 1985, Sadek, et al, 1996]，它谈到了使用计划和推理生成自然语言语句，[McKeown & Swartout 1987]对这个技术做过评论，[McDonald & Bolc 1988]关于这个主题出过一本书。已有几个商用计算机接口系统具备一定的产生文本和讲话的能力）。

24.2 理解语言字符串

下面要考虑一个符号字符串——由一个企图传递一定命题的讲话方传播，是如何被一个受话方翻译成一个指称（我们希望）相同命题的公式的。这个翻译过程所需的一些（但不是全部）信息被嵌入在字符串的语法属性中。例如，一个讲英语的讲话方可能希望将字符串`block B is on block C`翻译成On (B, C)而不是On (C, B)。这些语法特性用句法描述最方便，句法不仅限定了哪些字符串是该语言的合法句子，而且定义了句子的结构。使用一个技术简单的“原型英语(*proto-english*)” 的例子来说明这些思想，它不但是介绍自然语言理解技术的第一步，同时也是可能的agent语言的一个例子。

24.2.1 短语结构语法

短语结构语法定义了符号串的基本组成——符号本身是如何被集成短语，以及这些短语是

如何最终集成为句子的。句子被分解成它的构成短语，最后终止于字符串符号级，这种方式定义了一个句子的结构，这个结构是把一个句子翻译成一个逻辑公式的关键。语言的元件是终结符号和非终结符号。最高级的非终结符是句子，由符号S表示。每个非终结符被定义为一组其他的符号——非终结符或终结符或者两者的组合。在所谓的上下文无关（*context-free*）的语法中，一个非终结符的定义在一个字符串中独立于它周围的符号。下面示例了一个上下文无关的语法例子，它适合于堆积木问题中的有限通信。

一个句子*S*被定义为某个名词短语（*NP*），其后面跟随一个动词短语（*VP*）。这个定义由下面的规则指称：

S ← *NP VP*

我们也允许由两个句子的合取（*Conj*）递归地构成一个句子：

S ← *S Conj S*

这两个规则能简化为巴科斯—诺尔范式（BNF）：

S ← *NP VP* | *S Conj S*

其中“|”读作“或”

现在，我们必须定义刚刚引入的非终结符号*NP*、*VP*和*Conj*。其中一些非终结符将由另外的非终结符定义，依次类推，直到所有的定义都终止于终结符号。一个合取可以被定义为终结符and或者or：

Conj ← and | or

一个名词短语被定义为一个名词（*N*）或者一个由名词跟随的形容词（*Adj*）

NP ← *N* | *Adj N*

对于简单的三积木块问题，一个名词是终结符A、B、C或者Floor之一（在积木问题中实体的通用名称）。我们也允许终结名词block A, block B 和block C[⊖]，这些定义能用下面的规则指称：

N ← A | B | C | block A | block B | block C | Floor

对形容词的定义如下：

Adj ← clear | empty | occupied

动词短语的定义如下：

VP ← is *Adj* | is *PP*

最后，介词短语（*PP*）和介词（*Prep*）被定义为：

PP ← *Prep NP*

Prep ← on | above | below

为方便起见，把所有的这些规则收集在表24-1中，在自然语言的处理系统中，终结符号是语言中的所有单词，它们被存在一个被称为词典（*lexicon*）的数据库中。

这些语法规则允许的句子结构可以方便地用树来显示。例如：句子block B is on block C and block B is clear的结构如图24-1所示。在一个完整的结构中，每一个非叶节点由一个非终结符号标识，且有由语法规则生成的后代。每个叶节点由一个终结符号标识。由非终结符号标识的节点是方形的，终结符标识的节点是圆形的。

⊖ 从技术上讲，block 和A 的组合被称为名词—名词组合。我们仅把它们做为附加的名词来简化讨论。

根据语法规则，可以确定下面的字符串是合法的句子[⊖]。

block C is occupied

A is on C

B is on occupied C

occupied B is on clear C

决定一个任意的字符串是否是一个合法的句子被称为解析(*parsing*)字符串，解析过程被称为语法分析（*syntactic analysis*）。存在着各种分析算法。在一个自上而下（*top-down*）的算法中，语法规则应用于（在一个“向后”的方向）非终结符*S*，按照它的构成短语重写它，直到产生一组匹配给定字符串的终结符。这个过程可以通过一个AND/OR树进行搜索，直到找到一个解决方案树（图24-1中的树就是这样一个解决方案树）。在一个自底向上（*bottom-up*）的算法中，被分析的字符串的子串用非终结符代替，这些非终结符被其他的非终结符代替（均依照语法规则），直到产生单个非终结符*S*。为了提高搜索效率，这个过程常常顺着字符串按从左到右的方式进行。这个过程能用深度优先、回溯搜索实现。

表24-1 堆积木问题的通信语法

S ← *NP VP* | *S Conj S*
Conj ← and | or
NP ← N | *Adj* N
N ← A | B | C | block A | block B | block C | floor
Adj ← clear | empty | occupied
VP ← is Adj | is PP
PP ← *Prep NP*
Prep ← on | above | below

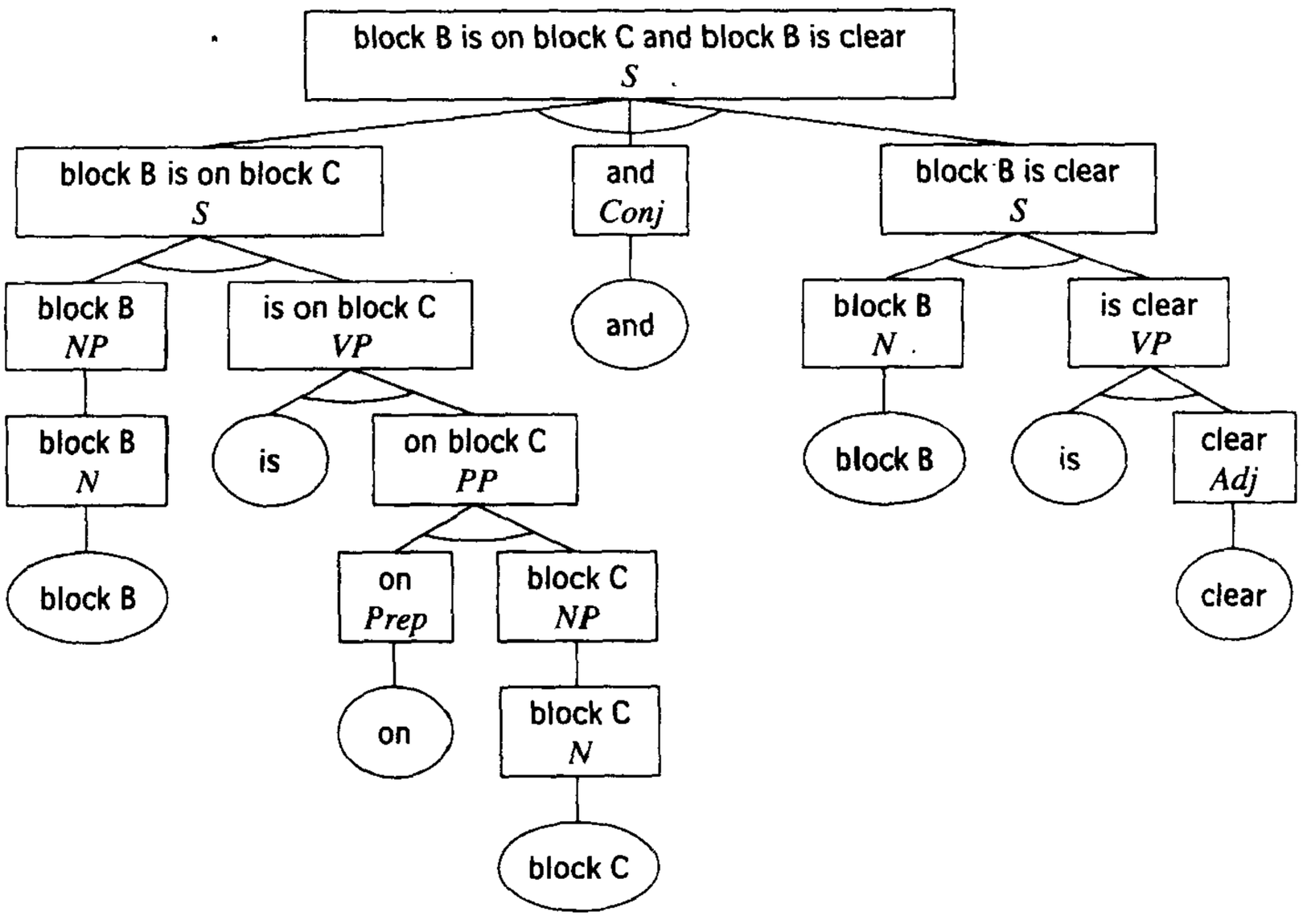


图24-1 一个句子的结构

24.2.2 语义分析

回忆一下，传输一个符号字符串的目的是该字符串能被一个受话方翻译成一个逻辑公式。前面已经指出字符串的语法结构能表达很多的预期含意。在此描述一种翻译方法，它是语法分

⊖ 为简单起见，这里忽略了大写和标点符号，尽管引入这些特征的文法也能写出来。

析过程本身的一个附加的效果。这个附加的好处通过把逻辑公式的元件与句子的每个短语连接起来而获得。短语结构语法的每一个规则指定了和一个给定的短语相关的公式元件是如何由与组成的子短语相关的公式元件构成的。例如：规则

PP ← Prep NP

必须按照*Prep*和*NP*的语义关系指定*PP*的语义关系。这些语义关系通过把每个非终符作为一个函数表达式来表示，函数表达式用语义关系作为参数；例如，*PP*（*sem*）。

作为分析的结论，和非终结符*S*相关的公式就是字符串的含意。使用这些关系的，语法称为扩充短语结构语法(*augmented phrase-structure grammar*)，分析过程完成了所谓的语义分析（*semantic analysis*）（以及常说的语法分析）。

通过扩充我们的短语结构语法的例子和对已经分析过的例句进行语法分析来说明这个过程。从将逻辑结构和每个终结符联结开始，先看一下那些语法规则分类为名词的终结符号。从直觉上讲，名词应该和谓词演算中对象常量相关。对这个简单的语法，这个关系通过将规则*N* ← A|B|C|block B|block C|Floor 如下表达来实现：

A → *Noun*(E(A))

B → *Noun*(E(B))

C → *Noun*(E(C))

block A → *Noun*(Block(A))

block B → *Noun*(Block(B))

block C → *Noun*(Block(C))

floor → *Noun*(Floor(Fl))

例如其中第一个规则，它陈述了和名词“A”相关的语法元件是原子E（A）（E(A)的预期意思是由A指称的对象是一个“实体”）。在语法分析中，这个规则陈述了终结符A在符号串的出现能被重写为*Noun* (E(A))。按从左到右的方式写规则（用一个向右的箭头），因为这些规则打算应用到“自底而上”（用箭头右边的公式代替终结符）的分析中。

如果在我们的例子中执行由这些规则指定的一些替换，会得到下面的部分已分析的句子：

Noun(Block(B)) is on *Noun*(Block(C)) and *Noun*(Block(B)) is clear

下面是我们为终结符and和or写的扩展规则：

and → *Conj*(∧)

or → *Conj*(∨)

应用and的规则生成：

Noun(Block(B)) is on *Noun*(Block(C)) *Conj*(∧) *Noun*(Block(B)) is clear

公式的元件并非与其他的终结符直接相关。形容词clear、empty和 occupied都陈述了一些对象的属性，因此它们应该用适当的关系常量和对象常量引入谓词演算原子。但是，在我们代替这些形容词时，可能不知道它们正在描述哪个对象的属性。因此，对形容词clear，它应当有如下的规则

clear → *Adj*(Clear(*x*))

表达式clear（x）将被解释为一个谓词演算范式，为了生成一个谓词演算公式，该范式需要

应用到一些对象常量上。就像计算机科学中的其他领域一样，我们用lambda表达式定义这种范式。因此，对clear规则的一个更精练的描述是：

$$\text{clear} \rightarrow Adj(\lambda x\ \texttt{Clear}(x))$$

在这里我们看到一个短语的“含意”有时用另一个短语的含意来表达。例如，如果我们把该范式应用到对象常量B，就会得到：

$$(\lambda x\ \texttt{Clear}(x))\texttt{B} = \texttt{Clear(B)}$$

把clear规则应用到我们已经部分解释的句子，产生：

$$Noun(\texttt{Block(B)})\ \text{is on}\ Noun(\texttt{Block(C)})\ Conj(\wedge)\ Noun(\texttt{Block(B)})\ \text{is}\ Adj(\lambda x\ \texttt{Clear}(x))$$

(对其他的形容词我们有相似的规则。)

在给出其他终结符的规则之前，我们先看看非终结符——名词短语。首先，一个名词是一个名词短语（在语法上没有任何改变）：

$$Noun(\phi(\sigma)) \rightarrow NP(\phi(\sigma))$$

其中$\phi(\sigma)$是一个一元原子（带有模式变量σ）。应用这个规则产生

$$NP(\texttt{Block(B)})\ \text{is on}\ NP(\texttt{Block(C)})\ Conj(\wedge)\ NP(\texttt{Block(B)})\ \text{is}\ Adj(\lambda x\texttt{Clear}(x))$$

一个名词短语也是一个跟随着名词短语的形容词。由于将在后面的归纳表中给出，在此就不需用该规则来分析我们的示例句子。

在语法例子中，只有一个动词is。它总是和一个形容词或一个介词短语合用。当使用一个形容词时，我们有规则

$$\text{is}\ Adj(\lambda x\ \phi(x)) \rightarrow VP(\lambda x\ \phi(x))$$

在这个规则中，ϕ是一个一元关系常量，指称该形容词的相关属性（在这个简单语言中的单词没有加任何附加的语法内容）。

在给出动词短语的其他规则之前，先给出一个名词短语是如何与一个动词短语组合而产生一个句子的：

$$NP(\phi(\sigma))VP(\lambda x\ \psi(x)) \rightarrow S((\lambda x\ \psi(x) \wedge \phi(\sigma))\sigma)$$

其中，ψ是任何关系常量（在我们的语法中它可以是任何元），当应用lambda表达式时，ψ的lambda变量x被约束。这里我们有了组合两个构成短语的语法的第一个实例。这个组合通过把来自动词短语的一个lambda 表达式应用到由名词短语给出的对象常量上而获得。这个应用能在语法分析产生下面的精简规则之前被使用：

$$NP(\phi(\sigma))VP(\lambda x\ \psi(x)) \rightarrow S(\psi(\sigma) \wedge \phi(\sigma))$$

继续应用这些规则会产生

$$NP(\texttt{Block(B)})\ \text{is on}\ NP(\texttt{Block(C)})\ Conj(\wedge)\ NP(\texttt{Block(B)})\ VP(\lambda x\texttt{Clear}(x))$$

$$NP(\texttt{Block(B)})\ \text{is on}\ NP(\texttt{Block(C)})\ Conj(\wedge)\ S(\texttt{Clear(B)} \wedge \texttt{Block(B)})$$

接下来，我们解决介词和介词短词

$$\text{on} \rightarrow Prep(\lambda xy\ \text{On}(x, y))$$

(注意on引入一个二位（*two-place*）谓词。Lambda符号跟踪哪个变量是哪一个)。

$$Prep(\lambda x\gamma\ \psi(x,\gamma))NP(\phi(\sigma)) \rightarrow PP(\lambda x\ (\lambda\gamma\ \psi(x,\gamma) \wedge \phi(\sigma))\sigma)$$

它能被简化为：

$$Prep(\lambda x\gamma\ \psi(x,\gamma))NP(\phi(\sigma)) \rightarrow PP(\lambda x\ \psi(x,\sigma) \wedge \phi(\sigma))$$

其中ψ是与介词相关的任何二位关系常量。

动词短语的其他规则是

$$\text{is}\ PP(\lambda x\ \psi(x,\sigma)) \rightarrow VP(\lambda x\ \psi(x,\sigma))$$

(注意，ψ是一个以模式变量σ 和lambda变量x为参数的二元关系常量)。我们继续应用这些规则和前面对S的规则产生：

$$NP(\texttt{Block(B)})\ \text{is}\ Prep(\lambda x\gamma\ \texttt{On}(x,\gamma))\ NP(\texttt{Block(C)})\ Conj(\wedge)\ S(\texttt{Clear(B)} \wedge \texttt{Block(B)})$$

$$NP(\texttt{Block(B)})\ \text{is}\ PP(\lambda x\ \texttt{On}(x,\texttt{C})) \wedge (\texttt{Block(C)})\ Conj(\wedge)\ S(\texttt{Clear(B)} \wedge \texttt{Block(B)})$$

$$NP(\texttt{Block(B)})\ VP(\lambda x\ \texttt{On}(x,\texttt{C})) \wedge (\texttt{Block(C)})\ Conj(\wedge)\ S(\texttt{Clear(B)} \wedge \texttt{Block(B)})$$

$$S(\texttt{Block(B)} \wedge \texttt{Block(C)} \wedge \texttt{On(B,C)})\ Conj(\wedge)\ S(\texttt{Clear(B)} \wedge \texttt{Block(B)})$$

最后，我们有联接两个句子的规则：

$$S(\gamma_1)Conj(\wedge)S(\gamma_2) \rightarrow S(\gamma_1 \wedge \gamma_2)$$

应用这个规则，重新整理，简化后产生

$$S(\texttt{On(B,C)} \wedge \texttt{Clear(B)} \wedge \texttt{Block(B)} \wedge \texttt{Block(C)})$$

显示这个句子的所有语义分析过程的语义分析树如图24-2所示。为了清楚直观，谓词演算公式显示在相邻的对应节点上（而不是做为相关的非终结符的参数）。注意和短语相关的语义结构是如何由子短语语义结构构成的。我们说这类扩充的语法是合成语义（*compositional semantics*），为了参考方便，这个语法的所有规则被收集在表24-2中。

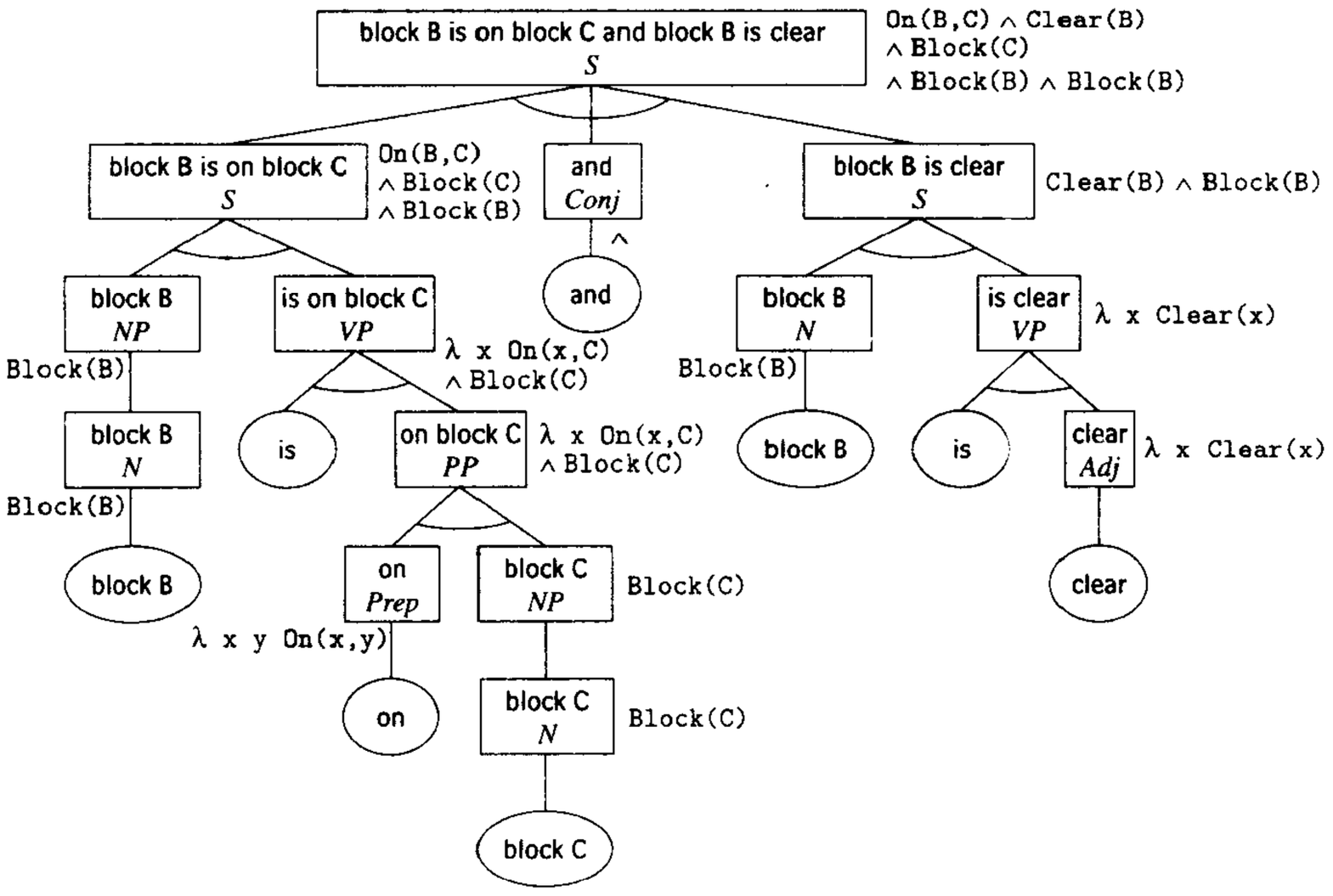


图24-2　一个语法分析树

表24-2 一个扩充的短语结构语法

A → *Noun*(E(A))
B → *Noun*(E(B))
C → *Noun*(E(C))
block A → *Noun*(Block(A))
block B → *Noun*(Block(B))
block C → *Noun*(Block(C))
floor → *Noun*(Floor(Fl))
and → *Conj*(∧)
or → *Conj*(∨)
clear → *Adj*(λx Clear(x))
empty → *Adj*(λx Clear(x))
occupied → *Adj*(λx ¬Clear(x))
on → *Prep*(λxy On(x, y))
above → *Prep*(λxy On(x, y))
below → *Prep*(λxy On(y, x))
is $Adj(\lambda x\ \phi(x)) \rightarrow VP(\lambda x\ \phi(x))$
is $PP(\lambda x\ \psi(x, \sigma)) \rightarrow VP(\lambda x\ \psi(x, \sigma))$
$Prep(\lambda xy\ \psi(x, y))NP(\phi(\sigma)) \rightarrow PP(\lambda x\ \psi(x, \sigma) \wedge \phi(\sigma))$
$Noun(\phi(\sigma)) \rightarrow NP(\phi(\sigma))$
$Adj(\lambda x\ \phi(x))NP(\psi(\sigma)) \rightarrow NP(\phi(\sigma) \wedge \psi(\sigma))$
$NP(\phi(\sigma))VP(\lambda x\ \psi(x)) \rightarrow S(\psi(\sigma) \wedge \phi(\sigma))$
$S(\gamma_1)Conj(\wedge)S(\gamma_2) \rightarrow S(\gamma_1 \wedge \gamma_2)$

24.2.3 扩展语法

虽然表24-2中的语法能够把很多积木世界中的句子翻译成逻辑公式，但构造它只是为了说明这种类型的语法分析的一般思想。在不会产生过于复杂的情况下，可以加几个明显的附加规则。例如：为附加的积木和agent加上其他的规则，使具有更多形容词(red, heavy ,...)、更多的介词（next to, between,...）和更多的名词。这个语言中的惟一动词is没有起到重要的作用。可以加上动词know，它会翻译成一个使用**K**的模式范式。翻译句子Sam moves block A from block C to block B 中的其他动词将使我们决定如何构思这种动作和如何用逻辑公式描述它们。加入时态动词（如moved）将要求翻译成一个能够描述时态事件的公式。

含有冠词，如the和a的句子常常涉及到翻译成量化公式。例如：block A is on a block 应该被翻译成(∃x) On(A, x)。在考虑量词的范围时，包含every、all和some这些单词的英语句子是具有二义性的。例如，句子all blocks are on a block 可能被译成（∀x）(∃y) On(x, y)或者（∀y）(∃x) On (x, y)。常常通过参考其他的知识源——不包含在该句子中的知识来解决这种二义性。在后面，更详细地讨论了二义性问题和它们的解决方案。语义分析通过先把它翻译成一个特殊的准逻辑范式（*quasi-logical form*）[Russell & Norvig 1995, pp. 676以后]语言（能够保留一定的二义性），而不是直接翻译成一阶逻辑公式（一般不能容纳二义性）来推迟对二义性的消解。然后，准逻辑范式到一阶逻辑的翻译可以在一个单独的阶段进行，它可以利用附加的知识来消解剩下的二义性。

通常，不能用上下文无关语言充分定义自然语言中的句子。例如，考虑句子block A and block B are on block C，接受一个带有上下文无关的规则*S* ← *NP VP*的语法句子也可以接受block A and block B is on block C，因为is on block C将作为一个合法的*VP*被接受。在主语和动词之间加强单—复数协议也需要附加的上下文无关规则来定义单数（复数）名词短语和单数（复数）动词短语，或者它需要一种上下文有关文法。有趣的是，允许相关的短语语义范式的相同扩充技术也能被用来加强各种上下文相关约束，包括那些涉及到数量（单数、复数）、人称（第一、第二或第三）、时态、动词类型（及物的或不及物的）和情形（主观、客观）的情况。为了加强单—复数协议，例如我们可以加一个额外的参数n（数字）给适当的短语名称。于是对句子的规则就变成*S*（*n*）←*NP*(*n*) *VP*(*n*),为简单起见，这里忽略了指定语义范式的参数。当应用这个规则时，对变量*n*的绑定，不管s（单数）还是p（复数）都必须一致。在*VP*（*n*）中绑定*n*将由分析树底部的下述规则决定：are ← *verb*(*p*)或 is← *verb*(*s*)。在*NP*（*n*）中绑定*n*也被同样确定。使用这种机制的文法叫做合一文法（*unification grammars*），因为术语“合一”就是在变量之间的强制统一。

24.3　有效通信

通信的实际效果常常能通过受话方使用其自身的知识来帮助决定一个谈话的意思来获得。“对一个聪明人而言一个单词就够了（A word to the wise is sufficient）”。如果一个讲话方知道一个受话方能明白他讲的意思，那么他就能发送更短更少的信息。计算机难以理解自然语言的一个主要原因就是理解需要很多知识源，它包括关于通信发生的上下文知识，还有讲话方和受话方共享的“共识”知识。在接下来的几个小节中，将简要谈论一下这些主题。

24.3.1　上下文的使用

如果受话方和讲话方共享相同的上下文（也就是说他们都知道彼此正在交谈的环境），那么该上下文能被作为一个知识源来决定一个谈话的意思。使用上下文允许语言中有代名词（如he和it）和索引词（*indexical*）（如here, now, I和you）。上下文可以包括前面的通信、当前的环境状态，或者两者都有。例如，考虑谈话block A is clear and it is on block B。受话方在已经听到且理解了短语block A is clear的上下文中处理短语it is on block B。受话方可以假定it指的是已经提到的那个积木A[⊖]。在示例句子中，单词it（代替重复的单词block A）总共只使用了一次，只是一个很小的节约，但是大量相同的节约加起来就能提高效率。当然，为了得到这个效率，讲话方必须知道受话方将能够明白那个指代。

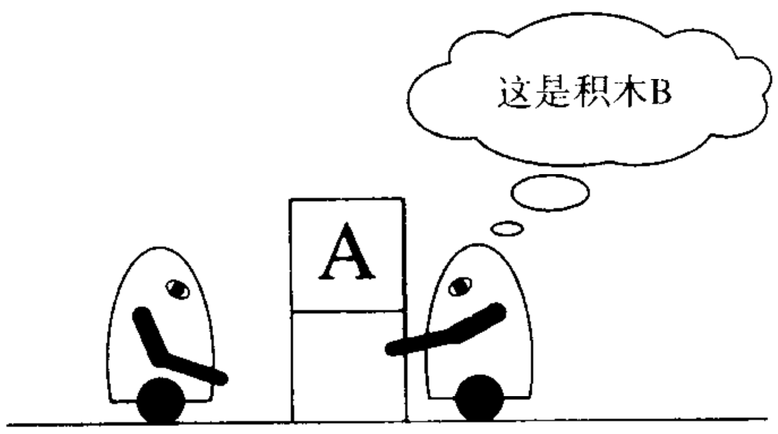


图24-3　指示建立一个索引词的含意

当一个讲话方使用索引词I时，在相同上下文中的受话方知道单词I指的是哪个人或机器。例如，如果机器人R1说I know that

⊖ 语言学家用术语anaphora 描述那些已经提到过的短语（包括代名词）。一般地讲，决定首语重复项的意思极其困难，这个例子是一个非常特殊的简单的例子。

block A is on block B，那么当R1正在讲话时，一个知道机器人R1身份的受话方能把这个谈话解释为K（R1，On(A,B)）。有时，像this这样的索引词被用来与指示动作结合使用。例如，在图24-3中，我们看到一个机器人指着一个积木说this is block B。看到讲话的机器人指向一个积木，收听的机器人把那个讲话解释为On（A，B）。讲话方这样讲因为它想让受话方知道On（A，B），并且它知道受话方看不到积木 *B*上的标签，但能看到积木*A*上的标签，且能看到*A*在那个被指定的积木上面。我们会再次得到一个小的节约，但对有效的通信还是有点贡献的。

24.3.2 使用知识解决歧义性

即使能够设计agent通信语言，使得命题能被传输和无二义性地、单一地理解，这种无二义性的通信将需要很难管理的大量词汇和极其复杂的表达。使用自然语言的人能用各种知识源来洞悉有歧义性的单词和短语的预期含意，这种能力允许人类在使用我们的语言时能更有效，而且它也许能赋予agent通信相似的好处。

计算语言学家已经对自然语言交谈中的歧义性进行过分类。下面用积木问题和机器人（和人）在积木问题中的通信来描述这些类型（在这些例子中用到的语言和语义学比表24-2中更复杂）。

1. 词汇歧义

同一个词可能有几个不同的意思，由此导致的歧义性叫做词汇歧义（*lexical ambiguity*）。句子robot R1 is hot既能表示R1非常擅长它的工作，也能表示R1非常热。解决这种歧义性将需要关于R1的附加知识和句子出现的环境。

2. 句法歧义

有时句子能用不止一种方式表达，即短语能被不同地组合。由此产生的岐义性称为句法岐义（*syntactic ambiguity*）。句子I saw R1 in room 37既能暗示讲话方正在进行看这个动作，也能指R1是在37号房间。如果受话方已经知道讲话方或者R1所指的位置，就能解决歧义性（这种问题的一个典型例子是I saw the Grand Canyon Flying to New York。这个句子的歧义性可通过常识知识有人飞走了而不是Canyons来解决）。词汇歧义性会产生不同的表达结果，在句子clear block A and B and C are on the Floor中，clear既能是动词也能是形容词。这类问题的一个典型例子是 time flies like an arrow。

3. 引用歧义

代名词和其他首语重复（*anaphora*）的使用可能会产生歧义性。解决这类引用歧义性（*referential ambiguity*）的过程涉及到对讲话方和受话方的部分内容进行复杂推理。在句子block A is on block B and it is not clear中，讲话方可能想让it指的是积木 *A* 而不是积木 *B*。讲话方可以合理地假定受话方将使用常识知识从短语block A is on block B中推断出¬ clear(B)，因此也会得出结论it在短语it is not clear中指的是积木*A*(因为讲话方经常不会传送冗余的信息)。

4. 语用歧义

使用上下文知识和其他知识解决歧义性的处理过程常被称为语用分析（pragmatic analysis）(相对于句法和语义分析过程)。但是，如果一个受话方的共识知识和上下文知识是不确定的，即使语用分析也不能解决歧义性。这种歧义性称为语用歧义（*pragmatic ambiguity*）。在句子

R1 is in the room with R2中，如果受话方认为R2是在37号房间或38号房间中，但不能确定是哪一间，那么短语the room的意思将是岐义的。

24.4　自然语言处理

在积木问题中，agent的通信仅仅是在一个能够理解由人类讲的（或写的）句子的系统中所需要的很少的一部分。自然语言处理（NLP）是一个极有潜力的应用领域，包括语言翻译，从数据库中浏览信息、人机接口和自动听写（*automatic dictation*）。具有语言能力的计算机系统将能够理解和产生在文本（即句子字符串）或在交互式交谈中所写和所说的句子。尽管已有一些研究系统和一些能力有限的商用产品，但是这个领域还远远不能达到上面的那些目标。

NLP已被描述为“AI 难题”（类似于NP 难题）。也就是说，为了产生一个具有和人类一样语言能力的系统将需要解决“AI问题”。大部分困难在于解决语用歧义性，它要求对大量的常识知识库进行推理。当然，在建立词汇表、文法和适合处理自然语言的分析系统中也有一些较大的困难。

几乎所有来自文本和交谈的句子都能用来说明这些问题。让我们看看一个有语言理解能力的系统为了分析一个取自交谈的句子都需要些什么（这里只略微谈到很少的几个问题！）考虑下面的在一个教授和一个研究生之间的一段谈话。

```
P: well, I'll need to see your printout
S: I can't unlock the door to the small computer room to get it.
P: Here's the key.
```

一个计算机系统如何分析那个学生的话呢？很多NLP系统被组织成一个步骤序列，首先，用一个带有语义组合规则的文法和一个词典，将句子翻译成逻辑范式。下面仅是该句子在这个处理级所处理的一些问题：

- 词典可能有像lock这样的词根和它的各种含意。它也需要关于前缀和后缀如何连接到词根的信息以及这些连接如何影响词根的含意的信息（形态学分析*morphological analysis*）。
- 文法和语义分析需要能够处理比简单的积木问题文法范围更广的各种交谈。像quickly这样的副词会改变它们修饰的动词的含意。考虑一个形容词，如small，它的意思会根据它修饰的名词而变化，一个小屋子可能比一个大计算机更大。
- 短语the small computer room可能是说一个小屋子中有计算机，也可能是小计算机放在屋子中。为了推迟解决这种歧义性，系统或者生成一个保留歧义性的逻辑范式或者生成两个不同的逻辑范式。
- 系统需要能够找到I和it的指代物。I可能会被容易地确定为讲话方。在这一级分析中，it可能指的是门、房子或者前面谈话中提到的任何东西（将要确定）。消解歧义性可能被再次推迟。
- 解释单词can't，即cannot的缩写形式，要求系统有处理否定的能力。在这种情况下，系统也要能解释can指的是自然的能力还是被允许。

在产生可选的逻辑范式后，系统用一般环境知识推理解决一些歧义性。这里，可以确定it一定指的是前面谈话中的某个东西，因为讲话方不可能去“get”一个门或一间屋子。同样，can可能指的是自然的能力，因为打开一个门需要钥匙，而钥匙是人们常常没有的东西。

系统也用确定的当前环境知识来解释短语small computer room。假如系统知道所有指定的计算机屋子都有小计算机，而只有一个房间是小房间，比如说246号房间；那么small computer room 一定指的是246号房间。

这个句子的最终分析需要谈话的知识和对讲话方意图的推理能力。教授要求打印结果，因此，it一定指的是它。这里的推理链相当复杂。对打印结果的要求为学生建立一个目标，可以假定那个学生要制定一个计划去到达那个目标。当那个学生意识到他没有进到放打印结果的屋子时，他变得支支吾吾。那个句子是关于学生不能到达目标的原因，系统也必须那样解释它，而且，系统需要解释学生的讲话是企图协商对目标的变更呢（不要烦我去拿打印结果因为我没有钥匙，也许你（教授）也不是真的需要打印结果），还是想要（间接的）那个屋子的钥匙（学生知道教授有那个房间的钥匙）以便为学生能生成一个可以执行的计划去达到目标。间接要钥匙也包含了为什么要钥匙的原因——学生假定教授需要一个原因被说服以给他钥匙。

执行所有这些推理的方法和表示被推理的知识的方法仍然是人工智能的前沿研究领域。因此，让计算机理解像我们的例子一样的谈话确实是一个**AI**难题。由于语言理解的这些困难可能还必须与一些反应型处理组合起来，这使语言理解问题变得更加困难了。例如，前述情况下的学生可能只说了一个不合文法规则的Key? 来代替那个更长的句子，教授可能的反应是把合适的计算机房的钥匙给学生——几乎不需要什么分析！但是，这个反应要求教授决定移交哪个钥匙，因此不知道究竟是用复杂的推理做出那个结论还是简单地根据相关的上下文反应过程做出结论。

自然语言处理，包括语音、文本的产生以及理解，是一个既有自己的惯例和技术，又与AI相辅相成的领域。在此不做详述，若对更详细的细节感兴趣，可从[Russell & Norvig 1995，第22和23章]和[Allen 1995，Grosz, Sparck Jones, & Webber 1986]开始学习（也可参见NLP的专刊《Artificial Intelligence》，vol.63，nos.1-2，October 1993）。

24.5 补充读物和讨论

[Cohen & Perrault 1979]描述了用AI计划系统如何来计划交谈。为了让一个谈话达到它的预期表达效果（即讲话方的期望效果），一个受话方有时可能必须通过观察讲话方的一系列动作才能猜出讲话方的总体意思。[Kautz 1991]评论了这个问题，并对计划识别和实现提出了一个形式理论。

[Chomsky,1965]介绍了语言的句法和分析的基础。对字符串的语法和语义处理是建立在由[Pereira & Warren 1980]最先研制的明确子句文法之上。还有很多其他的语法形式，包括[Woods 1970]的扩充转换网络（ATN）。一个对英语进行解释的典型大型文法形式是用在SRI国际TEAM系统中的TEAM[Grosz, et al. 1987]。

[Magerman 1993]描述了学习语法的统计方法，[Charniak 1993]推广了可能和那些规则有关的一个语法概念。

[Grosz , Sparck Jones & Webber 1986]和[Waibel & Lee 1990]分别是关于自然语言处理和语音识别的重要论文集。

和语言理解方法相比，基于向量的文档词频统计常常用来根据内容把文档分成有意义的类别[Masand , Linoff & Waltz 1992, Stanfill & Waltz 1986]。

习题

24.1 你能想像这样一种场合吗？一个agent 讲了一句tall，而且该agent 知道这个动作的命

题内容不是真的。

24.2 用表24-2中的文法（语义规则）对下面的句子进行语义分析：

```
block B is on floor or block B is on C
```

24.3 如何修改表24-2，使它包含单词not以便not既能与形容词（如在not clear中）也能与介词（如在not on中）连用。

24.4 简单讨论在自然语言理解和情景分析问题中大致相似之处。

24.5 机器人*A*有词典和表24-2中的语义句法。解释一下机器人*A*如何建立一个积木环境，并且在那种方式下交谈以便一个有理解能力的机器人*B*能推导出合适的语法（提示：通过让机器人*A*指向积木*B*同时发出声音*B*开始）。你必须假定机器人*B*具有什么能力？

第25章　agent 体系结构

前面采用agent讨论了人工智能。虽然主要讲agent，但也强调用于建立智能agent的技术也能应用于其他的领域：图象分类和分析、专家咨询和推理系统，调度和计划、自然语言处理系统，等等。常假定讨论中的agent是机器人，但是很多思想也可以应用到非物理agent。在这最后一章，将致力于如何将各种AI技术集成到智能agent结构中。

但是首先必须问一个问题，一般地讲，关于智能agent的体系结构是不是有一个统一的说法？就像有成千上万种动物占据着成千上万个不同的小环境一样，我们同样希望有很多很多的人工agent，它们执行着无数项人类想要它们做的任务。它们的结构的确切形式将取决于任务和执行这些任务的环境。例如，一些agent在一个强调时间的环境中工作，在这个环境中，对不可预料的和正在改变的环境状态的反应必须是快和不含糊的。其他的agent有时间和知识来预测动作的未来过程的结果，以便能做出更合理的选择。

当然，其中一些agent将比其他的更复杂和聪明，因此我们也许能把焦点放在这些agent的结构上。这些agent要求有本书中讨论的所有能力：固定的反应能力；基于图标和特征的存储器结构；搜索技术；推理、计划和通信能力。我们能指望存在一个理想的、拥有所谓“人类级”智能的agent体系结构吗？可能不会。即使是人也会因先天或后天的因素而成为大量的不同类型的人。一些人用图进行思考，一些人用符号，一些人用声音；一些人是冲动的，一些人是深思熟虑的；一些人想得详细，一些人比较毛糙；一些人容易适应新的环境，一些人却不易；等等。虽然我们在相同的“湿件（wetware）”上运行得都很好，但我们的认知结构可能很不一样。

可能没有一个统一的、理想的智能agent体系结构。但是下面会讨论一些我最为熟悉的、也是典型的可供参考的体系结构。当然，除此之外，专家们还提出并使用了另外的体系结构。

25.1 三级体系结构

最初的集成智能agent系统之一是集成计算机程序和硬件的一个系统，它叫做“Shakey the Robot”[Nilsson 1984b]。Shakey已经用到了很多在本书中描述的技术——虽然在形式上它们比现代的同类更原始。硬件由一个大约有一个小冰箱大小的小车、对接触敏感的“触角”、一个电视照相机和一个光学探测器构成——所有这些都由一个计算机通过一个二路声频/视频连接控制（见图25-1）。小车能在一个由墙隔开的、用门连接的、堆有很多大盒子的房间中移动。它能在自己的环境中把盒子从一个地方推到另一个地方。程序组件包括可视场景分析程序（它能认出盒子、门和屋角）、进行计划的程序（用STRIPS,它能生成一个到达目标的动作序列）和把计划转化为环境中的中间级和低级动作的程序。Shakey结构如图25-2所示。

可将Shakey的设计视为所谓的三级结构的一个早期例子。每一级对应从传感信号到电机命令的不同路径。在图25-2中，用如下的方法强调了这些路径：低级用灰箭头，中级用灰色点划线，高级用黑色点划线表示。低级动作（LLA）使用一个短而快的、从传感信号到操纵装置的路径，该路径处理重要的“反应”——如接触传感器检测到接近一个对象时stop。到达指定

的轴角目标点的电机伺服控制等等也由低级机制处理。

中级把LLA组合成更复杂的行为——它的实现取决于执行任务时的状态（检测到的和模型化的）。作为一个例子，一个中间级动作（ILA）是gothrudoor例程，它使Shakey通过一个已命名的门。它由roll-forward和turns组成，它们由接触探测器和一个内部模型中的信息指导，直到它断定Shakey已完全通过了那个门。Shakey的ILA用一种与第2章提到的T-R程序非常相似的方式编码。

高级路径涉及到一个STRIPS计划者。将一个用谓词演算合式公式表达的目标赋予执行系统，它监督STRIPS构造计划。完成的计划被表达为一个ILA序列，序列和它们的前提条件以及结果通过一个三角表表示。像执行一个T-R程序一样执行该三角表。三角表形式的计划也被存在一个计划库（经推广后，像第22章讨论的一样）中，以为将来所使用。

最近，三级结构已经用在各种机器人系统中。一些原始动作的伺服控制常被用在最低级，很多不同的AI子系统被用在中级和高级，包括黑板系统、动态贝叶斯信念网络、模糊逻辑到计划空间计划者。作为一个典型的例子，可以参考[Connell 1992]。

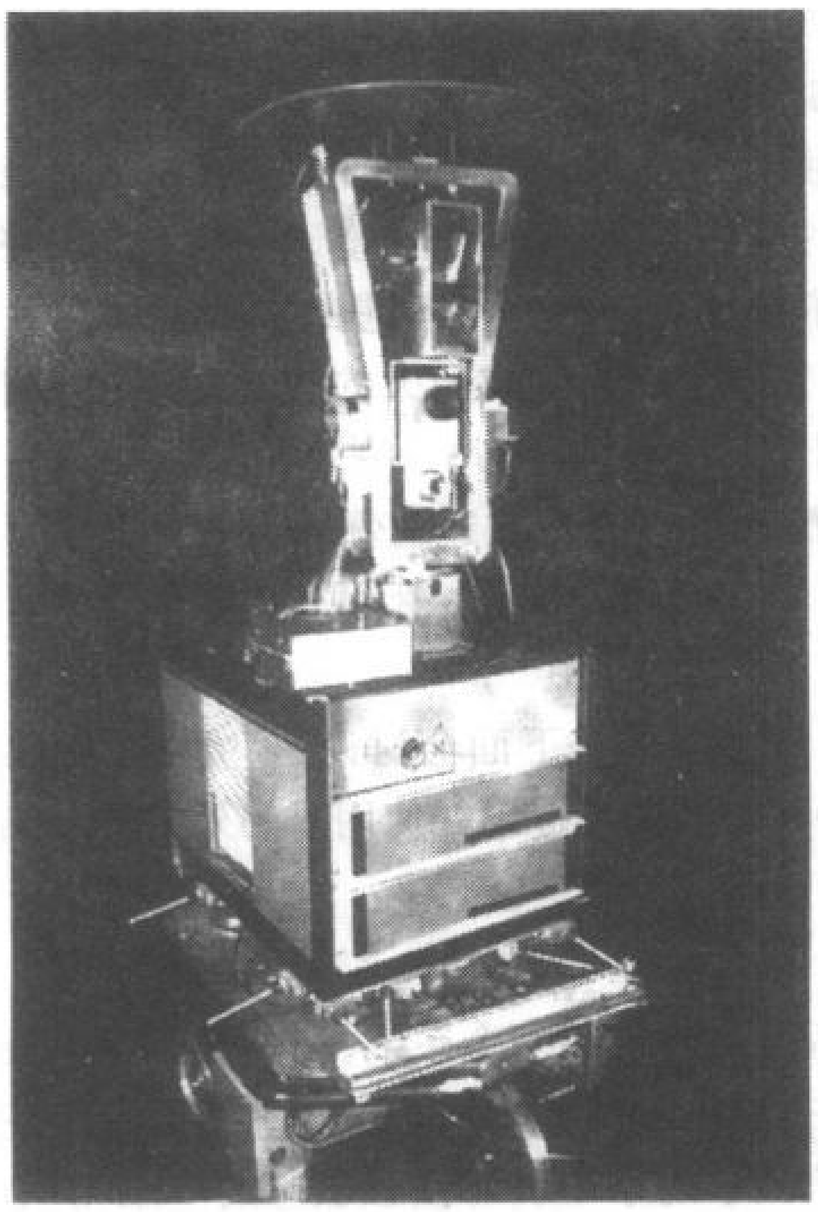


图25-1　Shakey 机器人

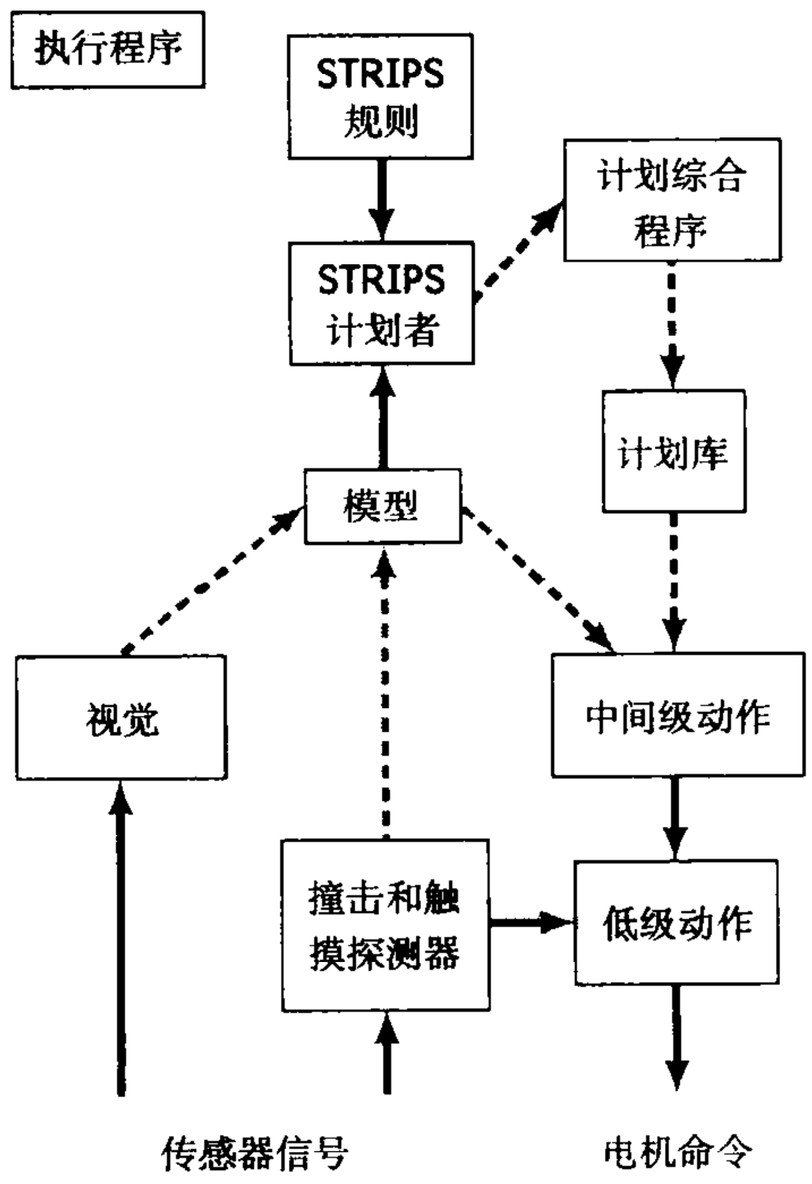


图25-2　Shakey体系结构

25.2　目标仲裁

一个三级体系结构是解决把快速反应能力和计划动作序列以完成复杂任务相结合这类问题的一种方式。在大多数agent体系结构中，反应动作要优先于计划制定。对反应动作的要求仅仅是一个agent要同时参与几个（有时是竞争）目标这类一般情况的一个实例。agent常常有几个需要达到的目标。其中一些由它们的用户（有不同的优先级）给出，有些是内在的（像安全需要，补充燃料、自维护等）。一组目标中的每一个，按照当时的目标优先级和从当前状态达

到目标的相对代价，都有一定的“紧迫性”[⊖]。有些目标用低级例程最好处理，有些能用保存的ILA到达，有些将需要做出计划。由于目标的紧迫性会随着agent动作和在新的，有时是无法预料的环境而变化，因此agent结构必须能在竞争的ILA和计划中做出仲裁。

[Benson & Nilsson 1995]提出了一个agent结构，它把反应、学习和计划能力集成在一起。和仲裁有关的结构部分如图25-3所示。目标及优先级通过执行器传给系统，它们保持活动直到被用户解除。系统有若干个ILA存储在计划库中——就像T-R程序一样被存储且与特定的目标相符合。如果任何活动目标能通过存储在计划库中的T-R程序实现，这些T-R程序就成为活动的ILA。没有现存的T-R程序的目标将导致计划者去试图产生新的T-R程序——通过扩展计划库中的合适T-R程序或者开始一个新的程序。任何由计划者产生来到达一个活动目标的新T-R程序成为活动的ILA。内建目标总是有提前建立的T-R程序，它们一直是活动的ILA。

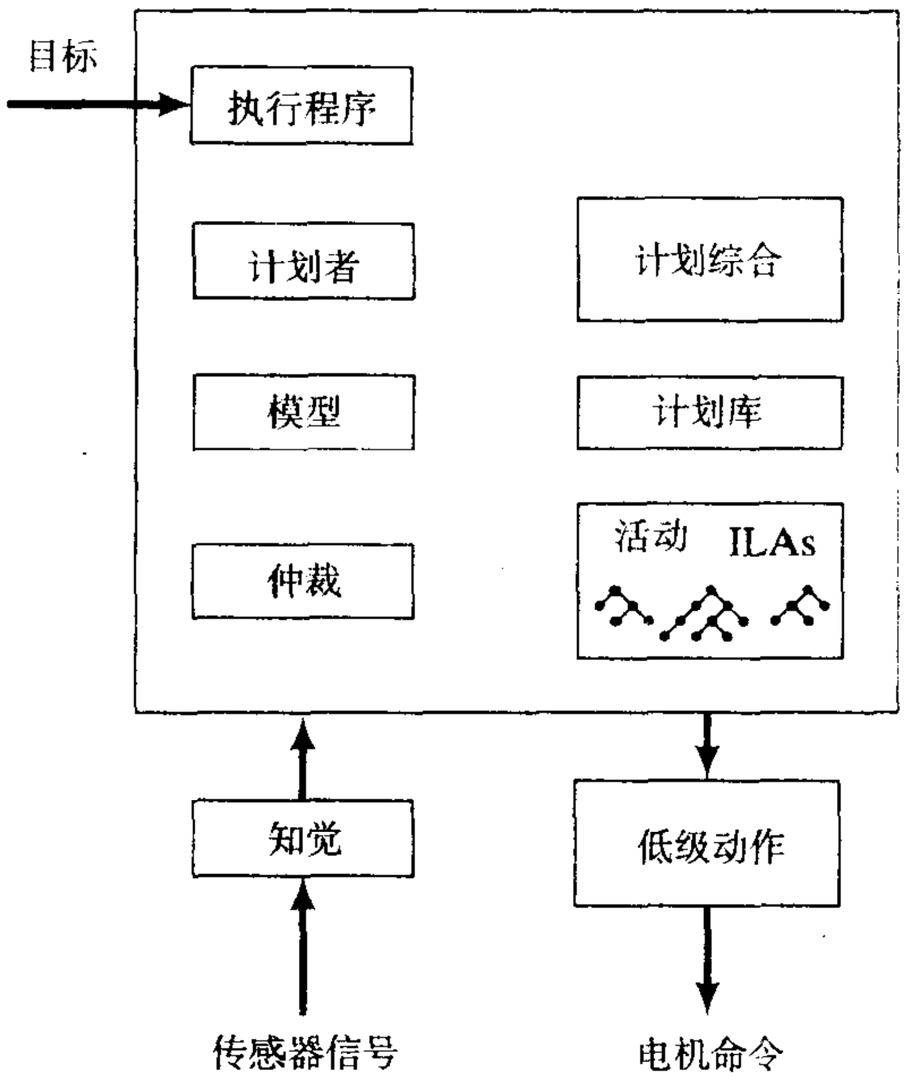


图25-3 组合计划和反应

实际由agent执行的动作是由一个激活的ILA调用的动作。仲裁程序的任务就是在每个时刻选择由哪一个T-R程序来“管理”当前的agent。这个选择用一个简单的代价—收益计算来实现。该计算考虑了目标的优先级和到达目标的估计费用（更详细的内容参见[Benson & Nilsson 1995]）。维护安全性和避免危险总是有高的优先级，因此，当任何紧急情况出现时总是先选择它们。仲裁模块和计划者模块并发工作以便在计划的同时，agent能够动作（合适的话）。如果较低优先级的目标可以用较低的估计费用实现（它们也能用昂贵的更高优先级实现），那么就先执行较低优先级的目标[⊜]。

25.3 三层塔式结构

在三级体系结构中，高级别比低级别使用了更抽象的知觉谓词和更复杂的动作。但是反应动

⊖ 这些优先级和代价能用和第10章讨论的相似的奖赏调度来满足。

⊜ 这个特殊的agent 结构也满足动作学习模型（计划者能用它）。关于学习元件的详细内容，参见[Benson 1997]。

作通常是由基本的传感信号引发的，故高级动作协调将要求更详细的知觉处理。[Albus1991]提出了层次式或“塔式”知觉和动作处理。知觉处理塔从基本的传感信号开始，一层一层向前推进，直到对检测到的东西的表示更精确和更抽象。动作塔由越来越复杂的基本动作序列组成。知觉塔和动作塔之间的联接可以存在于层次结构的所有级别。最低级的连接对应简单的反应，而更高级的连接则对应由知觉谓词指定的复杂动作。

为了显式说明agent的内部表示，建议加一个如图25-4所示的第三个或者叫“模型”塔。给定agent环境表示的感知规则被存在一个层次模型中——每一级包含由任何计划和动作（那一级的相应处理）使用的信息。例如，在中间级，可能有一些模型（像潜在的函数或拓朴状态空间）适合于路线计划。像那些基于逻辑或语义网络的表示一样，在更高级别，逻辑推理、计划和通信将要求以声明方式表示。

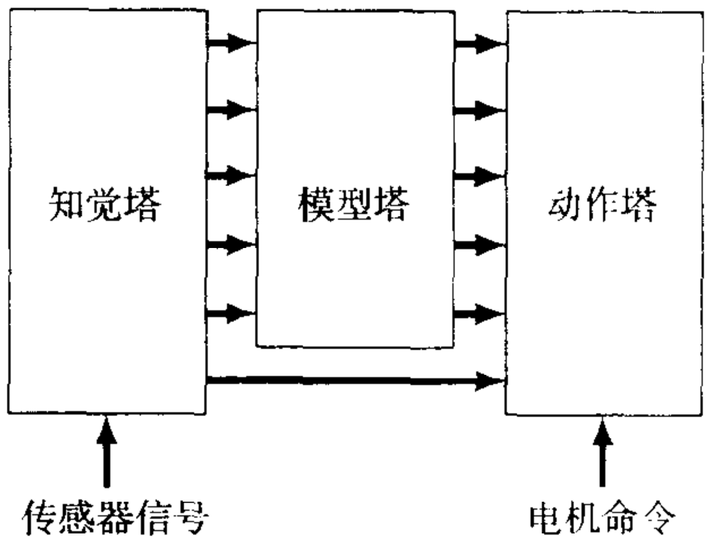


图25-4　三层塔式体系结构

虽然三层塔式结构确实不错，但它仍仅仅是一个粗糙的、空洞的理论，还需要很多东西来完善它。相信将来的发展将要求更多的实验工作以把AI技术（像这本书中所讨论的）与具有感知和动作的agent集成起来。

25.4　自举

“人类级（*human-level*）”的人工智能毫无疑问将比当代的机器人和agent需要更精练的知觉和动作指令系统。当前最缺乏的是那些必须移入模型塔更高层的常识知识。大部分AI研究者认为甚至在agent执行人类很容易就能找到的路线任务前，这些知识也必须被编码进我们的agent。CYC工程是由一组研究人员从事的研究计划，他们把认为需要的知识进行编码。从AI的早期开始，一些研究者就认为这些知识能通过自动学习程序、自组织系统、模拟演化或者一些捷径等“容易方式”得到。CYC的研究者甚至希望在CYC达到一定的关键阶段后，系统将能够通过自己阅读文本、和人交谈和受教育等方式来学习更多的知识。人类从以前得到的知识中通过一个“自举”过程学会大量他们使用的知识——开始在婴儿时，有天生的脾性；后来通过各种Piagetian阶段（瑞士心理学家皮亚杰关于儿童思维发育理论）从以前获得的技巧和概念中学习，当成年时，通过锻炼、阅读和交流学习。对我来讲，AI agent需要用来展示人类级智能的知识是如此之多，以致要求一个相似的自举过程。这个过程可能将涉及到一些技术，它们和本书描述的自动学习过程比较相似。我们必须找到一种方式，以使一个agent在它的设计者设计了适当的较低层塔式结构后，它自己能把其他的层加到它的三层塔结构上。

25.5　补充读物和讨论

[Whitehead, Karlsson, & Tenenberg 1993]开发了一个系统，它能知道如何达到几个目标中的每一个，并且当几个目标都是活动时能对它们进行仲裁。

在谈到agent结构中不同积木图的多样性时，Stan Rosenschein过去常常说“盒子，盒子，我们都已得到了盒子（*Boxes, Boxes, we've all got boxes*。）”。除了本章提到的几个结构方案外，还引用了下面几个（要知道有很多我没有提到）[Laird ,et al. 1991, Hayes-Roth 1995, Gat 1992,

Wilkins,et al 1995, Bates, Loyall & Reilly 1992, Firby & Kahn 1995]。

在所有这些问题中，一个关键的问题是在运行时是精炼一个计划还是执行当前的计划（可能有一个计划，当时间不是关键因素时它可以被编译，例如在设计时）。像本书前面提到的，元级结构（例如[Russell & Wefald 1991, 第2章]）能用来做这种选择。在很多情况下，计算的时空权衡特性是大多数agent 动作应该有反应的、具有扩展agent已知边缘问题的计划和学习能力。

在本书中，我尽力覆盖大多数当前已知的AI技术，我认为它们对达到人类级AI是需要的。它是一个大的集合，但在接近我们的目标之前它必须变得更大。我认为随着发展，我们将继续从心理学、神经心理学、控制论、信号处理、经济学以及计算机科学等方面来汲取思想。对于有志于从事人工智能事业的读者，我的建议是当面对现存AI系统的低下能力与动物和人类智能的异常复杂能力的巨大差距时，我们应该折衷地、富有想像力地、谦逊地对待。但是我们的谦逊没有必要也不应该削弱我们的目标和胆识。

习题

25.1 Isaac Asimov 关于机器人的三个定律是：

1) 一个机器人不能伤害一个人，也不允许一个人来伤害机器人。

2) 一个机器人必须服从人类给出的命令，除非这个命令与第一个定律相矛盾。

3) 一个机器人必须保护它自己的存在，只要这个保护不和第一、第二两个定律矛盾。

当用这些定律构造成自治agent（物理的agent和软件agent）时，描述一下你能想到的各种困难。需要什么AI能力？

25.2 你认为本书中谈到AI技术、感知设备和可用的理论足以支持构造我们真正称谓的“智能”机器吗？为什么？如果你认为当前的AI技术不充分，至少回答下面的一个问题，给出你的答案证据：

1) 需要实质性增加新的理论和思想吗？如果需要，在什么领域？

2) 我们仅仅需要更大和更快（也许还要更多的并行）的计算机吗？

3) AI研究在走不合适的路吗？需要全新的方法吗？如果这样，AI研究应该把重点放在哪里？

4) AI的长期研究目标在原理上是不可能的吗？

25.3 既然你已经研究了AI中的很多重要思想，回到第1章做习题1.5，评论一下强AI与弱AI（没有“正确”“答案”，但有一些有说服力的答案）。

参考文献

[Abramson & Yung 1989] Abramson, B., and Yung, M., "Divide and Conquer under Global Constraints: A Solution to the N-Queens Problem," *Journal of Parallel and Distributed Computing*, 6:649–662, 1989.

[Agre & Chapman 1987] Agre, P., and Chapman, D., "Pengi: An Implementation of a Theory of Activity," *Proceedings of the Sixth National Conference on Artificial Intelligence (AAAI-87)*, pp. 268–272, Menlo Park, CA: AAAI Press, 1987.

[Agre & Chapman 1990] Agre, P., and Chapman, D., "What Are Plans For?" *Robotics and Autonomous Systems*, 6:17–34, 1990. Also in [Maes 1990a].

[Albus 1991] Albus, J. S., "Outline for a Theory of Intelligence," *IEEE Systems, Man, and Cybernetics*, 21(3):473–509, May/June 1991.

[Allen 1983] Allen, J., "Maintaining Knowledge about Temporal Intervals," *Communications of the ACM*, 26(11):832–843, 1983. (Reprinted in Weld, D., and de Kleer, J. (eds.), *Readings in Qualitative Reasoning about Physical Systems*, San Francisco: Morgan Kaufmann, 1990.)

[Allen 1984] Allen, J., "Towards a General Theory of Action and Time," *Artificial Intelligence*, 23:123–154, 1984.

[Allen 1991a] Allen, J., "Time and Time Again: The Many Ways to Represent Time," *International Journal of Intelligent Systems*, 6:341–355, 1991.

[Allen 1991b] Allen, J., "Temporal Reasoning and Planning," in Allen, J., Kautz, H., Pelavin, R., and Tenenberg, J. (eds.), *Reasoning About Plans*, Ch. 1, San Francisco: Morgan Kaufmann, 1991.

[Allen 1995] Allen, J., *Natural Language Understanding*, Menlo Park, CA: Benjamin/Cummings, 1995.

[Allen, et al. 1990] Allen, J., Hendler, J., and Tate, A. (eds.), *Readings in Planning*, San Francisco: Morgan Kaufmann, 1990.

[Almeida 1987] Almeida, L. B., "A Learning Rule for Asynchronous Perceptrons with Feedback in a Combinatorial Environment," in M. Caudill and C. Butler (eds.), *IEEE First International Conference on Neural Networks*, San Diego, 1987, vol. II, pp. 609–618, New York: IEEE, 1987.

[Aloimonos 1993] Aloimonos, Y. (ed.), *Active Perception*, Hillsdale, NJ: Lawerence Erlbaum Associates, 1993.

[Anderson & Donath 1990] Anderson, T., and Donath, M., "Animal Behavior as a Paradigm for Developing Robot Autonomy," *Robotics and Autonomous Systems*, 6:145–168, 1990. Also in [Maes 1990a].

[Andre 1995] Andre, D., "The Automatic Programming of Agents That Learn Mental Models and Create Simple Plans of Action," in *Proceedings of the Fourteenth International Joint Conference on Artificial Intelligence (IJCAI-95)*, pp. 741–747, San Francisco: Morgan

Kaufmann, 1995.

[Appelt 1985] Appelt, D., "Planning English Referring Expressions," *Artificial Intelligence*, 26(1):1–33, 1985. (Also in Grosz, B., Jones, K., and Webber, B. (eds.), *Readings in Natural Language Processing*, pp. 501–517, San Francisco: Morgan Kaufmann, 1986.)

[Bacchus & Yang 1992] Bacchus, F., and Yang, Q., "The Expected Value of Hierarchical Problem-Solving," in *Proceedings of the Tenth National Conference on Artificial Intelligence (AAAI-92)*, pp. 369–374, Menlo Park, CA: AAAI Press, 1992.

[Baker 1991] Baker, A., "Nonmonotonic Reasoning in the Framework of Situation Calculus," *Artificial Intelligence*, 49:5–23, 1991.

[Ballard 1991] Ballard, D. H., "Animate Vision," *Artificial Intelligence*, 48(1):57–86, 1991.

[Ballard & Brown 1982] Ballard, D. H., and Brown, C. M., *Computer Vision*, Englewood Cliffs, NJ: Prentice Hall, 1982.

[Barr & Feigenbaum 1981] Barr, A., and Feigenbaum, E. (eds.), *The Handbook of Artificial Intelligence, Volume 1*, Reading, MA: Addison-Wesley, 1981. (See the following two entries and [Cohen & Feigenbaum 1982] for the other three volumes.)

[Barr & Feigenbaum 1982] Barr, A., and Feigenbaum, E. (eds.), *The Handbook of Artificial Intelligence, Volume 2*, Reading, MA: Addison-Wesley, 1982.

[Barr, Cohen, & Feigenbaum 1989] Barr, A., Cohen, P. R., and Feigenbaum, E. (eds.), *The Handbook of Artificial Intelligence, Volume 4*, Reading, MA: Addison-Wesley, 1989.

[Barto, Bradtke, & Singh 1995] Barto, A., Bradtke, S., and Singh, S., "Learning to Act Using Real-Time Dynamic Programming," *Artificial Intelligence*, 72(1,2):81–138, January 1995.

[Barwise & Etchemendy 1993] Barwise, J., and Etchemendy, J., *The Language of First-Order Logic: Including the Macintosh Program Tarski's World 4.0*, Center for the Study of Language and Information (CSLI), Stanford, California, third revised and expanded edition, 1993.

[Bates 1994] Bates, J., "The Role of Emotion in Believable Agents," *Communications of the ACM*, 37(7):122–125, 1994.

[Bates, Loyall, & Reilly 1992] Bates, J., Loyall, A. B., and Reilly, W. S., "Integrating Reactivity, Goals, and Emotion in a Broad Agent," Technical Report CMU-CS-92-142, School of Computer Science, Carnegie Mellon University, Pittsburgh, PA, May 1992. (Also appeared in the *Proceedings of the Fourteenth Annual Conference of the Cognitive Science Society*, Bloomington, Indiana, July 1992.)

[Bayes 1763] Bayes, T., "An Essay Towards Solving a Problem in the Doctrine of Chances," *Phil. Trans.*, 3:370–418, 1763. Reproduced in Deming, W. (ed.), *Two Papers by Bayes*, New York: Hafner, 1963.

[Beer 1995] Beer, R., "A Dynamical Systems Perspective on Agent-Environment Interaction," *Artificial Intelligence*, 72(1–2):173–215, 1995.

[Beer, Chiel, & Sterling 1990] Beer, R., Chiel, H., and Sterling, L., "A Biological Perspective on Autonomous Agent Design," *Robotics and Autonomous Systems*, 6:169–186, 1990. Also in [Maes 1990a].

[Benson 1997] Benson, S., *Learning Action Models for Reactive Autonomous Agents*, Stanford University Computer Science Department Ph.D. dissertation, Report No. STAN-CS-TR-97-1589, Stanford, CA 94305, 1997.

[Benson & Nilsson 1995] Benson, S., and Nilsson, N., "Reacting, Planning and Learning in an Autonomous Agent," in Furukawa, K., Michie, D., and Muggleton, S. (eds.), *Machine Intelligence 14*, Oxford: The Clarendon Press, 1995.

[Berliner 1979] Berliner, H., "The B* Tree-Search Algorithm: A Best-First Proof Procedure," *Artificial Intelligence*, 12(1):23–40, 1979.

[Bhanu & Lee 1994] Bhanu, B., and Lee, S., *Genetic Learning for Adaptive Image Segmentation*, Boston: Kluwer Academic Publishers, 1994.

[Binford 1982] Binford, T. O., "Survey of Model-Based Image Analysis Systems," *The International Journal of Robotics Research*, 1(1):18–64, 1982.

[Binford 1987] Binford, T. O., "Generalized Cylinder Representation," in Shapiro, S. C. (ed.), *Encyclopedia of Artificial Intelligence*, pp. 321–323, New York: John Wiley & Sons, 1987. (This article is based on an unpublished 1971 paper by Binford entitled "Visual Perception by Computer.")

[Bledsoe 1977] Bledsoe, W., "Non-Resolution Theorem Proving," *Artificial Intelligence*, 9(1):1–35, 1977.

[Blum & Furst 1995] Blum, A., and Furst, M., "Fast Planning through Planning Graph Analysis," in *Proceedings of the Fourteenth International Joint Conference on Artificial Intelligence (IJCAI-95)*, pp. 1636–1642, San Francisco: Morgan Kaufmann, 1995.

[Blumberg 1996] Blumberg, B., *Old Tricks, New Dogs: Ethology and Interactive Creatures*, Ph.D. Dissertation, MIT Media Lab, Massachusetts Institute of Technology, 1996.

[Bobrow 1968] Bobrow, D., "Natural Language Input for a Computer Problem Solving System," in Minsky, M. (ed.), *Semantic Information Processing*, pp. 133–215, Cambridge, MA: MIT Press, 1968.

[Bobrow, Mittal, & Stefik 1986] Bobrow, D., Mittal, S., and Stefik, M., "Expert Systems: Perils and Promise," *Communications of the ACM*, 29(9):880–894, 1986.

[Bond & Gasser 1988] Bond, A., and Gasser, L. (eds.), *Readings in Distributed Artificial Intelligence*, San Francisco: Morgan Kaufmann, 1988.

[Boole 1854] Boole, G., *An Investigation of the Laws of Thought on Which Are Founded the Mathematical Theories of Logic and Probabilities*, New York: Dover Publications, 1854.

[Börger 1989] Börger, E., *Computability, Complexity, Logic*, Amsterdam: North-Holland, 1989.

[Borgida, et al. 1989] Borgida, A., Brachman, R., McGuinness, D., and Resnick, A., "CLASSIC: A Structural Data Model for Objects," *SIGMOD Record*, 18(2):58–67, 1989.

[Boyer & Moore 1979] Boyer, R., and Moore, J., *A Computational Logic*, New York: Academic Press, 1979.

[Brachman & Levesque 1985] Brachman, R., and Levesque, H. (eds.), *Readings in Knowledge Representation*, San Francisco: Morgan Kaufmann, 1985.

[Brachman, Gilbert, & Levesque 1985] Brachman, R., Gilbert, V., and Levesque, H., "An Essential Hybrid Reasoning System: Knowledge and Symbol Level Accounts of KRYPTON,"

in *Proceedings of the Ninth International Joint Conference on Artificial Intelligence (IJCAI-85)*, pp. 532–539, San Francisco: Morgan Kaufmann, 1985.

[Brain, et al. 1962] Brain, A. E., et al., "Graphical Data Processing Research Study and Experimental Investigation," Report No. 8 (pp. 9–13) and No. 9 (pp. 3–10), Contract DA 36–039 SC–78343, SRI International, Menlo Park, CA, June 1962 and September 1962.

[Braitenberg 1984] Braitenberg, V., *Vehicles: Experiments in Synthetic Psychology*, Cambridge, MA: MIT Press, 1984.

[Bratko 1990] Bratko, I., *PROLOG Programming for Artificial Intelligence*, second edition, Reading, MA: Addison-Wesley, 1990.

[Bratko & Michie 1980] Bratko, I., and Michie, D., "An Advice Program for a Complex Chess Programming Task," *Computer Journal*, 23(4):353–359, 1980.

[Brave 1996] Brave, S., "The Evolution of Memory and Mental Models Using Genetic Programming," in Koza, J., et al. (eds.), *Genetic Programming 1996: Proceedings of the First Annual Conference*, pp. 261–266, Stanford University, July 28–31, 1996, Cambridge, MA: MIT Press, 1996.

[Breiman, et al. 1984] Breiman, L., Friedman, J., Olshen, R., and Stone, C., *Classification and Regression Trees*, Belmont, CA: Wadsworth, 1984.

[Brewka, Dix, & Konolige 1997] Brewka, G., Dix, J., and Konolige, K., *Nonmonotonic Reasoning: An Overview*, CSLI Lecture Notes, No. 73, Center for the Study of Language and Information, Stanford, CA: Stanford University, 1997.

[Brooks 1981] Brooks, R., "Symbolic Reasoning among 3-D Models and 2-D Images," *Artificial Intelligence*, 17:285–348, 1981.

[Brooks 1986] Brooks, R. A., "A Robust Layered Control System for a Mobile Robot," *IEEE Journal of Robotics and Automation*, RA-2(1):14–23, March 1986.

[Brooks 1990] Brooks, R., "Elephants Don't Play Chess," *Robotics and Autonomous Systems*, 6:3–15, 1990. Also in [Maes 1990].

[Brooks 1991a] Brooks, R. A., "Intelligence without Representation," *Artificial Intelligence*, 47(1/3):139–159, January 1991.

[Brooks 1991b] Brooks, R. A., "Intelligence Without Reason," in *Proceedings of the Twelfth International Joint Conference on Artificial Intelligence (IJCAI-91)*, pp. 569–595, San Francisco: Morgan Kaufmann, 1991.

[Brooks & Mataric 1993] Brooks, R., and Mataric, M., "Real Robots, Real Learning Problems," in Connell, J., and Mahadevan, S. (eds.), *Robot Learning*, Ch. 8, Boston: Kluwer Academic Publishers, 1993.

[Brownston, et al. 1985] Brownston, L., Farrell, R., Kant, E., and Martin, N., *Programming Expert Systems in OPS5*, Reading, MA: Addison-Wesley, 1985.

[Bryson & Ho 1969] Bryson, A., and Ho, Y.-C., *Applied Optimal Control*, New York: Blaisdell, 1969.

[Buchanan & Shortliffe 1984] Buchanan, B., and Shortliffe, E. (eds.), *Rule-Based Expert Systems: The MYCIN Experiments of the Stanford Heuristic Programming Project*, Reading, MA: Addison-Wesley, 1984.

[Bylander 1993] Bylander, T., "An Average Case Analysis of Planning," in *Proceedings of the Eleventh*

National Conference on Artificial Intelligence (AAAI-93), pp. 480–485, Menlo Park, CA: AAAI Press, 1993.

[Bylander 1994] Bylander, T., "The Computational Complexity of Propositional STRIPS Planning," *Artificial Intelligence*, 69(1/2):165–204, 1994.

[Campbell, et al. 1982] Campbell, A. N., Hollister, V., Duda, R., and Hart, P., "Recognition of a Hidden Mineral Deposit by an Artificial Intelligence Program," *Science*, 217(4563):927–929, 1982.

[Canny 1986] Canny, J., "A Computational Approach to Edge Detection," *IEEE Trans. on Pattern Analysis and Machine Intelligence*, PAMI-8(6):679–698, November 1986.

[Carbonell, et al. 1992] Carbonell, J., Blythe, J., Etzioni, O., Gil, Y., Kahn, D., Knoblock, C., Minton, S., Perez, A., Reilly, S., Veloso, M., and Wang, X., "PRODIGY 4.0: The Manual and Tutorial," Carnegie-Mellon University Computer Science Tech. Report CMU-CS-92-1560, Pittsburg, PA, 1992.

[Cassandra, Kaelbling, & Littman 1994] Cassandra, A., Kaelbling, L., and Littman, M., "Acting Optimally in Partially Observable Stochastic Domains," in *Proceedings of the Twelfth National Conference on Artificial Intelligence (AAAI-94)*, pp. 1023–1028, Menlo Park, CA: AAAI Press, 1994.

[Chakrabarti, Ghose, & DeSarkar 1986] Chakrabarti, P., Ghose, S., and DeSarkar, S., "Heuristic Search through Islands," *Artificial Intelligence*, 29(3):339–347, 1986.

[Chang & Lee 1973] Chang, C.-L., and Lee, R., *Symbolic Logic and Mechanical Theorem Proving*, Boston: Academic Press, 1973.

[Chapman 1987] Chapman, D., "Planning for Conjunctive Goals," *Artificial Intelligence*, 32(3):333–377, 1987.

[Chapman 1989] Chapman, D., "Penguins Can Make Cake," *AI Magazine*, 10(4):45–50, 1989.

[Charniak 1993] Charniak, E., *Statistical Language Learning*, Cambridge, MA: MIT Press, 1993.

[Chauvin & Rumelhart 1995] Chauvin, Y., and Rumelhart, D., *Backpropagation: Theory, Architectures, and Applications*, Hillsdale, NJ: Lawrence Erlbaum, 1995.

[Chen 1990] Chen, S., (ed.), *Advances in Spatial Reasoning*, Norwood, NJ: Ablex Publishing, 1990.

[Chomsky 1965] Chomsky, N., *Aspects of the Theory of Syntax*, Cambridge, MA: MIT Press, 1965.

[Christensen 1990] Christensen, J., "A Hierarchical Planner That Generates Its Own Hierarchies," in *Proceedings of the Eighth National Conference on Artificial Intelligence (AAAI-90)*, pp. 1004–1009, Menlo Park, CA: AAAI Press, 1990.

[Churchland, Ramachandran, & Sejnowski 1994] Churchland, P. S., Ramachandran, V. S., and Sejnowski, T. J., "A Critique of Pure Vision," in Koch, C., and Davis, J. (eds.), *Large-Scale Neuronal Theories of the Brain*, pp. 23–60, Cambridge, MA: MIT Press, 1994.

[Clocksin & Mellish 1987] Clocksin, W., and Mellish, C., *Programming in PROLOG* (third edition), New York: Springer-Verlag, 1987.

[Clowes 1971] Clowes, M., "On Seeing Things," *Artificial Intelligence*, 2:79–116, 1971.

[Cohen & Feigenbaum 1982] Cohen, P. R., and Feigenbaum, E. (eds.), *The Handbook of Artificial Intelligence, Volume 3*, Reading, MA: Addison-Wesley, 1982.

[Cohen & Levesque 1990] Cohen, P., and Levesque, H., "Intention Is Choice with Commitment," *Artificial Intelligence*, 42(2-3):213-361, 1990.

[Cohen & Perrault 1979] Cohen, P., and Perrault, C. R., "Elements of a Plan-Based Theory of Speech Acts," *Cognitive Science*, 3:177-212, 1979.

[Colmerauer 1973] Colmerauer, A., et al., "Un Système de Communication Homme-Machine en Français," Research Report, Université Aix-Marseille II, Groupe d'Intelligence Artificielle, France, 1973.

[Colmerauer 1978] Colmerauer, A., "Metamorphosis Grammars," in Bolc, L. (ed.), *Natural Language Communication with Computers*," Berlin: Springer-Verlag. (This article is an English translation of a 1975 technical report written in French.)

[Connell 1990] Connell, J. H., "A Colony Architecture Applied to Robot Navigation," *Technical Report 1151*, MIT AI Lab, MIT, Cambridge, MA, June 1990.

[Connell 1992] Connell, J. H., "SSS: A Hybrid Architecture Applied to Robot Navigation," in *Proc. 1992 IEEE International Conf. on Robotics and Automation*, pp. 2719-2724, 1992.

[Connell & Mahadevan 1993a] Connell, J., and Mahadevan, S., "Rapid Task Learning for Real Robots," in Connell, J., and Mahadevan, S. (eds.), *Robot Learning*, Ch. 5, Boston: Kluwer Academic Publishers, 1993.

[Connell & Mahadevan 1993b] Connell, J., and Mahadevan, S. (eds.), *Robot Learning*, Boston: Kluwer Academic Publishers, 1993.

[Cook 1971] Cook, S., "The Complexity of Theorem-Proving Procedures," in *Proc. of the 3rd Annual ACM Symposium on Theory of Computing*, pp. 151-158, New York: Association for Computing Machinery, 1971.

[Cook 1988] Cook, S., "Short Propositional Formulas Represent Nondeterministic Computations," *Information Processing Letters*, 26(5):269-270, 1988.

[Cooper 1990] Cooper, G., "Computational Complexity of Probabilistic Inference Using Bayesian Belief Networks (Research Note)," *Artificial Intelligence*, 42(2/3):393-405, 1990.

[Cooper & Herskovitz 1992] Cooper, G., and Herskovitz, E., "A Bayesian Method for the Induction of Probabilistic Networks from Data," *Machine Learning*, 9:309-347, 1992.

[Cormen, Leiserson, & Rivest 1990] Cormen, T., Leiserson, C., and Rivest, R., *Introduction to Algorithms*, Cambridge, MA, and New York: MIT Press and McGraw-Hill, 1990.

[Cover & Thomas 1991] Cover, T., and Thomas, A., *Elements of Information Theory*, New York: John Wiley & Sons, 1991.

[Currie & Tate 1991] Currie, K., and Tate, A., "O-PLAN: The Open Planning Architecture," *Artificial Intelligence*, 52(1):49-86, 1991.

[Davis 1990] Davis, E., *Representations of Commonsense Knowledge*, San Francisco: Morgan Kaufmann, 1990.

[Davis & Putnam 1960] Davis, M., and Putnam, H., "A Computing Procedure for Quantification Theory," *Journ. Assoc. of Comp. Mach.*, 7(3):201-215, 1960.

[Davis 1980] Davis, R., "Meta-Rules: Reasoning about Control," *Artificial Intelligence*, 15(3):179-222, 1980.

[de Kleer 1986a] de Kleer, J., "An Assumption-Based TMS," *Artificial Intelligence*, 28(2):127-162, 1986.

[de Kleer 1986b] de Kleer, J., "Extending the ATMS," *Artificial Intelligence*, 28(2):163–196, 1986.

[de Kleer 1986c] de Kleer, J., "Problem Solving with the ATMS," *Artificial Intelligence*, 28(2):197–224, 1986.

[Dean & Boddy 1988] Dean, T., and Boddy, M., "An Analysis of Time-Dependent Planning," in *Proceedings of the Seventh National Conference on Artificial Intelligence (AAAI-88)*, pp. 49–54, Menlo Park, CA: AAAI Press, 1988.

[Dean & Wellman 1991] Dean, T., and Wellman, M., *Planning and Control*, San Francisco: Morgan Kaufmann, 1991.

[Dean, Basye, & Kaelbling 1993] Dean, T., Basye, K., and Kaelbling, L., "Uncertainty in Graph-Based Map Learning," in Connell, J., and Mahadevan, S. (eds.), *Robot Learning*, Ch. 7, Boston: Kluwer Academic Publishers, 1993.

[Dechter 1996] Dechter, R., "Bucket Elimination: A Unifying Framework for Probabilistic Inference," in *Proceedings of the Twelfth Conference on Uncertainty in AI*, pp. 211–219, San Francisco: Morgan Kaufmann, 1996.

[Dempster 1968] Dempster, A. P., "A Generalization of Bayesian Inference," *Journal of the Royal Statistical Society, Series B*, 30:205–247, 1968.

[Dempster, Laird, & Rubin 1977] Dempster, A., Laird, N., and Rubin, D., "Maximum Likelihood from Incomplete Data via the EM Algorithm," *Journal of the Royal Statistical Society, Series B*, 39:1–38, 1977.

[Dennett 1971] Dennett, D., "Intentional Systems," *The Journal of Philosophy*, 68(4):87–106, 1971.

[Dennett 1995] Dennett, D., *Darwin's Dangerous Idea*, New York: Simon & Schuster, 1995.

[Deutsch 1960] Deutsch, J. A., *The Structural Basis of Behavior*, Chicago: The University of Chicago Press, 1960.

[Dietterich 1990] Dietterich, T., "Machine Learning," in Traub, J., et al. (eds.), *Annual Review of Computer Science*, 4:255–306, 1990.

[Dietterich & Bakiri 1991] Dietterich, T., and Bakiri, G., "Error-Correcting Output Codes: A General Method for Improving Multiclass Inductive Learning Programs," in *Proceedings of the Ninth National Conference on Artificial Intelligence (AAAI-91)*, pp. 572–577, Menlo Park, CA: AAAI Press, 1991.

[Dietterich & Bakiri 1995] Dietterich, T., and Bakiri, G., "Solving Multiclass Learning Problems via Error-Correcting Output Codes," *Journal of Artificial Intelligence Research*, 2:263–286, 1995.

[Dijkstra 1959] Dijkstra, E., "A Note on Two Problems in Connection with Graphs," *Numerische Mathematik*, (1):269–271, 1959.

[Doran & Michie 1966] Doran, J., and Michie, D., "Experiments with the Graph Traverser Program," *Proc. Royal Society of London*, vol. 294 (series A), pp. 235–259, 1966.

[Dowling & Gallier 1984] Dowling, W., and Gallier, J., "Linear-Time Algorithms for Testing the Satisfiability of Propositional Horn Formulas," *Journal of Logic Programming*, (3):267–284, 1984.

[Doyle 1979] Doyle, J., "A Truth Maintenance System," *Artificial Intelligence*, 12(3):231–272, 1979. (Reprinted in Webber, B., and Nilsson, N. (eds.), *Readings in Artificial Intelligence*, San

Francisco: Morgan Kaufmann, 1981.)

[Dreyfus 1979] Dreyfus, H., *What Computers Can't Do: The Limits of Artificial Intelligence*, revised edition, New York: Harper and Row, 1979.

[Dreyfus 1992] Dreyfus, H., *What Computers Still Can't Do: A Critique of Artificial Reason*, Cambridge, MA: MIT Press, 1992.

[Dreyfus & Dreyfus 1986] Dreyfus, H., and Dreyfus, S., *Mind over Machine: The Power of Human Intuition and Expertise in the Era of the Computer*, (with T. Athanasiou), Oxford: Blackwell, 1986.

[Duda, Hart, & Nilsson 1976] Duda, R. O., Hart, P. E., and Nilsson, N., "Subjective Bayesian Methods for Rule-Based Inference Systems," *Proc. AFIPS Nat. Computer Conference*, vol. 47:1075–1082, 1976. (Reprinted in Webber, B., and Nilsson, N. (eds.), *Readings in Artificial Intelligence*, San Francisco: Morgan Kaufmann, 1981.)

[Duda, Hart, & Stork 1998] Duda, R. O., Hart, P. E., and Stork, D., *Pattern Classification*, second edition, New York: John Wiley & Sons, 1998.

[Duda, Gaschnig, & Hart 1979] Duda, R., Gaschnig, J., and Hart, P., "Model Design in the PROSPECTOR Consultant System for Mineral Exploration," in Michie, D. (ed.), *Expert Systems in the Microelectronic Age*, pp. 153–167, Edinburgh: Edinburgh University Press, 1979. (Reprinted in Webber, B., and Nilsson, N. (eds.), *Readings in Artificial Intelligence*, pp. 337ff, San Francisco: Morgan Kaufmann, 1981.)

[Dym & Levitt 1991] Dym, C., and Levitt, R., *Knowledge-Based Systems in Engineering*, New York: McGraw-Hill, 1991.

[Elkan 1992] Elkan, C., "Reasoning about Action in First-Order Logic," in *Proc. of the Ninth Biennial Conf. of the Canadian Society for Computational Studies of Intelligence*, pp. 221–227, San Francisco, Morgan Kaufmann, 1992.

[Elkan 1993] Elkan, C., "The Paradoxical Success of Fuzzy Logic," in *Proceedings of the Eleventh National Conference on Artificial Intelligence (AAAI-93)*, pp. 698–703, Menlo Park, CA: AAAI Press, 1993.

[Elman 1990] Elman, J., "Finding Structure in Time," *Cognitive Science*, 14:179–211, 1990.

[Emerson 1989] Emerson, E., "Temporal and Modal Logic," in van Leeuwen, J. (ed.), *Handbook of Theoretical Computer Science*, pp. 995–1072, Amsterdam: North-Holland, 1989.

[Enderton 1972] Enderton, H., *A Mathematical Introduction to Logic*, New York: Academic Press, 1972.

[Ephrati, Pollack, & Milshtein 1996] Ephrati, E., Pollack, M., and Milshtein, M., "A Cost-Directed Planner: Preliminary Report," in *Proceedings of the Thirteenth National Conference on Artificial Intelligence (AAAI-96)*, pp. 1194–1201, Menlo Park, CA: AAAI Press, 1996.

[Erman, et al. 1980] Erman, L., Hayes-Roth, F., Lesser, V., and Reddy, R., "The Hearsay-II Speech-Understanding System: Integrating Knowledge to Resolve Uncertainty," *Computing Surveys*, 12:213–253, 1980.

[Ernst, Millstein, & Weld 1997] Ernst, M., Millstein, T., and Weld, D., "Automatic SAT-Compilation of Planning Problems," in *Proceedings of the Fifteenth International Conference on Artificial Intelligence (IJCAI-97)*, pp. 1169–1176, San Francisco: Morgan Kaufmann, 1997.

[Erol, Nau, & Subrahmanian 1992] Erol, K., Nau, D., and Subrahmanian, V., "On the Complexity of Domain-Independent Planning," in *Proceedings of the Tenth National Conference on Artificial Intelligence (AAAI-92)*, pp. 381–386, Menlo Park, CA: AAAI Press, 1992.

[Erol, Hendler, & Nau 1994] Erol, K., Hendler, J., and Nau, D., "HTN Planning: Complexity and Expressivity," in *Proceedings of the Twelfth National Conference on Artificial Intelligence (AAAI-94)*, pp. 1123–1128, Menlo Park, CA: AAAI Press, 1994.

[Etzioni 1993] Etzioni, O., "Acquiring Search-Control Knowledge via Static Analysis," *Artificial Intelligence*, 62(2):255–301, 1993.

[Etzioni & Weld 1994] Etzioni, O., and Weld, D., "A Softbot-Based Interface to the Internet," *Communications of the ACM*, 37(7):72–76, July 1994.

[Evans 1968] Evans, T. G., "A Program for the Solution of a Class of Geometric-Analogy Intelligence-Test Questions," in Minsky, M. (ed.), *Semantic Information Processing*, pp. 271–353, Cambridge, MA: MIT Press, 1968.

[Fagin & Halpern 1985] Fagin, R., and Halpern, J., "Belief, Awareness, and Limited Reasoning," in *Proceedings of the Ninth International Joint Conference on Artificial Intelligence (IJCAI-85)*, pp. 491–501, San Francisco: Morgan Kaufmann, 1985.

[Fagin, et al. 1995] Fagin, R., Halpern, J., Moses, Y., and Vardi, M., *Reasoning about Knowledge*, Cambridge, MA: MIT Press, 1995.

[Faugeras 1993] Faugeras, O., *Three-Dimensional Computer Vison: A Geometric Viewpoint*, Cambridge, MA: MIT Press, 1993.

[Feigenbaum & Feldman 1963] Feigenbaum, E., and Feldman, J. (eds.), *Computers and Thought*. New York: McGraw-Hill, 1963.

[Feigenbaum, Buchanan, & Lederberg 1971] Feigenbaum, E., Buchanan, B., and Lederberg, J., "On Generality and Problem Solving: A Case Study Using the DENDRAL Program," in Meltzer, B., and Michie, D. (eds.), *Machine Intelligence 6*, pp. 165–190, Edinburgh: Edinburgh University Press, 1971.

[Feigenbaum, McCorduck, & Nii 1988] Feigenbaum, E., McCorduck, P., and Nii, H. P., *The Rise of the Expert Company: How Visionary Companies Are Using Artificial Intelligence to Achieve Higher Productivity and Profits*, New York: Times Books, 1988.

[Feller 1968] Feller, W., *An Introduction to Probability Theory and Applications*, vol. 1, New York: John Wiley & Sons, 1968.

[Fikes & Nilsson 1971] Fikes, R., and Nilsson, N., "STRIPS: A New Approach to the Application of Theorem Proving to Problem Solving," *Artificial Intelligence*, 2(3/4):189–208, 1971.

[Fikes, Hart, & Nilsson 1972] Fikes, R., Hart, P., and Nilsson, N., "Learning and Executing Generalized Robot Plans," *Artificial Intelligence*, 3(4):251–288, 1972.

[Finin, Labrou, & Mayfield 1997] Finin, T., Labrou, Y., and Mayfield, J., "KQML as an Agent Communication Language," in Bradshaw, J. (ed.), *Software Agents*, Cambridge, MA: MIT Press, 1997.

[Firby & Kahn 1995] Firby, R. J., and Kahn, R., "An Architecture for Vision and Action," in *Proceedings of the Fourteenth International Joint Conference on Artificial Intelligence (IJCAI-95)*, pp. 72–79, San Francisco: Morgan Kaufmann, 1995.

[Fischler & Firschein 1987] Fischler, M. A., and Firschein, O. (eds.), *Readings in Computer Vision: Issues, Problems, Principles, and Paradigms*, San Francisco: Morgan Kaufmann, 1987.

[Fleischmann, et al. 1995] Fleischmann, R., et al., "Whole-Genome Random Sequencing and Assembly of *Haemophilus influenzae Rd*," *Science*, 269:496–512, July 28, 1995.

[Forbes, et al. 1995] Forbes, J., Huang, T., Kanazawa, K., and Russell, S., "The BATmobile: Towards a Bayesian Automatic Taxi," in *Proceedings of the Fourteenth International Joint Conference on Artificial Intelligence (IJCAI-95)*, pp. 1878–1885, San Francisco: Morgan Kaufmann, 1995.

[Forbus & de Kleer 1993] Forbus, K., and de Kleer, J., *Building Problem Solvers*, Cambridge, MA: MIT Press, 1993.

[Forgy 1982] Forgy, C., "RETE: A Fast Algorithm for the Many Pattern/Many Object Pattern Match Problem," *Artificial Intelligence*, 19(1):17–37, 1982.

[Frege 1879] Frege, G., "Begriffsschrift, a Formula Language Modelled upon That of Arithmetic, for Pure Thought," (1879), in van Heijenoort, J. (ed.), *From Frege to Gödel: A Source Book in Mathematical Logic, 1879–1931*, pp. 1–82, Cambridge, MA: Harvard University Press, 1967.

[Friedman 1997] Friedman, N., "Learning Belief Networks in the Presence of Missing Values and Hidden Variables," *Proceedings of the Fourteenth International Conference on Machine Learning (ICML '97)*, San Francisco: Morgan Kaufmann, 1997.

[Friedman & Goldszmidt 1996a] Friedman, N., and Goldszmidt, M., "Learning Bayesian Networks with Local Structure," in *Proceedings of the Twelfth Conference on Uncertainty in Artificial Intelligence*, pp. 252–262, San Francisco: Morgan Kaufmann, 1996.

[Friedman & Goldszmidt 1996b] Friedman, N., and Goldszmidt, M., "Building Classifiers Using Bayesian Networks," *Proceedings of the Thirteenth National Conference on Artificial Intelligence (AAAI-96)*, pp. 1277–1284, Menlo Park, CA: AAAI Press, 1996.

[Fu 1994] Fu, L. M., *Neural Networks in Computer Intelligence*, New York: McGraw-Hill, 1994.

[Galton 1987] Galton, A., *Temporal Logics and Their Applications*, London: Academic Press, 1987.

[Gardner 1982] Gardner, M., *Logic Machines and Diagrams* (second edition). Chicago: The University of Chicago Press, 1982.

[Garey & Johnson 1979] Garey, M., and Johnson, D., *Computers and Intractability*, New York: W. H. Freeman, 1979.

[Gaschnig 1979] Gaschnig, J., "Performance Measurement and Analysis of Certain Search Algorithms," Carnegie-Mellon University Computer Science Tech. Report CMU-CS-79-124, Pittsburg, PA, 1979.

[Gaschnig 1979] Gaschnig, J., "A Problem-Similarity Approach to Devising Heuristics: First Results," in *Proceedings of the Sixth International Joint Conference on Artificial Intelligence (IJCAI-79)*, pp. 301–307, San Francisco: Morgan Kaufmann, 1979. (Reprinted in Webber, B., and Nilsson, N. (eds.), *Readings in Artificial Intelligence*, pp. 23–29, San Francisco: Morgan Kaufmann, 1981.)

[Gat 1992] Gat, E., "Integrating Planning and Reacting in a Heterogeneous Asynchronous Architecture for Controlling Real-World Mobile Robots," in *Proceedings of the Tenth National Conference on Artificial Intelligence (AAAI-92)*, pp. 809–815, Menlo Park, CA:

AAAI Press, 1992.

[Gelernter 1959] Gelernter, H., "Realization of a Geometry Theorem-Proving Machine," *Proc. Intern. Conf. Inform Proc.*, UNESCO House, Paris, pp. 273–282, 1959. (Reprinted in Feigenbaum, E., and Feldman, J. (eds.), *Computers and Thought*, pp. 134–152, New York: McGraw-Hill, 1963.)

[Genesereth & Nilsson 1987] Genesereth, M., and Nilsson, N., *Logical Foundations of Artificial Intelligence*, San Francisco: Morgan Kaufmann, 1987.

[Genesereth & Fikes 1992] Genesereth, M., and Fikes, R. (eds.), *Knowledge Interchange Format, Version 3.0 Reference Manual*, Computer Science Department, Stanford University, Technical Report Logic-92-1, June 1992.

[Gentner 1983] Gentner, D., "Structure Mapping: A Theoretical Framework for Analogy," *Cognitive Science*, 7:155–170, 1983.

[Gibson 1950] Gibson, J. J., *The Perception of the Visual World*, Boston: Houghton Mifflin, 1950.

[Gibson 1979] Gibson, J. J., *The Ecological Approach to Visual Perception*, Boston: Houghton Mifflin, 1979.

[Gil 1992] Gil, Y., *Acquiring Domain Knowledge for Planning by Experimentation*, Ph.D. dissertation, School of Computer Science, Carnegie-Mellon University, 1992.

[Ginsberg 1987] Ginsberg, M. (ed.), *Readings in Nonmonotonic Reasoning*, San Francisco: Morgan Kaufmann, 1987.

[Ginsberg 1993] Ginsberg, M., "Dynamic Backtracking," *Journal of Artificial Intelligence Research*, 1:25–46, 1993.

[Ginsberg 1996] Ginsberg, M., "Do Computers Need Commonsense?" in *Proceedings of the Fifth International Conference on Principles of Knowledge Representation and Reasoning (KR-96)*, pp. 620–626, San Francisco: Morgan Kaufmann, 1996.

[Ginsberg, et al. 1990] Ginsberg, M., Frank, M., Halpin, M., and Torrance, M., "Search Lessons Learned from Crossword Puzzles," in *Proceedings of the Eighth National Conference on Artificial Intelligence (AAAI-90)*, pp. 210–215, Menlo Park, CA: AAAI Press, 1990.

[Ginsberg & Harvey 1992] Ginsberg, M. L., and Harvey. W. D., "Iterative Broadening," *Artificial Intelligence*, 55(2/3):367–383, 1992.

[Glesner & Koller 1995] Glesner, S., and Koller, D., "Constructing Flexible Dynamic Belief Networks from First-Order Probabilistic Knowledge Bases," *Proceedings of the European Conference on Symbolic and Quantitative Approaches to Reasoning and Uncertainty (ECSQARU)*, Fribourg, Switzerland, July 1995, in Froidevaux, C., and Kohlas, J. (eds.), *Lecture Notes in Artificial Intelligence*, pp. 217–226, Berlin: Springer-Verlag, 1995.

[Gogic, et al. 1995] Gogic, G., Kautz, H., Papadimitriou, C., and Selman, B., "The Comparative Linguistics of Knowledge Representations," in *Proceedings of the Fourteenth International Joint Conference on Artificial Intelligence (IJCAI-95)*, pp. 862–869, San Francisco: Morgan Kaufmann, 1995.

[Goldberg 1979] Goldberg, A., "On the Complexity of the Satisfiability Problem," Courant Computer Science Report No. 16, New York University, NY, 1979.

[Goldberg 1989] Goldberg, D., *Genetic Algorithms in Search, Optimization, and Machine Learning*, Reading, MA: Addison-Wesley, 1989.

[Goldszmidt, Morris, & Pearl 1990] Goldszmidt, M., Morris, P., and Pearl, J., "A Maximum Entropy Approach to Nonmonotonic Reasoning," in *Proceedings of the Eighth National Conference on Artificial Intelligence (AAAI-90)*, pp. 646–652, Menlo Park, CA: AAAI Press, 1990.

[Green 1969a] Green, C., "Application of Theorem Proving to Problem Solving," in *Proceedings of the First International Joint Conference on Artificial Intelligence (IJCAI-69)*, pp. 741–747, San Francisco: Morgan Kaufmann, 1969.

[Green 1969b] Green, C., "Theorem-Proving by Resolution as a Basis for Question-Answering Systems," in Meltzer, B., and Michie, D. (eds.), *Machine Intelligence 4*, pp. 183–205, Edinburgh: Edinburgh University Press, 1969.

[Gregory 1966] Gregory, R., *Eye and Brain: The Psychology of Seeing*, New York: McGraw-Hill, 1966.

[Grimson 1990] Grimson, W. E. L., *Object Recognition by Computer: The Role of Geometric Constraints*, Cambridge, MA: MIT Press, 1990.

[Grosz, Sparck Jones, & Webber 1986] Grosz, B., Sparck Jones, K., and Webber, B. (eds.), *Readings in Natural Language Processing*, San Francisco: Morgan Kaufmann, 1986.

[Grosz, et al. 1987] Grosz, B., Appelt, D., Martin, P., and Pereira, F., "Team: An Experiment in the Design of Transportable Natural-Language Interfaces," *Artificial Intelligence*, 32(2):173–244, 1987.

[Gruber 1997] Gruber, T., "Toward Principles for the Design of Ontologies Used for Knowledge Sharing," in Guarino, N., and Poli, R. (eds.), *Formal Ontology in Conceptual Analysis and Knowledge Representation*, Amsterdam: Kluwer Academic Publishers, 1997. (Original paper presented at the International Workshop on Formal Ontology, March 1993, Stanford Knowledge Systems Laboratory Report KSL-93-04.)

[Gu 1989] Gu, J., *Parallel Algorithms and Architectures for Very Fast AI Search*, Ph.D. thesis, University of Utah, 1989.

[Guha & Lenat 1990] Guha, R. V., and Lenat, D., "Cyc: A Midterm Report," *AI Magazine*, pp. 33–59, Fall 1990.

[Gupta & Nau 1992] Gupta, N., and Nau, D., "On the Complexity of Blocks-World Planning," *Artificial Intelligence*, 56(2/3):223–254, 1992.

[Guzman 1968] Guzman, A., "Decomposition of a Visual Scene into Three-Dimensional Bodies," *Proc Fall Joint Computer Conference*, vol. 33, pp. 291–304, 1968.

[Hanks & McDermott 1986] Hanks, S., and McDermott, D., "Default Reasoning, Nonmonotonic Logics, and the Frame Problem," in *Proceedings of the Fifth National Conference on Artificial Intelligence (AAAI-86)*, pp. 328–333, Menlo Park, CA: AAAI Press, 1986. (Reprinted in Ginsberg, M. (ed.), *Readings in Nonmonotonic Reasoning*, pp. 390–395, San Francisco: Morgan Kaufmann, 1987.)

[Harnad 1990] Harnad, S., "The Symbol Grounding Problem," *Physica D*, 42(1–3):335–346, 1990.

[Hart, Nilsson, & Raphael 1968] Hart, P., Nilsson, N., and Raphael, B., "A Formal Basis for the Heuristic Determination of Minimum Cost Paths," *IEEE Trans. Syst. Science and Cybernetics*, SSC-4(2):100–107, 1968.

[Hart, Nilsson, & Raphael 1972] Hart, P., Nilsson, N., and Raphael, B., "Correction to 'A Formal

Basis for the Heuristic Determination of Minimum Cost Paths,' " *SIGART Newsletter*, no. 37, pp. 28–29, December 1972.

[Harvey 1994] Harvey, W. D., *Nonsystematic Backtracking Search*, Ph.D. dissertation, Department of Computer Science, Stanford University, December 1994.

[Haussler 1988] Haussler, D., "Quantifying Inductive Bias: AI Learning Algorithms and Valiant's Learning Framework," *Artificial Intelligence*, 36:177–221, 1988. (Reprinted in Shavlik, J., and Dietterich, T. (eds.), *Readings in Machine Learning*, pp. 96–107, San Francisco: Morgan Kaufmann, 1990.)

[Haussler 1990] Haussler, D., "Probably Approximately Correct Learning," *Proceedings of the Eighth National Conference on Artificial Intelligence (AAAI-90)*, pp. 1101–1108, Menlo Park, CA: AAAI Press, 1990.

[Hayes 1978] Hayes, P. J., "The Naive Physics Manifesto," in Michie, D. (ed.), *Expert Systems in the Microelectronic Age*, Edinburgh: Edinburgh University Press, 1978.

[Hayes 1985a] Hayes, P. J., "The Second Naive Physics Manifesto," in Hobbs, J., and Moore, R. (eds.), *Formal Theories of the Commonsense World*, Ch. 1, pp. 1–36, Norwood, NJ: Ablex, 1985.

[Hayes 1985b] Hayes, P. J., "Naive Physics I: Ontology for Liquids," in Hobbs, J., and Moore, R. (eds.), *Formal Theories of the Commonsense World*, Ch. 3, pp. 71–107, Norwood, NJ: Ablex, 1985.

[Hayes & Ford 1995] Hayes, P., and Ford, K., "Turing Test Considered Harmful," in *Proceedings of the Fourteenth International Joint Conference on Artificial Intelligence (IJCAI-95)*, pp. 972–977, San Francisco: Morgan Kaufmann, 1995.

[Hayes-Roth 1985] Hayes-Roth, B., "A Blackboard Architecture for Control," *Artificial Intelligence*, 26(3):251–321, 1985.

[Hayes-Roth 1995] Hayes-Roth, B., "An Architecture for Adaptive Intelligent Systems," *Artificial Intelligence*, 72(1/2):329–365, 1995.

[Hayes-Roth, et al. 1992] Hayes-Roth, B., Washington, R., Ash, D., Hewett, R., Collinot, A., Vina, A., Seiver, A., "Guardian: An Intelligent Agent for ICU Monitoring," *J. AI in Medicine*, 4:165–185, 1992.

[Haykin 1994] Haykin, S., *Neural Networks: A Comprehensive Foundation*, New York: Macmillan College Publishing, 1994.

[Hebert, et al. 1997] Hebert, M., et al., "Mobility for Unmanned Ground Vehicles," in Firschein, O., and Strat, T. (eds.), *Reconnaissance, Surveillance, and Target Acquisition for the Unmanned Ground Vehicle: Providing the Surveillance "Eyes" for an Autonomous Vehicle*, San Francisco: Morgan Kaufmann, 1997.

[Heckerman 1991] Heckerman, D., *Probabilistic Similarity Networks*, Cambridge, MA: MIT Press, 1991.

[Heckerman 1996] Heckerman, D., "A Tutorial on Learning with Bayesian Networks," Microsoft Research Technical Report, MSR-TR-95-06, Redmond, WA, March 1995 (revised November 1996).

[Heckerman & Nathwani 1992] Heckerman, D., and Nathwani, B., "An Evaluation of the Diagnostic Accuracy of Pathfinder," *SIAM Journal on Computing*, 25:56–74, 1992.

[Heckerman, Geiger, & Chickering 1995] Heckerman, D., Geiger, D., and Chickering, D., "Learning Bayesian Networks: The Combination of Knowledge and Statistical Data," *Machine Learning*, 20:3, 197–243, 1995.

[Heinsohn, et al. 1992] Heinsohn, J., Kudenko, D., Nebel, B., and Profitlich, H.-J., "An Empirical Analysis of Terminological Representation Systems," *Proceedings of the Tenth National Conference on Artificial Intelligence (AAAI-92)*, pp. 767–773, Menlo Park, CA: AAAI Press, 1992.

[Hendrix 1979] Hendrix, G., "Encoding Knowledge in Partitioned Networks," in Findler, N. (ed.), *Associative Networks*, pp. 51–92, New York: Academic Press, 1979.

[Henrion 1988] Henrion, M., "Propagation of Uncertainty in Bayesian Networks by Probabilistic Logic Sampling," in Lemmer, J., and Kanal, L. (eds.), *Uncertainty in Artificial Intelligence*, 2:149–163, New York: Elsevier/North-Holland, 1988.

[Henrion 1990] Henrion, M., "An Introduction to Algorithms for Inference in Belief Nets," in Henrion, M., Shachter, R., Kanal, L., and Lemmer, J. (eds.), *Uncertainty in Artificial Intelligence*, 5, Amsterdam: North Holland, 1990.

[Hertz, Krogh, & Palmer 1991] Hertz, J., Krogh, A., and Palmer, R., *Introduction to the Theory of Neural Computation*, Reading, MA: Addison-Wesley, 1991.

[Hintikka 1962] Hintikka, J., *Knowledge and Belief*, Ithaca, NY: Cornell University Press, 1962.

[Hobbs & Moore 1985] Hobbs, J., and Moore, R., (eds.), *Formal Theories of the Commonsense World*, Norwood, NJ: Ablex, 1985.

[Holland 1975] Holland, J., *Adaptation in Natural and Artificial Systems*, Ann Arbor: The University of Michigan Press, 1975. (Second edition printed in 1992 by MIT Press, Cambridge, MA.)

[Holland 1986] Holland, J. H., "Escaping Brittleness: The Possibilities of General-Purpose Learning Algorithms Applied to Parallel Rule-Based Systems," in Michalski, R., Carbonell, J., and Mitchell, T. (eds.), *Machine Learning: An Artificial Intelligence Approach, Volume 2*, Ch. 20, San Francisco: Morgan Kaufmann, 1986.

[Horn 1951] Horn, A., "On Sentences Which Are True of Direct Unions of Algebras," *Jour. of Symbolic Logic*, 16:14–21, 1951.

[Horn 1986] Horn, B. K. P., *Robot Vision*, Cambridge, MA: MIT Press, 1986.

[Horowitz & Pavlidis 1976] Horowitz, S., and Pavlidis, T., "Picture Segmentation by a Tree Traversal Algorithm," *Jour. Assoc. Comp. Mach.*, 23(2):368–388, April 1976.

[Horswill 1993] Horswill, I., "Polly: A Vision-Based Artificial Agent," in *Proceedings of the Eleventh National Conference on Artificial Intelligence (AAAI-93)*, pp. 824–829, Menlo Park, CA: AAAI Press, 1993.

[Horvitz 1987] Horvitz, E., "Reasoning about Beliefs and Actions under Computational Resource Constraints," in *Proc. of the 1987 Workshop on Uncertainty in Artificial Intelligence*, pp. 429–444, 1987. (Also in Kanal, L., et al. (eds.), *Uncertainty in Artificial Intelligence 3*, pp. 301–324, New York: Elsevier, 1989.)

[Horvitz, Breese, & Henrion 1988] Horvitz, E., Breese, J., and Henrion, M., "Decision Theory in Expert Systems and Artificial Intelligence," *International Journal of Approximate*

Reasoning, 2:247–302, 1988.

[Hubel 1988] Hubel, D., *Eye, Brain, and Vision*, New York: W. H. Freeman, 1988.

[Hubel & Wiesel 1968] Hubel, D., and Wiesel, T., "Receptive Fields and Functional Architecture of Monkey Striate Cortex," *Journal of Physiology (London)*, 195(1):215–243, March 1968.

[Huberman 1968] Huberman, B. J., *A Program to Play Chess End Games*, Stanford University Computer Science Department Report CS 106, August 19, 1968.

[Hueckel 1973] Hueckel, M., "A Local Visual Operator Which Recognizes Edges and Lines," *Journal of the Assoc. for Comp. Machinery*, 20(4):634–647, October 1973.

[Huffman 1971] Huffman, D., "Impossible Objects as Nonsense Sentences," in *Machine Intelligence 6*, Meltzer, B., and Michie, D. (eds.), pp. 295–323, New York: American Elsevier, 1971.

[Jain, Kasturi, & Schunck 1995] Jain, R., Kasturi, R., and Schunck, B., *Machine Vision*, New York: McGraw-Hill, 1995.

[Jensen 1996] Jensen, F., *An Introduction to Bayesian Networks*, New York: Springer-Verlag, 1996.

[Johnson-Laird 1988] Johnson-Laird, P., *The Computer and the Mind: An Introduction to Cognitive Science*, Cambridge, MA: Harvard University Press, 1988.

[Jordan & Rumelhart 1992] Jordan, M., and Rumelhart, D., "Forward Models: Supervised Learning with a Distal Teacher," *Cognitive Science*, 16:307–354, 1992.

[Juels & Wattenberg 1996] Juels, A., and Wattenberg, M., "Stochastic Hillclimbing as a Baseline Method for the Evaluation of Genetic Algorithms," in *Neural Information Processing Systems 8*, Cambridge, MA: MIT Press, 1996.

[Julesz 1971] Julesz, B., *Foundations of Cyclopean Perception*, Chicago: The University of Chicago Press, 1971.

[Kaelbling & Rosenschein 1990] Kaelbling, L., and Rosenschein, S., "Action and Planning in Embedded Agents," *Robotics and Autonomous Systems*, 6:35–48, 1990. Also in [Maes 1990].

[Kaelbling, Littman, & Moore 1996] Kaelbling, L. P., Littman, M. L., and Moore, A. W., "Reinforcement Learning: A Survey," *Journal of Artificial Intelligence Research*, 4:237–285, 1996.

[Kanal & Kumar 1988] Kanal, L., and Kumar, V. (eds.), *Search in Artificial Intelligence*, Berlin: Springer-Verlag, 1988.

[Karmarkar 1984] Karmarkar, N., "A New Polynomial-Time Algorithm for Linear Programming," *Combinatorica*, 4(4):373–395, 1984.

[Kartha 1994] Kartha, G. N., "Two Counterexamples Related to Baker's Approach to the Frame Problem," *Artificial Intelligence*, 69(1–2):379–391, 1994.

[Kautz 1985] Kautz, H., "Formalizing Spatial Concepts and Spatial Language," in *Commonsense Summer: Final Report*, Center for the Study of Language and Information (CSLI) Technical Report CSLI-85-35, Stanford University, Stanford, CA, 1985.

[Kautz 1991] Kautz, H., "A Formal Theory of Plan Recognition and Its Implementation," in Allen, J., Kautz, H., Pelavin, R., and Tenenberg, J. (eds.), *Reasoning About Plans*, Ch. 2, San Francisco: Morgan Kaufmann, 1991.

[Kautz & Selman 1992] Kautz, H., and Selman, B., "Forming Concepts for Fast Inference," in *Proceedings of the Tenth National Conference on Artificial Intelligence (AAAI-92)*, pp. 787–793, Menlo Park, CA: AAAI Press, 1992.

[Kautz & Selman 1996] Kautz, H., and Selman, B., "Pushing the Envelope: Planning, Propositional Logic, and Stochastic Search," in *Proceedings of the Thirteenth National Conference on Artificial Intelligence (AAAI-96)*, pp. 1194–1201, Menlo Park, CA: AAAI Press, 1996.

[Kautz, Kearns, & Selman 1993] Kautz, H., Kearns, M., and Selman, B., "Reasoning with Characteristic Models," *Proceedings of the Eleventh National Conference on Artificial Intelligence (AAAI-93)*, pp. 34–39, Menlo Park, CA: AAAI Press, 1993.

[Kautz, McAllester, & Selman 1996] Kautz, H., McAllester, D., and Selman, B., "Encoding Plans in Propositional Logic," in *Proceedings of the Fifth International Conference on Principles of Knowledge Representation and Reasoning (KR-96)*, pp. 374–384, San Francisco: Morgan Kaufmann, 1996.

[Kearns & Vazirani 1994] Kearns, M., and Vazirani, U., *An Introduction to Computational Learning Theory*, Cambridge, MA: MIT Press, 1994.

[Khardon & Roth 1998] Khardon, R., and Roth, D., "Learning to Reason," *Journal of the ACM*, to appear, 1998.

[Kierulf, Chen, & Nievergelt 1990] Kierulf, A., Chen, K., and Nievergelt, J., "Smart Game Board and Go Explorer: A Study in Software and Knowledge Engineering," *Comm. of the ACM*, 33(2):152–167, 1990.

[Kim & Pearl 1983] Kim, J., and Pearl, J., "A Computational Model for Combined Causal and Diagnostic Reasoning in Inference Systems," in *Proceedings of the Eighth International Joint Conference on Artificial Intelligence (IJCAI-83)*, pp. 190–193, San Francisco: Morgan Kaufmann, 1983.

[Kirkpatrick, Gelatt, & Vecchi 1983] Kirkpatrick, S., Gelatt, C., and Vecchi, M., "Optimization by Simulated Annealing," *Science*, 220:671–680, 1983.

[Kirsh 1991] Kirsh, D., "Today the Earwig, Tomorrow Man?" *Artificial Intelligence*, 47(1–3):161–184, 1991.

[Knuth & Moore 1975] Knuth, D. E., and Moore, R. W., "An Analysis of Alpha–Beta Pruning," *Artificial Intelligence*, 6(4), 293–326, 1975.

[Knoblock 1990] Knoblock, C. A., "Learning Abstraction Hierarchies for Problem Solving," in *Proceedings of the Eighth National Conference on Artificial Intelligence (AAAI-90)*, pp. 923–928, Menlo Park, CA: AAAI Press, 1990.

[Koenderink 1984] Koenderink, J., "The Structure of Images," *Biological Cybernetics*, 50:363–370, 1984.

[Kolodner 1993] Kolodner, J., *Case-Based Reasoning*, San Francisco: Morgan Kaufmann, 1993.

[Konolige 1982] Konolige, K., "A First-Order Formalization of Knowledge and Action for a Multi-Agent Planning System," in Hayes, J., Michie, D., and Pao, Y. (eds.), *Machine Intelligence 10*, Chichester, England: Ellis Horwood, Ltd., 1982.

[Konolige 1986] Konolige, K., *A Deduction Model of Belief*, London: Pitman, 1986.

[Korf 1985] Korf, R., "Depth-First Iterative Deepening: An Optimal Admissible Tree Search

Algorithm," *Artificial Intelligence*, 27:97–109, 1985.

[Korf 1987] Korf, R., "Planning as Search: A Quantitative Approach," *Artificial Intelligence*, 33(1):65–88, 1987.

[Korf 1990] Korf, R., "Real-Time Heuristic Search," *Artificial Intelligence*, 42, 1990.

[Korf 1991] Korf, R., "Multi-Player Alpha-Beta Pruning," *Artificial Intelligence*, 48:99–111, 1991.

[Korf 1992] Korf, R., "Search," in Shapiro, S. (ed.), *Encyclopedia of Artificial Intelligence, Second Edition*, pp. 1460–1467, New York: John Wiley & Sons, 1992.

[Korf 1993] Korf, R., "Linear-Space Best-First Search," *Artificial Intelligence*, 62, 41–78, 1993.

[Korf 1996] Korf, R., "Space-Efficient Search Algorithms," *ACM Computing Surveys*, 27(3):337–339, 1996.

[Korf 1997] Korf, R., "Finding Optimal Solutions to Rubik's Cube Using Pattern Databases," in *Proceedings of the Fourteenth National Conference on Artificial Intelligence (AAAI-97)*, pp. 700--705, Menlo Park, CA: AAAI Press, 1997.

[Korf & Taylor 1996] Korf, R., and Taylor, L., "Finding Optimal Solutions to the Twenty-Four Puzzle," in *Proceedings of the Thirteenth National Conference on Artificial Intelligence (AAAI-96)*, pp. 1202–1207, Menlo Park, CA: AAAI Press, 1996.

[Kowalski 1974] Kowalski, R., "Predicate Logic as a Programming Language," in *Proceedings of the IFIP-74 Congress*, pp. 569–574, Amsterdam: Elsevier/North-Holland, 1974.

[Kowalski & Kim 1991] Kowalski, R., and Kim, J.-S., "A Metalogic Programming Approach to Multi-Agent Belief," in Lifschitz, V. (ed.), *Artificial Intelligence and Mathematical Theory of Computation: Papers in Honor of John McCarthy*, pp. 231–246, Boston: Academic Press, 1991.

[Koza 1992] Koza, J., *Genetic Programming: On the Programming of Computers by Means of Natural Selection*, Cambridge, MA: MIT Press, 1992.

[Koza 1994] Koza, J., *Genetic Programming II: Automatic Discovery of Reusable Programs*, Cambridge, MA: MIT Press, 1994.

[Koza, et al. 1996] Koza, J., Bennett, F., III, Andre, D., and Keane, M., "Automated WYSIWYG Design of Both the Topology and Component Values of Analog Electrical Circuits Using Genetic Programming," in Koza, John R., Goldberg, David E., Fogel, David B., and Riolo, Rick L. (eds.), *Genetic Programming 1996: Proceedings of the First Annual Conference*, Stanford University, July 28–31, 1996, Cambridge, MA: MIT Press.

[Kripke 1963] Kripke, S., "Semantical Considerations on Modal Logic," *Acta Philosophica Fennica*, 16:83–94, 1963.

[Kuipers, et al. 1993] Kuipers, B., Froom, R., Lee, W-Y., Pierce, D., "The Semantic Hierarchy in Robot Learning," in Connell, J., and Mahadevan, S. (eds.), *Robot Learning*, Ch. 6, Boston: Kluwer Academic Publishers, 1993.

[Kuipers & Byun 1991] Kuipers, B., and Byun, Y-T., "A Robot Exploration and Mapping Strategy Based on a Semantic Hierarchy of Spatial Representations," *Robotics and Autonomous Systems*, 8:47–63, 1991.

[Kumar & Kanal 1988] Kumar, V., and Kanal, L., "The CDP: A Unifying Formulation for Heuristic Search, Dynamic Programming, and Branch-and-Bound," in Kanal, L., and Kumar, V. (eds.), *Search in Artificial Intelligence*, Ch. 1, pp. 1–27, Berlin:

Springer-Verlag, 1988.

[Kumar 1992] Kumar, V., "Algorithms for Constraint-Satisfaction Problems: A Survey," *Artificial Intelligence Magazine*, 13(1):32–44, Spring 1992.

[Laird, Newell, & Rosenbloom 1987] Laird, J., Newell, A., and Rosenbloom, P., "SOAR: An Architecture for General Intelligence," *Artificial Intelligence*, 33(1):1–64, 1987.

[Laird, et al. 1991] Laird, J., Yager, E., Hucka, M., and Tuck, C., "Robo-Soar: An Integration of External Interaction, Planning, and Learning Using SOAR," *Robotics and Autonomous Systems*, 8:113–129, 1991.

[Lakoff 1987] Lakoff, G., *Women, Fire, and Dangerous Things: What Categories Reveal about the Mind*, Chicago: The University of Chicago Press, 1987.

[Lakoff & Johnson 1980] Lakoff, G., and Johnson, M., *Metaphors We Live By*, Chicago: The University of Chicago Press, 1980.

[Langley 1996] Langley, P., *Elements of Machine Learning*, San Francisco: Morgan Kaufmann, 1996.

[Latombe 1991] Latombe, J.-C., *Robot Motion Planning*, Dordrecht: Kluwer Academic Publishers, 1991.

[Lauritzen 1991] Lauritzen, S., "The EM Algorithm for Graphical Association Models with Missing Data," Tech. Report TR-91-05, Dept. of Statistics, Aalborg University, Denmark, 1991.

[Lauritzen & Spiegelhalter 1988] Lauritzen, S., and Spiegelhalter, D., "Local Computations with Probabilities on Graphical Structures and Their Application to Expert Systems," *Journal of the Royal Statistical Society*, B 50(2):157–224, 1988.

[Lavrač & Džeroski 1994] Lavrač, N., and Džeroski, S., *Inductive Logic Programming*, Chichester, England: Ellis Horwood, 1994.

[LeCun, et al. 1989] LeCun, Y., Boser, B., Denker, J., Henderson, D., Howard, R., Hubbard, W., and Jackel, L., "Backpropagation Applied to Handwritten Zip Code Recognition," *Neural Computation*, 1(4), 1989.

[Lee & Mahajan 1988] Lee, K.-F., and Mahajan, S., "A Pattern Classification Approach to Evaluation Function Learning," *Artificial Intelligence*, 36(1):1–26, 1988.

[Lenat 1995] Lenat, D., "CYC: A Large-Scale Investment in Knowledge Infrastructure," *Comm. ACM*, 38(11):33–38, November 1995.

[Lenat & Guha 1990] Lenat, D., and Guha, R., *Building Large Knowledge Bases*, Reading, MA: Addison-Wesley, 1990.

[Leonard-Barton 1987] Leonard-Barton, D., "The Case for Integrative Innovation: An Expert System at Digital," *Sloan Management Review*, pp. 7–19, Fall 1987.

[Letvinn, et al. 1959] Letvinn, J., Maturana, H., McCulloch, W., and Pitts, W., "What the Frog's Eye Tells the Frog's Brain," *Proc. IRE*, 47:1940–1951, 1959.

[Levesque 1984a] Levesque, H., "Foundations of a Functional Approach to Knowledge Representation," *Artificial Intelligence*, 23(2):155–212, 1984.

[Levesque 1984b] Levesque, H., "A Logic of Implicit and Explicit Belief," in *Proceedings of the Fourth National Conference on Artificial Intelligence (AAAI-84)*, pp. 198–202, Menlo Park, CA:

AAAI Press, 1984.

[Levesque 1986] Levesque, H., "Making Believers Out of Computers," *Artificial Intelligence*, 30(1):81–108, 1986.

[Levesque & Brachman 1987] Levesque, H., and Brachman, R., "Expressiveness and Tractability in Knowledge Representation and Reasoning," *Computational Intelligence*, 3(2):78–93, 1987.

[Levesque, et al. 1997] Levesque, H., Reiter, R., Lesprance, Y., Lin, F., and Scherl, R., "GOLOG: A Logic Programming Language for Dynamic Domains," *Journal of Logic Programming*, Special Issue on Reasoning about Action and Change, 31(1–3):59–83, 1997.

[Lifschitz 1986] Lifschitz, V., "On the Semantics of STRIPS," in Georgeff, M., and Lansky, A. (eds.), *Reasoning about Actions and Plans: Proceedings of the 1986 Workshop*, Timberline, Oregon, pp. 1–9, San Francisco: Morgan Kaufmann, 1986. (Also in Allen, J., Hendler, J., and Tate, A. (eds.), *Readings in Planning*, pp. 523–530, San Francisco: Morgan Kaufmann, 1990.)

[Lindsay, et al. 1980] Lindsay, R., Buchanan, B., Feigenbaum, E., and Lederberg, J., *Applications of Artificial Intelligence for Organic Chemistry: The DENDRAL Project*, New York: McGraw-Hill, 1980.

[Lovejoy 1991] Lovejoy, W., "A Survey of Algorithmic Methods for Partially Observed Markov Decision Processes," *Annals of Operations Research*, 28(1–4):47–66, 1991.

[Loveland 1978] Loveland, D. W., *Automated Theorem Proving: A Logical Basis*, New York: North-Holland, 1978.

[Löwenheim 1915] Löwenheim, L., "Über Möglichkeiten im Relativekalkül," *Mathematische Annalen*, 76:447–470, 1915. (English translation appears in van Heijenoort, J. (ed.), *From Frege to Gödel: A Source Book in Mathematical Logic, 1879–1931*, Cambridge, MA: Harvard University Press, 1967.)

[Lowry & McCartney 1991] Lowry, M., and McCartney, R., *Automating Software Design*, Cambridge, MA: MIT Press, 1991.

[Lucas 1961] Lucas, J. R., "Minds, Machines, and Gödel," *Philosophy*, 36:112–127, 1961. (Also in Anderson, A. R. (ed.), *Minds and Machines*, pp. 43–59, Englewood Cliffs, NJ: Prentice Hall, 1964.)

[Luckham 1970] Luckham, D., "Refinement Theorems in Resolution Theory," *Proc. IRIA 1968 Symp. on Automatic Demonstration*, Springer-Verlag Lecture Notes in Mathematics, No. 125, pp. 163–190, 1970.

[Luckham & Nilsson 1971] Luckham, D. C., and Nilsson, N., "Extracting Information from Resolution Proof Trees," *Artificial Intelligence*, 2(1): 27–54, 1971.

[Maes 1990a] Maes, P. (ed.), *Designing Autonomous Agents: Theory and Practice from Biology to Engineering and Back*, Cambridge, MA: MIT Press, 1990.

[Maes 1990b] Maes, P., "Guest Editorial," *Robotics and Autonomous Systems*, 6:1–2, 1990. Also in [Maes 1990a].

[Magerman 1993] Magerman, D., *Natural Language Parsing as Statistical Pattern Recognition*, Ph.D. thesis, Department of Computer Science, Stanford University, 1993.

[Mahadevan 1992] Mahadevan, S., "Enhancing Transfer in Reinforcement Learning by Building

Stochastic Models of Robot Actions," in *Proceedings of the Ninth International Workshop on Machine Learning (ML92)*, pp. 290–299, San Francisco: Morgan Kaufmann, 1992.

[Mahadevan & Connell 1992] Mahadevan, S., and Connell, J., "Automatic Programming of Behavior-Based Robots Using Reinforcement Learning," *Artificial Intelligence*, 55(1–2):311–365, 1992.

[Manna & Waldinger 1985] Manna, Z., and Waldinger, R., *The Logical Basis for Computer Programming, Volume 1: Deductive Reasoning*, Reading, MA: Addison-Wesley, 1985.

[Manna & Waldinger 1990] Manna, Z., and Waldinger, R., *The Logical Basis for Computer Programming, Volume 2: Deductive Systems*, Reading, MA: Addison-Wesley, 1990.

[Manna & Waldinger 1992] Manna, Z., and Waldinger, R., "Fundamentals of Deductive Program Synthesis," *IEEE Transactions on Software Engineering*, 18(8):674–704, 1992.

[Marr 1982] Marr, D., *Vision: A Computational Investigation into the Human Representation and Processing of Visual Information*, New York: W. H. Freeman, 1982.

[Marr & Hildreth 1980] Marr, D., and Hildreth, E., "Theory of Edge Detection," *Proc. Royal Soc. of London, Series B*, Vol. 207, pp. 187–217, 1980.

[Marr & Poggio 1979] Marr, D., and Poggio, T., "A Computational Theory of Human Stereo Vision," *Proceedings of the Royal Society London, B*, 204:301–328, 1979.

[Martelli & Montanari 1973] Martelli, A., and Montanari, U., "Additive AND/OR Graphs," in *Proceedings of the Third International Joint Conference on Artificial Intelligence (IJCAI-73)*, pp. 1–11, San Francisco: Morgan Kaufmann, 1973.

[Masand, Linoff, & Waltz 1992] Masand, B., Linoff, G., and Waltz, D., "Classifying News Stories Using Memory Based Reasoning," in *Proceedings of ACM/SIGIR*, pp. 59–65, 1992.

[Mataric 1990] Mataric, M., "A Distributed Model for Mobile Robot Environment Learning and Navigation," *Technical Report No. AIM-TR-1228*, MIT AI Lab, MIT, Cambridge, MA, 1990.

[Mataric 1996] Mataric, M., "Designing and Understanding Adaptive Group Behavior," *Adaptive Behavior*, 4(1):51–80, 1996.

[Mataric 1997] Mataric, M., "Studying the Role of Embodiment in Cognition," *Cybernetics and Systems*, 28(6):457–470, (Special Issue on Epistemological Aspects of Embodied AI), July 1997.

[Mauldin 1994] Mauldin, M. L., "Chatterbots, Tinymuds, and the Turing Test: Entering the Loebner Prize Competition," in *Proceedings of the Twelfth National Conference on Artificial Intelligence (AAAI-94)*, pp. 16–21, Menlo Park, CA: AAAI Press, 1994.

[McAdams & Shapiro 1995] McAdams, H., and Shapiro, L., "Circuit Simulation of Genetic Networks," *Science*, 269:650–656, August 4, 1995.

[McAllester & Rosenblitt 1991] McAllester, D., and Rosenblitt, D., "Systematic Nonlinear Planning," in *Proceedings of the Ninth National Conference on Artificial Intelligence (AAAI-91)*, pp. 634–639, Menlo Park, CA: AAAI Press, 1991.

[McCarthy 1958] McCarthy, J., "Programs with Common Sense," *Mechanisation of Thought*

Processes, Proceedings of the Symposium of the National Physics Laboratory, Vol. I, pp. 77–84, London: Her Majesty's Stationary Office, 1958. (Also in Minsky, M. (ed.), *Semantic Information Processing*, pp. 403–410, Cambridge, MA: MIT Press, 1968, and Brachman, R., and Levesque, H. (eds.), *Readings in Knowledge Representation*, pp. 299–307, San Francisco: Morgan Kaufmann, 1985.)

[McCarthy 1977] McCarthy, J., "Epistemological Problems in Artificial Intelligence," in *Proceedings of the Fifth International Joint Conference on Artificial Intelligence (IJCAI-77)*, pp. 1038–1044, San Francisco: Morgan Kaufmann, 1977.

[McCarthy 1979a] McCarthy, J., "Ascribing Mental Qualities to Machines," in Ringle, M. (ed.), *Philosophical Perspectives in Artificial Intelligence*, pp. 161–195, Atlantic Highlands, NJ: Humanities Press, 1979.

[McCarthy 1979b] McCarthy, J., "First-Order Theories of Individual Concepts and Propositions," in Hayes, J., Michie, D., and Mikulich, L. (eds.), *Machine Intelligence 9*, pp. 129–147, Chichester, England: Ellis Horwood, Ltd., 1979.

[McCarthy 1980] McCarthy, J., "Circumscription: A Form of Non-monotonic Reasoning," *Artificial Intelligence*, 13(1–2):27–39, 1980.

[McCarthy 1986] McCarthy, J., "Applications of Circumscription to Formalizing Commonsense Knowledge," *Artificial Intelligence*, 28(1):89–116, 1986.

[McCarthy 1997] McCarthy, J., "AI as Sport," (a Review of [Newborn 1996]), *Science*, 276:1518–1519, June 6, 1997.

[McCarthy & Hayes 1969] McCarthy, J. and Hayes, P., "Some Philosophical Problems from the Standpoint of Artificial Intelligence," in Meltzer, B., and Michie, D. (eds.), *Machine Intelligence 4*, Edinburgh: Edinburgh University Press, 1969.

[McClelland & Rumelhart 1981] McClelland, J., and Rumelhart, D., "An Interactive Activation Model of Context Effects in Letter Perception, Part I: An Account of Basic Findings," *Psychological Review*, 88:375–407, 1981.

[McClelland & Rumelhart 1982] McClelland, J., and Rumelhart, D., "An Interactive Activation Model of Context Effects in Letter Perception, Part II: The Contextual Enhancement Effect and Some Tests and Extensions of the Model," *Psychological Review*, 89:60–84, 1982.

[McCorduck 1979] McCorduck, P., *Machines Who Think*, San Francisco: W. H. Freeman, 1979.

[McCulloch & Pitts 1943] McCulloch, W., and Pitts, W., "A Logical Calculus of the Ideas Immanent in Nervous Activity," *Bulletin of Math. Biophysics*, 5:115–133, 1943.

[McCune 1992] McCune, W., "Automated Discovery of New Axiomatizations of the Left Group and Right Group Calculi," *Automated Reasoning*, 9(1):1–24, 1992.

[McCune 1994] McCune, W., *OTTER 3.0 Reference Manual and Guide*, Technical Report ANL-94/6, Argonne National Laboratory, Argonne, IL, 1994.

[McDermott 1987] McDermott, D., "A Critique of Pure Reason," *Computational Intelligence*, 3(3):151–237, 1987.

[McDermott & Doyle 1980] McDermott, D., and Doyle, J., "Non-monotonic Logic I," *Artificial Intelligence*, 13(1–2):41–72, 1980.

[McDermott 1982] McDermott, J., "R1: A Rule-Based Configurer of Computer Systems," *Artificial*

Intelligence, 19(1):39–88, 1982.

[McDonald & Bolc 1988] McDonald, D., and Bolc, L., *Natural Language Generation Systems*, New York: Springer-Verlag, 1988.

[McFarland 1987] McFarland, D. (ed.), *The Oxford Companion to Animal Behavior*, Oxford: Oxford University Press, 1987.

[McFarland & Bösser 1993] McFarland, D., and Bösser, T., *Intelligent Behavior in Animals and Robots*, Cambridge, MA: MIT Press, 1993.

[McKeown & Swartout 1987] McKeown, K., and Swartout, W., "Language Generation and Explanation," in *Annual Review of Computer Science*, vol. 2, Palo Alto, CA: Annual Reviews, 1987.

[Mérõ 1984] Mérõ, L., "A Heuristic Search Algorithm with Modifiable Estimate," *Artificial Intelligence*, 23:13–27, 1984.

[Michalewicz 1992] Michalewicz, Z., *Genetic Algorithms + Data Structures = Evolution Programs*, Berlin: Springer-Verlag, 1992.

[Michalski 1969] Michalski, R., "On the Quasi-Minimal Solution of the General Covering Problem," *Proc. of the Fifth International Symposium on Information Processing (FCIP 69)*, Vol. A3, pp. 125–128, (Switching Circuits), Bled, Yugoslavia, 1969.

[Michie 1966] Michie, D., "Game-Playing and Game-Learning Automata," in Fox, L. (ed.), *Advances in Programming and Non-numerical Computation*, pp. 183–200, New York: Pergamon, 1966.

[Miller, Pople, & Myers 1982] Miller, R., Pople, H., and Myers, J., "INTERNIST-1: An Experimental Computer-Based Diagnostic Consultant for General Internal Medicine," *New England Journal of Medicine*, 307:468–476, 1982.

[Minker 1997] Minker, J., "Logic and Databases: Past, Present, and Future," *AI Magazine*, 18(3):21–47, Fall 1997.

[Minnix, McVey, & Iñigo 1991] Minnix, J., McVey, E., and Iñigo, R., "Multistage Self-Organizing Neural Network with Biologically Inspired Preprocessing Features for Rotation and Scale Invariant Pattern Recognition," *Proc. of the IEEE*, pp. 1605–1610, 1991.

[Minsky 1967] Minsky, M., *Computation: Finite and Infinite Machines*, Englewood Cliffs, NJ: Prentice Hall, 1967.

[Minsky 1986] Minsky, M., *Society of Mind*, New York: Simon & Schuster, 1986.

[Minton 1988] Minton, S., *Learning Search Control Knowledge: An Explanation-Based Approach*, Boston: Kluwer Academic Publishers, 1988.

[Minton 1990] Minton, S., "Quantitative Results Concerning the Utility of Explanation-Based Learning," *Artificial Intelligence*, 42(2–3):363–391, 1990.

[Minton 1993] Minton, S. (ed.), *Machine Learning Methods for Planning*, San Francisco: Morgan Kaufmann, 1993.

[Minton, et al. 1989] Minton, S., Carbonell, J., Knoblock, C., Kuokka, D., Etzioni, O., and Gil, Y., "Explanation-Based Learning: A Problem Solving Perspective," *Artificial Intelligence*, 40(1–3):63–118, 1989.

[Minton, et al. 1990] Minton, S., Johnston, M., Phillips, A., and Laird, P., "Solving Large-Scale Constraint-Satisfaction and Scheduling Problems Using a Heuristic Repair

Method," in *Proceedings of the Eighth National Conference on Artificial Intelligence (AAAI-90)*, pp. 17–24, Menlo Park, CA: AAAI Press, 1990.

[Minton, et al. 1992] Minton, S., Johnston, M., Philips, A., and Laird, P., "Minimizing Conflicts: A Heuristic Repair Method for Constraint Satisfaction and Scheduling Problems," *Artificial Intelligence*, 58(1–3):161–205, 1992.

[Minton, Bresina, & Drummond 1994] Minton, S., Bresina, J., and Drummond, M., "Total-Order and Partial-Order Planning: A Comparative Analysis," *Journal of Artificial Intelligence Research*, 2:227–262, 1994.

[Mitchell, M. 1996] Mitchell, M., *An Introduction to Genetic Algorithms*, Cambridge, MA: MIT Press, 1996.

[Mitchell, T. 1997] Mitchell, T., *Machine Learning*, New York: McGraw-Hill, 1997.

[Monahan 1982] Monahan, G., "A Survey of Partially Observable Markov Decision Processes: Theory, Models, and Algorithms," *Management Science*, 28:1–16, January 1982.

[Montague, et al. 1995] Montague, P., Dayan, P., Person, C., and Sejnowski, T., "Bee Foraging in Uncertain Environments Using Predictive Hebbian Learning," *Nature* 377:725–728, 1995.

[Moore 1959] Moore, E. F., "The Shortest Path through a Maze," in *Proceedings of an International Symposium on the Theory of Switching, Part II*, pp. 285–292, Cambridge, MA: Harvard University Press, 1959.

[Moore 1985a] Moore, R., "Semantical Considerations on Nonmonotonic Logic," *Artificial Intelligence*, 25(1):75–94, 1985.

[Moore 1985b] Moore, R., "A Formal Theory of Knowledge and Action," in Hobbs, J., and Moore, R. (eds.), *Formal Theories of the Commonsense World*, pp. 319–358, Norwood, NJ: Ablex, 1985.

[Moore 1993] Moore, R., "Autoepistemic Logic Revisited," *Artificial Intelligence*, 59(1–2):27–30, 1993.

[Moore & Atkeson 1993] Moore, A., and Atkeson, C., "Prioritized Sweeping–Reinforcement Learning with Less Data and Less Time," *Machine Learning*, 13:103–130, 1993.

[Mostow & Prieditis 1989] Mostow, J., and Prieditis, A., "Discovering Admissible Heuristics by Abstracting and Optimizing: A Transformational Approach," in *Proceedings of the Eleventh International Joint Conference on Artificial Intelligence (IJCAI-89)*, pp. 701–707, San Francisco: Morgan Kaufmann, 1989.

[Muggleton 1992] Muggleton, S., *Inductive Logic Programming*, New York: Academic Press, 1992.

[Muggleton & Buntine 1988] Muggleton, S., and Buntine, W., "Machine Invention of First-Order Predicates by Inverting Resolution," in Laird, J. (ed.), in *Proceedings of the Fifth International Conference on Machine Learning*, pp. 339ff, San Francisco: Morgan Kaufmann, 1988.

[Muggleton, King, & Sternberg 1992] Muggleton, S., King, R., and Sternberg, J., "Protein Secondary Structure Prediction Using Logic-Based Machine Learning," *Protein Engineering*, 5(7):647–657, 1992.

[Muggleton & De Raedt 1994] Muggleton, S., and De Raedt, L., "Inductive Logic Programming: Theory and Methods," *Journal of Logic Programming*, 19,20:629–679, 1994.

[Myers 1994] Myers, K., "Hybrid Reasoning Using Universal Attachment," *Artificial Intelligence*, (67)2:329–375, 1994.

[Nalwa 1993] Nalwa, V. S., *A Guided Tour of Computer Vision*, Reading, MA: Addison-Wesley, 1993.

[Nalwa & Binford 1986] Nalwa, V., and Binford, T., "On Detecting Edges," *IEEE Trans on Pattern Analysis and Machine Intelligence*, PAMI-8(6):699–714, November 1986.

[Nalwa & Pauchon 1987] Nalwa, V., and Pauchon, E., "Edgel Aggregation and Edge Description," *Computer Vision, Graphics, and Image Processing*, 40:79–94, 1987.

[Neal 1991] Neal, R., "Connectionist Learning of Belief Networks," *Artificial Intelligence*, 56:71–113, 1991.

[Neapolitan 1990] Neapolitan, R., *Probabilistic Reasoning in Expert Systems: Theory and Algorithms*, New York: John Wiley & Sons, 1990.

[Newborn 1996] Newborn, M., *Computer Chess Comes of Age*, New York: Springer-Verlag, 1996.

[Newell 1964] Newell, A., "The Possibility of Planning Languages in Man-Computer Communication," in *Communication Processes*, Proceedings of a Symposium held in Washington, 1963, New York: Pergamon, 1964.

[Newell 1973] Newell, A., "Production Systems: Models of Control Structures," in Chase, W. (ed.), *Visual Information Processing*, New York: Academic Press, 1973.

[Newell 1982] Newell, A., "The Knowledge Level," *Artificial Intelligence*, 18(1):87–127, 1982.

[Newell 1991] Newell, A., *Unified Theories of Cognition*, Cambridge, MA: Harvard University Press, 1991.

[Newell, Shaw, & Simon 1957] Newell, A., Shaw, J., and Simon, H., "Empirical Explorations of the Logic Theory Machine," *Proc. West. Joint Computer Conf.*, vol. 15, pp. 218–239, 1957. (Reprinted in Feigenbaum, E., and Feldman, J. (eds.), *Computers and Thought*, pp. 109–133, New York: McGraw-Hill, 1963.)

[Newell, Shaw, & Simon 1958] Newell, A., Shaw, J. C., and Simon, H. A., "Chess-Playing Programs and the Problem of Complexity," *IBM Jour. R & D*, 2:320–355, 1958. (Reprinted in Feigenbaum, E., and Feldman, J. (eds.), *Computers and Thought*, pp. 109–133, New York: McGraw-Hill, 1963.)

[Newell, Shaw, & Simon 1959] Newell, A., Shaw, J. C., and Simon, H. A., "Report on a General Problem-Solving Program for a Computer," *Computers and Automation*, 8(7):10–16, 1959.

[Newell & Simon 1963] Newell, A., and Simon, H., "GPS, A Program That Simulates Human Thought," in Feigenbaum, E., and Feldman, J. (eds.) *Computers and Thought*, pp. 279–293, New York: McGraw-Hill, 1963.

[Newell & Simon 1972] Newell, A., and Simon, H., *Human Problem Solving*, Englewood Cliffs, NJ: Prentice Hall, 1972.

[Newell & Simon 1976] Newell, A., and Simon, H. A., "Computer Science as Empirical Inquiry: Symbols and Search," *Communications of the Association for Computing Machinery*, 19(3):113–126, 1976.

[Nii 1986a] Nii, H. P., "Blackboard Systems: The Blackboard Model of Problem Solving and the Evolution of Blackboard Architectures," *The AI Magazine*, 7(2):38–64, Summer

1986.

[Nii 1986b] Nii, H. P., "Blackboard Systems (Part Two): Blackboard Application Systems, Blackboard Systems from a Knowledge Engineering Perspective," *The AI Magazine*, 7(3):82–106, 1986.

[Nilsson 1965] Nilsson, N., *The Mathematical Foundations of Learning Machines*, San Francisco: Morgan Kaufmann, 1990. (This book is a reprint of Nilsson, N., *Learning Machines: Foundations of Trainable Pattern-Classifying Systems*, New York: McGraw-Hill, 1965.)

[Nilsson 1969] Nilsson, N., "Searching Problem-Solving and Game-Playing Trees for Minimal Cost Solutions," in Morrell, A. (ed.), *Information Processing 68*, vol. 2, pp. 1556–1562, Amsterdam: North-Holland, 1969.

[Nilsson 1980] Nilsson, N., *Principles of Artificial Intelligence*, San Francisco: Morgan Kaufmann, 1980.

[Nilsson 1984a] Nilsson, N., "Artificial Intelligence, Employment, and Income," *The AI Magazine*, 5(2):5–14, Summer 1984.

[Nilsson 1984b] Nilsson, N., *Shakey the Robot*, Technical Note 323, SRI International, Menlo Park, CA, 1984.

[Nilsson 1986] Nilsson, N., "Probabilistic Logic," *Artificial Intelligence*, 28(1):71–87, 1986.

[Nilsson 1991] Nilsson, N., "Logic and Artificial Intelligence," *Artificial Intelligence*, 47:31–56, 1991.

[Nilsson 1994] Nilsson, N., "Teleo-Reactive Programs for Agent Control," *Journal of Artificial Intelligence Research*, 1, pp. 139–158, January 1994.

[Nilsson & Raphael 1967] Nilsson, N., and Raphael, B., "Preliminary Design of an Intelligent Robot," in Tou, J. T. (ed.), *Computer and Information Sciences-II*, pp. 235–259, New York: Academic Press, 1967.

[Norvig 1992] Norvig, P., *Paradigms of Artificial Intelligence Programming: Case Studies in Common LISP*, San Francisco: Morgan Kaufmann, 1992.

[Nourbakhsh 1997] Nourbakhsh, I., *Interleaving Planning and Execution for Autonomous Robots*, Boston: Kluwer Academic Publishers, 1997.

[O'Reilly & Oppacher 1994] O'Reilly, U.-M., and Oppacher, F., "Program Search with a Hierarchical Variable Length Representation: Genetic Programming, Simulated Annealing and Hill Climbing," in Davidor, Y., Schwefel, H., and Manner, R. (eds.), *Lecture Notes in Computer Science, No. 866*, Berlin: Springer-Verlag, 1994.

[Paterson & Wegman 1978] Paterson, M. and Wegman, M., "Linear Unification," *Journal of Computer and System Science*, 16:158–167, 1978.

[Pearl 1982a] Pearl, J., "Reverend Bayes on Inference Engines: A Distributed Hierarchical Approach," in *Proceedings of the Second National Conference on Artificial Intelligence (AAAI-82)*, pp. 133–136, Menlo Park, CA: AAAI Press, 1982.

[Pearl 1982b] Pearl, J., "A Solution for the Branching Factor of the Alpha-Beta Pruning Algorithm and its Optimality," *Comm. ACM*, 25(8):559–564, 1982.

[Pearl 1984] Pearl, J., *Heuristics: Intelligent Search Strategies for Computer Problem Solving*, Reading, MA: Addison-Wesley, 1984.

[Pearl 1986] Pearl, J., "Fusion, Propagation, and Structuring in Belief Networks," *Artificial Intelligence*, 29:241–288, 1986.

[Pearl 1988] Pearl, J., *Probabilistic Reasoning in Intelligent Systems: Networks of Plausible Inference*, San

Francisco: Morgan Kaufmann, 1988.

[Pearl 1990] Pearl, J., "Reasoning under Uncertainty," *Annual Review of Computer Science*, vol. 4, 1989–1990, pp. 37–72, Palo Alto, CA: Annual Reviews, 1990.

[Pednault 1986] Pednault, E., "Formulating Multiagent, Dynamic-World Problems in the Classical Planning Framework," in Georgeff, M., and Lansky, A. (eds.), *Reasoning about Actions and Plans: Proceedings of the 1986 Workshop*, Timberline, Oregon, pp. 47–82, San Francisco: Morgan Kaufmann, 1986. (Also in Allen, J., Hendler, J., and Tate, A. (eds.), *Readings in Planning*, pp. 675–710, San Francisco: Morgan Kaufmann, 1990.)

[Pednault 1989] Pednault, E., "ADL: Exploring the Middle Ground between STRIPS and the Situation Calculus," in Brachman, R., Levesque, H., and Reiter, R. (eds.), *Proceedings of the First International Conference on Principles of Knowledge Representation and Reasoning (KR-89)*, pp. 324–332, San Francisco: Morgan Kaufmann, 1989.

[Penberthy & Weld 1992] Penberthy, J., and Weld, D., "UCPOP: A Sound, Complete Partial-Order Planner for ADL," in *Proceedings of the Third International Conference on Principles of Knowledge Representation and Reasoning (KR-92)*, pp. 103–113, San Francisco: Morgan Kaufmann, 1992.

[Penrose 1989] Penrose, R., *The Emperor's New Mind: Concerning Computers, Minds, and the Laws of Physics*, Oxford: Oxford University Press, 1989.

[Penrose 1994] Penrose, R., *Shadows of the Mind: Search for the Missing Science of Consciousness*, Oxford: Oxford University Press, 1994.

[Pereira & Shieber 1987] Pereira, F., and Shieber, S., *PROLOG and Natural Language Analysis*, CSLI Lecture Notes, No. 10, Center for the Study of Language and Information, Stanford University, Stanford, CA, 1987.

[Pereira & Warren 1980] Pereira, F., and Warren, D., "Definite Clause Grammars for Language Analysis: A Survey of the Formalism and a Comparison with Augmented Transition Networks," *Artificial Intelligence*, 13:231–278, 1980.

[Perlis 1985] Perlis, D., "Languages with Self-Reference, I: Foundations," *Artificial Intelligence*, 25:301–332, 1985.

[Perlis 1988] Perlis, D., "Languages with Self-Reference, II: Knowledge, Belief, and Modality," *Artificial Intelligence*, 34:179–212, 1988.

[Perona & Malik 1990] Perona, P., and Malik, J., "Scale-Space and Edge Detection Using Anisotropic Diffusion," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, PAMI-12:7:629–639, July 1990.

[Pineda 1987] Pineda, F. J., "Generalization of Back-Propagation to Recurrent Neural Networks," *Physical Review Letters*, 59:2229–2232, 1987.

[Pingle 1969] Pingle, K., "Visual Perception by a Computer," in *Automatic Interpretation and Classification of Images*, Grasselli, A. (ed.), pp. 277–284, New York: Academic Press, 1969.

[Pohl 1971] Pohl, I., "Bi-directional Search," in *Machine Intelligence 6*, Meltzer, B., and Michie, D. (eds.), pp. 127–140, Edinburgh: Edinburgh University Press, 1971.

[Pohl 1973] Pohl, I., "The Avoidance of (Relative) Catastrophe, Heuristic Competence, Genuine Dynamic Weighting and Computational Issues in Heuristic Problem Solving," in *Proceedings of the Third International Joint Conference on Artificial Intelligence (IJCAI-73)*,

pp. 20–23, San Francisco: Morgan Kaufmann, 1973.

[Pollack & Ringuette 1990] Pollack, M. E., and Ringuette, M., "Introducing the Tileworld: Experimentally Evaluating Agent Architectures," in *Proceedings of the Eighth National Conference on Artificial Intelligence (AAAI-90)*, pp. 183–189, Menlo Park, CA: AAAI Press, 1990.

[Pomerleau 1991] Pomerleau, D., "Rapidly Adapting Artificial Neural Networks for Autonomous Navigation," in Lippmann, P., et al. (eds.), *Advances in Neural Information Processing Systems, 3*, pp. 429–435, San Francisco: Morgan Kaufmann, 1991.

[Pomerleau 1993] Pomerleau, D., *Neural Network Perception for Mobile Robot Guidance*, Boston: Kluwer Academic Publishers, 1993.

[Port & van Gelder 1995] Port, R., and van Gelder, T., *Mind as Motion: Explorations in the Dynamics of Cognition*, Cambridge, MA: Bradford Books/MIT Press, 1995.

[Pospesel 1976] Pospesel, H., *Introduction to Logic: Predicate Logic*, Englewood Cliffs, NJ: Prentice Hall, 1976.

[Powers 1995] Powers, R., *Galatea 2.2*, New York: Farrar, Straus & Giroux, 1995.

[Powley, Ferguson, & Korf 1993] Powley, C., Ferguson, C., and Korf, R., "Depth-First Heuristic Search on a SIMD Machine," *Artificial Intelligence*, 60:199–242, 1993.

[Pradhan, et al. 1994] Pradhan, M., Provan, G., Middleton, B., and Henrion, M., "Knowledge Engineering for Large Belief Networks," in *Proceedings of the Tenth Conference on Uncertainty in Artificial Intelligence*, pp. 484–490, San Francisco: Morgan Kaufmann, 1994.

[Prawitz 1965] Prawitz, D:, *Natural Deduction: A Proof Theoretical Study*, Stockholm: Almquist and Wiksell, 1965.

[Prieditis 1993] Prieditis, A, "Machine Discovery of Effective Admissible Heuristics," *Machine Learning*, 12(1–3):117–141, 1993.

[Purdom & Brown 1987] Purdom, P. W., Jr., and Brown, C., "Polynomial Average-Time Satisfiability Problems," *Information Science*, 41:23–42, 1987.

[Puterman 1994] Puterman, M., *Markov Decision Processes—Discrete Stochastic Dynamic Programming*, New York: John Wiley & Sons, 1994.

[Quinlan 1979] Quinlan, J., "Discovering Rules from Large Collections of Examples: A Case Study," in Michie, D. (ed.), *Expert Systems in the Microelectronic Age*, Edinburgh: Edinburgh University Press, 1979.

[Quinlan 1990] Quinlan, J. R., "Learning Logical Definitions from Relations," *Machine Learning*, 5(3):239–266, 1990.

[Quinlan 1993] Quinlan, J. R., *C4.5: Programs for Machine Learning*, San Francisco: Morgan Kaufmann, 1993.

[Ramadge & Wonham 1989] Ramadge, P., and Wonham, M., "The Control of Discrete Event Systems," *Proc. of the IEEE*, 77(1):81–93, 1989.

[Reichardt 1965] Reichardt, W., "On the Theory of Lateral Nervous Inhibition in the Complex Eye of Limulus," *Progress in Brain Research*, 17:64–73, 1965

[Reiter 1980] Reiter, R., "A Logic for Default Reasoning," *Artificial Intelligence*, 13(1–2):81–132, 1980.

[Reiter 1991] Reiter, R., "The Frame Problem in the Situation Calculus: A Simple Solution

(Sometimes) and a Completeness Result for Goal Regression," in Lifschitz, V. (ed.), *Artificial Intelligence and Mathematical Theory of Computation: Papers in Honor of John McCarthy*, pp. 359–380, New York: Academic Press, 1991.

[Reitman & Wilcox 1979] Reitman, W., and Wilcox, B., "The Structure and Performance of the INTERIM.2 Go Program," in *Proceedings of the Sixth International Joint Conference on Artificial Intelligence (IJCAI-79)*, pp. 711–719, San Francisco: Morgan Kaufmann, 1979.

[Resnick 1993] Resnick, M., "Behavior Construction Kits," *Communications of the ACM*, 36(7):64–71, July 1993.

[Rich & Knight 1991] Rich, E., and Knight, K., *Artificial Intelligence* (second edition), New York: McGraw-Hill, 1991.

[Rissanen 1984] Rissanen, J., "Universal Coding, Information, Prediction, and Estimation," *IEEE Transactions on Information Theory*, IT-30(4):629–636, 1984.

[Rivest 1987] Rivest, R., "Learning Decision Lists," *Machine Learning*, 2, 229–246, 1987.

[Rivest & Schapire 1993] Rivest, R., and Schapire, R., "Inference of Finite Automata Using Homing Sequences," *Information and Computation*, 103(2):299–347, 1993.

[Roberts 1963] Roberts, L., *Machine Perception of Three-Dimensional Solids*, Tech. Report 315, MIT Lincoln Laboratory, Ph.D. dissertation, Massachusetts Institute of Technology, 1963.

[Robinson 1965] Robinson, J. A., "A Machine-Oriented Logic Based on the Resolution Principle," *Journal of the Association for Computing Machinery*, 12(1):23–41, 1965.

[Rohwer & Forrest 1987] Rohwer, R., and Forrest, B., "Training Time-Dependence in Neural Networks," in Caudill, M., and Butler, C. (eds.), *IEEE First International Conference on Neural Networks*, San Diego, 1987, vol. II, pp. 701–708, New York: IEEE, 1987.

[Rosenblatt 1962] Rosenblatt, F., *Principles of Neurodynamics*, Washington, DC: Spartan Books, 1962.

[Rosenschein & Kaelbling 1995] Rosenschein, S., and Kaelbling, L., "A Situated View of Representation and Control," *Artificial Intelligence*, 73:149–173, 1995.

[Rosenbloom, Laird, & Newell 1993] Rosenbloom, P., Laird, J., and Newell, A., (eds.), *The Soar Papers: Research on Integrated Intelligence*, vols. 1 and 2, Cambridge, MA: MIT Press, 1993.

[Ross 1988] Ross, S., *A First Course in Probability*, third edition, London: Macmillan, 1988.

[Roussel 1975] Roussel, P., "PROLOG: Manual de Reference et d'Utilization," Technical Report, Université Aix-Marseille II, Groupe d'Intelligence Artificielle, France, 1975.

[Rumelhart, et al. 1986] Rumelhart, D. E., Hinton, G. E., and Williams, R. J., "Learning Internal Representations by Error Propagation," in Rumelhart, D. E., and McClelland, J. L. (eds.), *Parallel Distributed Processing*, Vol 1., pp. 318–362, 1986.

[Russell 1997] Russell, S., "Rationality and Intelligence," *Artificial Intelligence*, (94)1:57–77, 1997.

[Russell & Norvig 1995] Russell, S., and Norvig, P., *Artificial Intelligence: A Modern Approach*, Englewood Cliffs, NJ: Prentice Hall, 1995. (Revised edition to appear in 1998.)

[Russell & Wefald 1991] Russell, S., and Wefald, E., *Do the Right Thing*, Cambridge, MA: MIT Press, 1991.

[Sacerdoti 1974] Sacerdoti, E., "Planning in a Hierarchy of Abstraction Spaces," *Artificial Intelligence*,

5(2):115–135, 1974. (Reprinted in Allen, J., Hendler, J., and Tate, A. (eds.), *Readings in Planning*, pp. 98–108, San Francisco: Morgan Kaufmann, 1990.)

[Sacerdoti 1975] Sacerdoti, E., "The Non-linear Nature of Plans," in *Proceedings of the Fourth International Joint Conference on Artificial Intelligence (IJCAI-75)*, pp. 206–214, San Francisco: Morgan Kaufmann, 1975. (Reprinted in Allen, J., Hendler, J., and Tate, A. (eds.), *Readings in Planning*, pp. 162–170, San Francisco: Morgan Kaufmann, 1990.)

[Sacerdoti 1977] Sacerdoti, E., *A Structure for Plans and Behavior*, New York: American Elsevier, 1977.

[Sadek, et al. 1996] Sadek, M., Ferrieux, A., Cozannet, A., Bretier, P., Panaget, F., and Simonin, J., "Effective Human-Computer Cooperative Spoken Dialogue: The AGS Demonstrator," in *Proc. ICSLP'96 International Conf. on Spoken Language Processing*, pp. 546–549, Philadelphia, PA, October 3–6, 1996.

[Samuel 1959] Samuel, A. L., "Some Studies in Machine Learning Using the Game of Checkers," *IBM Jour. R & D*, 3:211–229, 1959. (Reprinted in Feigenbaum, E., and Feldman, J. (eds.), *Computers and Thought*, pp. 71–105, New York: McGraw-Hill, 1963.)

[Samuel 1967] Samuel, A. L., "Some Studies in Machine Learning Using the Game of Checkers II—Recent Progress," *IBM Jour. R & D*, 11(6), 601–617, 1967.

[Schaeffer 1997] Schaeffer, J., *One Jump Ahead: Challenging Human Supremacy in Checkers*, New York: Springer-Verlag, 1997.

[Schaeffer, et al. 1992] Schaeffer, J., Culberson, J., Treloar, N., Knight, B., Lu, P., and Szafron, D., "A World Championship Caliber Checkers Program," *Artificial Intelligence*, 53(2–3):273–289, 1992.

[Schaeffer & Lake 1996] Schaeffer, J. and Lake, R., "Solving the Game of Checkers," in Nowakowski, R. J. (ed.), *Games of No Chance*, pp. 119–133, Cambridge: Cambridge University Press, 1996.

[Scherl & Levesque 1993] Scherl, R., and Levesque, H., "The Frame Problem and Knowledge Producing Actions," in *Proceedings of the Eleventh National Conference on Artificial Intelligence (AAAI-93)*, pp. 689–695, Menlo Park, CA: AAAI Press, 1993.

[Schoppers 1987] Schoppers, M. J., "Universal Plans for Reactive Robots in Unpredictable Domains," in *Proceedings of the Tenth International Joint Conference on Artificial Intelligence (IJCAI-87)*, pp. 1039–1046, San Francisco: Morgan Kaufmann, 1987.

[Schraudolph, Dayan, & Sejnowski 1994] Schraudolph, N., Dayan, P., and Sejnowski, T., "Temporal Difference Position Evaluation in the Game of GO," in Cowan, J., et al. (eds.), *Advances in Neural Information Processing Systems, 6*, pp. 817–824, San Francisco: Morgan Kaufmann, 1994.

[Schubert 1990] Schubert, L., "Monotonic Solution of the Frame Problem in the Situation Calculus: An Efficient Method for Worlds with Fully Specified Actions," in Kyberg, H., Loui, R., and Carlson, G. (eds.), *Knowledge Representation and Defeasible Reasoning*, pp. 23–67, Boston: Kluwer Academic Publishers, 1990.

[Schultz, Dayan, & Montague 1997] Schultz, W., Dayan, P., and Montague, P. R., "A Neural Substrate for Prediction and Reward," *Science*, 275:1593–1599, March 14, 1997.

[Schwartz 1987] Schwartz, J., "The Limits of Artificial Intelligence," in the *Encyclopedia of Artificial*

Intelligence, New York: John Wiley & Sons, 1987.

[Searle 1969] Searle, J., *Speech Acts: An Essay in the Philosophy of Language*, Cambridge: Cambridge University Press, 1969.

[Searle 1980] Searle, J., "Minds, Brains, and Programs," *The Behavioral and Brain Sciences*, 3:417–457, 1980 (with open peer commentary). (Reprinted in Hofstadter, D. R., and Dennett, D. C. (eds.), *The Mind's Eye: Fantasies and Reflections on Self and Soul*, pp. 351–373, New York: Basic Books, 1981.)

[Searle 1992] Searle, J., *The Rediscovery of the Mind*, Cambridge, MA: MIT Press, 1992.

[Sejnowski & Rosenberg 1987] Sejnowski, T., and Rosenberg, C., "Parallel Networks That Learn to Pronounce English Text," *Complex Systems*, 1:145–168, 1987.

[Selman & Kautz 1991] Selman, B., and Kautz, H., "Knowledge Compilation Using Horn Approximations," in *Proceedings of the Ninth National Conference on Artificial Intelligence (AAAI-91)*, pp. 904–909, Menlo Park, CA: AAAI Press, 1991.

[Selman & Kautz 1993] Selman, B., and Kautz, H., "An Empirical Study of Greedy Local Search for Satisfiability Testing," in *Proceedings of the Eleventh National Conference on Artificial Intelligence (AAAI-93)*, pp. 46–51, Menlo Park, CA: AAAI Press, 1993.

[Selman, Kautz, & Cohen 1994] Selman, B., Kautz, H., and Cohen, B., "Noise Strategies for Improving Local Search," in *Proceedings of the Twelfth National Conference on Artificial Intelligence (AAAI-94)*, pp. 337–343, Menlo Park, CA: AAAI Press, 1994.

[Selman, Kautz, & Cohen 1996] Selman, B., Kautz, H., and Cohen, B., "Local Search Strategies for Satisfiability Testing," in Du, D., Gu, J., and Pardalos, P. (eds.), *Satisfiability Problem: Theory and Applications*, Vol. 35, Dimacs Series in Discrete Mathematics and Theoretical Computer Science, Providence, RI: American Mathematical Society, 1996.

[Selman & Levesque 1990] Selman, B., and Levesque, H., "Abductive and Default Reasoning: A Computational Core," in *Proceedings of the Eighth National Conference on Artificial Intelligence (AAAI-90)*, pp. 343–348, Menlo Park, CA: AAAI Press, 1990.

[Selman, Levesque, & Mitchell 1992] Selman, B., Levesque, H., and Mitchell, D., "A New Method for Solving Hard Satisfiability Problems," in *Proceedings of the Tenth National Conference on Artificial Intelligence (AAAI-92)*, pp. 440–446, Menlo Park, CA: AAAI Press, 1992.

[Shachter & Kenley 1989] Shachter, R., and Kenley, C., "Gaussian Influence Diagrams," *Management Science*, 35:527–550, 1989.

[Shafer 1979] Shafer, G., *A Mathematical Theory of Evidence*, Princeton, NJ: Princeton University Press, 1979.

[Shafer & Pearl 1990] Shafer, G., and Pearl, J. (eds.), *Readings in Uncertain Reasoning*, San Francisco: Morgan Kaufmann, 1990.

[Shanahan 1997] Shanahan, M., *Solving the Frame Problem: A Mathematical Investigation of the Commonsense Law of Inertia*, Cambridge, MA: MIT Press, 1997.

[Shannon 1950] Shannon, C., "Programming a Computer for Playing Chess,"*Philosophical Magazine* (Series 7), vol. 41, pp. 256–275, 1950.

[Shannon & McCarthy 1956] Shannon, C., and McCarthy, J. (eds.), *Automata Studies, Annals of*

Mathematical Studies, 34, Princeton, NJ: Princeton University Press, 1956.

[Shapiro 1992] Shapiro, S., *Encyclopedia of Artificial Intelligence*, Vols. 1 and 2 (second edition), New York: John Wiley & Sons, 1992.

[Shavlik & Dietterich 1990] Shavlik, J., and Dietterich, T. (eds.), *Readings in Machine Learning*, San Francisco: Morgan Kaufmann, 1990.

[Shen 1994] Shen, W.-M., *Autonomous Learning from the Environment*, San Francisco: W. H. Freeman, 1994.

[Shirai 1987] Shirai, Y., *Three-Dimensional Computer Vision*, Berlin: Springer-Verlag, 1987.

[Shoham 1987] Shoham, Y., "Temporal Logics in AI: Semantical and Ontological Considerations," *Artificial Intelligence*, 33(1):89–104, 1987.

[Shoham 1988] Shoham, Y., *Reasoning about Change: Time and Causation from the Standpoint of Artificial Intelligence*, Cambridge, MA: MIT Press, 1988.

[Shoham 1993] Shoham, Y., "Agent-Oriented Programming," *Artificial Intelligence*, 60:51–92, 1993.

[Shoham 1994] Shoham, Y., *AI Programming in PROLOG*, San Francisco: Morgan Kaufmann, 1994.

[Shoham 1996] Shoham, Y., "The Open Scientific Borders of AI, and the Case of Economics," *ACM Computing Surveys*, 28(4):11ff, December 1996.

[Shortliffe 1976] Shortliffe, E. H., *Computer-Based Medical Consultations: MYCIN*, New York: Elsevier, 1976.

[Shrobe 1988] Shrobe, H. (ed.), *Exploring Artificial Intelligence: Survey Talks from the National Conference on Artificial Intelligence*, San Francisco: Morgan Kaufmann, 1988.

[Skolem 1920] Skolem, T., "Logisch-kombinatorische Untersuchungen über die Erfüllbarkeit oder Beweisbarkeit Mathematischer Sätze Nebst einem Theoreme über die Dichte Mengen," *Videnskapsselskapets Skrifter, I, Matematisk-naturvidenskabelig Klasse*, 4, 1920.

[Slate & Atkin 1977] Slate, D., and Atkin, L., "Chess 4.5—The Northwestern University Chess Program," in Fry, P. (ed.), *Chess Skill in Man and Machine*, pp. 82–118, New York: Springer-Verlag, 1977.

[Slagle 1963] Slagle, J. R., "A Heuristic Program That Solves Symbolic Integration Problems in Freshman Calculus," *Jour. Assoc. of Comp. Mach.*, 10:507–520, 1963. (Also in Feigenbaum, E., and Feldman, J. (eds.), *Computers and Thought*, pp. 191–203, New York: McGraw-Hill, 1963.)

[Slagle & Dixon 1969] Slagle, J., and Dixon J., "Experiments with Some Programs that Search Game Trees," *Jour. Assoc. Comp. Mach.*, 16:2:189–207, 1969.

[Smith 1989] Smith, D. E., "Controlling Backward Inference," *Artificial Intelligence*, 39(2):145–208, 1989.

[Smith, Genesereth, & Ginsberg 1986] Smith, D. E., Genesereth, M., and Ginsberg, M., "Controlling Recursive Inference," *Artificial Intelligence*, 30(3):343–389, 1986.

[Soderland & Weld 1991] Soderland, S., and Weld, D., "Evaluating Nonlinear Planning," Technical Report TR-91-02-03, University of Washington Department of Computer Science and Engineering, Seattle, WA, 1991.

[Sowa 1991] Sowa, J. (ed.), *Principles of Semantic Networks*, San Francisco: Morgan Kaufmann, 1991.

[Spirtes & Meek 1995] Spirtes, P., and Meek, C., "Learning Bayesian Networks with Discrete

Variables from Data," in *Proceedings of First International Conference on Knowledge Discovery and Data Mining*, San Francisco: Morgan Kaufmann, 1995.

[Stallman & Sussman 1977] Stallman, R., and Sussman, G., "Forward Reasoning and Dependency-Directed Backtracking in a System for Computer-Aided Circuit Analysis," *Artificial Intelligence*, 9(2):135–196, 1977.

[Stanfill & Waltz 1986] Stanfill, C., and Waltz, D., "Toward Memory-Based Reasoning," *Communications of the ACM*, 29(12):1213–1228, 1986.

[Stefik 1995] Stefik, M., *Introduction to Knowledge Systems*, San Francisco: Morgan Kaufmann, 1995.

[Stentz 1995] Stentz, A., "The Focussed D* Algorithm for Real-Time Replanning," in *Proceedings of the Fourteenth International Joint Conference on Artificial Intelligence (IJCAI-95)*, pp. 1652–1659, San Francisco: Morgan Kaufmann, 1995.

[Stentz & Hebert 1995] Stentz, A., and Hebert, M., "A Complete Navigation System for Goal Acquisition in Unknown Environments," *Autonomous Robots*, 2(2):127–147, 1995.

[Sterling & Shapiro 1986] Sterling, L., and Shapiro, E., *The Art of PROLOG*, Cambridge, MA: MIT Press, 1986.

[Stickel 1985] Stickel, M., "Automated Deduction by Theory Resolution," *Journal of Automated Reasoning*, 1(4):333–355, 1985.

[Stickel 1988] Stickel, M., "A PROLOG Technology Theorem Prover: Implementation by an Extended PROLOG Compiler," *Journal of Automated Reasoning*, 4:353–380, 1988.

[Stickel & Tyson 1985] Stickel, M., and Tyson, M., "An Analysis of Consecutively Bounded Depth-First Search with Applications in Automated Deduction," in *Proceedings of the Ninth International Joint Conference on Artificial Intelligence (IJCAI-85)*, pp. 1073–1075, San Francisco: Morgan Kaufmann, 1985.

[Strat 1992] Strat, T., *Natural Object Recognition*, Berlin: Springer-Verlag, 1992.

[Sussman 1975] Sussman, G., *A Computer Model of Skill Acquisition*, Amsterdam: Elsevier/North-Holland, 1975.

[Sutton 1988] Sutton, R., "Learning to Predict by the Methods of Temporal Differences," *Machine Learning*, 3:9–44, 1988.

[Sutton 1990] Sutton, R., "Integrated Architectures for Learning, Planning, and Reacting Based on Approximating Dynamic Programming," in *Proceedings of the Seventh International Conference on Machine Learning*, pp. 216–224, San Francisco: Morgan Kaufmann, 1990.

[Tarski 1935] Tarski, A., "Die Wahrheitsbegriff in den Formalisierten Sprachen," *Studia Philosophica*, 1:261–405, 1935.

[Tarski 1956] Tarski, A., *Logic, Semantics, Metamathematics: Papers from 1923 to 1938*, Oxford: Oxford University Press, 1956.

[Tate 1977] Tate, A., "Generating Project Networks," in *Proceedings of the Fifth International Joint Conference on Artificial Intelligence (IJCAI-77)*, pp. 888–893, San Francisco: Morgan Kaufmann, 1977. (Reprinted in Allen, J., Hendler, J., and Tate, A. (eds.), *Readings in Planning*, pp. 291–296, San Francisco: Morgan Kaufmann, 1990.)

[Tate 1996] Tate, A. (ed.), *Advanced Planning Technology, The Technological Achievements of the ARPA/Rome Laboratory Planning Initiative*, Menlo Park, CA: AAAI Press, May 1996.

[Teller 1994] Teller, A., "The Evolution of Mental Models," in Kinnear, K., Jr. (ed.), *Advances in Genetic Programming*, Ch. 9, Cambridge, MA: MIT Press, 1994.

[Tenenberg 1991] Tenenberg, J., "Abstraction in Planning," in Allen, J., Kautz, H., Pelavin, R., and Tenenberg, J. (eds.), *Reasoning about Plans*, Ch. 4, San Francisco: Morgan Kaufmann, 1991.

[Tesauro 1992] Tesauro, G., "Practical Issues in Temporal Difference Learning," *Machine Learning*, 8, nos. 3/4:257–277, 1992.

[Tesauro 1995] Tesauro, G., "Temporal-Difference Learning and TD-Gammon," *Comm. ACM*, 38(3):58–68, March 1995.

[Theraulaz & Bonabeau 1995] Theraulaz, G., and Bonabeau, E., "Coordination in Distributed Building," *Science*, 269:686–688, August 4, 1995.

[Thorpe, et al. 1992] Thorpe, C., Hebert, M., Kanade, T., and Shafer, S., "The New Generation System for the CMU Navlab," in Masaki, I. (ed.), *Vision-Based Vehicle Guidance*, pp. 30–82, Berlin: Springer-Verlag, 1992.

[Towell & Shavlik 1992] Towell, G., and Shavlik, J., "Interpretation of Artificial Neural Networks: Mapping Knowledge-Based Neural Networks into Rules," in Moody, J., Hanson, S., and Lippmann, R. (eds.), *Advances in Neural Information Processing Systems, 4*, pp. 977–984, San Francisco: Morgan Kaufmann, 1992.

[Towell, Shavlik, & Noordweier 1990] Towell, G., Shavlik, J., and Noordweier, M., "Refinement of Approximate Domain Theories by Knowledge-Based Artificial Neural Networks," in *Proceedings of the Eighth National Conference on Artificial Intelligence (AAAI-90)*, pp. 861–866, Menlo Park, CA: AAAI Press, 1990.

[Tracy & Bouthoorn 1997] Tracy, K., and Bouthoorn, P., *Object-Oriented Artificial Intelligence Using C++*, New York: Computer Science Press, 1997.

[Trappl 1986] Trappl, R., *Impacts of Artificial Intelligence: Scientific, Technological, Military, Economic, Societal, Cultural, and Political*, New York: North-Holland, 1986.

[Turing 1950] Turing, A. M., "Computing Machinery and Intelligence," *Mind*, 59:433–460, 1950. (Reprinted in Feigenbaum, E., and Feldman, J. (eds.), *Computers and Thought*, pp. 11–35, New York: McGraw-Hill, 1963.)

[Tversky & Kahneman 1982] Tversky, A., and Kahneman, D., "Causal Schemata in Judgements under Uncertainty," in Kahneman, D., Slovic, P., and Tversky, A. (eds.), *Judgements under Uncertainty: Heuristics and Biases*, Cambridge: Cambridge University Press, 1982.

[Ullman 1988] Ullman, J. D., *Principles of Database and Knowledge-Base Systems, Vol. I: Classical Database Systems*, New York: Computer Science Press, 1988.

[Ullman 1989] Ullman, J. D., *Principles of Database and Knowledge-Base Systems, Vol. II: The New Technologies*, New York: Computer Science Press, 1989.

[Unger 1989] Unger, S., *The Essence of Logic Circuits*, Englewood Cliffs, NJ: Prentice Hall, 1989.

[Venn 1880] Venn, J., "On the Diagrammatic and Mechanical Representation of Propositions and Reasonings," *Phil. Mag.*, pp. 123ff, 1880.

[Waibel & Lee 1990] Waibel, A., and Lee, K.-F. (eds.), *Readings in Speech Recognition*, San Francisco:

Morgan Kaufmann, 1990.

[Waibel, et al. 1988] Waibel, A., Hanazawa, T., Hinton, G., Shikano, K., and Lang, K., "Phoneme Recognition: Neural Networks versus Hidden Markov Models," in *Proc. of the International Conf. on Acoustics, Speech and Signal Processing*, New York, 1988.

[Waldinger 1975] Waldinger, R., "Achieving Several Goals Simultaneously," in Elcock, E., and Michie, D. (eds.), *Machine Intelligence 8*, pp. 94–138, Chichester, England: Ellis Horwood, 1975.

[Walter 1953] Walter, G., *The Living Brain*, New York: Norton and Company, 1953.

[Waltz 1975] Waltz, D., "Understanding Line Drawings of Scenes with Shadows," in Winston, P. (ed.), *The Psychology of Computer Vision*, pp. 19–91, New York: McGraw-Hill, 1975.

[Wang 1995] Wang, X., "Learning by Observation and Practice: An Incremental Approach for Planning Operator Acquisition," in *Proceedings of the Twelfth International Conference on Machine Learning*, pp. 549–557, San Francisco: Morgan Kaufmann, 1995.

[Warren, Pereira, & Pereira 1977] Warren, D., Pereira, L., and Pereira, F., "PROLOG: The Language and Its Implementation Compared with LISP," *SIGPLAN Notices*, 12(8):109–115, 1977.

[Weiss & Kulikowski 1991] Weiss, S., and Kulikowski, C., *Computer Systems That Learn*, San Francisco: Morgan Kaufmann, 1991.

[Weizenbaum 1965] Weizenbaum, J., "ELIZA—A Computer Program for the Study of Natural Language Communication between Man and Machine," *Communications of the Association for Computing Machinery*, 9(1):36–45, 1965.

[Weizenbaum 1976] Weizenbaum, J., *Computer Power and Human Reason: From Judgment to Calculation*, New York: W. H. Freeman, 1976.

[Weld 1994] Weld, D., "An Introduction to Least Commitment Planning," *AI Magazine*, 15(4):27–61, 1994.

[Weld & de Kleer 1990] Weld, D., and de Kleer, J., *Readings in Qualitative Reasoning about Physical Systems*, San Francisco: Morgan Kaufmann, 1990.

[Wellman 1990] Wellman, M., "Fundamental Concepts of Qualitative Probabilistic Networks," *Artificial Intelligence*, 44:257–303, 1990.

[Wellman 1996] Wellman, M., "Market-Oriented Programming: Some Early Lessons," in Clearwater, S. (ed.), *Market-Based Control—A Paradigm for Distributed Resource Allocation*, Singapore: World Scientific, 1996.

[Werbos 1974] Werbos, P., *Beyond Regression: New Tools for Prediction and Analysis in the Behavioral Sciences*, Ph.D. Thesis, Harvard University, 1974.

[Weyhrauch 1980] Weyhrauch, R., "Prolegomena to a Theory of Mechanized Formal Reasoning," *Artificial Intelligence*, 13(1–2):133–170, 1980.

[Whitehead, Karlsson, & Tenenberg 1993] Whitehead, S., Karlsson, J., and Tenenberg, J., "Learning Multiple Goal Behavior via Task Decomposition and Dynamic Policy Merging," in Connell, J., and Mahadevan, S. (eds.), *Robot Learning*, Ch. 3, Boston: Kluwer Academic Publishers, 1993.

[Widrow & Hoff 1960] Widrow, B., and Hoff, M. E., "Adaptive Switching Circuits," *1960 IRE*

WESCON Convention Record, pp. 96–104, New York, 1960.

[Widrow 1962] Widrow, B., "Generalization and Information Storage in Networks of Adaline 'Neurons,'" in Yovitz, M., Jacobi, G., and Goldstein, G. (eds.), *Self-Organizing Systems 1962*, pp. 435–461, Washington, DC: Spartan Books, 1962.

[Wiener 1948] Wiener, N., *Cybernetics: Control and Communication in the Animal and in the Machine*, New York: John Wiley & Sons, 1948.

[Wilkins 1988] Wilkins, D. E., *Practical Planning: Extending the Classical AI Planning Paradigm*, San Francisco: Morgan Kaufmann, 1988.

[Wilkins, et al. 1995] Wilkins, D. E., Myers, K., Lowrance, J., and Wesley, L., "Planning and Reacting in Uncertain and Dynamic Domains," *Journal of Experimental and Theoretical Artificial Intelligence*, 7(1):197–227, 1995.

[Wilson 1991] Wilson, S., "The Animat Path to AI," in Meyer, J. A., and Wilson, S. (eds.), *From Animals to Animats; Proceedings of the First International Conference on the Simulation of Adaptive Behavior*, Cambridge, MA: MIT Press/Bradford Books, 1991.

[Winograd 1972] Winograd, T., *Understanding Natural Language*, New York: Academic Press, 1972.

[Winograd & Flores 1986] Winograd, T., and Flores, F., *Understanding Computers and Cognition: A New Foundation for Design*, Norwood, NJ: Ablex, 1986. (Four reviews and a response appear in *Artificial Intelligence*, 31(2): 213–261, 1987.)

[Wooldridge 1968] Wooldridge, D., *Mechanical Man: The Physical Basis of Intelligent Life*, New York: McGraw-Hill, 1968.

[Woods 1970] Woods, W., "Transition Network Grammars for Natural Language Analysis," *Communications of the Association for Computing Machinery*, 13(10):591–606, 1970.

[Woods 1973] Woods, W., "Progress in Natural Language Understanding: An Application to Lunar Geology," in *AFIPS Conf. Proc.*, vol. 42, pp. 441–450, 1973.

[Wos 1993] Wos, L., "Automated Reasoning Answers Open Questions," *Notices of the AMS*, 5(1):15–26, January 1993.

[Wos, Carson, & Robinson 1965] Wos, L., Carson, D., and Robinson, G., "Efficiency and Completeness of the Set-of-Support Strategy in Theorem Proving," *Journal of the Association for Computing Machinery*, 12:536–541, 1965.

[Wos & Robinson 1968] Wos, L., and Robinson, G., "Paramodulation and Set of Support," in *Proc. of the IRIA Symposium on Automatic Demonstration*, pp. 276–310, Berlin: Springer-Verlag, 1968.

[Wos & Winker 1983] Wos, L., and Winker, S., "Open Questions Solved with the Assistance of AURA," in Bledsoe, W., and Loveland, D. (eds.), *Automated Theorem Proving: After 25 Years: Proceedings of the Special Session of the 89th Annual Meeting of the American Mathematical Society*, pp. 71–88, Denver, Colorado: American Mathematical Society, 1983.

[Wos, et al. 1992] Wos, L., Overbeek, R., Lusk, E., and Boyle, J., *Automated Reasoning: Introduction and Applications*, second edition, New York: McGraw-Hill, 1992.

[Zadeh 1975] Zadeh, L., "Fuzzy Logic and Approximate Reasoning," *Synthese* 30, pp. 407–428, 1975.

[Zadeh 1978] Zadeh, L., "Fuzzy Sets as a Basis for a Theory of Possibility," *Fuzzy Sets and Systems*, 1:3–28, 1978.

[Zhu, Wu, & Mumford 1998] Zhu, S. C., Wu, Y., and Mumford, D., "FRAME: Filters, Random Fields, and Maximum Entropy Towards a Unified Theory for Texture Modeling," *Int'l Journal of Computer Vision*, to appear.

[Zobrist 1970] Zobrist, A., *Feature Extraction and Representation for Pattern Recognition and the Game of Go*, Ph.D. Dissertation, University of Wisconsin, 1970.

[Zweben & Fox 1994] Zweben, M., and Fox, M. (eds.), *Intelligent Scheduling*, San Francisco: Morgan Kaufmann, 1994.