

出版说明

21世纪初的5至10年是我国国民经济和社会发展的关键时期,也是信息产业快速发展的关键时期。在我国加入WTO后的今天,培养一支适应国际化竞争的一流IT人才队伍是我国高等教育的重要任务之一。信息科学和技术方面人才的优劣与多寡,是我国面对国际竞争时成败的关键因素。

当前,正值我国高等教育特别是信息科学领域的教育调整、变革的重大时期,为使我国教育体制与国际化接轨,有条件的高等院校正在为某些信息学科和技术课程使用国外优秀教材和优秀原版教材,以使我国在计算机教学上尽快赶上国际先进水平。

电子工业出版社秉承多年来引进国外优秀图书的经验,翻译出版了“国外计算机科学教材系列”丛书,这套教材覆盖学科范围广、领域宽、层次多,既有本科专业课程教材,也有研究生课程教材,以适应不同院系、不同专业、不同层次的师生对教材的需求,广大师生可自由选择 and 自由组合使用。这些教材涉及的学科方向包括网络与通信、操作系统、计算机组织与结构、算法与数据结构、数据库与信息处理、编程语言、图形图像与多媒体、软件工程等。同时,我们也适当引进了一些优秀英文原版教材,本着翻译版本和英文原版并重的原则,对重点图书既提供英文原版又提供相应的翻译版本。

在图书选题上,我们大都选择国外著名出版公司出版的高校教材,如Pearson Education培生教育出版集团、麦格劳-希尔教育出版集团、麻省理工学院出版社、剑桥大学出版社等。撰写教材的许多作者都是蜚声世界的教授、学者,如道格拉斯·科默(Douglas E. Comer)、威廉·斯托林斯(William Stallings)、哈维·戴特尔(Harvey M. Deitel)、尤利斯·布莱克(Uyless Black)等。

为确保教材的选题质量和翻译质量,我们约请了清华大学、北京大学、北京航空航天大学、复旦大学、上海交通大学、南京大学、浙江大学、哈尔滨工业大学、华中科技大学、西安交通大学、国防科学技术大学、解放军理工大学等著名高校的教授和骨干教师参与了本系列教材的选题、翻译和审校工作。他们中既有讲授同类教材的骨干教师、博士,也有积累了几十年教学经验的老教授和博士生导师。

在该系列教材的选题、翻译和编辑加工过程中,为提高教材质量,我们做了大量细致的工作,包括对所选教材进行全面论证;选择编辑时力求达到专业对口;对排版、印制质量进行严格把关。对于英文教材中出现的错误,我们通过与作者联络和网上下载勘误表等方式,逐一进行了修订。

此外,我们还将与国外著名出版公司合作,提供一些教材的教学支持资料,希望能为授课老师提供帮助。今后,我们将继续加强与各高校教师的密切联系,为广大师生引进更多的国外优秀教材和参考书,为我国计算机科学教学体系与国际教学体系的接轨做出努力。

电子工业出版社

教材出版委员会

主任	杨芙清	北京大学教授 中国科学院院士 北京大学信息与工程学部主任 北京大学软件工程研究所所长
委员	王 珊	中国人民大学信息学院院长、教授
	胡道元	清华大学计算机科学与技术系教授 国际信息处理联合会通信系统中国代表
	钟玉琢	清华大学计算机科学与技术系教授 中国计算机学会多媒体专业委员会主任
	谢希仁	中国人民解放军理工大学教授 全军网络技术研究中心主任、博士生导师
	尤晋元	上海交通大学计算机科学与工程系教授 上海分布计算技术中心主任
	施伯乐	上海国际数据库研究中心主任、复旦大学教授 中国计算机学会常务理事、上海市计算机学会理事长
	邹 鹏	国防科学技术大学计算机学院教授、博士生导师 教育部计算机基础课程教学指导委员会副主任委员
	张昆藏	青岛大学信息工程学院教授

中文版序言

I am honored to write this Preface for the Chinese edition of my book *Natural Language Understanding*, second edition. I want to thank the publishers and translators for the great effort they have made to introduce my book to Chinese readers. Because the book is about language itself, the translation task was particularly challenging. It brings me great pleasure to think that Chinese readers may find this book helpful with their studies and research on Chinese language understanding.

I believe that natural language understanding will fuel the next revolution in computing. While in recent years we have seen great advances in hardware, and in applications such as graphics and multimedia, the interfaces we currently use are much the same as they were twenty years ago. We still interact with computer by selecting commands from a set of options presented by the computer. In other words, we adapt our behavior for the convenience of the computer. I believe the future lies in adapting computers to more human styles of problem solving and action. Such interfaces must be goal-driven, and depend strongly on sophisticated algorithms to infer the user's intent, and to present back information in an intuitively natural way. Natural language understanding will play a key role in this endeavour, as it is the most natural form of human communication.

This book focuses on natural language understanding (NLU) as opposed to natural language processing (NLP). The former emphasizes the importance of interpreting meaning and intent in order to achieve "deep" understanding on what is said. The latter refers to any techniques that are used to process linguistic information, and currently involve little semantics and intent. While NLP can serve many useful tasks, such as helping design better search engines, better information retrieval, and rough summarization and rough translation of documents, it so far has had little to suggest about meaning and intent. This book attempts to combine the best NLP techniques with work concerned with meaning, understanding, intent, reasoning, and acting. It describes how recent advances in statistical language processing can be used to advantage in language understanding.

At the time I wrote this second edition, there was a rift forming in the language processing community. Work on language understanding and work on statistical methods were considered to be competing incompatible methodologies. In fact, one of my main motivations on producing the second edition was to show how in fact work in both areas was complementary. I am happy to see in recent work that this is becoming more widely acknowledged in the community. Researchers working on statistical models are increasingly turning their attention to semantics and meaning, and work in language understanding is increasingly using statistical techniques to improve algorithms, ranging from

parsing to intent recognition. It is my sincere hope that this merging of interests continues and researchers will more and more see work across the fields as part of a unified effort to achieve one of the most ambitious goals ever undertaken in computation, namely, the understanding of natural language.

James Allen
University of Rochester
Rochester, New York

——译文——

为我的《自然语言理解》(第二版)一书的中文版做序我十分荣幸。我要感谢出版社和译者将本书介绍给中国读者所付出的巨大努力。因为本书是关于语言本身的,所以翻译任务非常具有挑战性。想到中国的读者可能会发现本书对于他们在学习和研究中文的自动理解方面有所帮助,我感到非常高兴。

我相信,自然语言理解将为下一次的计算机革命提供动力。近年来,虽然在计算机硬件方面有了巨大的进步,但是在诸如图形和多媒体之类的计算机应用软件方面,我们所使用的用户界面和20年前仍然毫无二致。我们与计算机的交互方式还停留在从计算机给出的一个可选集合中选择命令的阶段。也就是说,不得不为适应计算机来改变我们的行为。在未来,我相信为适应人类解决问题的行动的方式,应该是计算机做出改变。界面应该是目标驱动的,并且充分建立在一种复杂的算法基础上,这种算法能够自动推测用户的意图,并以一种符合用户直觉的自然方式向用户返回信息。而自然语言理解将在这种进程中扮演一个关键的角色,因为语言是人类交流的最自然的形式。

本书着重于介绍自然语言理解(NLU, natural language understanding),而不是自然语言处理(NLP, natural language process)。前者强调对意义和意图的解释,以实现语言的“深层”理解。而后者主要指用于语言信息处理的各种技术手段,目前还很少涉及到语义和意图。自然语言处理能够在很多有用的任务中发挥作用,如帮助设计出更好的搜索引擎、更好的信息检索,以及文本的粗略摘要和翻译,但是对于了解语义和意图来说,这还远远不够。本书试图将最好的自然语言处理技术与涉及意义、理解、意图、推理和行为等方面的工作结合起来。书中介绍了将近年来自然语言处理中取得的进展用于改进自然语言理解方面的研究。

在我写本书第二版的时候,语言处理界出现了一个巨大的鸿沟。语言理解方面的工作和统计方法方面的工作被认为是相互完全不兼容的。实际上,我写第二版的主要动机之一就是要说明这两个领域的工作实际上是可以互补的。我很高兴地看到,在近期这逐渐成为研究界的共识。统计模型方面的研究人员逐渐将他们的注意力转移到语义和意义方面,而语言理解方面的研究人员也逐步开始采用统计技术来改进包括句法分析到意图识别等各个方面的算法。我真诚地希望,这种融合将会持续下去,并且研究人员将越来越多地在交叉的领域中工作,共同努力实现有史以来计算领域中最宏伟的目标——自然语言理解。

译者序

自然语言理解,又可以称为自然语言处理或计算语言学,是一个非常有魅力的研究领域。

语言现象看似简单,实际上却反映了人类智慧中最复杂也最本质的特点。自然语言理解的研究不仅仅是一种方法和工具,而且对我们了解人类语言的奥秘、开启人类的智慧之门有着重要的影响。

现代计算机虽然具有非常强大的计算能力,但是在处理语言的时候,应用起来仍然让人感到有些力不从心。当电子计算机刚刚诞生的时候,人们首先想到的应用领域之一就是语言的处理——利用计算机进行自动翻译。不过,语言理解和处理的复杂性和困难程度远远超乎人们的预料。自然语言理解研究也因此经历过一段跌宕起伏的历史时期。到现在,人们已经充分认识到自然语言理解的复杂性和难度,不过这并未降低世界各地的研究者对这个领域的研究热情。近年来,这一研究领域的各种新思想和新方法不断涌现,一方面是由于信息社会所带来的巨大需求在有力地推动着这个领域的研究工作,另一方面也是因为这个学科本身的魅力在吸引着众多的科学家投入到这项研究工作中来。

自然语言理解是一门非常复杂的学科,涉及数学、语言学、逻辑学、心理学和计算机科学等多个研究领域。虽然现在的技术还达不到像人类那样真正“理解”自然语言的要求,但在这个领域已经取得的研究成果仍然是相当丰富多彩的。在长期的研究中,这门学科形成了一整套的理论和方法,不仅能够加深我们对人类语言现象的理解,而且能够确实解决涉及自然语言的应用中遇到的很多实际问题。

James Allen 撰写的本书是对这门学科的理论和方法的全面总结。James Allen 是这一领域非常著名的学者,曾担任本学科最权威的学术刊物 *Computational Linguistics* 的主编达 10 年之久(1983 ~ 1993)。本书第一版从 1989 年问世之初,立即受到本领域研究工作者的高度重视,并被世界各著名大学选做教材。第二版的影响则更加广泛,在第一版的基础上加入了很多反映自然语言理解新进展的研究成果,特别是在统计方法方面的研究成果。并且,第二版对全书的结构进行了调整,使之更易于理解和掌握。

本书是以教材的形式编写的,不要求读者有很深的背景知识,而且在正文和附录中介绍了一些本学科的入门知识。本书作者从事自然语言理解的教学长达 14 年之久,丰富的教学经验使他在介绍复杂的理论时能够深入浅出、易于理解。同时,他所掌握的资料的全面性和丰富性也是令人惊叹的。本书的脉络非常清楚,对本学科已有的各种理论和技术进行了比较完整和科学的总结归纳。因此,本书不仅可以作为教材使用,对于本领域的研究人员来说,也是一本可供随时查阅的优秀参考书。虽然在本书第二版出版以后的多年期间,这门学科在很多方面又发生了一些变化,不过本书所介绍的这些知识仍然是从事本领域研究应该掌握并且必不可少的。

近年来,由于受国际学术界的影响,以及应用方面巨大需求的驱动,国内研究自然语言理解的人越来越多。很多高校都开设了计算语言学或者相关的课程。我们希望本译著的出版有助于国内的研究人员和学生更全面、更准确地了解和掌握本领域已有的研究工作,最终有助于促进我国自然语言处理,特别是中文信息处理研究的进展。

本书的几位译者都是中国科学院计算技术研究所自然语言处理领域的研究人员或者博士生,从事这个领域的研究工作从几年到十几年不等。翻译这本书对于我们来说,既是一个艰巨的任务,也是一个学习的过程。

全书的审校由刘群负责。刘群负责翻译前言、第 1 章和附录,张华平负责翻译第一部分,骆卫华负责翻译第二部分,孙健负责翻译第三部分。

本书作者 James Allen 教授撰写了中译本的序言,为本书添色不少,对此我们表示特别的感谢。自动化所的陶建华博士、Rochester 大学的博士生张浩、北京市外事办信息中心主任姜伟、中科院计算所数字化室林守勋研究员,刘卫玲和刘玉东为本书的翻译提供了很多帮助,在此一并表示感谢。

由于我们学识有限,虽然已经尽了最大努力,翻译中还是难免有不恰当之处。恳请各位读者不吝赐教!

前 言

本书的主要目的是对自然语言理解领域的理论和技术进行广泛而深入的探讨。要在单独的一本书内做到这一点,就要求我们避免某些方法所特有的复杂性,并且完整地给出该领域中隐含的概念。同时,也要求我们规定数量不多的记法,并用这些记法来描述领域中的各种技术。要照搬现有文献中已经使用的各种记法是不可能的,这样做不仅无助于阐明那些重要的思想,反而会使之难于理解。本书对于读者的背景也尽可能少做要求。在可能的情况下,所有的问题和技术都用浅显的英语进行讲解。这样,任何对基本的编程符号有所了解的读者都能够理解书中所介绍的主要思想和技巧。即便如此,本书中还是给出了足够多的细节,使得一名熟练的程序员能够根据本书的介绍写出可运行的自然语言理解系统。

我们试图使本书既可以作为教材使用,也可以成为对自然语言理解感兴趣的人的一般性参考书。作为教材使用时,可以供计算机科学或计算语言学专业的本科生或者研究生使用;作为参考书使用时,既可以给自然语言相关领域的研究人员使用,也可以给自然语言系统的开发人员使用,甚至于对那些仅仅对计算机如何处理语言感兴趣的个人也很有用。

从事自然语言理解的工作者要求有广泛的背景知识,特别是计算机科学、语言学、逻辑学、心理语言学以及语言哲学等方面的背景知识。由于很少有人具有所有这些背景知识,本书在必要时尽可能给出所需的背景材料。对那些不了解编程符号和逻辑表示形式的读者,本书专门给出了两个附录,提供了足够多的基本知识,使得没有编程经验的读者也能理解本书。语言学方面和其他领域的背景知识贯穿于全书之中,在必要时进行介绍。

本书不是按照技术方法而是按照问题的领域来组织的。这样,我们就可以一次性给出各种不同的方法,并有利于对各种技术进行比较。比如,我们不是用专门的一章来介绍上下文无关语法,另一章介绍转移网络语法,再用一章来介绍逻辑编程技术,而是在很多章节中都讨论了这三种技术,每次都针对一个特定的问题集合。此外,有一章介绍基本的句法分析技术,另一章讨论特征的使用,还有一章讨论语言中移位现象的处理。每一章都首先给出问题,然后讨论如何用每一种方法来解决这些问题。这样,读者就可以清楚地了解这些方法的相同和不同之处。

为什么出第二版

自从本书第一版出版后,我一直以此为教材来讲授这门课程。由于这个研究领域的发展,我发现第一版在一些方面已经有些落伍了。在另一方面,虽然书中的内容是正确的,但侧重点不合适。出版新版本的另一个动机是我看到统计方法和语料库研究的引入导致本研究领域的一次潜在的分裂。已经有人声称新技术的引入将导致旧技术被废弃。不过,通过仔细研究这种观点,我清楚地看到,新旧两种方法是互补的,不能相互替代。我在第二版中引入了有关统计方法的新材料,并强调了如何将这些方法与传统方式相结合,以此来说明上述观点。

出版第二版的第三个动机是从教学的角度来改进这本书。原来的版本期望能面向不同背景、不同水平和不同需求的学术和研究人员,而该书的组织使得实际上要做到这一点很难。

第二版用几种方法来解决这一问题,详见下面的小节。这样,对于只有很少背景的学生,即使是只上过简单的编程课程的语言学专业本科生,本书也是易于理解的。而作为更高级的研究生教材,以及作为本领域研究人员的一般性参考书,本书也包含了足够全面的信息。

全书还有很多其他的变化,反映了从第一版出版以来这个研究领域的变化,或者至少反映了我个人观点的变化。其中很大一部分是侧重点的变化。比如句法部分,现在使用基于特征的上下文无关语法作为阐述思想的主要形式体系,而不是采用扩充转移网络。这更好地反映了本领域目前的研究工作状况,也可以更好地集成语言学家的非凡洞察力。本书对于转移网络形式体系依然进行了详细的介绍,因为该体系是这个领域中很多工作的基础。关于语义的章节也有实质性的变化,更多地集中于介绍基本的原则与问题,而不是特定的计算技巧。此外,更新了逻辑形式语言,以便覆盖更多复杂的句子。语义解释的主要方法改成了一种逐条规则的组合分析法。这不仅使得我们可以更好地讨论语义解释的基本问题,而且同样可以更好地集成语言学家的洞察力。关于在最近几年的一个很活跃的研究重点——特定领域的语义解释技术方面,本书也加入了新的内容。并且,书中基本上重写了上下文处理和话语方面的内容,以便更好地反映新的研究成果以及我对相关技术原理的新的理解。

如何使用本书

对于大学本科水平的读者来说,要完整学习全书需要两个学期。作为研究生课程,进度可以加快,可以在一个学期内学完本书的大部分内容。不过,本书最常见的用途是用于一个学期的本科课程。我在设计上考虑到了这一点,允许教师选择要重点讲授的内容。有几种方法可以帮助你本书的内容进行裁剪以满足需求。首先,每一章都分成了核心内容和可选内容。可选的章节都用一个空心的圆圈(○)标识出来。一般情况下,如果你打算选择某一章,则必须讲授其核心内容。在可选内容中,你可以选择那些最符合课程设计的部分。附加的可选内容都放在框中。框中的内容都不是必须讲授的,但可以将某些领域的讨论引向深入。

参考文献

本书包含一个本领域的详尽的参考文献列表,由此可以很容易地找到书中所描述的技术的特定细节。而且,参考文献中还给出了其他一些关键性的论文,它们讨论的问题虽然在本书范围之外,但有助于读者了解研究项目的来源。参考文献的选择主要看其是否易于获得。因此,我试图将引用的文献限制在杂志文章、书籍和主要的会议上,而避免引用那些发表十年后就很难找到的技术报告以及未发表的记录。一般来说,这个领域的很多工作都是首先以技术报告的形式发表的,更易于获取的出版物通常在这些想法被介绍很多年以后,或者在它已经开始影响本领域的其他工作时才能得到。虽然我试图尽可能提供广泛的资料来源并将本书中所介绍的思想归功于此,但我知道还是遗漏了一些关键性论文。每当我想到这些论文时都会感到无限的遗憾。对于这些作者,我非常抱歉。

致谢

如果没有罗切斯特大学的休假年,以及最近三年来计算机科学系持续不断的支持,本书是不可能完成的。我毫无阻碍地得到了完成本书所需要的超额的资源,为此我心存感激。我还必须感谢 Peggy Meeker,感谢她为本书的草稿和你们所看到的最终稿的编辑和整理所付出的努

力。在本书无数次的修改中,在长期忍受我的书写习惯所导致的看似外文的符号和语法风格时,Peggy 都表现出惊人的耐心和可贵的幽默感。

我很幸运,有许许多多的人审阅了本书各种形式的草稿,这些人包括出版商邀请的评审人以及我的很多朋友,他们都花费了很多时间来阅读书稿并做出评论。这份名单很长,而且每个人都付出了很大的努力,以至于仅仅列出他们的名字似乎都是不够的,但也只好如此了。感谢 Glenn Blank, Eugene Charniak, Michelle Degraff, George Ferguson, Mary Harper, Peter Heeman, Chung-Hee Hwang, Susan McRoy, Brad Miller, Johanna Moore, Lynn Peterson, Victor Raskin, Len Schubert, Mark Steedman, Mike Tanenhaus, David Traum, 以及在 1994 年春季使用本书的所有 CSC 447 班的学生们。获益于我收到的坦诚的和批评性的反馈,本书和早期的草稿相比已经有了显著的不同和改进之处。

最后,要谢谢我的妻子 Judith Hook。在经历了第一版写作对她造成的创伤之后,她再一次忍受所有这些痛苦。在将近一年的时间里,我像着魔一样日思夜想的就是完成本书。感谢 Judi, Jeffrey, Daniel 和 Michael,他们在这段时间里一直容忍我并对此表示出恰如其分的幽默感。

目 录

第 1 章	自然语言理解引论	1
1.1	语言的研究	1
1.2	自然语言理解的应用	3
1.3	语言理解系统的评价	4
1.4	语言分析的不同层面	7
1.5	表示与理解	9
1.5.1	句法:句子结构的表示	9
1.5.2	逻辑形式	11
1.5.3	最终的意义表示	11
1.6	自然语言理解系统的组织	11
1.7	小结	13
1.8	相关工作与深入阅读材料	13
1.9	习题	14

第一部分 句 法 处 理

第 2 章	语言学背景知识:英语句法概要	16
2.1	词语	16
2.2	简单名词短语的组成元素	18
2.3	动词短语与简单句	20
2.3.1	及物性与被动语态	22
2.3.2	语助词	23
2.3.3	从句补语	23
2.3.4	介词短语补语	24
2.4	再论名词短语	25
2.5	形容词短语	26
2.6	副词短语	27
2.7	小结	28
2.8	相关工作与深入阅读材料	28
2.9	习题	28
第 3 章	语法及其分析	31
3.1	语法与句子结构	31
3.2	优秀语法的特征	33
3.3	自顶向下的句法分析器	36

3.3.1	简单的自顶向下句法分析算法	38
3.3.2	句法分析与搜索过程	40
3.4	自底向上的 chart 句法分析器	42
3.4.1	效率方面的考虑	48
3.5	转移网络语法	48
3.5.1	基于递归转移网络的自顶向下句法分析方法	50
◦ 3.6	自顶向下的 chart 句法分析	53
◦ 3.7	有限状态模型与词语形态处理	56
◦ 3.8	语法与逻辑程序设计	58
3.9	小结	60
3.10	相关工作与深入阅读材料	60
3.11	习题	61
第 4 章	特征与扩充语法	65
4.1	特征体系与扩充语法	65
4.2	英语的一些基本特征体系	68
4.2.1	人称和数的特征	68
4.2.2	动词形式特征和动词次范畴	68
4.2.3	二元特征	71
4.2.4	特征的默认值	71
4.3	词语形态分析和词典	71
4.4	采用特征的简单语法	74
4.5	基于特征的句法分析	78
◦ 4.6	扩充转移网络	80
4.6.1	简单陈述句的 ATN 语法	82
4.6.2	寄存器的预设置	82
◦ 4.7	确定子句语法	85
◦ 4.8	广义特征体系与合一文法	87
4.8.1	形式化拓展:特征的有向无环图结构	89
4.9	小结	93
4.10	相关工作与深入阅读材料	94
4.11	习题	95
第 5 章	自然语言的语法	98
5.1	助动词和动词短语	98
◦ 5.1.1	被动句式	100
5.2	语言中的移位现象	102
5.3	上下文无关文法中的疑问句处理	107
5.3.1	带缺位的句法分析	110
◦ 5.4	关系从句	115

5.5	ATN 中的保留机制	117
5.5.1	方法比较	118
5.6	缺位索引方法	121
5.7	小结	123
5.8	相关工作与深入阅读材料	124
5.9	习题	125
第 6 章	通向高效的句法分析	129
6.1	句法分析中人的优选策略	129
6.1.1	最小附着原则	130
6.1.2	右关联原则	131
6.1.3	词汇优先原则	131
6.2	对不确定性进行编码:移进归约句法分析器	132
6.2.1	确定句法分析器的状态	132
6.2.2	移进归约句法分析器	134
6.2.3	移进归约句法分析器与歧义性	136
6.2.4	词汇的歧义性	137
6.2.5	有歧义的句法分析状态	138
6.3	确定型句法分析器	138
6.3.1	心理学上的有效性说明	142
6.4	高效的歧义表示技术	143
6.5	浅层句法分析	147
6.6	小结	148
6.7	相关工作与深入阅读材料	149
6.8	习题	149
第 7 章	歧义消解的统计方法	152
7.1	概率论基础	152
7.2	概率估计	155
7.2.1	评测	156
7.3	词性标注	157
7.3.1	Viterbi 算法	162
7.4	词汇概率的获取	164
7.5	概率上下文无关文法	168
7.6	最佳优先句法分析	172
7.7	简单的上下文相关最佳优先句法分析器	173
7.8	小结	176
7.9	相关工作与深入阅读材料	176
7.10	习题	177

第二部分 语义解释

第 8 章 语义和逻辑形式	180
8.1 语义和逻辑形式简介	180
8.2 词义与歧义	183
8.3 基本逻辑形式语言	185
8.4 逻辑形式中的歧义表示	188
8.5 动词与逻辑形式中的状态	191
8.6 论旨角色	193
8.7 言语行为与内嵌句	198
◦ 8.8 定义语义结构:模型论	200
8.8.1 句子间的语义关系	202
8.8.2 情态运算符与可能的世界语义	203
8.9 小结	204
8.10 相关工作与深入阅读材料	204
8.11 习题	205
第 9 章 句法和语义的衔接	207
9.1 语义解释与组合理论	207
9.2 带语义解释的简单语法和词典	210
9.3 介词短语与动词短语	214
9.4 词汇化语义解释与语义角色	217
◦ 9.4.1 层次词典	218
9.5 简单疑问句的处理	220
◦ 9.5.1 介词短语的特殊疑问句	222
9.6 特征合一的语义解释	223
◦ 9.7 用逻辑形式生成句子	226
9.8 小结	229
9.9 相关工作与深入阅读材料	229
9.10 习题	229
第 10 章 歧义消解	232
10.1 选择限制	232
10.2 用选择限制进行语义筛选	238
10.3 语义网络	240
10.4 统计词义消歧	244
◦ 10.4.1 搭配与互信息	246
10.5 统计语义优选	247
10.6 组合消歧方法	250
10.7 小结	251
10.8 相关工作与深入阅读材料	251
10.9 习题	252

第 11 章	语义解释的其他策略	254
11.1	语法关系	254
11.2	语义语法	258
11.3	模板匹配	260
11.4	语义驱动的分析技术	265
11.5	小结	268
11.6	相关工作与深入阅读材料	269
11.7	习题	269
第 12 章	辖域指定与名词短语的解释	271
12.1	辖域指定现象	271
12.1.1	对量词进行分类	272
12.1.2	进行辖域判定	273
12.1.3	横向辖域判定	274
12.1.4	横向辖域歧义解析	274
12.1.5	纵向辖域关系	275
12.2	定指描述与辖域指定	277
12.3	句法分析过程中进行辖域指定的方法	278
12.3.1	一个例子	280
12.4	互指与绑定约束	282
12.4.1	和全称量词的相互影响	285
12.4.2	计算互指约束	286
12.5	形容词短语	287
12.5.1	形容词比较级	289
12.6	关系名词与名词化	290
12.7	其他语义问题	292
12.7.1	不可数名词	292
12.7.2	泛指	293
12.7.3	意愿运算符与辖域解析	293
12.7.4	名词 – 名词修饰语	294
12.8	小结	295
12.9	相关工作与深入阅读材料	295
12.10	习题	296

第三部分 上下文和世界知识

第 13 章	知识表示和推理	300
13.1	知识表示	300
13.1.1	推理的类型	302
13.1.2	推理技术	303
13.2	基于一阶谓词演算的知识表示	304

13.3	框架:表示固定模式的信息	307
13.3.1	带约束的槽	308
13.4	处理自然语言中的量词	310
○ 13.5	时间和动词的体态类	311
13.5.1	时态的表示	314
○ 13.6	基于逻辑的表示方法中的自动演绎	315
○ 13.7	程序语义学与问答系统	318
○ 13.8	综合的知识表示方法	322
13.9	小结	324
13.10	相关工作与深入阅读材料	324
13.11	习题	326
第 14 章	局部篇章上下文和指代	329
14.1	定义局部篇章上下文和篇章实体	329
14.1.1	生成篇章实体	331
14.1.2	全称量词辖域内的不定指性名词短语	332
14.2	一个基于历史记录列表的简单回指模型	333
14.3	代词和中心	334
14.3.1	寻找可能的先行词	337
14.4	定指性描述	338
14.4.1	存在性解读和间接指代	340
○ 14.5	定指性指代和集合	342
14.5.1	集合元素的指代	343
14.5.2	全称量词和集合	345
14.6	省略	346
14.6.1	关于省略的句法约束	347
14.6.2	基于语法的算法	348
14.6.3	语义的倾向性	349
14.7	表层回指	351
14.8	小结	352
14.9	相关工作与深入阅读材料	353
14.10	习题	354
第 15 章	使用世界知识	356
15.1	使用世界知识:保持前后连贯性	356
15.2	与期望进行匹配	357
15.2.1	尝试 1:从期望中证明解释	358
15.2.2	尝试 2:从解释中证明期望	358
15.2.3	基于等价性的技术	358
15.2.4	基于溯因推理的技术	360

15.3	指代和匹配期望	361
15.4	使用关于行为和因果关系的知识	363
15.5	脚本:理解固定模式的情境	366
15.5.1	脚本和角色的实例化	367
15.6	使用层次化的规划	368
◦ 15.7	基于行为 – 结果的推理	371
15.7.1	规划推理算法	372
15.7.2	算法的一些局限	375
◦ 15.8	使用关于理性行为的知识	376
15.8.1	规划更新	378
15.8.2	基于执行的更新	378
15.8.3	目标更新	378
15.9	小结	380
15.10	相关工作与深入阅读材料	380
15.11	习题	382
第 16 章	篇章结构	384
16.1	篇章结构的必要性	384
16.2	切分和提示语	385
16.3	篇章结构和指代	390
16.4	篇章结构和推理的关联	393
16.5	篇章结构、时态和体态	396
16.6	管理关注栈	403
16.6.1	处理篇章状态	404
16.6.2	确定栈的更新	406
16.6.3	结构化的提示语	407
16.7	例子	408
16.8	小结	410
16.9	相关工作与深入阅读材料	410
16.10	习题	412
第 17 章	会话 agent	414
17.1	会话 agent 的必要组成部分	414
17.2	作为多 agent 活动的语言	415
17.3	认知状态:信念的表示	417
17.3.1	关于其他 agent 所持信念的知识	419
17.4	认知状态:期望、意图和规划的表示	421
17.5	言语行为和交流行为	424
17.6	规划交流行为	426
17.7	交流行为和意图判别	430

17.8 对话中意图的来源 432

17.9 识别语内表现行为 434

17.10 篇章层次的规划 437

17.11 小结 438

17.12 相关工作与深入阅读材料 439

17.13 习题 440

附录 A 逻辑和模型论语义学介绍 442

附录 B 符号计算 453

附录 C 语音识别与口语 465

参考文献 479

索引 494

第 1 章 自然语言理解引论

本章描述了自然语言理解这一研究领域,并介绍了它的一些基本特点。1.1 节讨论自然语言理解研究与一般的语言研究的关系。1.2 节讨论一些自然语言理解系统的应用并初步探讨了对于一个系统来说“理解语言”意味着什么。1.3 节描述如何评价一个系统是否理解了语言。1.4 节介绍了通过对语言的研究总结出的一些基本的特征,而 1.5 节讨论了计算系统通常是如何实现这些特征的。最后,1.6 节讨论了自然语言理解系统通常是如何组织的,并介绍了由作者所设想的贯穿全书的一种具体组织形式。

1.1 语言的研究

语言是人类行为的一个基本方面,也是我们生活的一个极为重要的组成部分。书面形式的语言记录了人类一代一代长期积累下来的知识,而口语则是在我们日常生活中与其他人沟通协调的基本方法。本书描述了有关语言的理解和生成方面的研究工作。这种研究的目的是为语言构造出足够精细的计算模型,以便你能够写出由计算机程序来完成的涉及自然语言的各种任务。其终极目标是能够给出一些模型,这些模型在完成阅读、写作、听、说等任务时能够接近人的行为。然而,本书并不关注与所使用的特定媒介相关的问题,例如手写输入、键盘输入或是语音输入的问题。相反,本书所关注的是在词语识别完成后理解和使用语言的过程。计算模型有两方面的用途,作为科学研究的目的——可以探索语言交流的本质;作为实用的目的——能够实现有效的人机通信。

在几个不同的学科领域中都有关于语言的研究。每个学科都定义了自身的一套问题,并有其自己的解决方法。例如,语言学家研究语言本身的结构。他们考虑这样一些问题,为什么特定词语的组合能形成句子而其他的词语组合则不能,为什么一个句子可能具有某种意义而不是另外一种意义,等等。另一方面,心理语言学家研究人类生成和理解语言的过程。他们考虑这样一些问题,人类是如何识别一个句子的合理结构的,何时确定词语的合理意义,等等。哲学家考虑的是词语如何能够表示所有的事物,以及它们怎样标识现实世界中的实体。哲学家也考虑拥有信仰、目标和意图意味着什么,以及这些认知能力和语言是如何联系的。计算语言学家的目标是利用计算机科学中的算法和数据结构的概念建立一个语言的计算理论。当然,为了建立一个计算模型,你必须利用所有其他学科中已有的知识。图 1.1 总结了研究语言的这些不同方法。

如前所述,建立计算模型有两个动机。科学方面的动机是为了对语言如何工作这个问题获得更好的理解。我们认识到任何一个传统学科都没有提供工具来全面地处理语言的理解和生成问题。即使你结合了所有的方法,这种综合的理论也会过于复杂以至于不能用传统的方法进行研究。但是,我们也许可以用计算机程序来实现这样一个复杂的理论,然后通过观察它们的执行情况来进行测试。通过观察它们在哪里失败了,我们可以逐步对其进行完善。计算模型可能提供对人类行为的详尽预测,而心理语言学家对这些行为进行研究。持续这个过程,我们最终也许会逐步对人类语言处理是如何发生的这个问题获得更为深入的理解。实现这个

梦想需要语言学家、心理语言学家、哲学家和计算机科学家的共同努力。这个共同的目标已经促成了一个新的交叉学科——通常称为认知科学。

学科	典型问题	工具
语言学家	词语如何形成短语和句子？什么因素限制了一个句子可能具有的意义？	关于句子合法性和意义的直觉；结构的数学模型(例如,形式语言理论、模型论语义学等)
心理语言学家	人们如何识别句子的结构？词语的意义是如何被识别的？理解发生在什么时候？	基于评估人类行为表现的实验技术；对观察资料的统计学分析
哲学家	什么是意义？词语和句子如何获得意义？词语如何标识现实世界中的实体？	使用有关反例的直觉知识进行的自然语言论证；数学模型(例如,逻辑和模型论)
计算语言学家	如何识别句子的结构？如何为知识和推理建模？语言怎样才能被用来完成特定的任务？	算法、数据结构；表示与推理的形式模型；人工智能技术(搜索和表示方法)

图 1.1 研究语言的主要学科

实践或技术方面的动机在于自然语言处理能力将给使用计算机的方法带来一场革命。既然大多数人类的知识都是以语言的形式记录下来的,那么能够理解自然语言的计算机就能够获取所有这些信息。另外,计算机的自然语言接口将使得每个人都能够使用复杂的系统。这种系统所具有的灵活性和智能性将远远超过现有的计算机技术所能达到的水平。从技术的目的看,模型与人类处理语言的方式是否一致并不重要,关键在于它是否有效。

框 1.1 框和可选节

本书使用了几种技巧来让你识别哪些内容是核心的而哪些是可选的。另外,可选的内容中有一些被归类为高级部分,这说明你也许需要本书中没有涵盖的其他背景知识来充分理解这些内容。框,就像这里一样,总是包含可选的内容,这些内容或者给正文中讨论的特定方法提供更多的细节,或者讨论与正文相关的另外一些问题。本书通过在标题前面加一个空心的圆圈(○),将节或小节标记为可选的。可选部分对该章内容在广度和深度上进行了扩充,但对以后各章的理解不是必要的。根据你的兴趣和关注点的不同,你可以在可选节中进行取舍,对常规节中提及的核心内容进行补充。另外,各章之间有从属关系,所以如果某部分内容不符合你的兴趣则可以跳过整个章节。各章的从属关系在正文中没有清楚地标记出来,但是在引言中给出了一张从属关系图。

本书涉及的是科学目的和技术目的之间的一个中间地带。一方面,这反映了一个信念:自然语言是如此复杂,以至于对一个专门的方法来说,如果没有一个特定的理论做基础,是不会成功的。因此,如果没有语言学家、心理语言学家和哲学家已经建立的精密的理论作为基础,技术的目标是不可能实现的。另一方面,现有的关于自然语言处理的知识还仅仅处于一个非常初级的水平。因此,从认知角度看,试图建立一个正确反映人类认知过程的模型也是不现实的。为此,我们还是尝试建立任何实际有用的模型。

本书的目的是描述这样一些工作,其目标是从语言学知识的角度出发构造关于语言理解和生成的计算模型,而这些模型在特定的领域中能够有良好的表现。虽然本书主要关注于语言处理的计算方面,书中还是有相当多的部分介绍来自其他学科的背景知识,这些知识是这些计算手段的驱动力和正确性的保证。本书假定读者只有程序设计的基础知识,而一些有语言学、人工智能(AI, artificial intelligence)和逻辑学背景知识的学生将会欣赏到书中的另外一些精妙之处。

1.2 自然语言理解的应用

定义自然语言研究的一个好方法是考虑其研究者们所从事的不同应用。在你考虑这些例子的同时,这也是一个很好的考察机会:当我们说一个计算机系统理解了自然语言的时候,到底意味着什么。自然语言理解的应用可以分为两大类,基于文本的应用和基于对话的应用。

基于文本的应用涉及对书面文本的处理,例如书籍、报纸、报告、手册、电子邮件消息,等等,这些都是基于阅读的任务。基于文本的自然语言研究主要在如下一些应用领域进行:

- 从一个文本数据库中找到与特定话题相关的合适的文本(举例来说,从一个图书馆里找到相关的书籍)
- 从消息或文章中抽取与特定话题相关的信息(举例来说,根据给定某一天的大量新闻报道的描述来建立一个股票交易数据库)
- 将文档从一种语言翻译成另一种语言(举例来说,产生多语种的汽车自动维修手册)
- 为特定目的对文本进行总结(举例来说,把一份 1000 页的政府报告压缩成一篇 3 页的总结)

并不是所有执行上述任务的系统都要使用我们在本书中介绍的自然语言理解技术。举例来说,考虑在一个大型数据库中寻找有关特定话题的新闻报道的任务。人们开发了许多根据文本中出现的特定关键词对文档进行分类的技术。然后,通过查找含有与某特定话题相关的关键词的文章的办法,你就可以检索与该话题相关的文章。举个例子,关于法律的文章,可能含有诸如“律师”、“法院”、“控告”、“书面陈述”之类的关键词,而关于股票交易的文章通常含有诸如“股票”、“接管”、“融资收买”、“期权”之类的词语。这样一个系统可以检索与任何一个话题有关的文章,只要我们预先定义了该话题的关键词集合。很明显,我们不会说这个系统理解了文本;相反,它只不过是使用了一种简单的匹配技术。虽然这样的技术也能产生出有用的应用,但它们有着固有的局限性。举例来说,一些可用自然语言轻易表达的复杂的检索任务,诸如“给我找出所有有关在 1986 年到 1990 年之间曾经尝试但最终失败且金额超过 1 亿美元的融资收买的文章”之类,就几乎不太可能应用这些技术进行处理。要处理这样的查询,系统必须有能力从数据库的每篇文章中抽取足够多的信息,以判定这篇文章是否满足查询中所设定的要求。也就是说,它必须为文章中的信息建立一种表示形式,然后使用这种表示形式进行检索。这也就是理解系统的一个关键特点,必须计算信息的某种表示形式,而这种表示形式能用于后续的推理。

考虑另外一个例子。有些机器翻译系统是建立在模式匹配的基础上的,就是说,一种语言中的一个单词序列与另一种语言中的某个单词序列相关。要完成翻译,就是要找到与输入语句匹配的最佳的模式集合,然后在另一种语言中产生相应的输出。在某些情况下,这种技术可

以产生合理的结果,但有时也会产生完全错误的翻译,因为该技术不可能通过对内容的理解来合理地消除单词和句子的歧义。与其相反,另外有一些机器翻译系统的原理是:为一种语言中的每个句子的意义生成一种表示形式,然后在另外一种语言中产生能够表达同样意义的句子。后面这种方法因为涉及了意义的表示形式的计算,因此可以说是使用了自然语言理解的技术。

基于文本进行研究的一个非常有吸引力的领域是故事理解。在这个任务中,系统先要处理一个故事,然后回答有关这个故事的问题。这类似于学校中进行的阅读理解测验,而且这种方法为评价一个系统所能达到的理解深度提供了丰富的手段。

基于对话的应用涉及人机之间的通信。大部分情况下这都涉及口语,但也包括使用键盘的交互。潜在的典型应用包括:

- 问答系统,其中自然语言用于查询一个数据库(举例来说,一个个人数据库的查询系统)
- 通过电话的自动客户服务系统(举例来说,进行银行交易或者根据商品目录下订单)
- 教学系统,在系统中机器可以与学生进行交互(举例来说,一个自动的数学教学系统)
- 通过口语控制机器(举例来说,用声音来控制一台录像机或者计算机)
- 通用的协作式问题解决系统(举例来说,一个帮助人进行船舶货运计划和调度的系统)

对话系统所面对的一些问题和基于文本的系统有很大不同。首先,所使用的语言有很大差异。而且,系统必须积极参与对话,使得对话能够自然、流畅地持续下去。对话要求使用确认性的句子,告知对方确认事物得到理解。并且还要具有这样一种能力,即能够识别出没有弄清楚的问题,并产生一些用于澄清理解的子对话。不过,虽然具有这些区别,这两种系统的基本处理技术本质上还是一样的。

区分语音识别问题和语言理解问题是非常重要的。一个语音识别系统不需要涉及任何语言理解。举例来说,声音控制的计算机和录像机已经进入了市场。通常,这并不涉及任何自然语言理解。相反,所识别出的单词只是当做命令来使用,类似于你通过遥控器对录像机发出的命令。语音识别只关心从给定的声音信号中识别出所说的单词,并不理解这些单词是如何用于交流的。要成为一个理解系统,语音识别器需要将它的输入送到一个自然语言理解系统,从而产生一个我们通常所说的口语理解系统。

本书中讨论的所有技术与基于文本和基于对话的自然语言理解几乎都同等相关,并且不管输入所使用的是文本、键盘或者语音,都有同样的应用效果。任何一个理解系统的关键特点都在于它要将句子的意义用某种表示语言表达出来,而这种表示语言能用于后续的进一步处理。

1.3 语言理解系统的评价

就像你看到的,理解的含义随着应用的不同而有所差异。如果是这样,怎样才能说一个系统正常工作了呢?系统的一种显而易见的评价办法是运行该程序并观察它在完成设定任务时表现如何。如果一个系统设计用于回答有关某个事实数据库的问题,你可以问它问题,看它在产生正确的回答方面做得怎样。如果一个系统设计用于参与一个特定话题的简单对话,你就可以尝试着和它对话。这称做黑箱评价,因为这种方法是在不了解一个系统如何工作的情况下对系统性能进行评价的。虽然这种方法最终是测试一个系统能力的最好方法,不过在研究的早期阶段应用这种方法会出现问题,因为早期的评价结果会引起误解。有时,一种在短期内

能够产生最好结果的技术从长远看并不能产生最好的结果。举例来说,如果某一个应用领域内所有已知系统的综合性能一律很低,那么就得出不出什么结论。一个系统在 50% 的情况下正确而另一个系统在 40% 的情况下正确这样一个事实,从长远看对这两种方法的发展前景说明不了任何问题。只有当成功率很高,并使得实际的应用可行时,这种综合系统性能测量方法才能显现出其重要性。

系统评价的另外一种方法是确定一个系统的各个组成部分,然后用合适的测试方法来评价每一个部分。这称做白箱评价,因为你要了解系统的内部结构。白箱评价的问题在于它要求对一个自然语言理解系统具有哪些组成部分达成某种一致。目前,为达成这种一致而进行的研究是相当活跃的一个领域。

尽管有种种困难,对这个研究领域来说,评价仍是至关重要的,因为只检查一个特定系统所能处理的少数例子几乎得不出什么结论。如果你不是亲自和一个程序自由地进行交互,而只是参考其他人交互的记录,则很容易被误导以至于高估一个程序的能力。人们很容易误以为一个系统具有比实际多得多的智能。

考虑一个具体的例子——20 世纪 60 年代中期在麻省理工学院开发的 ELIZA 程序。一直以来,这个程序都是最流行的人工智能程序之一,对于大部分机器它都有相应的版本,包括大部分的个人计算机。ELIZA 从来没有声称它实现了某种语言的理解和生成理论,但它可以作为一个很好的例子,因为它的行为一开始往往给人留下很深的印象。这个系统扮演一个心理医生的角色,并且为了获得最好的结果,用户应该相应地扮演病人的角色。图 1.2 给出了这个系统运行的一次记录。仅仅看这个记录,甚至你亲自花几分钟去操作这个系统将会发现,ELIZA 的性能看上去确实能给人留下很深的印象。

用户	ELIZA
Men are all alike.(男人都一样。)	In what way?(在哪方面?)
They're always bugging us about something or other. (他们总是用这样或者那样的事情来麻烦我们。)	Can you think of a specific example? (你能想到一个特定例子吗?)
Well, my boyfriend made me come here. (是啊,我男朋友要我来这儿。)	Your boyfriend made you come here. (你男朋友要你来这儿。)
He says I'm depressed much of the time. (他说我大部分时间都情绪低落。)	I am sorry to hear you are depressed. (我很遗憾听说你情绪低落。)
It's true, I am unhappy. (真的,我不快乐。)	Do you think coming here will help you not to be unhappy? (你认为来这儿可以帮助你快乐起来吗?)
I need some help; that much seems certain. (我需要某种帮助,肯定是这样的。)	What would it mean to you if you got some help? (如果你得到某种帮助,那意味着什么呢?)
Perhaps I could learn to get along with my mother. (也许我可以学会如何跟我母亲相处。)	Tell me more about your family. (告诉我更多你家里的情况。)
My mother takes care of me. (我母亲照顾我。)	Who else in your family takes care of you? (你家里还有谁照顾你?)
My father.(我父亲。)	Your father.(你父亲。)
You are like my father in some ways. (你在某些方面像我父亲。)	What resemblance do you see? (你看到了什么相似之处?)

图 1.2 和 ELIZA 的一次对话

下面是关于 ELIZA 如何工作的一个简单描述。有一个数据库中存放了一些特定的单词,称为关键词。对于每一个关键词,系统保存一个整数、一个用于匹配输入的模式和一个输出的格式。算法如下:给定一个句子 S,在 S 中查找一个关键词,这个关键词的模式要能够与 S 匹配。如果有多个关键词,选择其中整数值最大的那个。使用与该关键词相关联的输出格式产生下一个句子。如果没有匹配的关键词,输出一个无伤大雅的句子继续对话,如“还有呢”或“继续”。

图 1.3 显示了一个关键词数据库的片段。在这个数据库中,一个模式由一些单词和变量组成。一个字母前面的前缀“?”表示一个变量,该变量可以匹配任意的单词序列。例如,模式:

?X are you ?Y

可以匹配句子“Why are you looking at me?”其中,变量“?X”与“why”匹配而“?Y”与“looking at me”匹配。输出规范同样可以使用这些变量。在这个例子中,ELIZA 对输入中与变量匹配的单词做少许变换(例如,将“me”换成“you”)后,将这些单词插入到输出中。这样,对于上面的模式,如果输出规范是:

Would you prefer it if I weren't ?Y?

这条规则将产生一个回应“Would you prefer it if I weren't looking at you ?”当数据库中对于一个给定模式列出多个输出规范时,ELIZA 会在该关键词规则每次被使用时做出不同的选择,这样可以避免对话中不自然的重复。使用这些规则,你就可以看到 ELIZA 是如何产生图 1.2 中给出的前两条对话了。ELIZA 根据关键词“alike”的第一条输出产生了第一个回答,并根据关键词“always”的第一个输出产生了第二个回答。

词语	级别	模式	输出
alike	10	?X	In what way?
			What resemblance do you see?
are	3	?X are you ?Y	Would you prefer it if I weren't ?Y?
	3	?X are ?Y	What if they were not ?Y?
always	5	?X	Can you think of a specific example?
			When?
			Really, always?
what	2	?X	Why do you ask?
			Does that interest you?

图 1.3 ELIZA 中的一些数据样例

以上描述覆盖了这个程序的所有基本点。你可能会同意以下观点,这个程序并没有理解它所参与的对话。相反,它只是一些技巧的集合。为什么 ELIZA 看上去表现得这么好呢?这有几个原因。可能最重要的原因在于,当人们听到或读到单词序列并将它理解为一个句子时,他们会给这个句子赋予意义,并假设产生这个句子的人(或机器)确实想表达这种意义。人类特别擅长区别词语的意义并根据上下文来解释句子。如此一来,ELIZA 看上去就像有了智能,因为它使用了你自己的智能来使它所说的话变得有意义。

这个对话场景的其他一些关键性特点也有助于造成 ELIZA 有智能的假象。举例来说,这个系统不需要任何世界知识,因为它从不提出一个要求、支持一种观点或者回答一个问题。相反,它只是简单地回答一系列的问题。如果不是患者和心理医生对话这样一种场合,这将是无

法接受的。ELIZA 通过提出另外一个问题来回避所有的直接提问,如“Why do you ask?”(你为什么这么问?)你不可能用任何方法来强迫系统说出关于任何话题的一些具体事情。

即使在一个受限的场合,要证明这个程序并没有真正理解,还是很容易做到的。它有时会产生一个完全异乎寻常的回答。比如,如果你说“Necessity is the mother of the invention”(需求是发明之母),根据关于词语“mother”(母亲)的模式,它可能会回答“Tell me more about your family!”(告诉我更多你家里的情况!)还有,因为 ELIZA 没有任何关于语言结构的知识,所以它会把任何的胡言乱语当做正常句子一样来接受。如果你输入“Green the adzabak are the a ran four”,ELIZA 会产生诸如“What if they were not the a ran four?”之类的回答。而且,随着对话的进展,有一点越来越明显,就是这个程序没有保留对话的任何上下文。它开始问一些根据先前的对话来看完全不合适的问题,而且它的回答通常会逐渐显得没有重点。当然,如果你不能直接操作这个程序而只能依赖于其他人的对话记录,就没有办法检测到这些问题,除非它们被清楚地提及。

设想你要在仅仅 6 个月时间内为一个特定的应用构造一个自然语言程序。如果你从头开始构造一个一般性的语言理解模型,将无法在指定的期限内完成,这样在测试中会表现得很糟糕。相反,一个类似 ELIZA 的系统,可以在不到几个月的时间内编写出来,而且可以产生类似前面讨论过的行为,在测试中会表现得远远超过其他系统。如果测试数据仅仅包括典型的领域交互行为,而不是设计为测试系统的能力极限,这种差别会更为明显。这样,如果把短期表现作为我们取得进展的惟一准则,每个人都会构造一个精心调试的 ELIZA 风格的系统,而这个研究领域也不会超越这个简单方法的局限而取得进展。

为避免这种问题,我们要么必须接受关于自然语言系统结构的一些理论假设并且对于每一个不同的组成部分制订特定的评价方法,要么不理睬这种总体的评测,直到系统达到一个足够高的性能水平。只有到那个时候,系统之间的比较才能够反映出它们在这个领域中取得远期成功的潜力。

1.4 语言分析的不同层面

一个自然语言系统必须使用相当多的关于语言自身结构的知识,包括什么是词、词如何组成句子、词的意义是什么、词的意义对句子的意义有什么影响,等等。然而,如果不考虑构成人类智能的其他方面的因素——人类的一般性世界知识和人类的推理能力,我们就不可能完全解释人类的语言行为。比如,一个人要回答问题或者参与对话,他不仅需要知道所使用的语言结构的很多知识,而且要知道关于世界的一般性知识以及了解特定的对话场景。

下面是与自然语言理解有关的一些不同形式的知识:

语音和音韵知识——关心词语与其发音如何关联。这种知识对于基于语音的系统是至关重要的。相关的细节在附录 C 中讨论。

词语形态学知识——关心词语如何由被称为词素的更基本的意义单位构成。词素是语言中一种最基本的意义单位(例如,单词“friendly”的意义可以从名词“friend”的意义以及将名词转换为形容词的后缀“ly”推导出来)。

句法知识——关心词语如何排列以组成正确的句子,并决定每个单词在句子中所充当的结构角色,以及短语之间的构成关系。

语义知识——关心词语的意义以及在句子中词语意义是如何互相结合以形成句子意义的。这是上下文无关的意义研究:一个句子在不考虑其所处的上下文的情况下所具有的意义。

语用知识——关心句子如何在不同的情形下被使用,以及这种使用如何影响句子的解释。

篇章知识——关心前面的句子如何影响对下一个句子的解释。这种信息对于代词的解释以及所传递信息的时态的解释特别重要。

世界知识——包括关于这个世界结构的一般性知识,这种知识对语言的使用者来说是必需的。比如要维持一个对话,就需要利用这种知识。这包括语言的每一个使用者必须了解其他使用者的信念和目标。

以上这些都不是一种精确的定义。事实上,它们更多的是这些知识的一些特征,而不是对这些知识进行严格的分类。任何一个特定的事实都可能涉及到多个不同的层面,而且一个算法也可能需要同时从几个不同的层面进行提炼。尽管如此,为教学的目的,本书还是组织为三个部分,每一部分都描述了一组内在相关的技术。第一部分的重点是句法和形态处理,第二部分着重介绍语义处理,第三部分主要介绍上下文的影响,如语用、话语和世界知识。

框 1.2 句法、语义和语用

下面的例子可能会有助于你理解句法、语义和语用之间的区别。考虑把下面每个例子都当做本书的首句的候选,当然你已经知道本书是讨论自然语言处理的:

1. 语言是人类行为的一个基本方面,也是我们生活的一个极为重要的组成部分。
2. 绿青蛙有大鼻子。
3. 绿思想有大鼻子。
4. 大有绿思想鼻子。

第一句看上去是个合理的开始。(我希望是这样!)它符合所有的句法、语义和语用方面的要求。其余的每个句子都在一个或者多个层面上违反了这些要求。第二句符合句法和语义的要求,但违反了语用要求。作为起始句它确实很不合适,因为读者不知道为什么要使用这个句子。不过,虽然第二句放在本书的开始非常不好,第三句却更加糟糕。它不仅在语用上明显是不合适的,并且在语义上也是不合适的。为了说明这一点,可以设想你和我可以就第二句是对还是错进行争论,但对于第三句我们就无法这样做。我无法在一次合理的会话中对第三句进行肯定或者否定。尽管如此,这个句子还是有某种结构的,因为我们可以讨论它错在什么地方:思想不能是绿色的,即使能,思想肯定也不会有大鼻子。而第四句更加糟糕。实际上,它完全不可理解,即使组成它的单词和第三句完全相同仍是这样。它甚至没有形成基本的句子结构,以便你能说出这个句子到底错在什么地方。顺便提一句,也有一些句子可能在语用上是合适的,但在句法上并不合适。例如,如果我问你去哪儿而你回答“I go store”(我去商场——英语中 go 不能直接带宾语)。这个回答是可以理解的,即使它是不符合句法的。至少它在语用上是合适的,甚至可能在语义上也是合适的。

1.5 表示与理解

正如前面所说,理解的一个关键组成部分是对句子或文本的意义表示进行计算。然而,如果不对表示这个概念进行定义,上述说法就毫无意义。举例来说,为什么不简单地使用句子本身作为其意义的表示形式?一个原因是大部分词都有多重意义,我们称之为词义。例如单词“cook”,有一个动词词义和一个名词词义;“dish”有一个动词词义和多个名词词义;而“still”具有名词、动词、形容词和副词词义。这种歧义使得系统无法进行适当的推理,因而不能建立理解的模型。由于人们通常很少注意到歧义,所以排歧问题看上去好像很容易,但实际上要困难得多。人在理解一个句子的时候似乎不会考虑一个词所有可能的词义,但一个程序必须明确地一个一个地考虑这些词义。

为了表达意义,我们需要一种更加精确的语言。做这件事的工具来自于数学和逻辑学,并且需要用到形式化定义的表示语言。形式语言的规定从非常简单的构造单位开始。最基本的概念是仅仅依靠书写形式来相互区分的原子符号。可用的表示语言具有以下两个特点:

- 这种表示必须是精确的和无歧义的。你应该可以用这种表示语言中不同的公式来表示一个句子的每一种不同的解读。
- 这种表示应该能刻画它所表示的自然语言句子的内在结构。例如,看上去结构类似的句子应该具有类似的结构表示形式,两个互为解释的句子的意义应该有紧密的关联。

对应于上一节中所讨论的几个分析层次,有几种不同的表示形式。更具体地说,我们将开发出一些形式语言来表示句法结构、上下文无关的词语和句子的意义,以及表示一般的世界知识。

1.5.1 句法:句子结构的表示

句子的句法结构指明了句子中的单词互相关联的方式。这种结构指明了词语是如何组合成短语的,哪一个词修饰另一个词,而哪一个词处于句子中心的重要位置。另外,这种结构还可以指明存在于短语之间的关系类型,并保存后续处理所必需的其他一些特定句子结构的信息。举例来说,考虑以下句子:

1. John sold the book to Mary. (约翰把这本书卖给了玛丽。)
2. The book was sold to Mary by John. (这本书被约翰卖给了玛丽。)

这两个句子具有某些共同的结构特点。每个句子中的名词短语都是“John”,“Mary”和“the book”,而且所描述的行为也都是“卖”这个行为。而从另一方面看,这两个句子又有显著的不同。例如,虽然这两个句子在完全相同的情况下总是同为真或同为假,但是,你只能把句子1作为问题“What did John do for Mary?”(约翰对玛丽做了什么?)的回答,而把句子2作为一个以短语“After it fell in the river”(在它掉到河里以后)开始的句子的后半部分就更为合适,就像句子3和句子4所显示的那样。按照语言学的标准惯例,本书在任何一个不合乎语法的句子或有问题的句子前面加上一个星号(*)。

3. * After it fell in the river, John sold Mary the book.
(* 当这本书掉到河里以后,约翰把它卖给了玛丽。)
4. After it fell in the river, the book was sold to Mary by John.
(当这本书掉到河里以后,它被约翰卖给了玛丽。)

在分析一些不合乎语法的句子的时候,可以看到很多其他的结构特点。句子 5 是不合乎语法的,因为主语和谓语不一致(主语是单数而谓语是复数);句子 6 也是不合乎语法的,因为动词 put 需要一个修饰语来描述约翰把东西放在什么地方了。

5. * John are in the corner.(约翰在角落里。)
6. * John put the book.(约翰把这本书放下。)

根据语法做出判断并不是自然语言理解的目标。事实上,一个健壮的系统应该能够在尽可能的情况下理解一个不合乎语法的句子。这似乎意味着一致性检查是可以忽略的,但事实并非如此。一致性检查对于消除潜在的歧义来说是非常重要的。考虑句子 7 和句子 8,除了主动词的数特征不同以外,其余部分完全相同,却有两种非常不同的解释。

7. Flying planes are dangerous.(正在飞行的飞机是危险的。)
8. Flying planes is dangerous.(开飞机是危险的。)

如果不检查主语和谓语的一致性,这两个句子就是不可区分的,而且也是有歧义的。对于本书介绍和使用的每一个句法特征,你都可以找到类似的例子。

语言的大部分句法表示形式都是基于上下文无关语法这一概念,这种语法将句子结构以一些短语是另一些短语的组成部分这样一种形式表示出来。这种信息通常用树的形式来表示,如图 1.4 所显示的那样。图中显示了“Rice flies like sand.”这个句子的两种不同的结构。按照第一种解读法,句子由一个名词短语(NP)和一个动词短语(VP)组成。名词短语描述了一种类型的苍蝇——米苍蝇,动词短语断言这些苍蝇像沙子。在第二个结构中,句子也是由一个名词短语和一个动词短语组成的。名词短语描述了一种物质——米,动词短语陈述说这种物质撒开来像沙子(假如你抛撒它)。这两个结构还给出了关于其中的名词短语和动词短语的更多细节,并指明了每一个词的词性。更具体地说,词“like”在第一种解读法中是动词(V),而在第二种解读法中是介词(P)。

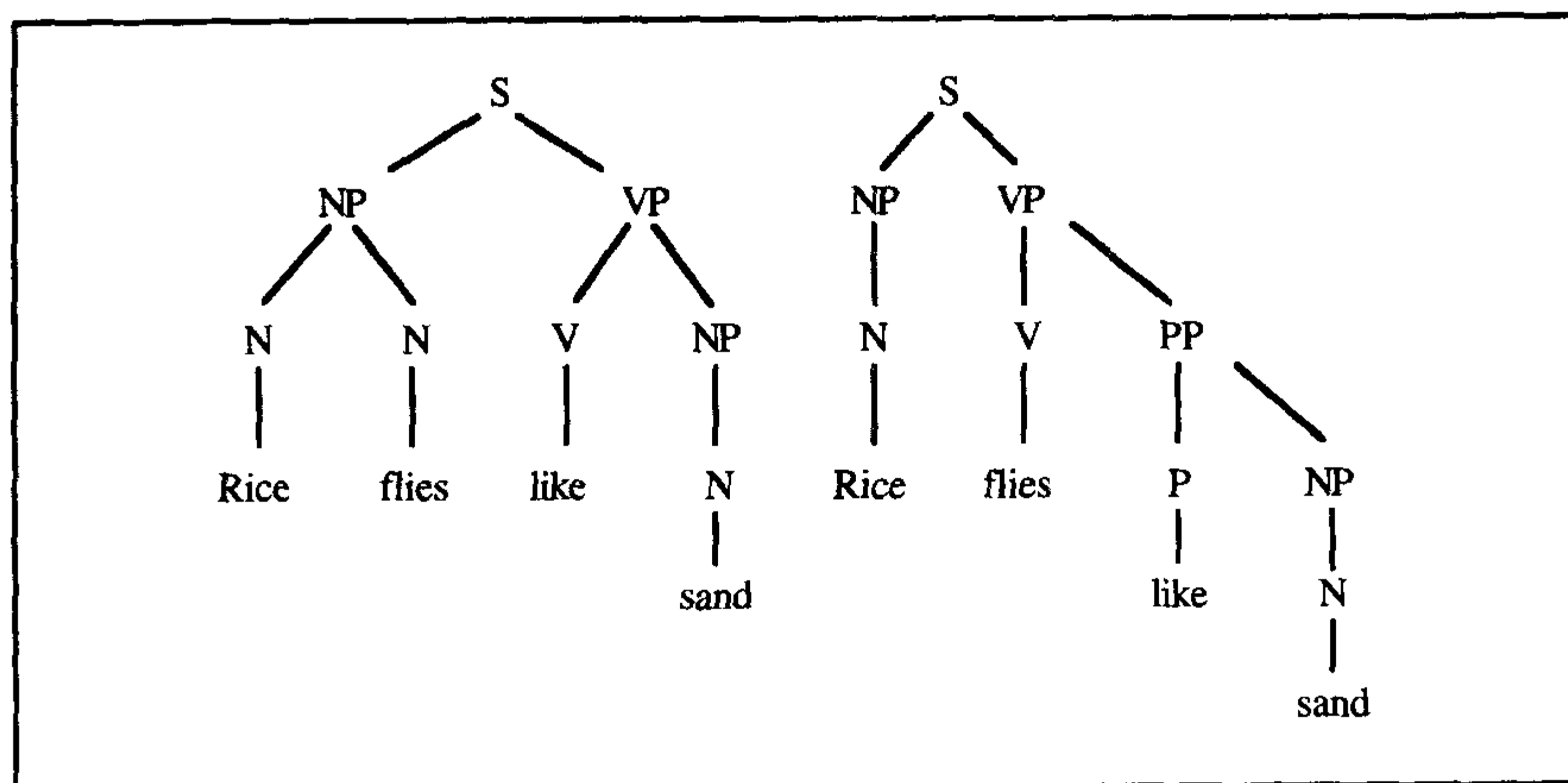


图 1.4 “Rice flies like sand.”的两种结构化表示法

1.5.2 逻辑形式

不过,句子的结构并不反映句子的意义。举例来说,名词短语“the catch”根据说话者是在谈论一次棒球比赛还是一次垂钓从而有不同的意义。这两种解释都有相同的句法结构,而意义上的不同来自于单词“catch”在词义上的歧义。一旦确定了正确的词义,比如是钓鱼时的词义,在确定所指的是什么鱼方面,仍然有问题。句子的意图依赖于产生这个句子的情境。本书不是把所有这些问题结合起来考虑,而是分别考虑每一个问题。分界线在于上下文相关的语义和上下文无关的语义之间。“catch”这个词可能指的是移动的棒球,也可能是指垂钓的结果,这一事实是由英语的语言知识决定的,与使用这个词时所处的情境无关。另一方面,一个特定的名词短语“the catch”指的是昨天钓鱼的收获,这一事实是上下文相关的。一个句子的上下文无关意义的表示被称为它的逻辑形式。

逻辑形式对可能的词义进行编码,并给出词语和短语之间的语义关系。这种关系通常大多用一个抽象的集合来表示,这个集合中包含动词及与其相关的名词短语之间的各种关系。具体来说,在前面给出的句子1和句子2中,所描述的行为都是一个出售事件。其中,“John”是卖方,“the book”是被卖的对象,而“Mary”是买方。这些角色分别是抽象的语义角色 AGENT(施事)、THEME(论题)和 TO-POSS(目标所有者)的具体实例。

一旦语义关系确定了,某些词义就不再可能出现,也就不再被考虑。例如下面这个句子:

9. Jack invited Mary to the Halloween ball.(杰克邀请玛丽去万圣节前夕的舞会。)

“ball”这个词本身是有歧义的,可能是指有弹性的玩具(球),也可以指正式的舞会活动。在句子9中,这个词只能是后一种词义,因为动词“invite”只有在这种解释下才有意义。语义解释的一个关键任务就是考虑哪些单个的词义能够互相结合产生出前后连贯的句子意义。利用这种词语意义之间的互相关联可以大大减少一个给定的句子中每一个词语可能的词义数量。

1.5.3 最终的意义表示

系统需要的最终表示方法是通用的知识表示(KR, knowledge representation),系统利用它进行应用领域的表示和推理。这是一种所有基于该应用领域的特定知识表示所使用的语言。上下文相关解释的目标就是得到一个句子的结构及其逻辑形式的表示,并把这种表示映射为通用的知识表示,以便系统在该应用领域中完成适当的任务。在回答问题的应用中,一个问题可能要映射到对一个数据库的查询;而在一个理解故事的应用中,一个句子可能要映射到一个表达式的集合,这个表达式表示了这个句子所描述的情境。

在大部分情况下,我们会假设一阶谓词演算(FOPC, first-order predicate calculus)就是最终的表示语言,因为它被精确定义、被充分研究并且被大家所熟悉的。虽然一阶谓词演算也有一些不足(后面将会进行探讨),不过对于我们要讨论的大部分问题来说,这些不足是无关紧要的。

1.6 自然语言理解系统的组织

本书围绕刚才所讨论的表示的三个层面来组织:句法结构、逻辑形式和最终的意义表示。

按这种方式对问题进行分割允许你深入研究每一个问题,而不必考虑其他问题的复杂性。不过,实际系统的组织通常相对有些差异。具体来说,图 1.5 显示了本书所假设的系统组织结构。

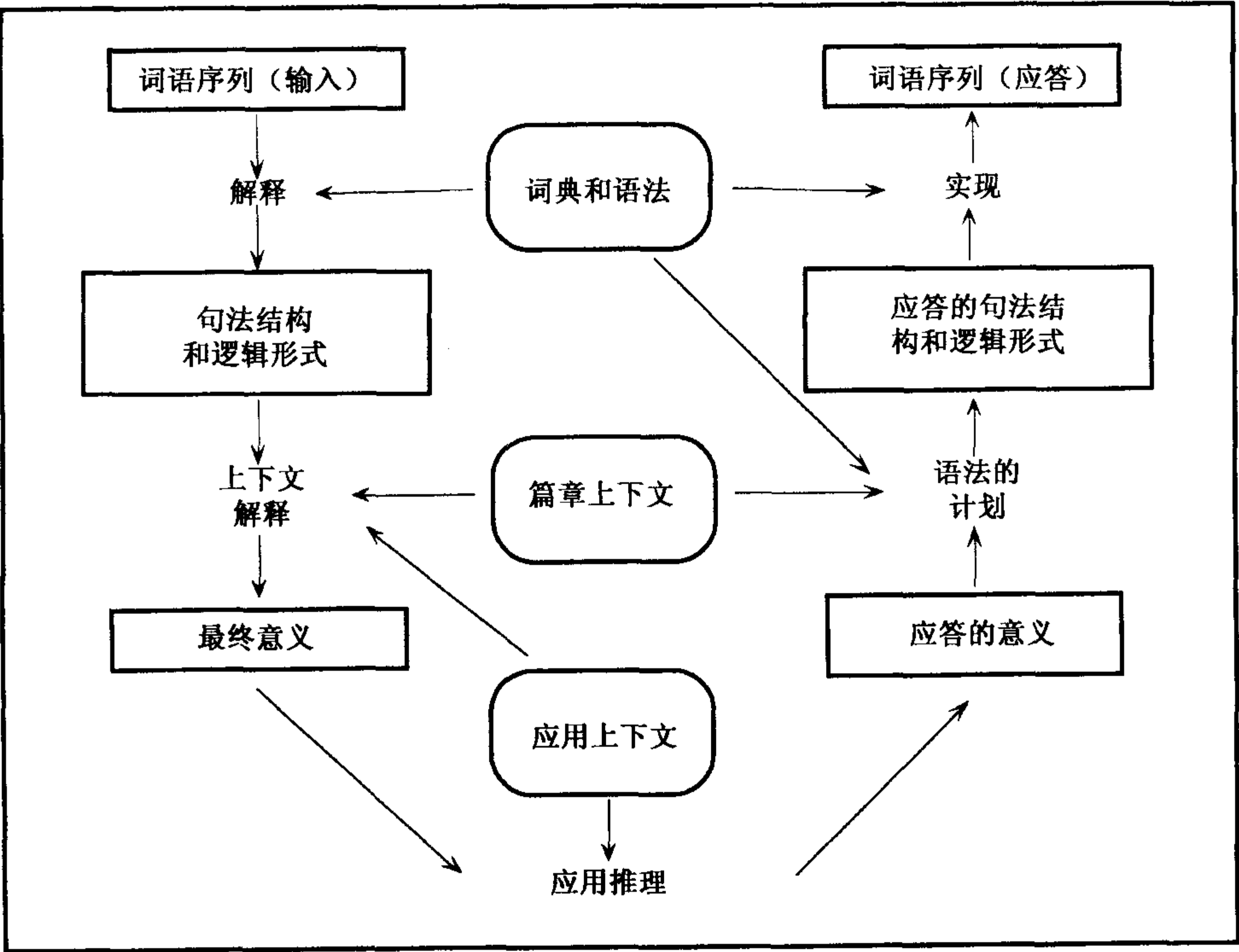


图 1.5 信息流

就像你能看到的,有一些解释过程将一种表示形式映射到另一种表示形式。比如,将一个句子映射到其句法结构和逻辑形式的程序称为句法分析器。它使用有关词语和词语意义的知识(词典)和一个定义合乎语法结构的规则集合(语法),以便给一个输入句子赋予句法结构和逻辑形式。一种可选的组织形式是首先进行句法处理,然后在得到的结构上进行语义解释。不过,把这二者结合起来有相当多的好处。这会导致可能的解释数量大大减少,因为提出的每一种解释必须同时在语法上和语义上都是合乎语法的。举例来说,考虑下面两个句子:

- 10. Visiting relatives can be trying.
- 11. Visiting museums can be trying.

这两个句子有相同的句法结构,因而也具有相同的结构歧义。在句子 10 中,主语可以是拜访(visiting)你的亲戚(relatives),或者是你拜访亲戚(visiting relatives)这个事件。这两种选择在语义上都是合理的,你需要使用上下文机制来决定合适的意义。不过,句子 11 只有一种可能的语义解释,因为博物馆(museums)不是一种能够访问其他人的实体,而是能被参观(be visited)。在一个将句法处理和语义处理分割开来的系统中,句子 11 会有两种句法解释,其中之一在后续的语义解释中会被消除。不过,如果将句法处理和语义处理结合起来,系统就可以在一开始解释短语“visiting museums”时检测到语义的异常,而永远不会在第一步中构造这个错误的句法

结构。虽然从本例来看节省的工作并不多,但在实际的应用中,一个合理的句子通常会有数百个可能的句子结构,其中的许多在语义上都是不通的。

继续看图 1.5,将句法结构和逻辑形式转化为最终意义表示的过程叫做上下文处理。这个过程包括处理诸如以下这样一些问题:确定名词短语——如指定性描述(举例来说,“the man”)和代词——所指向的物体,分析句子所传达的新信息的时态,确定说话者的意图(例如,“Can you lift that rock”是一个是否问句还是一个请求),以及所有在这个应用领域内正确解释这个句子所需要的推理过程。这个过程使用话语上下文知识(由当前句子之前的句子所决定)和应用领域知识来产生最终的表示。

然后,系统将要执行应用所需的恰当的推理任务。如果需要给用户一个应答,那么需要表达的意义就会被传递给系统的生成部分。它使用话语上下文知识(加上语法和词典中的信息)来规划表述的形式,然后通过一个实现过程映射为词语序列。当然,如果这是一个口语环境中的应用,则词语序列并不是最终的输入和输出,而是一个语音识别器的输出和一个语音合成器的输入。

虽然本书主要关注语言的理解,但要注意的是,同样的知识层次也在生成任务中被使用。举例来说,句法结构的知识被编写到语法中。而这部语法既可以用于识别给定句子的结构,也可以用于将一个结构实现为一个词语序列。支持这两个过程的语法被称为双向语法(bidirectional grammar)。虽然大部分研究者都同意双向语法是最理想的模型,但实际上使用的语法通常都为理解任务或者生成任务进行了裁剪。产生这种现象的原因是每一种任务都有各自不同的重点问题,而研究者一般只关注与其特定任务相关的那些问题。但是,即使在理解和生成中实际使用的语法并不相同,所使用的语法形式还是一致的。

1.7 小结

本书描述了自然语言理解的计算理论。理解系统最主要的特点就是它们对句子意义的表示进行计算并在推理任务中使用这些表示。相应于本书的三个主要部分,我们介绍了表示的三个主要层面。句法处理关心句子的结构特征,语义处理计算表达句子上下文无关意义的逻辑形式,而上下文处理将语言与应用领域相关联。

1.8 相关工作与深入阅读材料

要对本领域工作有更好的了解,可以阅读 Shapiro(1992)的两篇文章,标题是 *Computational Linguistics* 和 *Natural Language Understanding*。关于一些特定的子领域,如机器翻译、自然语言界面、自然语言生成等也有很多文章。Slocum(1985)给出了一个机器翻译的综述,Perrault 和 Grosz(1986)给出了一个自然语言界面的综述。

你可以在 Weizenbaum(1966)中找到关于 ELIZA 程序的描述,其中包括图 1.2 中给出的对话记录。使用模板匹配的基本技术在 PARRY 系统中有进一步的探讨,并且在 Schank 和 Colby(1973)一书中 Colby 的文章里有相关的描述。同一本书中还包括早期的自然语言系统的描述,包括 Winograd 和 Schank 的系统。另一个重要的早期系统是 LUNAR 系统,在 Woods(1977)中进行了全面的介绍。关于自然语言中人工智能方法的另一种观点,请参考 Winograd(1983)中的介绍。

1.9 习题

1. 【易】为 ELIZA 定义一个数据规则的集合,用这个集合可以生成图 1.2 中的对话的前七个回合。
2. 【易】给出下面句子所有的意义,对于每一种意义进行解释。对于每个句子,确定不同的意义是来自于结构歧义、语义歧义还是语用歧义。
 - a. Time flies like an arrow.
 - b. He drew one card.
 - c. Mr.Spock was charged with illegal alien recruitment.
 - d. He crushed the key to my heart.
3. 【易】假定说出这个句子的人是在抱怨汽车太冷,请从以下各个方面对下列句子进行分类:(i)句法是否正确;(ii)语义是否正确;(iii)语用是否正确。
 - a. The heater are on.
 - b. The tires are brand new.
 - c. Too many windows eat the stew.
4. 【中】实现一个 ELIZA 程序,这个程序可以使用你在习题 1 中写出的规则运行产生的那一段对话。如果不加入更多的规则,对于图 1.2 的对话中的后续几个句子你的程序有什么反应? 如果你在不同的上下文中,比如在酒吧的一次随意的交谈中,运行你的程序将会如何?

第一部分 句法处理

我们在导论中讨论过,本书将句子的理解分为三个阶段。本书的第一部分讨论第一个阶段——句法处理。其目标是判定句子的各种结构性成分,比如怎样将一个句子分解为短语,短语怎样分解为子短语,依次类推,一直分解到句子中所有的词,从而得到句子的实际结构。这些结构性关系对判定句子的意义非常重要,我们会在第二部分和第三部分给出句子意义判定的具体方法。

第一部分主要讨论两个问题。语言中可能存在哪些句子?采取什么样的形式化手段来进行判断,是我们关心的第一个问题。可以采用一套规则来给出这种信息,使用的规则集合称为语法。对此,在编撰自然语言语法时,既要考虑普遍的问题,即好的形式化体系应该具备哪些要素;同时,又要考虑采用什么样的语法规则能较好地表示出英语的句法。第二个问题研究的是,一旦知道语言的语法,如何分析出给定句子的结构。我们把这个过程称为句法分析。目前,有许多不同的句法分析算法,本书将介绍几种在本领域中最有影响的句法分析技术,并给出示例。

第2章介绍基本的英文句法背景知识。面向从来没有学过语言学的读者,我们主要介绍关键的概念和特性,这些在所有的句法理论中都很常见。第3章介绍几种表示语法的常用的形式化手段,并且详细地叙述基本的句法分析算法。第4章引入特征的思想,特征可以扩展基本的语法形式化体系,并能简洁明了地表达出自然语言多方面的属性。然后,第5章阐述了自然语言中一些较难的领域,特别是疑问句的处理、关系从句以及其他形式的语言移位现象。它表明,特征系统能做进一步的扩展,扩展后的语法可以处理这些复杂的句子。第6章探讨了歧义消解的相关问题。一些技术旨在研究更高效的表示方法,来存储多种分析结果;而另外一些技术的目标是,在句法分析的过程中,利用局部信息,在候选的分析结果集合中做出抉择。最后,在第7章中,我们讨论了一个相对较新的研究领域,即分析大规模的句子数据库并获取统计信息。这些信息可以用来判定歧义词语最可能的类别,还可用来判定结构歧义句子最可能的分析结果。

第2章 语言学背景知识:英语句法概要

本章提供英语句法基本结构方面的背景资料,主要针对那些从未上过语言学课程的读者。这些资料概括了主要的短语类别,并一一介绍了它们最重要的子成分。按照这种方式,本书中所有需要用到的基本词类在这里都做了介绍。本章并没有探讨任何与计算相关的内容,具有语言学背景的读者可以只进行快速浏览。学习第一部分的其他章节时,你可能需要参看本章的内容。

2.1节探讨词与词类的相关问题。2.2节讨论简单名词短语,在其基础上,我们会在2.3节叙述简单动词短语。2.4节探讨一些复杂的名词短语,在这些短语中存在一些内嵌句(embedded sentences),比如关系从句。本章的剩下部分简要地回顾了其他类型的短语,2.5节讲述形容词短语,2.6节讲述副词短语。

2.1 词语

乍一看,语言中最基本的单元似乎是词语。实际上,词语远远不是语言学研究的基本单元,它是由一些更基本的单元组成的集合。词语形态学研究的就是这些更基本的成分如何根据大致的语义单元来构建词语。新词有两种基本的组成方式,传统上,一般分为屈折式(inflexional form)和派生式(derivational form)两种。屈折式主要对词根(root form)进行变化,典型的方式是加上后缀(suffix),从而使词语以正确的形态出现在给定的句子中。在英语中,动词就是最好的例证。每一个动词都有自己的原型形式,具体使用的时候,它还会根据当前句子的主语和时态做相应的变化。例如,动词“sigh”加上s,ing和ed等形式的后缀,分别形成相应的动词形式“sighs”,“sighing”和“sighed”。这些新词都是动词,而且表示的基本语义相同。派生词语形态学主要研究其他形式的词语如何派生出新的词语。新词词类可能与它所包含的词词类完全不同。如:名词“friend”加上后缀ly,派生出形容词“friendly”。在更复杂的情况下,你还可以从形容词形式派生出名词形式“friendliness”。词语如何派生?句子的句法结构如何影响词语的形式并对词语进行约束?在这些方面,我们有很多有趣的研究课题。

传统上,语言学家会根据具体的使用特征将词语划分为不同的类别。在划分词类时,我们有两个相关方面的根据。一方面考虑词语在表达短语意思方面的贡献程度,另一方面考虑词语在真实句法结构中所起的作用。比如,我们讨论对象、概念或者位置时,你可以将那些表明其基本类型的词都划分为名词词类,而将那些进一步限定并修饰对象、概念或者位置的词划分为形容词词类。因此,“green”(绿色的)是一个形容词,而“book”(书)是名词,可形成短语“the green book”(这本绿色的书)和“green books”(绿色的书)。但是,事情并没有这么简单。“green”也可以充当名词的角色,如“That green is lighter than the other”(那种绿色比另一种绿色要浅)中的“green”(绿色)。同时,“book”也可以作为修饰语使用,如“the book worm”(书虫)。事实上,在某些情况下,大多数名词都可以作为修饰语。也许这些词类应该做一些归并,因为它们有很多重叠的地方。不过,我们也有一些其他的依据。考虑一下,什么样的词可以用来填充句子

“It’s so...”(这是如此的……),你可以说“It’s so green”(这是如此的绿),“It’s so hot”(这是如此的热),“It’s so true”(这是如此的真实),诸如此类。需要注意的是,尽管在“the book worm”中,“book”可以作为修饰语,但是你不能说“* It’s so book”(这是如此的书)。因此,存在两类修饰语,形容词性修饰语和名词性修饰语。

再来考虑一下形容词作为名词使用的情况,比如“the green”(绿色)。实际上,并不是所有的形容词都可以这么用。例如,如果给定的上下文中提到了热盘子和冷盘子,那么,在“The hot are on the table”(热盘子在桌上)之类的句子中,我们才可以使用名词短语“the hot”(热的)。“the green”指的是绿色,不同的是,在这里,“the hot”指代的是热盘子,它并不能指代“热度”(hotness)。根据这一点,你可以将形容词分为两个子类,其中一类可以直接描述概念或性质,另一类则不能。换句话说,你可以简单地说“green”存在作为名词或形容词的歧义,所以,它同时属于两个词类。“green”可以像其他名词一样使用,因此,我们给出的第二种解决办法似乎更直接。

采取类似的论证方式,我们可以将英语中的词语分为四大主类,它们都有助于表达句子的意思。这些词类分别是名词、形容词、动词和副词。句子都是在以这四大词类为中心的短语之上建立的。当然,在组成句子时,还有许多其他词类也是必不可少的,如冠词、代词、介词、语助词(particle)、量词和连词等。不过,这些词类都比较固定,语言中很少增加这类新词。另一方面,在语言的进化过程中,人们常常会引进新的名词、新的动词、新的形容词以及新的副词。因此,我们称这些词类为开放类词语(open class word),而其他词类为封闭类词语(closed class word)。

属于这四种开放型词类中的词一般都可以构成短语的基本框架,我们称这个词为短语的中心语(the head of the phrase)。在短语的描述中,中心语表示事物、活动或者性质的类型。以名词短语为例,中心语表示所描述对象的通用类型。例如:

the dog(这只狗)

the mangy dog(这只有疥癣的狗)

the mangy dog at the pound(这只在兽栏里有疥癣的狗)

它们都是描述狗类中一个特定对象的名词短语。其中,第一个短语描述的是所有狗类中的某个成员,第二个短语描述的是有疥癣狗类的某个对象,第三个短语描述的是在兽栏里而且有疥癣狗类的某个对象。在这些短语中,中心语都是“dog”(狗)。

类似地,有下面的形容词短语:

hungry(饥饿的)

very hungry(很饥饿的)

hungry as a horse(非常饥饿的)

它们都描述了饥饿的程度,每个短语的中心语都是“hungry”(饥饿的)。

在一些情况下,短语可以只包含单个的中心语。例如,词语“sand”(沙子)可以是一个名词短语,“hungry”(饥饿的)可以是形容词短语,“walked”(行走)可以是动词短语。除此之外,在大多数情况下,短语中心语都需要在后面附加另外的短语来表达所需的语意。例如,动词“put”(放置)不能单独组成一个动词短语;所以,下面的词语就不能组成一个有意义的句子:

* Jacd put.(Jack 放置。)

要想有意义,动词“put”必须接一个名词短语,再接一个表示位置的短语,如动词短语“put the dog in the house”(把狗放在房间里)。这些使语意更完整的短语或短语集合称为中心语的补足语(complement)。在刚才的例子中“put”(放置)是中心语,而“the dog in the house”(狗在房间里)是补足语。所有主要词类的中心语一般都需要补足语。图 2.1 给出了一些短语的例子,其中,粗体字表示的是中心语,斜体字表示的是补足语。在这一章的剩余部分,我们会更详细地研究这些不同类型的短语,看看它们的结构是如何组织的,它们如何为句子意思的表达服务。

名词短语	动词短语
The president <i>of the company</i> (这家公司的总裁)	looked <i>up the chimney</i> (顺着烟囱向上看)
His desire <i>to succeed</i> (他对成功的渴望)	believed <i>that the world was flat</i> (相信世界是平坦的)
Several challenges <i>from the opposing team</i> (来自竞争团队的几个挑战)	ate <i>the pizza</i> (吃比萨)
形容词短语	副词短语
easy <i>to assemble</i> (易于装配)	rapidly <i>like a bat</i> (迅速地像只蝙蝠)
happy <i>that he'd won the prize</i> (他很高兴能获奖)	intermittently <i>throughout the day</i> (整天间歇不断)
angry <i>as a hippo</i> (像只河马似的生气)	inside <i>the house</i> (在房子的里面)

图 2.1 中心语和补足语示例

2.2 简单名词短语的组成元素

名词短语(NP)用来表示事物,如物体、地点、概念、事件、性质等。最简单的 NP 仅包含单一的代词,如 he(他),she(她),they(他们),you(你),me(我),it(它),I(我)等。代词可以指代实物,例如下面的句子:

It hid under the rug.(它藏在地毯下。)

代词也可以指代事件,如句子:

Once I opened the door, I regretted *it* for months.
(我曾经打开过一次门,为此后悔了几个月。)

还可以用来指代属性,如句子:

He was so angry, but he didn't show *it*.(他非常气愤,但并没有表现出来。)

代词一般都不带修饰语,极少数的形式例外,如句子:

He who hesitates is lost.(犹豫不决的他迷失了。)

名词短语的另外一个基本形式包含的是一个名称或者说是一个专有名词,比如“John”或“Rochester”。在正规的书面英语中,这些名词需要采用大写字母形式。名字也可能由多个词语组成,如“New York Times”(纽约时报)和“Stratford-on-Avon”^①。

① 埃文河畔的斯特拉特福德,莎士比亚的故乡。——译者注

除了代词和专有名词,名词短语的中心语通常是一个普通名词。名词可以划分为两大类:

可数名词(count noun)——描述特定物体或者物体集合的名词。

物质名词(mass noun)——描述合成物或物质的名词。

可数名词因可计数而得名。可以说一只狗或许多只狗,一本书或许多本书,一个人群或许多人群。如果采用单一的可数名词来描述一整类的对象,则必须使用它的复数形式。因此,你可以说“Dogs are friendly”(狗们很友好),但不可以说“* Dog is friendly”。

物质名词不能计数。可以说一些水、一些小麦或者一些沙子。你如果试着对物质名词计数,那么就会最终改变了它的真实意思。如,“some wheat”(一些小麦)指一部分、具有一定数量的小麦,而“one wheat”(一种小麦)指小麦品种中的一种,而不是指谷物中的一粒小麦。物质名词描述整个种类的物质可以不用复数形式。这样,你就可以说“Water is necessary for life”(水是生命必不可少的),而不能说“* Waters are necessary for life”。

除了中心语之外,名词短语可能会在中心语之前包含指定词(specifier)或者修饰词(qualifier)。中心语指定对象的通用类别,而修饰词进一步描述这个通用类别,指定词则指明总共描述了多少个这样的物体。同时,指明被描述物体如何与说者和听者之间建立联系。指定词由序数词(比如第一,第二)、基数词(如一,二)和限定词(determiners)组成。其中,限定词可以分为下面几个通用类:

冠词(article)——词语 the, a, an。

指示词(demonstrative)——如词语 this(这), that(那), these(这些), those(那些)。

所有格(possessive)——带后缀's 的名词短语,如 John's(约翰的), the fat man's(这个胖男人的), 以及代词所有格,如 her(她的), my(我的), whose(谁的)。

wh 限定词(wh-determiner)——用在疑问句中的词语,如 which(哪一个), what(什么)。

数量限定词(quantifying determiner)——如词语 some(一些), every(每一个), most(大多数), no(没有一个), any(任何一个), both(两个都)以及 half(一半)。

简单的名词短语最多可能只有一个限定词、一个序数词和一个基数词。也可能三种都包含,如“the first three contestants”(前三个竞争者)。这个规则也存在一些例外的情况,通常和一些数量限定词有关,如“many”(一些),“few”(很少一些),“several”(几个)和“little”(很少的)。这些词都可以将冠词放在前面,并生成名词短语,如“the few songs we knew”(我们所知为数不多的歌曲)。基于这一点,你可以将数量限定词进一步细分为两个子类,其中一个子类允许这种形式,而另一个则不允许。不过,对于我们当前的目标来说,现在这种粗略的划分已经足以满足需求了。

一个名词短语中的修饰词常常出现在指定词(如果有的话)之后,中心语之前。修饰词包括形容词以及可以作为修饰语的名词。下面是一些更精确的定义:

形容词——表示事物性质的词语,然而,并不是指代性质本身。(举例来说,“angry”是一个形容词,表示的是对某件事情生气的性质。)

名词修饰语——修饰另一个名词的可数或物质名词,如短语“the cook book”(烹调的书)或“the ceiling paint can”(天花板油漆罐)。

在讨论别的结构之前,我们先考虑一下名词的不同屈折形式,再看看如何在英语中实现这些形式。在前文中我们已经提到过,名词有两种形式——单数和复数。代词采用什么样的形式主要根据人称(第一人称、第二人称和第三人称)和性(阳性、阴性和中性)。每一种区别都反映了语言的一种系统性特征。这种区别在某些语言中是显性的,如拉丁语。不过,这种区别在另一些语言中是隐性的。例如,在法语中,名词通过词性来区分。在英语中,除了在极少数情况下,很多这种区别都没有显性的表示,而代词形式则是显性表示的最好示例。代词按照数、人称、性和格(也就是它们是否用做所有格、主语或者宾语)进行区分,如图 2.2 ~ 图 2.4 所示。

数	第一人称	第二人称	第三人称
单数	I(我)	you(你)	he(他,阳性)
			she(她,阴性)
			it(它,中性)
复数	we(我们)	you(你们)	they(他们,它们)

图 2.2 代词系统(用做主语)

数	第一人称	第二人称	第三人称
单数	my(我的)	your(你的)	his, her, its(他的,她的,它的)
复数	our(我们的)	your(你们的)	their(他们的,它们的)

图 2.3 代词系统(所有格)

数	第一人称	第二人称	第三人称
单数	me(我)	you(你)	him(他)her(她)it(它)
复数	us(我们)	you(你们)	them(他们,它们)

图 2.4 代词系统(用做宾语)

2.3 动词短语与简单句

NP 用来指代事物,而句子(S)则用来断言、询问或者命令。你可以断言某个句子是对的,可以询问某个句子是否真实,或者命令某个人按照句子的指示做事。句子的使用方式称为语气(mood)。图 2.5 列出了四种基本的句子语气。

语气	示例
陈述句(或者断言句)	The cat is sleeping. (猫正在睡觉。)
一般疑问句	Is the cat sleeping? (猫正在睡觉吗?)
特殊疑问句	What is sleeping? (什么在睡觉?)或者 Which cat is sleeping? (哪只猫在睡觉?)
祈使句(或者命令句)	Shoot the cat! (向那只猫射击!)

图 2.5 句子的基本语气

简单陈述句包含一个 NP 作为主语,后面再跟一个动词短语(VP)作为句子的谓语。简单 VP 可能包含一些副词修饰语,再跟着动词中心语及其补足语。图 2.6 给出了动词的五种可能的形式,每个动词一定要以其中的一种形式出现。

形式	例子	用法示例
原型	hit, cry,	<i>Hit</i> the ball! (击球!)
	go, be	I want to <i>go</i> . (我想走。)
一般现在形式	hit, cries,	The dog <i>cries</i> every day. (狗每天都叫。)
	go, am	I <i>am</i> thirsty. (我很渴。)
一般过去形式	hit, cried,	I <i>was</i> thirsty. (我原来很渴。)
	went, was	I <i>went</i> to the store. (我去过那家商店。)
现在分词形式	hitting, crying,	I'm <i>going</i> to the store. (我正在去那家商店。)
	going, being	<i>Being</i> the last in line aggravates me. (在队伍的最后一个使我恼怒。)
过去分词形式	hit, cried,	I've <i>been</i> there before. (我以前去过那里。)
	gone, been	The cake was <i>gone</i> . (蛋糕不见了。)

图 2.6 动词的五种形式

动词可以分为几种不同的类别:助动词(auxiliary verb),如 be, do, have;情态动词(modal verb),如 will, can, could;以及主要动词(main verb),如 eat, ran, believe。助动词和情态动词常常将动词短语作为补足语,这样就可以生成一个动词序列。其中,每个动词都是它所在动词短语的中心语。这些动词序列用来形成不同时态的句子。

时态体系判定句子叙述的命题什么时候为真。时态体系很复杂,图 2.7 只给出了基本的形式。另外,动词可以采用进行时态。与图 2.7 中给出的时态相对应,图 2.8 给出了这些复杂的进行时态。每种进行时态都由 be 动词的一般时态结构,加上后面的现在分词组成。

时态	动词顺序	示例
一般现在时	一般现在形式	He <i>walks</i> to the store. (他去那家商店。)
一般过去时	一般过去形式	He <i>walked</i> to the store. (他去过那家商店。)
一般将来时	will + 不定式	He <i>will walk</i> to the store. (他要去那家商店。)
现在完成时	have 的现在式 + 过去分词	He <i>has walked</i> to the store. (他去那家商店了。)
将来完成时	will + have 的不定式 + 过去分词	I <i>will have walked</i> to the store. (我到时候就去那家商店。)
过去完成时	have 的过去式 + 过去分词	I <i>had walked</i> to the store. (那时候,我就曾去过那家商店。)

图 2.7 基本时态

动词词组的第一个词也可以表达人称和数量的信息。动词短语的人称和数量必须和作为主语的名词短语一致。有一些动词几乎对所有的不同情况都要进行区别处理,不过,大部分动词只区分第三人称单数(一般加后缀 s)。图 2.9 给出了一些例子。

时态	结构	示例
现在进行时	be 的现在式 + 现在分词	He is walking. (他正在散步。)
过去进行时	be 的过去式 + 现在分词	He was walking. (他那时候在散步。)
将来进行时	will + be 的动词不定式 + 现在分词	He will be walking. (他到那个时候在散步。)
现在完成进行时	have 的现在式 + be 的过去分词 + 现在分词	He has been walking. (他到那个时候已经在散步。)
将来完成进行时	will + have 的过去式 + be 的过去分词 + 现在分词	He will have been walking. (他到那个时候将在散步。)
过去完成进行时	have 的过去式 + be 的过去分词 + 现在分词	He had been walking. (他到那个时候已经散过步了。)

图 2.8 进行时态

	第一人称	第二人称	第三人称
单数形式	I <i>am</i> (我是)	you <i>are</i> (你是)	he <i>is</i> (他是)
	I <i>walk</i> (我走)	you <i>walk</i> (你走)	she <i>walks</i> (她走)
复数形式	we <i>are</i> (我们是)	you <i>are</i> (你们是)	they <i>are</i> (他们是)
	we <i>walk</i> (我们走)	you <i>walk</i> (你们走)	they <i>walk</i> (他们走)

图 2.9 动词的人称/数形式

2.3.1 及物性与被动语态

动词序列中的最后一个动词称为主要动词,它们一般都是开放型动词。依据不同的动词,我们允许它采取各种不同形式的补足语结构。比如,某些动词可以单独存在而无需补语成分,我们将这些动词称为不及物动词 (intransitive verb),不及物动词的例子有“laugh”(如 Jack laughed;Jack 笑)和“run”(如 He will have been running;他那个时候肯定已经在跑了)。另一个常见的补足语形式需要在动词的后面接一个名词短语。这种动词称为及物动词(transitive verb),如“find”(例如,Jack found a key;Jack 找到了一把钥匙)。需要注意的是,“find”不能作为不及物动词使用(例如,“* Jack found”就是一个不合理的句子);同时,“laugh”不能作为及物动词使用(例如,“* Jack laughed a key”也是一个不合理的句子)。另外一方面,“run”这样的动词既能作为及物动词也能作为不及物动词。不过,在这两种情况下,它表达的意思截然不同(例如,“Jack ran”;Jack 跑步和“Jack ran the machine”;Jack 操作这台机器)。

及物动词还允许另一种形式的动词词组,即被动形式。被动形式由助动词 be 后面接一个过去分词构成。在被动语态里,通常在宾语位置上的名词短语会在主语的位置上使用,具体的例子见图 2.10。注意,句子的时态仍由动词序列中的第一个动词决定。而且,在被动式句子中,从语义上分析,第一个名词短语看起来好像是动词的宾语,但它仍然是句法上的主语。检查代词的形式就可以看出这一点。如“I was hit”(我被打中了)是正确的,而“* Me was hit”则不对。而且,动词的时态和数也应该与句法结构上的主语相一致。因此,你可以说“I was hit by them”,但不能说“* I were hit by them”。

主动句	相应的被动句
Jack saw the ball. (Jack 看见了这个球。)	The ball was seen by Jack. (这个球被 Jack 看见了。)
I will find the clue. (我会发现线索的。)	The clue will be found by me. (线索会被我发现的。)
Jack hit me. (Jack 打了我。)	I was hit by Jack. (我被 Jack 打了。)

图 2.10 主动句与相应的被动句

在句子中,有些动词允许接两个名词短语。例如,“Jack gave Sue a book”(Jack 给了 Sue 一本书)或“Jack found me a key”(Jack 帮我找了一把钥匙)。在这样的句子中,第二个名词短语是早期提到的宾语,我们有时称之为直接宾语,而另外的名词短语称为间接宾语。一般来说,在间接宾语前加上相应的介词,我们可以得到另外一个对等的句子,如“Jack gave a book to Sue”或“Jack found a key for me”。

2.3.2 语助词

一些动词形式由动词和一个附加词组成,我们称这个附加词为语助词。语助词一般都会与下一节要探讨的介词存在交叉融合的地方,这样的例子包括“up”,“out”,“over”和“in”。以“look”,“take”或者“put”之类的动词为例,将动词和语助词结合,你可以构造出不同形式的动词(例如,look up,look out,look over,等等)。在一些句子中,语助词与介词的差别会导致对同一句子的不同解读。比如,在“look over the paper”中,如果把“over”当成一个语助词(动词为“look over”),那么该短语可以理解为翻阅论文。相反,如果把“over”看成介词(其中动词为“look”),那么该句又可以理解为看着纸上边或者后边的其他某样东西。

如果动词的宾语是代词,那么,你就可以严格地区分出语助词和介词。在动词-语助词的句子中,代词必须放在语助词之前,比如句子“I looked it up”(我查询它)。而解读为介词的时候,代词必须放在介词之后,比如“I looked up it”(我向上看它)。语助词也可以带着一个名词短语作为宾语,因此,你可以说“I gave up the game to Mary”(我把游戏让给 Mary 玩)或者“I gave the game up to Mary”(我把游戏让给 Mary 玩)。而这对于介词而言是不允许的,比如,我们不可说“* I climbed the ladder up”。

2.3.3 从句补语

许多动词允许从句作为其补语。从句具有和句子相同的大部分性质,它可以有主语、时态,也可以以被动语态出现。普通从句的形式通常是一个由引导词 that 引导的句子,比如“that Jack ate the pizza”(Jack 吃比萨)。这个从句可以用谓词 S[that]来表示,它表示的是 S 结构的一个专门子类。这个从句也可以作为动词“know”的补语出现,比如“Sam knows that Jack ate the pizza”(Sam 知道 Jack 吃比萨了)。当然,从句中也可能采用被动语态,比如“Sam knows that the pizza was eaten by Jack”(Sam 知道比萨被 Jack 吃了)。

另一种从句类型和动词的不定式有关。简单而言,VP[inf]形式的从句就是一个以不定式形式开始的动词短语,比如在“Jack wishes to eat the pizza”(Jack 希望吃比萨)中,动词“wish”的补语“to eat the pizza”就是个不定式。另外,在有“for”短语指明逻辑主语的前提下,不定式S[inf]也是可能的,比如“Jack wishes for Sam to eat the pizza”(Jack 希望 Sam 吃比萨)。

另外一种重要的从句类型是那些以特殊疑问词为补语标记的句子,特殊疑问词有“who”(谁),“what”(什么),“where”(哪里),“why”(为什么),“whether”(是不是)和“how many”(多少),等等。这些疑问从句 S[WH]可以用来充当“know”等动词的补语,比如“Sam knows whether we went to the party.”(Sam 知道我们是否参加这个派对。)和“The police know who committed the crime.”(警方知道是谁犯的罪行。)

2.3.4 介词短语补语

许多动词需要与特定介词短语(PP, preposition phrase)相关的补语。动词“give”就可以跟一个由名词短语 NP 和介词 to 构成的补语,比如“Jack gave the book to the library”(Jack 把这本书还给了图书馆)。在这里不能使用其他的介词短语。考虑下面的句子:

* Jack gave the book from the library. (Jack 给了一本从图书馆借来的书。只有当“from the library”修饰“book”时,这句话才是正确的。)

与此形成对比的是,像“put”这样的动词可以带任何一个描述位置的介词短语,比如:

- Jack put the book in the box. (Jack 把书放到了盒子里。)
- Jack put the book inside the box. (Jack 把书放到了盒子里面。)
- Jack put the book by the door. (Jack 把书放在了门的旁边。)

为了说明这一点,我们允许采用补语的某种格式来说明特定介词的介词短语。因此,动词“give”的补语形式为 NP + PP[to]。类似地,动词“decide”的补语形式为 NP + PP[about]。同时,动词“blame”的补语形式为 NP + PP[on]。例如,“Jack blamed the accident on the police”(Jack 把这个事故归咎于警方)。

像“put”这样的动词可以带任何一个表示地点的短语(补语形式为 NP + 位置),这一点在英语中也是很常见的。虽然位置都是典型的介词短语,但也可以是名词短语,比如“home”(家);或者是语助词,比如“back”(回)或者“here”(这里)。描述位置的短语和描述运动轨迹的短语存在着本质的区别,尽管位置短语可以同时解释为这两种形式。在某些情况下,我们可以把它们区分开来。比如,以“to”开头的介词短语一般表示运动轨迹。因此,不可以和“put”这样的动词在一起使用,“put”需要位置短语做补语。[比如, * I put the ball to the box. (我把球放到盒子里去。)]在第 4 章中,我们会对这种差异做进一步的探讨。

图 2.11 总结了许多在英语中发现的动词补语结构,完整的列表包含有 40 多种不同的形式。需要注意的是,在这里,对于每种不同形式,我们给出的典型例子均使用了不同的动词。实际上,大多数动词都允许几种不同形式的补语。

动词	补语结构	示例
laugh(笑)	空(不及物)	Jack laughed. (Jack 笑了。)
find(找到)	NP(及物)	Jack found a key. (Jack 找到了一把钥匙。)
give(给)	NP + NP(双宾语)	Jack gave Sue the paper. (Jack 把那张纸给了 Sue。)
give(给)	NP + PP[to]	Jack gave the book to the library. (Jack 把那本书还给了图书馆。)

图 2.11 英语中一些常用动词的补语结构

动词	补语结构	示例
reside(居住)	位置短语	Jack resides in Rochester. (Jack 住在 Rochester。)
put(放置)	NP + 位置短语	Jack put the book inside. (Jack 把书放进去了。)
speak(说,讨论)	PP[with] + PP[about]	Jack spoke with Sue about the book. (Jack 和 Sue 讨论那本书。)
try(尽力)	VP[to]	Jack tried to apologize. (Jack 尽力道歉。)
tell(告诉)	NP + VP[to]	Jack told the man to go. (Jack 让那个男人走。)
wish(希望)	S[to]	Jack wished for the man to go. (Jack 希望那个男人走。)
keep(保持)	VP[ing]	Jack keeps hoping for the best. (Jack 依然持有最好的愿望。)
catch(抓到)	NP + VP[ing]	Jack caught Sam looking in his desk. (Jack 抓到 Sam 在查看他的书桌。)
watch(观察,看)	NP + VP[原形]	Jack watched Sam eat the pizza. (Jack 看到 Sam 在吃比萨。)
regret(遗憾)	S[that]	Jack regretted that he'd eaten the whole thing. (Jack 遗憾的是他把所有东西都吃了个精光。)
tell(告诉)	NP + S[that]	Jack told Sue that he was sorry. (Jack 告诉 Sue 他很抱歉。)
seen(似乎,好像)	ADJP	Jack seems unhappy in his new job. (Jack 对他新工作的感受似乎不是很开心。)
think(认为)	NP + ADJP	Jack thinks Sue is happy in her job. (Jack 认为 Sue 对她的工作很满意。)
know(知道)	S[WH]	Jack knows where the money is. (Jack 知道钱放在哪里。)

图 2.11 英语中一些常用动词的补语结构(续)

2.4 再论名词短语

2.2 节介绍了简单的名词短语,本节将考虑更复杂的名词短语形式,复杂的名词短语组成成分中可能包括句子或者动词短语。

2.2 节中所有例子的中心语都不带补语,实际上,许多名词都可能带有补语。这些补语大部分都需要特定的介词短语。举例来说,名词“love”有一种补语形式 PP[of],比如“their love of France”(他们对法国的热爱);名词“reliance”有一种补语形式 PP[on],比如“his reliance on handouts”(他信赖那些分发的印刷品);名词“familiarity”有一种补语形式 PP[with],比如“a familiarity with computers”(熟悉计算机)。

许多名词可以接动词不定式作为补语,这类名词如“desire”(愿望),“reluctance”(不愿)和“research”(研究),名词短语如“his desire to release the guinea pig”(他想释放豚鼠的愿望),“a reluctance to open the case again”(对再公开这件案子的不情愿),“the doctor’s research to find a cure for cancer”(大夫进行的找到治疗癌症方法的研究)。实际上,这些名词也可以接不定式 S[inf]形式的补语,例如,“my hope for John to open the case again”(我对 John 能重新公开此案的希望)。

名词短语也可以是从句,我们在上一节中讲述动词的补语时,已经对从句做了介绍。例如,that 从句 S[that]可以作为一个句子的主语,如“That George had the ring was surprising.”(George 有戒指这件事很让人惊奇。)动词短语不定式(VP[inf])和句子的不定式形式(S[inf])也可以当做名词短语使用,例如句子“To own a car would be delightful.”(拥有一辆车是件让人高兴的事情。)以及“For us to complete a project on time would be unprecedented.”(让我们按时完成这个

项目可是无先例的。)此外,动名词形式的形容词(*gerundive*)(VP[ing]和S[ing])也可以当做名词短语,比如句子“Giving up the game was unfortunate”(放弃这场比赛是很遗憾的)以及“John’s giving up the game caused a riot”(John 放弃这场比赛会导致骚乱)。

关系从句是在名词短语中作为修饰语使用的句子形式。这些从句常常由“who”(谁),“which”(哪一个),“that”(那个)等关系代词(*relative pronoun*)来引导,例如:

The man *who gave Bill the money*...(那个给了 Bill 钱的男人……)

The rug *that George gave to Ernest*...(那块 George 给 Ernest 的垫子……)

The man *whom George gave the money to*...(George 付给钱的那个男人……)

在这些关系从句中,除了缺少名词短语外,从句中内含的句子和正常的句子结构别无两样。如果用该从句修饰的名词短语置换这个空缺,那么句子就完整了,而且其传达的意思和从句完全相同。在前面的三个例子中,缺少的名词短语分别出现在从句的主语位置、宾语位置以及介宾位置。在这些例子中删除关系介词,并且把缺少的名词短语补上,即可得到下面的结果:

The man gave Bill the money.(那个男人把钱给了 Bill。)

George gave *the rug* to Ernest.(George 把那块垫子给了 Ernest。)

George gave the money to *the man*.(George 把钱给了那个男人。)

和前面正确的例子一样,关系从句可以按照正常句子一样的方式进行改造。在这里,我们特别给出上述句子的被动形式,结果如下:

Bill was given the money by the man.(Bill 从那个男人那里得到了钱。)

The rug was given to Ernest by George.(那块垫子由 George 给了 Ernest。)

The money was given to the man by George.(钱由 George 给了那个男人。)

相应地,这些句子可以转换为被动形式的关系从句,如下所示:

The man *Bill was given the money by*...(给了 Bill 钱的那个男人……)

The rug *that was given to Ernest by George*...(由 George 给了 Ernest 的那块垫子……)

The man *whom the money was given to by George*...(从 George 那里得到钱的那个男人……)

需要注意的是,有一些关系从句并不需要由关系代词来引导,关系代词经常被省略。这样的关系从句叫基本关系从句(*base relative clause*),比如名词短语“the man George gave the money to”(那个 George 给他钱的男人)。而另外一种形式把关系代词和助动词 *be* 形式删除,形成简化的关系从句(*reduced relative clause*),比如名词短语“the man given the money”(得到钱的那个男人),和名词短语“the man who was given the money”的意思相同。

2.5 形容词短语

在前面的几个例子里,你已经见过一些只包含单个形容词的简单形容词短语(*ADJP, adjective phrase*)。其实,也可能存在更复杂的形容词短语,形容词可以和动词一样带很多种不同的补语。这些补语包括特定的介词短语,比如形容词“pleased”(高兴的,满足的)就可以接形式为 PP[with]的介词短语作为补语(例如,“Jack was pleased with the prize”,Jack 很高兴能获得这个奖

项)。又如,“angry”(生气的,愤怒的)可以接形式为 PP[at]的介词短语作为其补语。“angry”也可以接形式为从句 S[that]的补语,比如“Jack was angry that he was left behind”(Jack 对于自己被拉下很是恼火)。其他的形容词还可以带不定式形式作为补语,例如,形容词“willing”(愿意的)就可以接 VP[inf]形式的不定式补语,例如,“Jack seemed willing to lead the chorus”(Jack 好像很愿意领唱)。

这些较复杂的常见形容词短语一般是作为“be”或者“seem”等动词的补语成分,或者跟在名词短语中心语之后。一般来说,它们不能用在名词短语中心语之前作为其修饰语。(比如,考虑一下这些短语“* the angry at the committee man”,对委员会人员感到愤怒的男人,“the angry man”,愤怒的男人,和“the man angry at the committee”,这个对委员会很恼怒的男人。)

我们也可以在形容词短语的中心语之前加上一个表示程度的修饰语。例如,形容词短语“very angry”(很生气)或者“somewhat fond of Mary”(有点喜欢 Mary)。同时,我们也允许采用更加复杂的程度修饰成分,比如,“far too heavy”(实在是太重了)和“much more desperate”(非常失望)。最后,应该知道某些结构有自己的程度修饰语,而这些修饰语又可能有它们自己的补语形式。比如,“too stupid to come in out of the rain”(太愚蠢而不能从雨中走出来),“so boring that everyone fell asleep”(烦得大家昏昏欲睡)和“as slow as a dead horse”(慢得像死马一样)。

2.6 副词短语

你已经在前面介绍的几种结构中见过副词的使用情况,比如程度副词(例如“very”,非常,很;“rather”,相当;“too”,太)和地点位置副词(例如“here”,这里;“everywhere”,到处)。其他形式的副词可以表示做事的方式(例如“slowly”,慢慢地;“hesitantly”,犹豫地),事件发生的时间(例如“now”,现在;“yesterday”,昨天),或者事件发生的频率(例如“frequently”,频繁地;“rarely”,极少地;“never”,从来不)。

副词可以在句子中几个不同的位置上出现,其中包括句子的开始位置(例如,“Then, Jack will open the drawer”,然后,Jack 要打开抽屉),动词序列中(例如,“Jack then will open the drawer”,然后,Jack 要打开抽屉;“Jack will then open the drawer”,然后,Jack 要打开抽屉),以及句子的末尾(例如,“Jack opened the drawer then”,然后,Jack 打开了抽屉)。不过,对于不同的副词来说,副词究竟可以出现在什么样的位置,这种恰当的约束关系是很不一样的。

除了这些副词之外,副词性修饰语可以采用多种多样的结构来构造。它可以是表示位置(比如“in the box”,盒子里面)或者方式(比如“in great haste”,很匆忙之间)的介词短语;此外,还可以是表示频率的名词短语(比如“every day”,每天);以及表明时间的从句(比如“when the bomb exploded”,当炸弹爆炸的时候)。然而,这些副词短语通常只能在句子的开始或者结束的位置上出现。例如,可以说“Every day John opens his drawer”(每天 John 都开抽屉)或者“John opens his drawer every day”(John 每天都开抽屉),但是不可说“* John every day opens his drawer”。

由于副词构成形式的多样性,一般来讲,相对于句法形式而言,从功能上来讨论副词短语(ADVP, adverbial phrase)更加有用。因此,可以按照各自的形式依次探讨状态、时间、延续、位置、程度和频次副词短语。在前面,已经讨论过位置副词短语和程度副词短语;在这里,接着介绍其他的副词短语。

时间副词短语来源十分广泛,形式多样,其中包括副词性语助词(比如“now”,现在)、名词短语(比如“today”,今天;“yesterday”,昨天)、介词短语(例如“at noon”,在正午;“during the flight”,飞行期间)以及从句(例如“when the clock struck noon”,当钟敲正午十二点的时候;“before the flight started”,起飞前)。

频次副词短语的形式也是多种多样的,其中包括语助词(比如,“often”,常常)、名词短语(比如,“every day”,每天)、介词短语(比如,“at every party”,在每次的派对上)以及从句(比如,“every time that John comes for a visit”,每次 John 来访的时候)。

延续性副词短语最常见的形式是介词短语(例如,“for three hours”,历时三个小时;“about 20 feet”,20 英尺左右)和从句(例如,“until the moon turns blue”,直到月色变蓝)。

状态副词短语以多种形式出现,其中包括语助词(比如,“slowly”,慢慢地)、名词短语(比如,“this way”,这种方式)、介词短语(比如,“in great haste”,很匆忙中)以及从句(比如,“by holding the embers at the end of the stick”,采取保留棍子末端灰烬的办法)。

接下来,我们继续分析一下。一般而言,大多数副词都是作为句子描述动作或者状态的修饰语。基于这一点,一个很自然的问题就出现了:如何区分动词的补语和副词短语呢?它们的一个不同之处在于副词短语总是可选的,在句子中可要可不要。因此,你可以从句子中将副词短语删除,而剩下的句子仍然可以表达与原句相近的意思(显然,会丢失副词所贡献的含义)。考虑下面的两个句子:

Jack put the box by the door. (Jack 把盒子放在门旁。)

Jack ate the pizza by the door. (Jack 在门旁吃了比萨。)

在第一个句子中,介词短语显然是一个补语,因为删除了该介词短语之后,生成的句子为“* Jack put the box”(Jack 放盒子),而这是一个没有实际意义的句子。与此截然不同的是,如果将第二个句子中的介词短语删除,影响却微乎其微。因为相对于“Jack ate the pizza by the door”而言,“Jack ate the pizza”(Jack 吃了比萨)描述的情形只是一种更一般化的叙述。

2.7 小结

在这一章里,我们介绍了主要的英语短语结构,即:名词短语、句子、介词短语、形容词短语和副词短语。这些介绍的内容将组成句法的构建单元,我们会在接下来的章节中一一介绍具体的句法结构。

2.8 相关工作与深入阅读材料

Baker(1989)对英语句法做了非常好的总结和概括。最全面的综合资料是那些尝试描述英语整个结构的书籍,比如 Huddleston(1988),Quirk 等(1972)以及 Leech 和 Svartvik(1975)。

2.9 习题

- 1.【易】课文中描述了几个不同的词类区分测试实例。比如,名词可以出现在形式为“I saw the X”(我看见X)的句子中。同样,副词可以出现在“It's so X”(是如此的X)形式

的句子中。请另外给出一些测试实例,来区分这些形式,并区分可数名词与物质名词。判断下面的词语哪个可以作为形容词、可数名词或者物质名词。假如词语存在歧义,给出该词所有可能的用法:

milk, house, liquid, green, group, concept, airborne

牛奶,房屋,液体,绿色,组,概念,空运的,空气传播的,空降的

2. 【易】在下面的句子中,请辨别出每个主要的短语(名词、动词、形容词或者副词短语)。对于每个短语,指出短语的中心语以及中心语的补语。注意区分补语和可选的修饰语:

The man played his fiddle in the street.

(这个男人在街上拉小提琴。)

The people dissatisfied with the verdict left the courtroom.

(对判决不满的人们离开了法庭。)

3. 【易】连接测试是判定词语和短语句法角色的一种很有用的测试方法。连接词,比如 and 和 or,往往可以把两个类型相同的短语连接起来。可以连接名词,比如“the man and woman”(男人和女人);也可以连接名词短语,例如,“the angry men and the large dogs”(愤怒的男人和巨型狗);连接形容词短语,例如,“the cow, angry and confused, broke the gate”(带着愤怒和慌乱,奶牛破门而入)。在下面的每个句子里,判别出连接的短语类型,并用下划线表示出每个短语。

He was tired and hungrier than a herd of elephants.

(他比一群大象更累更饿。)

We have never walked home or to the store from here.

(我们从来不步行回家或者从这里去商店。)

The dog returned quickly and dropped the stick.

(这只狗很快就回来了,还丢了那根棍子。)

4. 【易】使用本章中的术语,请详细解释为什么下面的每个句子都是错误的。同时,需要具体指出它们究竟违反了本章给出的哪条规则。

a. He barked the wrong tree up.

b. She turned waters into wine.

c. Don't take many all the cookies!

d. I feel floor today.

e. They all laughed the boy.

5. 【易】把下面的动词分为不及物动词、及物动词、双宾语动词(即带两个名词短语作为补语)。如果动词的形式超过一种,应该给出每种可能的类别。对于每种形式,给出一个例句来证实你的分析:

a. cry

b. sing

c. donate

d. put

6. 【易】使用动词“to be”(是),按照图 2.7 中给出的 6 种基本时态,以及图 2.8 中给出的 6 种进行时态,给出不同形式的例句。
7. 【易】使用动词“donate”(捐赠,赠予),按照图 2.7 中给出的 6 种基本时态给出该词的被动形式。
8. 【易】使用图 2.11 中定义的补语结构形式,按照允许使用的不同补语结构,对下面的动词进行分类。

give, know, assume, insert

请另外给出一个图中没有列出的补语结构示例,要求这里的某个动词允许使用该补语结构。

9. 【易】找出 5 个能够带间接宾语,而且没有在本章中讨论过的动词。对于每个动词,请采用介词短语替代间接宾语,并要求能表达出与原句相同的意思。在尽可能广的范围内进行尝试,看看你能否找到一个不能用介词 for 或者 to 来替代的句子?
10. 【中】特殊疑问句就是那些使用一类疑问词的问句,这些疑问词包括“what”(什么),“where”(哪里),“who”(谁),“when”(什么时候),“whose”(谁的),“which”(哪一个)和“how”(怎么)。对于其中的每一个词,请给出该词可以使用的词类(比如动词、名词、名词词组、形容词、量词、介词短语,等等)。用一些例子来证实你所给的每种分类。在必要的时候,可以同时使用正面和反面的论据。(比如,“这个词可以归为其中的某一个词类,原因是……”或者“尽管看上去似乎可以,实际上这个词并不能归为这个词类,因为……”)

第3章 语法及其分析

一个句子的句法结构如何计算？为了解决这个问题，必须综合考虑两个方面的因素，一是语言的语法，二是句法分析技术。其中，语法指的是语言所允许的、合法句子结构的形式化定义；句法分析技术指的是依据语法规则来确定句子结构的分析方法。本章主要讨论各种简单语法的定义方法，并介绍一些基本的句法分析技术。然后，我们将会在第4章介绍句法表示的构建方法，而句法表示对后面要讨论的语义解释非常有用。

一开始，我们首先讨论表示自然语言结构的标记体系以及简单的句法分析技术。3.2节描述一部好的语法所应该具备的一些特征。随后，我们会在3.3节介绍一个简单的句法分析技术，同时引入了将句法分析看成搜索过程的思想。3.4节介绍一种采用 chart 结构来构造高效句法分析器的方法。然后，3.5节将介绍另外一种基于转移网络的语法表示体系。剩下的几节我们将讨论一些高难度的研究课题，这部分作为可选内容。3.6节介绍一种自顶向下的 chart 句法分析器，它综合了自顶向下与自底向上方法各自的优点。3.7节介绍有限状态转录机的相关理论，并讨论它在词语形态处理中的应用。3.8节引入逻辑语法的思想，并展示了将上下文无关文法转换成 PROLOG 语言断言的过程。

3.1 语法与句子结构

本节讨论描述句子结构的方法，并进一步探究语言中所有合法结构的描述方法。一个句子如何拆分成主要的子成分，这些子成分又如何继续拆分，树结构表示是描述这些过程最常用的方法。图3.1给出了句子“John ate the cat”(John 吃猫)的树结构。这幅图表示的意思可以解读为：句子(S)包含首部名词短语(NP)和动词短语(VP)。其中首部名词短语仅由简单的名字“John”组成。动词短语由动词(V)“ate”和名词短语(NP)组成，而该名词短语又由冠词(ART, article)“the”和普通名词(N)“cat”组成。采用列表结构，同样的句子结构可以表示为：

```
(S (NP (NAME John))
   (VP (V ate)
        (NP (ART the)
              (N cat)))))
```

纵览全书，树结构一直扮演着非常重要的角色，因此，我们有必要介绍一些相关的术语。树是图的一种特殊形式，由相互连接的、有标记的节点（譬如，图3.1中标为S, NP等节点）组成。我们之所以称之为树，主要是因为它们看上去像一棵倒置的树，许多术语也是从实际树的类比中演化而来的。其中，树顶上的节点称为树的根节点，最底层的节点称为叶节点。我们说的连接指的是从父节点指向子节点的边。在图3.1中，节点S是节点NP和VP的父节点；同样，节点NP是节点NAME的父节点。每个子节点都有惟一的父节点，而一个父节点可以指向多个子节点。节点N的祖先可以定义为节点N的父节点，或者是其父节点的父节点，依次类推。任何节点都要受其祖先的支配，根节点支配树中所有的其他节点。

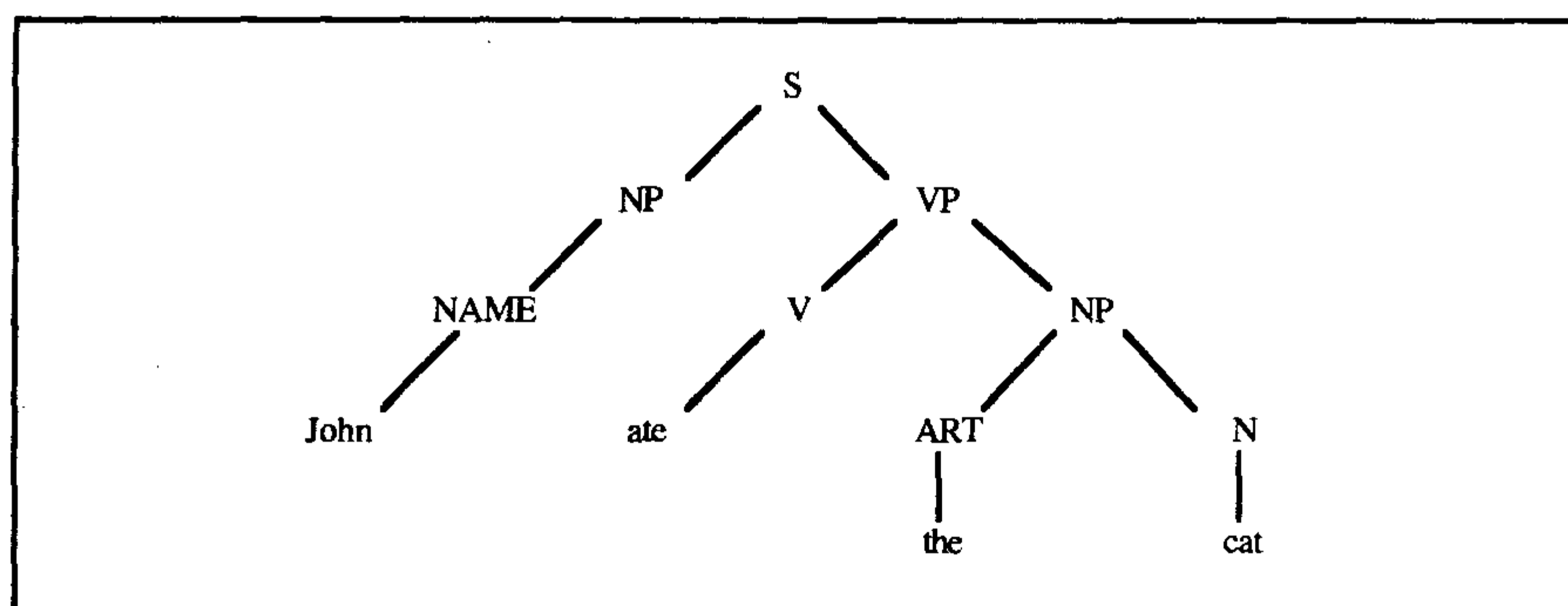


图 3.1 句子“John ate the cat”的树结构表示

当构造句子的树结构时,你必须知道在英语中哪些结构是合法的。产生式规则集合就是要表示哪些树结构是合法的,是允许使用的。这些产生式规则告诉我们,在树结构的生成过程中,某个符号可以展开成一串其他的符号。语法 3.2 给出了一组描述图 3.1 中树结构的产生式规则。其中,产生式规则 1 表示 S 可以由一个 NP 后面跟 VP 构成。产生式规则 2 表示 VP 可以由一个 V 后面跟 NP 构成。而产生式规则 3 和规则 4 分别表示, NP 可以仅由一个 NAME 构成或者由一个 ART 后面跟 N 构成。产生式规则 5 ~ 规则 8 表示的是词类中可能出现的词语。如果语法中所有产生式的左部都只有一个符号,那么该符号称为母亲符号,而这种语法就叫上下文无关文法(CFG, context-free grammar)。上下文无关文法是一种非常重要的语法类,原因有二:上下文无关文法的形式化能力足以描述大部分的自然语言结构;同时,它又足够严格,所以可以在此基础上搭建出分析句子的高效分析器。在语法中,不能进一步分解的符号,我们称之为终结符(terminal symbol),譬如前面例子中的词。除此之外的符号称为非终结符(nonterminal symbol),如 NP, VP 和 S。描述词类的语法标记称为词法符号(lexical symbol),如 N 和 V。显然,许多词语会同时属于多个词类。譬如,词“can”既可以作为 V 使用,也可以作为 N 使用。

- | | |
|---------------------------|----------------------------|
| 1. $S \rightarrow NP VP$ | 5. $NAME \rightarrow John$ |
| 2. $VP \rightarrow V NP$ | 6. $V \rightarrow ate$ |
| 3. $NP \rightarrow NAME$ | 7. $ART \rightarrow the$ |
| 4. $NP \rightarrow ART N$ | 8. $N \rightarrow cat$ |

语法 3.2 一部简单的语法

语法中有一个特殊的符号,我们称之为开始符。本书中,开始符一律用符号 S 表示。如果依据一系列的产生式规则序列,可以从开始符产生出某个句子,那就可以说该语法能派生该句子。例如,语法 3.2 可以产生句子“John ate the cat”(John 吃猫)。这可以从下面的产生序列中看出,下面的派生过程从符号 S 开始,具体过程如下:

S	
$\Rightarrow NP VP$	(改写 S)
$\Rightarrow NAME VP$	(改写 NP)
$\Rightarrow John VP$	(改写 NAME)

⇒John V NP	(改写 VP)
⇒John ate NP	(改写 V)
⇒John ate ART N	(改写 NP)
⇒John ate the N	(改写 ART)
⇒John ate the cat	(改写 N)

在句法分析方面,有两个重要的过程都是基于派生的。第一个过程是句子的生成,即根据产生式规则(rewrite rule)派生并构造出合法的句子。我们可以实现一个简单的句子生成器:从开始符 S 开始,随机地选取一些产生式规则并进行应用,直到最终生成一个词语序列为止。前面的示例表明句子“John ate the cat”可以从语法 3.2 派生而来。第二个基于派生的过程是句法分析,即在给定语法的前提下,甄别出句子的结构。句法分析有两种基本的搜索策略,自顶向下和自底向上。自顶向下策略从符号 S 开始搜索,然后通过不同的方式搜索并改写非终结符,直到生成了输入的句子或者遍历了所有可能的句子为止。前面给出的派生过程采取的就是这种策略,最终结果表明句子“John ate the cat”是合法的。

自底向上策略从句子中的词语开始,逆向使用产生式规则,逐渐减少符号序列,直到只剩下开始符 S 为止。在此过程中,我们用每条产生式左边的符号来改写右边的符号。句子“John ate the cat”自底向上的一种句法分析过程如下:

⇒NAME ate the cat	(改写 John)
⇒NAME V the cat	(改写 ate)
⇒NAME V ART cat	(改写 the)
⇒NAME V ART N	(改写 cat)
⇒NP V ART N	(改写 NAME)
⇒NP V NP	(改写 ART N)
⇒NP VP	(改写 V NP)
⇒S	(改写 NP VP)

图 3.1 显示的树结构表示可以视为在句子结构分析过程中 CFG 产生式规则的使用纪录。换句话说,无论采取自顶向下还是自底向上的分析策略,只要保留句法分析过程的记录,那么,在某种程度上,这项记录都会接近其分析树。

3.2 优秀语法的特征

在创建一种语言的语法时,我们关心的是语法的通用性、选择性和可理解性。通用性指的是语法能正确分析的句子范围,选择性指的是语法能判定出有问题的非句子范围,可理解性指的是语法自身的简易程度。

在规则比较少的小型语法中,比如那些只描述几个句子类型的语法,我们对句子进行结构化分析的时候,某种分析方法看上去可以和另一种分析方法一样来加以理解。为此,很难判断出孰优孰劣,很难说出哪种分析方法比其他的分析方法都好。但是,一旦我们将语法扩展到更大范围的句子以后,常常会发现某种分析更容易扩展,而另一种分析却需要进行复杂的修正。当然,我们更倾向于那种扩展后仍能够保持简易性和通用性的分析。

遗憾的是,这里探讨的主要是小型语法,因而你只有很少的机会去评价语法扩展后的分析。不过,只要铭记任何解决方案应有的一些属性,就可以着手将解决方案推而广之了。尤其要注意的是句子分解为各个子块的方式,句子的子块,一般称为语法成分(constituent,简称成分)。判断语法成分的时候,除了根据直觉之外,你还可以应用一些专门的检测手段,我们在这里会一一讨论。

无论在什么时候,如果你要判断一组词语能否构成某个特定的语法成分,可以试着构造一个新的句子。在该句中,将这组词和另外一组同类成分的词并列使用。这是一个很好的检测方法,因为在绝大多数情况下,只有相同类型的语法成分才可以并列使用。譬如,图 3.3 中的句子就是可接受的,而下面的句子则不对:

- * I ate a hamburger and on the stove.
- (* 我吃了一个汉堡包和在炉子上。)
- * I ate a cold hot dog and well burned.
- (* 我吃了一个凉热狗和烧得很好。)
- * I ate the hot dog slowly and a hamburger.
- * 我吃热狗时很缓慢和一个汉堡包。

总之,在某些句子中,如果假定的语法成分不能和同类型的成分并列使用,那么它很可能就是不正确的。

NP-NP(名词短语 - 名词短语): I ate *a hamburger* and *a hot dog*.

(我吃了一个汉堡包和一个热狗。)

VP-VP(动词短语 - 动词短语): I will *eat the hamburger* and *throw away the hot dog*.

(我要吃汉堡包并且扔掉热狗。)

S-S(句子 - 句子): I ate *a hamburger* and John ate *a hot dog*.

(我吃了汉堡包并且 John 吃了热狗。)

PP-PP(介词短语 - 介词短语): I saw a hot dog *in the bag* and *on the stove*.

(我看到一只热狗放在包里和火炉上。)

ADJP-ADJP(形容词短语 - 形容词短语): I ate a *cold* and *well burned* hot dog.

(我吃了一个凉的但烤得很熟的热狗。)

ADVP-ADVP(副词短语 - 副词短语): I ate the hot dog *slowly* and *very carefully*.

(我吃热狗时很缓慢而且细细品味。)

N-N(名词 - 名词): I ate a *hamburger* and *hot dog*.

(我吃了一个汉堡包和热狗。)

V-V(动词 - 动词): I will *cook* and *burn* a hamburger.

(我要制做并烘烤一个汉堡包。)

AUX-AUX(助词 - 助词): I *can* and *will* eat the hot dog.

(我能够并且将要吃这个汉堡包。)

ADJ-ADJ(形容词 - 形容词):

I ate the very *cold* and *burned* hot dog(that is, very cold and very burned).

(我吃了这个凉的并且烤过了的汉堡包。)

图 3.3 语法成分并列的不同形式

另外一种检测方法是将待测的成分插入到其他的句子中,看看它能否充当相同类型的语法成分。譬如,如果你认为“John's hitting of Mary”(John 打 Mary 这种行为)在句子“John's hitting of Mary alarmed Sue”(John 打 Mary 这种行为警告了 Sue)中是名词短语,那么它在其他句子中同样能当做名词短语。实际上也确实如此,这个名词短语可以充做动词的宾语,如在句子“I cannot explain John's hitting of Mary”(我不能解释 John 打 Mary 这种行为),又比如开始那个句子的被动形式“Sue was alarmed by John's hitting of Mary”(Sue 被 John 打 Mary 这种行为警告了)。基于以上的事实依据,你就可以推断出该语法成分看上去和其他的名词短语一样。

框 3.1 语法的生成能力

基于产生式规则的不同语法形式化体系可以根据其生成能力进行比较,语法生成能力指的是每种形式的语法所能描述的语言范围。本书讨论的是自然语言,实际的情况表明:没有一种自然语言可以足够精确地进行表述,以至于我们很难定义其生成能力。然而,形式化语言却允许精确的、数学化的特征表示。

考虑一种形式化语言,它包含的符号有 a, b, c 和 d(可以把它们当做词来看待)。然后,定义语言 L1 为允许按字母先后顺序排列的任何字母串。譬如,abd, ad, bcd, b 和 abcd 都是合法的句子。为了描述这种语言,我们可以写一种语法,在该语法中,每条规则的右部都由一个终结符,或者由一个终结符再跟一个非终结符组成。这种语法称为正则文法(regular grammar)。对于 L1 来说,语法可以为:

$$\begin{array}{llll} S \rightarrow aS1 & \dot{S} \rightarrow d & S1 \rightarrow d & S3 \rightarrow d \\ S \rightarrow bS2 & S1 \rightarrow bS2 & S2 \rightarrow cS3 & \\ S \rightarrow cS3 & S1 \rightarrow cS3 & S2 \rightarrow d & \end{array}$$

考虑另一种语言 L2,该语言中的所有句子都必须以 a 的连续序列开始,后面接同样数量的 b,如 ab, aabb, aaabbb, 依次类推。在这里,我们就无法写出一个能准确生成 L2 的正则文法。然而,如果采用上下文无关的文法来生成 L2,就变得轻而易举了,其结果如下:

$$S \rightarrow ab \quad S \rightarrow aSb$$

还有一些语言不能用上下文无关文法来生成。譬如某一种语言,它包含的句子要求由一串 a 跟着一串同样数量的 b,再紧接着一串同样数量的 c 组成,如:abc, aabbcc, aaabbbccc, 依次类推。类似地,上下文无关文法也无法产生任意字符串按照相同的次序重复出现两次的句子,如 abab, abcabc, acdabacdab, 依次类推。然而,我们可以采用一些更通用的语法体系,从而达到产生上面这些序列的目的。上下文相关文法是其中的一类重要语法,这些文法中包含的规则形式为:

$$\alpha A \beta \rightarrow \alpha \psi \beta$$

其中 A 是个符号, α 和 β (可能为空) 是符号序列, ψ 是一个非空的符号序列。更加通用的语法是 0 型文法,它可以接受任意形式的产生式规则。

形式语法理论的研究工作始于乔姆斯基(Chomsky, 1956),由正则文法生成的语言是上下文无关文法生成语言的子集,后者又是上下文相关文法生成语言的子集,而最后它们都是 0 型文法生成语言的子集。这些文法形成了语法的层次结构(称为乔姆斯基层次体系, Chomsky Hierarchy)。

在这里,我们再举一个应用这些原理的例子。考虑两个句子“I looked up John’s phone number”(我查 John 的电话号码)和句子“I looked up John’s chimney”(我向上看 John 家的烟囱),这两个句子的语法结构相同吗?如果是,你可以假定这两个句子都可以分析为“主语-动词-补语”结构,其中补语都是介词短语。也就是说,“up John’s phone number”(查 John 的电话号码)是个介词短语。

如果采用并列检测,你就会对刚才的分析产生疑问。现在,将“up John’s phone number”和另一个介词短语并列起来使用,如句子“* I looked up John’s phone number and in his cupboards”(我查 John 的电话号码和他的食橱里面),这当然很怪异。而句子“I looked up John’s chimney and in his cupboards”(我向上看 John 家的烟囱,并看看他家的食橱里面)则是完全可以接受的。因此,将“up John’s phone number”判断为介词短语显然是不正确的。

不能把上例分析为介词短语的进一步依据是,除了在那些包含“look”或“thought”等动词的句子中之外,“up John’s phone number”似乎都不能作为介词短语来使用。即使和动词 look 连用,和句子“Up John’s chimney, I looked”比较而言,与前面意思对等的句子“* Up John’s phone number, I looked”显然令人难以接受。

我们还可以进一步做这类检测。比如,将介词短语按照通常可以允许的方式进行适当的改变。具体而言,可以将名词短语“John’s phone number”(John 的电话号码)替换成代词“it”,这样我们可以得到句子“I looked up it”(我向上看它)。然而,在表达的意思上,改变后的句子和原句“I looked up John’s phone number”却完全不一样。实际上,使用代词并能保持原意的惟一形式就是“I looked it up”(我查它),它可以和句子“I looked John’s phone number up”相对应。

因此,对上述两个句子需要进行不同的分析。如果“up John’s phone number”不是 PP,那么就剩下两种可能的分析结果。第一种是动词短语,它由复合动词“looked up”后面跟一个名词短语组成;第二种分析对象包含三个组成部分,动词“looked”、语助词“up”和名词短语。这两种都比原来的分析更加合理。最后,在这两个分析结果中,应采取什么样的检测方法进行选择呢?

在研究语法的时候,你会发现每个语法成分都会有非常多的使用方式。因此,为了判断一种新的分析合理与否,你进行检测的次数也会随之增加。有些时候,为了保证正确分析新的形式,你不得不倒退回去,修正已有的语法。在现有的语言知识背景下,这种回溯步骤是不可避免的。必须记住的关键点是,为语法增加一条新规则时,必须详尽地考察它和已有规则的相互关系。

3.3 自顶向下的句法分析器

句法分析算法可以描述成这样一个过程,对各种语法规则进行组合,从而得到不同的解析路径。然后,在这些不同的组合路径中,搜索出一种能生成输入句结构语法树的组合方式。为了表述简便,我们不会显式地构造出语法树。对于能否构造出一棵语法树这个问题,算法会返回肯定或否定的答案。换言之,句法分析算法判断的是某个句子能否为语法所接受。在这一节里,我们会比较详细地介绍简单的自顶向下句法分析算法。随后,我们会介绍人工智能(AI, artificial intelligence)中与搜索相关的一些研究工作。

自顶向下的句法分析器都是从符号 S 开始分析的,它会试着将 S 改写成与输入句子词类相匹配的终结符序列。任何一个时刻的分析状态都可以表示为一个符号列表,该列表是迄今为止所有操作的最终结果,我们称之为符号表(symbol list)。举例来说,句法分析器从状态(S)开始,应用了规则 $S \rightarrow NP VP$ 后,符号表为(NP VP)。如果再应用规则 $NP \rightarrow ART N$,那么符号表就是(ART N VP),依次类推。

句法分析器会按照这种方式继续分析下去,直至它的状态全部由终结符组成为止。随后,它会检查当前状态与输入句子是否完全匹配。然而,这种方法非常费时,因为前期犯的错误(比如,选择规则分解开始符 S 时)直到相当晚的时候才会被发现。更好的算法会尽可能早地对输入句进行检查。此外,与其用单个规则描述每个词语可能归属的词类,还不如直接采用词典的结构有效地存储每个词语所有可能的词类。目前,这种词典非常简单,在本例中用到的是 一部非常小的词典,其内容有:

cried: V
dogs: N, V
the: ART

有了一部特定的词典之后,如语法 3.4 所示的语法就不再需要包含任何词法规则了。

1. $S \rightarrow NP VP$	4. $VP \rightarrow V$
2. $NP \rightarrow ART N$	5. $VP \rightarrow V NP$
3. $NP \rightarrow ART ADJ N$	

语法 3.4 语法示例

完成这些改动之后,可以将句法分析过程中的状态定义为:与前面示例类似的符号表以及一个表示句子当前位置的数。位置信息标在词语中间,其中 1 表示第一个词前面的位置。举个例子,下面就是一个标识了位置的句子:

₁The ₂ dog ₃ cried ₄

一个典型的分析状态为:

((N VP) 2)

它表示句法分析器需要从位置 2 开始,寻找一个后面接动词短语 VP 的名词 N。新的状态都是由旧状态转换而来的,转换的依据是第一个符号是否是词汇符号。如果它是词汇符号,比如前面例子中的 N,同时句子中的下一个词语也属于这个词类,则可以将当前状态进行如下更新:删除第一个符号,同时更新位置计数器。在本例中,因为词“dogs”在词典中标记为名词 N,则句法分析器的下一个状态为:

((VP) 3)

这意味着分析器需要从位置 3 开始寻找一个动词短语 VP。如果第一个符号是个非终结符,比

如 VP,那么,我们就要采用语法中的一条规则替换它。例如,使用语法 3.4 中的规则 4,新的状态为:

((V) 3)

这意味着分析器需要从位置 3 开始寻找一个动词。另一方面,如果这时候我们使用的是产生式 5,那么新状态为:

((V NP) 3)

假设句子的句法分析树确实存在,为了确保能找到该分析树,句法分析算法必须要系统地遍历每一个可能的状态。其中一个很简单的技术就是回溯。采取这种方法,就不是仅仅生成一个新状态((VP) 3),而是生成所有可能的状态。选择这些新状态中的一个作为下一个状态,剩余的新状态作为后备状态。假如当前状态不能得到一个正确的解答,就可以简单地从后备状态列表中选择一个新的状态作为当前状态。下面,我们会更详细地讨论该算法。

3.3.1 简单的自顶向下句法分析算法

该算法处理的是一个可能状态的列表,我们称之为可能状态列表。该表的第一个元素为当前状态。当前状态包含一个符号表和句子中的当前位置,搜索状态中剩余的元素为后备状态。每个后备状态都表示的是<符号表,词语位置>候选二元组。例如,下面就是一个可能的状态列表:

((N) 2)((NAME) 1)((ADJ N) 1))

它表示的当前状态为位置 2 上的符号表(N),同时还有两个可能的后备状态。其中一个后备状态是位置 1 上的符号表(NAME),另一个是位置 1 上的符号表(ADJ N)。

算法从初始状态((S)1)开始,没有后备状态。算法如下:

- 选择当前状态:从可能状态列表中取出第一个状态,称其为 C。如果列表为空,则该算法失败返回(即,不可能存在成功的句法分析结果)。
- 如果 C 是一个空符号链表,而且当前词语的位置恰好处于句子末尾,那么算法成功返回。
- 否则,生成下一个可能的状态。
 - ◇ 如果 C 中符号表的第一个符号是词法符号,并且句子中的下一个词语可以归入该词类,则生成新的状态。生成过程为从符号表中删除第一个符号,更新当前词语的位置。将新状态加入可能状态列表中。
 - ◇ 否则,如果 C 符号表的第一个符号是非终结符,依据语法中所有能改写该非终结符的规则,生成一个新状态,并将这些状态加入可能状态列表。

我们以句子“The dogs cried”为例,采用语法 3.4,图 3.5 给出了该算法对例句的整个分析过程。首先,用产生式 1 替换开始符 S,生成步骤 2 中新的当前状态((NP VP)1)。紧接着,我们要对 NP 进行替换操作。由于语法中有两个关于 NP 的产生式,所以,同时产生两个可能状态:一个是在位置 1 处新的当前状态(ART N VP),而另一个是在位置 1 处的后备状态((ART ADJ N VP) 1)。第四步的时候,在句子的位置 1 处发现一个冠词,当前状态变成(N VP)。第三步生成

的后备状态保持不变。按照这种方法,句法分析进行到步骤 5,这时有两条不同的规则可以替换 VP。第一条规则生成新的当前状态,第二条规则生成的状态压到后备状态栈中。在步骤 7,当前状态为空,而且输入句子的所有词语都已遍历,因此句法分析成功。

步骤	当前状态	后备状态	备注
1.	((S) 1)		开始位置
2.	((NP VP) 1)		采用规则 1 改写 S
3.	((ART N VP) 1)		采用规则 2 和规则 3 改写 NP
		((ART ADJ N VP) 1)	
4.	((N VP) 2)		用“the”匹配 ART
		((ART ADJ N VP) 1)	
5.	((VP) 3)		用“dogs”匹配 N
		((ART ADJ N VP) 1)	
6.	((V) 3)		采用规则 4 和规则 5 改写 NP VP
		((V NP) 3)	
		((ART ADJ N VP) 1)	
7.			将 V 与“cried”匹配之后,只剩下空的语法符号和空的句子,最终分析成功

图 3.5 “₁ The ₂ dogs ₃ cried ₄”自顶向下深度优先的句法分析过程

让我们采用相同的算法和语法对下面的句子进行分析处理:

₁The ₂ old ₃ man ₄ cried ₅

本例中,我们假定词语“old”的词性既可能是形容词也可能是名词。“man”是名词或者是动词(如句子“The sailors man the boats”[水手划船]中的“man”)。具体来说,词典为:

the: ART
old: ADJ, N
man: N, V
cried: V

整个分析过程如下所示(见图 3.6)。开始符 S 由规则 1 改写,产生新的当前状态((NP VP) 1)。随后,继续改写 NP,产生一个新的状态((ART N VP) 1)和后备状态((ART ADJ N VP) 1)。继续分析,我们会发现“the”可以作为定冠词使用。因此,可以生成状态((N VP) 2)。然后,“old”可以作为名词,从而得到状态((VP) 3)。这时,我们有两种方法改写 VP,分别得到当前状态((V) 3)和后备状态((V NP)3),以及以前产生的后备状态((ART ADJ N)1)。若把词语“man”看成动词,则状态变成(()4)。遗憾的是,尽管此时符号表为空,但当前词语的位置并不是在句末。因此,无法生成新的状态,我们必须使用一个后备状态。在下一次循环中,在步骤 8 的时候,尝试选用后备状态((V NP)3)。同样,将词语“man”作为动词对待,可以生成新的状态((NP) 4)。任何关于 NP 的规则都不能成功地分析出最终结果。最后,在步骤 12 的时候,我们尝试最后一个后备状态((ART ADJ N VP) 1),它能够产生成功的句法分析结果。

步骤	当前状态	后备状态	备注
1.	((S) 1)		
2.	((NP VP) 1)		S 改写成 NP VP
3.	((ART N VP) 1)		NP 改写后产生两个新的状态
		((ART ADJ N VP) 1)	
4.	((N VP) 2)		
		((ART ADJ N VP) 1)	
5.	((VP) 3)		保留该后备状态
		((ART ADJ N VP) 1)	
6.	((V) 3)		
		((V NP) 3)	
		((ART ADJ N VP) 1)	
7.	(() 4)		
		((V NP) 3)	
		((ART ADJ N VP) 1)	
8.	((V NP) 3)		选用第一个后备状态
		((ART ADJ N VP) 1)	
9.	((NP) 4)		
		((ART ADJ N VP) 1)	
10.	((ART N) 4)		用规则 4 选择 ART 失败
		((ART ADJ N) 4)	
		((ART ADJ N VP) 1)	
11.	((ART ADJ N) 4)		再次失败
		((ART ADJ N VP) 1)	
12.	((ART ADJ N VP) 1)		现在,探究在步骤 3 得到的后备状态
13.	((ADJ N VP) 2)		
14.	((N VP) 3)		
15.	((VP) 4)		
16.	((V) 4)		
		((V NP) 4)	
17.	(() 5)		成功

图 3.6 句子“₁The₂ old₃ man₄ cried₅”的自顶向下句法分析过程

3.3.2 句法分析与搜索过程

可以将句法分析看成是人工智能中搜索问题的一个特例。具体而言,本节介绍的自顶向下句法分析器可以描述为下面这样一个通用的搜索过程。其中,可能状态列表初始化为句法分析器的开始状态。随后,就可以按照如下步骤反复运行,直到返回成功或失败为止:

1. 从可能状态列表中选择第一个状态(并从列表中将它删除)。
2. 从选中的状态出发,尝试所有的可能选择,并生成新的状态。(如果搜索路径选择不当,可能生成不了任何新的状态。)
3. 将步骤 2 生成的状态加到可能状态列表中。

对于深度优先策略来说,可能状态列表是个栈结构。也就是说,步骤 1 总是取列表的第一个元素,步骤 3 总是将新状态放到列表的最前面,形成后进先出(LIFO, last-in first-out)策略。

与此形成对比的是,广度优先策略中的可能状态列表是作为队列结构操作的。步骤3将新状态加到列表的末尾,而不是列表的开头,形成先进先出(FIFO, first-in first-out)策略。

我们可以采用树的形式来比较这两种不同的搜索策略。如图3.7所示,该图给出了最后一个例子上分析状态的整个空间。其中,树中的每个节点代表一个句法分析状态,其子节点表示的是从该节点状态出发能到达的可能状态。节点边上的数字记录了该状态在算法中的处理顺序。左边是深度优先策略的处理顺序,右边是广度优先策略的处理顺序。记住,我们分析的句子为:

₁ The ₂ old ₃ man ₄ cried ₅

在这个简单的例子中,广度优先搜索和深度优先搜索的主要区别体现在对第一个 NP 的解析处理上。检测到的这个 NP 存在两种可能的句法解释结果,不同的搜索策略对这两种解释的遍历顺序不同。如果是深度优先,则只考虑采纳其中的一个解释,并对结果进行扩展。除非最终失败,否则它是不会考虑第二种解释的。如果是广度优先策略,则会交替地考虑、选择并处理每种解释,每次选择一种,只对它扩展一步。在这个例子中,深度优先和广度优先策略都成功地找到了分析结果,但他们对状态空间的搜索顺序不同。在有些情况下,深度优先搜索常常会很快地移向成功节点;但在另外一些情况下,有时也会在无效的路径上浪费相当可观的时间。然而,广度优先策略会遍历每个可能的解析方案直至一定的深度,然后再向前推进。在本例中,深度优先策略比广度优先策略早一步分析成功。在图3.7中,深度优先策略不会访问右边标为15的状态。

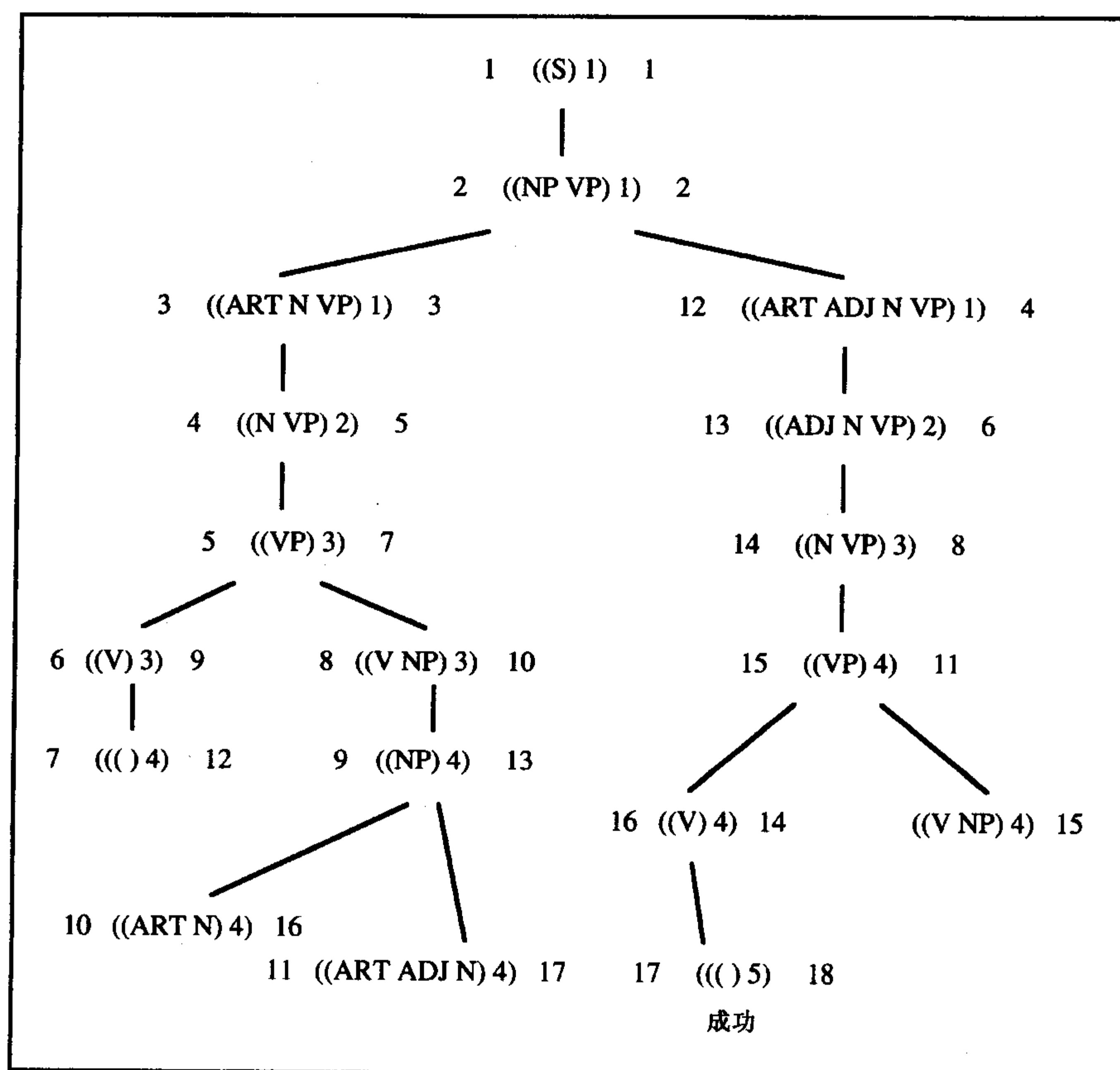


图3.7 两种不同分析策略的搜索树(左为深度优先策略,右为广度优先策略)

在某些情况下,这些简单的搜索策略可能会陷入无限的循环当中,无法继续分析其他可能的情况。譬如,我们来考虑如下的一个左递归规则,人们常常会首先考虑采用这个规则描述英语的所有格(例如名词短语“the man's coat”):

$$\text{NP} \rightarrow \text{NP}'\text{s N}$$

采用简单的深度优先策略,以非终结符 NP 开始的状态会被改写为“NP's N”开始的新状态。但这个新状态仍然是以 NP 开头的,算法会采取同样的方式继续改写。如果句法分析器中没有明确的监测机制,那么它将永远这样改写 NP,直至无穷!对于左递归规则,广度优先策略会处理得更好一些,因为到达新 NP 的状态之前,它会尝试所有其他的方法来改写 NP。但是,一旦碰上不合乎文法的句子,这个过程就无法停止,因为它会永不停歇地改写 NP 以寻找正确的分析结果。基于这种原因,许多系统都禁止语法中采用左递归规则。

现在的大多数句法分析器往往采用深度优先策略,因为深度优先策略所需的后备状态往往最少,因此可以较少地占用内存,并且需要的记录数也会更少。

3.4 自底向上的 chart 句法分析器

自顶向下与自底向上句法分析器的主要区别在于语法规则的使用方式上。譬如下面的规则:

$$\text{NP} \rightarrow \text{ART ADJ N}$$

在自顶向下的分析系统中使用这条规则的时候,找到 NP 需要通过寻找序列“ART ADJ N”来实现。而在自底向上的分析器中,如果碰到序列“ART ADJ N”,则将其识别为一个 NP。因此,在自底向上的句法分析过程中,一个基本操作就是将一个符号序列匹配归约为其产生式的右部。这个匹配过程可以形式化为一个搜索过程,从而可以简单地构建出自底向上的分析器。其分析状态仅包含一个符号列表,其初始值为句子中的词语。我们根据该词所有可能的词类,采用各种方式改写词语,主要是通过如下方式来产生新的后继状态:

- 将词语替换成其所属词类。
- 把与语法规则右部相匹配的一串符号替换成该规则左部的符号。

遗憾的是,即使是简单地实现该算法,其代价也非常高。因为,这种句法分析器常常会一遍又一遍地重复同样的匹配,我们没有必要重复大量的类似操作。为了避免这个问题,我们引进了一个称为 chart 的数据结构来存储已经部分匹配的结果。这样,已经完成的匹配就不必再重复去做。

我们考虑匹配的时候,总是从一个语法成分出发的,在这里,我们称之为键(key)。为了找到一些能够匹配与键相关的符号串的规则,我们要寻找那些以该键开始的规则,或者是以前面键开头的规则,并且需要当前键来做进一步扩展,或者需要当前键完成匹配。以语法 3.8 为例,让我们来做进一步的讨论。

假设你正在分析一个以 ART 开始的句子。规则 2 和规则 3 均以 ART 开头,这时我们的键是 ART,所以匹配成功。为了便于对下一个键进行分析,需要将当前的状态记录下来,记录规则是:规则 2 和规则 3 应该从 ART 后面的那个位置继续分析匹配。你可以在规则中加一个圆

圈标记(\circ)来表示这些事实。这个标记表示的是迄今为止已经匹配过的位置。于是,上面这个事实可以表示为:

- 2'. $NP \rightarrow ART \circ ADJ\ N$
- 3'. $NP \rightarrow ART \circ N$

如果下一个输入的键是一个 ADJ(形容词),那么或许要启用规则 4,同时进一步扩展规则 2',得到:

- 2''. $NP \rightarrow ART\ ADJ \circ N$

1. $S \rightarrow NP\ VP$

2. $NP \rightarrow ART\ ADJ\ N$

3. $NP \rightarrow ART\ N$

4. $NP \rightarrow ADJ\ N$

5. $VP \rightarrow AUX\ VP$

6. $VP \rightarrow V\ NP$

语法 3.8 简单的上下文无关文法

chart 中保存了迄今为止所有从句子中派生出来的成分的纪录,同时还保存了已经部分匹配但还没有完全成功匹配的规则纪录。我们称这些纪录为活动边(active arc)。譬如在前例中,当碰到第一个符号是 ART,紧随其后的是 ADJ 时,你就可以得到一幅 chart 图,如图 3.9 所示。可以将其解释为有两个匹配完的成分,分别是位置 1 和位置 2 之间的 ART1 以及位置 2 和位置 3 之间的 ADJ1。其中有四条活动边,表示可能会遇到的成分。这些活动边由箭头表示,依次可以解释为(按照从上到下的顺序):可能会产生一个从位置 1 开始的 NP,还需要找到一个从位置 2 开始的 ADJ;可能产生另一个从位置 1 开始的 NP,还需要找到一个从位置 2 开始的 N;可能产生一个 NP,它从位置 2 开始有一个 ADJ,还需要找到一个从位置 3 开始的 N;最后,可能产生一个 NP,它从位置 1 开始有一个 ART,并紧跟着一个 ADJ,还需要找到一个从位置 3 开始的 N。

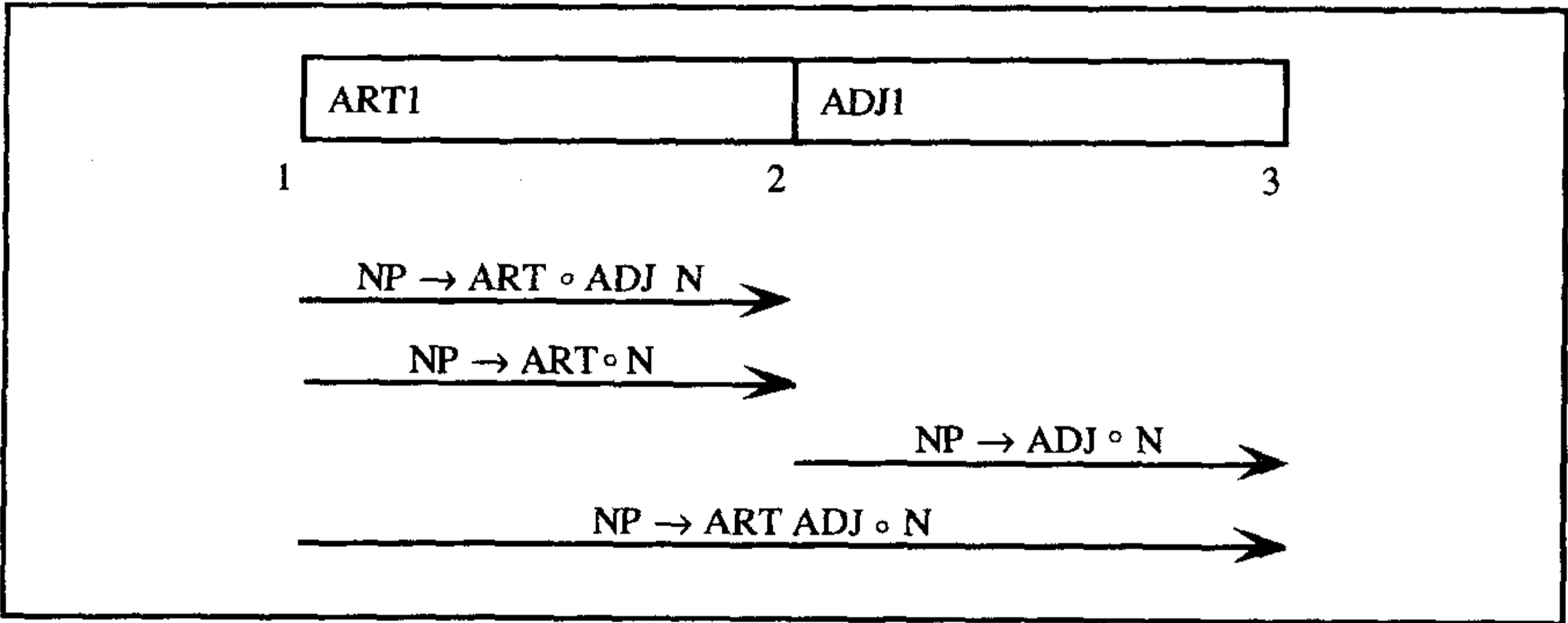


图 3.9 在位置 2 遇到 ADJ 后的 chart 图

基于 chart 的句法分析器,其基本操作需要将活动边和已匹配的成分结合起来。处理的结果可能是一个新匹配的成分,也可能是原来活动边的一条扩展边。新匹配的成分由一个称为

待处理表(*agenda*)的结构来维护,直到这些成分都加入到图表中为止。图 3.10 给出的边扩展算法对这个过程做了更精确的阐述。在该算法的基础上,图 3.11 给出了自底向上的 *chart* 句法分析算法。

添加一个从位置 p_1 到位置 p_2 的成分 C :

1. 将 C 加入到 *chart* 的位置 p_1 和位置 p_2 之间
2. 对任意一条形式为 $X \rightarrow X_1 \dots \circ C \dots X_n$ 的活动边,如果该活动边在位置 p_0 和 p_1 之间,则在位置 p_0 和 p_2 之间添加一条活动边 $X \rightarrow X_1 \dots C \circ \dots X_n$
3. 对任意一条形式为 $X \rightarrow X_1 \dots X_n \circ C$ 的活动边,如果该活动边在位置 p_0 和 p_1 之间,则在待处理表中添加一个新的成分 X ,该成分在位置 p_0 和 p_2 之间

图 3.10 边扩展算法

执行下列过程,直到输入为空:

1. 如果待处理表为空,在词典中查找下一个输入词语的解释,并将它们都加入到待处理表中
2. 从待处理表中选择一个成分。(我们假定该成分为 C ,其跨度为从位置 p_1 到位置 p_2)
3. 对语法中每条形式为 $X \rightarrow C X_1 \dots X_n$ 的规则,增加一条活动边 $X \rightarrow \circ C X_1 \dots X_n$,其跨度为从位置 p_1 到位置 p_2
4. 采取上述的边扩展算法将 C 加入到 *chart* 中

图 3.11 自底向上的 *chart* 句法分析算法

在自顶向下的句法分析器中,可以使用深度优先搜索策略,也可以使用广度优先搜索策略,这取决于待处理表究竟是由栈结构还是由队列结构来实现的。对于完全的广度优先策略来说,在算法开始前,我们就要读入整个输入,并将词语的所有解释都加入到待处理表中。在下例中,我们假定使用的是深度优先策略。

在语法 3.8 上,我们采用本算法对句子“The large can can hold the water”(大罐头盒能盛水)进行分析,下面是我们使用到的词典:

the: ART
 large: ADJ
 can: N, AUX, V
 hold: N, V
 water: N, V

为了更好地理解这个例子,我们画出算法每一步扩展后的 *chart* 图。待处理表初始为空,所以读入词语“the”后,成分 ART1 放入待处理表中。

输入 ART1:(位置 1 和位置 2 之间的 the)

在位置 1 和位置 2 之间增加活动边 $NP \rightarrow ART \circ ADJ N$

在位置 1 和位置 2 之间增加活动边 $NP \rightarrow ART \circ N$

这两条活动边都是在句法分析算法的步骤 3 中增加进来的,推导的依据分别是语法中的规则 2 和规则 3。紧接着,该算法读入下一个词语“large”,生成成分 ADJ1。

输入 ADJ1:(位置 2 和位置 3 之间的 large)

在位置 2 和位置 3 之间增加活动边 $NP \rightarrow ADJ \circ N$

在位置 1 和位置 3 之间增加活动边 $NP \rightarrow ART \ ADJ \circ N$

在这里,第一条边是在算法的步骤 3 中增加的。第二条增加到 chart 中的活动边是第一条活动边在输入 ART1 时,根据边扩展算法增加的(见步骤 4)。

图 3.9 给出了此时的 chart 状态图。需要注意的是,活动边绝不会从 chart 图中被删除掉。譬如,当位置 1 和位置 2 之间的边 $NP \rightarrow ART \circ ADJ \ N$ 扩展后,生成位置 1 和位置 3 之间的边,这两条边都保留在 chart 中。这是非常必要的,因为还可能存在不同的解析方式,很可能会重新使用这些边。

下一个词语“can”有三种解释,相应地可以创建三种成分 N1, AUX1 和 V1。

输入 N1:(位于位置 3 和位置 4 之间的“can”)

在算法的步骤 2,并没有增加新的活动边。但是在第四步的时候,根据边扩展算法,两个活动边匹配完毕,生成两个 NP,将它们加入到待处理表中。其中,在位置 1 和位置 4 之间的第一个 NP,依据语法中的规则 2 构造而成;在位置 2 和位置 4 之间的第二个 NP,依据规则 4 构造而成。这些 NP 现在都位于待处理表的顶部。

输入 NP1:NP(在 1 和 4 之间的“the large can”)

在位置 1 和位置 4 之间增加活动边 $S \rightarrow NP \circ VP$

输入 NP2:NP(在 2 和 4 之间的“large can”)

在位置 2 和位置 4 之间增加活动边 $S \rightarrow NP \circ VP$

输入 AUX1:(在 3 和 4 之间的“can”)

在位置 3 和位置 4 之间增加活动边 $VP \rightarrow AUX \circ VP$

输入 V1:(在 3 和 4 之间的“can”)

在位置 3 和位置 4 之间增加活动边 $VP \rightarrow V \circ NP$

图 3.12 给出了此时的 chart 图,该图给出了所有已完成匹配的成分(NP2, NP1, ART1, ADJ1, N1, AUX1, V1),同时也给出了还未完全匹配的活动边。下一个词语又是“can”,则生成成分 N2, AUX2 和 V2。

输入 N2:(位于 4 和 5 之间的“can”,第二个“can”)

无活动边可增加

输入 AUX2:(位于 4 和 5 之间的“can”)

在位置 4 和位置 5 之间增加活动边 $VP \rightarrow AUX \circ VP$

输入 V2:(位于 4 和 5 之间的“can”)

在位置 4 和位置 5 之间增加活动边 $VP \rightarrow V \circ NP$

下一个词语是“hold”,产生成分 N3 和 V3。

输入 N3:(位于 5 和 6 之间的“hold”)

无活动边可增加

输入 V3:(位于 5 和 6 之间的“hold”)

在位置 5 和位置 6 之间增加活动边 $VP \rightarrow V \circ NP$

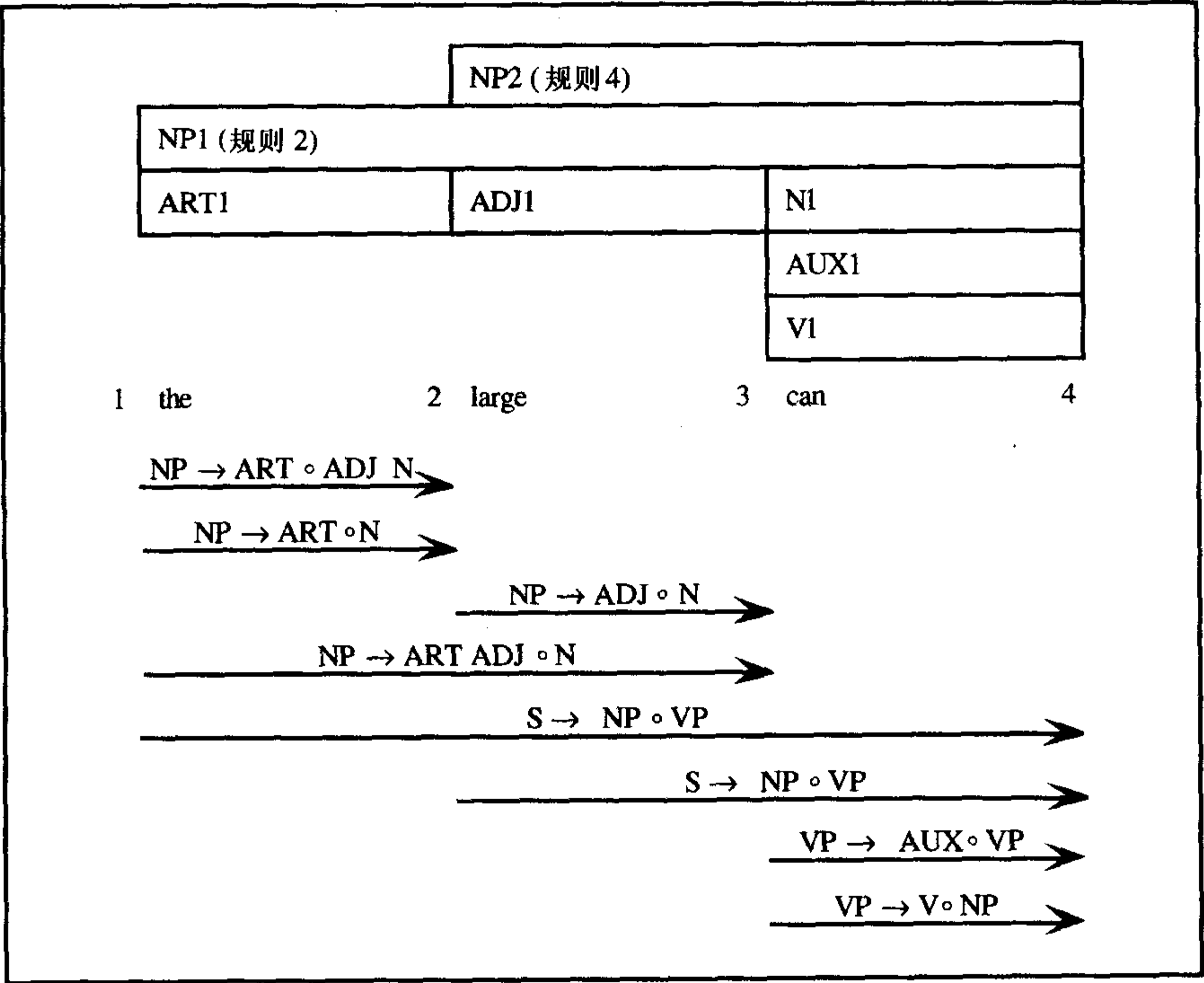


图 3.12 “the large can”分析之后的 chart 图

除了图 3.12 中已给出的第一个 NP 外,图 3.13 所示的 chart 给出了所有已完成匹配的成分和所有的活动边。

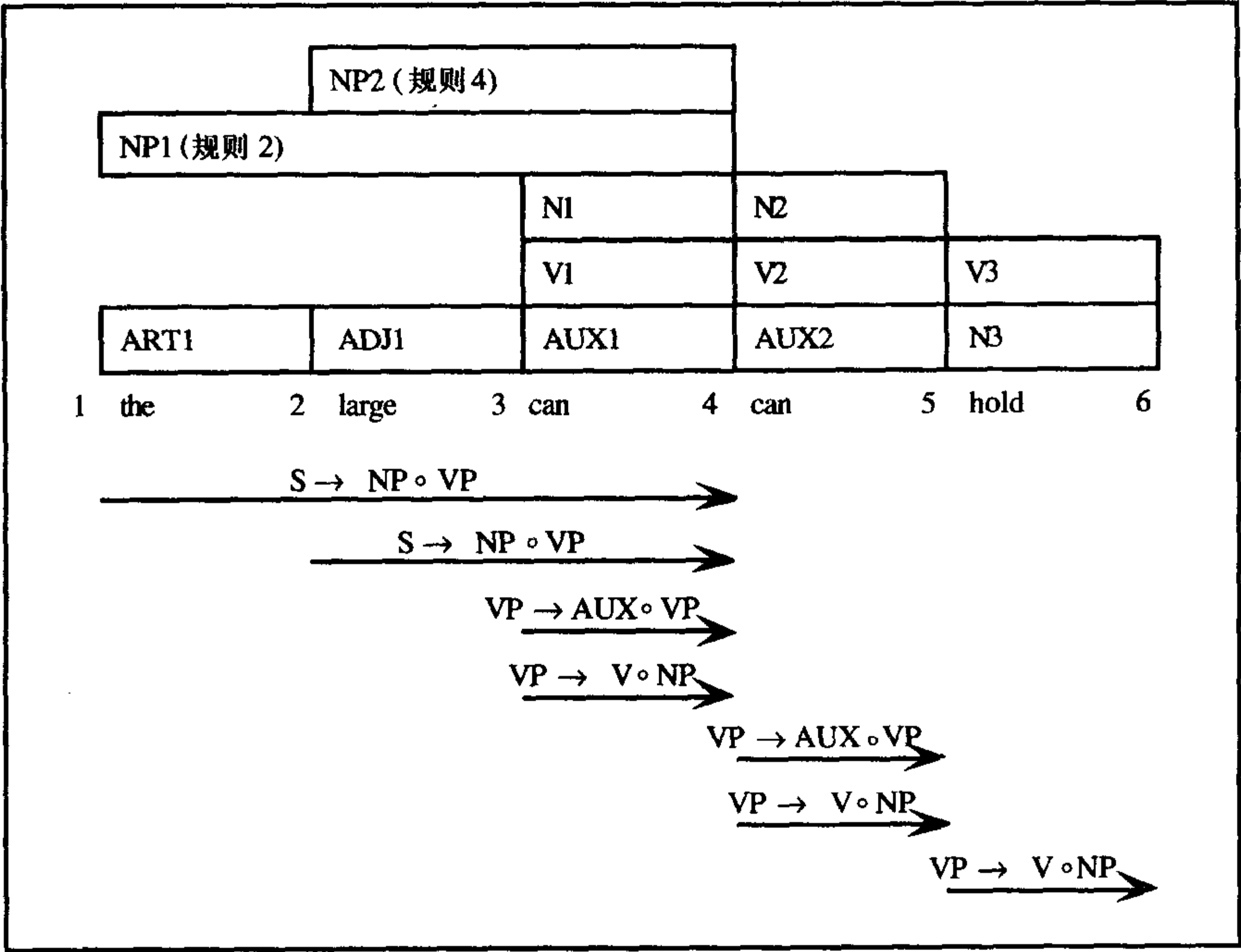


图 3.13 加入“hold”之后的 chart 图(省略了生成第一个 NP 的活动边)

输入 ART2:(位于 6 和 7 之间的“the”)
在位置 6 和位置 7 之间增加活动边 $NP \rightarrow ART \circ ADJ\ N$
在位置 6 和位置 7 之间增加活动边 $NP \rightarrow ART \circ N$

输入 N4:(位于 7 和 8 之间的“water”)
步骤 3 中,没有活动边可以增加
位置 6 和位置 7 之间的活动边 $NP \rightarrow ART \circ N$ 匹配完成后,将位置 6 和位置 8 之间的名词短语 NP3 加入到待处理表中。

输入 NP3:(位于 6 和 8 之间的“the water”)
位置 5 和位置 6 间的活动边 $VP \rightarrow V \circ NP$ 匹配完成后,将位置 5 和位置 8 之间的动词短语 VP1 加入到待处理表中。
在位置 6 和位置 8 之间,增加活动边 $S \rightarrow NP \circ VP$

这时,chart 图如图 3.14 所示,该图中只给出了一些在后续分析过程中需要用到的活动边。

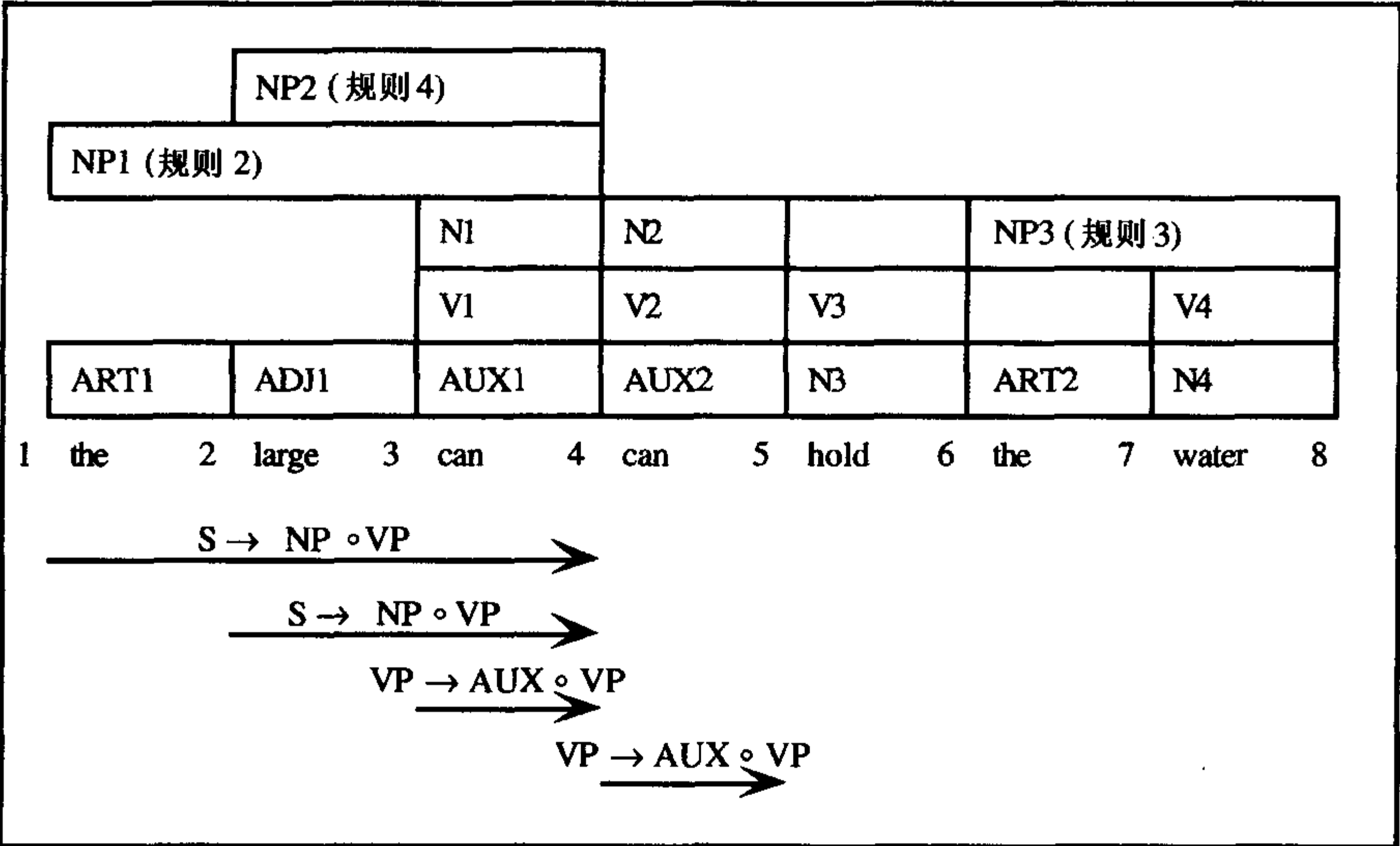


图 3.14 找到所有 NP 后的 chart 图,这里忽略了重要活动边之外的所有边

输入 VP1:(位于 5 和 8 之间的“hold the water”)
位于位置 4 和位置 5 之间的活动边 $VP \rightarrow AUX \circ VP$ 匹配完成后,将位置 4 和位置 8 之间的动词短语 VP2 加入到待处理表中。
输入 VP2:(位于 4 和 8 之间的“can hold the water”)
位于位置 1 和位置 4 之间的活动边 $S \rightarrow NP \circ VP$ 匹配完成后,将位置 1 和位置 8 之间的开始符 S1 加入到待处理表中。
位于位置 3 和位置 4 之间的活动边 $VP \rightarrow AUX \circ VP$ 匹配完成后,将位置 3 和位置 8 之间的动词短语 VP3 加入到待处理表中。
位于位置 2 和位置 4 之间的活动边 $S \rightarrow NP \circ VP$ 匹配完成后,将位置 2 和位置 8 之间的开始符 S2 加入到待处理表中。

至此,已经推导出了一个表示全句的开始符 S,句法分析过程就可以成功终止了。如果想找出句子所有的句法分析结果,则需要继续分析,直到待处理表为空。由于存在不同的结构分析结果,最后 chart 图都会包含所有不同的 S 结构,每个 S 结构都覆盖了所有的位置。而且,和句法分析结果列表比起来,结构完全集表示形式更加高效。因为不同的 S 结构可以共享公共子结构,而这些子结构在 chart 中只出现一次。图 3.15 给出了最终的 chart 图。

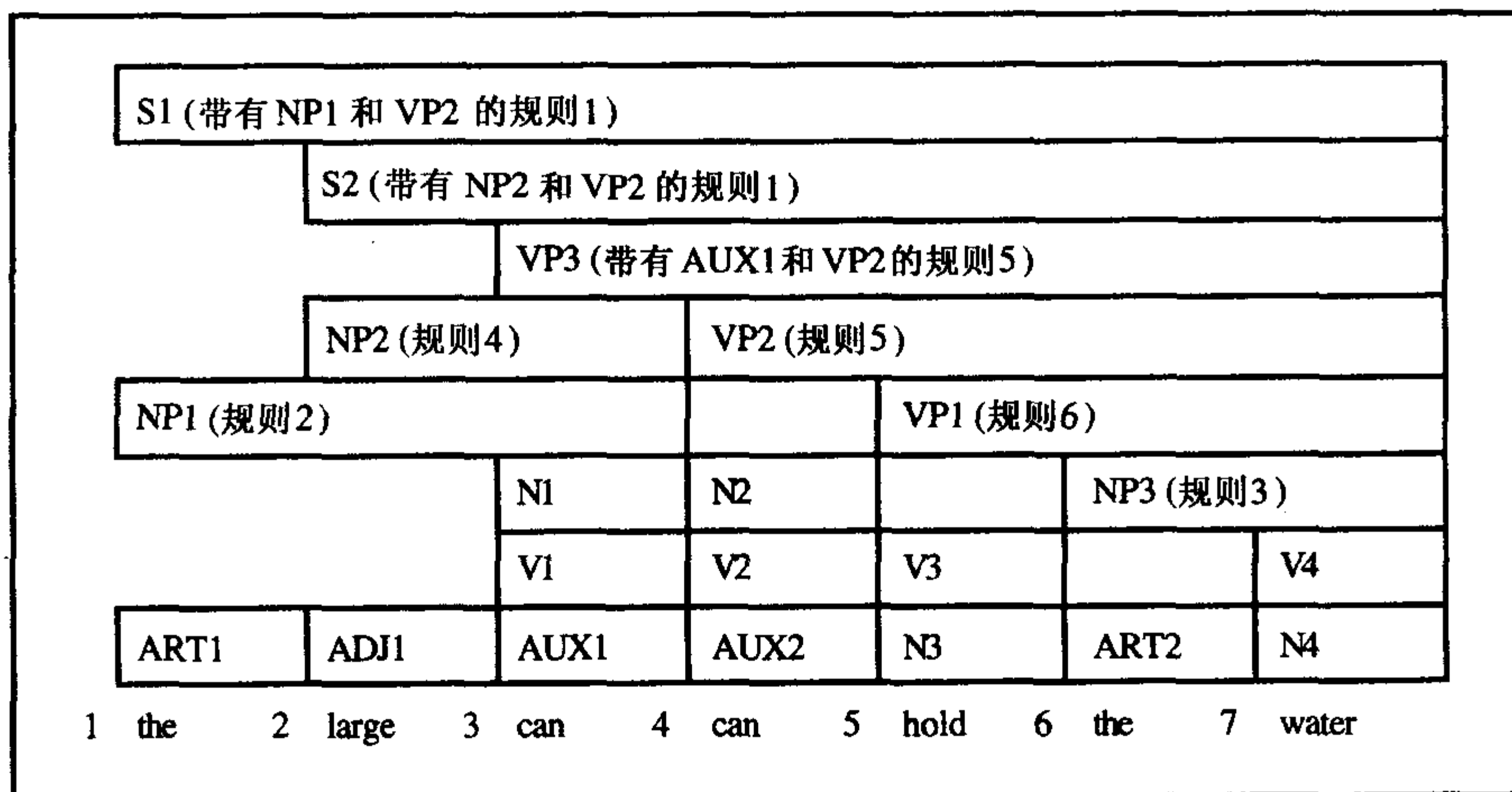


图 3.15 最终的 chart 图

3.4.1 效率方面的考虑

基于 chart 的句法分析器会比仅仅依赖搜索过程的句法分析器高效得多,这是因为同一个成分绝不会重复构造多次。譬如,对于一个长度为 n 的句子来说,纯粹的自顶向下或者自底向上的搜索策略需要进行 C^n 次操作。其中, C 是一个仅和具体算法相关的常数。即使 C 非常小,指数级的增长速度也会让该算法无法使用。另一方面,在最坏的情况下,基于 chart 的句法分析器会在所有的位置间构造出所有可能的成分。在这种条件下,时间复杂度为 $K * n^3$ 。其中, n 为句子长度, K 是一个仅和具体算法相关的常数。当然, chart 句法分析器在每一步都有很多操作,因此, K 会比 C 大一些。在此,我们对这两种方法进行对比。假定 C 为 10, K 为 C 的 100 倍,即 K 等于 1000。对于一个长度为 12 的句子,纯粹的搜索大约要进行 10^{12} (即 1 000 000 000 000) 次操作。与此形成对比的是, chart 句法分析器仅需要 $1000 * 12^3$ (即 1 728 000) 次操作。在这种假定条件之下,对于某些句子, chart 句法分析器的效率比全搜索要快 500 000 倍!

3.5 转移网络语法

到现在为止,我们仅仅探讨了一种语法表示的形式化手段,即上下文无关的产生式规则。在本节中,让我们来考虑另一种形式化手段。该方法非常有用,并且应用领域广泛。这种形式化方法的理论基础主要是转移网络,而转移网络是由节点和标记边组成的。其中一个节点被确定为初始状态,或者称为启动状态。让我们看看语法 3.16 中定义的 NP 转移网络,其初始状态的标记为 NP,每条边上的标记为词类。从初始状态出发,如果句子中当前的词属于该边标

记的词类,那么就可以穿过该边。通过这条边后,下一个词语更新为当前词。如果一个短语中所有的词都能够从节点 NP 依次沿着路径到达弹出边(标记为 pop 的边),则该短语就是一个合法的 NP。该转移网络与下面的上下文无关文法等价,它们所能识别出的句子集合相同:

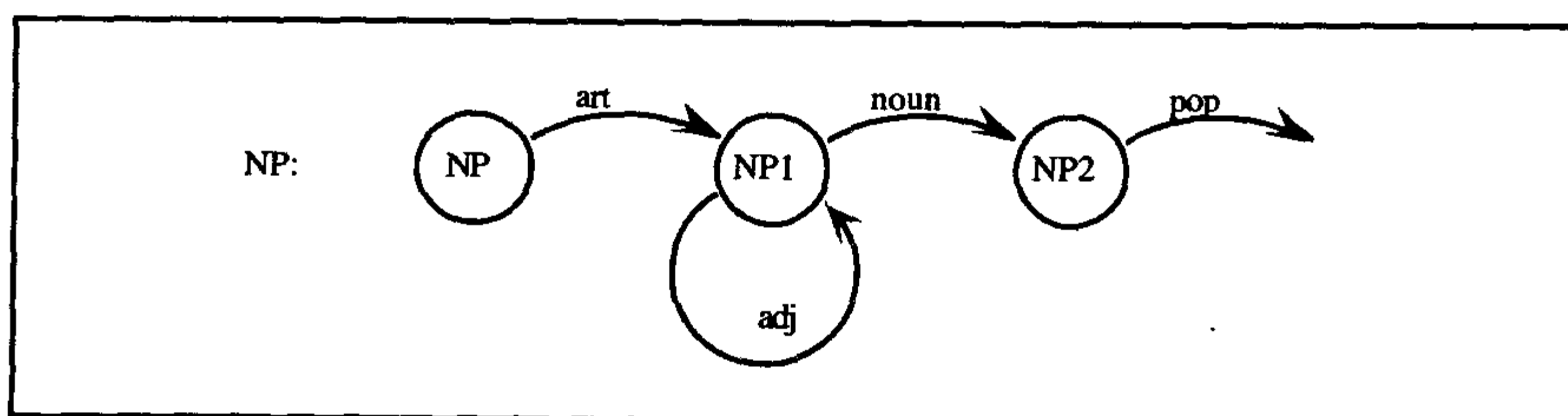
$NP \rightarrow ART\ NP1$

$NP1 \rightarrow ADJ\ NP1$

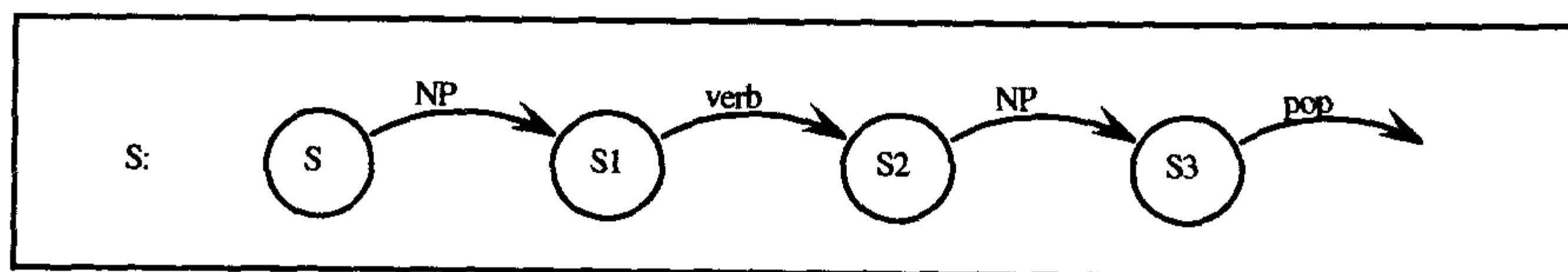
$NP1 \rightarrow N$

现在,让我们考虑用这个转移网络来分析名词短语“a purple cow”(紫色的奶牛)。从节点 NP 开始,当前词语 a 是一个冠词,因此可以通过标为 art 的边。从节点 NP1 开始,可以根据形容词“purple”通过标为 adj 的边。最后,还是从 NP1 开始,可以根据名词“cow”通过标为 noun 的边。现在,已经到达了弹出边。因此,“a purple cow”是合法的 NP。

简单的转移网络通常也叫做有限状态自动机(FSM, finite state machine)。有限状态自动机的表达能力和正则语法相同(参见框 3.2)。因此,有限状态自动机还不够强大,还不能描述所有能被上下文无关文法表达的语言。要想获得上下文无关文法的表达能力,需要在网络语法中引进递归的概念。递归转移网络(RTN, recursive transition network)类似于简单转移网络,不同的是,递归转移网络中的边不仅可以标记为词类,还可以标记为其他的网络。因此,在语法 3.16 的 NP 网络的基础上,还可以定义一个表示简单英语句子的转移网络,如语法 3.17 所示。其中,边上大写的标记指向转移网络。只有当 NP 网络成功地到达弹出边之后,从节点 S 到节点 S1 的边才可以通过。RTN 允许真正的递归——网络上的边可以标记为自身所在的网络名称,尽管在本例中没有给出递归。



语法 3.16



语法 3.17

现在,让我们来找找句子“The purple cow ate the grass”(紫色的奶牛吃草)在转移网络 S 中的路径。从节点 S 开始,我们需要通过标记为 NP 的边,所以必须穿过 NP 网络。从节点 NP 开始,像前面的短语分析一样,输入的短语“the purple cow”遍历了网络 NP。在 NP 网络中,沿着弹出边,我们又回到网络 S,沿着 NP 边到达节点 S1。从节点 S1 出发,依据动词“ate”,可以通过标为 verb 的边。最后,如果可以再次穿过 NP 网络,就可以通过当前这条 NP 边了。现在,剩下的

输入串为“the grass”。在 NP 网络中,可以依次通过 art 边和 noun 边。然后,分别从节点 NP2 和节点 S3 通过弹出边。到现在为止,我们已经遍历了整个网络,而且用到了句子中所有的词。因此,“The purple cow ate the grass”可以作为合法的句子被接受。

实际上,RTN 系统中还另外集成了一些有用的边类型。不过,从形式上说,这些边类型并非是必不可少的。图 3.18 中总结了各种边的类型,以及表示边类型的标记体系,我们会在本书中使用这些标记。根据这个术语体系,我们称标记网络的边为压入边(push arc),标记词类的边称为词类边(cat arc)。另外,永远都能成功通过的边称为跳跃边(jump arc)。

边的类型	举例	用法
CAT	noun	仅当当前词语属于标记的词类时成功
WRD	of	仅当当前词语与标记一样时成功
PUSH	NP	仅当穿过该命名网络时成功
JUMP	jump	永远成功
POP	pop	成功,并给出整个网络成功结束的信号

图 3.18 递归转移网络中的边标记

3.5.1 基于递归转移网络的自顶向下句法分析方法

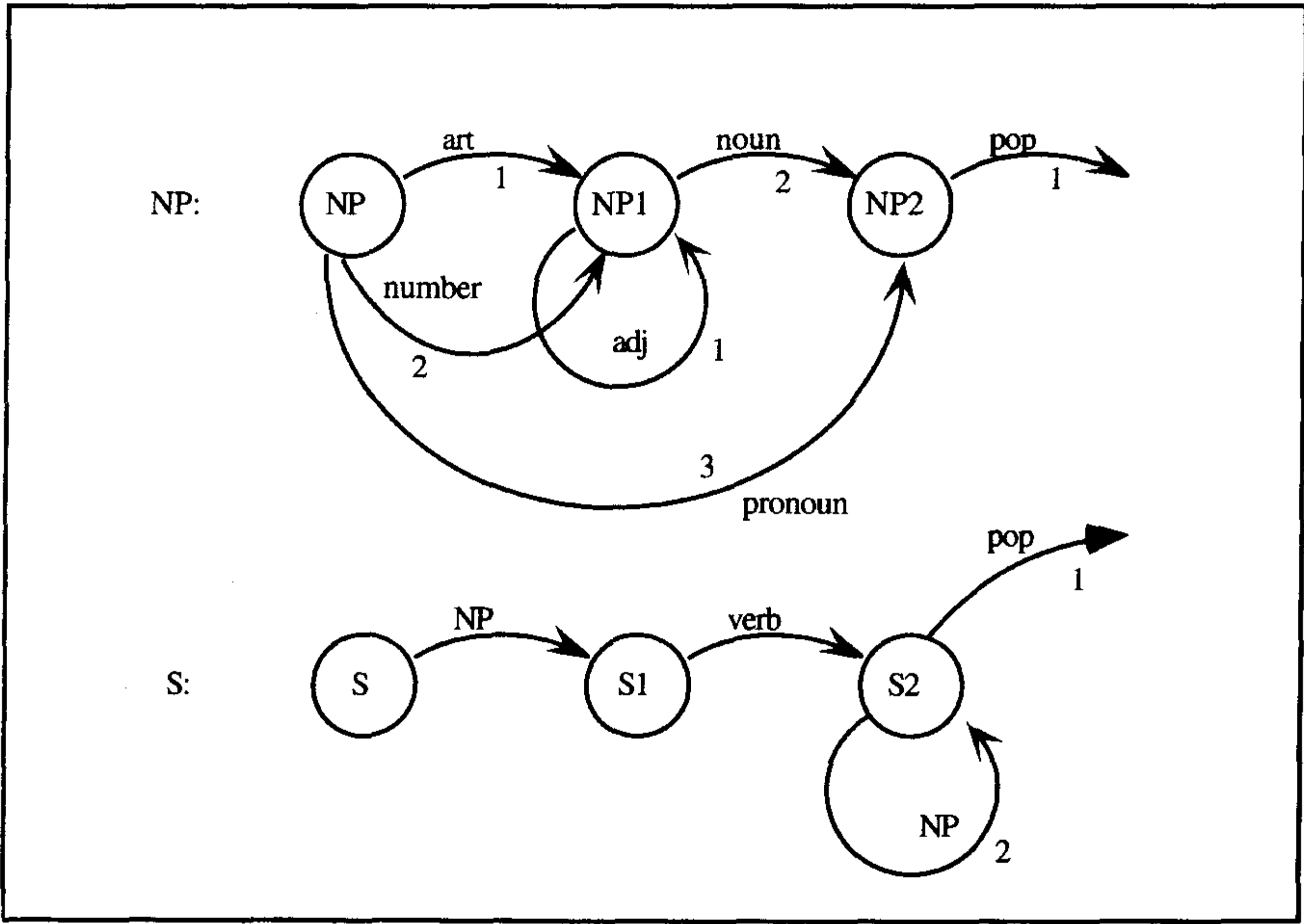
基于递归转移网络的句法分析算法可以采取和上下文无关文法分析一样的方法路线。其任何一个时刻的分析状态均可以表示为：

- 当前位置——指向下一个待分析词的位置指针。
- 当前节点——在转移网络中当前所在的节点。
- 返回点——从当前网络中返回退出后进入其他网络的节点栈。

首先,让我们考虑一种 RTN 的搜索算法。这种算法假定一旦可以沿着一条边往前推进,那么在最终的句法分析中,它将是正确分析结果中的一步。假定正处在句法分析过程的一个中间状态,并且知道我们刚才提到的三条信息。在下列情形中,可以离开当前的节点,并成功穿过一条边：

- 情况 1:如果当前边标记的是词类,并且句子中的下一个词就属于该词类,则：
 - (1)更新当前位置,当前位置从下一个词开始。
 - (2)更新当前节点,当前节点指向当前边的终点。
- 情况 2:如果当前边是指向转移网络 N 的压入边,则：
 - (1)将当前边的终点加入到返回点栈的顶部。
 - (2)更新当前节点,当前节点指向转移网络 N 的起点。
- 情况 3:如果当前边是弹出边,而且返回点栈非空,则：
 - (1)取出返回点栈中的第一个返回点,并将其设为当前节点。
- 情况 4:如果当前边是弹出边,返回点栈为空,而且没有词语剩下,则：
 - (1)句法分析成功。

语法 3.19 给出了一个网络语法。其中,有向边的上面有一些数字,其作用体现在当节点存在多条出边的时候,边上的数字表示了分析器尝试访问该边的先后次序。



语法 3.19

给定例句“₁ The ₂ old ₃ man ₄ cried ₅”,图 3.20 给出了算法生成的整个分析状态序列。最终结果表明该句可以被语法接受,是合法的句子。在整个过程中,起始节点和访问次序表示惟一的一条边。因此,S/1 边表示的是从节点 S 出发,标记为 1 的边。如果从节点 S 开始分析,接下来要通过的边只有压入边 NP。根据算法的情况 2,计算之后,新的分析状态设置为当前节点为 NP,并且将节点 S1 置于返回点栈中。从节点 NP 出发,我们先尝试通过边 NP/1,根据算法第一种情况的处理方法,需要检查输入的词是否属于冠词。检查成功,则该边通过,并修改当前位置(步骤 3)。按这种方式继续分析到步骤 5,此时遇到弹出边,则重新设置当前节点为 S1(也就是说,NP 边分析成功)。在步骤 6 中分析算法找到了一个动词,并且在步骤 7 中遇到了 S 网络的弹出边,最终句法分析成功。

步骤	当前节点	当前位置	返回点	要通过的边	注释
1.	(S,	1,	NIL)	S/1	原始位置
2.	(NP,	1,	(S1))	NP/1	通过压入边进入到 NP 转移网络,最后要返回到 S1
3.	(NP1,	2,	(S1))	NP1/1	通过 NP/1 边(the)
4.	(NP1,	3,	(S1))	NP1/2	通过 NP1/1 边(old)
5.	(NP2,	4,	(S1))	NP2/2	NP1/1 边不可行,因此,尝试通过 NP1/2 边(man)
6.	(S1,	4,	NIL)	S1/1	弹出边让我们返回到 S1
7.	(S2,	5,	NIL)	S2/1	通过 S2/1 边(cried)
8.					从 S2 出发,访问到弹出边,最后句法分析成功

图 3.20 自顶向下的句法分析流程

在这个例子中,分析成功的原因在于:成功分析出来的第一条边在最后的每一种可能分析结果中恰恰都是正确无误的。然而,如果分析的句子是“The green faded”,那么,该算法最后会分析失败。因为分析算法开始就将 green 归为形容词,接下来又找不到名词,而实际上 green 既可以是形容词,也可以是名词。要解决此类问题,需要在算法的运行中保存尽可能多的后续状态,而我们在 CFG 自顶向下句法分析算法中就是这样做的。

现在,我们采用这项技术来分析下面的句子:

1 One 2 saw 3 the 4 man 5

句法分析器最初会试着将句子的开始片段“one saw”分析为名词短语,但是找不到相应的动词。因此,算法回溯,从名词短语“one”开始,最终找到成功的分析结果。整个分析的流程如图 3.21 所示。在每个步骤当中,当前分析状态采用三元组(当前节点、当前状态和返回点)的形式表示。同时,我们还给出了各种可能的回溯状态。图中也给出了用来产生新状态和对应后备状态的各条边。

步骤	当前状态	要通过的边	后备状态
1.	(S,1,NIL)	S/1	NIL
2.	(NP,1,(S1))	NP/2(NP/3 作为后备)	NIL
3.	(NP1,2,(S1))	NP1/2	(NP2,2,(S1))
4.	(NP2,3,(S1))	NP2/1	(NP2,2,(S1))
5.	(S1,3,NIL)	没有边可以通过	(NP2,2,(S1))
6.	(NP2,2,(S1))	NP2/1	NIL
7.	(S1,2,NIL)	S1/1	NIL
8.	(S2,3,NIL)	S2/2	NIL
9.	(NP,3,(S2))	NP/1	NIL
10.	(NP1,4,(S2))	NP1/2	NIL
11.	(NP2,5,(S2))	NP2/1	NIL
12.	(S2,5,NIL)	S2/1	NIL
13.	句法分析成功		NIL

图 3.21 可回溯的自顶向下的 RTN 句法分析

本例的跟踪流程和前面的例子基本一样,不同之处有两点。在步骤 2 中,从节点 NP 出发,有两条边可以接受词语“one”。边 NP/2 将“one”归入数词并产生了下一个当前节点;而边 NP/3 认为它是个名词,并生成一个后备状态。实际上,在后续过程中,当我们发现从节点 S1 出发的边没有一条能接受输入词“the”时,就在步骤 6 中用到了该后备状态。

当然,一般来说,句子产生的后备状态会比这个简单的例子多得多。这种情况下,一般都会有一个可能后备状态表。按照表的组织方式,可以设定访问后备状态的不同顺序。

为了利用 chart 句法分析的长处,我们还可以采用类似 chart 的结构,构造一个 RTN 分析器。在 RTN 系统中,这种 chart 通常称为良构子串表(WFST,well-formed substring table)。当通过弹出边的时候,当前分析的成分就加入到 WFST 中;而当我们遇到压入边的时候,在调用子网之前,需要核对 WFST。如果 chart 中已经包含了压入边类型的成分,就直接利用这些成分,而不重复调用子网。采用 WFST 的 RTN 与上一节中介绍的 chart 分析器具有相同的复杂度 $K * n^3$ (数学表达式,指数形式),其中 n 是句子的长度。

◦ 3.6 自顶向下的 chart 句法分析

迄今为止,针对上下文无关文法的分析,你已经了解了简单的自顶向下分析方法,以及基于 chart 的自底向上分析方法。这两种方法各有利弊。本节要提供一种新的分析方法,该方法结合了两者的优点。讨论之前,我们先要评价这些方法的优劣。

自顶向下的方法有很好的预测能力。孤立的一个词可能会有歧义,但是,在所有可能的词类中,如果有一些词类在合法的句子中都不会使用到,那么,这些词类很可能永远都不会被考虑到。例如,以语法 3.8 为基准,我们采用自顶向下的方法分析“The can holds the water”。其中,“can”可以是 AUX(助动词),V 或者是 N,这和以前是一样的。

自顶向下的分析器会将(S)改写为(NP VP),随后分析 NP,此时存在三种可能性(ART ADJ N VP),(ART N VP)和(ADJ N VP)。取出第一种形式,分析器需要检测的是,第一个词“the”能否用做 ART(冠词),接着要检查下一个词“can”是否是 ADJ,最后检测失败返回。因此,需要尝试第二种可能的解释,分析器需要再次检测“the”,然后再检测“can”是否能归为 N,这次检测成功。我们发现,在自顶向下的分析过程中,“can”这个词在当前位置上,绝对不会解释成 AUX 或者 V。其原因在于,根据语法生成的句法树,任何一棵树都不会在当前位置预测到会出现 AUX 或者 V。与此形成对比的是,自底向上的算法会考虑所有的三种解释——也就是说,这三种解释都会加到 chart 中,并与活动边进行结合。从这个角度来讲,自顶向下的算法好像效率更高。

另外一方面,你会发现在上述例子中,自顶向下的句法分析器对“the”是不是 ART 做了两次检测判断,其中,每个规则检测一次。在纯粹的自顶向下方法中,操作的重复出现非常普遍,这就成了一个很严重的问题:对于大的语法成分来说,它会在不同的规则使用中一遍又一遍地重构。与此形成对比的是,自底向上的算法只对输入串进行一次检测,而且每个成分也只构造一次。所以,从这个角度来说,自底向上的方法看上去好像更加高效。

对这两个方法进行有机的融合,最终可以获得两者各自具有的优势。我们可以对自底向上的 chart 算法进行一次小的改动,最终得到和自顶向下方法类似的预测能力,而且还可以避免自底向上方法的重复操作。

和前面的算法一样,改进算法也是由完备成分的待处理表以及活动边扩展算法驱动的。在成分加入到待处理表的时候,需要将它们与活动边进行合并。在此,这两种算法都采用了通过成分扩展活动边的方法,不同之处主要表现在新活动边从语法中生成的方式上。在自底向上方法中,新增一个完成匹配的成分,而且该成分是某条规则右部的第一个成分时,就增加一条新的活动边。而在自顶向下的方法中,只要有新的活动边加入到 chart 中,就可以生成一条新的活动边。自顶向下的边导入算法给出了具体的描述,如图 3.22 所示。在此基础上,句法分析算法也就很容易表述了,具体的算法也在图 3.22 中给出。

采取新的算法,在 3.4 节给出的语法基础上,让我们来重新分析一下 3.4 节中的例句:“The large can can hold the water”。在开始阶段,加入一条标记为 $S \rightarrow \circ NP VP$ 的边。然后,根据每条可以产生 NP 的规则,分别增加活动边 $NP \rightarrow \circ ART ADJ N$, $NP \rightarrow \circ ART N$ 和 $NP \rightarrow \circ ADJ N$,这些活动边的跨度都是从位置 1 到位置 1。这样,我们可以得到初始的 chart,具体形式如图 3.23 所示。整个句法分析的流程如下:

输入 ART1(the),跨度为从位置 1 到位置 2

采用边扩展算法,可以对下面两条边进行扩展:

NP → ART ◦ N, 跨度为从位置 1 到位置 2
NP → ART ◦ ADJ N, 跨度为: 从位置 1 到位置 2
输入 ADJ1(large), 跨度为从位置 2 到位置 3
有一条边可以扩展:
NP → ART ADJ ◦ N, 跨度为从位置 1 到位置 3
输入 AUX1(can), 跨度为从位置 3 到位置 4
忽略该成分, 不执行任何操作。
输入 V1(can), 跨度为从位置 3 到位置 4
忽略该成分, 不执行任何操作。
输入 N1(can), 跨度为从位置 3 到位置 4
对其中一条边进行扩展, 匹配完成, 生成从位置 1 到位置 4 的 NP1(“the large can”)。
输入 NP1, 跨度为从位置 1 到位置 4
有一条边可以扩展:
S → NP ◦ VP, 跨度为从位置 1 到位置 4
采用自顶向下规则(步骤 4), 给 VP 加上新的活动边
VP → ◦ AUX VP, 跨度为从位置 4 到位置 4
VP → ◦ V NP, 跨度为从位置 4 到位置 4

自顶向下的边导入算法
增加一条末尾位置为 j 的边 $S \rightarrow C_1 \dots \circ C_i \dots C_n$, 执行如下步骤:
对于语法中每一条形式为 $C_i \rightarrow X_1 \dots X_k$ 的规则, 从位置 j 到位置 j 递归地加入新的边 $C_i \rightarrow \circ X_1 \dots X_k$ 。

自顶向下的 chart 句法分析算法:
初始化: 对于语法中每一条形式为 $S \rightarrow X_1 \dots X_k$ 的规则, 采用边导入算法加入标记为 $S \rightarrow \circ X_1 \dots X_k$ 的边。
句法分析过程: 执行如下操作, 直至输入为空:
1. 如果待处理表为空, 查找下一个词的解释, 并将它们全部加入到待处理表中。
2. 从待处理表中选择一个语法成分(我们称之为 C)。
3. 采取边扩展算法, 将 C 与 chart 中的每条活动边相结合。任何一个新生成的成分都要加到待处理表中。
4. 对于步骤 3 中新增的每条活动边, 采用自顶向下的边导入算法将它们加入到 chart 中。

图 3.22 自顶向下的边导入算法以及 chart 句法分析算法

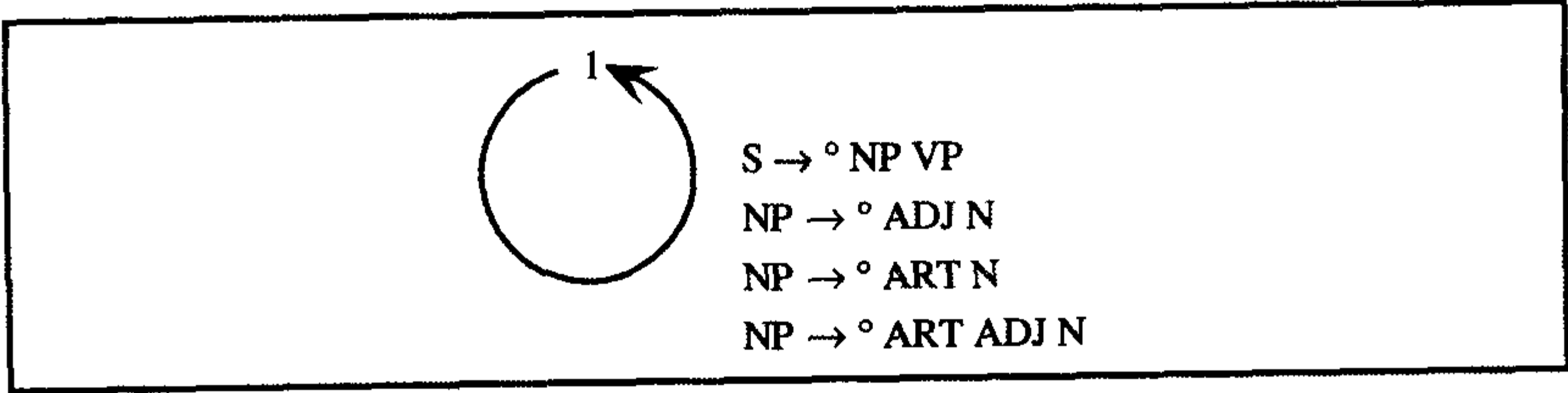


图 3.23 初始的 chart 图

在这个阶段, 其 chart 图如图 3.24 所示。与图 3.10 相比, chart 图中新增的已匹配成分要少一些, 这是因为 chart 中仅仅构建了那些自顶向下算法可以接受的语法成分。

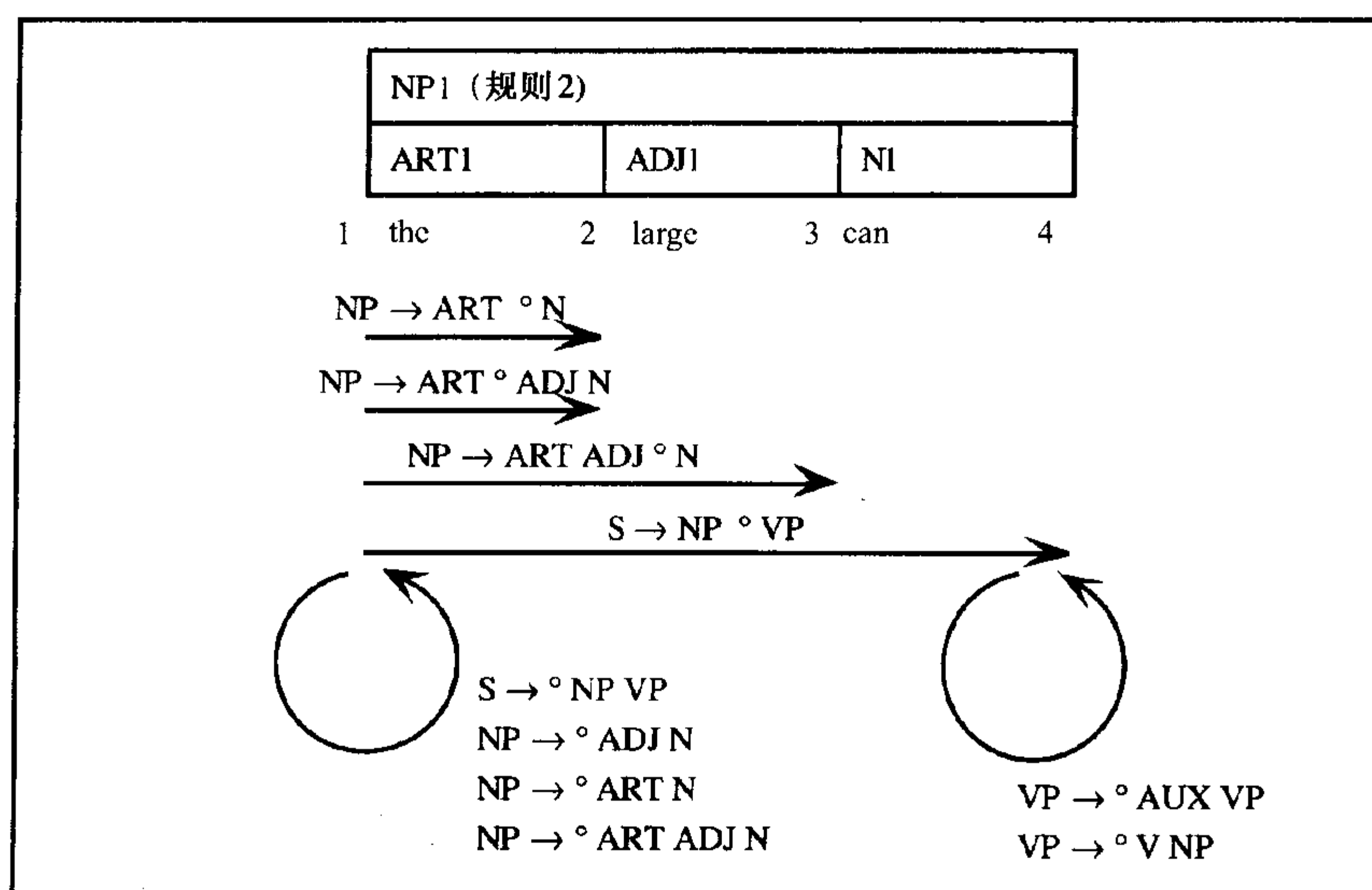


图 3.24 构建了第一个 NP 之后的 chart 图

算法继续往下分析处理。接下来,我们要处理“can”。“can”有三种语法解释:AUX,V 和 N。读入 AUX 之后,扩展了位置 4 处的活动边 $VP \rightarrow \circ AUX VP$,同时增加了一条从位置 5 出发的 VP 活动边。读入 V 之后,扩展了活动边 $VP \rightarrow \circ V NP$,同时增加了一条从位置 5 出发的 NP 活动边。读入 N 之后,不扩展任何边,“can”的这种解释可以忽略掉。增加“hold”的两种解释(V 和 N)之后,当前的 chart 图如图 3.25 所示。再次与图 3.13 自底向上分析器对应的 chart 相比较,句子剩下部分的句法分析与此类似,图 3.26 给出了最终的 chart 图。将该图与自底向上分析器的最终 chart 图(见图 3.15)做比较可以发现,自底向上 chart 分析器生成的语法成分由 21 个减少到了 13 个。使用如此简单的语法时,差别不会很大。一旦使用规模稍大的语法,这种差别就会相当显著。

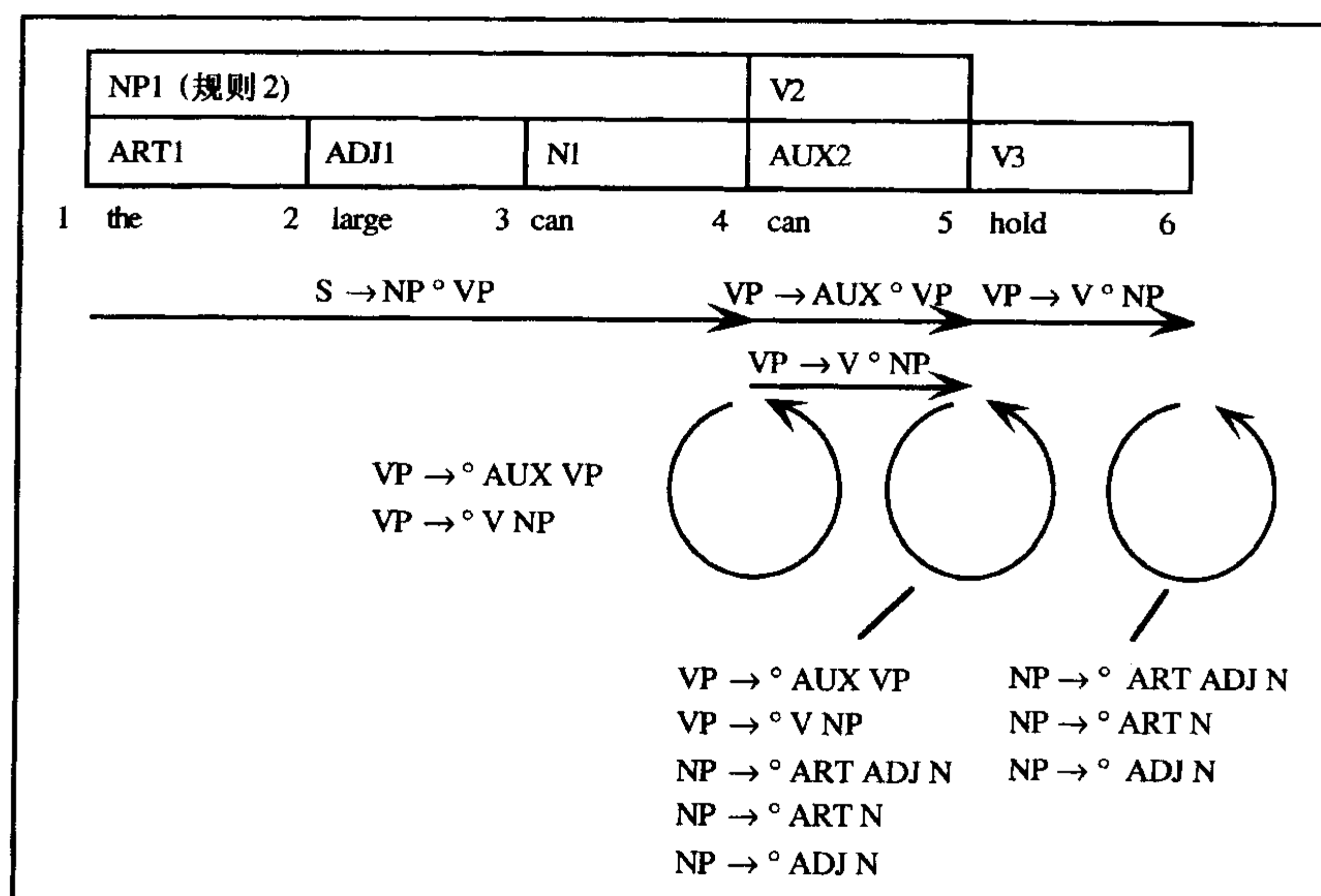


图 3.25 加入 hold 之后的 chart 图,这里省略了第一个 NP 的边

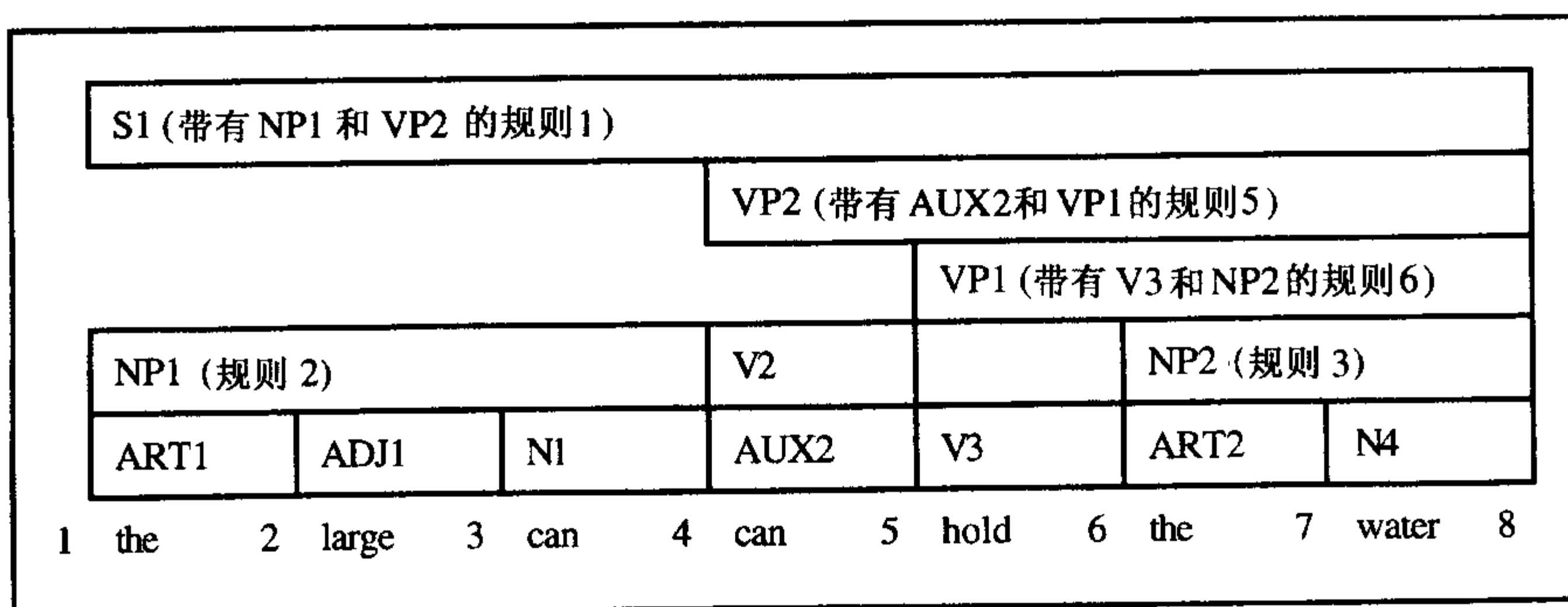
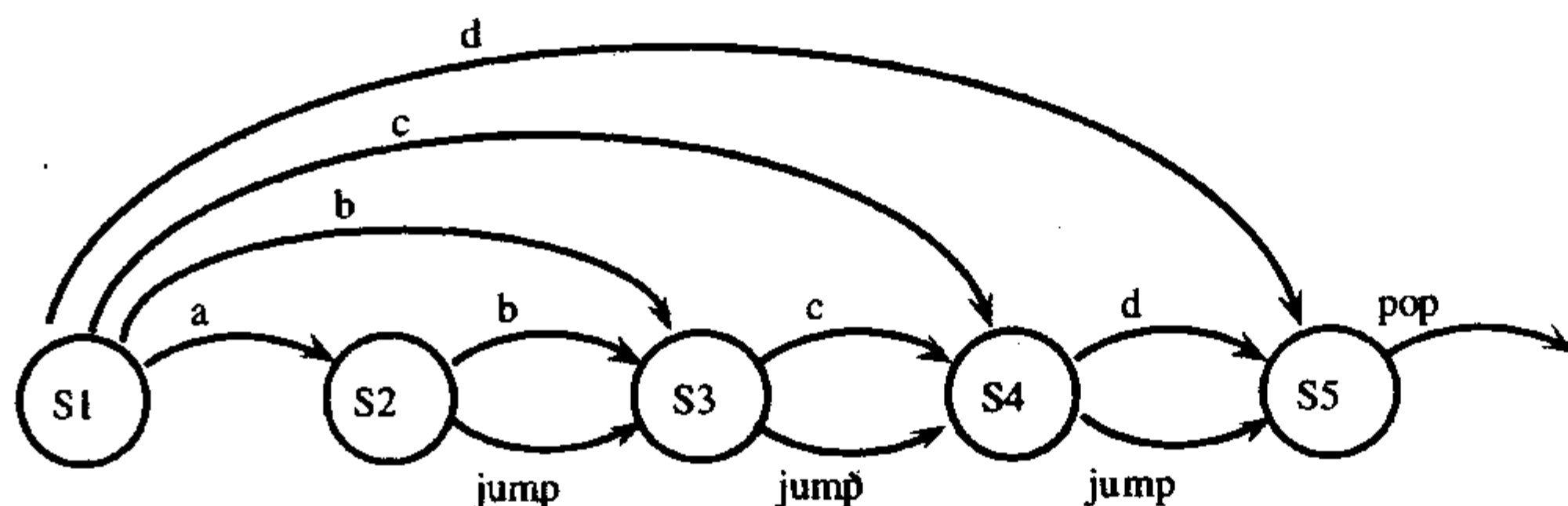


图 3.26 自顶向下过滤分析算法最终的 chart 图

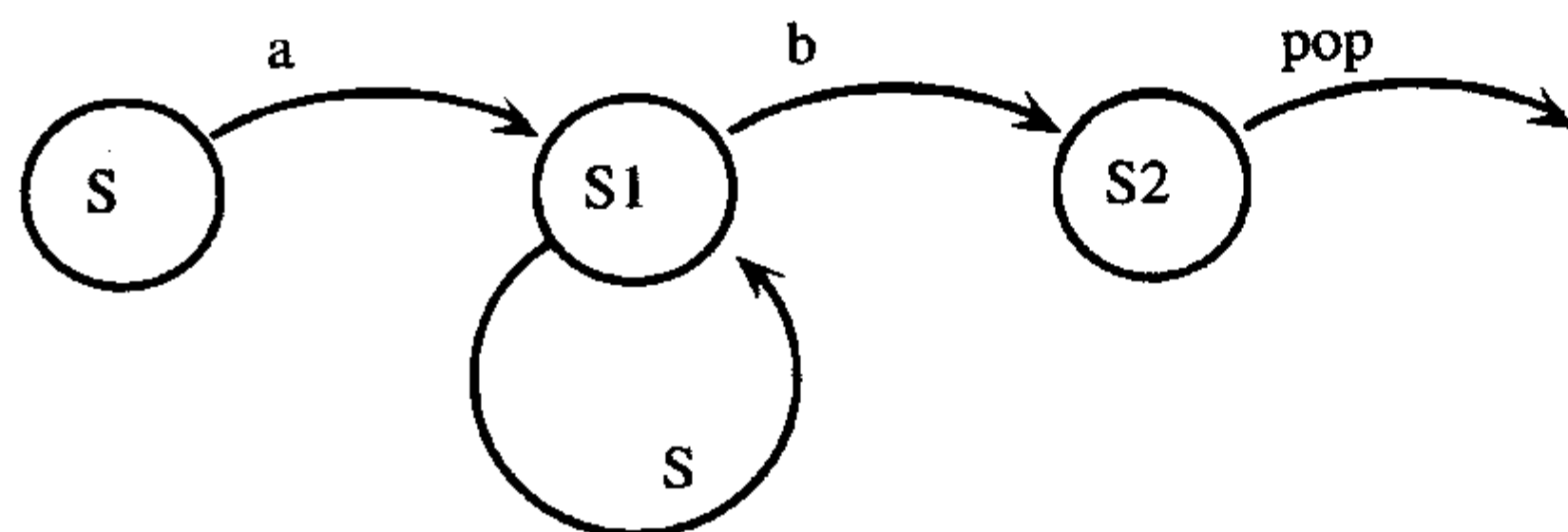
实际结果表明:在最坏的情况下,自顶向下 chart 句法分析器的效率并不比纯粹的自底向上句法分析器更高。对于一个长度为 n 的句子,两者最坏情况下的复杂度都是 $K * n^3$ 。不过,对于任何一个合理的语法来说,自顶向下的方法实际上都是非常高效的。

框 3.2 转移网络的语法生成能力

转移网络系统可以按照其所能描述的语言类型进行分类。实际上,可以在不同的转移网络系统和产生式规则系统之间建立对应的关系。比如,没有压入边的简单转移网络(即有限状态自动机)和正则文法的表达能力对等。也就是说,每一个可以用简单转移网络描述的语言,均可以采用正则文法描述,反之亦然。如果采用有限状态自动机描述框 3.1 中的第一个语言,则结果如下:



另外一方面,递归转移网络和上下文无关文法表达能力相同。因此,递归转移网络可以转化为上下文无关文法,反之亦然。下面的递归转移网络可以用来表达的语言是一定数目的“a”,后面跟着相同数目的“b”。



3.7 有限状态模型与词语形态处理

在简单的例子和小型的系统中,我们可以列出所有的词语。但是,大词汇量的系统在词典表示方面存在一个很严重的问题。实际的系统中不仅存在大量的词汇,而且每个词都可能结

合各种形式的词缀,形成另外的相关词语。解决这个问题的一种方法是对输入句子进行预处理,生成一个词素序列。词语可能仅由单个词素组成,但通常是由词根加上词缀形成的。例如,“eaten”就由词根“eat”和表示过去分词的后缀“en”组成。如果不经预处理,则词典必须列出“eat”的所有形式,其中包括“eats”,“eating”,“ate”和“eaten”。如果存在预处理过程,词典中仅需要存储词素“eat”,它可以与后缀“ing”,“s”和“en”结合。另外,可能还需要增加一个词条来存储其不规则形式“ate”。因此,词典存储的不是四个而是两个词条(“eat”和“ate”)。类似地,“happiest”这个词可以分解为词根“happy”和后缀“est”。这样,词典就不需要另外增加一个独立的词条。当然,并不是所有形式都是可以接受的。例如,“seed”就不能分解为词根“se”(或“see”)和后缀“ed”。对于每个词根来说,词典只需要记录它究竟可以接受什么样的形式。

基于有限状态转录机(FST, finite state transducer)的方法就是一个广为人们所接受的词典表示模型。有限状态转录机和有限状态自动机相似,不同的是转录机对于给定的输入可以产生一个输出。有限状态转录机中的边均采用一对符号来标记。例如,当输入为*i*时,标为*i:y*的边可以通过,而且输出*y*。有限状态转录机可以简明地表示词典,并且可以将词的各种表面形式转换成词素序列。图3.27给出了一个定义happy及其派生形式的简单FST,该转录机可以将“happier”转换为序列“happy + er”;同时,还能将“happiest”转换为序列“happy + est”。

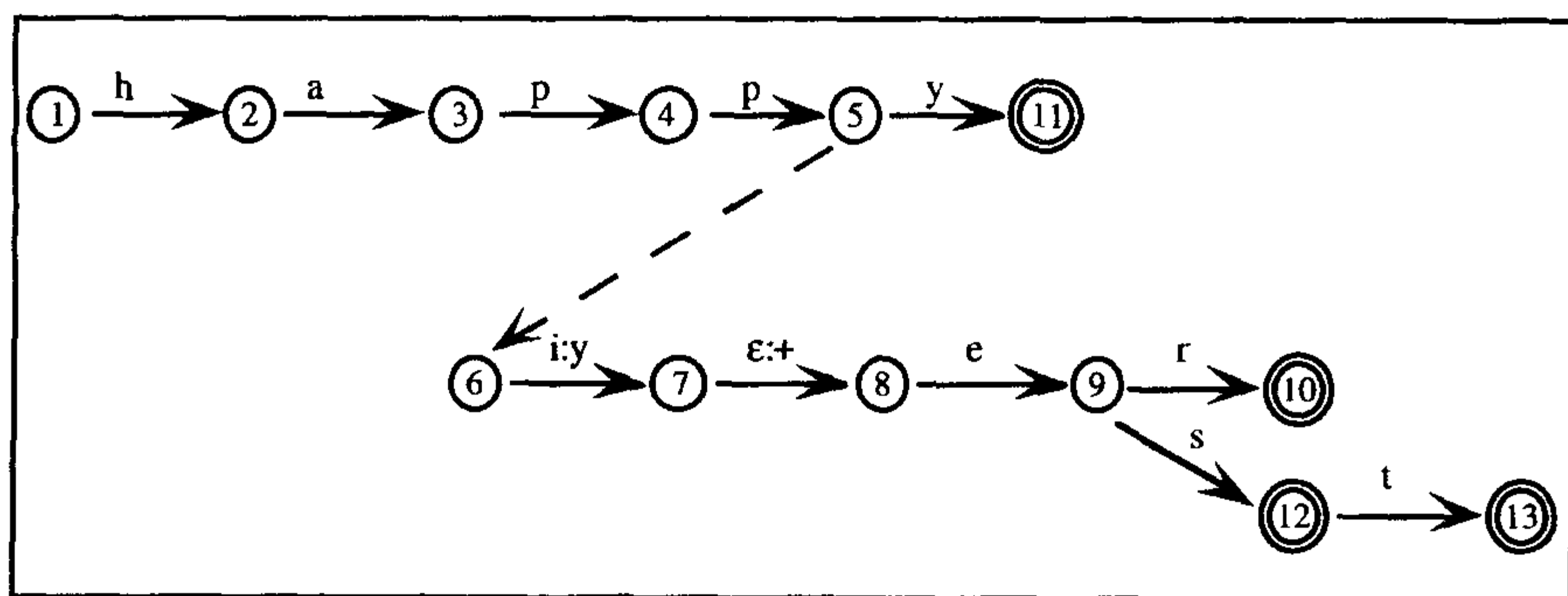


图 3.27 表示 happy 各种形式的简单有限状态转录机

标记为单个字母的边,其输入和输出都为该字母。标记为双圈的节点表示的是成功状态,即可以接受的词语。从状态1开始,我们来看看输入词语“happier”是如何处理的。上边的网络接受前面四个字符“happ”,并将输入复制为输出结果。从第5个字符开始,可以接收“y”生成一个完整的词语,或者跳到状态6准备考虑词缀。(这个虚线连接表示跳跃,通常来说并不必要,但是这样处理有利于表明词根分析与词缀分析之间的间断与转换。)在这里,对于输入词“happier”,必须跳到状态6,下一个字符必须是“i”,该字符被转换为*y*。接下来的输入为空(空符号ε)而输出为加号。从状态8开始,输入必须为“e”,而输出也是“e”。紧接着的是“r”,分析器转移到状态10,该状态用双圈表示,意味着可能到达了词尾(即FST的成功状态)。因此,FST接受相应的形式,输出了我们需要的词素序列。

整个词典都可以按照这种方式加工成FST,FST能表示所有合法的输入词语,并且可以将它们转换为相应的词素序列。有限状态转录机的不同后缀只须定义一次,所有可接受同一后缀的词根最后可以指向同一节点。前缀相同的词(如“torch”,“toss”和“to”)也可以共用一个节点。这样处理之后,网络的规模就大大缩减了。图3.28所示的FST可以接受如下以*t*开头的

词:“tie”(状态 4),“ties”(10),“trap”(7),“traps”(10),“try”(11),“tries”(15),“to”(16),“torch”(19),“torches”(15),“toss”(21),“tosses”(15)。此外,该转录机还可以输出对应的词素序列。

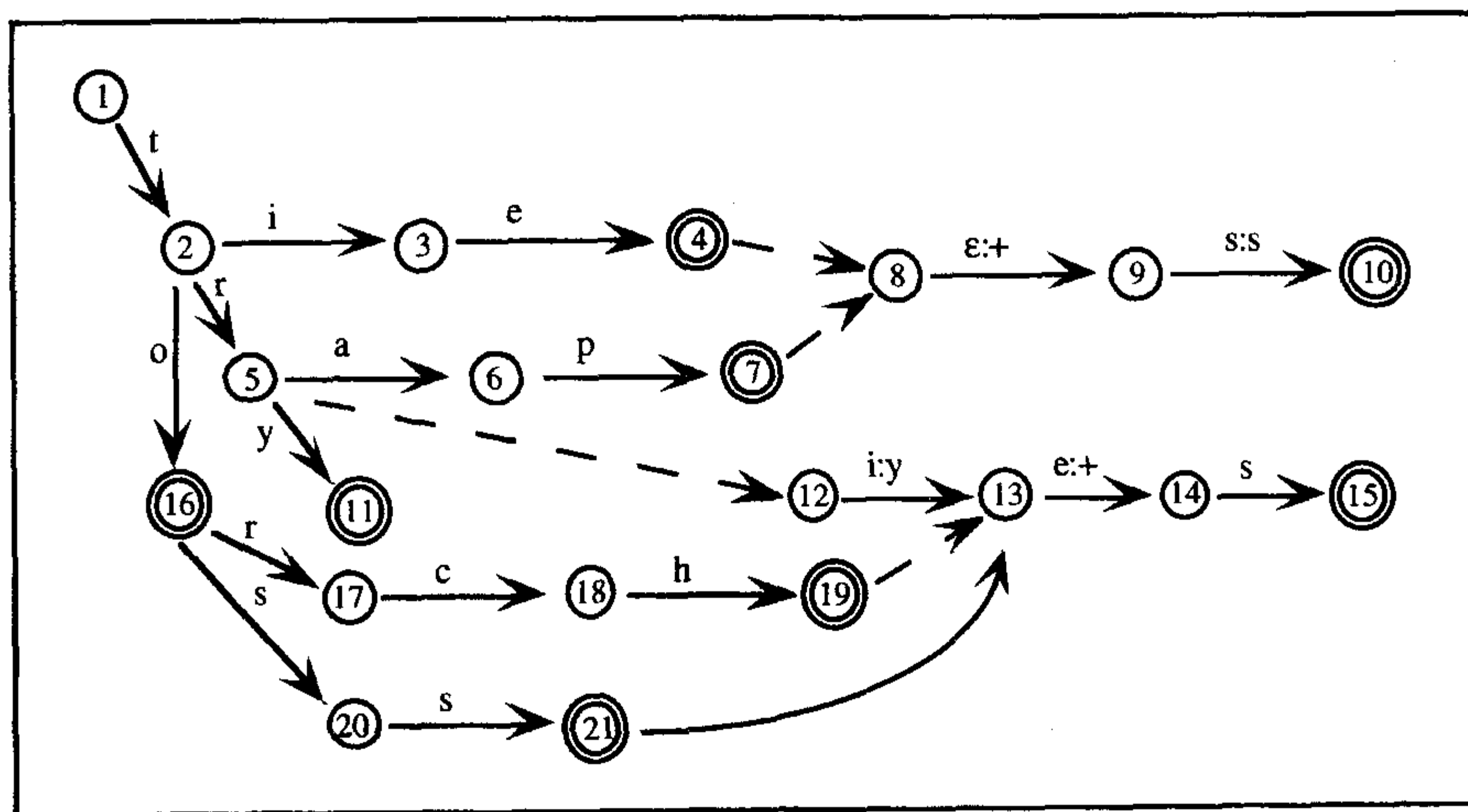


图 3.28 一些名词(单数和复数)定义的 FST 片段

需要注意的是,在词的处理过程中可能会通过某些可接受的成功状态。例如,输入词为“toss”时,就会通过状态 16,该状态表明“to”是个可接受的合法词语。然而,这个分析是无效的。原因是,如果“to”作为一个词被接受,那么就无法再考虑剩下的字母“ss”了。

采用这样的 FST,输入的句子就能够分析为一个词素序列。词语在极少数情况下存在歧义,它可以分解成多种不同的词素。然而,这种情况相当少。在本书中,我们忽略这种极少数的复杂情况。

3.8 语法与逻辑程序设计

搭建上下文无关文法分析器的时候,人们常常采取另外一种方法,即将语法规则直接用逻辑程序语言来表示,如 PROLOG。可以证实的是,标准 PROLOG 的解析算法使用的搜索策略恰恰就是深度优先的自顶向下分析算法。所以,需要做的工作仅仅是寻找将上下文无关文法规则重新整理成 PROLOG 子句形式的方法。考虑如下的上下文无关文法规则:

$$S \rightarrow NP VP$$

采用公理的形式,这条规则可以重新表述为:“一个词的序列是合法的 S。前提是该序列开头部分是一个合法的名词短语,其后跟着一个合法的动词短语。”如果按照位置顺序给句中的每个词编号,这个规则可以重述为:“如果存在位置 p2,使得 p1 和 p2 之间有一个 NP,同时 p2 和 p3 之间有一个 VP,那么 p1 和 p3 之间是一个 S。”在 PROLOG 中,这条规则可以表示为下面的公式。其中,变量采用大写字母开头的字符串表示:

$$s(P1, P3) \text{ :- } np(P1, P2), vp(P2, P3)$$

为了建立这个分析过程,我们还要增加公理来列出句中所有的词及其在句中的位置。例如,句子“John ate the cat”可以表述为:

```
word(john, 1, 2)
word(ate, 2, 3)
word(the, 3, 4)
word(cat, 4, 5)
```

词典可以用一系列的谓词来定义,结果如下:

```
isart(the)
isname(john)
isverb(ate)
isnoun(cat)
```

歧义词会产生多个断言——其中,每个断言对应它们所属的一个句法类别。

对于每个句法类别,可以定义一个谓词。该谓词为真的惟一条件是,在两个特定位置之间的词语属于该类别,如下所示:

```
n(I, O) :- word(Word, I, O), isnoun(Word)
art(I, O) :- word(Word, I, O), isart(Word)
v(I, O) :- word(Word, I, O), isverb(Word)
name(I, O) :- word(Word, I, O), isname(Word)
```

使用图 3.29 所示的公理,通过证明谓词 $s(1,5)$ 为真,可以证明“John ate the cat”是一个合法的句子,证明过程见图 3.30。图 3.30 中存在一些同名的不同变量,容易造成混淆,这时就给变量名加上一个单引号('),使之惟一。这个句子中谓词的证明流程和上下文无关文法的自顶向下分析流程是一致的,具体过程如下。任何时刻的待检验状态都可以表示为迄今为止需要检验的子目标列表。目标描述中已经包含了词的位置,因此在分析流程中,就不需要增加单独的列来记录词语的位置信息。后备状态也是采用子目标列表来表示的,由一个类似 PROLOG 的系统自动维护,并用来实现回溯过程。在这种系统中,典型的证明过程在任何时刻仅仅给出当前的状态。

1. $s(P1, P3) :- np(P1, P2), vp(P2, P3)$
2. $np(P1, P3) :- art(P1, P2), n(P2, P3)$
3. $np(P1, P3) :- name(P1, P3)$
4. $pp(P1, P3) :- p(P1, P2), np(P2, P3)$
5. $vp(P1, P2) :- v(P1, P2)$
6. $vp(P1, P3) :- v(P1, P2), np(P2, P3)$
7. $vp(P1, P3) :- v(P1, P2), pp(P2, P3)$

图 3.29 语法 3.4 基于 PROLOG 的表示

标准 PROLOG 搜索策略与深度优先的自顶向下的句法分析策略一样。因此,采用 PROLOG 构造的句法分析器与后者的计算复杂度都为 C^n ,即以计算的步数为底,输入句子的长度为指数。实际上,即使是在最坏的情况下,基于 PROLOG 的语法也是非常高效的。可以考虑加入类似 chart 的机制来进一步提高算法的效率。当然,这样做的结果会丧失上下文无关规则与 PROLOG 规则之间的一些简单联系。下一章将讨论这方面的一些问题。

步骤	当前状态	后备状态	注释
1.	s(1, 5)		
2.	np(1, P2) vp(P2, 5)		
3.	art(1, P2') n(P2', P2) vp(P2, 5)	name(1, P2) vp(P2, 5)	失败, 因为位置 1 上没有 ART
4.	name(1, P2) vp(P2, 5)		
5.	vp(2, 5)		name(1, 2) 得到证明
6.	v(2, 5)	v(2, P2) np(P2, 5) v(2, P2) pp(P2, 5)	失败, 因为没有 一个动词跨越位置 2 到 5
7.	v(2, P2) np(P2, 5)	v(2, P2) pp(P2, 5)	
8.	np(3, 5)	v(2, P2) pp(P2, 5)	v(2, 3) 得到证明
9.	art(3, P2) n(P2, 5)	name(3, 5) v(2, P2) pp(P2, 5)	
10.	n(4, 5)	name(3, 5) v(2, P2) pp(P2, 5)	art(3, 4) 得到证明
11.	√ 证明成功	name(3, 5) v(2, P2) pp(P2, 5)	n(4, 5) 得到证明

图 3.30 “John ate the cat”基于 PROLOG 算法的句法分析流程

尝试使用 PROLOG 写一些简单的语法,这是一件很值得我们去做的事情。由此,可以更好地理解自顶向下的深度优先搜索方法。通过 PROLOG 的跟踪工具,能得到和图 3.30 类似的算法运行过程。

3.9 小结

上下文无关文法和递归转移网络是两种基本的语法形式化方法,二者都适用于各种不同的句法分析算法。例如,简单的自顶向下回溯算法适用于这两种形式。事实上,同样的算法也可以用于标准的、基于逻辑编程的语法。最有效的句法分析器往往会在分析过程中使用一个类似于 chart 的结构,以便记录每一个在分析过程中构建的语法成分。在随后的搜索中,系统可以重复使用这种信息,从而可以节省大量的工作。

3.10 相关工作与深入阅读材料

目前,关于句法形式化和分析算法方面有大量的文献。上下文无关文法的概念是由乔姆斯基(Chomsky, 1956)引入的,此后,语言学和计算机科学对其进行了广泛而又深入的研究。这方面的一些研究工作会在后续的章节中进行详细的讨论,它们和后面章节的内容关联更紧密。大部分句法分析算法都是在 20 世纪 60 年代中期的计算机科学中发展起来的。不过,通常的研究目的是分析程序设计语言而不是自然语言。这方面的典型工作可以参考 Aho, Sethi 和 Ullman(1986)。如果找不到这份资料,也可以参阅 Aho 和 Ullman(1972)中的介绍。Kay (1973, 1980)率先描述了 chart 体系,之后,许多句法分析系统都使用并改进了 chart。本章中介绍的自底向上 chart 分析器与 Aho 和 Ullman(1972)中的左角(left-corner)分析算法非常接近,而

自顶向下的 chart 分析器也与 Earley(1970)介绍的方法类似。因此,自顶向下的 chart 分析也常称为 Earley 算法。

Woods(1970;1973)最先阐述了转移网络语法及其分析器,Pereira 和 Warren(1980)论述了基于逻辑程序设计的句法分析器,并和转移网络系统进行了比较。Winograd(1983)从一个稍微不同的角度讨论了这里提及的大部分算法。当某一专门技术很难理解时,这篇文章就显得非常有用。Gazdar 和 Mellish(1989a;1989b)叙述了采用 LISP 和 PROLOG 语言实现句法分析器的详细过程。另外,关于转移网络分析器的介绍在很多介绍人工智能的文章中都可以找到,例如 Rich 和 Knight(1992),Winston(1992)以及 Charniak 和 McDermott(1985)。这些书在介绍句法分析算法的同时,也会介绍搜索技术。Norvig(1992)是关于人工智能程序设计技术方面相当出色的一篇文献。

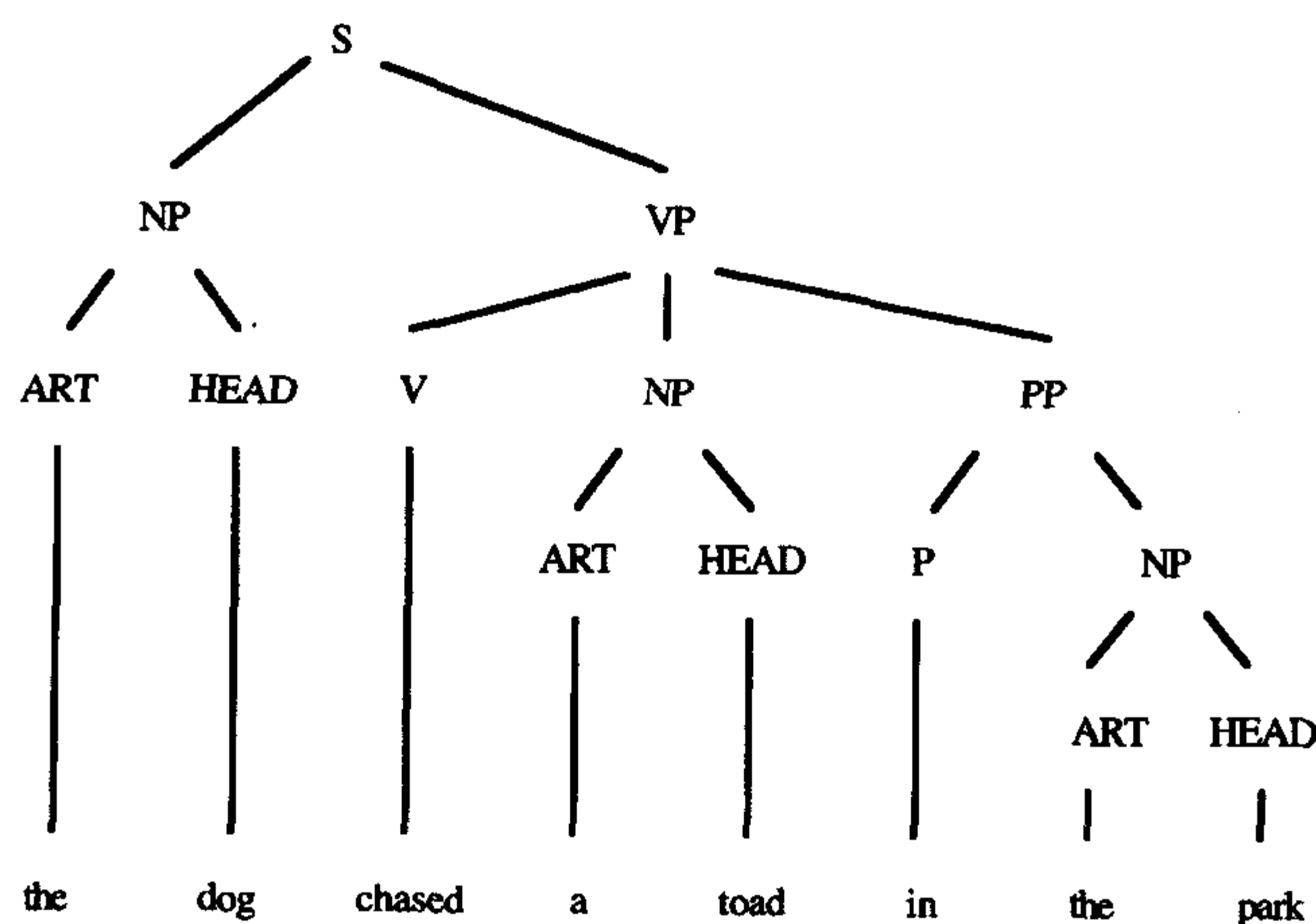
关于计算词语形态学方面的研究工作,最好的参考资源有两本书,Sproat(1992)和 Ritchie 等(1992)。在有限状态模型方面,大部分最近的研究工作都是在 KIMMO 系统(Koskenniemi, 1983)的基础上开展的。KIMMO 不需要建立大型的网络,而是采用能并行的 FST 集合,即所有的转录机可以同时接受输入并产生输出。一般来说,这些 FST 都是采用抽象语言表达的,这种语言可以简明地表示通用的词语形态规则。然后,我们就可以使用一个编译器为当前系统生成相应的网络。

有限状态模型除了用于词语形态分析(morphological analysis)之外,还可以广泛地应用于其他的处理任务。例如,Blank(1989)研制了一种仅仅使用有限状态表示的英语语法。有限状态语法也广泛地应用于语音识别系统之中。

3.11 习题

1. 【易】

a. 按 3.1 节中列表的表示方法,表示下面这棵树:



b. 是否存在不能用列表表示的树结构? 树无法表示的列表结构是什么样的?

2. 【易】已知语法 3.4 给定的上下文无关文法,请定义出相应的词典,并按照图 3.5 所示的格式,给出句子“The man walked the old dog”基于上下文无关文法,自顶向下的句法分析流程。

3. 【易】已知语法 3.19 所示的递归转移网络,同时知道 green 在词典中可以是形容词,也可以是名词。按照图 3.21 的形式,给出句子“The green faded”基于递归转移网络的自顶向下句法分析流程。
4. 【易】给定图 3.29 所定义的基于 PROLOG 的语法,按照图 3.30 所示的形式,给出“The cat ate John”是合法句子的证明流程。
5. 【中】只使用 S 网络、NP 网络和 PP 网络,将下面的上下文无关文法映射为相应的递归转移网络。要求生成的网络规模越小越好。

$S \rightarrow NP VP$	$NP2 \rightarrow ADJ NP2$
$VP \rightarrow V$	$NP2 \rightarrow NP3 PREPS$
$VP \rightarrow V NP$	$NP3 \rightarrow N$
$VP \rightarrow V PP$	$PREPS \rightarrow PP$
$NP \rightarrow ART NP2$	$PREPS \rightarrow PP PREPS$
$NP \rightarrow NP2$	$PP \rightarrow NP$
$NP2 \rightarrow N$	

6. 【中】给定习题 5 中的上下文无关文法和下面的词典,采用纯粹的自顶向下算法和纯粹的自底向上算法分别给出句子“The herons fly in groups”的分析流程。要求流程要尽量清晰,选择规则的时候按照习题 5 中给定的顺序,并且表示出全部的搜索部分。词典中的各个词条如下:

the: ART
herons: N
fly: N V ADJ
in: P
groups: N V

7. 【中】考虑下面的语法:

$S \rightarrow ADJS N$
 $S \rightarrow N$
 $ADJS \rightarrow ADJS ADJ$
 $ADJS \rightarrow ADJ$

词典: ADJ: red, N: house

- a. 对于这一语法,使用深度优先的自顶向下分析器,分析输入串“red red”,会产生什么结果? 具体地,要说出分析器最终是分析成功、分析失败还是永不停止?
- b. 如果使用广度优先的自顶向下分析器,对于同一输入,结果又会怎样?
- c. 使用广度优先的自顶向下分析器处理输入串“red house”,结果如何?
- d. 使用深度优先的自底向上分析器处理输入串“red house”,结果如何?
- e. 在分析器无法停止的情况下,请给出与本习题中等同的语法,并且保证能够正确分析。(正确的行为包括,在短语无法接受的条件下,返回失败;在短语可以接受的条件下,返回成功。)

f. 采用(e)中的新语法,前面所有的句法分析器是否都能正确地分析出前面的两个短语“red red”和“red house”?

8. 【中】讨论下面的上下文无关文法:

- $S \rightarrow NP\ V$
- $S \rightarrow NP\ AUX\ V$
- $NP \rightarrow ART\ N$

给出下面句子的 chart 句法分析流程:

₁ The ₂ man ₃ is ₄ laughing ₅

在该分析器中,用到的词典条目如下:

- the: ART
- man: N
- is: AUX
- laughing: V

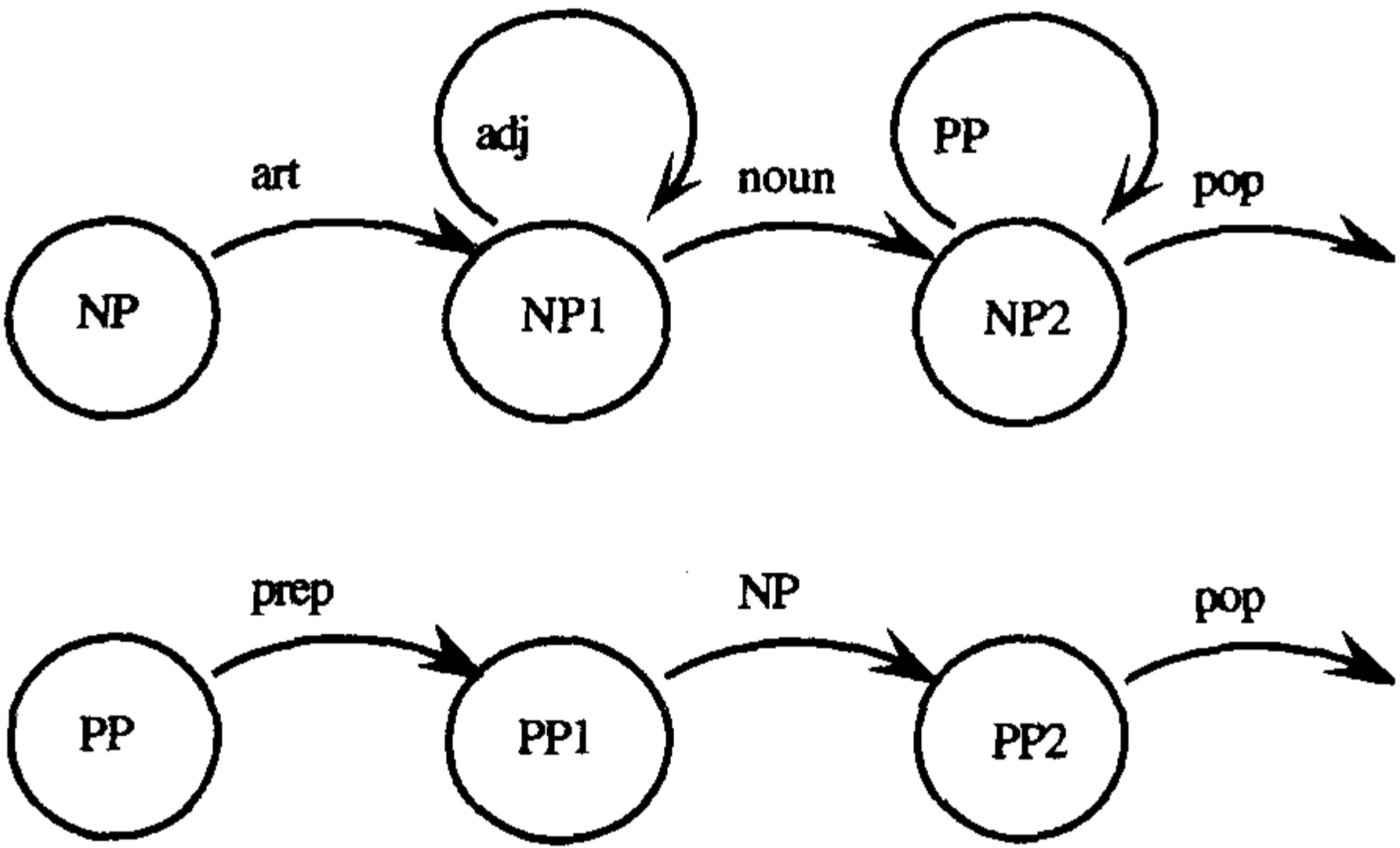
要求给出句法分析的每个步骤,给出分析工作栈,在每次添加非终结成分时,都要画出当时的 chart 图。

9. 【中】考虑下面这个能生成字母序列的上下文无关文法:

- $s \rightarrow a\ x\ c$
- $s \rightarrow b\ x\ c$
- $s \rightarrow b\ x\ d$
- $s \rightarrow b\ x\ e$
- $s \rightarrow c\ x\ e$
- $x \rightarrow f\ x$
- $x \rightarrow g$

- a. 如果要为这个语法编一个句法分析器,你觉得纯粹的自顶向下算法相对于纯粹的自底向上算法哪一个更好? 为什么?
- b. 采用你选择的分析器处理输入串“bffge”,并给出整个运行流程。

10. 【中】考虑下面的上下文无关文法和递归转移网络:



$NP \rightarrow ART\ NP1$

NP1 → ADJ N PPS

PPS → PP

PPS → PP PPS

PP → P NP

- a. 说出这两种语法描述的语言存在的两点不同之处。要求针对每一个不同点,分别给出一个例句,该例句可以被其中一个语法识别接受,而不能被另外一个语法识别。通过这种方式显示出这两种语法的不同点。
- b. 编写一个与该递归转移网络等同的上下文无关文法。
- c. 编写一个与该上下文无关文法等同的递归转移网络。

11. 【难】讨论下列句子:

列表 A

i. Joe is reading the book.

ii. Joe had won a letter.

iii. Joe has to win.

iv. Joe will have the letter.

v. The letter in the book was read.

vi. The letter must have been in the book by Joe.

vii. The man could have had one.

列表 B

i. * Joe has reading the book.

ii. * Joe had win.

iii. * Joe winning.

iv. * Joe will had the letter.

v. * The book was win by Joe.

vi. * Joe will can be mad.

vii. * The man can have having one.

- a. 编写一个上下文无关文法,要求该语法可以接受列表 A 中所有的句子,同时拒绝列表 B 中所有的句子。提示:参照第 2 章关于动词语法形式的探讨,或许会对你有所帮助。
- b. 使用(a)中给定的语法,实现一种基于 chart 的句法分析策略。需要论证你的分析器确实可以成功接受列表 A 中所有的句子,而拒绝列表 B 中所有的句子。你应当在 chart 的每一项中保存足够的信息,这样就可以为每种可能解释重构句法分析树。请确认你使用的结构记录方法组织得很好,而且能清晰地演示。
- c. 假设在(a)中的语法基础上实现了一个句法分析器,现在,要求给出三种(互不相同的)语法形式。该句法分析器无法识别这些语法形式。同时,对于每种语法形式,要求你自己提供一个例子。

第4章 特征与扩充语法

迄今为止,大部分的句法计算分析机制都是建立在上下文无关文法(CFG)基础之上的。但是,就目前我们所知道的情况来看,对于自然语言的获取与分析,上下文无关文法使用起来还是非常不方便的。本章在基本上下文无关文法机制的基础上,给出了一种扩展形式。在该扩展语法中,我们使用一个特征集合来定义各种成分,这种扩展可以让我们采用更直观、更简明的方式来处理自然语言中很多方面的问题,比如,一致性(agreement)和次范畴(subcategorization)。

4.1节介绍特征体系的概念,以及上下文无关文法加入特征后的泛化形式。4.2节论述英语中一些有用的特征体系,通常用在目前的各种语法之中。4.3节探讨词典定义方面的相关问题,并且说明如何采用特征使词典的定义更加简单。4.4节给出了基于特征的上下文无关文法示例,同时介绍一些用来简化处理过程的习惯用法。4.5节探讨如何将chart句法分析器扩展到含特征的语法中。余下一节属于可选内容,主要介绍特征如何应用于其他的语法体系,并探讨了一些深层次的问题。4.6节介绍扩充转移网络,扩充转移网络是一种泛化的、基于特征的递归转移网络。4.7节叙述基于PROLOG的子句语法。4.8节描述广义特征体系与合一语法。

4.1 特征体系与扩充语法

在自然语言中,词语和短语之间通常存在一致性方面的限制。例如,名词短语“a men”不正确,因为冠词“a”表示单个对象,而名词“men”表示复数形式的对象,这不符合英语中数的一致性约束。除此以外,英语中还存在很多其他形式的一致性约束,其中包括主谓一致性、代词词性的一致性、短语中心语与补语的一致性,等等。为了方便地对其加以处理,我们需要进一步扩展语法的形式化表示,使得语法成分都可以具备对应的特征。例如,可以定义特征NUMBER,NUMBER取值为s(单数)或者p(复数),因此可以编写出扩充CFG规则,例如:

$NP \rightarrow ART\ N$ 只有当 $NUMBER_1$ 和 $NUMBER_2$ 一致时,规则才成立

这条规则说的是一个冠词和紧跟其后的名词可以组成一个合法的名词短语,但只有当第一个词语的数特征和第二个词一致时,该规则方可成立。实际上,这条规则等同于两条CFG规则,这两条规则需要采用两种不同的终结符号来表示所有名词短语的单数形式和复数形式,如下所示:

$NP-SING \rightarrow ART-SING\ N-SING$

$NP-PLURAL \rightarrow ART-PLURAL\ N-PLURAL$

从易用性方面来说,该例中用到的这两种方法看上去好像类似,但实际上并非如此。现在,让我们看看语法中所有右部都用NP规则的情况。它们都必须复制成两条,其中一条针对NP-SING,另一条针对NP-PLURAL。因此,这部分语法规则的规模就增加了一倍。而且,如果还要处理一些其他的特征,比如人称一致性,会使得语法规模不断倍增。但是,如果采用特征描述,则扩充语法的规模与原语法一样,同时又考虑了一致性约束。

为了达到这个目的,我们将语法成分定义为一个特征结构——将成分的相关属性定义为从特征到具体值的映射。在本书的例子中,公式中的特征名采用黑体字。例如,成分 ART1 代表 a 的一个特定用法,其特征结构可以写为:

ART1: (**CAT** ART
ROOT a
NUMBER s)

这就是说,它是一个属于词类 ART 的成分,词根为“a”,表示的是单数。通常,我们会使用缩写形式,使 CAT 的值更加突出,而且能和简单的上下文无关文法建立一个直观的联系。成分 ART1 用缩写形式,可写为:

ART1: (**ART** **ROOT** a **NUMBER** s)

特征结构可以用来表示更大的成分。为了做到这一点,特征结构本身也可以作为特征值出现。特殊的特征用 1,2,3 等整数来表示,它们分别代表第一个子成分,第二个子成分,可以根据需要继续类推。这样,NP 成分“a fish”就可以表示为:

NP1: (**NP** **NUMBER** s
1 (**ART** **ROOT** a
NUMBER s)
2 (**N** **ROOT** fish
NUMBER s))

需要注意的是,这也可以看成是一种句法分析树的表示方法,如图 4.1 所示。其中,子成分特征 1 和子成分特征 2 分别对应句法分析树中连接的两个子成分。

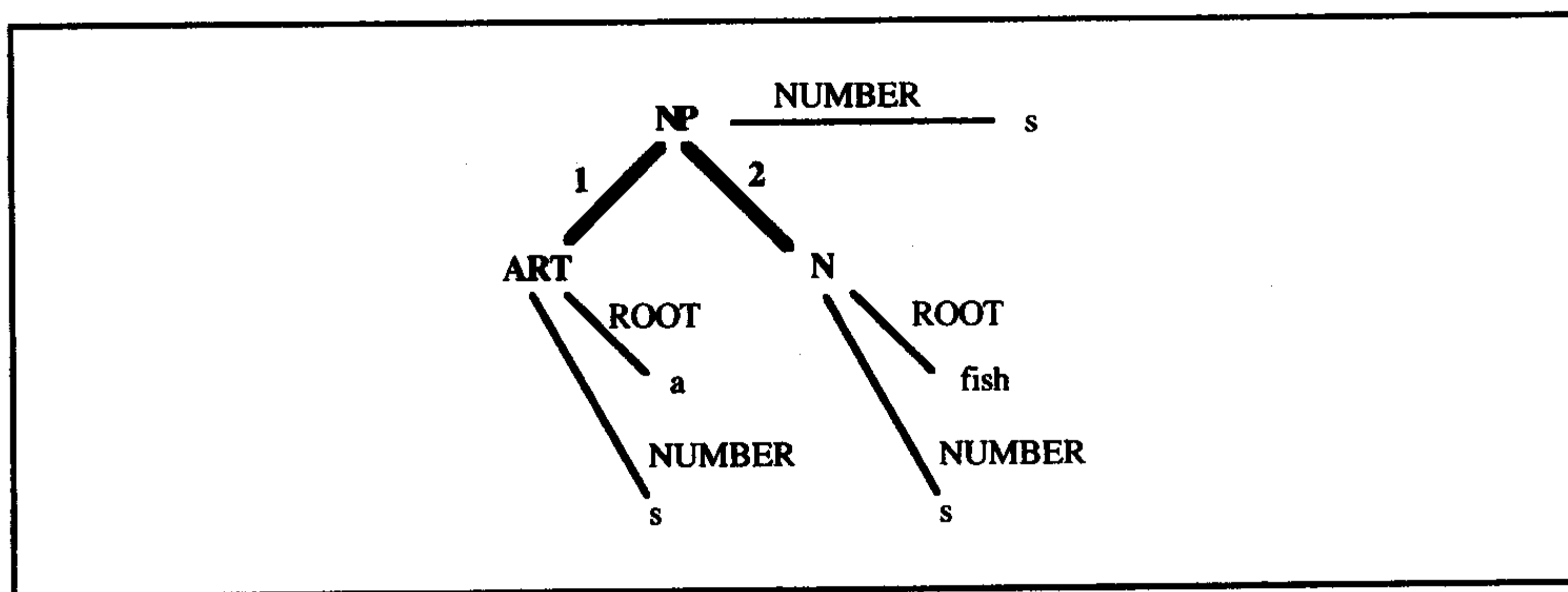


图 4.1 将特征结构看做扩展的句法分析树

扩充语法规则采用特征结构描述,而不采用简单的词类。变量也可以作为特征值,这样规则的适用范围就会更广泛。例如,简单名词短语的一条规则可以表示如下:

(NP **NUMBER** ?n) → (**ART** **NUMBER** ?n) (**N** **NUMBER** ?n)

这就是说,NP 成分可以包含两个子成分。其中,第一个子成分是 ART,第二个是 N,而这三个成分的 **NUMBER**(数)特征必须相同。根据这条规则,前面所给的 NP1 成分是合法的。另一方面,下面这个成分:

*(NP **1** (**ART** **NUMBER** s)
2 (**N** **NUMBER** s))

就不符合这条规则,因为 NP 中没有 NUMBER 特征。另外,下面的这个成分:

***(NP NUMBER s**
1 (ART NUMBER s)
2 (N NUMBER p))

也是不可接受的,因为 N 成分的 NUMBER 特征和其他两个 NUMBER 特征不同。

变量对成分中歧义表示也非常有用。比如,词语“fish”到底解读为单数还是复数,我们阅读的时候并不清楚。因此,词典通常会收录两个词条,它们的不同之处仅仅是 NUMBER 特征的值。换个方法,我们可以用变量来表示 NUMBER 的特征值,这样,只需要定义一个词条,即:

(N ROOT fish NUMBER ?n)

这种方法有效的原因是任何一个 NUMBER 的特征值都可以用于“fish”这个词。然而,在很多情况下,不是任何一个值都可行,而往往是一个可能的取值范围。为了处理这些问题,我们引入约束变量(constrained variable)。约束变量只能从一个指定的列表中取值,例如,变量?*n*{*s p*}指的是只能取值为 *s* 或者 *p* 的变量。写这些变量的时候,我们往往会把变量名全部去掉,仅仅列出所有可能的取值。基于这一点,词语“fish”可以表示为如下的成分:

(N ROOT fish NUMBER ?n{s p})

或者更简单地表示为:

(N ROOT fish NUMBER {s p})

框 4.1 特征结构的形式化

特征结构的形式化属性是研究的一个热点,这些研究工作将特征体系看做形式逻辑。特征结构可以定义为特征值的偏函数(partial function)。以下面这个特征结构为例:

ART1:(CAT ART
ROOT a
NUMBER s)

该特征结构可以看成是下面一阶谓词演算逻辑命题的缩写形式:

$$ART1(CAT) = ART \wedge ART1(ROOT) = a \wedge ART1(NUMBER) = s$$

含析取值的特征结构可以相应地映射为析取式,比如下面的这个结构:

THE 1:(CAT ART
ROOT the
NUMBER {s p})

该结构可以表示为:

$$THE1(CAT) = ART \wedge THE1(ROOT) = the \\ \wedge (THE1(NUMBER) = s \vee THE1(NUMBER) = p)$$

给定该结构之后,特征值的一致性就可以定义为等式方程。

接下来,我们会面临一个很有趣的问题:扩充上下文无关文法能否表达出简单上下文无关文法无法表达的语言?问题的答案取决于我们究竟采取什么样的限制来约束特征值。如果特征值集合是个有限集,那么每个特征的组合总有可能创建新的成分类。因此,它在表达能力上与上下文无关文法等同。但是,如果特征值集合不存在任何限制,那么,这种语法具有超强的计算能力。但实际上,即使不对特征值集合加以明显的限制,这种能力也是没有用处的,而标准的句法分析算法则可以用来分析带特征的语法。

4.2 英语的一些基本特征体系

本节叙述英语语法中一些常用的基本特征体系,并给出一些在本书中广泛应用的特征集合。具体地说,我们主要考虑数和人称的一致性、动词形式特征以及处理次范畴限制所需要的特征。应当阅读这些内容,并从总体上熟悉这些特征。这样,你日后需要知道某一特征的细节时,就可以回过头来查找相关的参考资料。

4.2.1 人称和数的特征

在上一节中,你已经接触到了英语的数体系,即每个词语描述的对象都可以分为单数或者复数。尽管英语中的几个不同地方均存在数的一致性限制,但最重要的还是体现在主语和动词的一致性上。并且,主语和动词还必须满足另外一方面的一致性要求,即人称的一致性。可能的人称有:

第一人称(1): 名词短语,指的是说话者,或者是包括说话者在内的一组人(例如, I, we, you and I)。

第二人称(2): 名词短语,指的是听众,或者包括听众在内而说话者除外的一组人(例如, you, all of you)。

第三人称(3): 名词短语,指的是说话者和听众之外的一个或多个对象。

数的特征和人称的特征总是同时出现。为方便起见,我们干脆将二者合并为一个特征 AGR。AGR 有六种可能的取值,包括第一人称单数(1s),第二人称单数(2s),第三人称单数(3s),以及第一、第二、第三人称复数(分别表示为 1p, 2p 和 3p)。例如,词语“is”只能和第三人称单数主语保持一致,因此,其 AGR 特征为 3s。不过,另外一个词“are”既可以匹配第二人称单数,也可以匹配任何一种人称的复数形式,所以它的 AGR 特征是一个取值范围为 {2s 1p 2p 3p} 的变量。

4.2.2 动词形式特征和动词次范畴

在英语中,另外一种非常重要的特征体系牵涉到动词的形式。这种特征在很多情况下使用,比如助词的分析,并且通常用在许多中心词的次范畴限制方面。正如第 2 章所述,动词有五种基本形式。为了更方便地表达某些语言现象,动词形式的特征体系会稍微复杂一些。具体而言,我们会使用特征 VFORM 以下的特征值:

base——原型(例如, go, be, say, decide)

pres——现在时态(例如, go, goes, am, is, say, says, decide)

- past——一般过去时态(例如, went, was, said, decided)
- fin——限定形式(一种时态形式, 等同于 {pres past})
- ing——现在分词(例如, going, being, saying, deciding)
- pastprt——过去分词(例如, gone, been, said, decided)
- inf——专指带 to 不定式的特征

为了处理词语与其补足语 (complement) 之间的相互作用, 我们另外引入特征 SUBCAT。第 2 章给出了一些常见的动词次范畴, 每个动词次范畴对应一个不同的 SUBCAT 特征值。图 4.2 中列出了一些补语的 SUBCAT 值, 这些补语是名词短语和动词短语的组成部分。为了帮助你记住特征值的含义, 我们定义的特征值均由补语各部分的主要语法类组成。如果某个语法类受特征值约束, 就在该成分与其后的特征值之间采用冒号隔开。因此, `_np_vp:inf` 值表示的是一个由名词短语后接动词短语构成的补足语。其中, 动词短语的 VFORM 为不定式 `inf`。当然, 这种命名方式只是为了便于读者理解, 它们的意义仅仅由包含这些特征的语法规则决定。因此, 你可以给这些值任意取名。例如, SUBCAT 值为 `_np_vp:inf` 的动词规则如下:

(VP) → (V SUBCAT `_np_vp:inf`)
 (NP)
 (VP VFORM `inf`)

这就是说, VP 可以包含 SUBCAT 值为 `_np_vp:inf` 的动词 V, 它后面跟着的是名词短语 NP, 而这个名词短语后面跟着的是 VFORM 值为 `inf` 的动词短语。显然, 我们可以用任何其他的惟一符号替换 `_np_vp:inf`。只要将词典也换成新的值, 就可以改写这条规则。

值	动词示例	示例
<code>_none</code>	laugh	Jack laughed.
<code>_np</code>	find	Jack found a key.
<code>_np_np</code>	give	Jack gave Sue the paper.
<code>_vp:inf</code>	want	Jack wants to fly.
<code>_np_vp:inf</code>	tell	Jack told the man to go.
<code>_vp:ing</code>	keep	Jack keeps hoping for the best.
<code>_np_vp:ing</code>	catch	Jack caught Sam looking at his desk.
<code>_np_vp:base</code>	watch	Jack watched Sam look at his desk.

图 4.2 NP/VP 组合的 SUBCAT 值

许多动词都用介词短语作为其补语结构, 在这些补语中, 往往会有某个专用的介词或者某个扮演特定角色的词语。例如, 动词“give”的补语可以是名词短语再加上一个由介词 `to` 构成的介词短语。例如, “Jack gave the money to the bank”(Jack 把钱给了银行)。其他动词, 比如“put”, 需要接表示方位的介词短语充当补足语, 使用的介词有“in”, “inside”, “on”和“by”。为了在特征体系里表示这些因素, 我们引入特征 PFORM 表示介词短语。例如, PFORM 值为 `TO` 的介词短语, 它必须以介词“to”作为中心词, 依次类推。PFORM 值为 `LOC` 的介词短语必须描述方位信息。另外一个很有用的 PFORM 值是 `MOT`, 一般用在“walk”之类的动词中。这些动词往

往接表示运动路径和方向的介词短语,如“*We walked to the store*”(我们走向商店)。可以产生这类短语的介词包括“to”,“from”和“along”。看起来好像很难区分 LOC 和 MOT 值,因为某些介词既可以用来表示方位也可以用来表示路径方向,不过,它们的意义完全不同。例如,“*Jack put the box{in on by} the corner*”(Jack 把盒子放到了角落里{或者角落旁边})是正确的,而“** Jack put the box{to from along} the corner*”(* Jack 把盒子放向{或者放离、沿着}角落)却是错误的表达方式,图 4.3 总结了 PFORM 特征。

值	介词示例	示例
TO	to	I gave it to the bank.
LOC	in, on, by, inside, on top of	I put it on the desk.
MOT	to, from, along, ...	I walked to the store.

图 4.3 介词短语 PFORM 特征的一些值

这种特征可以用来约束各种动词的补语形式。依据前面讨论过的命名习惯,诸如“put”这样的动词,其 SUBCAT 值可以表示为 `_np_pp:loc`,语法中相应的规则为:

(VP) → (V SUBCAT `_np_pp:loc`)
 (NP)
 (PP PFORM LOC)

对于内嵌句来说,我们常常需要引导分句的补充成分。同时,也必须对该成分划分次范畴。因此,特征 COMP 会有帮助。COMP 可能的取值有“for”,“that”或者“no-comp”(无连接词)。例如,动词“tell”的后面可以接由引导词“that”连接的从句 S,据此,我们可以对其进行二次分类, `_s:that`是“tell”的一个 SUBCAT 值。类似地,动词“wish”的次范畴后面可以接由补语成分引导词“for”连接的从句 S,因此, `_s:for`是“wish”的一个 SUBCAT 值。图 4.4 列举了这些新增的 SUBCAT 值以及不同动词的示例。在这一节中,所有 SUBCAT 特征的例子都和动词有关。实际上,名词、介词和形容词也可以使用 SUBCAT 特征,也可以采用同样的方法根据其补语成分划分次范畴。

值	动词示例	示例
<code>_np_pp:to</code>	give	Jack gave the key to the man.
<code>_pp:loc</code>	be	Jack is at the store.
<code>_np_pp:loc</code>	put	Jack put the box in the corner.
<code>_pp:mot</code>	go	Jack went ot the store.
<code>_np_pp:mot</code>	take	Jack took the hat to the party.
<code>_adjp</code>	be, seem	Jack is happy.
<code>_np_adjp</code>	keep	Jack kept the dinner hot.
<code>_s:that</code>	believe	Jack believed that the world was flat.
<code>_s:for</code>	hope	Jack hoped for the man to win the prize.

图 4.4 新增的 SUBCAT 特征值

4.2.3 二元特征

一个成分要么具有某种特征,要么不具有该特征,因此,特征是二元的。在我们的形式化定义中,二元特征是只能取值为“+”或“-”的特征。例如,INV 是一个二元特征,表示句子 S 的结构是否倒置主语(类似于一般疑问句)。句子“Jack laughed”的 S 结构的 INV 值为“-”;相反,句子“Did Jack laugh?”的 S 结构的 INV 值为“+”。通常,我们将这个值作为前缀使用。因此,可以说句子结构的特征为“+ INV”或者“- INV”。在进一步论述语法的过程中,我们会根据需要逐一介绍其他的二元特征。

4.2.4 特征的默认值

在很多情况下,允许特征设定默认值是一件有益的事情。我们构建带有特征的语法成分时,要是还没有给该特征赋值,它就会取默认值“-”。这对二元特征特别有用,对非二元特征同样有帮助。一般来讲,以后任何时候对该特征进行一致性检查时,默认值都会确保检查返回失败。第一次构建语法成分的时候,就会为各个特征赋上默认值。

4.3 词语形态分析和词典

在定义语法之前,必须先定义好词典。这一节,我们主要讨论词典设计的一些问题,并研究词语形态分析所需要的组成模块。

词典必须包含所有会用到的不同词语的信息,同时也包括所有的相关特征值约束。当词语存在歧义的时候,词典中必须采用多个词条来描述这些不同的信息,要求每个词条对应一种不同的用法。

词语通常都遵循一些有规律的词语形态模式,因此,没有必要将词语的各种形式都收录到词典中。比如,绝大多数英语动词都采用同样的后缀集合以表示不同的词语形式:加上“s”用于第三人称单数现在时,“ed”用于过去时,“ing”用于现在进行时,等等。如果没有词语形态分析,词典中就必须包含词的每一种变换形式。例如,动词“want”就需要有六个词条,分别是 want(基本形式和现在时),wants, wanting, wanted(过去时和过去分词形式)。

与此相反的是,如果我们采用 3.7 节给出的方法将后缀剥离出来,那么,词典就只须保留一个“want”词条。其主要思想是在词典中存储动词的基本形式,然后使用上下文无关规则将动词与后缀相结合,从而派生出其他的词条。下面是产生动词现在时的规则:

(V ROOT ?r SUBCAT ?s VFORM pres AGR 3s) →
(V ROOT ?r SUBCAT ?s VFORM base) (+S)

在这里, +S 是一种只包含后缀词素“s”的新词类。给定输入串 want -s, 该规则结合下面的词条:

want: (V ROOT want
SUBCAT {_np _vp:inf _np_vp:inf}
VFORM base)

会生成下面的语法成分:

want: (V ROOT want
 SUBCAT {_np _vp:inf _np_vp:inf}
 VFORM pres
 AGR 3s)

另外一条规则可以生成非第三人称单数情况下的现在时成分,对于大部分动词来说,该成分与词根形式完全一致:

(V ROOT ?r SUBCAT ?s VFORM pres AGR {1s 2s 1p 2p 3p}) →
 (V ROOT ?r SUBCAT ?s VFORM base)

为了避免产生错误的解析结果,我们还需要对这条规则做进一步的修改。现在的这条规则可以把任意的动词基本形式直接转化为现在时,但这对不规则动词来说显然是错误的。例如,基本形式 be 就不能用做现在时(如,“* We be at the store”, * 我们在商店)。为了涵盖这种情况,需要引入特征来区分不规则形式。具体地说,二元特征为 + IRREG-PRES 的动词存在不规则的现在时形式。因此,上述规则可以正确地表述如下:

(V ROOT ?r SUBCAT ?s VFORM pres AGR {1s 2s 1p 2p 3p}) →
 (V ROOT ?r SUBCAT ?s VFORM base IRREG-PRES -)

我们采取了默认机制,所以 IRREG-PRES 特征只需要对不规则动词进行说明,规则动词的默认值设为“-”。我们还需要采取类似的二元特征来标识不规则的过去时态(IRREG-PAST, 比如动词“saw”),还需要用这类特征将“en”的过去分词形式(EN-PASTPRT)和“ed”区分开来。这些特征对标准词法规则的应用进行了约束,而我们一般都会将不规则形式直接添加到词典中。语法 4.5 给出了一系列的规则,并通过使用这些特征派生不同的动词和名词形式。

现在时

1. (V ROOT ?r SUBCAT ?s VFORM pres AGR 3s) →
 (V ROOT ?r SUBCAT ?s VFORM base IRREG-PRES -) +S
2. (V ROOT ?r SUBCAT ?s VFORM pres AGR {1s 2s 1p 2p 3p}) →
 (V ROOT ?r SUBCAT ?s VFORM base IRREG-PRES -)

过去时

3. (V ROOT ?r SUBCAT ?s VFORM past AGR {1s 2s 3s 1p 2p 3p}) →
 (V ROOT ?r SUBCAT ?s VFORM base IRREG-PAST -) +ED

过去分词

4. (V ROOT ?r SUBCAT ?s VFORM pastprt) →
 (V ROOT ?r SUBCAT ?s VFORM base EN-PASTPRT -) +ED
5. (V ROOT ?r SUBCAT ?s VFORM pastprt) →
 (V ROOT ?r SUBCAT ?s VFORM base EN-PASTPRT +) +EN

现在分词

6. (V ROOT ?r SUBCAT ?s VFORM ing) →
 (V ROOT ?r SUBCAT ?s VFORM base) +ING

复数名词

7. (N ROOT ?r AGR 3p) →
 (N ROOT ?r AGR 3s IRREG-PL -) +S

语法 4.5 动词和名词常用后缀的一些词法规则

对于一个给定的大规模特征集合,编写词条的任务看上去异常困难。大部分系统都会提供一些机制来缓解这些问题。第一种技术就是允许特征被赋予默认值,这一点我们已经在前面提到过了。如果词条的某个特征取默认值时,默认机制允许该特征不出现在词条中。另一个常用技术是允许在编写词典时定义多个特征集,在随后使用的时候,只需要用单个符号表示这个特征集,而不用将所有的特征一一罗列。以后,我们还会讨论其他一些技术方法,从而保证可以在特征层次结构中继承各种特征。

图4.6包含了一部小词典,它包含的很多词语在下面的例子中都会使用到。其中,包含了三个和“saw”有关的词条,“saw”可以当做名词和规则动词使用,同时它还是动词“see”的不规则过去时态。下面的例句分别对此进行解释。

a:	(CAT ART ROOT A1 AGR 3s)	saw:	(CAT N ROOT SAW1 AGR 3s)
be:	(CAT V ROOT BE1 VFORM base IRREG-PRES + IRREG-PAST + SUBCAT {_adjp _np}))	saw:	(CAT V ROOT SAW2 VFORM base SUBCAT _np)
cry:	(CAT V ROOT CRY1 VFORM base SUBCAT _none)	saw:	(CAT V ROOT SEE1 VFORM past SUBCAT _np)
dog:	(CAT N ROOT DOG1 AGR 3s)	see:	(CAT V ROOT SEE1 VFORM base SUBCAT _np IRREG-PAST + EN-PASTPRT +)
fish:	(CAT N ROOT FISH1 AGR {3s 3p} IRREG-PL +)	seed:	(CAT N ROOT SEED1 AGR 3s)
happy:	(CAT ADJ SUBCAT _vp:inf)	the:	(CAT ART ROOT THE1 AGR {3s 3p}))
he:	(CAT PRO ROOT HE1 AGR 3s)	to:	(CAT TO)
is:	(CAT V ROOT BE1 VFORM pres SUBCAT {_adjp _np} AGR 3s)	want:	(CAT V ROOT WANT1 VFORM base SUBCAT {_np _vp:inf _np _vp:inf}))
Jack:	(CAT NAME AGR 3s)	was:	(CAT V ROOT BE1 VFORM past AGR {1s 3s} SUBCAT {_adjp _np}))
man:	(CAT N1 ROOT MAN1 AGR 3s)	were:	(CAT V ROOT BE VFORM past AGR {2s 1p 2p 3p} SUBCAT {_adjp _np}))
men:	(CAT N ROOT MAN1 AGR 3p)		

图4.6 一部词典

The saw was broken. (这把锯坏了。)

Jack wanted me to saw the board in half. (Jack 想让我把这块板子锯成两半。)

I saw Jack eat the pizza. (我看见 Jack 吃披萨。)

在 3.7 节中,我们给出了一个去除后缀,并对各种拼写进行规范化处理的算法。在语法 4.5 上采用任何一种句法分析算法,最后都可以生成派生词条。采用图 4.6 中的词典,根据语法 4.5,我们可以派生出下列词语正确的语法成分:been, being, cries, cried, crying, dogs, saws (有两种解释),sawed, sawing, seen, seeing, seeds, wants, wanting 和 wanted。例如,词语“cries”可以转换为序列“cry + s”,根据规则 1 可以在词典中现在形式的基础上,最后生成现在时态的词条。

如果采用不同词条,依据不同的词法规则可以生成同一个词语,而该词语通常会有多种含义。例如,词语“saws”可以转换成序列“saw + s”,这时,它是一个复数名词(根据规则 7 和“saw”的第一个词条);“saws”也可能是动词“saw”第三人称单数条件下的现在时态(根据规则 1 和“saw”的第二个词条)。需要注意的是,“saw”的 VFORM 并不是基本形式,因此,规则 1 不能作用于它的第三个词条。

这种方法的成功之处在于它能够避免错误的派生过程。例如,把“seed”分析成动词“see”的过去时,就是一种错误的派生。如果后缀去除的有限状态转录机(FST)设计正确,这种错误分析是绝对不会出现的。具体来说,有限状态转录机不会允许动词“see”转移到能接受后缀“ed”的状态中去。即使由于某种原因,出现了这种情况,规则 3 同样不能使用,因为“see”词条的 IRREG-PAST 值为 +。

4.4 采用特征的简单语法

在这一节中,采用特征系统,以及前面章节中介绍过的词典,我们将提出一种简单语法。该语法能够处理下面这些句子:

The man cries. (这个男人在哭。)

The men cry. (这些男人在哭。)

The man saw the dogs. (这个男人看见了那些狗。)

He wants the dog. (他想要这条狗。)

He wants to be happy. (他想要开开心。)

He wants the man to see the dog. (他想让这个男人去看那条狗。)

He is happy to be a dog. (他很乐意做一条狗。)

这种语法体系还可以发现下面的句子是不可接受的:

* The men cries.

* The man cry.

* The man saw to be happy.

* He wants.

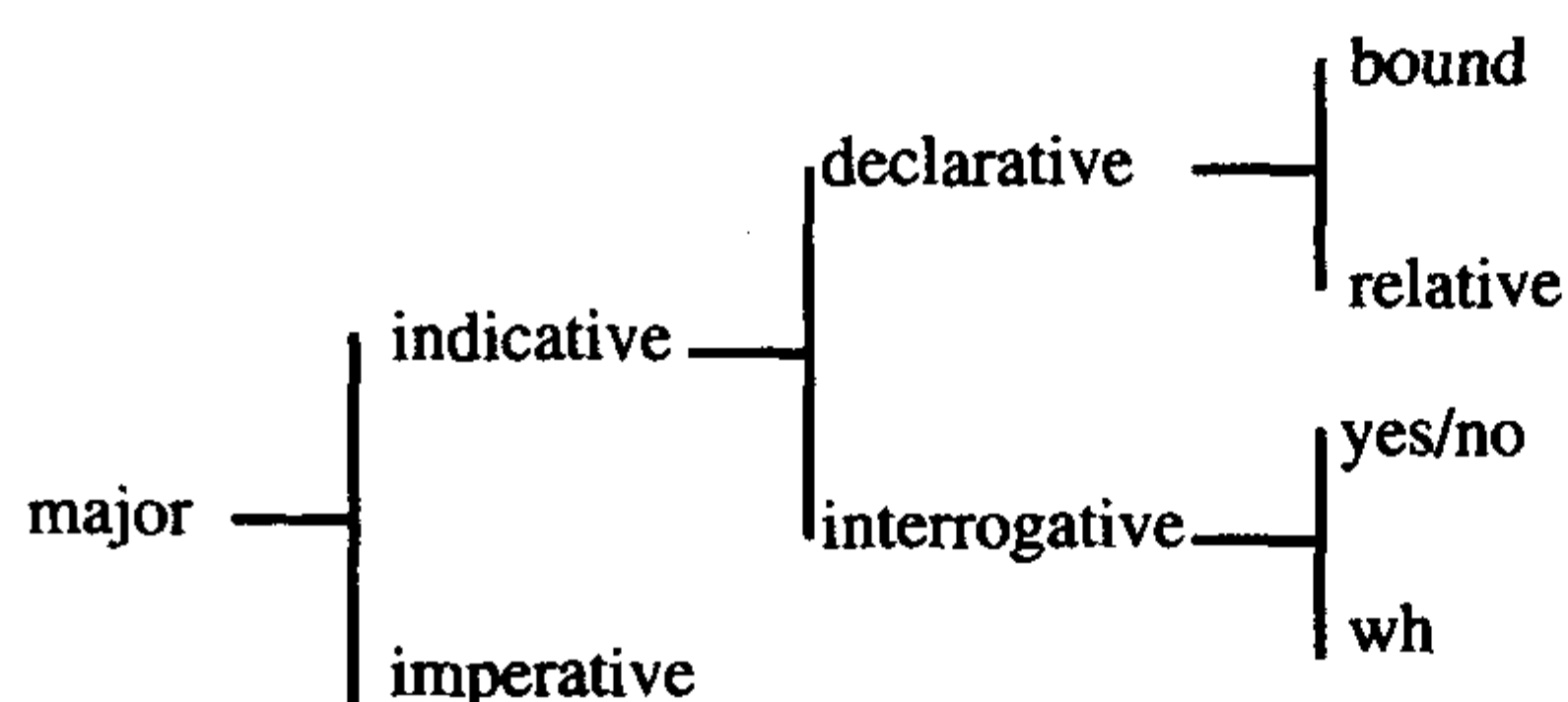
* He wants the man saw the dog.

进一步探讨语法之前,我们需要另外介绍一些习惯用法,这些习惯用法在整本书中都将很有用。编写语法规则的时候,如果考虑所有必需的特征,那将是一件非常繁琐的工作。然而,

在特征的使用过程中,实际存在一些很明显的规律,我们可以好好地利用这些规律,将规则编写的过程加以简化。例如,许多特征值只能用于描述某一个特定的特征(例如,值 *inf* 只能用于 *VFORM* 特征, *_np_vp:inf* 只能用于 *SUBCAT* 特征)。基于这一点,在不引起歧义的前提下,就可以省略特征的名称,将该特征值列在方括号中。所以, *(VP SUBCAT inf)* 可以缩写为 *VP[inf]*。因为二元特征不存在独有的特征值,所以我们为之引入一个特殊的规定:对于二元特征 *B*,语法成分 *C[+B]* 表示的是成分 *(C B+)*。

框 4.2 系统语法

在基于特征的计算机系统发展过程中,系统语法 (*systemic grammar*, Halliday, 1985) 是一个有重要影响的理论。这个理论强调的是语言结构在交流过程中所扮演的功能角色。语法组织成为篇章功能的选择项的集合,这些功能决定句子的结构。这些选项最后组织成为层次结构,我们称之为系统。比如,语态系统表示所有影响句子语态的选项。其中的部分结构如下所示:



上面这幅结构图表明,一旦我们做出了某些特定的选择,则必然存在其他一些相关的选项。比如,如果判定一个句子是陈述句,那么“*bound*”(确定)和“*relative*”(关系)之间的取舍和陈述句相关。而“*yes/no*”(是/非)和“*wh*”(特殊疑问)对于陈述句来说是无关的选项。

Winograd(1973)使用了系统语法,同时 Winograd(1983)还对形式化问题做了非常精辟的讨论。最近这些年来,系统语法主要应用在自然语言生成系统当中。系统语法在生成一个句子的过程中,对如何组织各种选项提供了很好的形式化手段。[具体示例见 Mann 和 Mathiesson(1985),以及 Patten(1988)。]

很多特征是受限的,所以母成分的值必须与其中中心子成分的值一致,我们称这些特征为中心特征(head feature)。例如,在所有 *VP* 规则中,*VP* 的 *VFORM* 特征值和 *AGR* 特征值均要和中心动词相一致。比如下面的这条规则:

(VP VFORM ?v AGR ?a) →
(V VFORM ?v AGR ?a SUBCAT _np_vp:inf)
(NP)
(VP VFORM inf)

如果中心特征可以脱离规则单独加以声明,那么系统就能够根据需要将这些特征自动地加到规则中去。如果将 *VFORM* 和 *AGR* 声明为中心特征,则前面的这条 *VP* 规则就可以缩写为:

VP → (V SUBCAT _np_vp:inf) NP (VP VFORM inf)

规则中的中心成分用斜体表示。结合所有的缩写约定 (abbreviation convention), 这条规则还可

以进一步简化为:

$VP \rightarrow V[_{np_vp:inf}] NP VP[inf]$

使用这些规定,我们可以编写一个简单的语法,如语法 4.7 所示。规则 1 和规则 2 必须满足数的一致性,除此以外,根据已介绍过的规定,剩下的其他约束特征均可自动获取。在图的底部,我们给出了每个语法类的中心特征,语法 4.7 是语法 4.8 的一种缩写形式。现在,让我们来看看语法 4.8 中的规则 1 如何缩减为语法 4.7 中对应的规则 1。缩写以后的规则形式为:

$S[-inv] \rightarrow (NP AGR ?a) (VP[\{pres\ past\}] AGR ?a)$

1. $S[-inv] \rightarrow (NP AGR ?a) (VP[\{pres\ past\}] AGR ?a)$
2. $NP \rightarrow (ART AGR ?a) (N AGR ?a)$
3. $NP \rightarrow PRO$
4. $VP \rightarrow V[_{none}]$
5. $VP \rightarrow V[_{np}] NP$
6. $VP \rightarrow V[_{vp:inf}] VP[inf]$
7. $VP \rightarrow V[_{np_vp:inf}] NP VP[inf]$
8. $VP \rightarrow V[_{adjp}] ADJP$
9. $VP[inf] \rightarrow TO VP[base]$
10. $ADJP \rightarrow ADJ$
11. $ADJP \rightarrow ADJ[_{vp:inf}] VP[inf]$

S, VP 的中心特征: **VFORM, AGR**

NP 的中心特征: **AGR**

语法 4.7 缩写形式的简单语法

1. $(S INV - VFORM ?v\{pres\ past\} AGR ?a) \rightarrow$
 $(NP AGR ?a) (VP VFORM ?v\{pres\ past\} AGR ?a)$
2. $(NP AGR ?a) \rightarrow (ART AGR ?a) (N AGR ?a)$
3. $(NP AGR ?a) \rightarrow (PRO AGR ?a)$
4. $(VP AGR ?a VFORM ?v) \rightarrow (V SUBCAT _{none} AGR ?a VFORM ?v)$
5. $(VP AGR ?a VFORM ?v) \rightarrow (V SUBCAT _{np} AGR ?a VFORM ?v) NP$
6. $(VP AGR ?a VFORM ?v) \rightarrow$
 $(V SUBCAT _{vp:inf} AGR ?a VFORM ?v) (VP VFORM inf)$
7. $(VP AGR ?a VFORM ?v) \rightarrow$
 $(V SUBCAT _{np_vp:inf} AGR ?a VFORM ?v) NP (VP VFORM inf)$
8. $(VP AGR ?a VFORM ?v) \rightarrow$
 $(V SUBCAT _{adjp} AGR ?a VFORM ?v) ADJP$
9. $(VP SUBCAT inf AGR ?a VFORM inf) \rightarrow$
 $(TO AGR ?a VFORM inf) (VP VFORM base)$
10. $ADJP \rightarrow ADJ$
11. $ADJP \rightarrow ADJ(SUBCAT _{inf}) (VP VFORM inf)$

语法 4.8 包含所有特征的扩展语法

惟一值显然可以这样扩展:值[- inv]扩展为(**INV** -),值[{pres past}]扩展为(**VFORM** ?v {pres past})。句子 S 的中心特征为 AGR 和 VFORM,这些特征都必须添加到 S 和 VP 的中心语上。因此,可以得到规则:

(S **INV** - **VFORM** ?v{pres past} **AGR** ?a) →
 (NP **AGR** ?a)
 (VP **VFORM** ?v{pres past} **AGR** ?a)

如语法 4.8 所示。

缩写形式对于句法分析树的概括也非常有用。例如,图 4.9 给出了前面两个例句的句法分析树,这说明每个例句都可以被语法接受,都是合法的句子。

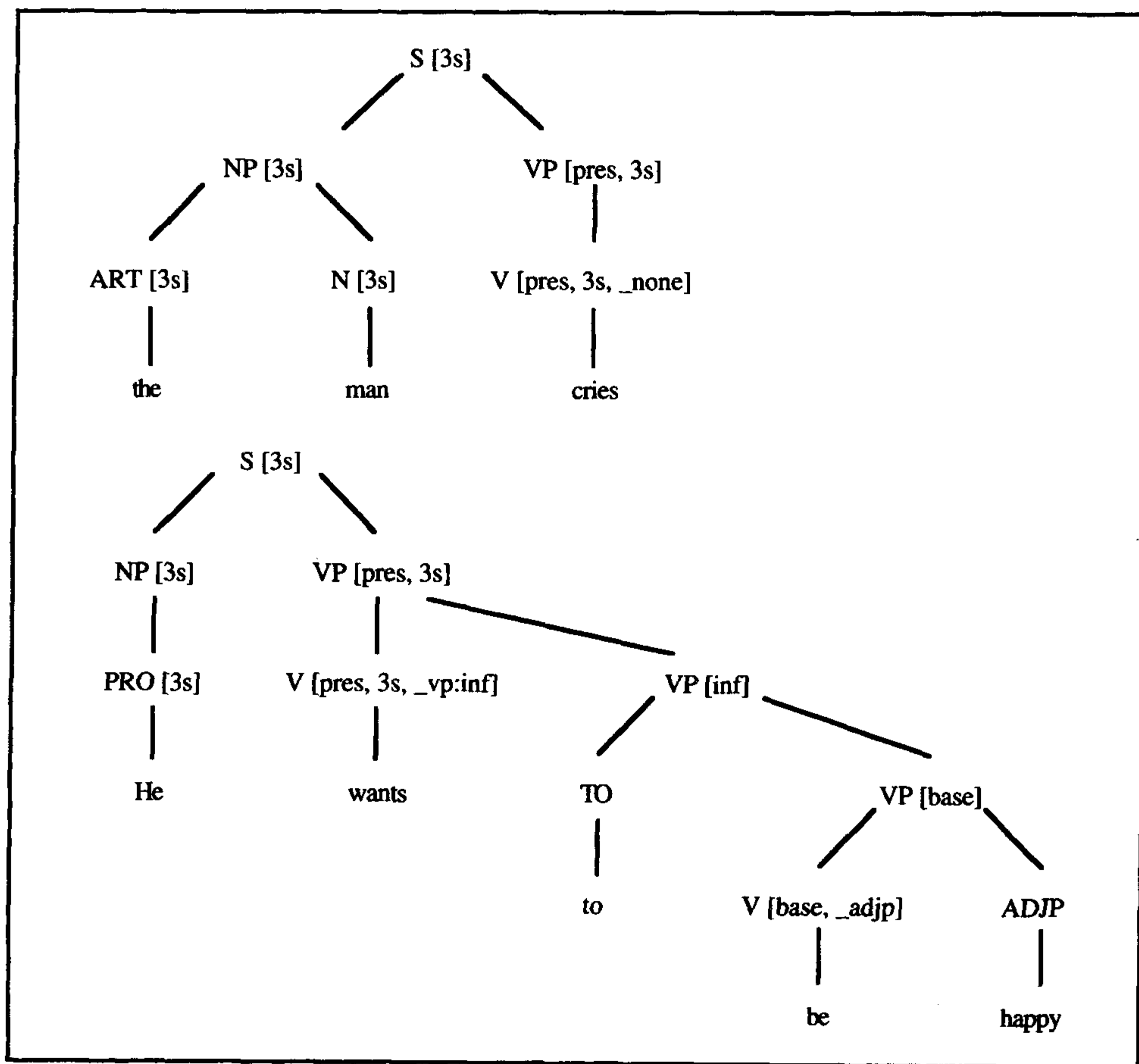


图 4.9 两棵带特征值的句法分析树

现在让我们看看,为什么本节开头给出的错误句子不能被语法 4.7 接受? 句子“* The men cries”和“* The man cry”不能被接受是因为它们不满足规则 1 关于数的一致性约束。名词短语(NP)成分“the men”的 AGR 值为 3p(第三人称复数),而动词短语(VP)“cries”的 AGR 值为 3s(第三人称单数),因此规则 1 不能适用。类似地,“the man”的 AGR 值为 3s,而动词短语“cry”的 AGR 的取值范围是{1s 2s 1p 2p 3p},所以“* the man cry”也不能被语法接受。短语“* the man saw to be happy”不能被接受,这是因为动词“saw”的 SUBCAT 值为_np,只能采用规则 5 来构

造动词短语。但是,规则 5 需要一个名词短语作为其补足语,而“to be happy”不可能是合法的名词短语。

短语“He wants”也是不可接受的。因为动词“wants”的 SUBCAT 值取值范围为{ $_np_vp:inf$, $_np_vp:inf$ }。此时,只有规则 5、规则 6 和规则 7 可以用于构造 VP,但是所有这些规则都需要某种类型的非空成分作为其补足语。同理,短语“* He wants the man saw the dog”也不能被语法接受,但是我们需要多做一点分析。同样,规则 5、规则 6 和规则 7 可能适用于动词“wants”。这时,语法需要匹配一个 NP 和 VP[inf]。因此,规则 5 和规则 6 不能使用,而规则 7 看起来比较接近。短语“the man”是符合我们要求的 NP,但是“saw the dog”匹配不定式 VP[inf]失败。具体来说,“saw the man”是合法的 VP,但是其 VFORM 特征是过去时,而不是不定式。

4.5 基于特征的句法分析

在第 3 章我们给出了分析上下文无关文法的句法分析算法。实际上,我们可以把这些算法扩展到扩充上下文无关文法,这涉及规则和语法成分匹配算法的泛化处理。例如,第 3 章中介绍的 chart 分析算法会根据新的语法成分来扩展活动边。语法成分 X 可以扩展如下形式的边:

$$C \rightarrow C_1 \cdots C_i \circ X \cdots C_n$$

扩展之后,产生的新边形式为:

$$C \rightarrow C_1 \cdots C_i X \circ \cdots C_n$$

类似的操作可以用于处理带特征的语法。不过,在语法成分 X 对原始边进行扩展之前,句法分析器可能需要对原始边中的变量进行实例化。如何精确定义这种匹配操作,其关键在于定义好带有特征的语法规则。例如,一条规则如下所示:

$$1. (NP \text{ AGR } ?a) \rightarrow \circ (ART \text{ AGR } ?a) (N \text{ AGR } ?a)$$

这条规则说的是 NP 可以由 ART 和 N 构成,前提是三者的 AGR 特征必须一致。但是,这里对 NP,ART 和 N 的其他特征并没有做任何限制。因此,一旦进行匹配的语法成分违反了这条语法规则,那么惟一有关的就是 AGR 特征,而语法成分中的其他特征均可以忽略。例如,我们考虑用如下语法成分对边 1 进行扩展:

$$2. (ART \text{ ROOT } A \text{ AGR } 3s)$$

为了使边 1 便于操作,需要将其中的变量“?a”实例化为 3s,因此,可以得到:

$$3. (NP \text{ AGR } 3s) \rightarrow \circ (ART \text{ AGR } 3s) (N \text{ AGR } 3s)$$

因为规则中每个特征都在成分 2 中,所以这条边现在就可以进行扩展:

$$4. (NP \text{ AGR } 3s) \rightarrow (ART \text{ AGR } 3s) \circ (N \text{ AGR } 3s)$$

现在,让我们考虑用词“dog”充当的语法成分扩展该边,可以得到:

$$5. (N \text{ ROOT } DOG1 \text{ AGR } 3s)$$

因为 AGR 特征完全一致,所以这步操作有效。这样,就可以完成这条边的扩展,即:

$$6. (NP \text{ AGR } 3s) \rightarrow (ART \text{ AGR } 3s) (N \text{ AGR } 3s) \circ$$

这就意味着句法分析器找到了一个形式为(NP AGR 3s)的语法成分。

给定一条边 A,其中点号之后的语法成分称为 NEXT,用来扩展这条边的新语法成分称为 X,这个算法可以更加准确地定义如下:

- a. 对变量进行实例化,保证 NEXT 中所有指定的特征都可以在 X 中找到。
- b. 复制边 A,创建一条新边 A',其中 A'将 A 中的变量进行了实例化。
- c. 在 chart 句法分析器中,和平常的处理一样对 A'进行更新。

例如,假定边 1 为 A,成分 2 的 ART 为 X,(ART AGR ?a)为 NEXT。在步骤 a 中,NEXT 和 X 不能匹配,你会发现变量“?a”必须实例化为 3s。在步骤 b 中,产生 A 的一个副本,如边 3 所示。然后,在步骤 c 中,对这条边进行更新,将产生新的边,如边 4 所示。

在算法运行的过程中,如果涉及到受限变量,比如变量“?a{3s 3p}”,此时匹配可以按照同样的方式正常进行。不过,变量的绑定值必须在取值范围之中。如果某个语法成分包含了变量,那么在它可能的取值之中,必须有一个满足规则的要求。如果规则和语法成分均包含了变量,那么最终结果的取值范围是两者取值范围的交集。例如,如果我们扩展边 1 的时候,使用的是语法成分(ART ROOT the AGR ?v{3s 3p}),即词“the”。在应用过程中,变量?a 必须实例化为?v{3s 3p}。因此,最终产生的规则为:

$$(NP \ AGR \ ?v\{3s \ 3p\}) \rightarrow (ART \ AGR \ ?v\{3s \ 3p\}) \circ (N \ AGR \ ?v\{3s \ 3p\})$$

(N ROOT dog AGR 3s) 可以对这条边进行扩展,因为?v{3s 3p}可以实例化为 3s。最终的结果和边 6 相同。而且,chart 中“the”的数据项不会因为这个操作而改变,它仍然保持原值?v{3s 3p},只有在这条边中它的 AGR 特征才限制为 3s。

另外一种扩展对于记录句法分析的结构非常有用。每次对边进行扩展时,句法分析器会自动添加子成分的特征(1,2,...,关键要看添加的是哪个子成分),这些子成分特征的值已经在 chart 中。采取这样的处理办法,并假定 chart 中已经包含了两个成分 ART1 和 N1,它们分别对应于词语“the”和词语“dog”,则加入到 chart 中关于短语“the dog”的语法成分为:

$$\begin{array}{ll} (NP \ AGR \ 3s \\ \quad 1 \quad ART1 \\ \quad 2 \quad N1) \end{array}$$

其中,ART1 = (ART ROOT the AGR {3s 3p}),而 N1 = (N ROOT dog AGR {3s})。注意,ART1 的 AGR 特征不变。因此,在可能的时候,它可以用于其他需要作为 3p 值的分析当中去。通过这些扩展边和语法成分创建的扩充手段,第 3 章中给出的任何一种 chart 分析算法均可以处理扩充的语法。现在,让我们来看一个例子,采用语法 4.8 对“He wants to cry”进行句法分析,产生的最终 chart 图见图 4.10。这一节剩下来的部分将具体考虑如何在 chart 图中构建一些非终结符。

用规则 3 生成语法成分 NP1。为了方便起见,这里再重复一下:

$$(NP \ AGR \ ?a) \rightarrow (PRO \ AGR \ ?a)$$

为了匹配成分 PRO1,变量?a 实例化为 3s。因此,产生的新成分为:

$$\begin{array}{l} NP1: \ (CAT \ NP \\ \quad \quad AGR \ 3s \\ \quad \quad 1 \ PRO1) \end{array}$$

接下来,考虑用规则 4 建立成分 VP1,即:

$(VP\ AGR\ ?a\ VFORM\ ?v) \rightarrow (V\ SUBCAT\ _none\ AGR\ ?a\ VFORM\ ?v)$

为了让规则的右边和成分 V2 进行匹配,变量?v 必须实例化为“base”(基本形式),而 V2 的 AGR 特征没有专门定义,所以取其默认值“-”。由此,得到的新成分为:

VP1: (CAT VP
AGR -
VFORM base
1 V2)

一般来说,chart 图中并不给出默认值。采取类似的方法,采用规则 9,TO1 和 VP1 可以生成 VP2;采用规则 6,V1 和 VP2 生成 VP3。最后,采用规则 1,由 NP1 和 VP3 最终可以生成 S1。

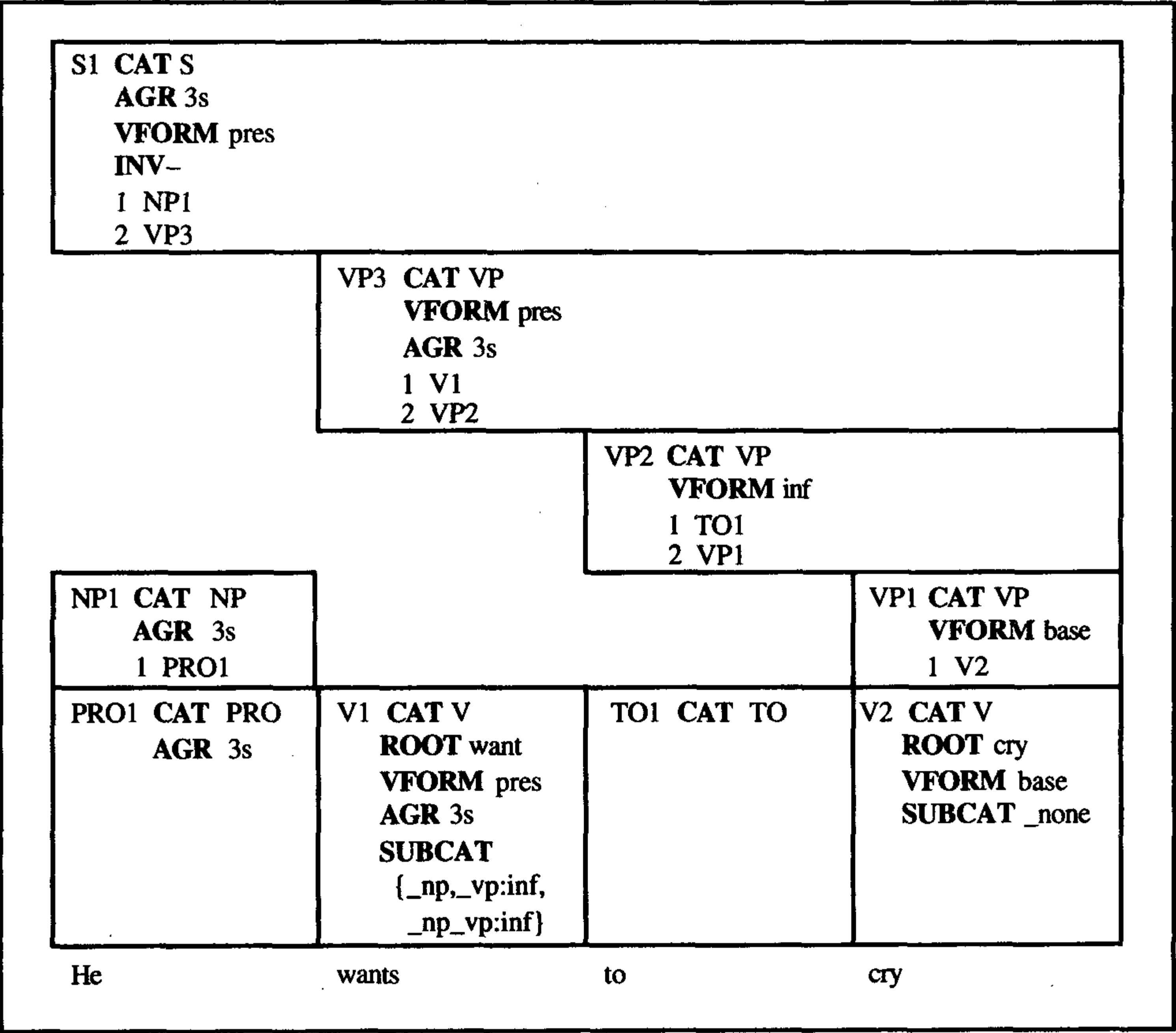


图 4.10 句子“He wants to cry”的 chart 图

◦ 4.6 扩充转移网络

我们也可以将特征加入到递归转移网络中生成扩充转移网络(ATN, augmented transition network)。在 ATN 中,我们通常称特征为寄存器。如果每个网络都允许有一组寄存器,那么,就可以创建语法成分结构。每次当对一个新网络进行压栈处理的时候,就会生成一组新的寄

寄存器。在遍历网络的时候,系统会根据每条边相关的操作对这些寄存器进行赋值。从当前网络中弹出时,可以结合这些寄存器,最终形成语法成分结构。其中,CAT 槽为该网络的名称。

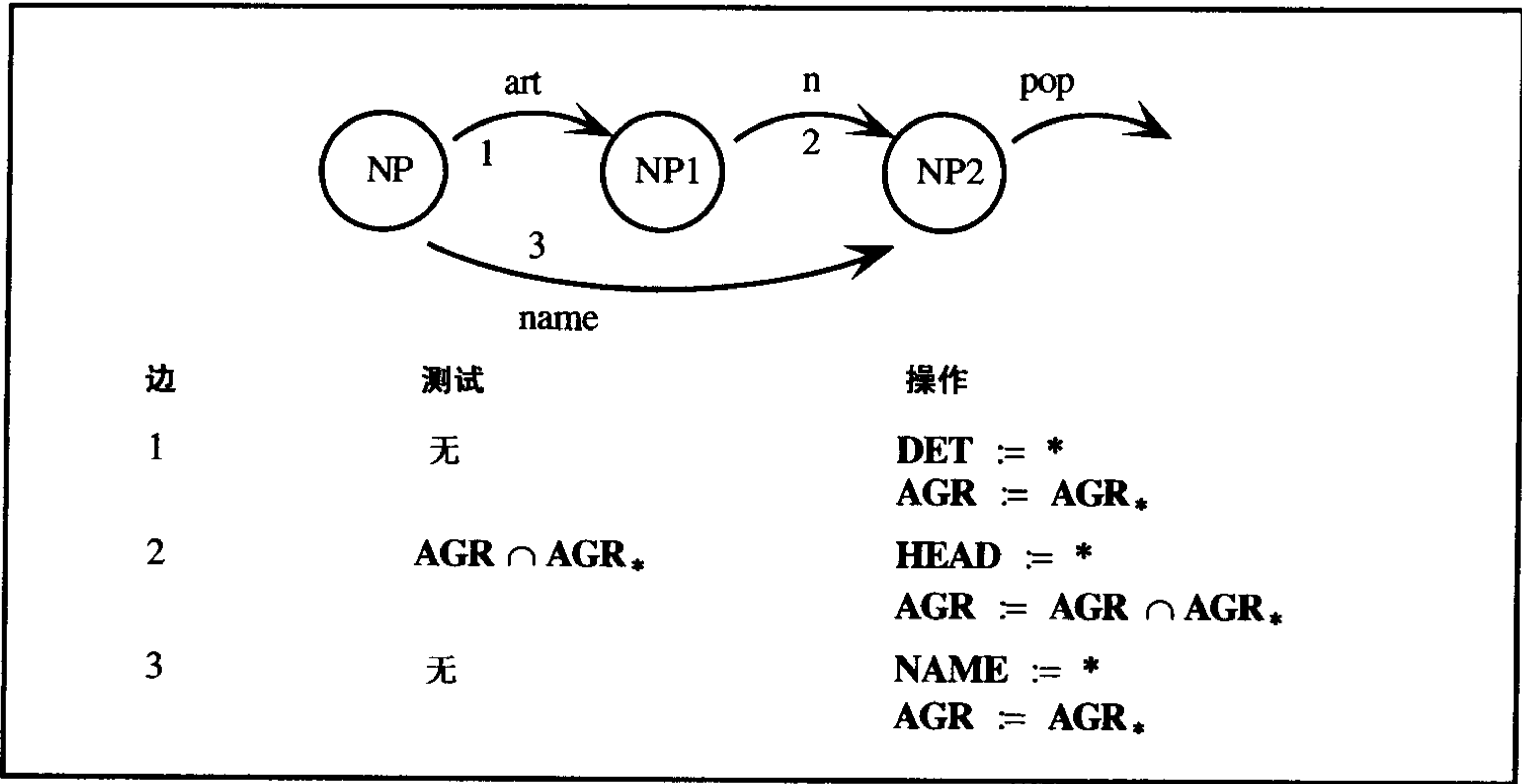
语法 4.11 是一个简单的 NP 网络,其中,相关的操作列在网络下面的表中。ATN 采用一种特殊的机制来抽取通过一条边之后的结果。如果通过的是一条词法边,比如边 1,就可以根据输入词语建立语法成分,同时赋予它特殊的变量名“*”。这个操作就是:

DET := *

然后,将这个语法成分指定为 DET 寄存器,这条边上的第二个操作为:

AGR := AGR*

即将网络中 AGR 寄存器的值赋给新词(* 中的语法成分)的 AGR 寄存器。



语法 4.11 简单的 NP 网络

一致性检查由测试集合指定,测试指的是布尔表达式。如果返回值非空,则表示成功;如果返回空集合或者返回零值,则表示失败。测试失败,则它对应的边就不可遍历。边 2 上的测试表明,只有当网络的 AGR 特征和新词的 AGR 寄存器存在非空交集时,才可以通过该边。

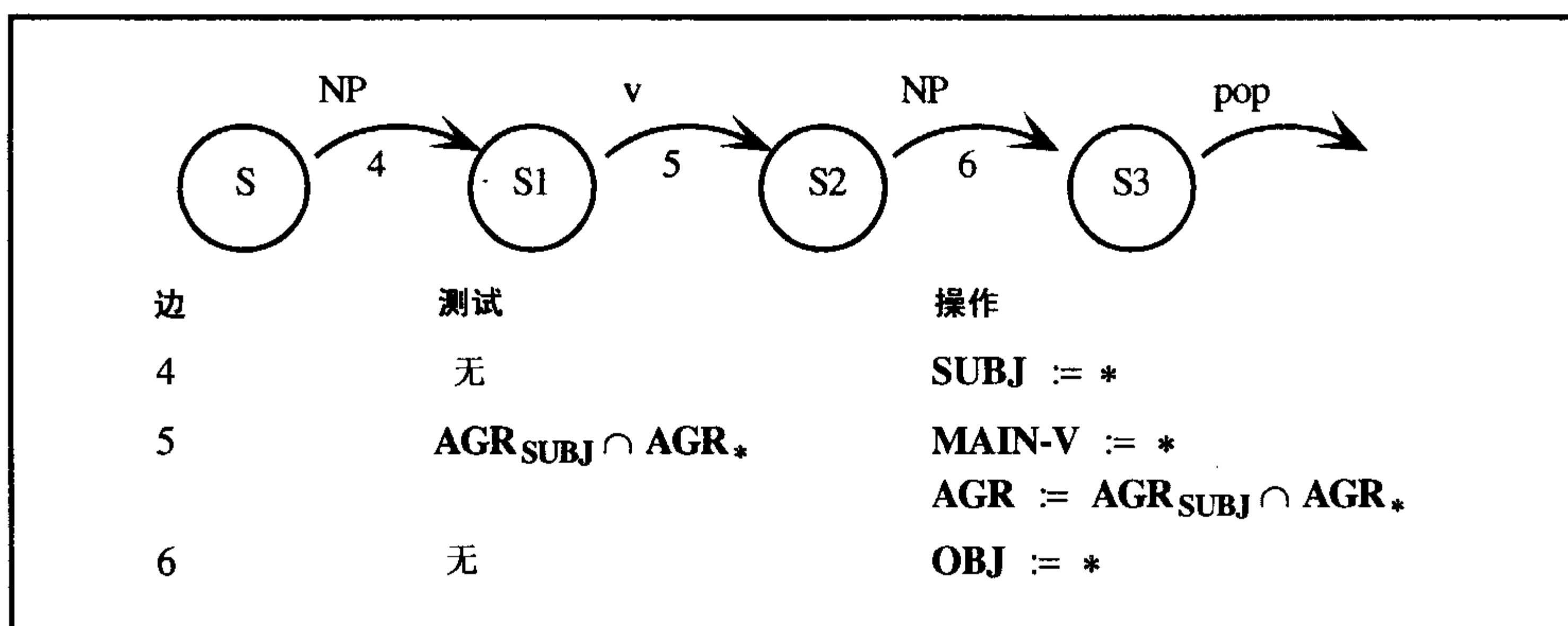
压入边上的特征按照类似的方法处理。通过遍历 NP 网络而生成的语法成分,返回的特征值为“*”。因此,在语法 4.12 中,从 S 到 S1 的边上操作是:

SUBJ := *

该操作会将 NP 网络返回的语法成分赋给寄存器 SUBJ。只有当 SUBJ 寄存器中语法成分的 AGR 寄存器和新语法成分(即该动词)的 AGR 寄存器存在非空交集时,边 2 上的测试才会成功。这种测试可以保证主谓语的一致性。

采用 4.3 节中的词典,ATN 可以接受下面这些句子:

- The dog cried.
- The dogs saw Jack.
- Jack saw the dogs.



语法 4.12 简单的 S 网络

现在让我们来看一个例子。图 4.13 给出了句子“The dog saw Jack”的句法分析流程。图中,我们给出了网络中的当前节点、当前词语的位置、从节点出去的边以及寄存器操作。寄存器操作的目标是成功的句法分析。句法分析从 S 网络开始启动,调用边 4 后立即移到 NP 网络中。读入词序列“The dog”后,NP 网络会检查数的一致性,然后建立一个名词短语,这个名词短语的 **AGR** 特征为复数。当通过弹出边时,也就意味着 S 网络中已经处理完了边 4。这个名词短语赋给 **SUBJ** 寄存器。接着,在通过边 3 时,需要对动词的一致性进行检查。最后,“Jack”会在另外一次调用 NP 网络之后被接受为名词短语。

4.6.1 简单陈述句的 ATN 语法

如何使用 ATN 来描述一些陈述句? 在这里,我们有个综合性更强的例子。句子结构允许以 NP 开头,名词短语后面跟着主动词,而主动词后面最多还可以跟两个 NP 和多个介词短语,具体的结构主要和动词有关。广泛地使用特征系统,你就可以建立一个语法,它能接受以前提到的任何一种形式的补语,将实际中的动词和补语一致性问题转化为特征约束来处理。语法 4.14 给出了这个 S 网络。其中,边上的数字符合第 3 章讨论的习惯用法,比如,边 S3/1 指的是从节点 S3 出发,标记为 1 的那条边。语法 4.15 中的 NP 网络允许接受简单的名称、不带修饰成分的复数名词、代词以及由限定词、形容词和中心名词构成的简单序列。合法的名词补语包括若干数目的介词短语。语法 4.16 中的介词短语网络非常直截了当。使用该语法进行句法分析的问题,作为练习留给你。

4.6.2 寄存器的预设置

我们还可以对 ATN 的特征操作机制做进一步的扩展,这就牵涉到网络调用时寄存器的预设置功能。ATN 的寄存器预设置和编程语言中的参数传递非常类似。在原始的 ATN 系统中,我们称这种机制为 **SENDER** 操作,它对网络中信息的传递非常有用,而传入的信息有助于新语法成分的分析。

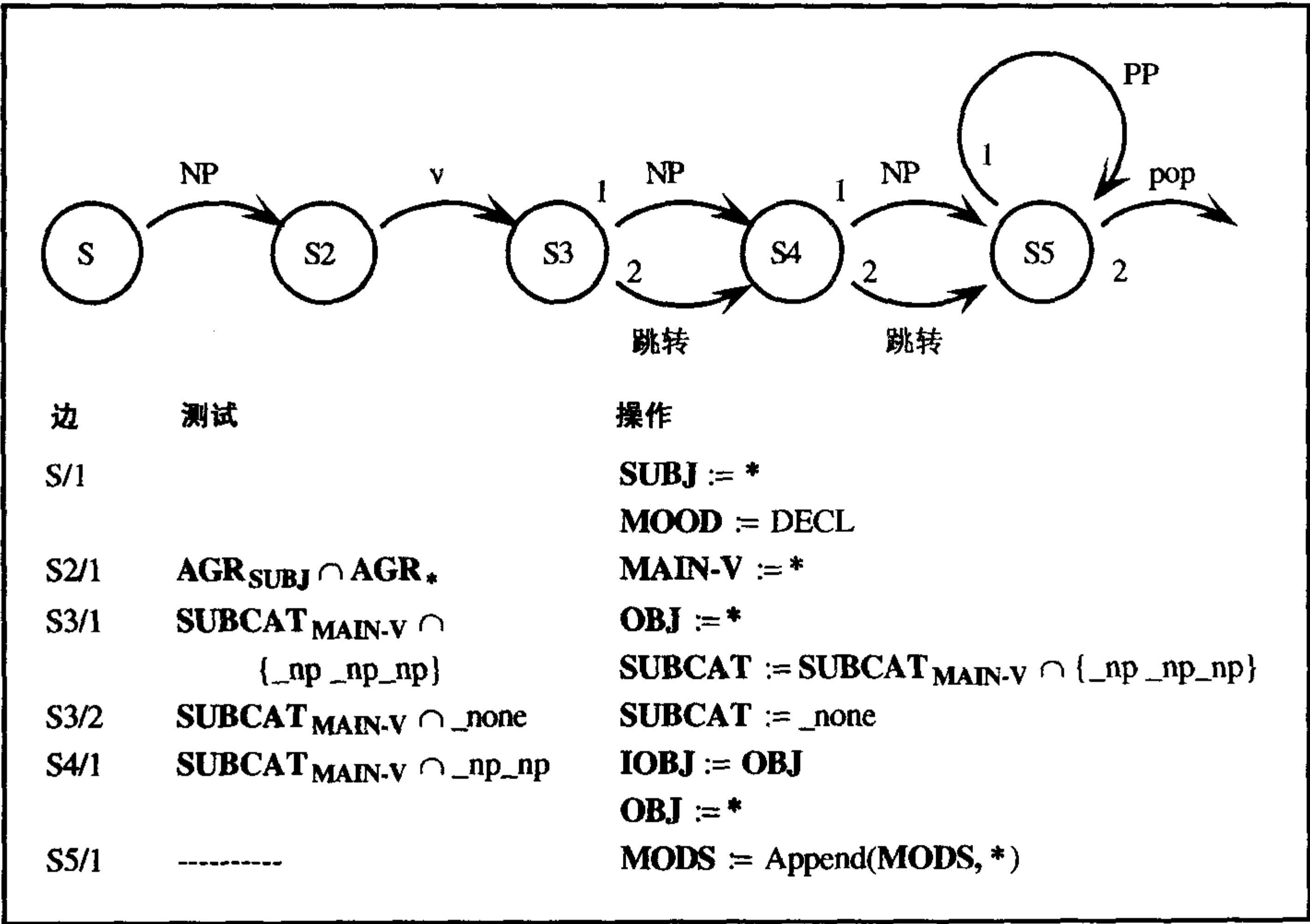
现在,我们看看那些以不定式作为补语的动词类别,“want”和“pray”就是其中的两个。不定式形式通常由词语“to”引导。根据 4.2 节的分类,不定式还包括如下形式:

- _ vp:inf Mary wants *to have a party*. (Mary 想开个派对。)
- _ np _ vp:inf Mary wants John *to have a party*. (Mary 想让 John 开个派对。)

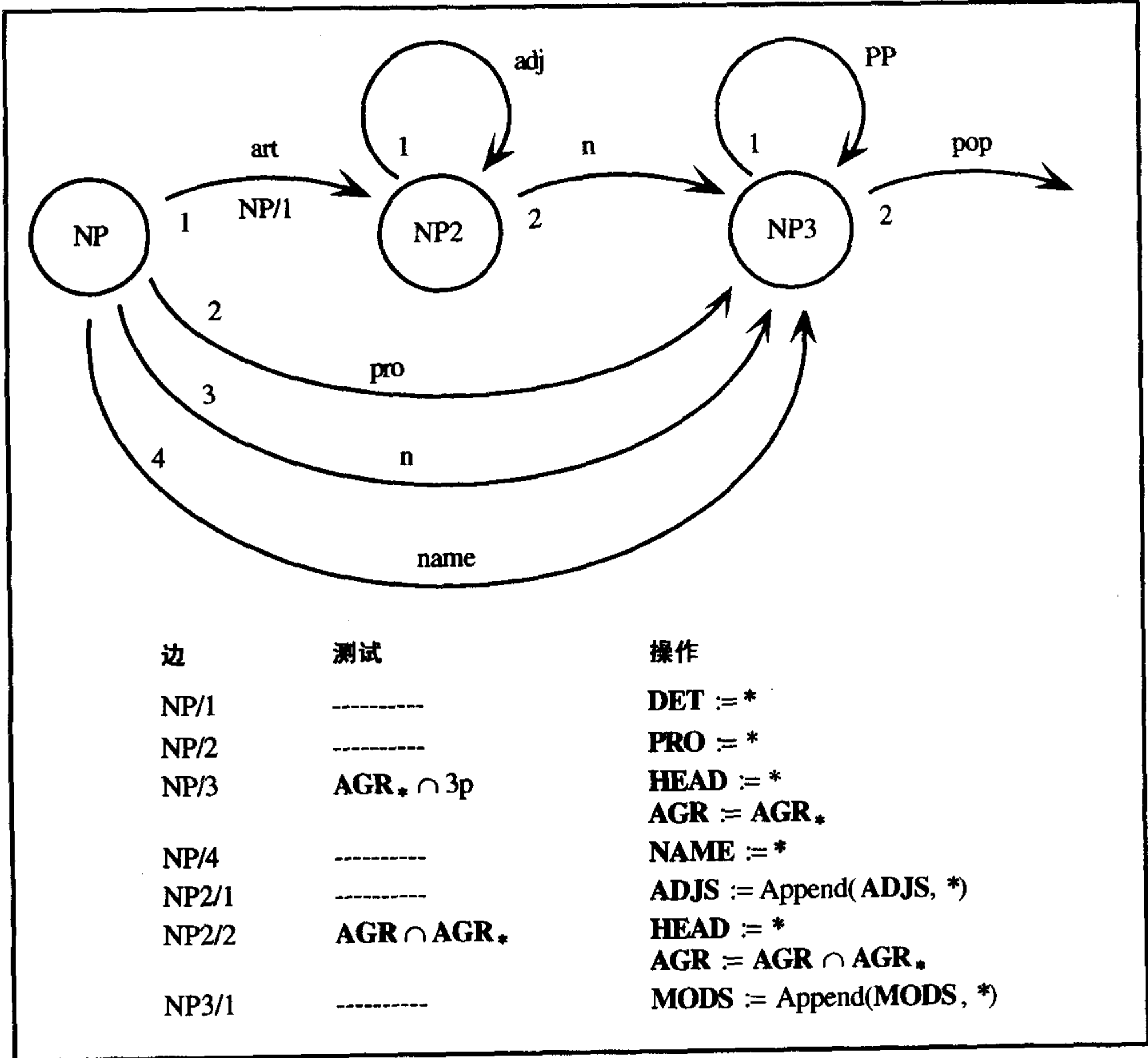
跟踪 S 网络				
步骤	节点	位置	通过的边	寄存器集合
1.	S	1	边 4 成功 (参见下面对递归调用的跟踪)	SUBJ ← (NP DET the HEAD dog AGR 3s)
5.	S1	3	边 5 (检查{3p ∩ 3p是否成立)	MAIN-V ← saw AGR ← 3p
6.	S2	4	边 6 成功 (参见下面对递归调用的跟踪)	OBJ ← (NP NAME Jack AGR 3s)
9.	S3	5	弹出边成功	返回 (S SUBJ (NP DET the HEAD dog AGR 3s) MAIN-V saw AGR 3p OBJ (NP NAME Jack AGR 3s))
跟踪对 NP 的第一次调用: 边 4				
步骤	节点	位置	通过的边	寄存器集合
2.	NP	1	1	DET ← the AGR ← {3s 3p}
3.	NP1	2	2 (检查{3s 3p} ∩ 3p是否成立)	HEAD ← dog
4.	NP2	3	弹出	返回 (NP DET the HEAD dog AGR 3s)
跟踪对 NP 的第二次调用: 边 6				
步骤	节点	位置	通过的边	寄存器集合
7.	NP	4	3	NAME ← John AGR ← 3s
8.	NP2	5	弹出	返回 (NP NAME John AGR 3s)

图 4.13 “₁ The ₂ dog ₃ saw ₄ Jack ₅”上的测试和操作处理流程

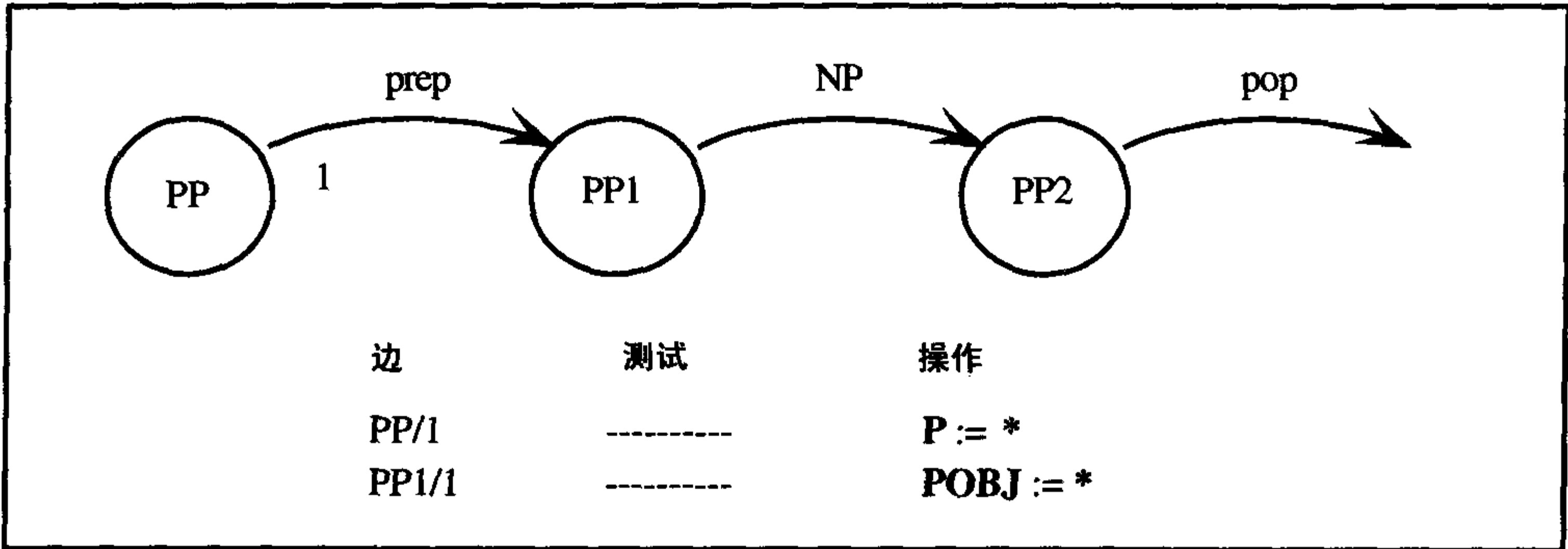
在我们以前探讨过的上下文无关文法中,这样的补语是作为 VFORM 值为不定式 inf 的 VP 来处理的。为了在 ATN 中进行同样的分析,需要调用 VP 的相应网络,同时还必须将这个网络中的 VFORM 寄存器预设为 inf。另外一种比较常用的分析是将补语看做特殊的句子形式,其中,句子中具备逻辑上可理解的主语。在第一个例子中,“Mary”就是逻辑主语;而在另一个例子中,逻辑主语是“John”。为了能进行这种分析,很多 ATN 语法都会在调用新的 S 网络时预设 SUBJ 寄存器的值。



语法 4.14 断言句的 S 网络



语法 4.15 NP 网络



语法 4.16 介词短语(PP)网络

◦ 4.7 确定子句语法

可以给每个谓词增加额外的参数来表示特征,这样就可以对一个逻辑语法进行扩展。举个非常简单的例子。可以增加一个表示数的参数,修改 PROLOG 的规则从而保证数的一致性特征,这个参数只需要增加到每个和数相关的谓词上。这样,就会得到一些带有特征的规则,如语法 4.17 所示。

```
1.  np(P1, Number, P3) :- art(P1, Number, P2), n(2, Number, P3)
2.  art(l, Number, O) :- word(Word, l, O), isart(Word, Number)
3.  isart(a, 3s) :-
4.  isart(the, 3s) :-
5.  isart(the, 3p) :-
6.  n(l, Number, O) :- word(Word, l, O), isnoun(Word, Number)
7.  isnoun(dog, 3s) :-
8.  isnoun(dogs, 3p) :-
```

语法 4.17

现在,让我们分析一下名词短语“the dog cried”。句法分析中有下面的断言:

```
word(the, 1, 2) :-
word(dog, 2, 3) :-
```

根据语法 4.17 中的规则 2 来分析词“the”。根据定理推导,我们可以得到该词的数特征。你得到这个结论,仅仅需要一个简单流程来证明下面的结论:

```
np(1, Number, 3)
```

采用规则 1,可以得到下面这些子目标:

```
art(1, Number, P2)
n(P2, Number, 3)
```

使用规则 2 证明下面这些谓词,就可以最终完成第一个子目标:

```
word(the, 1, 2) isart(the, 3s)
```

在规则 2 中,变量绑定为:

Number ← 3s
P2 ← 2

因此,现在的第二个子目标就是:

n(2, 3s, 3)

使用规则 6,又可以将其归约为子目标 word(Word,2,3)和 isnoun(Word,3s),将 Word 实例化为“dog”,根据输入串和规则 7,我们最终能分别证明这两个子目标。因此,句法分析成功,我们同时也验证了数的一致性。

在规则中另外增加一个参数,也可以进一步扩展该语法,扩展后它可以记录整个句法分析的结构。例如,为了建立句法分析树,可以定制一些规则,如语法 4.18 所示。针对句子“The dog cried”,这些规则可以保证你证明下面的结论:

S(1, 3s, s(np(art(the), n(dog)), vp(v(cried))), 4)

也就是说,在位置 1 和位置 4 之间有一个数特征为 3s 的句子,其结构为:

s(np(art(the), n(dog)), vp(v(cried)))

该结构就是图 4.19 所示句法分析树的一种表示方法。

1.

s(P1, Number, s(Np, Vp), P3) :-
 np(P1, Number, Np, P2), vp(P2, Number, Vp, P3)
2.

np(P1, Number, np(Art, N), P3) :-
 art(P1, Number1, Art, P2), n(P2, Number2, N, P3)
3.

vp(P1, Number, vp(Verb), P2) :-
 v(P1, Verb, P2)
4.

art(I, Number, art(Word), O) :-
 word(Word, I, O), isart(Word, Number)
5.

n(I, Number, n(Word), O) :-
 word(Word, I, O), isnoun(Word, Number)
6.

v(I, Number, v(Word), O) :-
 word(Word, I, O), isverb(Word, Number)

语法 4.18

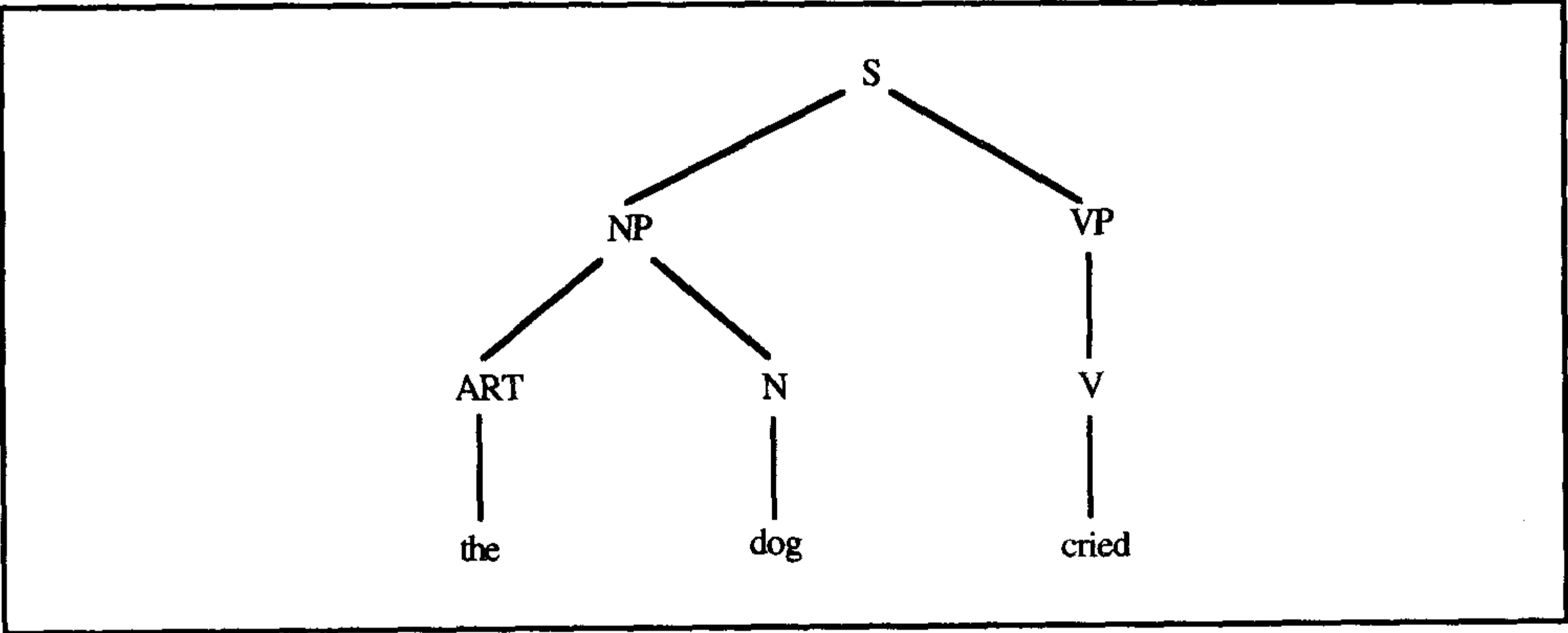


图 4.19 结构的树表示

为了制定语法,大多数基于逻辑的语法系统都会提供更便利的表示格式,这种格式可以自动地转换为和语法 4.18 类似的 PROLOG 子句。每个谓词上都有词的位置参数,它们可以由系统自动添加,因此可以忽略。类似地,所有表示终结符的谓词都可以由系统定义,因此,系统可以自动产生这些规则。这些缩减方式都可以表示为一种特定的格式,我们称之为确定子句语法(DCG, definite clause grammar)。

例如,要制定语法 4.18,只需要给出下面三条 DCG 规则(符号“ \rightarrow ”通常用于标记 DCG 规则):

```
s(Number, s(Np, Vp))  $\rightarrow$  np(N1, Np), vp(N2, Vp)
np(Number, np(Art, N))  $\rightarrow$  art(N1, Art), n(N2, N)
vp(Number, vp(Verb))  $\rightarrow$  v(Number, Verb)
```

在形式化方面,这个结果和 4.3 节中叙述的扩充上下文无关文法很相似。如果采用“特征/值”的形式表示结构,这种相似性会更加显著。现在,我们就采用特征结构作为单独的参数,并代替所有位置的参数。因此,可以得到一个与语法 4.7 前五条规则等同的 DCG,如语法 4.20 所示。其中,方括号表示的是 PROLOG 中的表结构。注意,S,NP 和 VP 采用同一个变量作为其参数。所以,只有在它们的 agr 特征完全一致的前提下,规则 1 才可以使用。

1. $s([inv - agr\ Agr]) \rightarrow np([agr\ Agr]), vp([agr\ Agr\ vform\ pres])$
2. $np([agr\ Agr2]) \rightarrow art([agr\ Agr2]), n([agr\ Agr2])$
3. $np([agr\ Agr3]) \rightarrow pro([agr\ Agr3])$
4. $vp([agr\ Agr4\ vform\ Vf3]) \rightarrow$
 $v([subcat\ _none\ agr\ Agr4\ vform\ Vf3])$
5. $vp([agr\ Agr4\ vform\ Vf3]) \rightarrow$
 $v([subcat\ _np\ agr\ Agr4\ vform\ Vf3])\ np()$

语法 4.20 语法 4.7 的 DCG 形式

显然,只有每个语法成分按照相同的顺序来指定特征值,并保证特征表的一致性,这种表示才能有效。比如,如果 S 中包括了“agr”特征、“inv”特征和“vform”特征,那么使用 S 的每条规则都必须按照同一顺序来指定所有这些特征。在实际语法中,这是非常困难的,因为每个语法成分都会有多种可能的特征,而且每种特征都必须各种情况下列出来。解决这个问题的一个办法是对 DCG 规则到 PROLOG 规则的转换程序进行扩展,使该转换程序能够采用变量值添加任何一个未确定的特征。采取这种方法,用户只需要给出那些重要的特征。

4.8 广义特征体系与合一文法

可以看到,特征结构对上下文无关文法与转移网络体系的一般化十分有用。实际上,特征结构还可以进一步泛化,到一定程度之后,上下文无关文法都不是十分必要了。整个语法可以表示为特征结构间约束关系的集合,这样的语法系统通常称为合一文法(unification grammar)。这一节主要介绍这种形式化手段内在的一些基本问题。

两个特征结构之间的扩展关系是合一文法的关键概念。如果特征结构 F1 中的每个特征值在特征结构 F2 中都已经指定了,则说明 F1 可以扩展 F2,或者说 F1 比 F2 更具体。例如,特征结构

(CAT V
ROOT cry)

就扩展了特征结构(CAT V),因为后者的 CAT 值为 V,而没有对 ROOT 特征进行限制。另一方面的例子如下所示:

(CAT V
ROOT cry)

(CAT V
VFORM pres)

在上面的两个例子中,任何一个特征结构都没有扩展另外一个特征结构,因为彼此都缺少对方的一些信息。具体地说,前者缺少后者的 VFORM 特征,而后者又缺少前者的 ROOT 特征。

如果存在一个特征结构能扩展某两个特征结构,那么这两个特征结构就能合一。最广合一(most general unifier)指的是能够同时扩展这两个特征结构的最小特征结构。以上两个特征结构的最广合一为:

(CAT V
ROOT cry
VFORM pres)

不难看到,这个特征是原来两个特征结构的扩展,同时不存在比它更小的扩展特征结构。相反,结构

(CAT V
AGR 3s)

(CAT V
AGR 3p)

不能合一,因为这两个特征结构的 AGR 特征值矛盾,它们不存在共同的扩展特征结构。

还可以进一步扩展这个形式化体系,可以允许自然存在的简单析取值(比如{3s 3p})。例如,(AGR 3s)可以扩展为(AGR{3s 3p})。

合一概念都是用来描述语法的,其中,所有的特征一致性检查及其操作都可以用合一关系来表示。语法 4.7 中的规则“ $S \rightarrow NP VP$ ”,采用合一文法可以表示为:

$$\begin{aligned} X_0 \rightarrow X_1 X_2 \quad & \text{CAT}_0 = S \\ & \text{CAT}_1 = NP \\ & \text{CAT}_2 = VP \\ & \text{AGR}_0 = \text{AGR}_1 = \text{AGR}_2 \\ & \text{VFORM}_0 = \text{VFORM}_2 \end{aligned}$$

这就意味着,成分 X_0 可以由成分 X_1 和 X_2 构成的序列生成。其前提是, X_0 的 CAT 为 S,成分 X_1 的 CAT 为 NP,成分 X_2 的 CAT 为 VP,所有成分的 AGR 值都一样,而且 X_0 和 X_2 的 VFORM 值相同。如果 CAT 值事先都给定了,那么规则可以缩写如下:

$$\begin{aligned} S \rightarrow NP VP \quad & \text{AGR} = \text{AGR}_1 = \text{AGR}_2 \\ & \text{VFORM} = \text{VFORM}_2 \end{aligned}$$

其中,规则中使用了 CAT 值,而且下标 0 也省略了。使用这种缩写方式,可以将语法 4.7 中的部分成分改写为合一文法,结果如语法 4.21 所示。因为这种语法仍然保留着上下文无关规则的结构,所以标准的句法分析算法均可应用于合一文法。下一节会具体说明如何通过特征等式的解析生成语法成分。

1'. $S \rightarrow NP VP$	$AGR = AGR_1 = AGR_2$ $VFORM = VFORM_2$
2'. $NP \rightarrow ART N$	$AGR = AGR_1 = AGR_2$
8'. $VP \rightarrow V ADJP$	$SUBCAT_1 = _adjp$ $VFORM = VFORM_1$ $AGR = AGR_1$
10'. $ADJP \rightarrow ADJ$	

语法 4.21 合一文法示例

4.8.1 形式化拓展:特征的有向无环图结构

如果将特征结构表示为有向无环图(DAG, directed acyclic graph),那么,我们就能准确地定义出基于合一形式的语法。其中,每个语法成分和相应的值用节点来表示,特征用标记边来表示。下面两个语法成分的 DAG 表示见图 4.22:

- N1: (CAT N
ROOT fish
AGR {3s 3p}))

N2: (CAT N
AGR 3s)

有向无环图的源点指的是那些没有输入边的节点。特征结构有向无环图的源点惟一,我们也称之为根节点。也可以说,该有向无环图源于这个节点。有向无环图的汇点指的是那些没有输出边的节点,特征结构的汇点采用一个原子特征(atomic feature)或者特征集合(例如,{3s 3p})来标记。

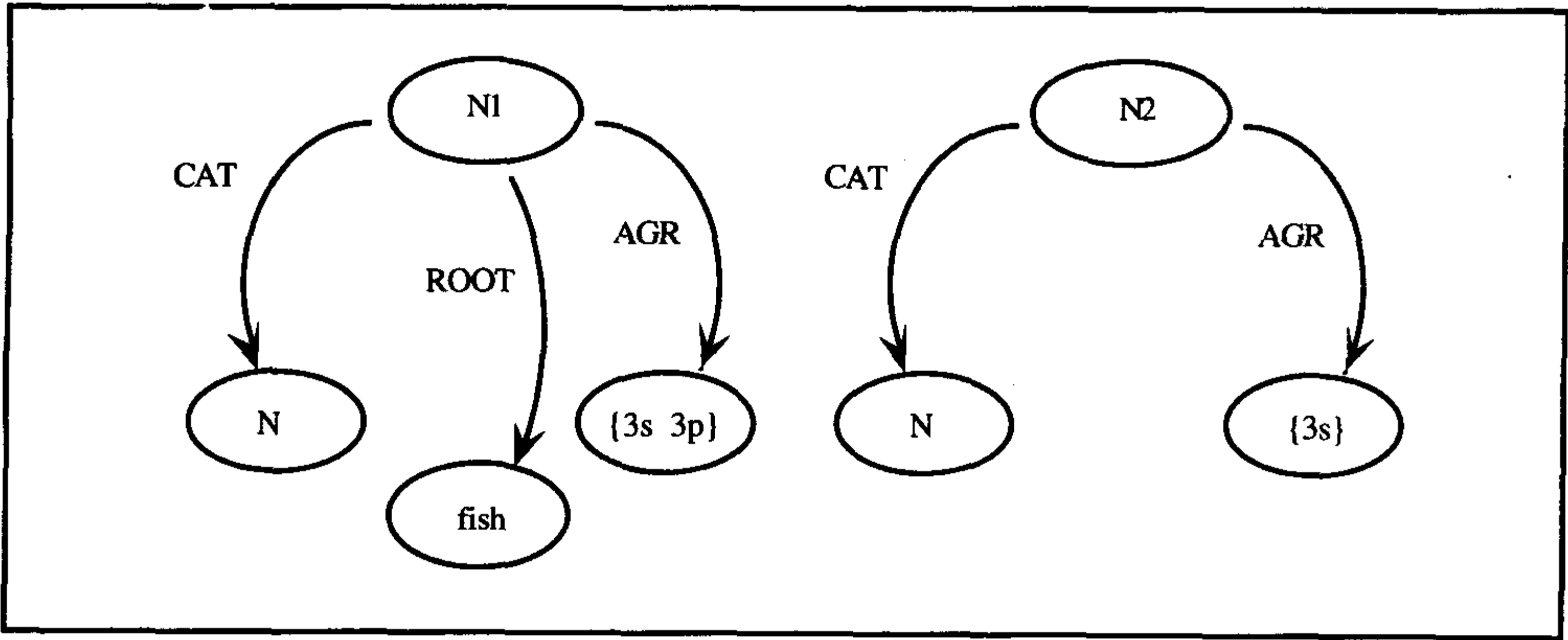


图 4.22 两个名词短语的有向无环图

根据图的匹配算法,我们现在就可以定义什么是两个特征结构的合一结构。这个算法的输入是两个带根节点的图,最后返回两者合一之后的新图,图 4.23 给出了具体的算法过程。将这个算法应用于图 4.22 中的节点 N1 和节点 N2,其结果是下面的这个新语法成分:

- N3: (CAT N
ROOT fish
AGR 3s)

采用该算法,你可以自己手工来分析处理这个例子,这样就可以明白该算法究竟是怎样运行的。这是个非常简单的例子,只存在一层递归。 N_1 和 N_2 的第一次调用在算法步骤 3 中处理,接下来,每一步递归调用仅仅和步骤 2 的汇点匹配操作相关。

将根节点分别为 N_i 和 N_j 的 DAG 进行合一,具体算法如下:

1. 如果 N_i 等于 N_j ,则合一结果为 N_i 并返回成功。
2. 如果 N_i 和 N_j 都是汇点,如果它们标记集合的交集非空,那么返回一个新的节点,该节点的标记为 N_i 和 N_j 标记集的交集,否则这两个 DAG 不能合一。
3. 如果 N_i 和 N_j 不都是汇点,则创建新的节点 N 。对于每条标记为 F 、从节点 N_i 到节点 NF_i 的边有:
 - 3a. 如果从节点 N_j 到节点 NF_j 存在一条标记为 F 的边,则递归地对 NF_i 和 NF_j 进行合一操作。生成一条标记为 F 的新边,它的起点为 N ,终点为递归操作之后的结果。
 - 3b. 如果不存在从节点 N_j 出发、标记为 F 的边,则生成一条标记为 F 、从节点 N 到节点 NF_i 的边。
 - 3c. 对于每条标记为 F 、从节点 N_j 到节点 NF_j 的边,如果不存在从 N_i 出发、标记为 F 的边,则生成一条标记为 F 、从节点 N 到节点 NF_j 的边。

图 4.23 图的合一算法

在这个算法的基础上,使用图合一方程式,我们可以给出新语法成分的构建算法。一旦实现了这个算法,就可以采用任何标准的句法分析算法来搭建一个句法分析器。根据含特征方程式的规则,该算法可以构建一个类别为 C 的新成分。特征方程式的形式为 $F_i = V$,其中 F_i 表示的是第 i 个子成分的 F 特征,图 4.24 给出了具体的算法。

给定规则 $X_0 \rightarrow X_1, \dots, X_n$ 和特征方程式集合,特征方程式的形式为 $F_i = V$ 。其中, SC_1, \dots, SC_n 是 X_1, \dots, X_n 对应的子成分,下面的算法将生成一个能满足所有特征方程式约束的 DAG 图。

1. 创建一个节点 CC_0 ,该节点为新特征结构的根。
2. 复制每个以 SC_i 为根的 DAG,新的根设为 CC_i ,增加一条从节点 CC_0 到节点 CC_i 、标记为 i 的边。
3. 对每个形式为 $F_i = V$ (其中 V 是具体的值)的特征方程式,沿着节点 CC_i 到节点 N_i 的 F 特征边,将 N_i 和 V 进行合一。
4. 对每一个形式为 $F_i = G_j$ 的特征方程式,有:
 - 4a. 如果存在从节点 CC_i 出发的 F 特征边,同时存在从节点 CC_j 出发的 G 特征边,则:
 - i. 沿着 F 链接边到达节点 N_i ,沿着 G 特征边到达节点 N_j 。
 - ii. 采用图合一算法,将 N_i 和 N_j 进行合一,创建新的节点 X 。
 - iii. 将所有指向节点 N_i 或者节点 N_j 的边,改为指向节点 X 。
 - 4b. 如果不存在从节点 CC_i 出发的 F 特征边,但从节点 CC_j 出发的 G 特征边存在,则创建一条从节点 CC_i 到节点 N_j 的 F 特征边。
 - 4c. 如果不存在从节点 CC_j 出发的 G 特征边,但从节点 CC_i 出发的 F 特征边存在,则创建一条从节点 CC_j 到节点 N_i 的 G 特征边。

图 4.24 采用特征方程式的新语法成分生成算法

现在,让我们来看一个具体的例子。假设我们事先已经定义好了图 4.25 中的两个语法成分,则采用 LISP 的标记体系,它们可以表示为:

ART1: (CAT ART
 ROOT the
 AGR {3s 3p})

N1: (CAT N
 ROOT fish
 AGR {3s 3p})

根据语法 4.21 中的规则 2',可以得到新的 NP。方程式为：

CAT₀ = NP
CAT₁ = ART
CAT₂ = N
AGR = AGR₁ = AGR₂

该算法生成的语法成分 DAG 表示见图 4.26。

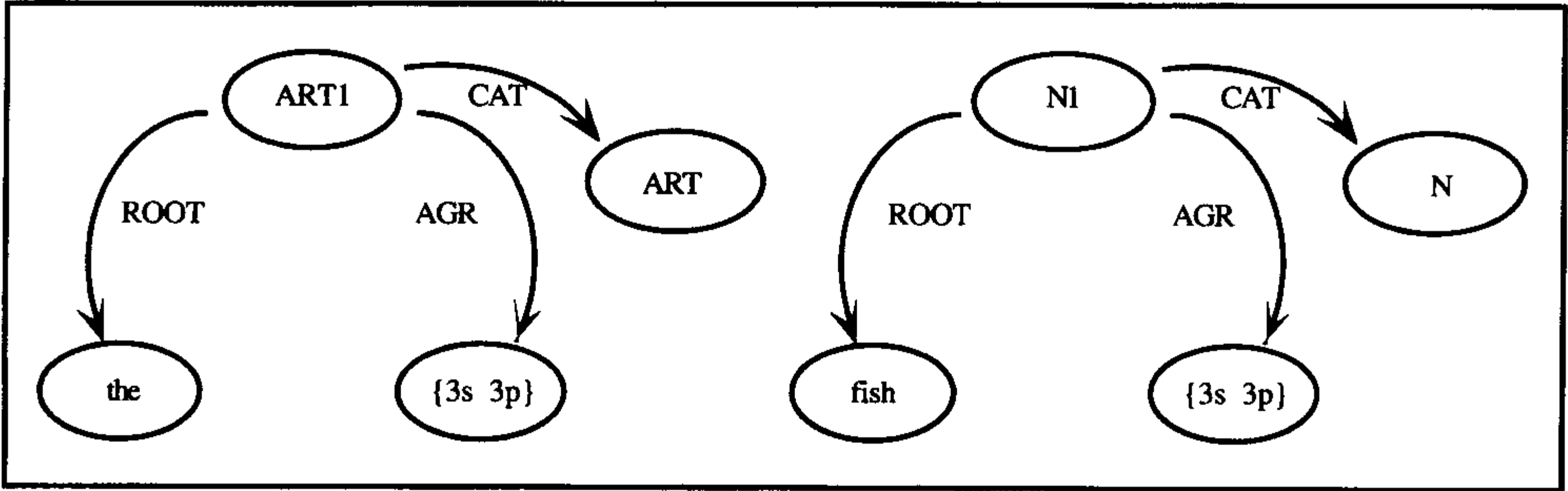


图 4.25 “the”和“fish”词条

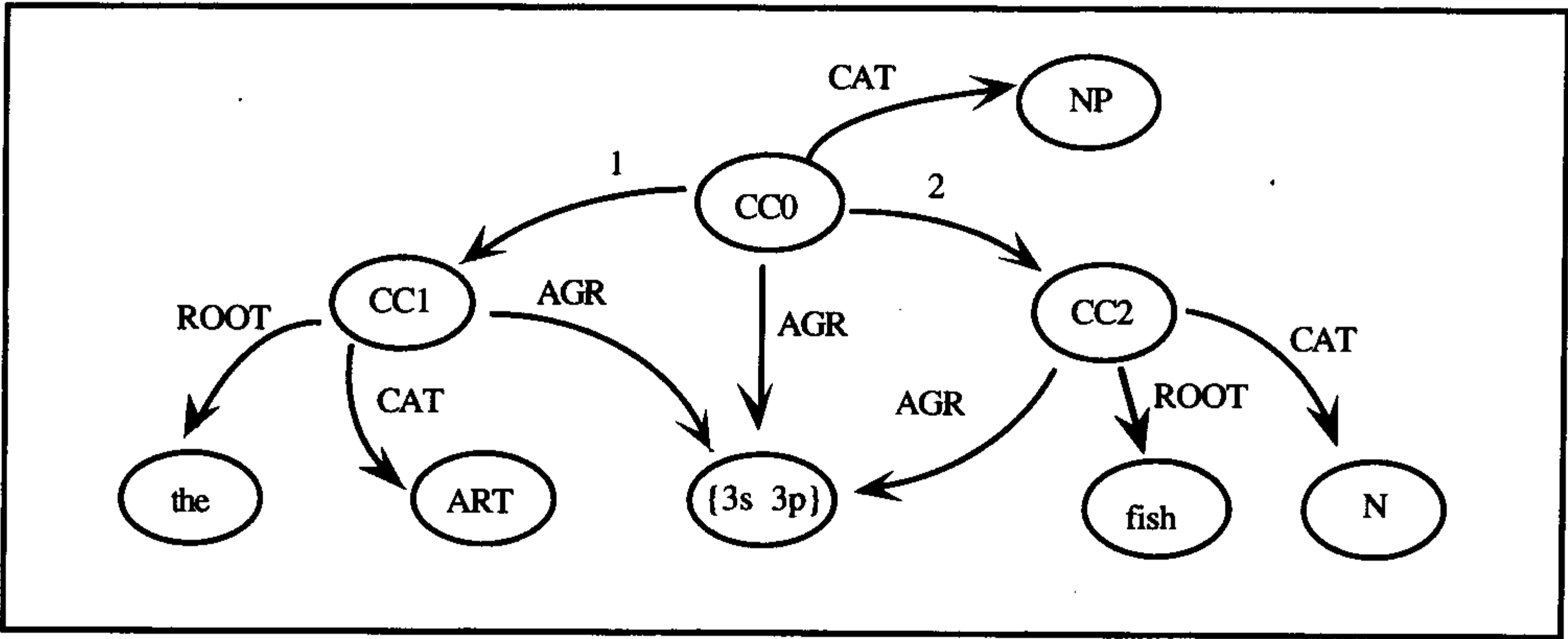


图 4.26 名词短语“the fish”的图结构

继续举例,采用类似的方法,我们可以分析动词短语“is happy”,分析结果如图 4.27 所示。为了使图简化,我们没有给出 CAT 边。根据语法 4.21 中的规则 1',我们可以构造出“The fish is happy”的句法分析结果,图 4.28 给出了具体的分析过程。其中,语法成分 S1, NP1, ART1, N1, VP1 和 V1 的 AGR 槽值指向同一节点,因此,CC1 的 AGR 特征值会相应变化。比如,在 NP1 和 VP1 的特征进行合一的时候,该值就会变化。

很明显,到目前为止,在语法表达能力上,我们所采用的合一文法仅仅和前面讨论过的扩充上下文无关文法相当。实际上,合一文法体系的表达能力还是强得多,因为它并不要求规则都必须基于句法类。例如,英语中就有一类用做表语的短语,通常出现在下面这种句式中:

NP be _

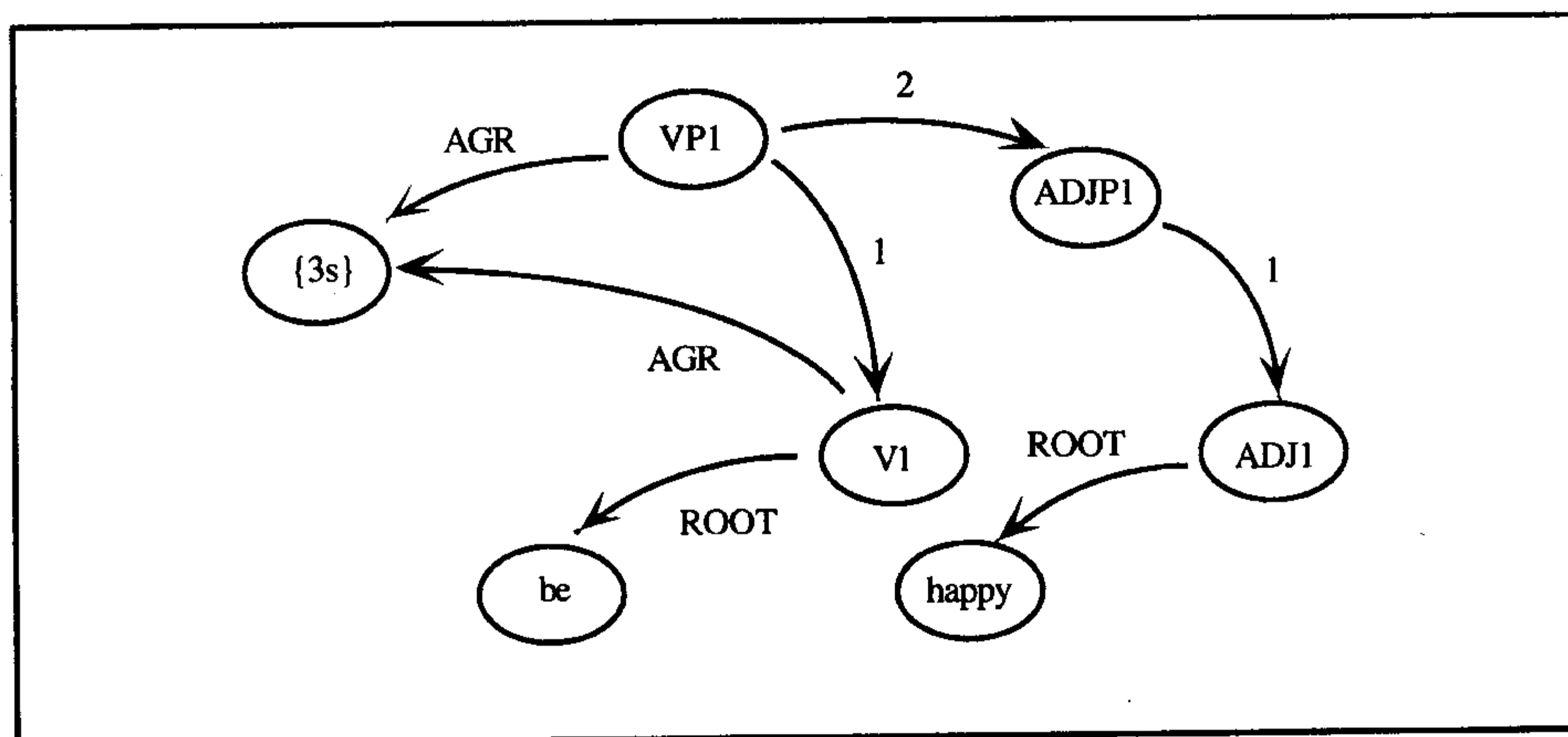


图 4.27 动词短语“is happy”的分析

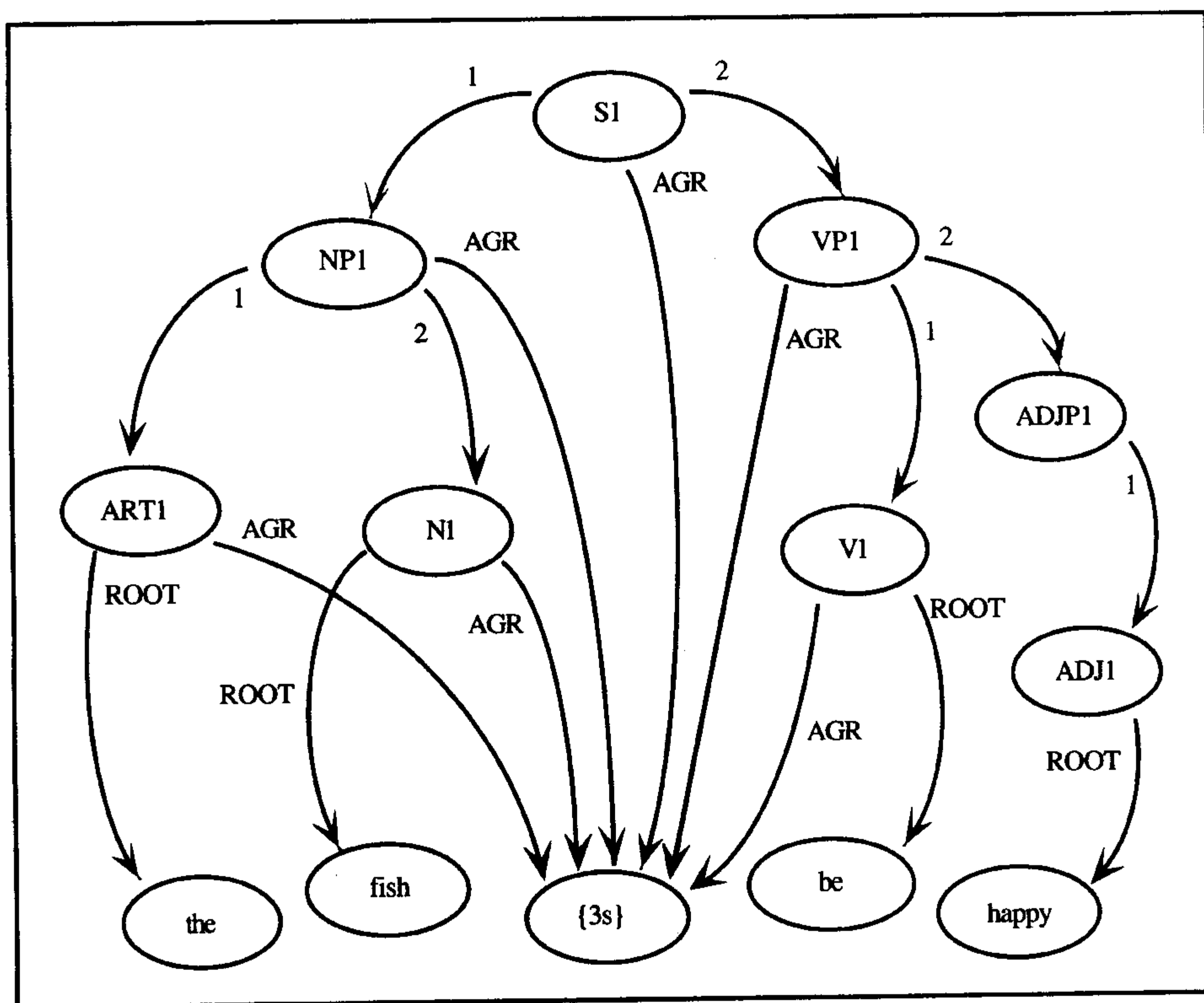


图 4.28 “The fish is happy”的句法分析过程

这种短语包括介词短语(“He is in the house”, 他在房子里)、名词短语(“He is a traitor”, 他是个叛徒)和形容词短语(“He is happy”, 他开心着呢)。语法中通常引入二元特征, 设为 PRED, 它对可以使用该短语为真。在标准的上下文无关文法中, 要对每类的不同规则做具体说明, 比如:

VP → (V **ROOT** be) (NP **PRED** +)
 VP → (V **ROOT** be) (PP **PRED** +)
 VP → (V **ROOT** be) (ADJP **PRED** +)

如果采用合一文法的框架,只要一条规则就可以处理所有的类别,即:

$$\begin{aligned} X_0 &\rightarrow X_1 X_2 & \text{CAT}_0 &= \text{VP} \\ & & \text{CAT}_1 &= \text{V} \\ & & \text{ROOT}_1 &= \text{be} \\ & & \text{PRED}_2 &= + \end{aligned}$$

允许任何一个具有 + PRED 特征的 X_2 成分使用这条规则。

当然,如果语法成分的类别没有具体指定,那么,如何调整标准上下文无关文法分析算法使之应用于合一文法? 这一点并不十分明确。不难看出,只要存在有限的特征子集,而且每个语法成分中至少指定了一个特征,那么,合一文法就可以转换为相应的上下文无关文法。这种特征集有时叫做语法的上下文无关骨干。

合一文法允许在词典中表示更多的信息是它的另一大特色。实际上,几乎整个语法都可以放到词典中表示,语法中只需要保留很少的规则。现在,我们来看看那些前面介绍过的、用来处理不同动词次范畴的规则。这些规则可以压缩为这样一些规则:一条规则针对那些只需要一个子成分的动词次范畴,另一条规则针对那些需要两个子成分的动词次范畴,依次类推。在实际运用过程中,我们需要对其补语结构的类别进行限制,而这些约束信息都可以在词典的动词词条中表示。例如,动词“put”的词条可能是:

put: (CAT V SUBCAT(FIRST(CAT NP)SECOND(CAT PP LOC +)))

针对那些只划分两个子成分的动词,其一般性规则如下:

$$\begin{aligned} \text{VP} &\rightarrow \text{V } X_2 X_3 & 2 &= \text{FIRSTSUBCAT}_1 \\ & & 3 &= \text{SECONDSUBCAT}_1 \end{aligned}$$

这条规则可以接受任何一个“V X_2 X_3 ”序列。其中, X_2 与 X_1 的 SUBCAT 中的第一个特征合一,而 X_3 与 X_1 的 SUBCAT 中的第二个特征合一。例如,假定 V 是动词“put”,那么,该规则要求 X_2 与 (CAT NP) 合一, X_3 与 (CAT PP LOC +) 合一。当然,对不同的动词来说,同一规则却可以对 X_2 和 X_3 施加完全不同的限制。比如“want”要求 X_3 与 (CAT VP VFORM inf) 合一。

这些技术可以大大地减少语法的规模。比如,合理的语法至少需要 40 条规则处理动词的次范畴,但采用特征合一技术后,牵涉到动词补语有三个子成分的时候并不需要任何规则。因此,这 40 条规则在合一文法中可以减少到 4 条,其中还包括空补足语的规则。而且,如果将 X_1 上的类别限制进行泛化,这些相同的规则还可以重新用在所有名词和形容词次范畴的处理上。

4.9 小结

通过在每个语法成分中加入特征,本章扩充了语法规则,并扩展了第3章介绍过的语法形式化体系。特征对于建立面向自然语言的、覆盖面更广的语法非常重要。我们定义了英语中一个有用的特征集,并进一步研究了将词语映射为特征结构(即语法成分)的词语形态分析技术。同时,本章还探讨了几种不同的语法扩展形式。第一种方法允许上下文无关规则在基本的语法类之上增加特征的一致性约束,这样就产生了扩充上下文无关文法。第3章中给出的标准句法分析算法可以进行扩展,扩展后可以处理这种扩充语法。第二种方法提供的是扩充转移网络。扩充转移网络可以在每条边上定义不同的检测和操作来分析处理特征结构。最后一种技术方法基于特征结构的合一处理,采用合一特征可以得到合一文法。

4.10 相关工作与深入阅读材料

Knuth(1968)引入了属性语法来分析程序语言,从此以后,扩充上下文无关文法就一直应用在计算模型当中。从那时候起,许多系统开始利用某种形式的符号规则。很多早期系统实现的操作类型常常是简单的特征测试,以及采用我们这里讨论的技术路线来创建各种结构。这些系统大多是依靠随意的 LISP 代码来表示各种符号标记,这样的系统例子包括 Sager(1981)和 Robinson(1982)。

框 4.3 词汇功能语法

词汇功能语法(Kaplan 和 Bresnan,1982)是形式计算发展过程中很有影响的语言学理论,词汇功能语法可以简写为 LFG。LFG 可以看成是一种合一语法,下面是一条典型的 LFG 规则:

$$\begin{array}{ccc} S \rightarrow & NP & VP \\ & (\uparrow \text{SUBJ}) = \downarrow & \uparrow = \downarrow \end{array}$$

其中,上箭头(↑)指向规则左部的语法成分(比如 S 成分),而下箭头(↓)指向规则右部黏附的语法成分。因此,该规则和下面的标记体系等同:

$$\begin{array}{ccc} S \rightarrow NP VP & \text{SUBJ} = 1 & \\ & 0 = 2 & \end{array}$$

需要注意的是,整个 S 结构和 VP 结构的合一使它们变成了同一成分。从计算的角度来看,这可以看成是将寄存器从 VP 传递到 S 的一个高效方法。不过,这也存在语言学上的负面影响,Kaplan 和 Bresnan(1982)对此做了探讨。负面影响主要在于 S 和 VP 是同一个成分,所以在 S 或者 VP 结构上做任何进一步的修正都会同时影响到这两个成分。

LFG 能够在词典中表示绝大多数信息。具体地说,词条可以表示它们对成分的哪些槽能够进行填充。例如,“a”,“the”和“bird”的词条可以表示如下:

<i>a</i>	ART	(↑ SPEC) = INDEF (↑ AGR) = (3s)
<i>the</i>	ART	(↑ SPEC) = DEF
<i>bird</i>	N	(↑ AGR) = (3s) (↑ HEAD) = BIRD

上箭头表示能实际填充 NP 结构的哪个槽。采用规则

$$NP \rightarrow ART N$$

分析短语“a bird”。当分析词语“a”的时候,NP 的 AGR 特征值设为 3s;而分析到词语“bird”时,会检查数的一致性,结果和 3s 一致。如果是冠词“the”,则不用进行一致性检查,因为“the”这个词并没有在它的 NP 中设置 AGR 特征。

本章使用到的一些特定特征和语法扩展的技术大都是基于传统的(Gazdar, Klein, Pullum 和 Sag, 1985)广义短语结构语法(GPSG, generalized phrase structure grammar)方面的工作。GPSG 引入了特征值的有限集,它们都可以和任意的语法符号相结合。除了没有特征名称之外,这些和本书中讨论的标记体系扮演相同的角色。所有的 GPSG 特征值都是惟一的,因此,GPSG 同时定义了特征的类型和值。例如,阴性、第三人称复数名词短语的语法类型为:

NP[PL,3,F]

特征值的数量有限,如果将每对语法类别和特征结合成一个符号,则语法和上下文无关文法在形式上对等。不过,GPSG 的一个重要贡献却是其丰富的结构,这种结构使得特征能够不断地衍化。GPSG 应用于所有语法规则主要依据特征衍化的通用准则,而不是采用显式的特征规则方程式。中心特征约定就是这种通用准则的很好例证。中心特征约定要求母成分所有的中心特征都必须和中心成分相同。另外一条通用准则要求各个语法成分之间必须保证一致性。我们本来需要为每一条规则手工定义很多特征方程式,不过本章中介绍了一些特征编写的习惯用法,它可以帮助减少特征方程式的数量,这个想法主要是受上述理论的启发。在 GPSG 和 LFG 方面,Sells(1985)做了非常出色的综述。

这里描述的 ATN 框架主要来源于 Woods(1970;1973)和 Kaplan(1973)的研究工作。Bates (1978)对 ATN 做了很好的综述。采用逻辑编程方法对自然语言进行句法分析起源于 20 世纪 70 年代早期 Colmerauer(1978)的工作,Pereira 和 Warren(1980)以及 Pereira 和 Shieber(1987)的论文或许是阐述这种方法最好的文献。其他一些有趣的研究可以参照 McCord(1980)和 Pereira (1981)。Alshaw(1992)是最近比较优秀的研究成果。

我们对合一文法的讨论大致是基于 Kay(1982)和 PATR-II 系统(Shieber, 1984;1986)方面的工作,在这个领域里,人们仍然积极地进行着大量的研究。Shieber(1986)在这方面做了出色的综述。在对不同形式的特征结构进行形式化方面,目前也有越来越多的工作逐步开展,比较好的例子有 Rounds(1988),Shieber(1992)和 Johnson(1991)。

4.11 习题

- 1. 【易】采用语法 4.7,对句子“The man cries”和句子“He wants to be happy”进行句法分析,在此基础上,请你画出完整的 chart 图。图 4.9 给出了这两个句子最终的分析。你可以使用任何一种句法分析算法,但必须声明具体使用的是哪种分析方法,而且最终的 chart 图必须包含每一个在分析过程中创建的完整的语法成分。
- 2. 【中】针对下面给出的词语,请定义出最小的词条集合。要求使用词语形态分析算法,我们可以识别出所有标准形式的动词,并防止生成不合法的词语形式。然后,讨论可能会引起的问题和你所做出的假设。如果必要,请对本章的词语形态分析算法提出修正建议。

原型	现在时	过去时	过去分词	现在分词
go	go, goes	went	gone	going
sing	sing, sings	sang	sung	singing
bid	bid, bids	bid	bidden	bidding

3. 【中】对图 4.6 中的词典和语法 4.7 进行扩展,要求扩展后的语法和词典可以接受以下两个句子:

He was sad to see the dog cry. (他看见狗在叫感到悲痛。)

He saw the man saw the wood with the saw. (他看见那个男人在用锯子锯木头。)

验证你的新规则,看看它们能正确处理的类似句子的范围。采用本章提供的句法分析器完成并测试扩展后的语法,或者采用自顶向下的 chart 句法分析器分析上面给出的每个句子,并画出完整的 chart 图。

4. 【中】

- a 编写一个特征语法,要求该语法可以将以下短语成功地识别为名词短语:

three o'clock	quarter after eight
ten minutes to six	seven thirty-five
half past four	

同时,该语法不允许如下的短语:

half to eight	three twenty o'clock
ten forty-five after six	

具体说明必要的特征操作。一旦句法分析过程完成, HOUR 特征和 MINUTES 特征就会出现在 NP 成分中。在处理过程中,如果需要对特征机制进行扩展,请详细描述出你的扩展过程。

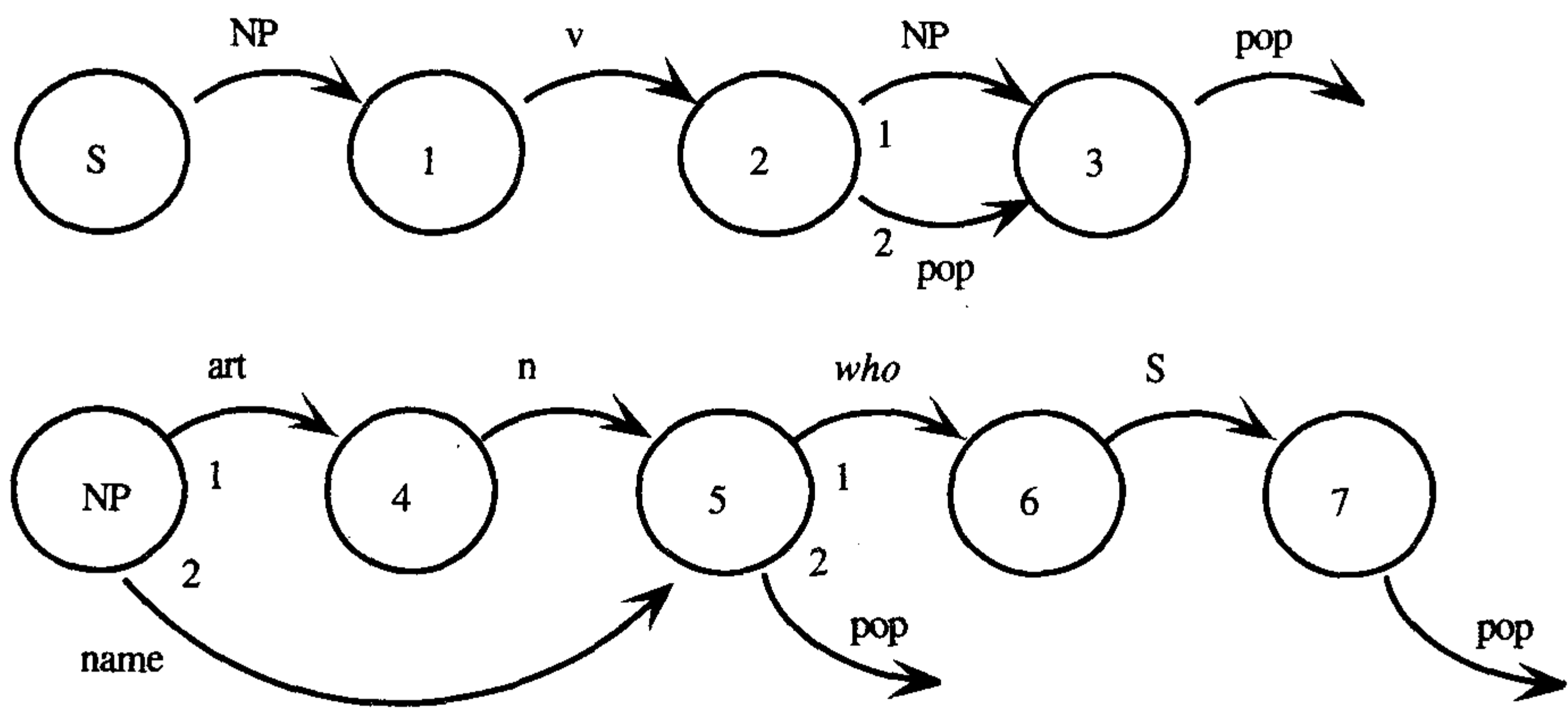
- b 另外选择两个可以被当前语法接受的短语形式,再找出一个不能被你的语法所接受而实际合理的短语。如果存在当前语法允许却不合法的短语,请给出一个例子。

5. 【中】在英语中,代词有主格和宾格之分,“I”可以作为主语使用,“me”可以作为宾语,类似的主格和宾格区别还有“he”和“him”,“we”和“us”以及“they”和“them”。代词“you”就不存在这种区别。现在,请给出扩充上下文无关文法和词典。该语法表示的是简单主谓宾结构的句子,要求主语和宾语位置上只能出现相应的代词,同时主语和谓语的数必须一致。因此,语法会接受句子“I hit him”,而不会接受“me love you”。编写语法时,必须考虑到刚才这个问题中提到的所有代词,但是动词词条只能有一个,同时语法中不能包含代词以外的其他名词短语。

- a. 按照词类给出一些词语,并给出四个结构不同的、能被当前语法网络接受的句子。
b. 采用本章中定义的标记体系,给出当前语法网络的扩充形式。要求主动词为“give”的句子,其主语必须是有生命的主体;而主动词为“be”的句子,其主语既可以是有生命的也可以是无生命的主体。给出包含部分词语的词典,要求这些词语能反映出这个网络的选择能力。

6. 【中】考虑以下简单的 ATN:

- a 指定一些词语的词类并给出四个网络可以接受的不同结构的句子。
b 用本章定义的符号来说明网络的扩充部分。如果主语是有生命的,那么主动词为“give”的句子是可以接受的,主动词为“be”的句子可以采用任何主语。给出一个网络,这个网络中的词语可以表示网络具有选择性。



7. 【中】使用下面的合一文法,采用句法分析器,对下面句子中的两个名词短语进行分析。当它们第一次创建完成后,请画出其语法结构的有向无环图。然后,给出整个句子“The fish is a large one”的有向无环图(同样包括 S 的所有子成分)。你可以将图 4.6 作为自己的词典,也可以在此基础上定义那些图 4.6 没有的词条。

1. S → NP VP

2. NP → ART N

3. NP → ART ADJ N

4. VP → V NP
- INV = -

VFORM₂ = pres

AGR = AGR₁ = AGR₂

AGR = AGR₁ = AGR₂

AGR = AGR₁ = AGR₃

VFORM = VFORM₁

AGR = AGR₁ = AGR₂

ROOT = BE1

第5章 自然语言的语法

扩充上下文无关文法为刻画自然语言的许多规律提供了一套强大的形式化体系。本章主要讨论英语结构几个方面的问题,并考察采用特征系统处理这些结构的方法。5.1节讨论助动词,接着引入一些特征来刻画顺序约束与一致性约束。随后,5.2节讨论一类普遍存在的问题,即语言的移位现象。剩下的章节则探讨几种移位现象的处理方法。5.3节讨论一般疑问句(yes/no question)和特殊疑问句(wh-question)的处理,5.4节讨论关系从句。余下部分讨论其他方法,5.5节讨论扩充转移网络中保留机制的运用,5.6节讨论逻辑语法中的缺位索引方法。

5.1 助动词和动词短语

英语中的句子一般都会包含一个助动词序列,并且在后面接上主动词。比如下面的句子:

I can see the house. (我可以看见这座房子。)

I will have seen the house. (我到那时就已经看过这座房子了。)

I was watching the movie. (我那个时候正在看那部电影。)

I should have been watching the movie. (我应当已经看过这部电影了。)

乍一看,这些句子好像是动词的随意组合,其中包括“have”,“be”,“do”,“can”,“will”等。实际上,这些动词具有非常丰富的结构。现在,我们看看这些助动词究竟如何来约束它们后面的动词。具体地说,助动词“have”后面必须跟过去分词形式(要么是另外的助动词,要么就是主动词),在助动词“be”的后面要么是现在分词形式,要么是在被动句中的过去分词形式。助动词“do”通常单独出现,不过,它后面也可以接动词的原型。(如,*I did eat my carrots!*,我的确吃了我的胡萝卜!)。而“can”和“must”之类的助动词在任何情况下都必须接动词的原型。另外,在简单陈述句中,第一个助动词(或者动词)必须与句子的主语一致,而且它还必须以限定形式(如过去或现在)出现。例如,**I going*, **we be gone*, **they am*都是不可接受的。

在这一节中,我们会探讨如何结合新的规则和特征约束来刻画带有助动词形式的句子结构。主要思想是引入助动词的次范畴特征,利用这些特征来约束动词短语的补语。为了达到这个目标,我们首先要明确地将助动词和主动词区分开来。某些助动词往往具备普通动词的许多属性,因此,区分它们非常重要。比如,在句子中,助动词可以放在副词“not”的前面,而主动词则不能。示例如下:

I am not going! (我不会去!)

You did not try it. (你没试过。)

He could not have seen the car. (他不可能看见过这部车子。)

除此以外,只有助动词才可以放在一般疑问句的主语 NP 之前,例如:

Did you see the car? (你看见那部车了吗?)

Can I try it? (我能试试吗?)

* Eat John the pizza?

相反,主动词可以作为句子的惟一动词出现。如果要把它变成一般疑问句,则需要添加助动词“do”,示例如下:

I ate the pizza. (我吃了比萨。)

Did I eat the pizza? (我吃过比萨吗?)

The boy climbed in the window. (那个男孩在窗户上爬。)

Did the boy climb in the window? (那个男孩在窗户上爬了吗?)

I have a pen. (我有只钢笔。)

Do I have a pen? (我有钢笔吗?)

主要的助动词大都是从原型“be”和“have”中衍生出来的。其他助动词则是情态助动词(modal auxiliary),它们一般只出现在限定时态形式中(如一般现在时和过去时)。简单地按照对应现在时和过去时形式进行组织,这些动词对包括“do(did)”,“can(could)”,“may(might)”,“shall(should)”,“will(would)”,“must”,“need”以及“dare”。另外,还有一些短语也可以当做情态助动词来使用,比如“ought to”,“used to”和“be going to”。

通过上面的例子可以发现,“have”和“be”既能作为助动词使用,也能作为主动词使用。这两种情况的差异很大,所以词典中将它们分别收录为助动词词条和主动词词条,两者具有不同的属性。例如,助动词“have”后面要接过去分词形式的动词短语,而动词“have”后面则需要接 NP 补语。

正如前面提到的,助动词处理的基本思想是将它们当做动词处理,其中,VP 是它的补语。而这个 VP 本身可能还包含另一个助动词和另一个 VP,它也可能是以主动词开头的 VP。因此,为了处理大多数这样的语法行为,我们可以采用下面的规则扩展语法 4.7:

$$VP \rightarrow (AUX \textit{COMPFORM} ?s) (VP \textit{VFORM} ?s)$$

VP 补语的 VFORM 用 COMPFORM 特征来表示。图 5.1 给出了助动词 COMPFORM 特征的取值范围。

助动词	补语形式	结构	示例
情态动词	base(原型)	情态语气	can see the house(可以看见那栋房子)
have	pastprt(过去分词)	完成时	have seen the house(已经看见了那栋房子)
be	ing(进行时)	进行时	is lifting the box(正在举那个盒子)
be	pastprt(过去分词)	被动式	was seen by the crowd(被乌鸦看到了)

图 5.1 助动词的 COMPFORM 特征约束

另外,助动词序列存在着其他一些约束。具体地说,助动词只能按照如下的顺序出现:

情态动词 + have + be(进行时) + be(被动式)

The song might have been being played as they left.
(当他们离开的时候,这首曲子已经被人演奏过了。)

为了刻画这些顺序的约束,我们可能还需要引入 8 条专门的规则。四个助动词的相应位置分别对应一条规则,另外还要再加上 4 条可选规则,这些可选规则保证每个助动词可以任意选用。不过,事实表明有一些限制并不符合特征约束。举例来说,情态动词没有分词形式。所以,在助动词序列里,情态动词不能跟在“have”或者“be”的后面。比如,下面就是一个错误的句子:

* He has might see the movie already.

这个句子违反了“have”的次范畴约束。你或许也会认为助动词“have”从来都不会以分词形式出现,它要么是在助动词序列的首位(而且是限定的),要么就跟在情态动词后面构成不定式。但是,这只有在简单主句中才是正确的。如果再看看某些动词 VP 补语中出现的助动词序列,你会发现这时需要将“have”的分词形式作为助动词。“regret”就是这样的动词,例句如下:

I regret having been chosen to go. (我后悔被选去了。)

因此,基于次范畴特征的公式表述存在过生成的问题。也就是说,它会产生一些本不该有的句子。虽然它会接受任何一个合法的助动词序列,但是,它同样会接受下面这种错误的句子:

* I must be having been singing.

加入新的特征就能够解决这个问题。新的特征会对助动词“be”的补语做进一步的限制,这样,它们就不会再接受额外的助动词(被动句中再出现“be”除外)。我们引入二元中心特征 MAIN,“+”号表示任意的主动词,“-”号表示助动词。这样,就可以对“be”的 VP 补语进行约束,具体的规则如下:

$VP \rightarrow AUX[be] VP[ing, +main]$

接着,我们必须对“be”词条做相应的变更,从而保证原始的助动词规则不再适用。我们可以将 COMPFORM 特征设置为“-”来实现这种变化。剩下的最后一个问题是如何让语法接受被动语态,这有几种可能的解决办法。例如,可以将被动结构中的“be”看成主动词,而不是助动词。另外一个办法是简单地加入一条规则,它允许被动式作为补语。该规则引入新的二元特征 PASS,仅当动词短语涉及到被动语态时,该特征值才为“+”,即:

$VP \rightarrow AUX[be] VP[ing, +pass]$

那么,被动规则就是:

$VP[+pass] \rightarrow AUX[be] VP[pastprt, main]$

这些新规则很好地维护了顺序的约束关系,不过,为了让示例尽量简短,在本书中,我们一般直接使用第一条规则来处理助动词。尽管它存在生成过多的问题,但是就我们给出的示例来说,并不会产生什么问题。

图 5.2 给出了一部包含一些助动词的词典,其中,定义了所有的不规则形式。

5.1.1 被动句式

前面的章节中,我们讨论了助动词序列中处理被动句式的规则,不过,这仅仅解决了被动结构的一个相关问题。大多数动词都允许被动形式,其补语中大都包含名词短语。在被动形式中,通常在宾语位置上的名词短语会作为句子的第一个名词短语,而通常在主语位置的名词短语要么省略,要么放在以介词“by”开头的介词短语中。例如,下面给出的是一些主动句:

I will hide my hat in the drawer.(我要把帽子藏在抽屉里面。)
I hid my hat in the drawer.(我把帽子藏在抽屉里面了。)
I was hiding my hat in the drawer.(我那时正把帽子藏到抽屉里面。)

如果采用被动形式,这些句子可以改写为:

My hat will be hidden in the drawer.(我的帽子要被藏在抽屉里面。)
My hat was hidden in the drawer.(我的帽子被藏在抽屉里面了。)
My hat was being hidden in the drawer.(我的帽子那时正被藏到抽屉里面。)

can:	(CAT AUX MODAL + VFORM pres AGR {1s 2s 3s 1p 2p 3p} COMPFORM base)	could:	(CAT AUX MODAL + VFORM {pres past} AGR {1s 2s 3s 1p 2p 3p} COMPFORM base)
do:	(CAT AUX MODAL + VFORM pres AGR {1s 2s 1p 2p 3p} COMPFORM base)	did:	(CAT AUX MODAL + VFORM past AGR {1s 2s 3s 1p 2p 3p} COMPFORM base)
be:	(CAT AUX VFORM base ROOT be COMPFORM ing)	have:	(CAT AUX VFORM base ROOT have COMPFORM pastprt)

图 5.2 一些助动词的词条

这里的困难是被动结构中的动词短语找不到作为宾语的名词短语。解决办法之一是为每个被动形式需要用到的动词次范畴加上一条新的语法规则,即所有允许接 NP 的动词规则。给定一个语法,我们可以很容易地编写程序来自动产生这种被动式规则。在这里,需要定义新的二元中心特征 PASSGAP。仅当语法成分找不到宾语 NP 时,该值为“+”。通常,如果规则的左部没有做特别指定,该特征的默认值为“-”。语法 4.5 给出了简单的_np次范畴规则,而在新语法中,该规则将分解为下面的两条规则:

VP[-passgap] → V[_np] NP
VP[+passgap] → V[_np]

PASSGAP 特征默认值为“-”,它的值变为“+”号的唯一途径是采用被动式规则。类似地,对于那些具有词法中心语,但补足语中没有名词短语的动词短语规则来说,该特征值也是“-PASSGAP”,因为它们不可能构成被动句。我们对语法 4.5 进行扩展,使得它能够处理助动词和被动语态,图 5.3 给出了扩展之后的语法片段。这些规则扩展如上所示,不过我们还需要用每条规则中的其他变量来相应地传递特征。

图 5.4 分别给出了主动句“Jack can see the dog”和被动句“Jack was seen”的分析过程。这两个分析都采用规则 1 来分析 S,采用规则 7 来分析 VP。惟一不同的是主动句必须使用助动词的规则 2 和 NP 的规则 9,而被动句必须使用助动词的规则 5 和 NP 的规则 8。

1. S[-inv] → (NP AGR ?a) (VP [fin] AGR ?a)

2. VP → (AUX COMPFORM ?v) (VP VFORM ?v)

3. VP → AUX[be] VP[ing, +main]

4. VP → AUX[be] VP[ing, +pass]

5. VP[+pass] → AUX[be] VP[pastprt, main, +passgap]

6. VP[-passgap, +main] → V[_none]

7. VP[-passgap, +main] → V[_np] NP

8. VP[+passgap, +main] → V[_np]

9. NP → (ART AGR ?a) (N AGR ?a)

10. NP → NAME

11. NP → PRO

VP 的中心特征为: AGP 和 VFORM

NP 的中心特征为: AGR

图 5.3 助动词处理的语法片段,其中包含被动语态的处理

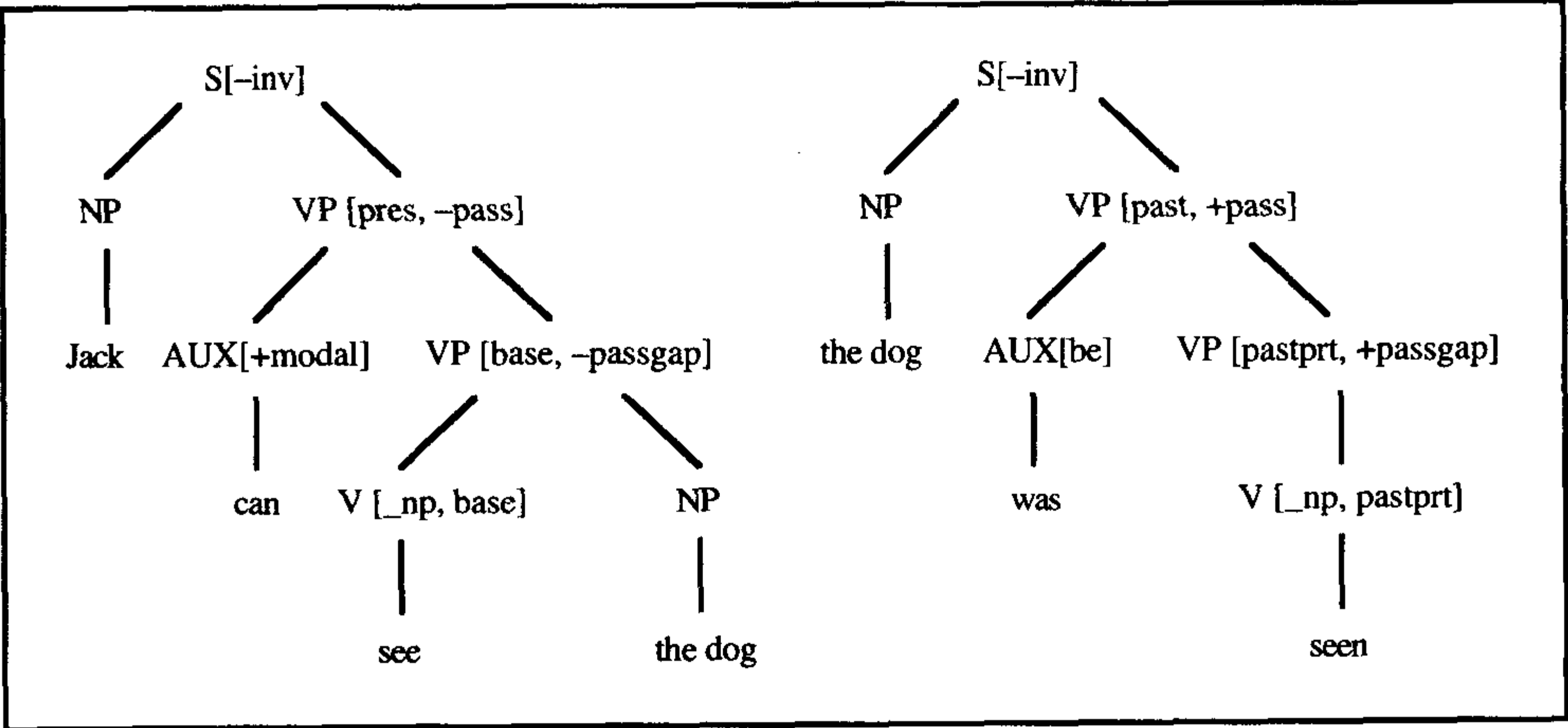


图 5.4 主动语态句和被动语态句

5.2 语言中的移位现象

许多句子结构看上去仅仅是其他结构的简单变体。在某些情况下,似乎仅仅是在局部上重新调整了一些简单词语或短语的顺序;除了明显地将基本句中的短语从特定位置上移走之外,两个句子完全一样。本节中就要研究英语中疑问句所具有的规律,并探索一些相关的技术。

首先,我们从一个例子开始看看一般疑问句的结构,并思考一下它们究竟怎样关联相应的陈述句。我们先具体地查看如下的两个例子:

- Jack is giving Sue a back rub.(Jack 正在给 Sue 做背部按摩。)
- Is Jack giving Sue a back rub?(Jack 在给 Sue 做背部按摩吗?)

He will run in the marathon next year. (他明年将参加马拉松比赛。)

Will he run in the marathon next year? (他明年会参加马拉松比赛吗?)

正如你所看到的,一般疑问句和对应的陈述句在结构上看起来几乎没有什么不一样,不同之处仅仅是主语 NP 和第一个助动词交换了位置。如果陈述句中没有助动词,那么,我们就会使用相应时态的助动词“do”,示例如下:

John went to the store. (John 去过商店。)

Did John go to the store? (John 去过商店了吗?)

Henry goes to school every day. (Henry 每天都去上学。)

Does Henry go to school every day? (Henry 每天都去上学吗?)

采用语言学中的术语,这种主语和助动词的重新排列就叫做主助倒置(subject-aux inversion)。

不太严格地说,可以认为一般疑问句就是从陈述句中派生而来的,具体地说,就是通过上述方式移动语法成分得到的,这是一个局部移位(或称有界移位)的例子。之所以认为这种移位是局部的,是因为语法成分的重组可以在有限的规则范围内准确地指定。这与特殊疑问句中的无界移位是完全不同的。在无界移位的情况下,语法成分可以任意移到离它们原始位置很远的地方。

例如,针对下面的陈述句,考虑一下相关的特殊疑问句:

The fat man will angrily put the book in the corner.

(这个胖男人生气地要把书放在那个角落里。)

如果你关心谁是行为的实施者,或许会提下面的其中一个问题:

Which man will angrily put the book in the corner?

(哪个男人生气地要把书放在那个角落里?)

Who will angrily put the book in the corner?

(谁生气地要把书放在那个角落里?)

另一方面,如果你感兴趣的是事情怎么发生的,可能会提下面的其中一个问题:

How will the fat man put the book in the corner?

(这个胖男人要怎样把书放在那个角落里?)

In what way will the fat man put the book in the corner?

(这个胖男人会采取什么样的方式把书放在那个角落里?)

如果你感兴趣的是其他方面,可能会提下面的其中一个问题:

What will the fat man angrily put in the corner?

(这个胖男人生气地把什么东西放在那个角落里?)

Where will the fat man angrily put the book?

(这个胖男人生气地要把书放在哪里?)

In what corner will the fat man angrily put the book?

(这个胖男人要生气地把书放在哪个角落里?)

What will the fat man angrily put the book in?

(这个胖男人生气地要把书放在什么地方?)

每个疑问句的原始陈述句形式均相同,不同之处仅仅是删去了要提问的部分,并替换为句首的疑问短语。另外,如果要提问的部分不是做主语 NP,那么,特殊疑问句中的主语和助动词显然要倒置,这和一般疑问句相同。甚至在没有助动词的句子中,特殊疑问句与一般疑问句仍然保持这种相似性。下面的两个例子就增加了助动词“do”:

I found a bookcase.(我找到了一个书架。)

Did I find a bookcase?(我找到了一个书架吗?)

What *did* I find?(我找到了什么呢?)

因此,你或许可以将一般疑问句语法中的很多规律再应用到特殊疑问句中。不过,在如何处理语法成分位置缺失这方面,仍然存在一个严重的问题。先让我们看看下面例句中用斜体字标出的动词短语(VP):

What will the fat man *angrily put in the corner*?

尽管这是一个语法可以接受的句子,但是“angrily put in the corner”看上去却不是一个可以接受的动词短语,因为你不能接受像“* I angrily put in the corner.”(错误的句子:* 我生气地放在角落里)这样的句子。只有在特殊疑问句中,这样的动词短语才是可接受的。进一步来说,只有当特殊疑问成分采用了正确的形式时,这种动词短语才可接受。例如,“What will the fat man angrily put in the corner?”是可以接受的,而“* Where will the fat man angrily put in the corner?”却不可接受。

如果要为特殊疑问句中的动词短语构造专门的语法,那么就需要为每种形式的动词短语和缺失成分都构造单独的语法。这样,语法的规模会大大地膨胀。

本章简要叙述处理这种现象的一些技术方法。通常,它们都使用相同类型的方法。句子中缺失成分的位置,我们称之为缺位(gap),而要移入的成分称为填充成分(filler)。当句子中存在相应的缺位填充成分时,随后的技术都涉及到如何填充缺位的问题。这样,对句子“What will the fat man angrily put in the corner?”中的动词短语进行分析时,可以将这个短语看成是“angrily put what in the corner”(生气地放什么东西在那个角落)。而分析句子“What will the fat man angrily put the book in?”(这个胖男人生气地要把书放在什么地方呢?)中的动词短语,可以看成是分析“angrily put the book in what”(生气地把书放在什么地方)。

我们还有进一步的论据可以证明这种分析的正确性。具体地说,可以假定疑问词填充缺位,并对疑问句进行形式正确性的所有测试,比如主谓一致性、代词的格(即 who 或者 whom)和动词的及物性。例如,处理动词及物性的方法,这一点你已经看到了。疑问句“What did you put in the cupboard?”(你把什么放到碗橱里去了?)是可接受的,其中“put”是及物动词,它需要接宾语。而这个宾语是个被疑问词填充的缺位,它满足及物性约束。与此对应的是,明确有宾语填充的句子反而是不可接受的,例句如下:

* What did you put the bottle in the cupboard?

(错误的句子:* 你把瓶子什么放到碗橱里去了?)

这个句子不能被语法接受。一个句子只要动词“put”带两个宾语都是不可接受的,这一点和当前句子的错误是一样的。实际上,它还等同于下面这个句子形式:

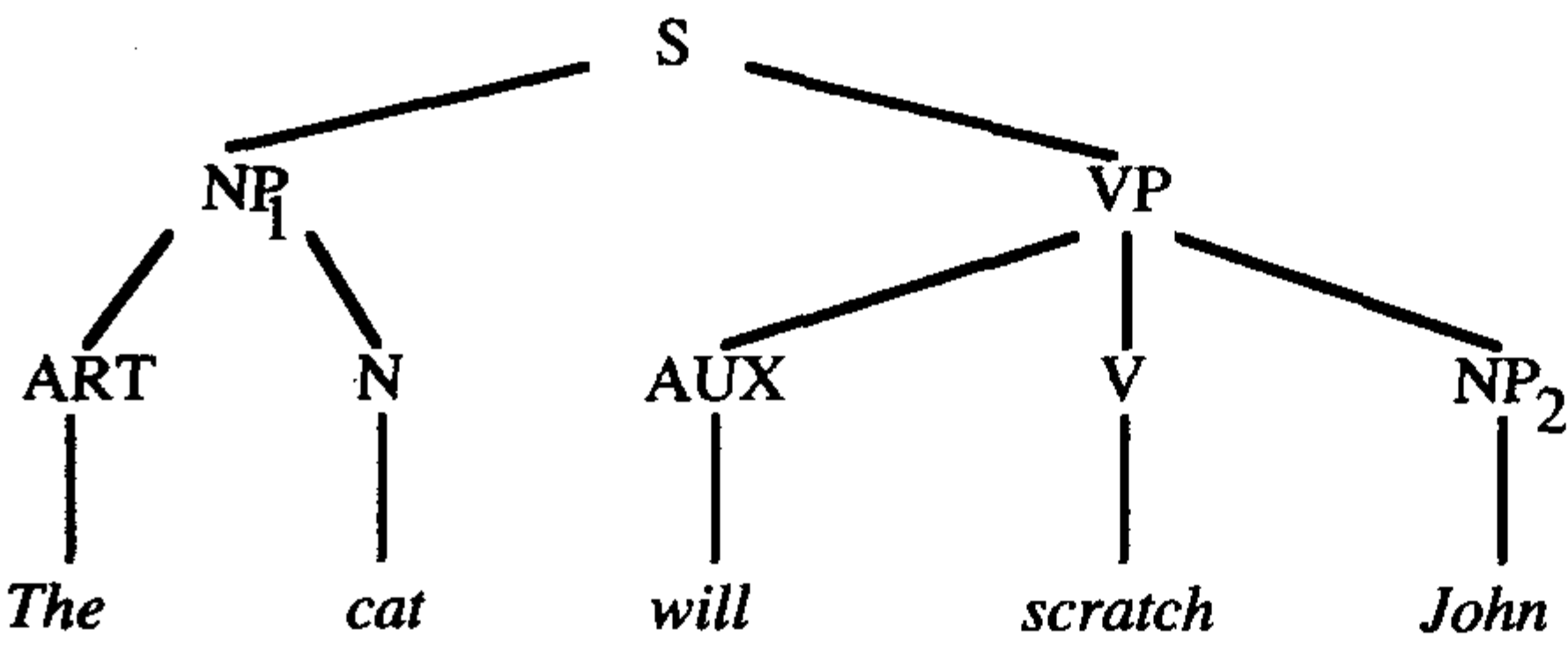
* You put what the bottle in the cupboard?
(错误的句子: * 你把瓶子什么放到碗橱里去了?)

因此,只有假定开始的疑问词可以在标准的宾语位置上使用,并满足约束时,标准的及物性测试方法才会生效。

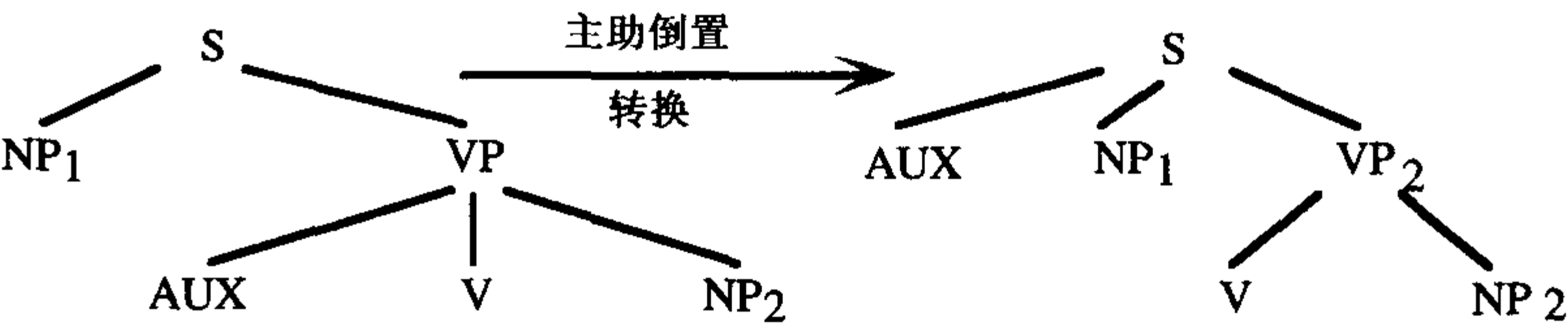
语法成分可以从一个位置移动到另一个位置,基于这种直觉,研究者建立了许多语言学理论。当你进一步探索这些技术的时候,会发现我们完全可以对它们进行泛化推广,从而大大简化语法的构建过程,因此,泛化推广意义重大。转换语法(见框 5.1)就是基于这种模型的语法体系。上下文无关文法产生一个基本的句子;然后经过一系列的转换,句子成分的移动,我们可以将初始得到的语法分析树转换成另一棵不同的树。扩充转移网络提供了一套新的处理机制,它采用更加高效的计算方式来刻画大部分的行为特征。扩充转移网络引入了一个叫保留表(hold list)的数据结构,保留表允许存储语法成分,并在以后的句法分析过程被新的边使用,使用该成分的边叫做虚边。这种计算机制在过去很长一段时期内一直占据着主导地位。不过,在 20 世纪 80 年代早期,语言学方面研究出了新的技术,这对当前的计算体系影响重大。

框 5.1 语言学中的移位现象

术语移位是在转换语法(TG, transformational grammar)中提出来的。TG 有两种不同层面的结构表示方法:表层结构(surface structure)和深层结构(deep structure)。其中,表层结构对应于实际的句子结构。(上下文无关文法)可以生成句子的深层结构,而转换集则将深层结构映射到表层结构。举例来说,句子“Will the cat scratch John?”(猫会抓伤 John 吗?)的深层结构为:



然后,可以从这个深层结构中生成得到一般疑问句,这个转换过程的示意图如下所示:



通过这个转换,句子的表层形式为:

```
graph TD
    S --> AUX[AUX]
    S --> NP1[NP]
    S --> VP[VP]
    AUX --> Will[Will]
    NP1 --> ART[ART]
    NP1 --> N[N]
    ART --> the[the]
    N --> cat[cat]
    VP --> V[V]
    VP --> NP2[NP]
    V --> scratch[scratch]
    NP2 --> John[John]
```

在早期的转换文法中(Chomsky, 1965),转换可以对树进行很多的操作,包括移动成分、添加成分以及删除成分。除了主(语)助(动词)倒置之外,我们还可以考虑的倒置有被动语态、特殊疑问句和内嵌句等。当代的转换语法继承了早期的转换语法,但是不再采用同样的方式进行转换。在管辖约束理论(GB theory, government-binding theory)中,单一的转换规则 Move- α 允许任何成分移动到任何地方! 其核心是实现移位约束,从而防止非法表层结构的产生。

第一种技术是引入斜杠(slash)类。斜杠类指的是形式为“X/Y”的复杂非终结符,它表示的是 X 类型的语法成分,而且该成分缺少了一个 Y 类型的子成分。给定一个上下文无关文法,我们可以根据一个简单的算法,自动生成复杂成分的新规则。掌握了这一点,语法编撰人员在表示无界移位约束时就有一套便利的语法标记体系。遗憾的是,最终语法的规模会大大超过原始的语法。更好的方法一般是采用特征体系,语法成分存储在专门的 GAP 特征之中,GAP 特征会从一个成分传到能接受 GAP 特征的子成分。在分析过程中,GAP 特征为 NP 的语法成分 S 简写为 S/NP。这样,最终的系统对语法规规模就不需要扩充很多,但是,移位现象的处理效果很好。接下来的章节将详细地讨论上下文无关文法的两种基于特征的方法以及扩充转移网络(ATN)中的保留表技术。

框 5.2 不同的移位类型

本章主要讨论特殊疑问句,而我们会广泛而深入地研究疑问词的移位问题。不过,这里探讨的技术也可以用于处理其他形式的移位。在此,我们列举一些最普遍的移位形式,这些在语言学的文献中都进行过讨论。细节请参阅语言学的相关教材,比如 Baker(1989)。

wh-移位(wh-movement)——将疑问词移到句子的前端,并形成一个特殊疑问句。

主题化(topicalization)——将一个成分移到句首,以示强调,如:

I never liked this picture.(我从不喜欢这张图片。)

This picture, I never liked.(这张图片,我从不喜欢。)

副词前置(adverb preposing)——将副词移到句首,如:

I will see you tomorrow.(我明天要去看你。)

Tomorrow, I will see you.(明天,我要去看你。)

外置转换(extraposition)——将某个 NP 补足语移到句末,如:

A book discussing evolution was written.(写了一本讨论进化论的书。)

A book was written discussing evolution.(写了一本书讨论进化论。)

当你考虑移位的处理策略时,需要记住一点:不是任何位置上的语法成分都可以移到前面,并形成一个疑问句。例如:

The man who was holding the two balloons will put the box in the corner.
(那个曾拿着两个气球的男人要把盒子放到角落里。)

这是一个形式正确的句子,但是你不能提下面这样的问题,其中破折号表示缺位:

* What will the man who was holding — put the box in the corner?
(* 那个过去拿着什么的男人要把盒子放到角落里?)

这个例子中,通用的移位约束与关系从句不符。

5.3 上下文无关文法中的疑问句处理

我们的目标是最低限度地扩展上下文无关文法,并且保证它能够处理疑问句。换句话说,我们要尽可能再利用原来的语法。对于一般疑问句来说,这很容易就能实现。你可以引入一条规则来扩展语法 4.7,这条规则允许在第一个 NP 前面加上助动词,这样就可以处理大多数的情况。规则如下:

$S [+inv] \rightarrow (AUX\ AGR\ ?a\ SUBCAT\ ?v)\ (NP\ AGR\ ?a)\ (VP\ VFORM\ ?v)$

它增强了助动词 AUX 与主语 NP 间的主谓一致性约束,保证了助动词 AUX 后的动词短语能有正确的 VFORM。一般疑问句的处理仅仅需要这条规则,而且,所有原来用来分析断言句的语法均可以直接处理一般疑问句。

正如 5.2 节中提到的那样,我们引入专门的 GAP 特征来处理特殊疑问句。这个特征会一直从母成分传递到子成分,直到在句子中找到相应的缺位为止。在这个位置上,我们可以在没有输入的情况下直接构造出相应的语法成分。这可以通过引入右部为空的附属规则来实现,比如,可以有下面的这条规则:

$(NP\ GAP\ ((CAT\ NP)\ (AGR\ ?a))\ AGR\ ?a) \rightarrow \epsilon$

即,在寻找名词短语时,如果发现当前成分的 GAP 特征值为 NP,根据这条规则,不需要任何输入就可以直接建立一个 NP。此外,这个空 NP 的 AGR 特征就设为当前成分的 AGR 特征。需要注意的是,母成分的 GAP 特征值是另一个特征结构。为了帮助你辨别这些复杂特征值的结构,我们用小号的字体来表示充当特征值的特征结构。

现在,可以编写一套能正确传递 GAP 特征的语法了。不过这会很枯燥乏味,我们并没有利用一些可以阻止 GAP 特征进一步传播的规律。具体地说,GAP 特征好像存在两条通用的传播方式,划分的标准主要是看中心成分究竟是词类还是非词类。如果中心成分是非词类,那么 GAP 特征会从母成分传递到中心成分,而不传递给任何其他的子成分。比如,下面这条带 GAP 特征的典型 S 规则:

$(S\ GAP\ ?g) \rightarrow (NP\ GAP\ -)\ (VP\ GAP\ ?g)$

GAP 可以位于中心子成分 VP 中,但是绝不会在 NP 主语中。而对于中心成分为词类的规则来说,缺位可以移到任何一个非词汇子成分当中。例如,下面这条补足语为 $_np_vp:inf$ 的动词规则:

$VP \rightarrow V[_np_vp:inf]\ NP\ PP$

这条规则会导出下面两条和缺位有关的规则:

$$\begin{aligned} (VP \text{ GAP } ?g) &\rightarrow V[_{np_vp:inf}] (NP \text{ GAP } ?g) (PP \text{ GAP } -) \\ (VP \text{ GAP } ?g) &\rightarrow V[_{np_vp:inf}] (NP \text{ GAP } -) (PP \text{ GAP } ?g) \end{aligned}$$

换句话说,GAP 要么在 NP 中,要么在 PP 中,但是不会同时出现在两者之中。除了一个子成分的 GAP 特征设为“-”之外,其他所有成分都设置了 GAP 特征,这样可以保证缺位只能用在—个地方。

图 5.5 给出了一个将 GAP 特征自动添加到语法中的算法。注意,它不会更改任何一条已经明确设置了 GAP 特征的规则,这样,语法设计者就可以引入那些不遵循算法的约定。比如,主助倒置规则不允许缺位传播到 NP 主语当中。

对每一条中心成分为 H_i 的规则 $Y \rightarrow X_1 \dots H_i \dots X_n$, 算法如下:

1. 如果该规则已经给出了某个成分的 GAP 特征,则略过。
2. 如果中心成分 H_i 不是词类,则在中心成分和右部的母成分中加上 GAP 特征,在其他子成分中加上“- GAP”,产生如下形式的新规则:

$$(Y \text{ GAP } ?g) \rightarrow (X_1 \text{ GAP } -) \dots (H_i \text{ GAP } ?g) \dots (X_n \text{ GAP } -)$$

3. 如果中心成分 H_i 是词类,则对于每个非词类子成分 X_j ,增加一条如下形式的规则:

$$(Y \text{ GAP } ?g) \rightarrow (X_1 \text{ GAP } -) \dots (X_j \text{ GAP } ?g) \dots (X_n \text{ GAP } -)$$

图 5.5 往语法中添加 GAP 特征的算法

通过这个过程,我们就可以创建一个能够处理缺位的新语法。现在,剩下的事情仅仅是分析缺位填充成分的出处。在特殊疑问句中,填充成分一般都是位于句首的名词短语或者介词短语处。在此,我们引入一个新的特征 WH。WH 特征表示的是一类引导问句的短语,通过—个特征我们就能识别出填充成分。WH 特征主要以“who”,“what”,“when”,“where”,“why”和“how”(如“how many”和“how carefully”中的“how”)之类的词为标志。这些词语分别属于几个不同的语法类,这通过它们所替代的短语类型就可以看出来。具体地说,“who”,“whom”和“what”可以作为代词出现,用于指定简单的名词短语,例如:

Who ate the pizza? (谁吃了比萨?)

What did you put the book in? (你把书放到什么里面了?)

“what”和“which”可以作为名词短语中的限定词出现,如:

What book did he steal? (他偷了什么样的书?)

像“where”和“when”这样的词可以作为介词短语出现,如:

Where did you put the book? (你把书放到哪里啦?)

“how”可以作为形容词和副词短语的副词性修饰语,如:

How quickly did he run? (他跑得有多快?)

最后,“whose”可以充当物主代词:

Whose book did you find? (你找到的是谁的书?)

疑问词(wh-word)也可以在句子中扮演不同的角色。前面的所有例子表明这些词语可以用来引导特殊疑问句,特殊疑问句用 WH 特征值 Q 表示。实际上,它们中的一部分词也可以用来引导关系从句,此时 WH 特征值用 R 表示,这一点会在 5.4 节加以讨论。为了使 WH 特征起到相应的作用,语法应该满足下面的约束:如果一个短语包含具有 WH 特征的子短语,那么这个较大的短语也具有同样的 WH 特征。这样,就可以在疑问句中使用一些复杂的短语,该短语中可以包含带疑问词的子成分。例句如下:

In what store did you buy the picture? (你在哪家商店买的图片?)

名词短语“what store”的 WH 特征值为 Q(因为限定词“what”的 WH 特征值为 Q),那么,句首的介词短语(PP)“in what store”的 WH 特征值也就为 Q,所以,上面给出的是个合法的疑问句。通过这个约束,最终语法成分 S 的 WH 特征值也将会是 Q,这个值表明该句是一个特殊疑问句。图 5.6 给出了一些疑问词的词条。注意,我们引入的新词类 QDET 指的是引导疑问词的限定词,PP-WRD 用来表示具有介词短语功能的词语,比如“when”。语法 5.7 给出了一些 NP 和 PP 规则,这些规则可以正确地处理 WH 特征。

what:	(CAT PRO WH Q AGR {3s 3p}))	when:	(CAT PP-WRD WH {Q R} PFORM TIME)
what:	(CAT QDET WH Q AGR {3s 3p}))	who:	(CAT PRO WH {Q R} AGR {3s 3p}))
which:	(CAT QDET WH Q AGR {3s 3p}))	where:	(CAT PP-WRD WH {Q R} PFORM {LOC MOT}))
which:	(CAT PRO WH R AGR {3s 3p}))	whose:	(CAT PRO WH {Q R} POSS + AGR {3s 3p}))

图 5.6 一些疑问词的词典

1.	(NP POSS ?p WH ?w) → (PRO POSS ?p WH ?w)
2.	(NP WH ?w) → (DET WH ?w AGR ?a) (CNP AGR ?a)
3.	CNP → N
4.	CNP → ADJ N
5.	DET → ART
6.	(DET WH ?w) → (NP[+POSS] WH ?a)
7.	(DET WH ?w) → (QDET WH ?w)
8.	(PP WH ?w) → P (NP WH ?w)
9.	(PP WH ?w) → (PP-WRD WH ?w)
	NP, DET 和 CNP 的中心特征为: AGR
	PP 的中心特征为: PFORM

语法 5.7 处理疑问词的简单 NP 和 PP 语法

我们定义了 WH 特征,给出了能处理 GAP 特征的语法修改机制。基于这两项研究,只需要在语法中增加两条新的规则就可以处理大多数的特殊疑问句。下面就是两条分别基于 NP 和 PP 的疑问句处理规则:

$S \rightarrow (NP[Q, -gap] \text{ AGR } ?a) (S[+inv] \text{ GAP } (NP \text{ AGR } ?a))$
 $S \rightarrow (PP[Q, -gap] \text{ PFORM } ?p) (S[+inv] \text{ GAP } (PP \text{ PFORM } ?p))$

这两条规则都将 GAP 的特征值设置为首个 WH 成分的副本,这样,它就可以用来填充 S 成分中的缺位。因为 S 成分必须具有 + INV 特征,所以它还必须涵盖主助倒置规则。语法 5.7 中给定了 NP 和 PP 的规则,在此前提下,语法 5.8 给出了疑问句处理的 S 规则和 VP 规则。在增加 GAP 特征的过程之后,语法 5.9 列出了所有自动派生的新规则。规则 11、规则 12 以及规则 13 已经指定了 GAP 特征,所以,该算法没有改变这三条规则。

10. $(S[-inv] \text{ WH } ?w) \rightarrow$
 $(NP \text{ WH } ?w \text{ AGR } ?a)$
 $(VP[fin] \text{ AGR } ?a)$
 11. $(S[+inv] \text{ WH } ?w \text{ GAP } ?g) \rightarrow$
 $(AUX \text{ COMPFORM } ?s \text{ AGR } ?a)$
 $(NP \text{ WH } ?w \text{ AGR } ?a \text{ GAP } -)$
 $(VP \text{ VFORM } ?s \text{ GAP } ?g)$
 12. $S \rightarrow (NP[Q, -gap] \text{ AGR } ?a) (S[+inv] \text{ GAP } (NP \text{ AGR } ?a))$
 13. $S \rightarrow (PP[Q, -gap] \text{ PFORM } ?p) (S[+inv] \text{ GAP } (PP \text{ PFORM } ?p))$
 14. $VP \rightarrow (AUX \text{ COMPFORM } ?s) (VP \text{ VFORM } ?s)$
 15. $VP \rightarrow V[_{none}]$
 16. $VP \rightarrow V[_{np}] NP$
 17. $VP \rightarrow V[_{vp:inf}] VP[inf]$
 18. $VP \rightarrow V[_{np_vp:inf}] NP VP[inf]$
 19. $VP[inf] \rightarrow TO VP[base]$
 20. $VP \rightarrow V[_{np_pp:loc}] NP PP[loc]$
- S, VP 的中心特征为: VFORM, AGR

语法 5.8 处理特殊疑问句的未扩展 S 语法

5.3.1 带缺位的句法分析

允许缺位的语法给句法分析算法造成了一些新的困难。尤其是存在右部为空的规则,如:

$(NP \text{ AGR } ?a \text{ GAP } (NP \text{ AGR } ?a)) \rightarrow \epsilon$

这种规则可能会导致一些问题,因为它们允许任何位置出现一个空的 NP 成分。例如,在自底向上的分析策略中,这条规则可以在任意位置创建(或者是在任意位置多次创建)没有任何输入的 NP。自顶向下的分析策略稍微好些,因为,只有在预测到需要缺位时才会使用这条规则。不过,我们不使用这些规则,只需要对有向边扩展算法进行适当修改就能自动地处理缺位。这项技术对于任何一种句法分析策略都是有效的。

10. (S[-inv] WH ?w GAP ?g) →
(NP WH ?w AGR ?a)
(VP[fin] AGR ?a GAP ?g)
11. (S[+inv] WH ?w GAP ?g) →
(AUX COMPFORM ?s AGR ?a)
(NP WH ?w AGR ?a GAP -)
(VP VFORM ?s GAP ?g)
12. S → (NP[Q,-gap] AGR ?a) (S[+inv] GAP (NP AGR ?a))
13. S → (PP[Q,-gap] PFORM ?p) (S[+inv] GAP (PP PFORM ?p))
14. (VP GAP ?g) → (AUX COMPFORM ?s) (VP VFORM ?s GAP ?g)
15. VP → V[_none]
16. (VP GAP ?g) → V[_np] (NP GAP ?g)
17. (VP GAP ?g) → V[_vp:inf] (VP[inf] GAP ?g)
18. (VP GAP ?g) → V[_np_vp:inf] (NP GAP ?g) (VP[inf] GAP -)
- 18'. (VP GAP ?g) → V[_np_vp:inf] (NP GAP -) (VP[inf] GAP ?g)
19. (VP[inf] GAP ?g) → TO (VP[base] GAP ?g)
20. (VP GAP ?g) → V[_np_pp:loc] (NP GAP ?g) (PP[loc] GAP -)
- 20'. ((VP GAP ?g) → V[_np_pp:loc] (NP GAP -) (PP[loc] GAP ?g))

S,VP的中心特征为: VFORM, AGR

语法 5.9 带 GAP 特征的特殊疑问句的 S 语法

现在,我们来考虑一下究竟什么样的扩展是必需的。当语法成分有一个与它自己相匹配的 GAP 特征时,则该成分就必须由空成分来生成。这意味着下一个成分是缺位的有向边,可以直接进行扩展。例如,如果句法分析器给出了下面的有向边,让我们看看会发生什么:

(VP GAP (NP AGR 3s))
→ V[_np_pp:loc] ◦ (NP GAP (NP AGR 3s)) PP[LOC]

所需的下一个成分是 NP,但它也有一个 GAP 特征,该特征的值也是 NP。因此,这个成分必须为空。分析器在 chart 图中添加如下成分:

(NP AGR 3s EMPTY +)

这样,chart 图便能对上面的有向边进行扩展,使之产生新的有向边。新边如下所示:

(VP GAP (NP AGR 3s))
→ V[_np_pp:loc] (NP GAP (NP AGR 3s)) ◦ PP[LOC]

图 5.10 给出了该算法更准确的描述。

现在,我们试着采用自底向上的策略对“Which dogs did he see?”进行句法分析。在这里,只考虑那些对最后分析有贡献的规则。根据语法 5.7 中的规则 7,语法成分 QDET1(which)生成成分 DET1;采用规则 3,成分 N1(dogs)生成成分 CNP1;随后,通过规则 2,CNP1 与 DET1 结合生成如下形式的 NP 成分:

NP1: (NP AGR 3p
WH Q
1 QDET1
2 CNP1)

框 5.3 移位约束

缺位可能出现在哪些位置上? 在语言学中, 负责管理缺位出现位置的原则称为孤岛约束(island constraint)。这一术语用来比喻成分的移位约束。孤岛指的是不能从中再抽出子成分的成分(就像一个不能离开孤岛的人)。这里给出的是一些已经提出来的约束关系。

A 上 A 约束(The A over A Constraint): 即类别为 A 的成分中不能再抽出一个类型为 A 的子成分。这意味着不存在包含 NP 缺位的 NP, 也不存在包含 PP 缺位的 PP, 依次类推。我们根据这条约束不接受 NP/NP, PP/PP 等形式的非空成分。因此, 不接受下面的句子:

* What book_i did you meet the author of —_i?

复杂 NP 约束(Complex-NP Constraint): 不能从关系从句或名词补语中移出任何成分。这个约束使得语法不能接受下面的句子:

* To whom_i did the man who gave the book —_i laughed?

其中, 介词短语“to whom”原本是关系从句“who gave the book to whom”的一部分(这一点和句子“The man who gave the book to John laughed”一样)。

句子主语约束(Sentential Subject Constraint): 我们不能从作为句子主语的成分中移出任何成分。当主语是 NP 时, 该约束与其他约束有互相重叠的地方。不过, 非 NP 主语也有可能, 如句子“For me to learn these constraints is impossible”(对我来讲, 学习这些约束是不可能的)。这个约束排除了下面这样一个疑问句:

* What_i is for me to learn —_i impossible?

Wh-孤岛约束(Wh-Island Constraint): 在特殊疑问词做补语的嵌入句中, 不能移出任何一个成分。比如, “Did they wonder whether I took the book?”(他们怀疑我是否拿了这本书吗?)是一个可接受的句子, 但是, 不能问:

* What_i did they wonder whether I took —_i?

并列结构约束(Coordinate Structure Constraint): 不能从并列结构中移出成分。例如, “Did you see John and Sam?”(你看见了 John 和 Sam 吗?)是一个可接受的句子, 但是不能问:

* Who_i did you see John and —_i?

注意, 这些约束不限于特殊疑问句, 它们可用于所有形式的移位之中。比如, 也可以对主题化(topicalization)和副词前置(adverb preposing)进行约束。

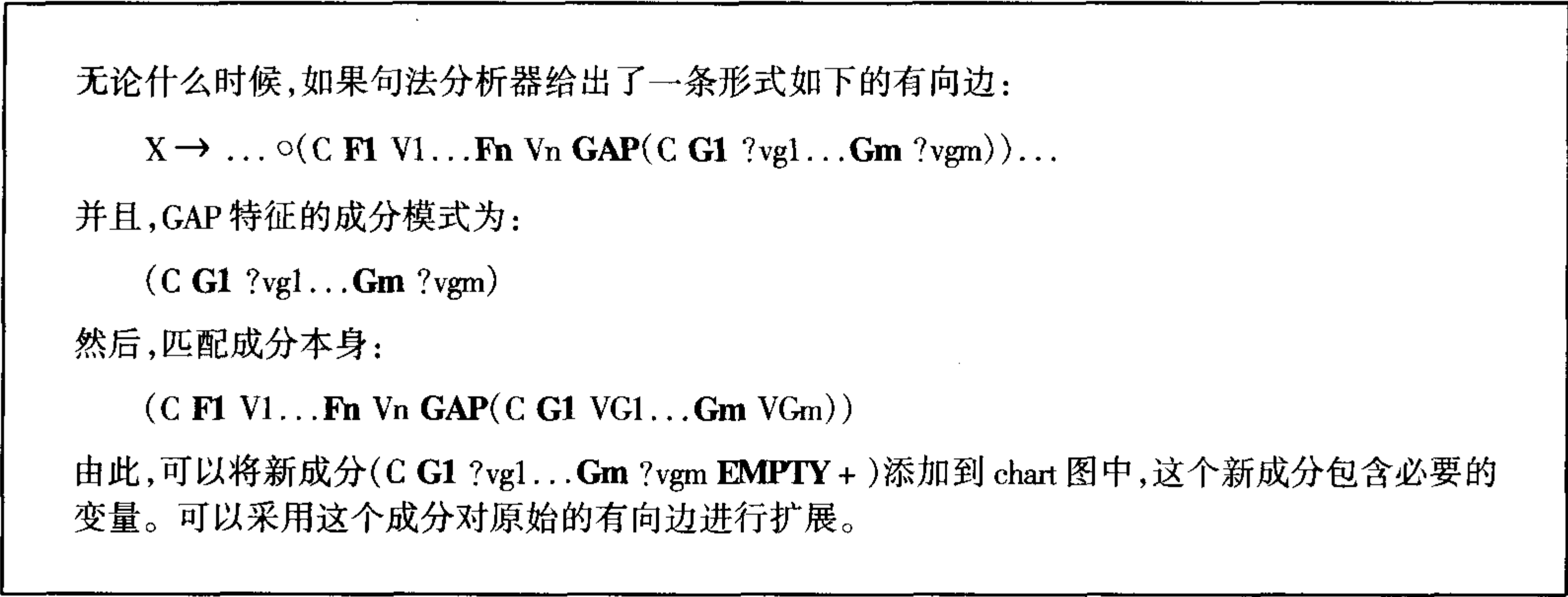


图 5.10 根据需要添加空成分的算法

基于语法 5.8 中的规则 12,这个 NP 会引入一条有向边。下一个词“did”是 AUX 成分,它引入一条基于规则 11 的有向边。接下来,词语“he”是一个 PRO,采用规则 1,它可以创建名词短语 NP2。而且,我们还可以根据规则 1 扩展这条有向边。图 5.11 给出了这一阶段的 chart 图。

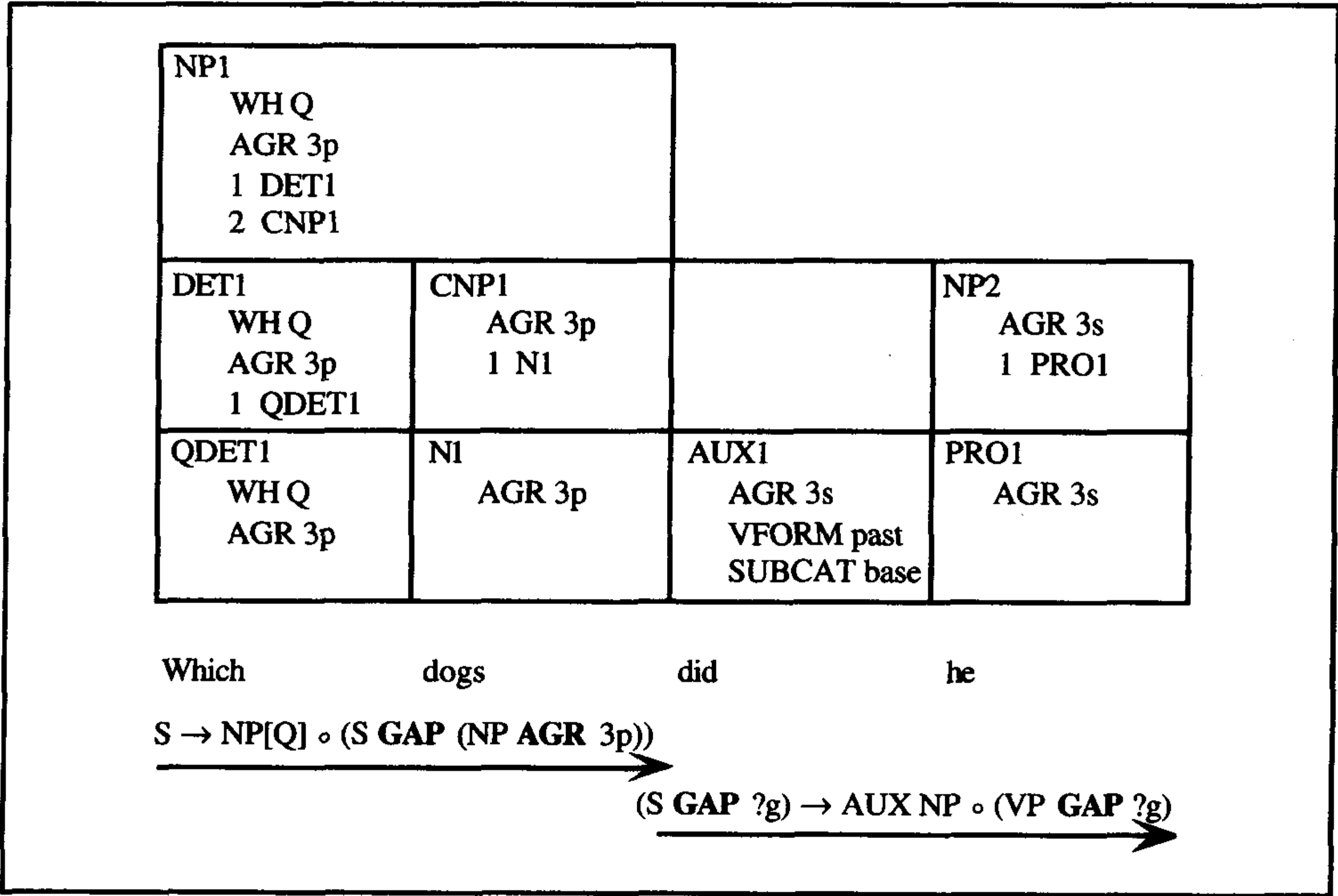


图 5.11 词语“he”分析之后的 chart 图

根据词语“see”,可以引入动词 V1,V1 扩展规则 16。这样就能添加一条边,该边标记为:
(VP GAP ?g) → V[_np] ◦ (NP GAP ?g)

GAP 值可以匹配所需的 NP(因为它是不受约束的),所以,这时可以在 chart 图中添加一个空的 NP,其形式为:

EMPTY-NP1: (NP AGR ?a GAP (NP AGR ?a)) EMPTY +)

然后,这个成分可以扩展 VP 有向边,产生如下成分:

VP1: (VP VFORM inf
GAP (NP AGR ?a)
1 V1
2 EMPTY-NP1)

现在,VP1 可以扩展基于规则 11 生成的有向边,这样就可以生成 S 结构,即:

S1: (S GAP (NP AGR ?a)
INV +
1 AUX1
2 NP2
3 VP1)

最后,采用规则 12,NP1 和 S1 相结合,产生一个 S 成分。图 5.12 给出了最终的 chart 图。

S2 VFORM past 1 NP1 2 S1				
NP1 WH Q AGR 3p 1 DET1 2 CNP1		S1 INV+ GAP (NP AGR 3p) VFORM past 1 AUX1 2 NP2 3 VP1		
DET1 WH Q AGR 3p 1 QDET1	CNP1 AGR 3p 1 N1		NP2 AGR 3s 1 PRO1	VP1 VFORM inf GAP (NP AGR 3p) 1 V1 2 EMPTY-NP1
QDET1 WH Q AGR 3p	N1 AGR 3p	AUX1 AGR 3s VFORM past SUBCAT base	PRO1 AGR 3s	
Which	dogs	did	he	see

图 5.12 “Which dogs did he see?”的最终 chart 图

因此,通过适当地使用 WH 特征和 GAP 特征,并将句法分析器进行扩展,使之能够根据需要添加能填充缺位的空成分,我们仅仅需要三条新规则就可以将表示陈述句的原始语法扩展到一般疑问句和特殊疑问句。这个方法成功与否主要是看我们是否能对 GAP 特征的传播进行正确有效的约束。比如,我们有一条规则要求缺位必须由母成分正确地传递到其中一个子成分之中,假如没有施加这条约束规则,语法就会接受很多非法的句子。当然,这条约束规则也存在与实际相冲突的地方,这一点我们会在习题 6 中进行探讨。

采用自底向上的分析策略,句法分析器会添加许多空成分,而且 chart 图会添加许多带有缺位的常量。其中,绝大部分都不可能在句子的任何一个合法分析中使用。自顶向下的策略可以大大减少空成分的数量,最终,我们能得到一个相当小的 chart 图。

◦ 5.4 关系从句

这一节将探讨关系从句的结构。我们可以采用一条规则引入关系从句,该规则用到了新的语法类别 REL。规则如下:

$CNP \rightarrow CNP \text{ REL}$

大多数类型的关系从句都可以采用现有的疑问句语法来处理。关系从句和疑问句均涉及到相似的结构:疑问词作为连接词,后面紧接着一个带缺位的 S 结构。这两者的主要区别是,关系从句中只允许使用某些特定的疑问词(“who”, “whom”, “which”, “when”, “where” 和 “whose”),而且 S 结构不能倒置。我们用 WH 特征值 R 表示这类可以用于关系从句的疑问词。图 5.6 给出了其中一些词的词条。

根据这些分析,处理关系从句有下面这些规则:

$REL \rightarrow (NP \text{ WH } R \text{ AGR } ?a) (S[-inv, fin] \text{ GAP } (NP \text{ AGR } ?a))$

$REL \rightarrow (PP \text{ WH } R \text{ PFORM } ?p) (S[-inv, fin] \text{ GAP } (PP \text{ PFORM } ?p))$

上面这一规则能处理的关系从句有多种形式,例如:

The man *who we saw at the store* (那个我们在商店看到的男人)

The exam *in which you found the error* (那次你发现错误的考试)

The man *whose book you stole* (那个被你偷了书的男人)

因为缺位不能传递到句子主语中,所以需要制订一条新规则来接受以疑问词充当主语的名词短语,比如“The man who read the paper”。能处理这种问题的规则如下所示:

$REL \rightarrow NP[R]VP[fin]$

另外,还有一类很普遍的关系从句,其中包含了“that”,“that”后面跟着 S,在 S 中有一个 NP 缺位,或者有一个 VP。例如,“The man that we saw at the party”(上次我们在派对上见过的那个男人)和“The man that read the paper”(那个在读报的男人)。如果允许“that”作为 WH 属性为 R 的关系代词,那么,上面的规则也可以处理这些情况。否则,需要为“that”另外增加规则。

最后,还有一类关系从句并不以相应的疑问短语开头,不过其他方面都像一个普通的关系从句。例如:

The paper *John read* (John 读的报纸)

The damage *caused by the storm* (风暴带来的损失)

The issue *creating the argument* (引起争论的事情)

上面给出的最后两个例子涉及到的是我们经常说的简略关系从句。需要两条附加的规则处理它们,该规则认为关系从句可以是一个有限的、具有 NP 缺位或者动词短语为分词形式的 S。这一规则如下:

$REL \rightarrow (S[fin] \text{ GAP } (NP \text{ AGR } ?a))$

$REL \rightarrow (VP \text{ VFORM } \{ing \text{ pastprt}\})$

注意,这里的缺位有两个主要的用途:一个用来处理疑问句,另一个用来处理关系从句。基于这一点,必须仔细判断两者可能存在的相互作用。如果关系从句出现在疑问句之中又如何呢?当然,这种情况很可能出现,例如下面的句子:

Which dog₁ did the man who₂ we saw —₂ holding the bone feed —₁?

(那个我们看到手里拿着骨头的男人喂的是哪只狗呢?)

框 5.4 广义短语结构语法的特征传播

广义短语结构语法(GPSG, generalized phrase structure grammar)(Gazdar 等, 1985)依据一组通用的原则, 对所有的特征传播进行形式化表示。GAP 特征对应于 GPSG 中的 SLASH 特征。他们声称, 处理 SLASH 特征的时候, 除了其他特征都需要的处理方式外, 不需要为之引入新的机制。具体地说, SLASH 特征符合两条通用的 GPSG 原则。一条是中心特征原则, 该原则要求中心子成分和它的母成分之间共享所有的中心特征, 这一点, 我们在前面的章节已经讨论过了。另一条是尾特征原则(foot feature principle), 要求无论尾特征出现在哪一个子成分中, 都必须与母成分进行共享。SLASH 特征既是中心特征, 又是尾特征, 所以, 我们会从这两个方面对 SLASH 特征的传播进行约束。

GPSG 采用元规则(meta-rule)从基本的语法中推导出另外的规则。元规则只能用于以词汇为中心的规则中。一条元规则采用初始的规则来说明缺位, 比如:

$$VP \rightarrow V NP$$

该规则能够产生一条新的规则:

$$VP/NP \rightarrow V NP[+NULL]$$

这里, “VP/NP”是一个 SLASH 特征为 NP 的 VP, 同时特征“+NULL”表示的是该成分“音韵为空”。也就是说, 在句子中没有被描述或者不会出现。

GPSG 元规则也可以用于处理其他现象。例如, 一条元规则可以从主动句的规则集合中自动产生相应的被动句规则。另外, 元规则还可以从非倒置的 S 规则中产生出主助倒置的规则。元规则在很大程度上实现的是转换功能。不过, 同一时间只能采用一条规则来操作, 而且该规则还必须有一个词法中心成分。否则, 这些转换就会在任意的语法树上操作。被动的元规则在形式上类似于:

$$VP \rightarrow V[TRANSITIVE]NP X \Rightarrow$$

$$VP[PASSGAP] \rightarrow V X$$

在这里, 符号 X 表示任何一个成分序列。因此, 这条元规则说的是如果存在一条动词短语规则, 该规则中依次包含了及物动词、名词短语, 或者其他可能的某些成分, 那么, 这时就相应地存在另一条被动式 VP 规则。该规则除了 NP 外, 动词和其他成分均相同。只要变量 X 可以匹配的符号数目存在上限, 那么, 通过所有可能的元规则对语法进行扩展, 你就可以看到扩展语法所描述的语言仍然是上下文无关的。

其中, 缺位用破折号表示, 数字表示的是缺位填充成分。现有的语法确实可以接受这些句子, 但是如何分析却不明了, 因为 GAP 属性一次只能存在于一个成分中, 而这个句子看起来有

两个缺位。这个令人困惑的答案是：GAP 特征传播的时候，不允许开始的 Q 成分移进关系从句。这是因为 REL 不是这条规则的中心成分。具体地说，规则：

$CNP \rightarrow CNP\ REL$

可以扩展为下面的规则：

$(CNP\ GAP\ ?g) \rightarrow (CNP\ GAP\ ?g)\ (REL\ GAP\ -)$

因此，疑问句中的缺位就不会出现在关系从句中。所以，当关系从句中使用到了新的缺位时，就不会出现任何问题。

所以，这种机制能够有效地处理这里出现的例子。但是，它很好地刻画了英语结构的本质吗？具体而言，疑问句缺位能不能出现在关系从句中？普遍被接受的答案是“不会”。否则，语法就会接受下面这样的疑问句：

* Which dog₁ did the man₂ we saw —₂ petting —₁ laughed? (错误的句子)

所以，这里提出的机制看上去很好地刻画了这个现象的本质。

5.5 ATN 中的保留机制

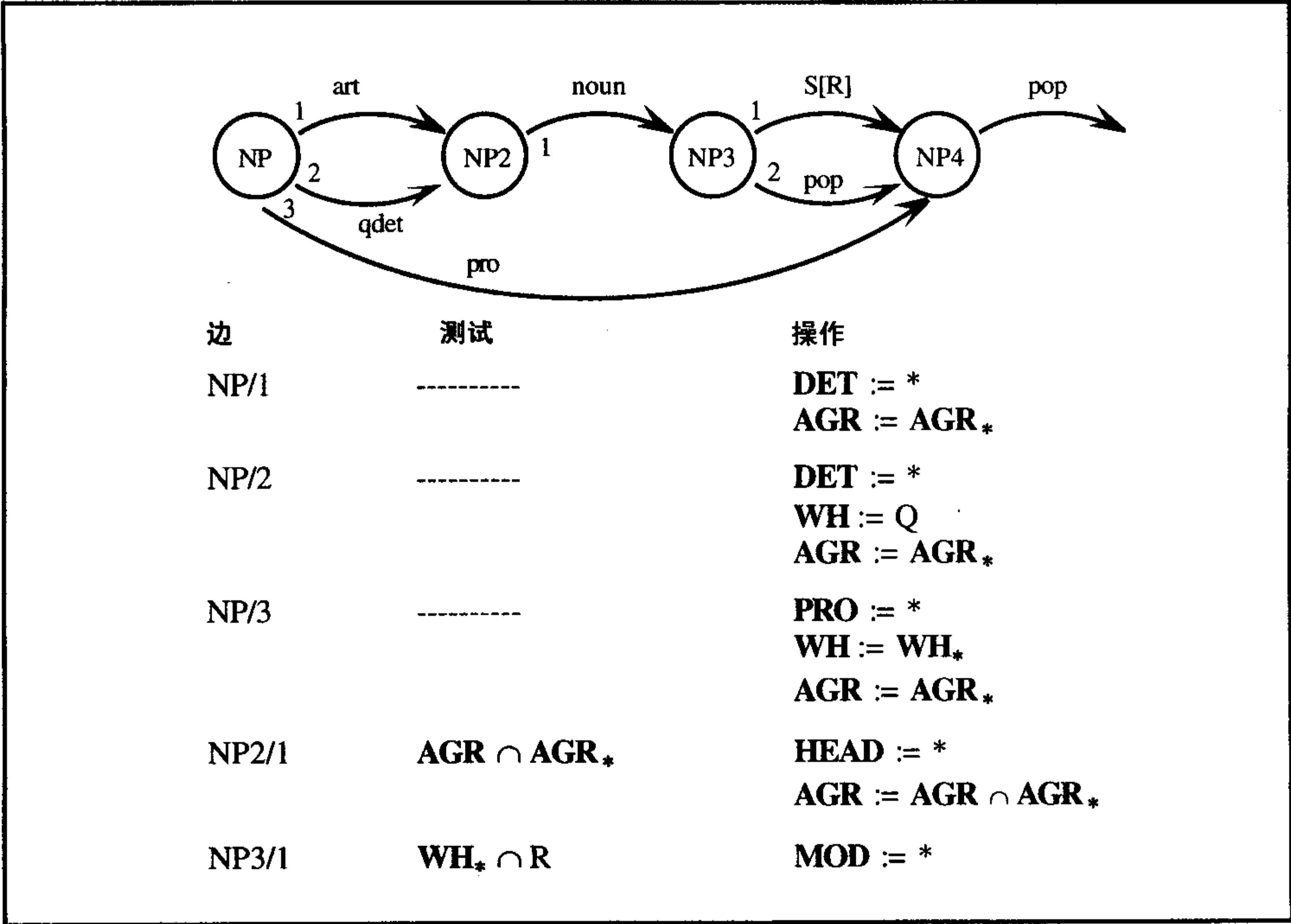
另一种移位的处理方法首先是在 ATN 架构下研究出来的。这个方法采用了一种叫做保留表的数据结构，保留表维护着那些要移位的成分。与 GAP 不同的是，某个时刻保留表可以同时包含多个成分。语法成分通过一个新操作加入到保留表中，这个新操作就是保留操作，它获取一个成分并将该成分置于保留表中。

保留操作可以存储当前在寄存器中的成分（例如，“HOLD SUBJ”操作会保留在 SUBJ 寄存器中的成分）。为了确保被保留的成分总能用来填充缺位，我们规定只有网络中边上某个操作保留的每个成分都被使用之后，ATN 系统才允许出栈边从该网络中成功返回。即被保留的成分必须用来填充当前成分或其子成分的缺位。

最后，还需要一个缺位的检测和填充机制。我们引入了一种叫做 VIR (virtual 的缩写) 的新边。VIR 边以某个语法成分的名称作为参数，当保留表中出现了这种语法成分的时候，该边方可通过。如果这个边成功通过，则相应的成分就从保留表中删除，并且返回相应边的值，这条边和 PUSH 边返回的成分形式一致。

语法 5.13 给出了 NP 网络，该网络可以识别那些用疑问词作为代词或限定词的 NP。和上下文无关文法一样，特征 WH 值设为 Q，表示该 NP 是以疑问句开头的疑问名词短语。通过压栈到 S 网络来接受关系从句，如语法 5.14 所示。这个 S 网络可以处理一般疑问句、特殊疑问句以及关系从句。特殊疑问句和关系从句的处理方法是将第一次出现的 NP (其 WH 特征值为 Q 或者 R) 加到保留表当中，然后在 VIR 边中使用。该网络会组织成为所有 + INV 的句子都可以通过 S-INV 节点，而所有 -INV 的句子都可以通过 VP 节点。所有的特殊疑问句和关系从句通过 WH-S 节点之后，我们依据该句是否倒置，重新将它们定位到标准网络之中。如果紧接着的是一条 S/2 边，那么，就可以将第一个 NP 放到保留表中。对于那些非倒置的疑问句，比如“Who ate the pizza?” (谁吃了比萨?)，以及那些 NP 形式中的关系从句，如“the man who ate the pizza” (那个吃了比萨的男人)，保留的 NP 马上会被 VIR 边 WH-S/2 使用。对于其他的关系从句，如名词短语“the man who we saw” (我们看到的那个男人) 中的从句，边 WH-S/1 会用来接受

关系从句中的主语,保留的 NP 会用在后面的 VCOMP/2 边中。对于倒置的疑问句,比如“Who do I see?”(我看见的是谁?),需要通过边 WH-S/3 来接受这里的助词,保留的 NP 必须在后面使用。



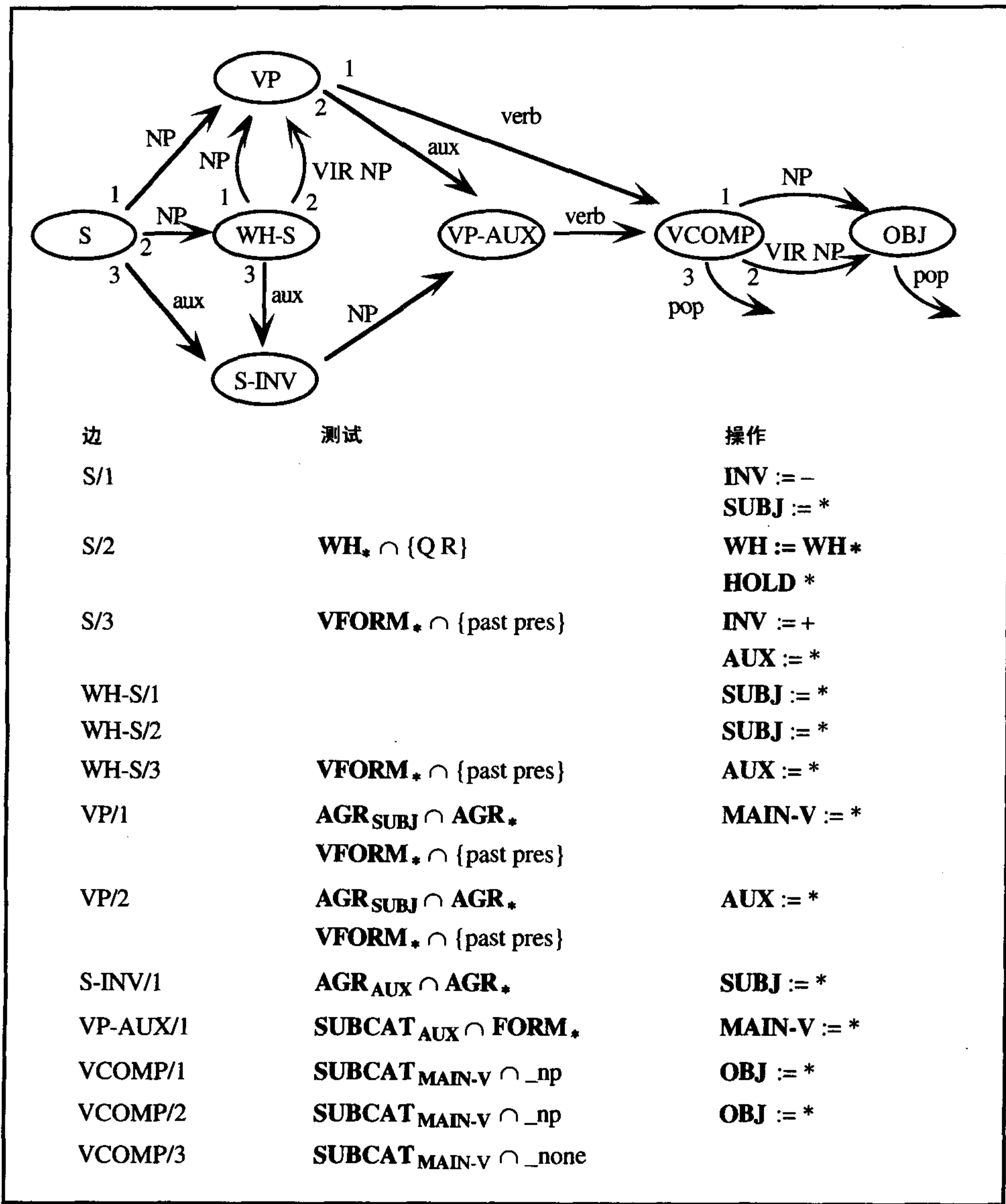
语法 5.13 包含疑问词的 NP 网络

这里的网络只能接受 SUBCAT 值为“_none”或“_np”的动词,但是,它可以很容易地进行扩展,用来处理其他的动词补语。扩展后,保留的名词短语 NP 可以适用的范围更广。图 5.15 给出了句子“The man who we saw cried”(我们看到的那个男人哭了)的处理分析流程。在这个关系从句的句法分析过程中,关系代词“who”在步骤 5 被保留,然后在步骤 8 被一条 VIR 边所使用。

注意,这个 ATN 不会接受“* Who did the man see the boy?”,因为保留的成分“who”不能被任何一个 VIR 边用到,这样,S 网络的出栈边就无法执行。类似地,“* The man who the boy cried ate the pie”也是不能接受的,因为这里的关系代词不能用在 S 网络的 VIR 边中,S 网络分析的就是关系从句。

5.5.1 方法比较

你已经看到了语法中处理疑问句的两种方法:运用 GAP 特征以及使用 ATN 的保留表。要想判定哪一种方法是最好的,必须首先确定对它们进行评价的标准。这里,需要考虑三种重要因素:覆盖率——这个方法能不能接受所有形式正确的例子;选择性——能否拒绝所有形式错误的例子;简洁性——规则编写的难易程度。



语法 5.14 疑问句和关系从句的 S 网络

在合理的假设条件下,上述两种方法看起来都可以达到需要的覆盖率。也就是说,采用任何一种方法写成的语法都可以接受任何一个合法的句子。在这种前提下,重要的因素就只有选择性和简洁性了。在这些方面,这两种方法有所不同。比如,GAP 特征理论一个潜在的假设就是含 NP 缺位的 NP 等符号必须是个空串。也就是说,NP 内部不能存在 NP 缺位。但是,使用保留表机制的 ATN 网可以接受这种结构。具体地说,形式如“* Who did the man who saw hit the boy?”的句子不可理解,但是却可以被语法 5.13 和语法 5.14 的 ATN 所接受。因为保留表中可以包含两个名词短语,它们分别对应于先后出现的“who”。在这个关系从句中,关系代词作为主语,开始时用来询问的名词短语作为宾语。但是,任何一个使用 GAP 特征的语法均不会接受这个句子,因为它不是下面规则的中心成分,GAP 特征是不能传递到关系从句中的:

CNP → CNP REL

即使它是中心成分,这个句子也不会被接受,因为在开始分析关系从句的时候,错误校验分析就会要求 GAP 特征必须具备两个值。

Trace of S Network				
Step	Node	Position	Arc Followed	Registers
1.	S	1	S/1 (for recursive call see trace below)	SUBJ ← (NP DET the HEAD man AGR 3s MOD (S who we saw))
11.	VP	6	VP/1	MAIN-V ← cried
12.	VCOMP	7	VCOMP/3 succeeds since no words left	returns (S SUBJ (NP DET the HEAD man AGR 3p MOD (S who we saw)) MAIN-V cried)
Trace of First NP Call: Arc S/1				
Step	Node	Position	Arc Followed	Registers
2.	NP	1	NP/1	DET ← the AGR ← {3s 3p}
3.	NP2	2	NP2/1	HEAD ← man AGR ← 3s
4.	NP3	3	NP3/1 (for recursive call see trace below)	MOD ← (S WH R SUBJ we MAIN-V saw OBJ who)
10.	NP4	6	NP4/1 pop	returns (NP DET the HEAD man AGR 3s MOD (S who we saw))
Trace of Recursive Call to S on Arc NP3/1				
Step	Node	Position	Arc Followed	Registers
5.	S	3	S/2 (call to NP network not shown)	WH ← {Q R} HOLDING (NP PRO who WH {Q R})
6.	WH-S	4	WH-S/1	WH ← R SUBJ ← (NP PRO we ...)
7.	VP	5	VP/1	MAIN-V ← saw
8.	VCOMP	6	VCOMP/2 (uses the NP on the hold list)	OBJ ← (NP PRO who ...)
9.	OBJ	6	OBJ/1 pop	returns (S WH R SUBJ we MAIN-V saw OBJ who)

图 5.15 “₁ The ₂ man ₃ who ₄ we ₅ saw ₆ cried ₇” 的 ATN 句法分析流程

如何防止保留成分在任何时候均可能被使用,目前还没有一个明确的方法。因此,为了刻画这些约束,必须扩展 ATN 架构中的保留表机制。采用现存的机制,惟一可能的约束途径是使用特征值来记录每一种上下文中可用的保留成分,而这样会十分混乱。但是,你可以做一个简单的扩展。可以引入一个新的操作 HIDE, HIDE 可以用来暂时隐藏保留表中已有的成分,直

到执行了一个显式的操作 UNHIDE, 或者当前的成分已经构建完成。通过这种扩展, 我们可以修正处理关系从句的 ATN 语法, 在执行保留当前成分的 Hold 操作之前, ATN 需要先执行 HIDE 操作。

不过, 在这个例子中, 形式化本身并没有包含这些约束条件, 而是用于表述约束关系。它可以同样简单地描述一种违反这些约束的语言。另一方面, 基于 GAP 特征的理论在特征传播的定义上引入了一套约束关系。因此, 对于表示违反移位约束的语言来说, 我们编写一部能描述它的语法将是十分困难的。

就目前来说, 真正地考虑所有的移位约束仍然是一个正处于研究阶段的领域, 这里给出的简单分析是一个重要的关键点。如果形式化太弱, 就不能描述语言的全部。如果形式化太强, 为了消除那些可以接受但是不合理的句子结构, 语法或许会异常复杂。为此, 最好的解决方案就是找到恰好能描述自然语言的形式化系统。在这样一种语言中, 许多约束最初看起来是语言中随意的限制, 而最终却是形式化自身发展的必然结果, 我们并不需要做进一步的考虑。

尽管我们在上下文无关文法中引入了 GAP 特征传播技术, 同时在 ATN 中引入了保留表机制, 不过, 并不一定要将这些技术限制在这些形式化体系中。例如, 可以研究出另一种上下文无关文法的扩展形式, 其中可以融入保留表的方法, 采用保留表来处理缺位。同样, 也可以在 ATN 中增加一些 GAP 特征的传递规则。不过, 这可能会更加困难, 因为 GAP 特征传播规则依赖中心子成分这种标记体系, 而它与 ATN 语法不存在直接的对应关系。

5.6 缺位索引方法

另一种处理缺位的方法综合了 GAP 特征方法和保留表方法。这种方法通常叫做缺位索引方法(gap threading), 常用在逻辑语法中。它为每个谓词增加两个额外的参数——其中一个表示当前成分可能会用到的填充成分列表, 另一个表示语法成分分析完成后不会用到的填充成分结果列表。所以, 会有下面这种形式的谓词:

s(position-in, position-out, fillers-in, fillers-out)

该谓词为真的前提是输入的 position-in 和 position-out 之间存在合法的 S 成分。如果构建 S 的时候用到了缺位, 那么该缺位的填充成分将出现在 fillers-in 中, 而不会在 fillers-out 中。例如, 缺位为 NP 的 S 成分对应的谓词为 *s(In, Out, [NP], nil)*。在没有缺位的情况下, fillers-in 与 fillers-out 相同。

现在, 让我们来看看关系从句所处理的一个示例。语法 5.16 给出了需要用到的规则。为了让这个例子更加简单, 我们在这里省略了各种特征约束, 比如那些必需的一致性约束和次范畴约束。

为了考察这些使用到的规则, 我们来看看句子“The man who we saw cried”(我们看到的那个男人哭了)的分析过程, 具体流程见图 5.17。关系从句是从步骤 7 开始分析的。应用规则 9, 词语“who”被识别为关系代词, 变量 Filler 绑定为列表[NP]。由于该填充成分没有在 NP 中使用, 随后, 会依次传递到内嵌的 S 成分(步骤 9)、NP 成分(步骤 10), 接着是 VP 成分(步骤 12)。从这里开始, 它在步骤 14 被传递到 NP 谓词中, 根据规则 10, NP 谓词使用该填充成分。FillersOut 变量为空, 所以必须使用填充成分, 而在这里又没有别的 NP 可用, 因此, 只有应用那些需要填充成分的规则。一旦使用了缺位, 位置 1 到位置 6 上的整个 NP 被识别出来, 剩下的分析就直截了当了。

1. $s(In, Out, FillersIn, FillersOut) :- np(In, In1, FillersIn, Fillers1),$
 $vp(In1, Out, Fillers1, FillersOut)$

2. $vp(In, Out, FillersIn, FillersOut) :- v(In, In1)$

3. $vp(In, Out, FillersIn, FillersOut) :- v(In, In1), np(In1, Out, FillersIn, FillersOut)$

4. $np(In, Out, Fillers, Fillers) :- art(In, In1), cnp(In1, Out)$

5. $np(In, Out, Fillers, Fillers) :- pro(In, Out)$

6. $cnp(In, Out) :- n(In, In1), np-comp(In1, Out)$

7. $np-comp(In, In) :-$
(这包括没有NP补语的情况。)

8. $np-comp(In, Out) :- rel-intro(In, In1, Filler),$
 $s(In1, Out, (Filler\ nil), nil)$
(这里保留了Rel-Intro成分, 在随后的S中可能使用它。)

9. $rel-intro(In, Out, [NP]) :- relpro(In, Out)$
(其中relpro接受任何带WH特征R的代词。)

10. $np(In, In, [NP \mid Fillers], Fillers) :-$
(这一规则从填充位构建一个空np。)

图 5.16 采用缺位索引方法的逻辑语法

步骤	状态	下一步操作
1.	$s(1, 7, nil, nil)$	应用规则 1
2.	$np(1, In1, nil, Fillers1) \quad vp(In1, 7, Fillers1, nil)$	应用规则 4
3.	$art(1, In2) \quad cnp(In2, In1)$	提供的 $art(1, 2)$
4.	$cnp(2, In1)$	应用规则 6
5.	$n(2, In3) \quad np-comp(In3, In1)$	提供的 $n(2, 3)$
6.	$np-comp(3, In1)$	应用规则 8
7.	$rel-intro(3, In4, Filler) \quad s(In4, In1, Filler, nil)$	应用规则 9
8.	$relpro(3, In4)$	提供的 $relpro(3, 4)$ 提供的 $rel-intro(3, 4, [NP])$
9.	$s(4, In3, [NP], nil)$	应用规则 1
10.	$np(4, In5, [NP], Fillers1) \quad vp(In5, In1, Fillers1, nil)$	应用规则 5
11.	$pro(4, In5)$	提供的 $pro(4, 5)$ 提供的 $np(4, 5, [NP], [NP])$
12.	$vp(5, In1, [NP], nil)$	应用规则 4
13.	$v(5, In6) \quad np(In6, In1, [NP], nil)$	提供的 $v(5, 6)$
14.	$np(6, In1, [NP], nil)$	提供的 $np(6, 6, [NP], nil)$ 提供的 $vp(5, 6, [NP], nil)$ 提供的 $s(4, 6, [NP], nil)$ 提供的 $np-comp(3, 6)$ 提供的 $cnp(2, 6)$ 提供的 $np(1, 6, nil, nil)$
15.	$vp(6, 7, nil, nil)$	应用规则 2
16.	$v(6, 7)$	提供的 $v(6, 7)$ 提供的 $vp(6, 7, nil, nil)$ 提供的 $s(1, 7, nil, nil)$

图 5.17 “₁ The ₂ man ₃ who ₄ we ₅ saw ₆ cried ₇”的句法分析流程

我们设计了一套非常方便的标记体系,可以用来设计有限子句语法,这样语法就可以很简单地翻译成 PROLOG 程序。同样,我们也可以设计一套标记体系来辅助缺位处理语法的编写。具体地说,我们有一套叫做外置转换语法(extraposition grammar)的形式化系统,它除了可以接受正常的上下文无关规则之外,还可以接受如下形式的规则:

REL-MARK...TRACE → REL-PRO

从本质上看,这些规则的含义是:成分 REL-MARK,加上句子中后来出现的成分 TRACE,可以改写为 REL-PRO。这样的规则违反了句法形式的树结构,并且接受名词短语“the mouse that the cat ate”(那只被猫吃了的老鼠),该短语的分析过程见图 5.18。这些规则可以采用缺位索引方法组织成逻辑语法。

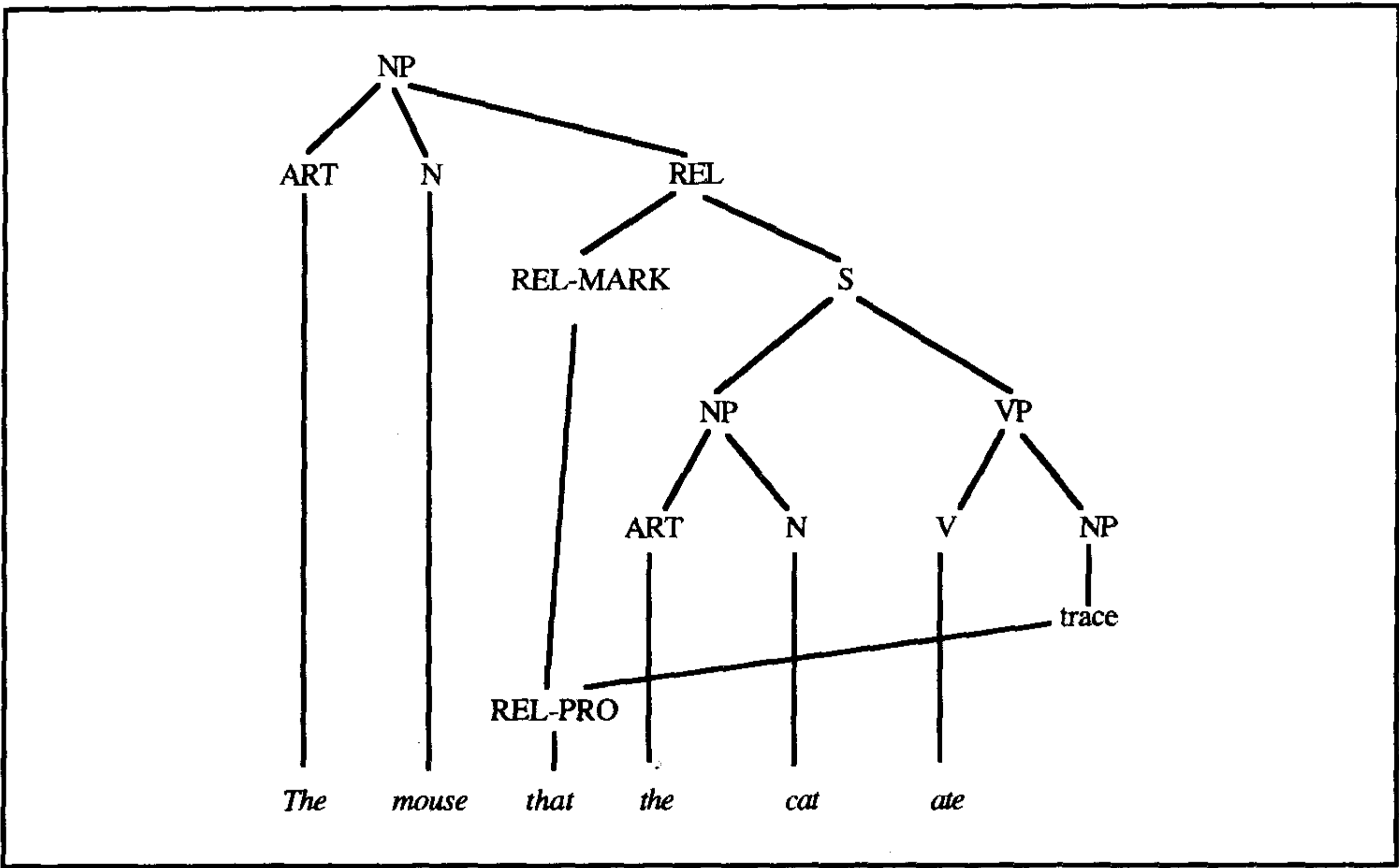


图 5.18 外置转换语法的分析树

现在,我们将缺位索引方法和其他方法进行比较。当然,可以通过在谓词上增加参数来实现任何一种方法。如果仅仅将缺位索引看做是实现方法,那么,看起来很像是在逻辑语法中实现了一个保留表。语法的设计者可以决定是否在语法中传递保留表。因此,它还具有一定的灵活性,这就避免了简单保留表机制中会出现的那些问题。例如,语法 5.16 不会将填充成分从名词短语之外传递到关系从句中。所以,上一节讨论的问题在这里均可以避免。它也可以采用 5.3 节给出的 GAP 特征引入算法,以便将缺位索引规则组织成为逻辑语法。因此,这种方法给了语法设计者很大的灵活性。不过,像保留表方法一样,我们必须明确地施加约束的传递,它不是形式化的必然结果。

5.7 小结

在自然语言中,很多复杂的方方面面都可以看成是移位现象,即一个位置的成分需要用来满足另外一个位置上的约束关系。这些现象分为两类:有界移位和无界移位。有界移位包括

一般疑问句和被动语态,无界移位包括特殊疑问句和关系从句。为了处理移位现象,目前研究出来的计算技术在很大程度上可以进行泛化处理,从而刻画出不同句子形式的本质。如果认真细致地使用特征系统,只要采用很少的规则,就可以将基本的陈述句语法进行扩展,进而处理其他的形式。在这里,我们介绍了三种不同的方法来处理这种移位现象。第一种方法使用了专门的特征传播规则,规则中用到了一个叫做 GAP 的特征。第二种方法在 ATN 中引入保留表机制,即增加一个新操作(hold 操作)和一条新边(VIR 边)。第三种涉及的是缺位索引方法。这种方法使用了类似于保留表的结构,该结构采用规则从一个成分传递到另一个成分中。如果进行合理的维护利用,所有这些方法都可以成功地构建出英语的语法,其覆盖率会相当可观。

5.8 相关工作与深入阅读材料

无界依存现象激励人们在语法形式化方面进行了相当多的研究。依据与上下文无关文法关系的远近,这些研究可以粗略地分为几类。比如,转换语法理论(Chomsky, 1965; Radford, 1981)提出要完全在上下文无关文法框架之外处理无界依存。而词汇功能语法理论(Kaplan 和 Bresnan, 1982)和广义短语结构语法理论(GPSG, Gazdar, 1982; Gazdar 等, 1985)提出的方法是,采用带特征的上下文无关文法来处理无界依存关系。在新的形式化系统定义方面,还有很多工作值得一提,由其定义的形式化比上下文无关文法功能稍强,并能处理长距离依存,这些理论包括树粘接语法(TAG, tree-adjointing grammar, 参见框 5.5),以及组合范畴语法(combinatory categorial grammars, Steedman, 1987)。

在 ATN 架构内, Woods(1970)第一个对移位现象进行了非常全面的研究。他用了相当多的篇幅来说明,如果使用寄存器测试,并与保留表机制一样设置寄存器,那么 ATN 就能够分析出转换语法(Chomsky, 1965)涉及到的大部分移位现象。本章的 ATN 部分就是根据这篇论文简化而来的,我们也结合了后期 Kaplan(1973)所进行的研究工作。作为 ATN 的另一种表示形式,也可以参考 Bates(1978)和 Winograd(1983)。

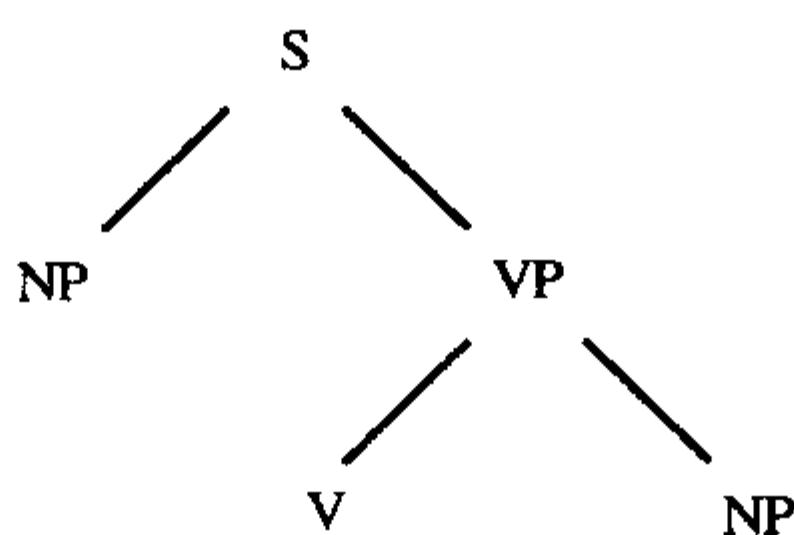
基于上下文无关文法的系统已经应用了相似的扩展技术,实际上,这些系统中的大部分都会接受很多不合理的句子。原因在于,这些语法系统构建之后,即使在没有移位现象的上下文中,缺位成分仍然可选。然后,句法分析器会依靠一些专门的特征来消除其中的一些错误,或者依靠语义分析来拒绝那些结构错误却被接受的句子。这种方法究竟能在多大程度上解决问题? Robinson(1982)提供了一个很好的例子。

处理移位的缺位索引方法以及外置转换语法定义方面的工作,可以参见 Pereira(1981)。类似的技术还可以参见当前许多的句法分析器(比如 Alshaw, 1992)。

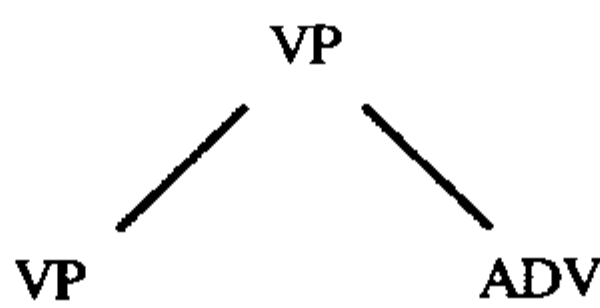
通过 GAP 特征传播来处理移位现象的章节部分是受 GPSG(Gazdar 等, 1985)的启发。在 GPSG 中,特征传播是由一个通用的原则集(参见框 5.4)决定的。中心驱动短语结构语法(head-driven phrase structure grammar, Pollard 和 Sag, 1987; 1993)派生于 GPSG,它更便于计算。GPSG 在短语的中心成分上广泛地使用了次范畴信息。通过这种做法,大大地简化了上下文无关文法,代价是词典更加复杂。

框 5.5 树粘接语法

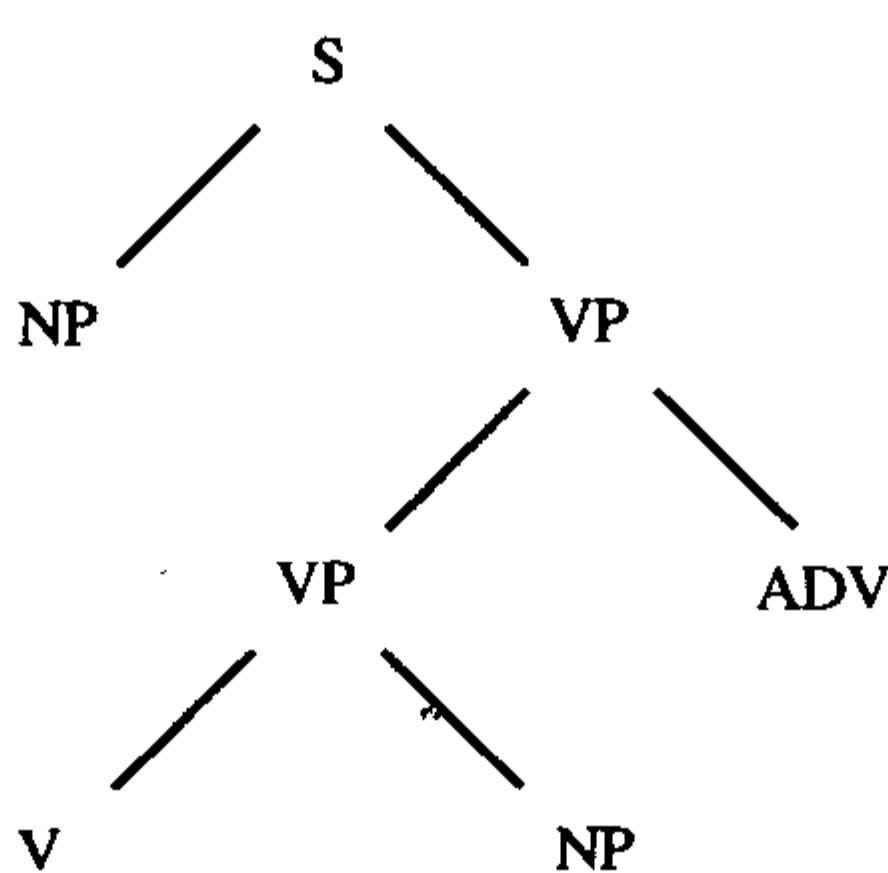
树粘接语法(TAG, Joshi, 1985)是另一种处理无界依存的方法。这个形式化体系没有语法规则,只有一组初始的树结构用来描述语言中最简单的句子。同时,还有一种叫做树粘接的操作,树粘接指的是将一棵树插入到另一棵树中,这样就可以产生更复杂的结构。例如,一棵简单的初始树如下:



更复杂的句子是用辅助树派生出来的,辅助树描述了语言中最小的递归形式。例如,一棵允许动词短语中出现副词的辅助树可以表示如下:



粘接操作需要将某个成分 C 的辅助树插入到另外一棵包含成分 C 的树中,这棵辅助树就是成分 C 的递归表达形式。将上面 VP 的辅助树粘接到初始的 S 树中,可以得到一棵新的树,其结构如下:



通过基于树的形式化表示,可以得到足够的上下文来描述长距离的依存关系。在这个理论中并不存在移位,取而代之的是,语法成分在开始的时候彼此靠近,接着,通过粘接操作往它们之间添加其他结构。最终,可以得到一种形式化体系,其表达能力稍强于上下文无关文法,但肯定弱于上下文有关文法。

5.9 习题

- 1. 【易】区别有界(局部的)移位和无界(非局部的)移位。为了处理无界移位,需要在扩充系统上做哪些扩展? 为什么特殊疑问移位是无界移位,给出例子来支持你的观点。

2. 【易】将下面的缩写成分和规则扩展为相应的完整特征结构格式。

(S[- inv, Q] AGR ?a)

NP[R, - gap]

VP → V [_np_s: inf] NP S[inf]

3. 【中】采用 5.3 节的语法, 请使用 chart 图给出下面疑问句的分析结果, chart 图的形式见图 5.12。

In which town were you born? (你出生在哪个镇子里?)

Where were you born? (你在哪里出生的?)

When did they leave? (他们什么时候离开的?)

What town were you born in? (你出生在什么样的镇子里?)

4. 【中】GPSG 允许某些规则包含多个中心子成分。比如, 下面给出了两个动词短语的连接规则, 该规则包含了两个中心成分:

VP → VP 和 VP

- a. 多个中心成分的引入对 GAP 特征传播算法有何影响? 为了回答这个问题, 我们看一些例子, 如下:

Who did you see and give the book to ? (你看见并把书给他的那个人是谁?)

What man did Mary hate and Sue love? (Mary 痛恨而 Sue 喜爱的是什么样的男人?)

同时, 也要考虑下面这些结构错误的句子:

* Who did you see and give the book to John?

* What man did Mary hate John and Sue love?

- b. 写出带 GAP 特征的 VP 规则, 然后, 根据你的规则扩展语法 5.8, 运用扩展后的语法画出下面句子的 chart 图:

Who did Mary see and Sue see? (Mary 看见的和 Sue 看见的那个人是谁?)

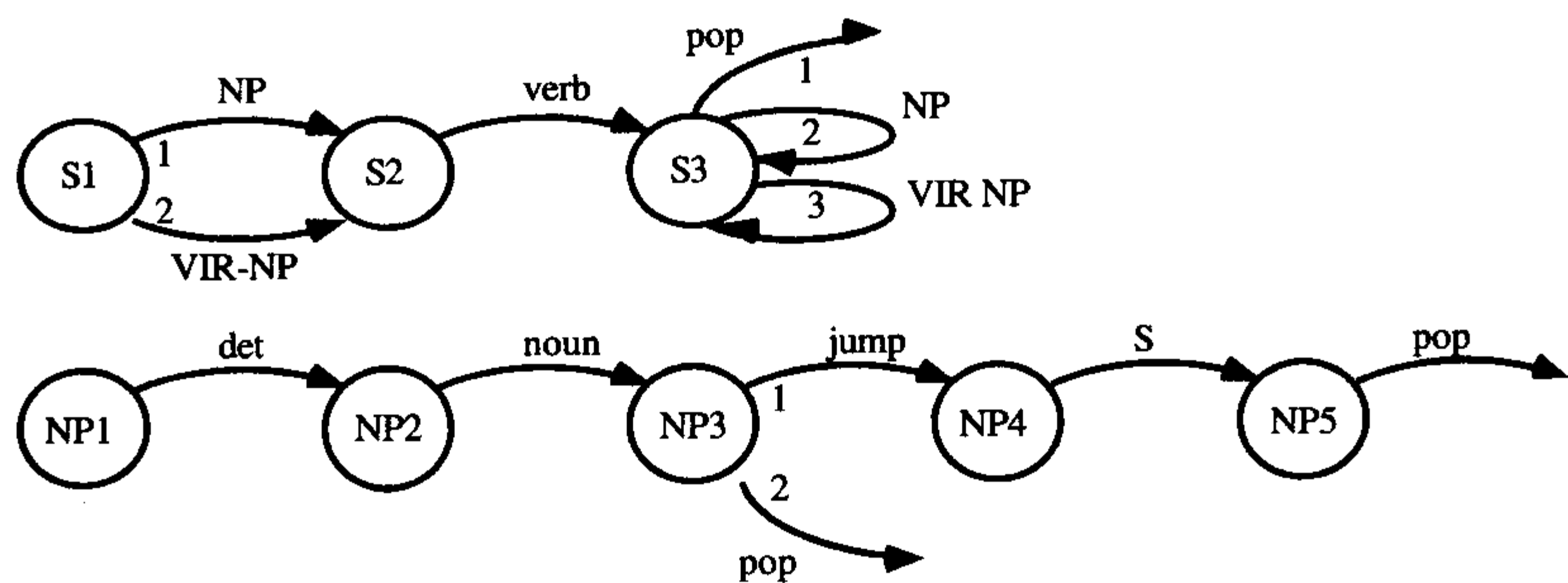
你只须给出最终分析中用到的成分, 但是一定要给出每个成分的所有特征值。

5. 【中】主题化是需要移位分析的另一种现象, 它允许下面形式的句子:

John, Mary saw. (John, Mary 看到了。)

To John, Mary told the story. (对 John, Mary 讲了这个故事。)

- a. 请说明特殊疑问句中的约束类型同样可以应用于主题化。GAP 特征传播技术也可以用在这里。
- b. 你是否需要引入一个类似于 GAP 的新特征来处理主题化问题? 或者, 是否可以再使用疑问句和关系从句中的 GAP 特征。
- c. 扩展语法 5.8, 保证它可以接受主题化形式, 具体形式在示例中已经给出。
6. 【中】看看下面的 ATN 语法。在边 NP3/1 上有一个保留当前名词短语的操作。
- a. 给出一个合法的句子序列, 要求长度任意但仍然被该 ATN 网络接受。



- b. 下面的哪些句子可以被该 ATN 网络接受(假定每个词语都有相应正确的词类)? 对于那些被接受的句子,给出该网络分析后的句子结构(即一个合理的寄存器的值结构)。
- i. The man Mary hit hit the ball.
 - ii. The man the man hit the man hit.
 - iii. The man hit hit.
 - iv. Hit the man hit the man.

7. 【难】下面的名词短语来自于计算机操作员和计算机系统不同用户之间的一些对话:

Could you mount *a magtape* for me? (你能帮我把磁带装上吗?)
No ring please. (请别响了。)
I am not exactly sure of *the reason*, but we were given *a list of users we are not supposed to mount magtapes for*, and you are on it.
(我不清楚原因,但是我们得到了不能帮助装磁带的用户名单,而你恰好又在名单之中。)
Could you possibly retrieve *the following two files*? I think *they* were on *our directory* last night.
(你能帮我找回下面这两个文件吗? 我想昨天晚上它们还在我们的目录中呢。)
Any chance I can recover from *the most recent system dump*?
(我还可能从最近转储的系统中恢复过来吗?)

扩展 5.3 节和 5.4 节中的语法,保证它们可以接受这些名词短语。给出这些名词短语中所有的新词词条,并给出每个 NP 最终分析得出的 chart 图。

8. 【难】

a. 5.1 节给出可分析英语助动词结构的语法,但是它不能处理以下形式的短语:

Jack *has to see* a doctor. (Jack 不得不去看医生。)
The cat *had to be found*. (不得不去找的那只猫。)
Joe *has to be winning* the race. (Jack 只能去赢得这场赛跑。)
The book *would have had to have been found* by Jack.
(这本书本来就应该由 Jack 去找到。)

请扩展这个语法,使之能接受上面的助动词序列。同时,要求该语法拒绝那些不合理的句子,比如:

* Jack *has to have to see* a doctor.

* Janet *had to played* the violin.

* Will *would to go* to the movies.

- b. 在这个助词系统中,使用“be going to”形式的短语,并进行类似的分析,要求给出正例和反例。

第6章 通向高效的句法分析

到目前为止,我们所讨论的句法分析框架都要依靠完全搜索技术,最终才能找到句子可能的解析结果。而在语言处理的过程中,大多数人的直觉和这种搜索过程都不一样。具体来说,人的句法分析过程看起来更接近于一种决策过程——这种过程并不是在众多可选项中进行搜索,而是利用当时已有的信息直接选择正确的解释。尽管我们都知道人类自身的感知过程很不可靠,但是,实验证据还是表明人类在句法分析过程中不会完全搜索整个语法。本章探讨了一些让句法分析更加高效的方法,而且在某些案例中采用了决策过程。如果句法分析不是你的兴趣所在,则可以将本章的所有内容视为可选材料。

在这里,我们有两种方案值得考虑。第一种是在不改变最终输出结果的前提下,通过减少搜索次数来提高句法分析算法的效率。第二种可以在句法分析器能找出的不同句子解释中,采用优选技术来提高最终效率。这两种方案都会在本章中进行探讨。

6.1节在句法分析器中引入优选的概念,并用来选择特定的解释。有时候,这种方法会消除一些其他的可能解释,从而带来损失。心理语言学在人类句法分析过程方面的理论是我们模型的根据,也是本章余下部分所讨论技术的背景资料。6.2节考察了一种提高句法分析效率的方法,具体的途径是将搜索过程预编译成表格形式。这些技术已经用在了移进归约(shift-reduce)句法分析器中,移进归约句法分析器可以用来分析程序语言。我们也可以对这些技术进行泛化,使之可以处理歧义,并能用来分析自然语言。6.3节描述了一个完整的确定型句法分析框架,在句法分析过程中,它不需要进行回溯。这意味着句法分析器可能会犯错误,可能找不到部分句子的正确解释。不过,如果能够正确使用这种方法,那么,系统只会在那些人容易错误分析的句子产生错误分析。6.4节总结了各种有效表示歧义的方法,其中有“坍塌”(collapsing)解释方法,或者通过句法分析目标的重新定义来消除歧义。6.5节探讨了对输入句子进行浅层句法分析的技术,浅层句法分析只分析句子中那些能进行可靠判断的部分。

6.1 句法分析中人的优选策略

人们是怎样分析句子的呢?到目前为止,本书只是在抽象的层面讨论句法分析,并没有参照人在这方面的心理学依据。对于这个问题,心理语言学家已经运用各种技术展开了多项研究与调查,从分析人在优先解释方面的直觉,到采用详尽的实验来监测人在语言读和听的过程中各个时刻的分析处理。这些研究已经揭示了人在消歧方面的一些通用原则。

这些研究得出的最基本结论是人们不会对所有可能的句法解释一视同仁。我们可以通过一些具有临时歧义的句子来说明这一点,临时歧义指的是会让人觉察出错误的分析,比如,句子“The raft floated down the river sank.”(沿着小河飘下去的救生艇沉没了。)当你读到词语“sank”(沉没)的时候,会意识到前面对这个句子所做的分析都是不正确的。在文献中,这种句子通常称为“花园幽径”(garden-path),意思是指像领着人走在花园里弯弯曲曲的小路上一样。下面,会提供一些通用原则,这些原则会帮助我们预测花园幽径将通向何方。

6.1.1 最小附着原则

最通用的原则是最小附着原则(minimal attachment principle),其特点是优先选择分析树节点数目最少的句法来分析结果。因此,给定语法 6.1 后,句子“The man kept the dog in the house.”(这个男人把狗留在房子里。)将解释为介词短语(PP)“in the house”(房子里)修饰动词,而不是修饰名词短语(NP)“the dog”(狗)。这两种解释见图 6.2。其中,PP 附着 VP 的解释分别运用了规则 1.1、规则 1.2 和规则 1.6,同时还运用了三次规则 1.4 来推导 NP。这棵句法分析树总共有 14 个节点。而 PP 附着 NP 的解释派生于规则 1.1、规则 1.3、规则 1.5 和规则 1.6,同时还运用了三次规则 1.4,最终生成的句法分析树有 15 个节点。根据上述规则,第一种解释优先,这种分析很可能更符合你的直觉。

1.1	$S \rightarrow NP VP$	1.4	$NP \rightarrow ART N$
1.2	$VP \rightarrow V NP PP$	1.5	$NP \rightarrow NP PP$
1.3	$VP \rightarrow V NP$	1.6	$PP \rightarrow P NP$

语法 6.1 简单的上下文无关文法

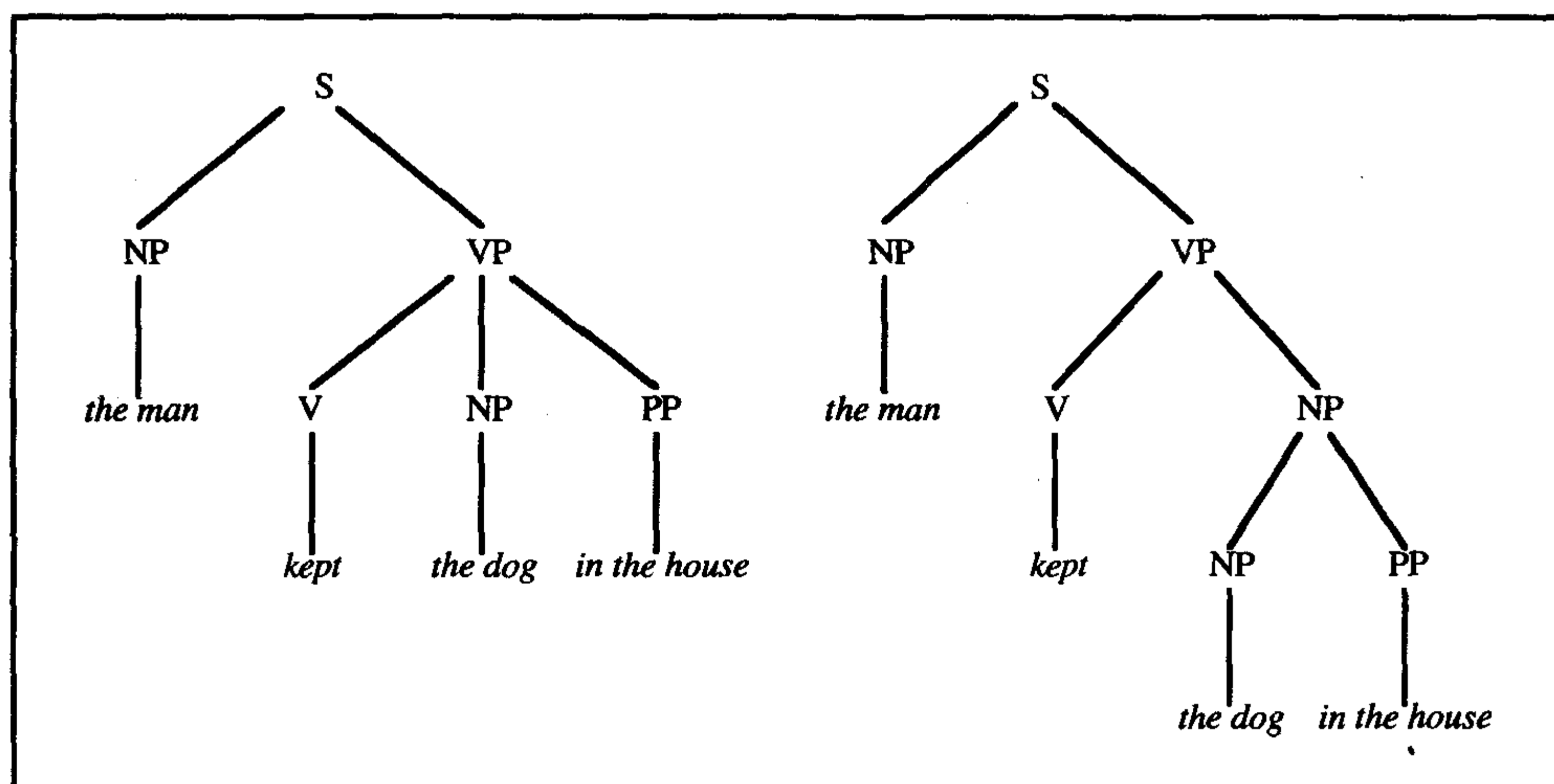


图 6.2 根据最小附着原则,左边的解释优先

这条原则看起来太强,某些句子几乎不可能正确地分析出结果。下面这个句子就是其中的一个例子:

We painted all the walls with cracks.(我们把所有有裂纹的墙壁都刷了一遍。)

这句话与所有的常识都相悖,人们通常将这个句子的意思解读为“裂纹被涂到墙壁上”,或者解读为“裂纹作为一种工具,用来涂墙壁”。这两种反常解读的原因都在于误认为 PP 附着的是 VP(paint),而不是 NP(the walls)。另外一个经典的例句如下:

The horse raced past the barn fell.(跑过这个谷仓的马摔倒了。)

这句话有一个非常合理的分析结果,所表达的意思与句子“The horse that was raced past the barn fell.”相对应。不过,在原来的句子中,遇到词语“raced”的时候创建一个省略的关系从句,相对于将“raced”直接解释为这个句子的主动词而言,会引入更多的节点。当然,当分析到词语“fell”的时候,后一种解释也是行不通的。

6.1.2 右关联原则

第二条原则叫做右关联(right association)或者后闭合(late closure)。这个原则表示的是,在所有其他条件均相同的前提下,新成分更倾向于解释为当前正在构建成分的一部分(而不是语法树中更高层的成分)。给定下面的句子:

George said that Henry left in his car.

较优的解释应该是“Herry 留在车里”,而不是“George 在车里说话”。当然,这两种解释在句法上都是可以接受的分析。图 6.3 给出了它们相应的句法分析树。前一种解释将 PP 附着在它前面紧邻着的 VP 上,而后一种解释将 PP 附着在句法树中更高层的 VP 上。因此,右关联原则认为前者优先。类似地,句子“I thought it would rain yesterday”的优先解释是:“昨天”是“下雨的时间”而不是“我思考的时间”。

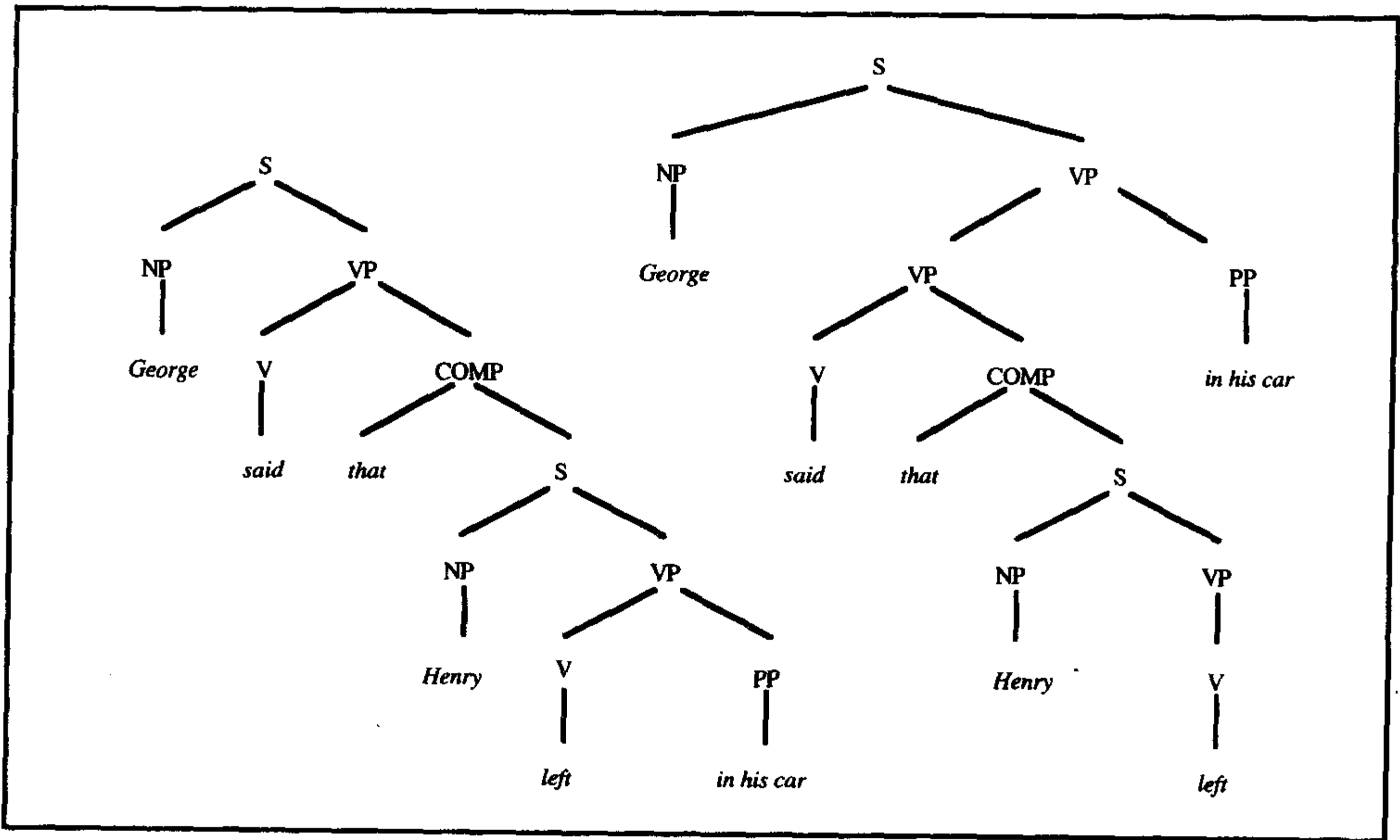


图 6.3 “George said that Henry left in his car.”的两种解释

6.1.3 词汇优先原则

在某些情况下,前面的两种原则看起来相互矛盾。对于句子“The man kept the dog in the house.”来说,根据右关联原则,我们好像应该更倾向于介词短语修饰“the dog”(狗)。然而,根据最小附着原则,又似乎应该更倾向于介词短语修饰动词短语。你可能会建议在这种情况下最小附着原则应该优先于右关联原则。然而,这种关系远远不止于此,往往要复杂得多。现在,让我们看看下面这些句子:

1. I wanted the dog in the house.
2. I kept the dog in the house.
3. I put the dog in the house.

介词短语“in the house”在句子 1 中看上去更可能是修饰“dog”的(尽管另一个解释也可能和“I wanted the dog to be in the house”的意思一样)。在句子 2 中,这个介词短语看上去好像更可能修饰 VP(虽然也有可能修饰 NP,和句子“I kept the dog that was in the house.”一样)。最后,在句子 3 中,该介词短语很明确地附着在动词短语上,而不存在任何其他的解释。

这些例子表明,词项能够影响句法分析的优先选择。在这个例子中,词项就是使用到的动词。在许多情况下,词汇优先原则的优先级会屏蔽其他通用原则的优先级。比如,如果动词次范畴需要一个介词短语,那么某些 PP 一定要附着在 VP 之上。当然,我们也可能发现其他 PP 比该动词短语附着的优先级更高。如果没有上述两种情况,介词短语的分析将遵循通用原则。

这样,对于前面提到的几个动词来说,“want”不能优先结合任何介词短语,而“keep”可能优先将介词“in”,“on”或“by”的 PP 附着 VP。最后,动词“put”需要(由其次范畴决定)一个以“in”,“on”或“by”等词开头的介词短语,这些介词短语必须要附着 VP。这种方法的前景很好,但是,如果没有一些形式化框架来统一表示这些信息,那么,我们就很难对方法进行评估。在下一章中将探讨这类方法。

6.2 对不确定性进行编码:移进归约句法分析器

提高句法分析器效率的一种途径是采用某些技术对不确定性进行编码。这样,句法分析器就不需要进行随意的选择,后期也不需要回溯。我们将不确定性一直贯穿句法分析的始终,直到输入的解析结果惟一为止。可以在句法分析过程中显式地实现这种思想,最后的算法与第 3 章论述的宽度优先分析器相似。这种技术预先考虑了所有的可能性,并将信息存储在句法分析器的控制表中,这使得句法分析算法的速度比前面介绍的所有方法都要快得多,我们对本节介绍的方法的效率估计就是基于以上事实的。

这些技术主要是针对无歧义的上下文无关文法而设计开发的。对任何一个给定的句子,这种语法最多只有一种解释。这种约束对于编程语言是合理的,但是,自然语言显然不存在没有歧义的语法。不过,这些技术还是可以通过各种途径加以扩展,使之可以应用到自然语言的句法分析中。

6.2.1 确定句法分析器的状态

现在,让我们看看如何将这种方法应用到小型语法中,如语法 6.4 所示。这种技术需要预先判定句法分析器所有可能的状态,以及状态之间的转化关系。句法分析器的状态定义为句法分析过程中可以在当前位置使用的点号规则完全集(即 chart 句法分析器中活动边上的标记)。所谓完全,指的是如果某个状态包含规则 $Y \rightarrow \dots \circ X \dots$,而 X 是非终结符,那么,所有的 X 规则也必须包含在该状态中。比如,句法分析器的初始状态包括规则:

$$S \rightarrow \circ NP VP$$

同时还包含所有的 NP 规则,在语法 6.4 中的 NP 规则只有:

$$NP \rightarrow \circ ART N$$

2.1 $S \rightarrow NP VP$	2.3 $VP \rightarrow AUX V NP$
2.2 $NP \rightarrow ART N$	2.4 $VP \rightarrow V NP$

语法 6.4 存在 AUX/V 歧义简单语法

因此,初始状态 S0 可以归结为如下形式:

初始状态: $S \rightarrow \circ NP VP$
 $NP \rightarrow \circ ART N$

换句话说,句法分析器的起始状态是在寻找一个 NP 并开始构建 S,同时需要寻找一个 ART 来构建该 NP。初始状态后面是哪些状态呢? 为了计算这个问题,需要考虑能将点号前移的终结符或者非终结符,并且考虑派生出的新状态。如果你选择的是符号 ART,那么推导出的状态是:

状态 S1: $NP \rightarrow ART \circ N$

如果你选择的是符号 NP,那么新状态中的规则是:

$S \rightarrow NP \circ VP$

现在,如果你扩展 VP 来找到所有可能的起始符号,会得到如下状态:

状态 S2: $S \rightarrow NP \circ VP$
 $VP \rightarrow \circ AUX V NP$
 $VP \rightarrow \circ V NP$

现在,对 S1 进行扩展,如果你的输入为 N,那么,就可以得到包含一条完备规则的状态,该状态为:

状态 S1': $NP \rightarrow ART N \circ$

采用 V 扩展 S2,可以得到状态:

状态 S3: $VP \rightarrow V \circ NP$
 $NP \rightarrow \circ ART N$

采用 AUX 扩展 S2,得到状态:

状态 S4: $VP \rightarrow AUX \circ V NP$

同时,采用 VP 扩展 S2,可以得到如下状态:

状态 S2': $S \rightarrow NP VP \circ$

接着,采用 ART 扩展状态 S3,你会发现自己再次进入状态 S1,这和从状态 S0 依据 ART 进行扩展一样。另一方面,根据 NP 继续扩展状态 S3,可以产生新的状态,结果如下:

状态 S3': $VP \rightarrow V NP \circ$

接着,我们依据 V 扩展状态 S4,可以产生下面的新状态:

状态 S5: $VP \rightarrow AUX V \circ NP$
 $NP \rightarrow \circ ART N$

然后,依据 ART 扩展状态 S5 的规则,再次产生状态 S1。最后,依据 NP 展开 S5,可以产生下面的状态:

状态 S5': $VP \rightarrow AUX V NP \circ$

现在,这个过程全部完成了。最终,你可以派生出控制句法分析的状态转移图,如图 6.5 所示。

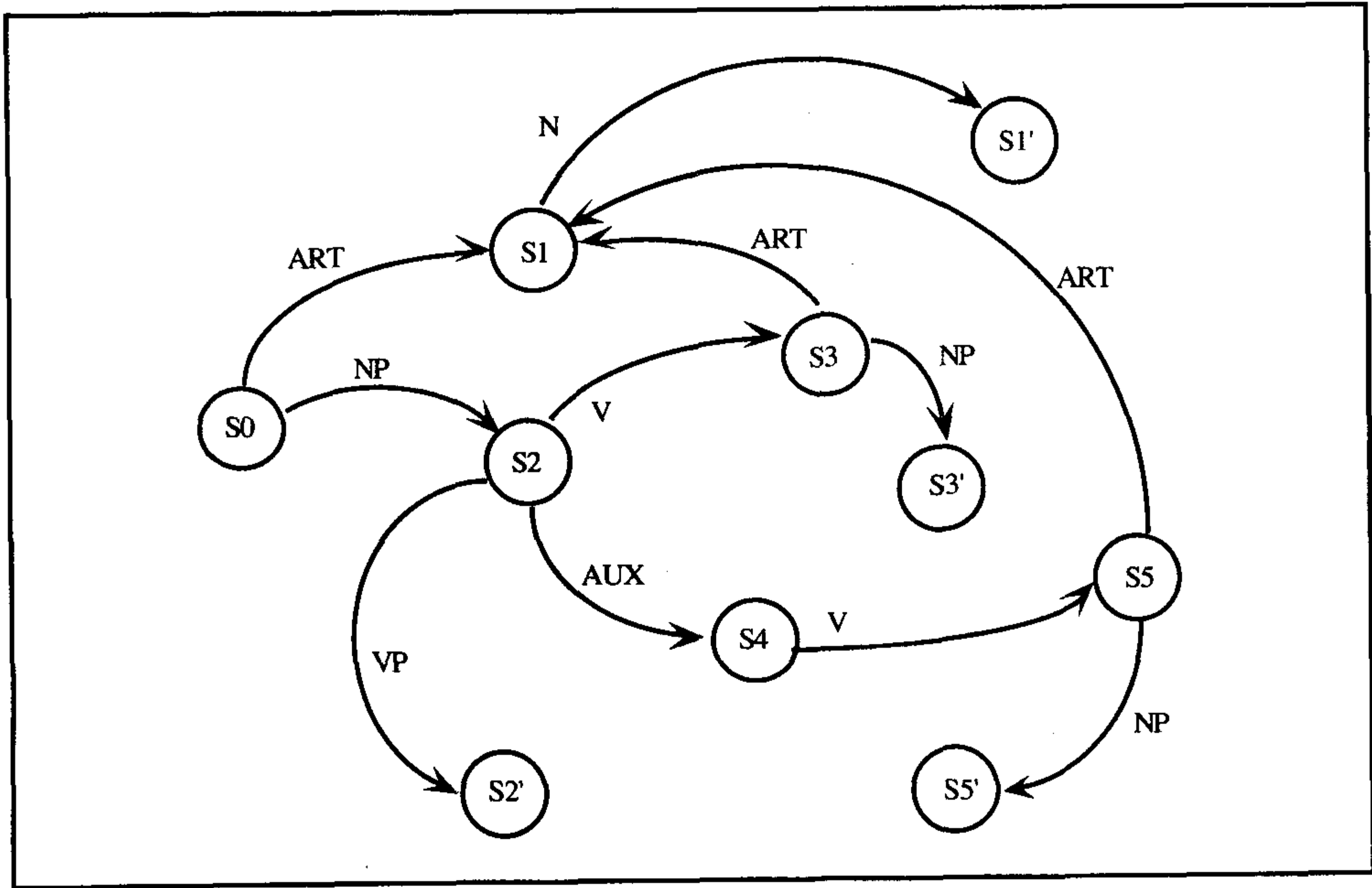


图 6.5 语法 6.1 派生出的状态转移图

6.2.2 移进归约句法分析器

这些状态可以用来控制句法分析器。在句法分析过程中,我们需要维护两个栈结构。其中一个分析栈,包含句法分析的状态(即图 6.5 中的节点)和语法符号;另一个是输入栈,包含输入的句子和一些语法符号。在任何时候,句法分析器均根据分析栈顶端状态的信息进行处理。

状态可以按照如下方式进行解释。有些状态只包含一条点号在最右边的简单规则,比如 S2':

$S \rightarrow NP VP \circ$

它表示的是,句法分析器应当依据该规则改写分析栈顶端的符号,这就叫归约(reduce)操作。新派生出来的符号(本例中的 S)压到输入栈顶端。

对于其他不包含完结规则的状态,我们通过状态转移图对其进行解释。如果输入栈顶端的符号与某条边匹配,那么,它和新状态(在边的末尾)会压到分析栈的顶端,我们称之为移进(shift)操作。根据对这些状态的解释,可以构建一张表,我们通常称之为预测表(oracle)。在各种不同的情况下,预测表都会告诉句法分析器应该如何去做。语法 6.4 的预测表如图 6.6 所示。对于每一个状态和可能的输入,它都指出了操作类型和下一个状态。不管下一个输入是什么,都可以直接使用归约操作,而接受操作只有在输入栈为空的时候(即,下一个符号为空符号 ϵ)才有可能执行。根据预测表进行句法分析的算法如图 6.7 所示。

状态	顶端输入符号	操作	压到
S0	ART	移进	S1
S0	NP	移进	S2
S0	S	移进	S0'
S0'	ϵ	成功	----
S1	N	移进	S1'
S1'	-----	用规则 2.2 归约	----
S2	V	移进	S3
S2	AUX	移进	S4
S2	VP	移进	S2'
S2'	-----	用规则 2.1 归约	----
S3	ART	移进	S1
S3	NP	移进	S3'
S3'	-----	用规则 2.4 归约	----
S4	V	移进	S5
S5	ART	移进	S1
S5	NP	移进	S5'
S5'	-----	用规则 2.3 归约	----

图 6.6 语法 6.4 的预测表

该算法使用了如下信息：

- $Action(S, W)$ ——将一个状态和一个输入符号映射为某个操作的函数，输出值范围是移进 (shift)、归约 (reduce i) 或者接受 (accept)。
- $GoTo(S, W)$ ——将一个状态和一个输入符号映射为一个新状态的函数。
- 分析栈的表示形式为 $(S_n C_n \dots S_1 C_1 S_0)$ 。其中， S_i 是句法分析的状态， C_i 是句子的成分。
- 输入栈的表示形式为 $(W_1 \dots W_n)$ ，其中， W_i 是一个成分符号或词语。

句法分析器不断地执行以下步骤，直到成功返回或失败返回为止：

1. 如果 $Action(S_n, W_1) = Shift$ ，而且 $GoTo(S_n, W_1) = S$ ，就从输入栈中移走 W_1 ，将它压到分析栈中，然后将 S 压到分析栈中。最后，栈的结果如下所示：

分析栈： $(S W_1 S_n C_n \dots S_1 C_1 S_0)$

输入栈： $(W_2 \dots W_n)$
2. 如果 $Action(S_n, W_1) = Reduce\ i$ ，并且语法规则 i 的右边有 n 个组成成分，则从分析栈中移出 $2n$ 个元素，并将规则 i 左边的元素压到输入栈中。比如，规则 i 是 $NP \rightarrow ART\ N$ ，则新状态为：

分析栈： $(S_{n-2}\ C_{n-2} \dots S_1\ C_1\ S_0)$

输入栈： $(NP\ W_1 \dots W_n)$
3. 如果 $Action(S_n, W_1) = Accept$ ，则句法分析器成功。
4. 如果 $Action(S_n, W_1)$ 无定义，则句法分析器失败。

图 6.7 移进归约句法分析器的分析算法

现在，让我们看看如何分析句子“The man ate the carrot.”(那个男人吃了胡萝卜。)句法分析器的初始状态为：

分析栈

(S0)

输入栈

(The man ate the carrot)

依据当前状态 S0 以及输入符号 ART(词语“the”的词类),在图 6.6 的表中查找相关表项。你可以看到需要进行移进操作,并且转移到状态 S1。因此,可以得到:

分析栈	输入栈
(S1 ART S0)	(man ate the carrot)

依据当前状态 S1 以及输入符号 N,查找相关表项。你可以看到需要进行移进操作,并且转移到状态 S1。因此,可以得到:

分析栈	输入栈
(S1' N S1 ART S0)	(ate the carrot)

依据当前状态 S1',查找相关表项。需要采用规则 2.2 对它们进行归约,即分别从分析栈中移出 S1',N,S1 和 ART,将 NP 压到输入栈中,最终结果为:

分析栈	输入栈
(S0)	(NP ate the carrot)

依据当前状态 S0 以及输入符号 NP,再次查询分析表。现在需要进行移进,并转移到状态 S2,即:

分析栈	输入栈
(S2 NP S0)	(ate the carrot)

接着,剩下的三个词语都需要移进并转移到新的状态。最终,句法分析器转入如下状态:

分析栈	输入栈
(S1' N S1 ART S3 V S2 NP S0)	()

在状态 S1'中,需要采用规则 2.2 进行归约。因此,将 N 和 ART 出栈(这样,状态 S1 和状态 S1' 也同样出栈),从而生成下面的状态:

分析栈	输入栈
(S3 V S2 NP S0)	(NP)

现在,又回到了状态 S3。当前的输入为 NP,转移到状态 S3'后,采用规则 2.4 进行归约,可以生成如下状态:

分析栈	输入栈
(S2 NP S0)	(VP)

最后,需要从状态 S2 转移到状态 S2',并且采用规则 2.1 进行归约,得到:

分析栈	输入栈
(S0)	(S)

从这个状态再转移到状态 S0',最终会到达某个位置并接受这个句子。

对这个句子的句法分析已经完成了。在此期间,我们没有错误地尝试语法中的任何规则,而且没有构建任何一个不在最终分析结果中的成分。

6.2.3 移进归约句法分析器与歧义性

该算法能够推迟决定到底应该使用哪一条规则,因此,移进归约句法分析器的效率很高。

举例来说,如果采用下面的规则来分析 NP:

1. NP \rightarrow ART N REL-PRO VP
2. NP \rightarrow ART N PP

那么,自顶向下的句法分析器不得不产生两条新的活动边。如果当前输入为 ART,这两条边都要继续扩展。但是,移进归约句法分析器只需要一个状态就能表示这两种情况,即:

NP1: NP \rightarrow \circ ART N REL-PRO VP
NP \rightarrow \circ ART N PP

如果 ART 出现,那么下一个状态将是:

NP2: NP \rightarrow ART \circ N REL-PRO VP
NP \rightarrow ART \circ N PP

这样,ART 可以被识别出来,我们不用知道到底采用了哪一条规则可以直接将它移进分析栈中。类似地,如果状态 NP2 处出现了 N,则下一个状态是:

NP3: NP \rightarrow ART N \circ REL-PRO VP
NP \rightarrow ART N \circ PP
PP \rightarrow \circ P NP

现在,根据 NP3,最终可以区分出这两种歧义情况。如果下一个出现的是 REL-PRO,那么状态为:

NP4: NP \rightarrow ART N REL-PRO \circ VP
VP \rightarrow \circ V NP

最后,可以采用规则 1 进行归约。如果找到 P,就可以转移到不同的状态。接着,可以构建 PP 短语,根据规则 2,我们可以采用 PP 进行归约。因此,直到有足够的信息可用来选择规则,句法分析器才会做出决策。

6.2.4 词汇的歧义性

我们可以采用这种推迟决策的能力来处理词汇的歧义性问题,只须对句法分析过程进行简单地扩展。在词汇移进的时候,需要划分出词条的语法类别(即,“carrot”在移进过程中转化为 N)。与此不同的是,现在可以将歧义词语都移进分析栈中,推迟判断它们的词类,直到需要做出与它们相关的归约为止。

为了完成这种扩展,必须扩充状态的数目,并涵盖歧义处理的状态。例如,如果“can”可以作为 V 或者是 AUX,那么,在状态 S2 中,预测表不能决定一个惟一的操作——如果它是 V,就要移进到状态 S3;如果它是 AUX,就要移进到状态 S4。不过,我们还可以从 S2 产生一个新的状态来表示这种歧义性,它可以同时覆盖两种可能的结果。这个新状态将是状态 S3 与状态 S4 的结合,即:

S3-4: VP \rightarrow AUX \circ V NP
VP \rightarrow V \circ NP
NP \rightarrow \circ ART N

在这个例子中,下一个输入就可以消除歧义。如果下一个输入为 V,就要转移到状态 S5 (就和从状态 S4 中转出一样)。如果下一个输入是 ART,就要转移到状态 S1;如果是 NP,就要转移到状态 S3' (和从状态 S3 中转出一样)。这样,新状态就可以长时间地保持这种歧义,直至后面的词语能够解决这一问题为止。当然,下一个词语通常也可能存在歧义,新状态的数目会

变得非常大。只要语法不存在歧义性(即句法分析完成后,句子只可能有一种解释),这个方法就可以根据需要来推迟决策。(实际上,对于那些了解自动控制理论的人来说,这个过程仅仅是为非确定性有限状态自动机建立了一个确定性的模拟结构。)

6.2.5 有歧义的句法分析状态

不过,就目前的情况看,句法分析器还有一些无法处理的其他歧义形式。前缀歧义是最简单的一个例子。前缀歧义指的是一条规则恰好是另一条规则的前缀,比如下面的这些规则:

3. $NP \rightarrow ART\ N$
4. $NP \rightarrow ART\ N\ PP$

运用这两条规则,可以生成一个新的分析状态,其中包含两个点号规则,即:

- $$NP \rightarrow ART \circ N$$
- $$NP \rightarrow ART \circ N\ PP$$

如果下一个输入是 N,那么将转移到一个新状态,即:

- NP5: $NP \rightarrow ART\ N \circ$
 $NP \rightarrow ART\ N \circ PP$
 $PP \rightarrow \circ P\ PP$

但是,现在出现了一个问题。如果下一个输入的是 P,最终将构建出一个 PP,并且返回到状态 NP5。但是句法分析器并不能决定它到底应该采用规则 3 进行归约,以后再附着 PP,还是应该移进 PP(然后采用规则 4 进行归约)。当然,在任何合理的英语语法中,这两种选择都可能产生句子可接受的分析结果。我们有两种策略来处理这个问题。第一种策略是维持一个确定型的句法分析器,并且接受它可能错误分析某些句子的事实。这里的目标是模仿人的句法分析过程,即只有在面对那些会给人带来麻烦的句子时,句法分析才失败返回。最新的观点表明下面的启发式规则有助于获得更符合人类直觉的解释,即:

- 在移进操作和归约操作之间,应当更倾向于前者。
- 解决归约之间的冲突时,长规则优先(即,从栈中调用符号最多的规则)。

使用这些策略,可以构造一个确定型的句法分析器。该句法分析器选择最优的规则,而忽略其他的规则。有人认为,第一条启发式规则对应于右关联优先原则,而第二条启发式规则对应于最小附着优先原则。我们可以举出具有说服力的例子来证实这一点。但是,需要记住的是,这些策略本身以及示例都和所使用的语法形式密切相关。

第二种歧义状态的处理方法是放弃确定性要求,再引入搜索过程。在这个方法中,你可以考虑每种可能的解释,既可以使用带回溯的深度优先搜索,也可以使用宽度优先搜索同时保留多种解释。这两种方法都可以生成高效的句法分析器。深度优先方法还可以与优先级选择策略相结合,从而保证首先尝试优先级高的解释。

6.3 确定型句法分析器

确定型句法分析器建立起来后,完全可以依靠分析状态的匹配来指导其操作。它允许的不仅仅是移进操作和归约操作,而是一组在输入栈上更加丰富的操作集。在这里,输入栈称为缓冲区。这种分析器不是将成分移进分析栈并等待以后的归约操作将其消除,而是将缓冲区

中的成分附着到其父成分之中,最后渐增式地生成各种不同的语法成分,这种操作类似于我们前面介绍过的特征赋值。例如,移进归约句法分析器会将 NP 移入栈中并等待以后用规则 $S \rightarrow NP VP$ 对其进行归约。与此不同的是,确定型句法分析器会在栈顶建立成分 S,然后将 NP 附着其中。具体来说,这种分析器有如下操作:

- 创建:在分析栈中创建一个新节点(即将符号压栈)
- 附着:将一个输入成分附着到分析栈的顶端节点中
- 移除:移除分析栈的顶端节点,并且将它置入缓冲区中

移除操作允许分析器重新检查一个完整的成分,而且会在分析栈更高层次的成分中为其赋予一个角色。这个技术使得有限向前看方法(limited lookahead technique)具备惊人的强大功能。

为了对这些操作有一个感性的认识,我们现在考虑一下图 6.8 所示的情境,它可能出现在句子“The cat ate the fish.”(猫吃了鱼。)的分析之中。假定我们已经分析了第一个 NP,并且将分析栈上 S 成分的 SUBJ 特征赋值给该短语。在前面介绍过的那些操作就可以完成这个分析。需要注意的是,究竟应该采用何种操作的实际机制,目前还没有讨论到。不过,我们先给出操作的效果,以便大家对这种数据结构有一些直观的认识。下面的操作为:

附着到 MAIN-V

该操作会从缓冲区中移去“ate”的词条,并将分析栈上 S 成分的 MAIN-V 特征赋值给“ate”。接下来的操作是:

创建 NP

我们会将一个空的 NP 成分压入分析栈,建立如图 6.9 所示的状况。

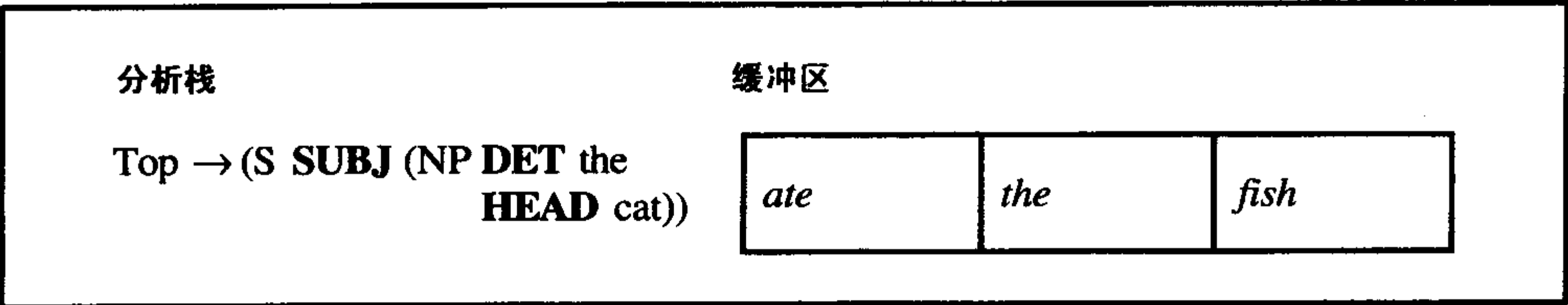


图 6.8 分析器中的某种状况

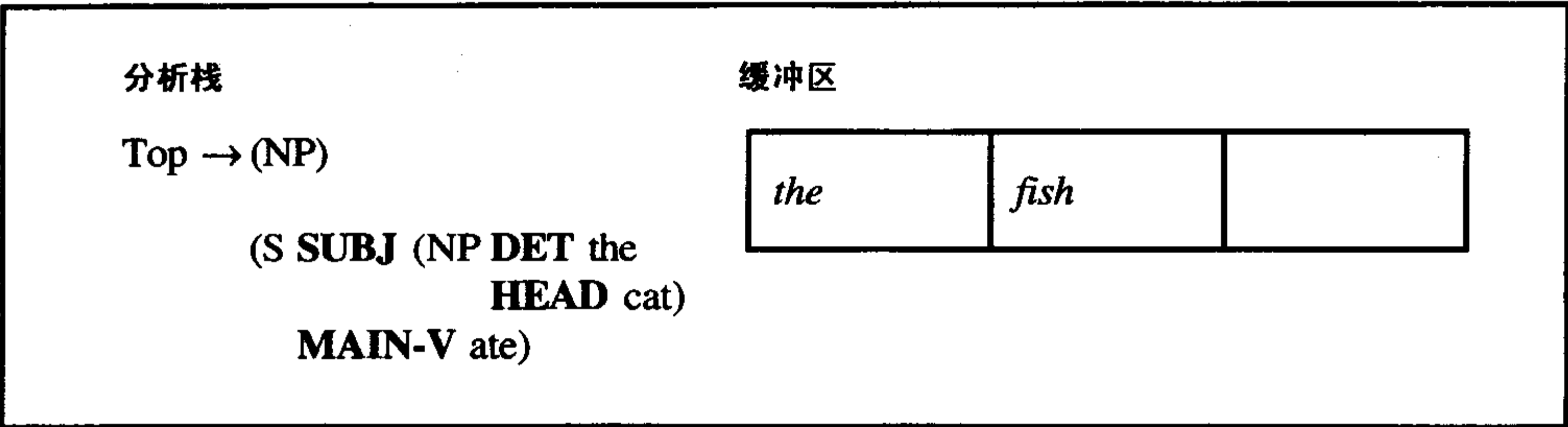


图 6.9 创建 NP 成分之后的状况

下面的两个操作是:

附着到 DET

附着到 **HEAD**

它们会在词条“the”和“fish”的基础上,成功地构造出 NP 成分。现在,输入缓冲区为空。
接下来的操作为:

Drop(移除)

即从分析栈中弹出 NP 成分,然后将其压回到缓冲区,从而建立如图 6.10 所示的状况。

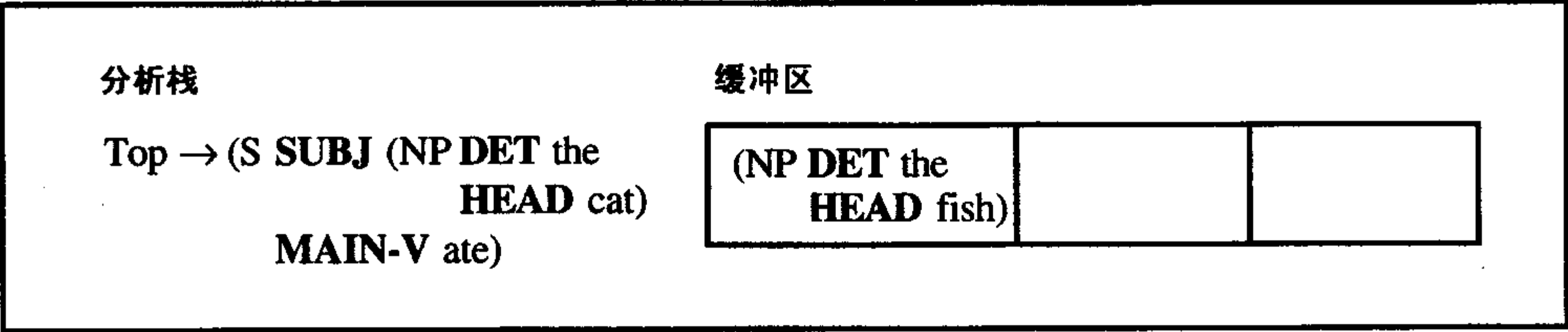


图 6.10 移除操作之后的状况

现在,分析器开始建立最终的结构,采用的操作如下:

附着到 **OBJ**

它将 NP 从缓冲区中移走,并且赋给 S 成分的 **OBJ** 槽。

在处理自然语言的通用性方面,我们可以证实还有三个操作非常有用:

- 交换:交换缓冲区中前两个位置上的节点
- 插入:在指定的缓冲区槽中插入特定的词项
- 插入:在缓冲区的第一个槽中,插入一个空的 NP 节点

每一条规则都有一个模式,该模式需要在缓冲区上进行特征检查,并以此来决定规则的适用性。规则都组织成包的形式,在分析过程中,这些包可以激活,也可以失效。还有另外一些操作,这些操作通过包的选择来改变分析器的状态。具体地说,这些操作有:

- 激活:激活包(即,包中所有的规则都要用来解释下一个输入)
- 使失效:使包失效

活动包与分析栈顶端的符号是关联在一起的。如果创建了一个新成分(即压栈的成分),所有与原来栈顶成分关联的活动包均会失效,除非原来栈顶的成分再次出现在栈的顶端。因此,移除操作总要让与栈顶成分相关联的规则失效。包所起的作用类似于移进归约分析器中的状态,不同的是,同一时刻可以有多个包处于活动状态。可以同时激活两个包,因此没有必要生成联合包来处理词语的歧义性问题。

语法 6.11 分析处理了助词结构,现在,我们来看看其中给出的示例规则。每条规则的模式表示的都是特征测试,规则适用的前提是,它在缓冲区每个位置上的模式测试都必须成功。因此,模式 $\langle = \text{AUX}, \text{HAVE} \rangle \langle = \text{V}, \text{pastprt} \rangle$ 为真的前提是,缓冲区的第一个语法成分是 AUX 结构,其 ROOT 特征值为 HAVE;同时第二个成分是 V 结构,其 VFORM 特征值为 pastprt(过去分词)。我们采用与每个规则关联的优先级在相互冲突的规则之间进行取舍。数字越小,优先级别越高。举例来说,规则 6 的模式 $\langle \text{true} \rangle$ 总可以匹配成功。但是它的优先级较低,因此,只要

还有其他的规则可以适用,分析器就永远不会使用它。它只在其他规则都不适用时才使用,从而完成对助词与动词结构的句法分析。

模式	操作	优先级
Packet BUILD-AUX:		
1. <=AUX, HAVE> <=V, pastprt>	附着到 PERF	10
2. <=AUX, BE> <=V, ing>	附着到 PROG	10
3. <=AUX, BE> <=V, pastprt>	附着到 PASSIVE	10
4. <=AUX, +modal> <=V, inf>	附着到 MODAL	10
5. <=AUX, DO> <=V, inf>	附着到 DO	10
6. <true>	移除	15

语法 6.11 BUILD-AUX 包的规则

图 6.12 给出了一个分析状态,其中 BUILD-AUX 为活动状态。栈顶端的 BUILD-AUX 包是活动的,它包含了一个 AUXS 结构。一旦 AUXS 成分移入缓冲区中,接下来就是 S 结构,PARSE-AUX 包和 CPOOL 包将成为活动的包。

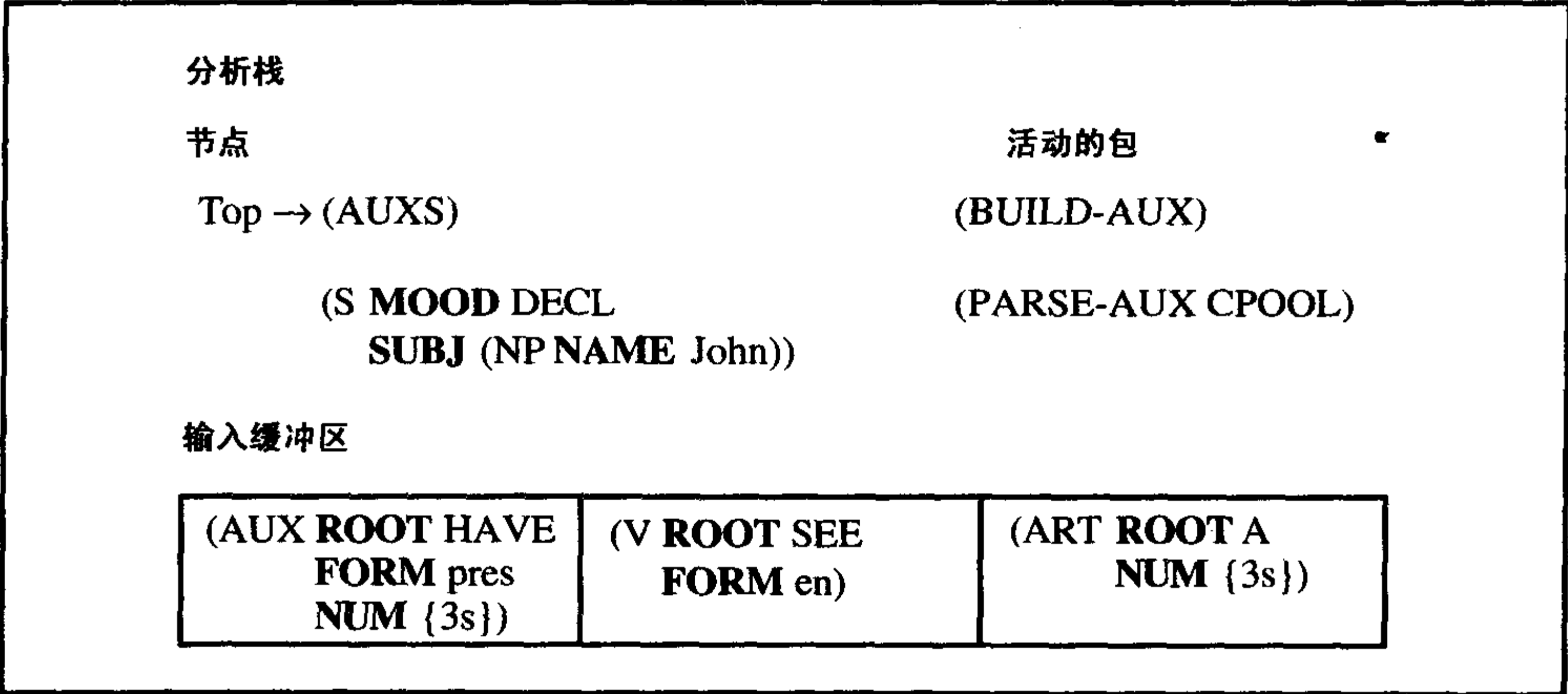


图 6.12 句法分析器的一个典型状态

图 6.11 中给定了当前的状况和规则,句法分析器的下一个操作将取决于哪个规则可以匹配而定。规则 1 和规则 6 均可适用,规则 1 的优先级较高,所以我们最终选择规则 1。应用规则 1 中的操作,可以得到如图 6.13 所示的状态。现在,我们再次应用 BUILD-AUX 中的规则。这次,只有规则 6 可以适用。因此,下一个操作是移除,生成的状态如图 6.14 所示。

在这一阶段,PARSE-AUX 包和 CPOOL 包中的规则是活动的,它们相互竞争以决定分析器下一步的操作(最后将 AUXS 结构附着到 S 结构之中)。

向前看规则受到一定的限制,它最多只能检查缓冲区的前三个单元,生成 NP 子成分时例外。如果 NP 开始于缓冲区的第二个或第三个位置,那么分析它的时候,直接把它当做第一个缓冲单元来处理,这就叫注意力转移(attention shifting)。在缓冲区三单元约束中,注意力转移是惟一的例外。尽管存在这种情况,我们仍然只允许规则查看三个缓冲位置,但起始位置可能会有所转移。注意力转移也是受限的,任何情况下,规则都不能查看前五个位置之外的单元。

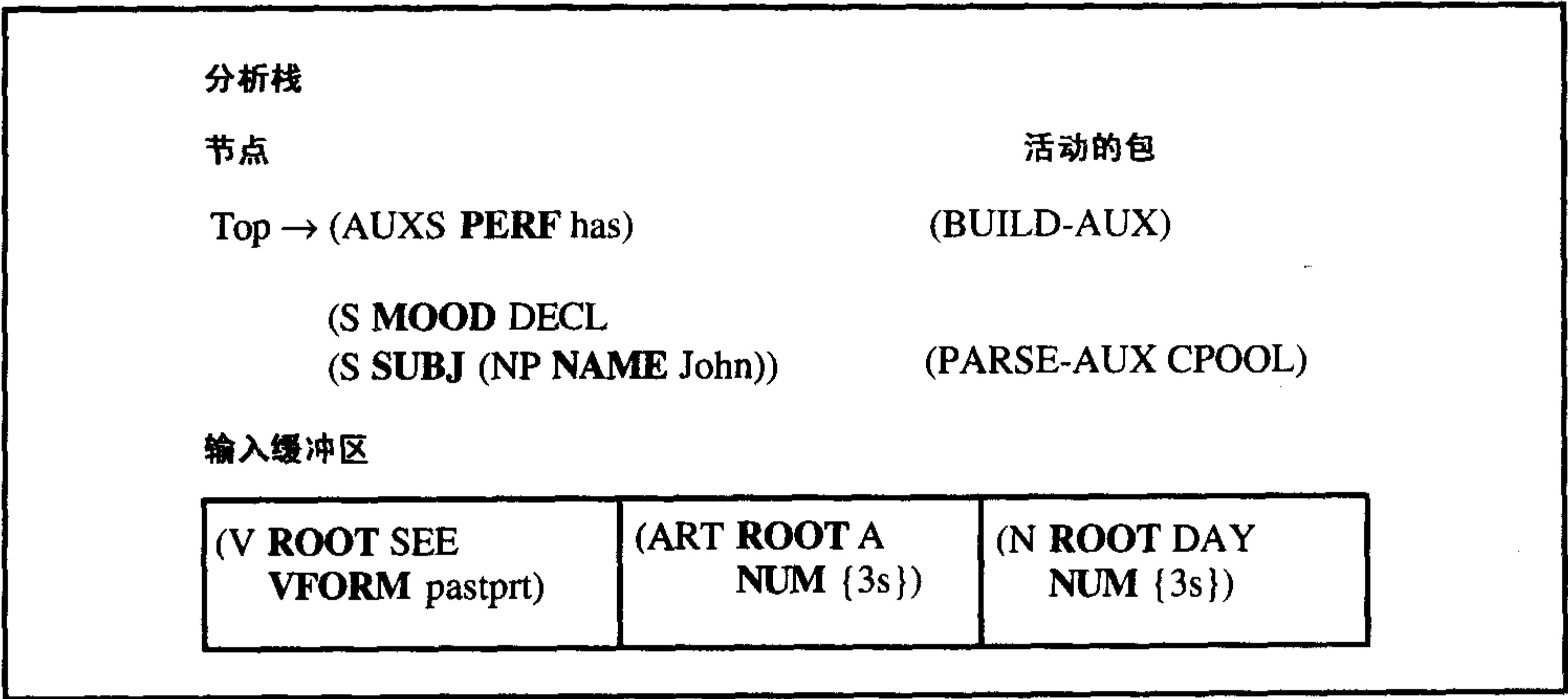


图 6.13 应用规则 1 之后的状态

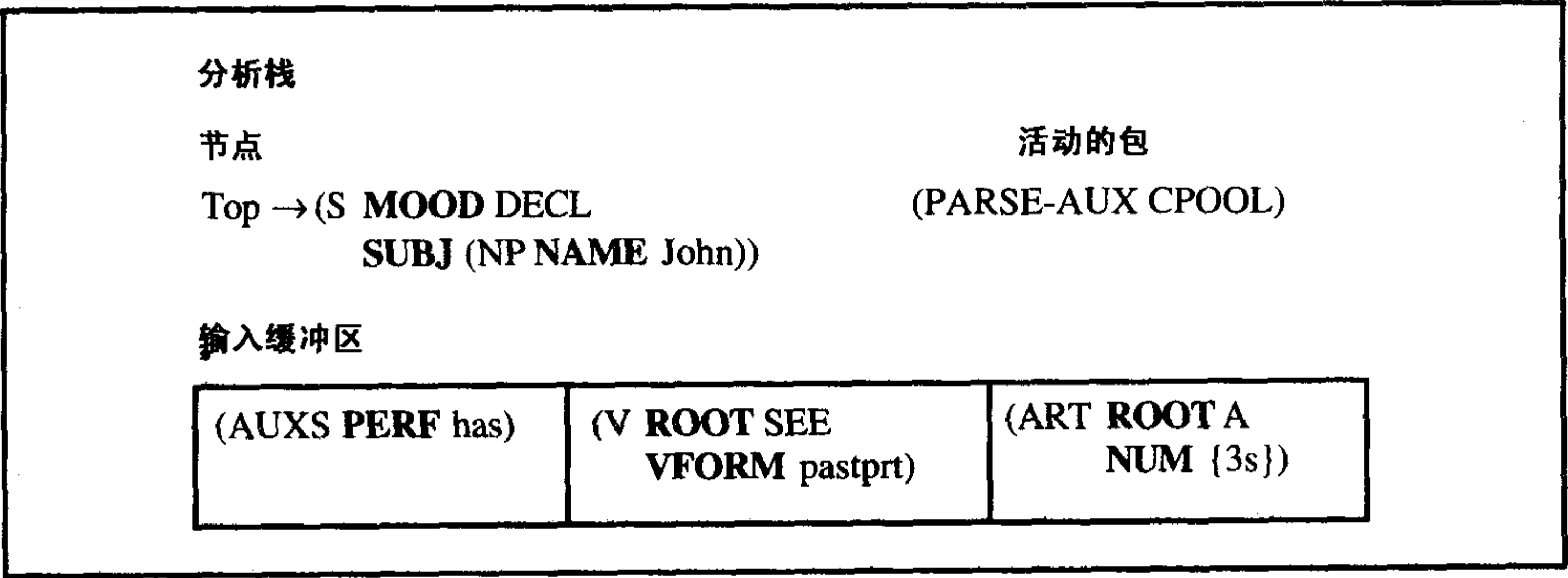


图 6.14 移除操作之后的状态

这种句法分析器更有趣的一点是,通过操作输入缓冲,采取了很巧妙的方式来刻画许多语言规律。例如,在这里,我们也探讨一下需要对分析断言句的语法进行怎样的扩展,才能使之可以分析一般疑问句。在第 5 章中介绍的处理一般疑问句的方法是:在语法中增加一些规则来处理初始的助词和名词短语,然后再回过头来与陈述句语法相结合。实际上,现在的句法分析器也需要重新使用那些初始的主助规则。具体地说,采用交换操作重新调整缓冲区中 NP 和 AUX 成分的顺序,这样能够更直接地反映出主助倒置的直观感觉。

6.3.1 心理学上的有效性说明

向前看方法以及这种分析器的确定性特征存在一定的限制,因此,需要仔细地研究其局限性,同时也要考虑它的覆盖率。一些研究者认为,机制自身的局限性主要是语言形式上存在各种约束,比如第 5 章给出的复杂 NP 约束。如果不对语法加上这些限制,而采用其他方式,那么,这种分析器会由于自身的局限而不能正常运转。

由于向前看方法的限制,在分析处理整个句子之前,这种机制必须先进行某些结构的分析。在某些情况下,句子开头部分可能会引起错误,从而使得句子不可能分析出结果。前面提到的花园幽径型句子就是这样的例子。这个现象有趣的一点是它提供了一个很具体的想法,而且这种想法还可以通过实验方法进行验证。具体地说,你可以问“对分析器有困难的句子结

构是否也会让人类同样感到棘手?”尽管答案还不清楚,但我们可以提出这样的问题,这本身就表明对这一框架可以进行实验性的探索研究。

在本质上,这种心理学理论认为,任何一个存在歧义的句子,而且歧义窗口多于三个成分,那么,这样的句子可能会让普通读者感到困扰。需要注意的是,向前看的是三个成分,而不是三个词语。因此,歧义可能会保留很长的时间而不会带来困扰。例如,在下面的两个句子中,前7个词语完全一样:

Have the students who missed the exam take it today.

(让那些错过了考试的学生今天补考。)

Have the students who missed the exam taken it today?

(那些错过了考试的学生今天补考了吗?)

因为有6个词语都在同一个名词短语中,所以,要判断它是祈使句或普通句,还是一般疑问句,不用考虑三个以上的成分。因此,这个句法分析器将到达某种状态,在这个状态下,可以采用下面的两个测试轻易地做出判断。具体的测试如下:

< = have > < = NP > < = V, **VFORM** = base > → 祈使句

< = have > < = NP > < = V, **VFORM** = pastprt > → 一般疑问句

另一方面,我们给出了另外一种句子,如下所示:

Have the soldiers given their medals by their sweethearts.

(让这些战士接受自己爱人给他们颁发的奖章。)

在这里,只用三个成分并不能消除歧义。这时候,分析器和大多数人一样,在开始的时候都会将这个句子误认为是一个错误形式的一般疑问句,而不会将其视为正确的祈使句,其含义对应于“Have their sweethearts give the soldiers their medals.”(让这些战士的爱人给他们颁发奖章。)

6.4 高效的歧义表示技术

减少歧义的另外一种方法是重新定义需要的输出结果以改变游戏的规则。比如,句子中相当多的歧义都是由类似于附着歧义(attachment ambiguity)这样的问题引起的,也有的是由并列关系结构引起的,并列关系通常会在同一重要的信息上存在许多不同的解释。动词短语“saw the man in the house with a telescope”存在介词短语附着歧义,图6.15给出了不同解释的示例。不同的附着可以生成5种解释,每种都对应于不同的语义解释。从图6.16中,可以看到名词修饰名词的歧义,比如“pot holder adjustment screw”,它所采用的语法分析如下:

5. NP → N2

6. N2 → N

7. N2 → N2 N2

这些过程中的复杂因素相互作用,因此,包含两个介词短语的VP和由四个名词序列组成的NP会有25种解释,其他情况也是类似的。采用一个合理的语法,一个长度中等的句子,比如包含12个词的句子,居然会有1000多种不同的结构解释!显然,我们需要引入一些技术来帮助解决这样的困难。

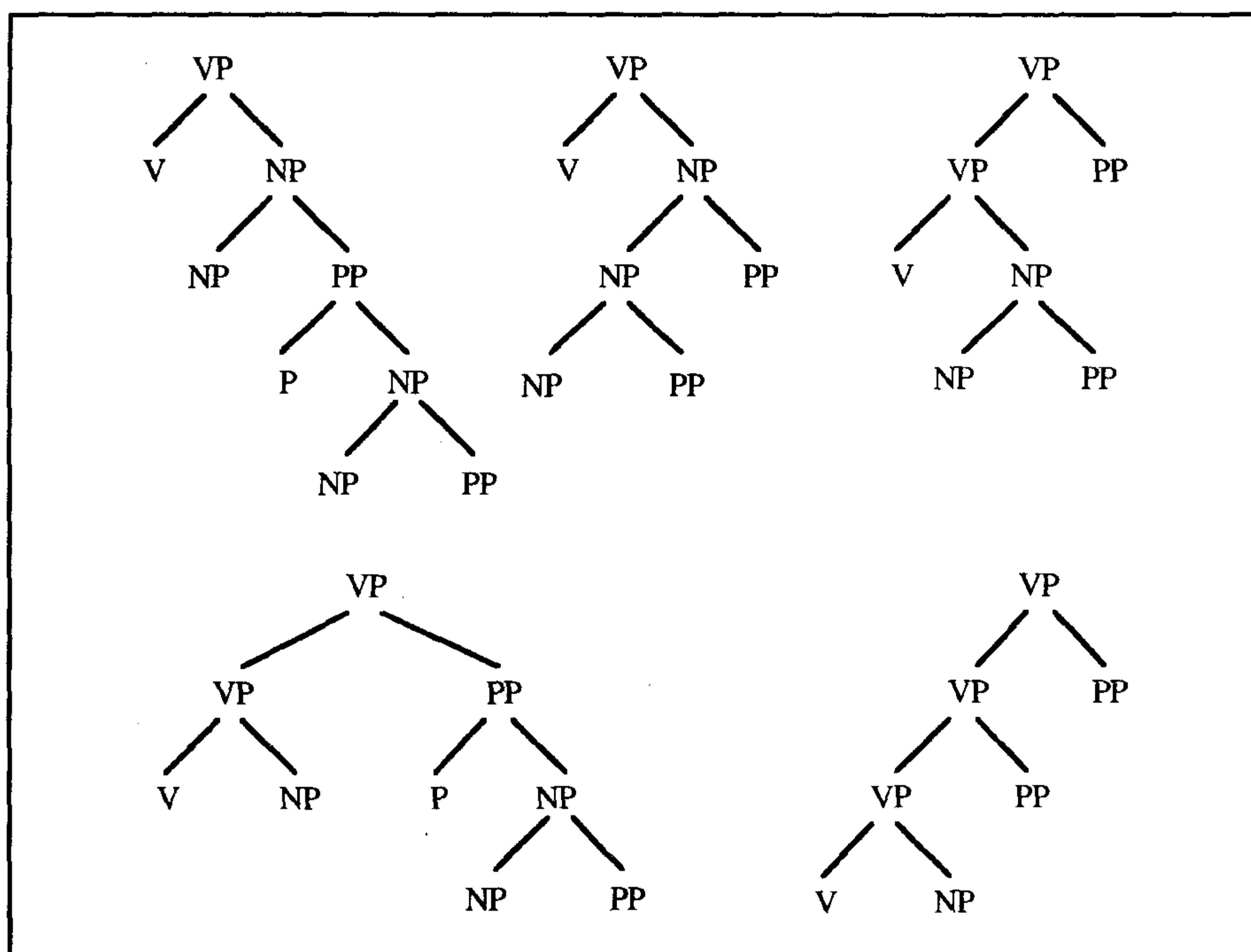


图 6.15 “saw the man in the house with a telescope”的 5 种解释

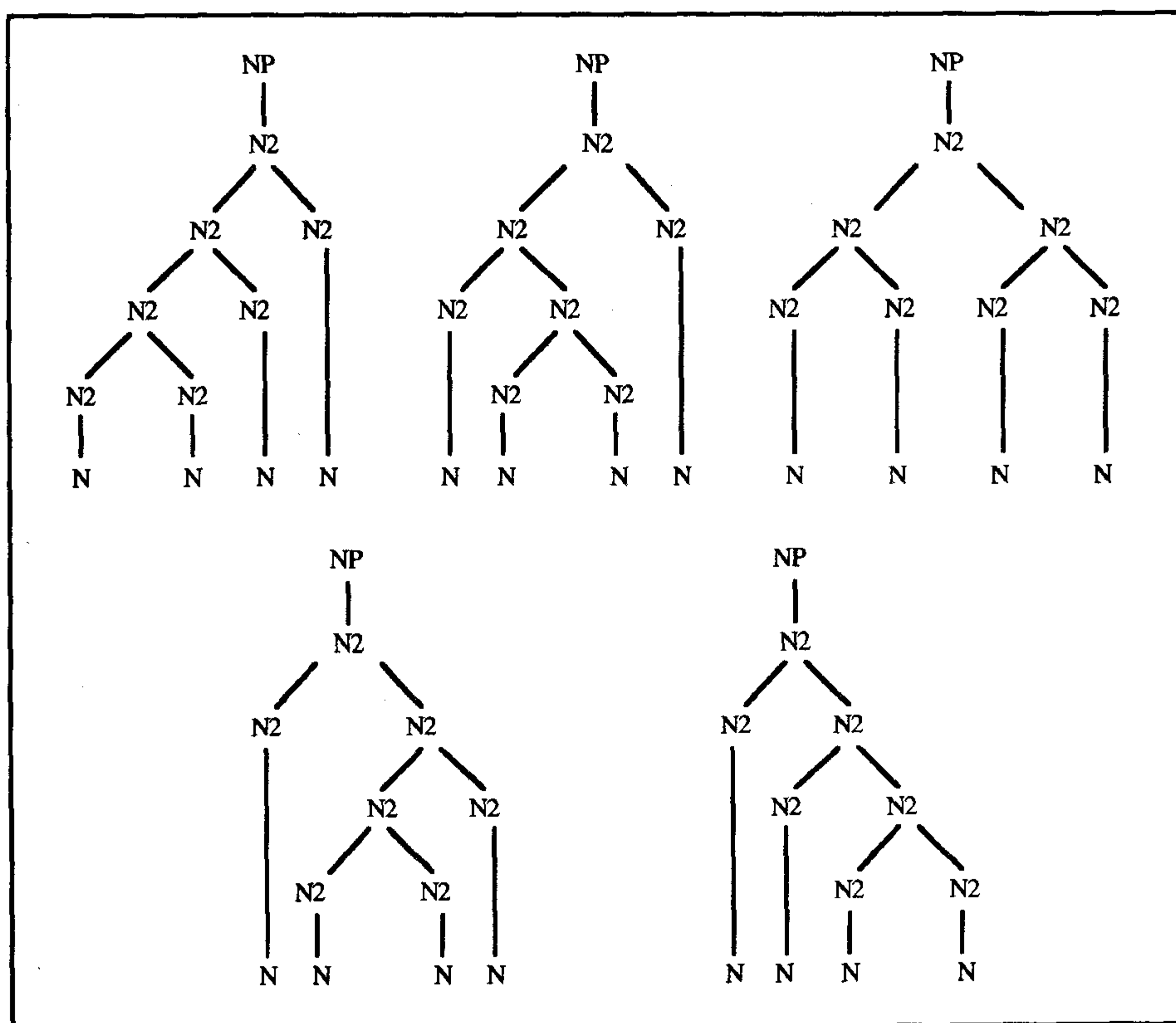


图 6.16 “pot holder adjustment screw”的 5 种解释

在这一部分,将简略地探讨表示大量解释结果的高效方法。实际上,到目前为止,我们一直在使用的 chart 数据结构已经往这一目标迈出了非常重大的一步,因为它允许在所有解释之间共享语法成分。例如,图 6.15 给出了 PP 附着的解析结果,如果采用 chart 图表示,其结果见图 6.17。我们对 chart 图中的每一项进行编号,并将每个子成分列在括号中。例如,对应于“the man in the house”(房子里的男人)的 NP 是 11 号成分,并且包含子成分 2 和子成分 7。在图 6.15 中,我们使用了 32 个非词汇节点,而 chart 分析表只用了 21 个非词汇节点就表示了同样的 5 种解析结果。这种节约下来的资源相当可观,而且会随着歧义数目的增长而越来越多。但是,在处理这个问题方面,chart 图常常达不到要求。

VP						21(1,13)					
VP						20(1,12)					
VP						19(16,8)					
NP						18(15,8)					
VP						17(14,10)					
VP						16(14,7)					
VP						15(1,11)					
VP						14(1,2)					
NP						13(11,8)					
NP						12(2,10)					
NP						11(2,7)					
PP						10(3,9)					
NP						9(4,8)					
PP						7(3,4)					
PP						8(5,6)					
V	1	NP	2	P	3	NP	4	P	5	NP	6
saw		the man		in		the house		with		a telescope	

图 6.17 “saw the man in the house with a telescope”解释的 chart 图

另外一种使用到的技术叫做压缩(packing),这种技术利用了可以在图 6.17 中看到的某种性质,即许多语法成分类型相同,涵盖的输入也一样。对于不带特征的句法分析器来说,每个成分只要是作为规则中的子成分,处理起来都是一样的。对于各种来自于同一输入而类型不同的成分,压缩 chart 形式只存储其中的一个成分,找到的其余成分均分解到这个现有成分之中。图 6.18 给出了对同一个句子采用压缩技术生成的 chart 图,它只使用了 15 个非词汇节点。压缩可以高效地生成解释结果的表示形式,并且不会丢失信息。不过,对于那些带特征的语法,情况就复杂得多。具体来说,如果两种 VP 的解释涵盖相同的输入而特征值不同,那么,这两种解释就不能合并,因为它们在组合起来生成更大成分的时候,处理方式并不相同。可以对

特征检测进行修改,只有当每个可能的成分均满足要求时,才允许它们成功合并。但是,chart图可能会过度生成解释结果,以至于最后不得不重新分析所有可能的解析结果,并进一步验证它们是否有效。

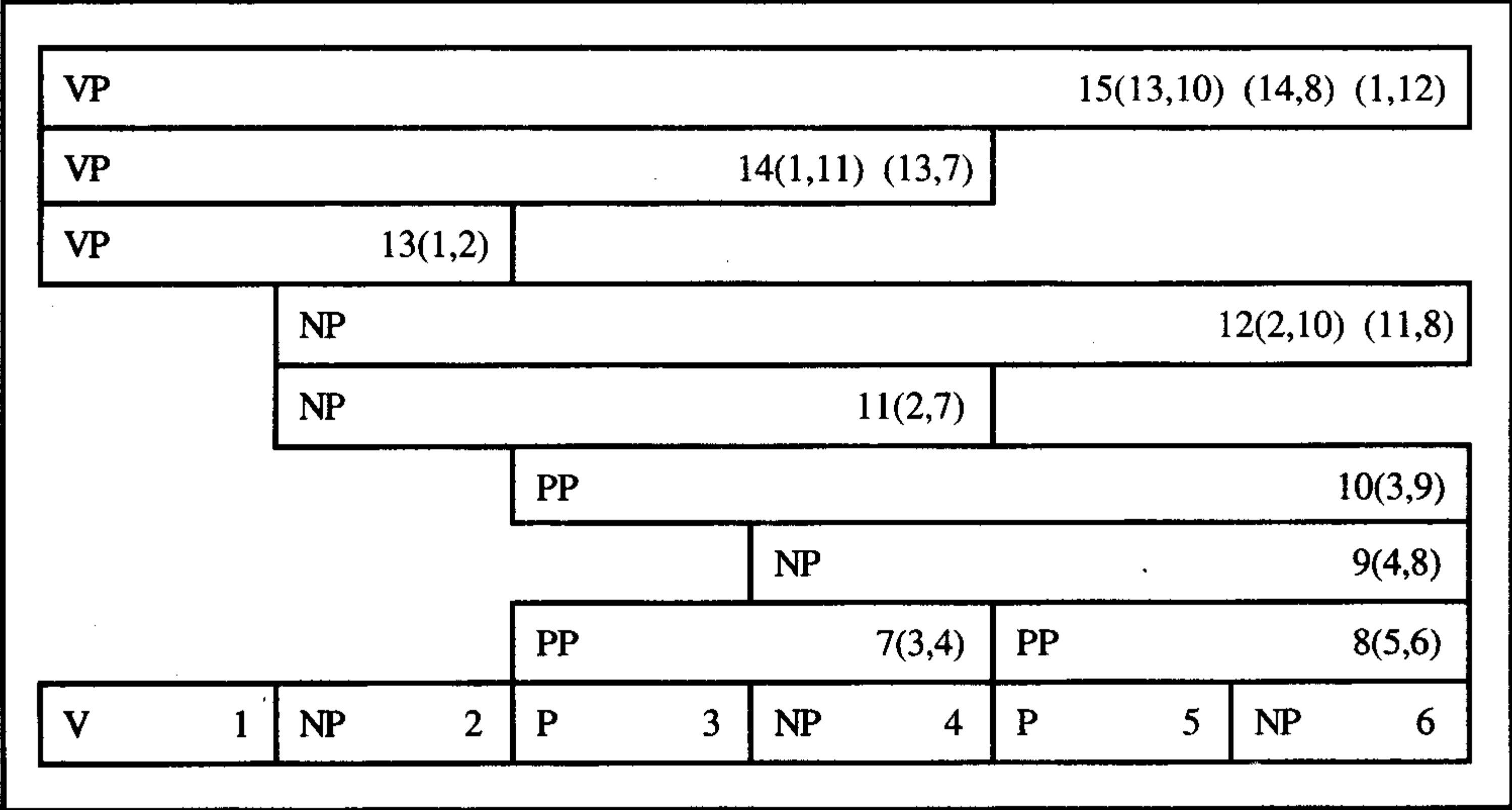


图 6.18 “saw the man in the house with a telescope”解释的压缩 chart 图

另一个减少解释结果的方法是对语法进行修改,使歧义都变成语义歧义,而不是句法歧义。例如,对于名词修饰名词的歧义消解,可以不用前面提到的规则 5、规则 6 和规则 7,它们都在名词修饰语上引入了二元结构。只须使用一条规则,该规则允许存在平面结构上的名词修饰语序列。在扩展的语法中,如果允许 Kleene + 操作^①,下面这条规则就足够了:

$N2 \rightarrow N +$

符号 $N +$ 允许符号 N 一次或多次重复。如果没有“+”号,这一语法需要采取一系列的规则,直至达到某个允许的最大修饰语数目为止。这些规则有:

- $N2 \rightarrow N$
- $N2 \rightarrow N N$
- $N2 \rightarrow N N N$
- $N2 \rightarrow N N N N$

这样的语法看起来更加复杂,但分析起来效率更高,因为已经消除了结构上的歧义。举例来说,名词短语“pot holder adjustment screw”就只存在一种解释。

也有人提出采用类似的方法来解决附着歧义问题。我们不生成所有的解释,而是仅仅生成一种解释结果,即 PP 覆盖范围最广的解释,有时候,也称之为经典解释 (canonical interpretation)。然后,需要设计语义解释器,以便能够附着句法分析树中较低层的 PP,基于语义的约束似乎就可以正确地实现这个目标。

^① 即正则表达式中的“+”号,表示前面的片段一次或多次重复出现。——译者注

另外一种方法叫做 D 理论(D-theory)或者叫做描述理论方法(description-theory approach)。在 D 理论中,我们会修改句法分析器输出结果的意义。从本书中一直使用的传统观念出发,通过直接子成分(immediate subconstituent)的定义来构建语法成分。而母成分直接控制其子成分。D 理论定义了一套控制转移关系,即成分 M 控制成分 C 的前提是,C 是 M 的直接子成分,或者是 M 子成分的子成分,依次类推。采用这个新术语,可以说传统句法分析器的输出是一个直接控制关系的集合,这些关系能定义出句法分析树。在 D 理论中,我们将输出结果看成是控制关系的集合。举例来说,在分析过程中,可以说 VP1 控制 PP1,这意味着 PP1 可能是 VP1 的子成分,也允许它作为 VP1 子成分的子成分。因此,通过输出结果本质属性的改变,可以简单有效地刻画出很多种歧义形式。

6.5 浅层句法分析

针对歧义问题,最极端的方法是放弃句子的完整分析,只寻找那些肯定可靠的分析片段。删除通用语法中能导致歧义问题的规则,这个简单的方法差不多就可以实现这种浅层的句法分析器了。例如,如果从语法中删掉下面的规则:

VP → VP PP
NP → NP PP

PP 附着问题就从根本上消失了。当然,你也不可能对大多数句子做出完整的分析。采用这种简化的语法,自底向上的 chart 句法分析器可以生成句子的语法片段序列,从而有助于后续的处理。采用 PP 修饰规则删除之后的语法,我们可以给出动词短语“saw the man in the park with a telescope”最终的 chart 图,如图 6.19 所示。尽管失去了大量的全局结构,但是所有识别出的局部结构均无歧义,而且 chart 图包含了大量有用的语法信息。

VP		7(1,2)		PP		8(3,4)		PP		9(5,6)	
V	1	NP	2	P	3	NP	4	P	5	NP	6

图 6.19 “saw the man in the house with a telescope”浅层分析的 chart 图

当然,这种方法本身并没有解决歧义问题,只不过是 将问题留待语义解释阶段处理。在语义层面上,歧义问题也许更容易解决,因为那时会有更多可用的信息资源。在受限领域的系统中,情况好像就是这样。关于这点,我们需要等到介绍语义解释的时候才能进行深入的讨论。在第 10 章会继续探讨这方面的问题。目前,我们先考虑究竟有多少结构能可靠地识别出来。

依据开发这类系统的经验,我们知道某些结构能可靠地识别出来,其中包括:

- 名词词组——由名词短语组成,从初始的限定词到中间前置修饰语,直到中心名词,但不包括后置的修饰语,如介词短语。
- 动词词组——组成单元为助动词序列、一些副词,直至中心动词。

专有名词短语——既包括简单的名称,如“John”,也包括更复杂的名称,如“New York Times”。如果输入的时候遵循标准的大小写约定,则即使名称不在词典中,仅仅根据大写,我们也能将它们分析出来。因此,专有名词短语的识别就特别容易了。

其他结构看起来好像也能可靠地识别出来,比如介词短语,以及由必需的次范畴成分组成的动词短语。可以生成这样的短语,但是,它们可能并不是句子完整分析中真正的部分,因为名词短语的处理存在一定的局限性。举例来说,浅层句法分析器可能会认为句子“We were punished by the leader of the group.”(我们被这个组长惩罚了。)中存在一个介词短语“by the leader”。但是,这个句子完整的分析结果并不包含这个 PP,它包含的是另外一个介词短语,其中介词的宾语是“the leader of the group”。因为浅层句法分析器不能决定附着关系,因此,并不能生成恰当的解读结果。这也给你留下了选择的余地。有一些系统识别介词但并不分析 PP;而另一些系统先生成不正确的解释,然后依靠语义分析过程来纠正分析中的错误。

由于覆盖范围的限制,浅层句法分析系统可以基于正则文法(等同于有限状态自动机),而不需要采用具有完全能力的上下文无关文法。它们还经常采用一个子系统来准确地估计出每个词的词性。我们会在下一章讨论这样的词性标注系统。目前,你可以简单地假定输入时给定了正确的词性。

这种系统输出的是语法片段序列,其中有一些语法片段和功能词语的词类一样小,而另一些则与复杂的动词序列和 NP 一样大。当面对未知词语的时候,浅层句法分析器会简单地跳过它并继续下一个分析。即使是在受限的语法中,仍然可能存在歧义,浅层句法分析器通常会使用启发式知识来减少这种可能性。一种通行的做法是在类型相同的条件下,较长的成分优先于短成分。因此,启发式知识会把“The house boats”解析为一个 NP,而不是两个 NP(“The house”和“boats”),尽管后者也有可能。

给定输入“We saw the house boats near the lake sink unexpectedly at dawn.”(黎明的时候,我们看到湖边宽敞的游艇出人意料地沉没了。)图 6.20 给出了这种系统可能生成的语法成分序列。

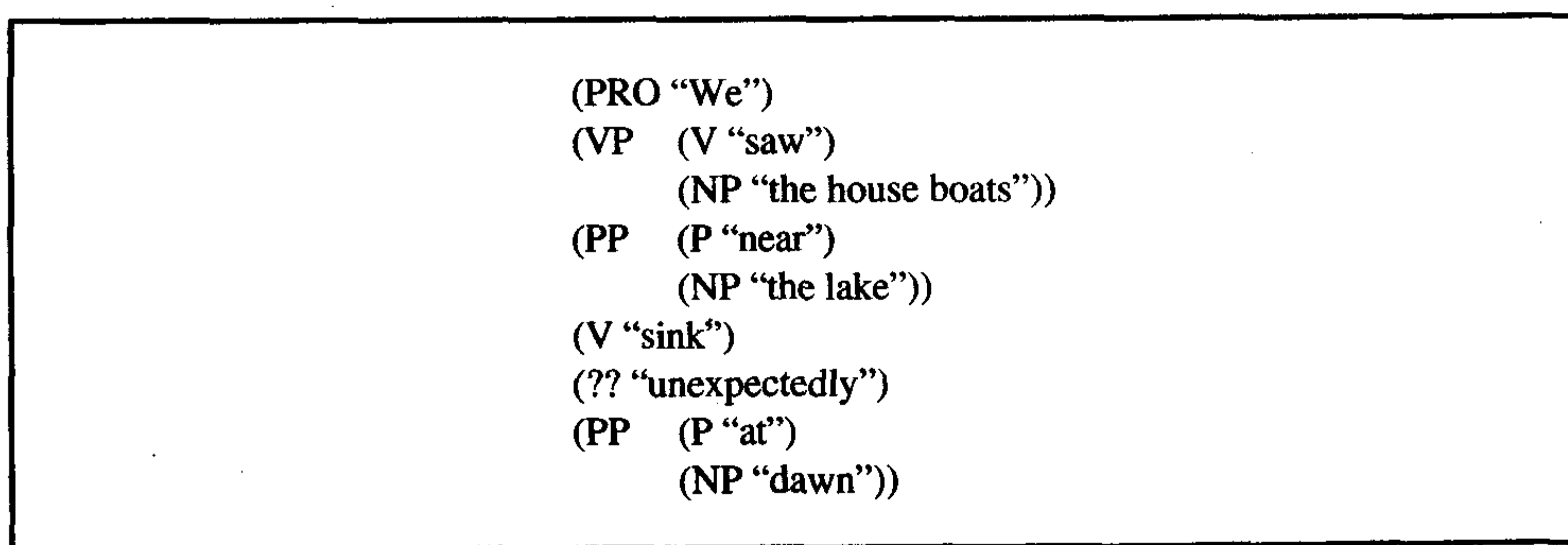


图 6.20 “We saw the house boats near the lake sink unexpectedly at dawn.”的浅层句法分析

6.6 小结

构建高效的句法分析器有几种不同的途径。有一些方法提高了基于搜索的句法分析模型的效率,另一些方法则给出了确定型的句法分析模型,或者仅仅对句子进行浅层分析。这些方法使用了几种不同的技术,分别如下所示:

- 在句法分析状态中对歧义进行编码
- 采用向前看方法来选择合适的规则
- 对编码歧义采用高效的编码技术
- 修改输出的定义从而忽略歧义

6.7 相关工作与深入阅读材料

从 Kimball(1973)的工作开始,研究人员在心理语言学句法分析的优先选择方面发表了大量的文献。Ford, Bresnan 和 Kaplan(1982)提出了词汇优先的思想。Clifton, Frazier 和 Rayner(出版中)是一本很优秀的论文集,收录了这个领域中最近期的工作。这一卷中有一些文章恰好集中探讨了句法分析中的原理与优先级问题,这在本章的 6.1 节中进行过讨论。研究人员似乎得出了同一个结论,即人类的语言处理系统考虑了不同词语与结构在不同句子环境中出现的频率,随后,人会综合运用语义信息和语境上下文知识,最终评价出最可能的结果。依据这种观点,最小附着和最右关联等原理很好地描述了优先选择的规律。不过,优先选择实际上还和其他因素有关。明确使用了分析优先级的计算模型有 Shieber(1984), Pereira(1985)和 Schubert(1986)。其他的计算模型将在第 7 章中进行讨论。

正如文中提到的那样,移进归约句法分析器传统上常用来分析无歧义的程序语言。Aho 等(1986)很精彩的文章讨论了这项工作,他们将本书 6.2 节中描述的移进归约分析器划分为 LR(1)句法分析器(即从左至右的句法分析器,而且只向前看一个符号)。这种技术已经进行了泛化处理,并应用了不同的方法来处理自然语言中的歧义。Shieber(1984)中介绍了这些技术,我们在 6.2 节中曾加以叙述,即对终结符进行泛化处理,推迟对词语的词法分类。同时,当有多条规则可以适用的时候,我们采用句法优先分析从中选择。Tomita(1986)重新结合了全搜索和优化的分析树表示形式,并利用了预测表驱动的移进归约句法分析的速度优势。

关于确定型句法分析的章节主要基于 Marcus(1980)。Charniak(1983)提出了一种变换形式,其中分析状态可以由上下文无关文法自动生成。Fidditch 分析器中的语法是采用这种形式化手段最综合的语法。我们很难找到这个句法分析器的相关文献,不过, Hindle(1989)对其做了简单的描述。Berwick(1985)在语言学习的调查研究中使用了这一框架。

许多系统中都运用了压缩技术,两个最近的例子是 Tomita(1986)和 Alshawi(1992)。Church 和 Patil(1982)系统地探索了自然语言中歧义的广度,并详细讨论了经典形式和压缩等方法。Marcus 等(1983)叙述了 D 理论, Vijay-Shankar(1992)对 D 理论进行了形式化,并应用于 TAG 形式体系中。在只对能可靠识别的结构进行分析方面, Fidditch(Hindle, 1989)是最好的句法分析器之一。它输出的是片段集合,而不是一个完整的解释结果。

6.8 习题

1. 【易】根据图 6.6 中的预测表,采用移进归约句法分析器,请给出句子“The dog was eating the bone.”的分析流程。
2. 【中】文中提到人们使用三种句法分析策略,请列出并定义。讨论下面句子存在的歧义,在这些分析策略的框架下,请给出优先的解读结果。

- a. It flew past the geese over the field.
- b. The artist paints the scene in the park.
- c. He feels the pain in his heart.

3. 【中】生成下列语法的预测表:

$S \rightarrow NP VP$
 $NP \rightarrow ART N$
 $NP \rightarrow ART N PP$
 $VP \rightarrow V NP$
 $VP \rightarrow V NP PP$

指出已有方法不能解决的歧义。使用 6.2 节末尾给出的启发式知识,为分析表选择一个不存在歧义的操作。给出一个反例,要求因为刚才的判断而导致例句不能正确分析。

4. 【中】

a. 下面是一个 VP 语法,其中包含了带 to 的不定式和介词,请构建出该语法的预测表:

$VP \rightarrow V NP$ $PP \rightarrow P NP$
 $VP \rightarrow V INF$ $NP \rightarrow DET N$
 $VP \rightarrow V PP$ $NP \rightarrow NAME$
 $INF \rightarrow to VP$

b. 如果不依靠猜测,该预测表能否正确地分析出下面给出的两个动词短语? 请从词语“walked”开始分析每个句子,并给出分析流程:

Jack walked to raise the money.

Jack walked to the store.

c. 请考虑一下语法中允许词语歧义的情况。比如,词语“sand”既可以当做名词也可以当做动词。请说明你的句法分析器如何处理下面的动词短语:

Jack turned to sand.

Jack turned to sand the board.

5. 【中】设计一个确定型语法,要求该语法可以成功分析出下面的两个句子:

I know	that boy	is	lazy.
	NP	V	ADJ

以及:

I know	that	boys	are	lazy.
		NP	V	ADJ

COMPLEMENTIZER

只要你愿意,可以尽量使用本章给出的语法。给出每个句子的句法分析流程,要求采用本章使用的风格。叙述分析过程的要点。你如何处理词语“that”的不同用法?讨论该解决方案的通用性。至少再给出一个含“that”的句子示例,简要说明你的语法是如何分析它(们)的。

6. 【中】修改现有的 chart 句法分析器,使之能够使用 6.4 节介绍的压缩 chart 图。在不带特征的语法上做一组实验,分别比较压缩 chart 图和未压缩 chart 图的大小。
7. 【中】采用压缩 chart 图来处理带特征的语法会引起什么问题?请构造一个例子来说明,并设计一个解决问题的策略。
8. 【中】假定 chart 分析器采用了压缩的表示形式,如果语法使用了非终结符 N 和终结符 T ,分析的句子长度为 S ,请给出分析器操作数目的上限。

第7章 歧义消解的统计方法

第6章建议采用启发式规则在多种句法解析结果中进行选择。但是,创建这样的启发式规则既困难又费时。而且,也没有一个系统的方法来评价这些规则的实际效果。在这一部分里,要探讨一些基于概率论的方法来解决这类问题。近年来,大规模的自然语言数据库,或者说语料库已经实用化,因此,这种方法非常盛行。这种数据让我们可以采用基于统计的方法自动地获取所需概率。布朗(Brown)语料库是最常用的,包含大约100万的词语,而且所有词语都标注了词性。最近又出现了规模更大的可用语料库,其格式从未加工的原始文本(如,美联社新闻电信和华尔街杂志的文章数据库)到完全标注的句法语料库(如宾州树库)。所有这些可用数据的存在意味着我们可以采用一些新的分析方法,而这些方法在以前几乎不可能实现。

7.1节介绍概率论中的基本概念,并探讨如何将它们应用到自然语言之中。7.2节探讨在语料库上进行概率估计的方法。7.3节给出词性标注的方法。7.4节定义了词语概率的计算方法。7.5节介绍概率上下文无关文法。7.6节探讨了与构建概率驱动的最佳优先句法分析器相关的问题。最后一节介绍上下文相关的概率估计,和上下文无关文法相比,这种方法具有更突出的优势。

7.1 概率论基础

直观地讲,某一事件的概率就是它发生的可能性。概率为1表明该事件一定会发生,同时,概率为0表明该事件一定不会发生。任何一个介于0和1之间的数均表示某种程度的不确定性。可以认为这种不确定性是该事件发生的机会,这和你打赌说事件会不会发生是一样的道理。概率为0.5说明的是该事件发生与不发生的机会均等,即赌注为“50/50”。概率为0.1的事件表示的是每10次机会中会出现一次(1/10的机会),而概率为0.75的事件在100次中就会发生75次(3/4的机会)。

可以根据随机变量更形式化地定义概率。随机变量的取值范围可能是一个预定义的数值集合。尽管取值范围可能是一个无限集,也可能取连续的值,但是,在这里只使用取值范围为有限集的随机变量。例如,考虑抛一枚硬币的事件。随机变量TOSS表示抛硬币的结果,它有两个可能的值:正面(h)或者反面(t)。一个可能的事件是硬币正面朝上,即 $TOSS = h$;而另一个事件是硬币反面朝上,即 $TOSS = t$ 。TOSS不可能取其他值,这反映的事实为抛出的硬币只能是正面朝上或者反面朝上。通常,我们会不提随机变量,只是将 $TOSS = h$ 的概率简单地称为事件h的概率。

概率函数PROB给随机变量的每个可能值都赋予一个概率。任何一个概率函数都必须具备以下属性,其中 e_1, \dots, e_n 是随机变量E所有不同的可能取值。

1. 对所有的 i ,有: $PROB(e_i) \geq 0$

2. 对所有的 i , 有: $PROB(e_i) \leq 1$

3. $\sum_{i=1,n} PROB(e_i) = 1$

现在, 让我们来看一个特定的例子。某匹赛马 Harry 在它的职业生涯中, 总共参加了 100 场赛马。赛马结果采用随机变量 R 表示。 R 可以取两种值, Win(赢)或者 Lose(输)。假定 Harry 总共赢了 20 场比赛。由此, Harry 赢得比赛的概率为 $PROB(R = \text{Win}) = 0.2$; 而输的概率为 $PROB(R = \text{Lose}) = 0.8$ 。注意, 这些值均满足上面定义的三个约束条件。

当然, 实际运用可能存在许多不同的随机变量, 而我们常常对它们之间的相互关系更感兴趣。继续探讨赛马的例子。现在, 再考虑另外一个随机变量 W , 表示天气状况, 取值范围是 Rain(下雨)或者 Shine(天晴)。我们假定 100 次比赛中有 30 次下雨, 而 Harry 在这 30 场比赛中赢了 15 次。直观地看, 如果给定天在下雨这个事实, 即 $W = \text{Rain}$, 那么, $R = \text{Win}$ 的概率为 0.5 (30 次中的 15 次)。可以采用条件概率刻画这种直觉, 记做 $PROB(\text{Win} | \text{Rain})$, 即在给定事件 Rain 的条件下 Win 的概率。条件概率定义为下面的公式:

$$PROB(e | e') = PROB(e \& e') / PROB(e')$$

其中, $PROB(e \& e')$ 是两事件 e 和 e' 同时发生的概率。

你已经知道 $PROB(\text{Rain}) = 0.3$ 以及 $PROB(\text{Win} \& \text{Rain}) = 0.15$, 根据条件概率的定义可以计算 $PROB(\text{Win} | \text{Rain})$, 看看是否与我们的直觉一致。

$$\begin{aligned} PROB(\text{Win} | \text{Rain}) &= PROB(\text{Win} \& \text{Rain}) / PROB(\text{Rain}) \\ &= 0.15 / 0.30 \\ &= 0.5 \end{aligned}$$

贝叶斯法则是一个与条件概率相关的重要定理。这个公式将 B 条件下事件 A 的概率与 A 条件下事件 B 的概率联系在一起, 即:

$$PROB(A | B) = \frac{PROB(B | A) * PROB(A)}{PROB(B)}$$

我们可以用赛马的例子来说明这一公式。使用贝叶斯法则, 可以计算 Harry 赢得比赛时下雨的概率, 即:

$$\begin{aligned} PROB(\text{Rain} | \text{Win}) &= (PROB(\text{Win} | \text{Rain}) * PROB(\text{Rain})) / PROB(\text{Win}) \\ &= (0.5 * 0.3) / 0.2 \\ &= 0.75 \end{aligned}$$

显然, 如果直接按定义计算条件概率, 其结果会和上述值相同, 直接计算过程如下:

$$\begin{aligned} PROB(\text{Rain} | \text{Win}) &= PROB(\text{Rain} \& \text{Win}) / PROB(\text{Win}) \\ &= 0.15 / 0.20 \\ &= 0.75 \end{aligned}$$

我们通常并没有某种情形的完整信息, 也可能并不知道所有必需的概率, 因此, 贝叶斯法则非常有用。我们经常合理地估计某些概率, 然后再用贝叶斯法则计算其他概率。

相互独立是概率论中另外一个重要的概念。两个事件相互独立的前提是, 其中任何一个事件的出现并不影响另一事件出现的概率。例如, 在赛马的例子中, 考虑另一个随机变量 L , 表示我是否带着吉祥物去比赛场。带着吉祥物的值为 Foot, 或不带吉祥物的值为 Empty。假设我 60 次比赛带了吉祥物, 而 Harry 赢了 12 场, 这意味着在我带吉祥物的前提下, Harry 赢得比

赛的概率为 $PROB(Win|Foot) = 12/60 = 0.2$ 。这和 Harry 平时跑赢的概率是一样的,为此,可以得出结论,跑赢与带吉祥物相互独立。

更形式化地说,两个事件 A 和 B 相互独立,当且仅当:

$$PROB(A|B) = PROB(A)$$

根据条件概率的定义,这等价于:

$$PROB(A \& B) = PROB(A) * PROB(B)$$

注意,跑赢事件和下雨事件并不相互独立,因为 $PROB(Win \& Rain) = 0.15$,而 $PROB(Win) * PROB(Rain) = 0.2 * 0.3 = 0.06$ 。换句话说,跑赢比赛和下雨一起出现比随机出现的可能性要大得多。

现在,考虑一下词性识别问题,这是概率论在语言方面的应用。词性识别指的是给定一个包含歧义词语的句子,我们要确定每个词最可能的词性。在 7.3 节会详细地研究这个问题。在这里,我们只是借助这个例子来说明应用概率论的一些思想。一个词既可以当做名词又可以当做动词,假定你需要准确确定它的词类。这可以形式化为两个随机变量,其中一个是 C,取值范围是所有可能的词性(即名词 N 和动词 V);另一个是 W,取值为所有可能的词。考虑一个 $W = flies$ 的例子,这个问题可以表述为在 $PROB(C = N|W = flies)$ 和 $PROB(C = V|W = flies)$ 中,判断哪个概率更大。请注意,在前面已经提过,随机变量在公式中可以省略,因此 $PROB(C = N|W = flies)$ 通常可以简化为 $PROB(N|flies)$ 。根据条件概率的定义,词语“flies”的两种词类的概率计算如下:

$$PROB(N|flies) = PROB(flies \& N) / PROB(flies)$$

$$PROB(V|flies) = PROB(flies \& V) / PROB(flies)$$

这两个式子的分母相同,所以问题可以简化为 $PROB(flies \& N)$ 和 $PROB(flies \& V)$ 哪个更大。

如何获得这些概率呢?显然,无法确定真实的概率,因为我们没有所有书面文本的记录,更不要说将来要生成的文本了。但是,如果有足够大的数据样本,则可以估计这些概率。

假设有一个简单句子的语料库,其中包含 1 273 000 个词。假如发现词语“flies”使用了 1000 次,其中,400 个作为名词 N 使用,600 个作为动词 V 使用。可以通过语料库中词语出现的频率来估计概率。例如,如果随机选择一个词语,该词是“flies”的概率为:

$$PROB(flies) = 1000/1\ 273\ 000 = 0.000\ 8$$

“flies”分别作为名词和动词出现的联合概率为:

$$PROB(flies \& N) = 400/1\ 273\ 000 = 0.000\ 3$$

$$PROB(flies \& V) = 600/1\ 273\ 000 = 0.000\ 5$$

因此,在每次用到“flies”时,动词是最好的猜测结果。可以使用下面的公式计算“flies”是动词的概率:

$$\begin{aligned} PROB(V|flies) &= PROB(V \& flies) / PROB(flies) \\ &= 0.000\ 5 / 0.000\ 8 = 0.625 \end{aligned}$$

使用这一方法,如果算法总是认为“flies”是动词,其准确率大约是 60%。这当然是一个比较差的策略,但是比猜测“flies”总是名词要好。要想获得更好的方法,必须考虑更多的上下文信息,如出现“flies”的句子。我们会在 7.3 节对其进行讨论。

7.2 概率估计

如果你有与某个问题相关的所有数据,那么,可以针对这些数据计算出精确的概率。例如,上一节中的赛马 Harry 只参加了 100 场比赛,然后就放归原野。现在,在这 100 场赛马中任选一场,我们能够计算出 Harry 赢得比赛的概率。但是,概率通常并不是这样应用的。一般来说,你会希望使用概率来预测未来的行为,即使用 Harry 以前的表现来预测它赢得第 101 场赛马的可能性(这是一场你要下注的比赛)。这就是概率论在现实生活中的实际应用。在使用概率来解决歧义问题时,情形相同,因为你感兴趣的是分析那些从未见过的句子。因此,需要使用以前出现的句子数据来预测下一个句子的解释。在实际的处理当中,我们使用的都是概率的估计值,而不是实际的概率。

从上一节中可以看出,一种估计方法是将语料库中的频率作为概率,并预测新句子的解释。如果我们以前曾见过词语“flies”1000 次,而且其中 600 次是动词,那么,就假定 $PROB(V|flies)$ 为 0.6。然后,可以根据这一结果来指导第 1001 次结果的预测。这种简单的频率估计叫做最大似然估计(MLE, maximum likelihood estimator)。如果拥有待估计事件的大量实例,则这种对真实概率的估计将是非常可信的。

一般说来,估计值的准确性随着使用的数据量的扩大而提高。大数定理(law of large number)是一个统计学上的定理,内容是如果拥有的数据没有限制,那么,估计值可以达到所需的精确程度。不过,如果只有少量的样本,估计值可能会非常不可靠。现在,试着估计一下硬币抛出后正面朝上的真实概率。因为我们已经知道正确答案是 0.5,为了讨论方便,假定估计值在 0.25 至 0.75 之间就认为足够准确,这一范围叫做可接受误差区间。如果只做两次试验,就会得到 4 种可能的结果:两次正面,先正面后反面,先反面后正面,两次反面,如图 7.1 所示。这意味着,如果扔两次硬币,会有一半的机会获得估计值 0.5。但在另一半机会里,会估计正面朝上的概率为 1(两次正面的情况)或者 0(两次反面的情况)。所以,只有 50%的概率获得可接受误差区间内的估计值。如果扔三次,获得足够可靠估计值的可能性升至 75%,如图 7.2 所示。如果扔 4 次,估计值足够精确的可能性为 87.5%;扔 8 次时,这种可能性为 93%;扔 12 次时,这种可能性为 95%,依次类推。无论扔多少次,总会发现估计值有不准确的时候,但是,如果进行足够多次的试验,就可以将这种不准确出现的概率减至你所期望的值。

结果	$Prob(H)$ 的估计值	估计是否可以接受?
HH	1.0	否
HT	0.5	是
TH	0.5	是
TT	0.0	否

图 7.1 给定两次投币结果 $Prob(H)$ 的估计值

在大数据量的前提下,几乎每种估值方法都很有效。遗憾的是,自然语言的应用中需要估计大量参数,而这些事件中的很大一部分是极少发生的。这就是数据稀疏问题。例如,布朗语料库中包含 100 万个词语,但由于词语的重复出现,实际上只有 49 000 个不同词语。基于这一

点,你可能会期望每个词平均出现的次数大约为 20,但是,实际上有40 000多个词只出现了5 次或者更少。根据这么小的数据量,我们对这些词的词性估计可能会极不准确。最坏的情况是某个低频词的某种可能的词性从来都没有出现过,它有这种词性的概率就会估计为 0。这样,任何使用该词这种词性的句子都不能够解释,因为包含该词的整个句子概率也为 0。非常遗憾,在自然语言的应用中小概率事件很常见,因此,算法可靠地估计出这些低频词的概率十分重要。

结果	<i>Prob</i> (H)的估计值	估计是否可以接受?
HHH	1.0	否
HHT	0.66	是
HTH	0.66	是
HIT	0.33	是
THH	0.66	是
THI	0.33	是
TTH	0.33	是
TTT	0	否

图 7.2 给定三次投币结果 *Prob*(H)的估计值

还有其他的估值方法来处理低频事件的概率估计问题。为了进行探究,先引入一个可以相互比较不同方法的框架。对于某一随机变量 *X*,所有的方法都会从取值集合 *V_i* 开始,*V_i* 可由 *X* = *x_i* 的次数计算得到。最大似然估计方法使用的是 *V_i* = |*x_i*|,即 *V_i* 就是 *X* = *x_i* 的次数。一旦 *V_i* 由每个 *x_i* 求出来了,那么,概率的估计值可由下面的公式计算得到:

$PROB(X = x_i) \cong V_i / \sum_i V_i$

注意,分母可以保证估计值遵循概率函数的三个属性,这在 7.1 节中进行过定义。

解决零概率问题的一个方法是保证所有的 *V_i* 都不为零。例如,可以给每一个计数值增加一个很小的数,假定是 0.5,则有 *V_i* = |*x_i*| + 0.5。这不仅保证不会出现零概率,而且相应地保持了高频值的可能性,这种估计方法称为期望似然估计(ELE, expected likelihood estimator)。为了考察这种方法和最大似然估计的区别,我们看一个在语料库中刚好没有出现的词语 *w*,估计 *w* 作为 40 个词类 *L₁*, ..., *L₄₀* 之中某一个词类的概率。这样,可以得到一个随机变量 *X*,仅当 *w* 以词性 *L_i* 出现的时候,才有 *X* = *x_i*。使用最大似然估计法, *PROB*(*X* = *x_i*)将无法定义,因为公式的分母为零。但是,期望似然估计对每一种可能的词类都给出一个概率。例如,在有 40 个词类时,每个 *V_i* 都将是 0.5,因此, *PROB*(*L_i* | *w*) ≈ 0.5/20 = 0.025。这个估计值更好地反映了事实,即我们没有这个词的信息。另一方面,期望似然估计非常保守。如果 *w* 在语料库中出现 5 次,一次作为动词,4 次作为名词,那么 *PROB*(*N* | *w*)的最大似然估计值会是 0.8,而期望似然估计值会是 4.5/25 = 0.18,和直观结果相比,这个值要小得多。

7.2.1 评测

一旦有了概率估计值的集合以及针对某一应用的算法,你就会想知道,和其他算法或者算法的变种相比,你的新技术的性能到底怎样? 解决这个问题的常用方法是将语料库分为两部

分:训练集和测试集。典型的情况是测试集占总数据的 10% ~ 20%。然后,训练集用来估计概率,而算法在测试集上运行,这样就可以观察它在新数据上的分析效果。算法运行在训练集上并不是个可靠的评测方法,因为它并不能测试出方法的通用性。例如,简单地记住所有的答案,并且在测试中照样反馈,就可以取得很好的效果。另一种更精细的测试方法叫做交叉验证。这种方法不断地将语料库的不同部分作为测试集,在剩下的数据上进行训练,然后在新的测试集上进行评测。这种方法降低了选择的测试集比你所期望的更加简单的可能性。

7.3 词性标注

词性标注就是为句子中的词语选择一个最可能的词类序列。宾州树库项目使用了一个典型的标记集,如图 7.3 所示。在 7.1 节中,介绍了完成这个任务最简单的算法,即总是选择在训练集中出现最多的解释。令人惊讶的是,这一方法常常能够达到大约 90% 的成功率,主要是因为大多数语料库中一半以上的词语不存在歧义。因此,以这个方法作为起点,可以评测那些使用了更复杂技术的算法。除非一个方法的效果远高于 90%,否则它的效果就不是很好。

1. CC	并列连词(coordinating conjunction)	20. RB	副词(adverb)
2. CD	基数词(cardinal number)	21. RBR	比较副词(comparative adverb)
3. DT	限定词(determiner)	22. RBS	副词的最高级(superlative adverb)
4. EX	表示存在的 there(existential there)	23. RP	语助词(particle)
5. FW	外来词(foreign word)	24. SYM	符号(数学或科学)
6. IN	介词/从属连词 (preposition /subord. conj)		[symbol(math or scientific)]
7. JJ	形容词(adjective)	25. To	to
8. JJR	比较形容词(comparative adjective)	26. UH	感叹词(interjection)
9. JJS	形容词最高级(superlative adjective)	27. VB	动词原型(verb, base form)
10. LS	表项标记(list item marker)	28. VBD	动词过去式(verb, past tense)
11. MD	情态动词(modal)	29. VBG	动词、动名词/现在分词 (verb, gerund/pres. participle)
12. NN	名词,单数或物质名词 (noun, singular or mass)	30. VBN	动词的过去分词(verb, past participle)
13. NNS	名词复数(noun, plural)	31. VBP	一般现在时动词,非第三人称单数 (verb, non-3s, present)
14. NNP	专有名词单数 (proper noun, singular)	32. VBZ	现在时动词,第三人称单数 (verb, 3s, present)
15. NNPS	专有名词复数(proper noun, plural)	33. WDT	wh 限定词(wh-determiner)
16. PDT	前置限定词(predeterminer)	34. WP	疑问代词(wh-pronoun)
17. POS	所有格结束(possessive ending)	35. WPZ	疑问物主代词 (possessive wh-pronoun)
18. PRP	人称代词(personal pronoun)		
19. PP\$	物主代词(possessive pronoun)	36. WRB	疑问副词(wh-adverb)

图 7.3 宾州树库标记集

提高可靠性的一般方法是使用词语在句子中出现的一些局部上下文信息。例如,在 7.1 节,我们看到在语料样本库中选择“flies”为动词是最好的选择,可以达到大约 60% 的准确率。但是,另一方面,如果这个词的前面是词语“the”,那么,它更可能是名词。本节要研究的方法会尽可能地利用这些信息。

我们完全形式化地考虑这个问题。设 w_1, \dots, w_T 是一个词语序列。我们要找到词类的序列设为 C_1, \dots, C_T , 它可使得下式取得最大值:

$$1. \quad \text{PROB}(C_1, \dots, C_T | w_1, \dots, w_T)$$

遗憾的是,需要特别多的数据才能合理地估计出这种序列的值,所以,直接的方法不能适用。不过,采用一些合理的近似方法,可以产生很好的效果。为了进一步推导,必须采用贝叶斯法则重新表述这个问题,即这个条件概率等于:

$$2. \quad (\text{PROB}(C_1, \dots, C_T) * \text{PROB}(w_1, \dots, w_T | C_1, \dots, C_T)) / \text{PROB}(w_1, \dots, w_T)$$

和以前一样,我们感兴趣的是如何找到使上式取最大值的 C_1, \dots, C_n , 而所有情况中的公共分母都不会影响最终结果。因此,问题可以简化为寻找序列 C_1, \dots, C_n , 使得下式最大:

$$3. \quad \text{PROB}(C_1, \dots, C_T) * \text{PROB}(w_1, \dots, w_T | C_1, \dots, C_T)$$

我们还是缺乏有效的方法准确地计算出这些长序列的概率,因为这需要特别大的数据量。不过,可以通过一些独立性假设,采用更易于估计的概率来逼近这个概率。尽管这些独立性假设并不真的成立,但是这种估值方法在实践中可以取得很好的效果。式 3 中的两个表达式都可以近似地计算出来。第一个表达式是词性序列的概率,我们可以通过一系列基于前面有限个词性的词性概率近似地计算出词性序列的概率。最常用的假设常常使用前面的一到两个词性。二元模型考察的是词类对(或者是词对),它利用词类 C_i 在另一个词类 C_{i-1} 后面出现的条件概率,记为 $\text{PROB}(C_i | C_{i-1})$ 。三元语法使用的条件概率是给定两个相邻词类(或者词),另一个词类(或者词)在后面出现的概率,即 $\text{PROB}(C_i | C_{i-2} C_{i-1})$ 。这些模型称为 n 元语法模型(n -gram model),其中 n 表示模型使用的词语数目。在实际系统中,三元语法模型的效果更好。在这里为了简单起见,我们只探讨二元语法模型。根据二元语法模型,可以进行如下估计:

$$\text{PROB}(C_1, \dots, C_T) \cong \prod_{i=1, T} \text{PROB}(C_i | C_{i-1})$$

考虑到句子的开头,我们在 0 位置增加一个伪词类 ϕ 作为 C_0 的值。这样,在以 ART 开始的句子中,第一个二元概率为 $\text{PROB}(\text{ART} | \phi)$ 。这样,使用二元语法模型,序列 ART N V N 的概率估计值为:

$$\begin{aligned} \text{PROB}(\text{ART N V N}) \cong & \text{PROB}(\text{ART} | \phi) * \text{PROB}(\text{N} | \text{ART}) \\ & * \text{PROB}(\text{V} | \text{N}) * \text{PROB}(\text{N} | \text{V}) \end{aligned}$$

式 3 中的第二个概率为:

$$\text{PROB}(w_1, \dots, w_T | C_1, \dots, C_T)$$

可以假定词语作为某个词类出现与其前后的词类相互独立。可以近似估计为每个词作为某个特定词性出现概率的乘积,即:

$$\text{PROB}(w_1, \dots, w_T | C_1, \dots, C_T) \cong \prod_{i=1, T} \text{PROB}(w_i | C_i)$$

通过这两个近似估计,问题就转换为寻找序列 C_1, \dots, C_T , 使得下式的值最大:

$$\prod_{i=1, T} \text{PROB}(C_i | C_{i-1}) * \text{PROB}(w_i | C_i)$$

这个新式子的优点在于,相关的概率可以很容易地从标注了词性的文本语料库中估计出来。具体地说,如果给定一个文本数据库,二元概率可以简单地估计为每对词类出现的次数和词类单独出现的次数之比。V 在 N 后面出现的概率可以估计如下:

$$PROB(C_i = V | C_{i-1} = N) \cong \frac{\text{Count}(N \text{ 在位置 } i-1 \text{ 并且 } V \text{ 在位置 } i)}{\text{Count}(N \text{ 在位置 } i-1)}$$

图 7.4 给出了一些二元词对的频率,这些频率是从人工生成的简单句子语料库中计算出来的。这个语料库包含 300 个句子,其中的词语仅使用了四种词性:名词(N)、动词(V)、冠词(ART)和介词(P)。而宾州树库真正用到的典型标记集包含了大约 40 个标记,如图 7.3 所示。这个人工生成的语料库包含 1998 个词语,其中名词 833 个,动词 300 个,冠词 558 个以及介词 307 个。采用上面的公式估计每个二元词对的概率。为了解决数据稀疏的问题,对于每个没有给出的词对,我们假定其是某个标记的概率为 0.000 1。

类别	从 i 开始计数	词对	从 i, i+1 开始计数	双图	估计
∅	300	∅, ART	213	$PROB(ART \emptyset)$	0.71
∅	300	∅, N	87	$PROB(N \emptyset)$	0.29
ART	558	ART, N	558	$PROB(N ART)$	1
N	833	N, V	358	$PROB(V N)$	0.43
N	833	N, N	108	$PROB(N N)$	0.13
N	833	N, P	366	$PROB(P N)$	0.44
V	300	V, N	75	$PROB(N V)$	0.35
V	300	V, ART	194	$PROB(ART V)$	0.65
P	307	P, ART	226	$PROB(ART P)$	0.74
P	307	P, N	81	$PROB(N P)$	0.26

图 7.4 生成语料库中估计出来的二元概率

按照词类统计每个词语出现的次数,通过这个频率,可以简单地估计出词语的生成概率 $PROB(w_i | C_i)$ 。图 7.5 给出了一些词语单独出现的次数,根据这些计数,可以相应地估算出词语的生成概率,结果如图 7.6 所示。注意,词语的生成概率是指在给定的词类中出现某个特定词的概率,而不是指给定的词以某一特定词类出现的概率。例如, $PROB(the | ART)$ 可以估计为 the 作为 ART 的次数/ART 出现的总次数,而另一概率 $PROB(ART | the)$ 的值可能与之相差甚远。

给定这些概率的估计值,怎样才能找到相应的词类序列,使之生成某个特定句子的概率最大? 最原始的方法是生成该句所有可能的词类序列,然后估计每一个序列的概率,最后选出概率最大的序列。这种方法的问题是最终序列的数量是指数级的——假定有 N 种词类并且有 T 个词语,那么最终会有 N^T 个可能的序列。值得庆幸的是,对于这些数据存在独立性假设,因此,可以采取更好的方法。

我们只探讨二元概率,因此,第 i 个词语属于词类 C_i 的概率仅仅依赖于第 i-1 个词语的词类 C_{i-1} 。这样,该过程的模型可以表示为一种特殊形式的概率有限状态自动机,结果如

图 7.7所示。图中的每个节点代表一个可能的词类,转移概率(即图 7.4 中的二元概率)表示的是某个词类跟在另一词类之后的概率。

	N	V	ART	P	总计
<i>flies</i>	21	23	0	0	44
<i>fruit</i>	49	5	1	0	55
<i>like</i>	10	30	0	21	61
<i>a</i>	1	0	201	0	202
<i>the</i>	1	0	300	2	303
<i>flower</i>	53	15	0	0	68
<i>flowers</i>	42	16	0	0	58
<i>birds</i>	64	1	0	0	65
其他	592	210	56	284	1142
总计	833	300	558	307	1998

图 7.5 语料库中部分词出现次数的汇总表

$PROB(the \mid ART)$	0.54	$PROB(a \mid ART)$	0.360
$PROB(flies \mid N)$	0.025	$PROB(a \mid N)$	0.001
$PROB(flies \mid V)$	0.076	$PROB(flower \mid N)$	0.063
$PROB(like \mid V)$	0.1	$PROB(flower \mid V)$	0.05
$PROB(like \mid P)$	0.068	$PROB(birds \mid N)$	0.076
$PROB(like \mid N)$	0.012		

图 7.6 词语的生成概率

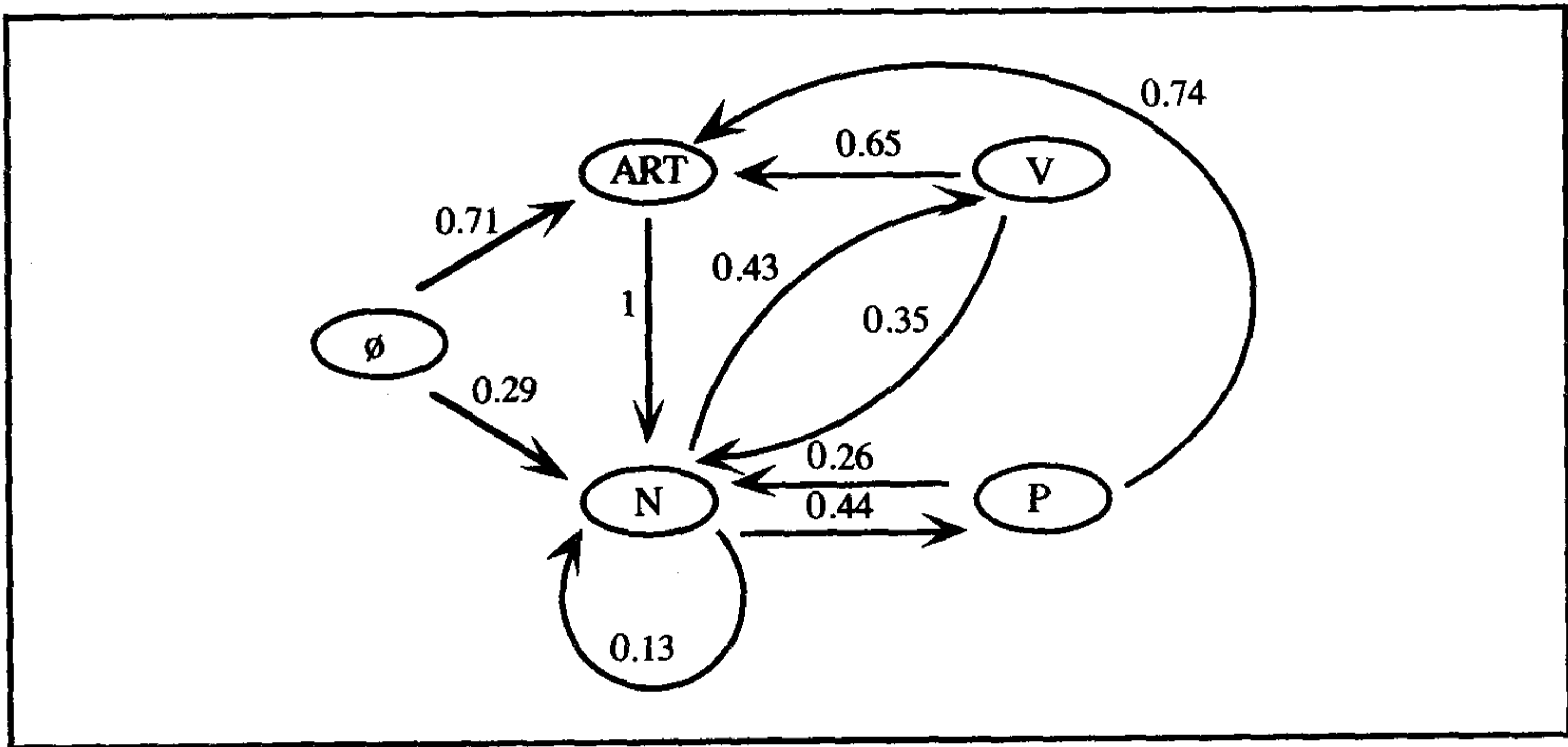


图 7.7 表示二元概率的马尔可夫链

采用这样一幅网络图,可以非常简单地计算出任何一个词类序列的概率。具体方法是,找到与序列相对应的、贯穿整个网络的路径,将所有的转移概率相乘,最终的积就是对应序列的概率。例如,序列 ART N V N 的概率为 $0.71 * 1 * 0.43 * 0.35 = 0.107$ 。当然,只有当词类出现的概率仅仅依赖于它前面的一个词类的时候,这种表示才是准确的。在概率论中,常称之为马尔可夫假设(Markov assumption),图 7.7 这样的网络就叫做马尔可夫链(Markov chain)。

现在,可以扩展网络表示方法,将词语的生成概率也囊括在内。具体地说,允许每个节点都有一个输出概率,即在每个与该节点相关的可能输出上增加一个相应的输出概率(output probability)。例如,图 7.7 中的节点 N 会关联一个概率表,表示如果随机选择一个名词,选中某个词语的可能性有多大。输出概率实际上就是词语的生成概率,如图 7.6 所示。图 7.7 中的每个节点都有相关的输出概率,这样的网络称为隐马尔可夫模型(HMM, Hidden Markov Model)。顾名思义,“隐”指的是对于某个特定的词语序列,并不清楚马尔可夫模型中对应的状态。例如,词语“flies”可以从状态 N 中生成,生成概率为 0.025(见图 7.6);也可以从状态 V 中生成,生成概率为 0.076。因为存在歧义,所以,根据这幅图计算词语序列的概率就不再微不足道了。给定某个特定序列,将路径上的概率乘以每一个输出的概率,这样,可以很容易地计算出产生某一特定输出的概率。例如,序列 N V ART N 生成输出“Flies like a flower”的概率计算如下。对于图 7.7 中的马尔可夫模型,路径 N V ART N 的概率为 $0.29 * 0.43 * 0.65 * 1 = 0.081$ 。同时,根据图 7.6 中的输出概率,可以计算输出序列为“Flies like a flower”的概率,计算过程如下:

$$\begin{aligned} & PROB(flies | N) * PROB(like | V) * PROB(a | ART) * PROB(flower | N) \\ &= 0.025 * 0.1 * 0.36 * 0.063 \\ &= 5.4 * 10^{-5} \end{aligned}$$

将这些概率相乘,可以得到隐马尔可夫模型产生这个句子的概率为 $4.37 * 10^{-6}$ 。更一般地,对于给定的序列 C_1, \dots, C_T , 句子 w_1, \dots, w_T 概率的计算公式为:

$$\prod_{i=1, T} PROB(C_i | C_{i-1}) * PROB(w_i | C_i)$$

给定一个词语序列,如何找到最可能的词类序列? 现在,我们继续讨论这个问题。由于马尔可夫假设的存在,所以,很重要的直觉是我们并不需要枚举所有可能的序列。实际上,可以将同一词类结尾的序列合并在一起,因为下一词类仅仅依赖于序列中的前一个词类。因此,对于每一个可能的终结词类,只追溯迄今为止最可能的序列,可以忽略所有其他的小概率序列。现在,让我们看看如何寻找句子“Flies like a flower”最可能的词类序列。其中,词语的生成概率和二元概率采用我们已经讨论过的数据。假定有 4 种可能的词类,那么,就有 $4^4 = 256$ 个长度为 4 的不同序列。原始算法必须生成全部 256 个序列,并且要比较它们的概率以找出概率最大的结果。不过,利用马尔可夫假设,这个序列集合可以进行压缩合并,合并形式中,只考虑每个词语的四种可能词类。这种形式表示了全部 256 个序列,转移图如图 7.8 所示。为了找到最可能的序列,我们每次读入一个词语,并找出每个结尾词类最可能的序列。比如,对于两个词语的片段“flies like”,需要先找到四个最佳序列,即 like 作为 V 结尾的最佳序列,作为 N 结尾的最佳序列,作为 P 结尾的最佳序列以及作为 ART 结尾的最佳序列。然后,可以使用这种信息寻找三个词语“flies like a”的四个最佳序列。其中,每个序列都以不同的词类结尾。重复这一过程,直到遍历所有的词语为止。这一算法通常称为 Viterbi 算法(Viterbi algorithm)。对于含 T 个词语和 N 个词类的问题,可以保证在 $k * T * N^2$ 步内找到最可能的序列,而原始方法的

搜索需要 N^T 步。对于某些常数 k , Viterbi 算法比原始方法好得多! 接下来我们会详细介绍 Viterbi 算法。

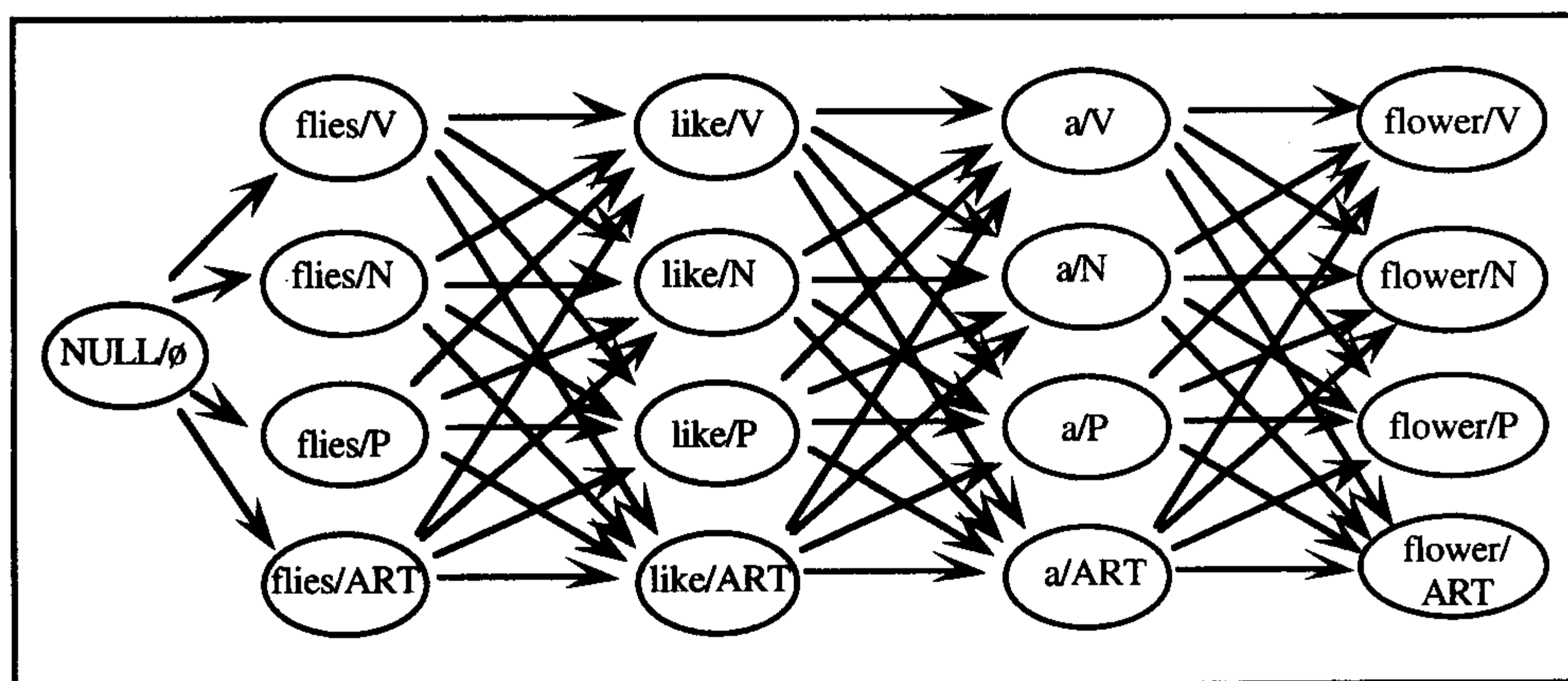


图 7.8 利用马尔可夫假设, 256 种可能序列的表示形式

7.3.1 Viterbi 算法

每个位置均存在指向每个可能词类的最佳序列, 我们采用 $N \times T$ 维数组来记录相应最佳序列的概率。其中, N 为所有词类 (L_1, \dots, L_N) 的数量, T 为句子 (w_1, \dots, w_T) 中词语的数量。数组 $SEQSCORE(n, t)$ 记录最佳序列的概率, 该序列截止到位置 t , 且终结词语的词类为 L_n 。要记录每个位置上每种词类实际的最佳序列, 只需要记录每个位置每个词类前面的那个词类即可。BACKPTR 是另一个 $N \times T$ 数组, 对于每个位置上的每个词类, 它表示的是最佳序列中第 $t-1$ 个位置上的词类。通过计算这两个数组的值, 最后可以给出该算法, 如图 7.9 所示。

已知词语序列 w_1, \dots, w_T , 词类 L_1, \dots, L_N , 词语概率 $PROB(w_i | L_i)$, 以及二元概率 $PROB(L_i | L_j)$; 寻找该词语序列上可能性最大的词类序列 C_1, \dots, C_T 。

初始化阶段

For $i = 1$ to N do

$SEQSCORE(i, 1) = PROB(w_1 | L_i) * PROB(L_i | \emptyset)$

$BACKPTR(i, 1) = 0$

迭代阶段

For $t = 2$ to T

For $i = 1$ to N

$SEQSCORE(i, t) = \max_{j=1, \dots, N} (SEQSCORE(j, t-1) * PROB(L_i | L_j)) * PROB(w_t | L_i)$

$BACKPTR(i, t) = \text{使上式最大的 } j \text{ 的索引号}$

序列识别阶段

$C(T) = \text{使 } SEQSCORE(i, T) \text{ 最大的 } i$

For $i = T-1$ to 1 do

$C(i) = BACKPTR(C(i+1), i+1)$

图 7.9 Viterbi 算法

假设我们已经分析了一个语料库,分别得到了图 7.4 和图 7.6 中的二元概率和词语的生成概率,同时假定任何一个不在图 7.4 中的二元概率均为 0.000 1。使用这些概率,在句子“Flies like a flower”上执行该算法,具体的操作过程如下。

在初始化阶段,根据下面的式子设置数组的第一行:

$$\text{SEQSCORE}(i, 1) = \text{PROB}(\text{Flies} \mid L_i) * \text{PROB}(L_i \mid \emptyset)$$

其中, L_i 的取值范围是 V, N, ART 和 P。根据图 7.6 中给出的词语生成概率,我们发现只有名词和动词词条的值大于零。以 V(= L_1) 结尾、只有一个词类的最佳序列生成“flies”的得分为 $7.6 * 10^{-6}$,而以 N(= L_2) 结尾的最佳序列得分为 0.007 25。

该算法第一步的结果如图 7.10 中左边的网络所示。我们计算出了“flies”属于各个词类的概率。在算法的第二阶段,将序列每次向前扩展一个词语,并始终跟踪每个词类当前找到的最佳序列。例如,状态 like/V 的概率计算如下:

$$\begin{aligned} \text{PROB}(\text{like/V}) &= \text{MAX}(\text{PROB}(\text{flies/N}) * \text{PROB}(\text{V} \mid \text{N}), \\ &\quad \text{PROB}(\text{flies/V}) * \text{PROB}(\text{V} \mid \text{V})) * \\ &\quad \text{PROB}(\text{like/V}) \\ &= \text{MAX}(0.00725 * 0.43, 7.6 * 10^{-6} * 0.0001) * 0.1 \\ &= 3.12 * 10^{-4} \end{aligned}$$

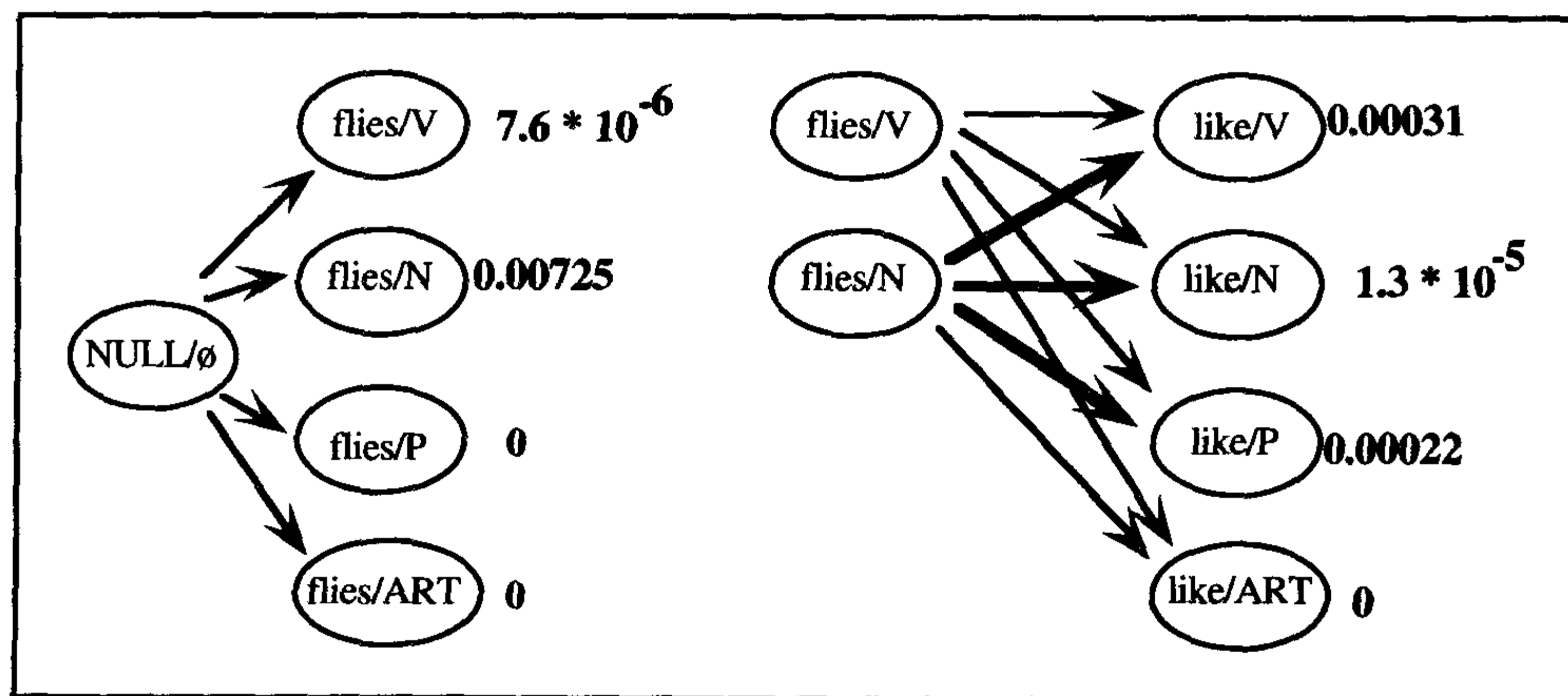


图 7.10 Viterbi 算法前两步的处理结果

与图 7.10 所示的值相比,这个值的不同之处在于,这里的计算简单地使用了概率的截断近似值。即,能生成“flies like”以 V 结尾长度为 2 的最可能的序列得分为 $3.1 * 10^{-4}$ (该序列为 N V);以 P 结尾、最可能的序列得分为 $2.2 * 10^{-5}$ (该序列为 N P);而以 N 结尾、最可能的序列得分为 $1.3 * 10^{-5}$ (该序列为 N N)。图中粗箭头表示的是指向每个节点的最佳序列。这种计算将以同样的方式继续,直到处理完每一个词为止。图 7.11 给出了下一次迭代后的结果,图 7.12 给出了最终的结果。最后,概率最高的序列在状态“flower/N”处终止。从这个词类开始回溯[使用 BACKPTR(1,4)等信息],可以容易地找到完整的序列 N V ART N,结果和我们的直觉一致。

如果从一个大数据量的语料库中计算得到这些概率估计值,而且语料库的数据和输入的数据风格相同,那么,这样的算法可以很有效地解决问题。研究者的报告一致表明,三元语法

模型可以达到 95% 或者更高的标注准确率。不过,需要记住的是,选取最大概率词类的简单算法同样可以达到 90% 的准确率。引入这些技术之后,错误率仍然减少了一半。

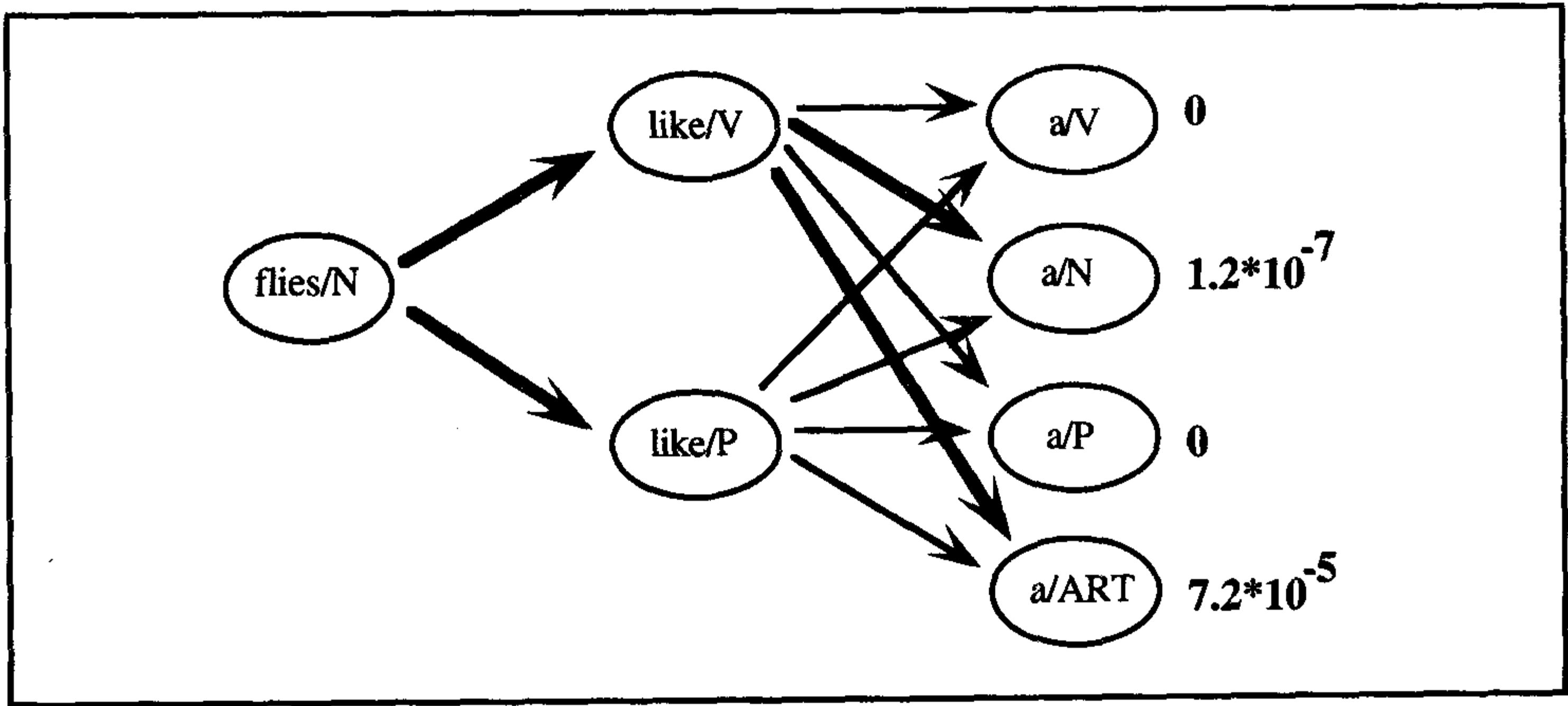


图 7.11 第二次迭代后的结果

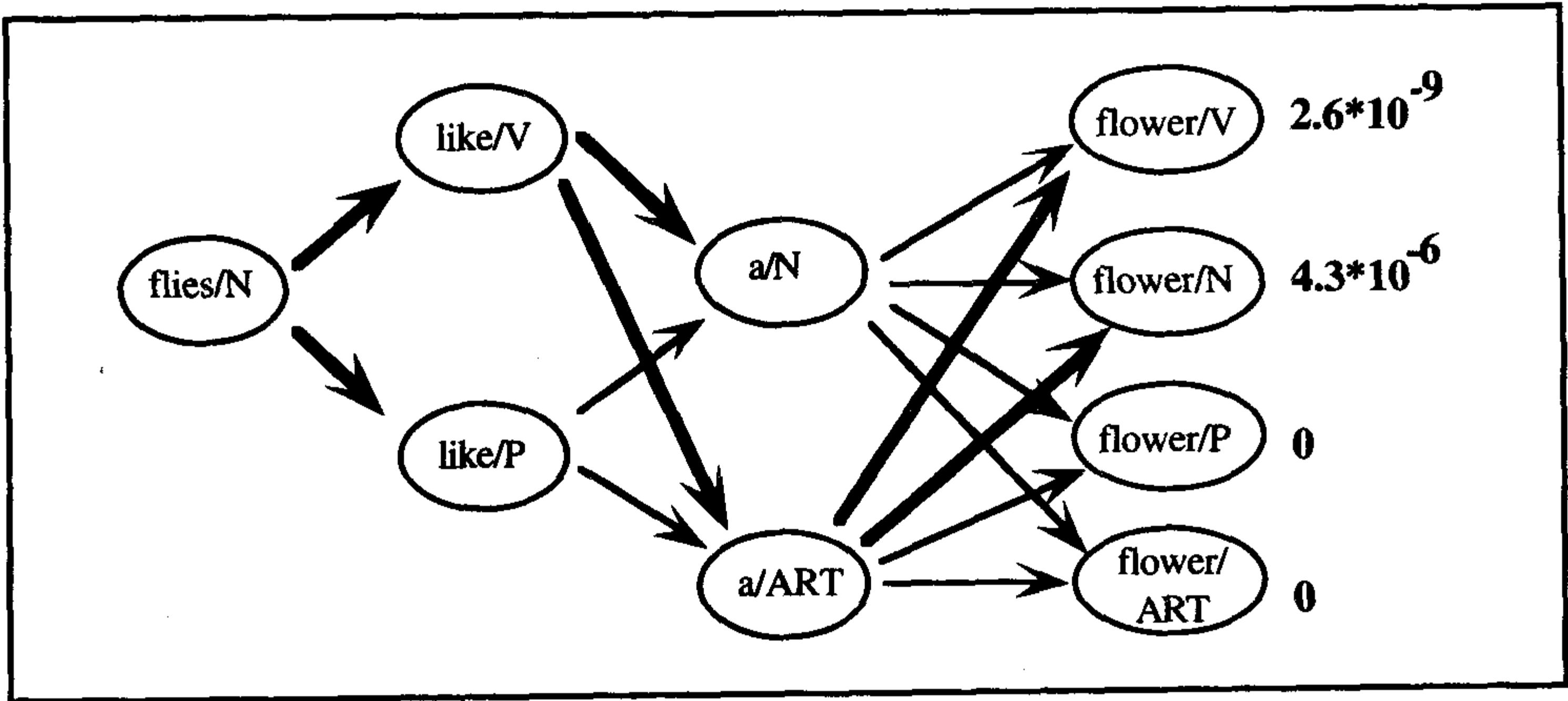


图 7.12 第三次迭代后的结果

7.4 词汇概率的获取

基于语料库的方法提供了一些新的句法分析器的控制方式。如果我们有一些已分析过的大型句子语料库,那么,就可以采取统计方法识别出英语中的一些常见结构,并在句法分析算法中优先生成这些结构。当一个句子存在歧义的时候,统计方法可以帮助我们选择出最可能的解析结果。而且,这也可能生成效率相当高的句法分析器,甚至可能是接近于确定型的句法分析器。这种分析过的句子语料库现在越来越多了。

第一个问题是这种句法分析器的输入会是什么? 一个简单的方法是采用上一节给出的词性标注算法为每个词选择一个词类,然后根据这些词类开始进行句法分析。如果词性标注是准确的,那么,在句法分析开始之前就排除了大量的词法歧义。因此,这将是一个很出色的方法。但是,一旦标注错误,将会导致句法分析器不可能找到正确的解析结果。更糟的是,以错

误标记的词为基础,句法分析器可能会找到一些合法但是根本不真实的解释,而且自身绝不会检测出这种错误。假定平均有 95% 的准确率,在一个只有 8 个词的句子中,每个词都正确标注的概率也只有 0.67;而对于 12 个词的句子来说,这种概率为 0.46——少于一半。因此,这一方法在一般情况下正确无误的几率就很小了。

框 7.1 可靠统计的获取

假设要估计词性和 n 元组的概率,那么,究竟需要多大的数据量才能让这些估计值可靠呢? 实际上,随着 n 的增加,需要估计的概率数目迅速增长,因此,需要的数据量高度依赖于所用 n 元组的长度。例如,典型的标注集包含大约 40 个不同的词类。如果要统计一元概率(在每个词类中,简单地统计词的总数),只需要 40 个统计量,其中每个词类对应一个统计量。在二元语法模型中,需要 1600 个统计量,其中每对词类对应一个统计量。在三元语法模型中,需要 64 000 个统计量,其中每个三元词组对应一个统计量。最后,在四元语法模型中,需要 2 560 000 个统计量。如你所见,即使是有一百万个词的语料库,四元语法模型分析中也会导致大多数类别为空。不过,对一百万个词的语料库应用三元语法,如果统计量分布均匀,则每个类别平均会有 15 个例子。尽管三元概率肯定不是均匀分布的,但是,这种级别的数据量仍然可以取得不错的效果。

有一种称为平滑的技术,对于解决数据稀疏问题非常有用。估计位置 i 上词类为 C_i 的概率时,不是单纯地使用三元语法模型,而是采用一个一元语法模型、二元语法模型以及三元语法模型相结合的公式。使用这一模式,给定前面的词类 C_1, \dots, C_{i-1} , 当前词类为 C_i 的概率可以采用下式估算:

$$\begin{aligned} \text{PROB}(C_i | C_1, \dots, C_{i-1}) \cong & \lambda_1 \text{PROB}(C_i) + \lambda_2 \text{PROB}(C_i | C_{i-1}) \\ & + \lambda_3 \text{PROB}(C_i | C_{i-2} C_{i-1}) \end{aligned}$$

其中, $\lambda_1 + \lambda_2 + \lambda_3 = 1$ 。采用这种估计方法,如果一个三元词组以前从未出现过,在很多需要这一概率的场合下,二元估计或者一元估计仍然可以保证返回一个非零的估计值。典型的情况是,如果 λ_3 比其他参数大得多,那么,三元信息对概率的影响作用最大。这时候,往往可以取得最佳的效果。给定一个特定的训练集,可以设计算法来学习出好的参数值[例如, Jelinek(1990)]。

更恰当的方法是计算每个词以各个可能词类出现的概率。如果将这些概率与给语法规则赋予概率的某种方法结合起来,可以研究出一种句法分析算法,从而为给定的句子找到可能性最大的句法分析结果。

不难看出,最简单的词汇概率估计方法就是统计每个词语以每种词类在语料库中出现的次数。已知词语 w 可能的词类为 L_1, \dots, L_N , 同样, w 以某个词类 L_j 出现的概率可以采用下面的公式估计:

$$\text{PROB}(L_j | w) \cong \text{count}(L_j \& w) / \sum_{i=1, N} \text{count}(L_i \& w)$$

使用图 7.5 给出的数据,可以派生出每个词和词类的上下文无关概率,如图 7.13 所示。

$PROB(ART the) \cong 0.99$	$PROB(N like) \cong 0.16$
$PROB(N flies) \cong 0.48$	$PROB(ART a) \cong 0.995$
$PROB(V flies) \cong 0.52$	$PROB(N a) \cong 0.005$
$PROB(V like) \cong 0.49$	$PROB(N flower) \cong 0.78$
$PROB(P like) \cong 0.34$	$PROB(V flower) \cong 0.22$

图 7.13 词类的上下文无关估计

但是,正如我们以前所知道的,这样的估计值实际上并不可靠,因为并没有考虑具体的上下文。一种更好的估计方法是给定输入 w_1, \dots, w_t , 考虑在所有的词性序列中,词类 L_i 出现在整个序列位置 t 上的概率。也就是说,要计算所有词性序列的概率和,而不是寻找一个以最大概率产生输入串的词性序列。

例如,将所有以名词“flies”结尾的词性序列概率相加求和,可以计算出“flies”在句子“The flies like flowers”中充当名词的概率。图 7.4 和图 7.6 分别给定了转移概率和词语生成概率,因此,可以得到非零值的序列,结果为:

The/ART flies/N	$9.58 * 10^{-3}$
The/N flies/N	$1.13 * 10^{-6}$
The/P flies/N	$4.55 * 10^{-9}$

这些值加起来是 $9.58 * 10^{-3}$ 。类似地,以“flies”作为 V 结尾也有三个序列,概率总和为 $1.13 * 10^{-5}$ 。当第二个词语为“flies”时,这些就是得分非零的全部序列。因此,所有这些序列的概率和就是序列“The flies”的概率,即 $9.591 * 10^{-3}$ 。现在,可以计算“flies”是名词的概率了,计算过程如下:

$$\begin{aligned}
 &PROB(flies/N | The flies) \\
 &= PROB(flies/N \& The flies) / PROB(The flies) \\
 &= 9.58 * 10^{-3} / 9.591 * 10^{-3} \\
 &= 0.9988
 \end{aligned}$$

类似地,“flies”作为动词的概率为 0.0012。

当然,枚举出实际例子的所有可能序列是不可行的。幸运的是,Viterbi 算法中的技巧同样可以用在这里。在算法的每个阶段,不用为每个节点都选择最高得分,而是计算出这些得分的总和。

为了更准确,我们定义前向概率(forward probability),记做 $\alpha_i(t)$,它是产生词语 w_1, \dots, w_t , 并且以状态 w_t/L_i 为结尾的概率:

$$\alpha_i(t) = PROB(w_t/L_i, w_1, \dots, w_t)$$

例如,对于句子“The flies like flowers”来说, $\alpha_2(3)$ 指的是给定输入“The flies like”,在位置 3 上以 V(第 2 个词类)结束的所有序列的概率之和。根据条件概率的定义,可以得出词语 w_t 是词类 L_i 实例的概率,计算公式如下:

$$PROB(w_t/L_i | w_1, ..., w_t) = PROB(w_t/L_i, w_1, ..., w_t) / PROB(w_1, ..., w_t)$$

我们将位置 t 上每个状态的所有可能序列加起来,就可以估计出 $PROB(w_1, ..., w_t)$ 的值,简单地表示就是 $\sum_{j=1,N} \alpha_j(t)$ 。换句话说,我们最终得到:

$$PROB(w_t/L_i | w_1, ..., w_t) \cong \alpha_i(t) / \sum_{j=1,N} \alpha_j(t)$$

这个算法的前两部分采用了 Viterbi 算法的变形计算前向概率,最后一步通过数据的归一化,将给定句子的前向概率转化为词汇概率。图 7.14 给出了整个算法的过程。

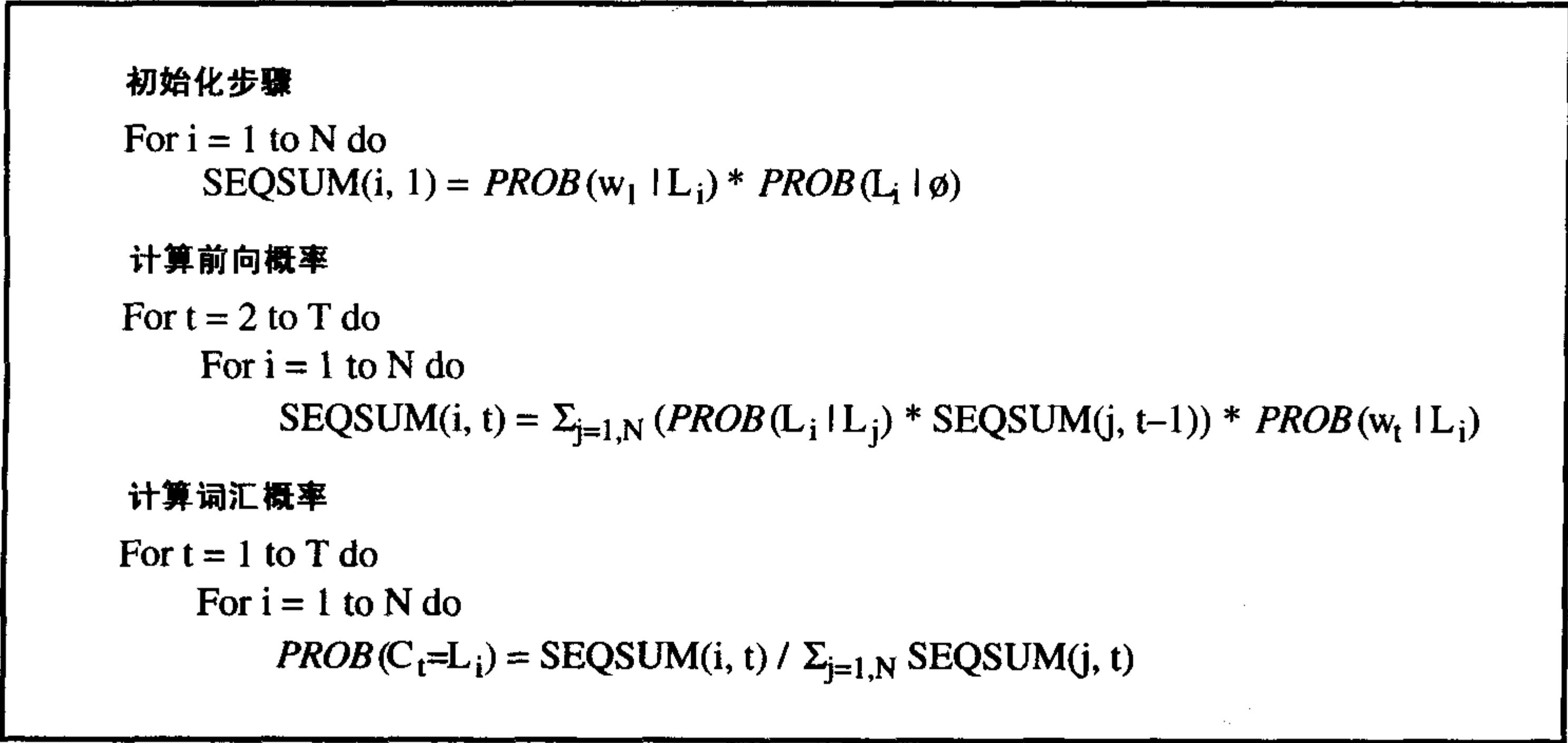


图 7.14 词汇概率计算的前向算法

采用图 7.4 和图 7.6 中的概率估计值,让我们看看如何推导出句子“The flies like flowers”中的词语概率。对于每个位置上的每个词类,图 7.14 中的算法会生成概率之和,如图 7.15 所示。最终,可以得到概率的估计值,如图 7.16 所示。

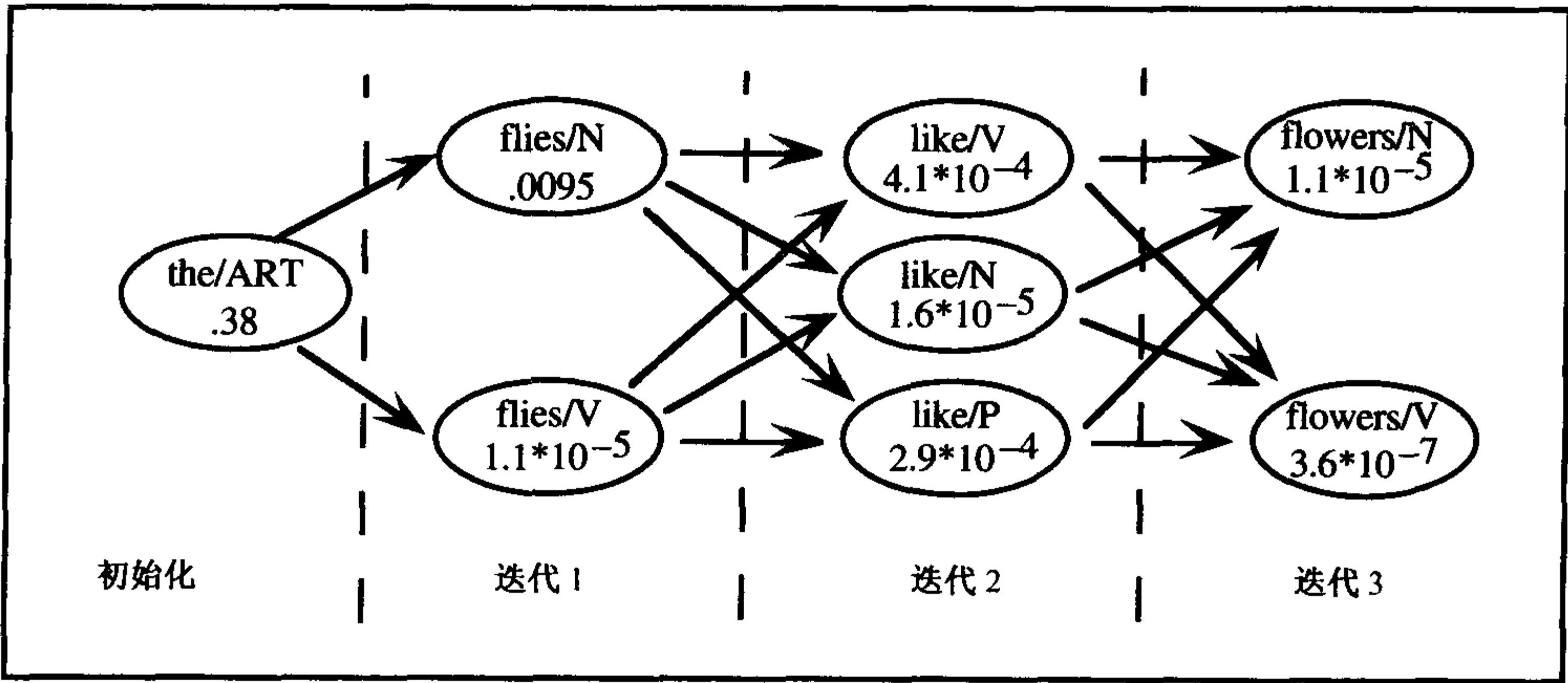


图 7.15 序列概率总和的计算

注意,图 7.13 给出了上下文无关的近似估计,它稍稍偏向“flies”是动词的解析结果;而上下文相关的近似估计却排除了这种解释,因为训练语料的任何句子中动词都不会紧跟在冠词后面。这些概率和上下文无关概率大不相同,它们更符合我们的直觉。

$PROB(\text{the/ART} \mid \text{the}) =$	1.0	$PROB(\text{like/P} \mid \text{the flies like}) \cong$	0.4
$PROB(\text{flies/N} \mid \text{the flies}) \cong$	0.9988	$PROB(\text{like/N} \mid \text{the flies like}) \cong$	0.022
$PROB(\text{flies/V} \mid \text{the flies}) \cong$	0.0011	$PROB(\text{flowers/N} \mid \text{the flies like flowers}) \cong$	0.967
$PROB(\text{like/V} \mid \text{the flies like}) \cong$	0.575	$PROB(\text{flowers/V} \mid \text{the flies like flowers}) \cong$	0.033

图 7.16 句子“The flies like flowers.”中词类的上下文相关估计

注意,也可以考虑使用后向概率(backward probability) $\beta_i(t)$,即从状态 w_t/L_t 开始产生序列 w_t, \dots, w_T 的概率。可以采用与前向概率算法类似的方法来计算这些值,只不过是从句子的尾部开始,依次向后扫描状态。因此,词语 w_t 概率估计的更好方法应当考虑整个句子,而不是仅仅考虑到位置 t 为止的词语。在这种情况下,估计值为:

$$PROB(w_t/L_t) = (\alpha_i(t) * \beta_i(t)) / \sum_{j=1, N} (\alpha_j(t) * \beta_j(t))$$

7.5 概率上下文无关文法

正如有限状态自动机可以推广到概率形式一样,上下文无关文法也能够同样进行推广。为了做到这点,必须对规则的使用进行一些统计。最简单的办法是,在已经分析过的句子语料中,统计规则出现的次数,然后利用这些数据来计算每个规则的使用概率。例如,考虑一个语法类 C ,该语法中包含了 m 条左部为 C 的规则,记为 R_1, \dots, R_m 。可以通过下面的公式计算采用规则 R_j 派生 C 的概率:

$$PROB(R_j \mid C) \cong \text{Count}(\# \text{ times } R_j \text{ used}) / \sum_{i=1, m} (\# \text{ times } R_i \text{ used})$$

语法 7.17 给出了一个概率上下文无关文法,它的概率是从已经分析过的示例语料库中统计而来的。

规则	对 LHS 计数	对规则计数	概率
1. $S \rightarrow NP VP$	300	300	1
2. $VP \rightarrow V$	300	116	0.386
3. $VP \rightarrow V NP$	300	118	0.393
4. $VP \rightarrow V NP PP$	300	66	0.22
5. $NP \rightarrow NP PP$	1023	241	0.24
6. $NP \rightarrow N N$	1023	92	0.09
7. $NP \rightarrow N$	1023	141	0.14
8. $NP \rightarrow ART N$	1023	558	0.55
9. $PP \rightarrow P NP$	307	307	1

语法 7.17 简单的概率上下文无关文法

然后,可以研究出一个在功能上类似于 Viterbi 的算法,即给定一个句子,该算法要找到最有可能生成该句的句法分析树。这个方法要求对规则的使用做一些独立性假设。比如,必须

假定规则 R_j 生成语法成分的概率与该成分作为子成分使用的方式相互独立。这个假定意味着 NP 规则的概率都是一样的,无论 NP 是动词的主语、动词的宾语还是介词的宾语。我们知道这个假设在很多情况下并不正确。例如,与非主语位置的名词短语相比,主语位置的名词短语为代词的可能性更大。但是,正如前面所说的,这些方法在实际使用中可以获得很有用的预测能力。

根据这个假设,可以计算语法成分 C 生成词语序列 w_i, w_{i+1}, \dots, w_j (记为 $w_{i,j}$) 的概率。基于这种概率,可以给出相应的形式化表示。这种类型的概率称为内部概率 (inside probability), 因为它将概率赋给语法成分内部的词语序列,记做:

$$PROB(w_{i,j} | C)$$

应如何生成内部概率? 词类的情况比较简单,实际上,它正好是 7.3 节中给出的词语生成概率。例如, $PROB(flower | N)$ 表示成分 N 中词语“flower”的内部概率,该值在我们假定的语料库中为 0.06,如图 7.6 所示。

使用这样的词语生成概率。然后,可以计算出 NP 生成序列“a flower”的概率。其计算过程为,语法 7.17 中只有两条 NP 规则能产生这两个词语的序列。图 7.18 给出了这两条规则生成的句法分析树。每条规则的概率可以从语料库中估计得到,语料库如语法 7.17 所示。既然已经知道每条规则的概率,那么,成分 NP 产生“a flower”的概率将是两种派生方式的概率之和,计算如下:

$$\begin{aligned} PROB(a \text{ flower} | NP) &= \\ &PROB(\text{Rule 8} | NP) * PROB(a | ART) * PROB(flower | N) + \\ &PROB(\text{Rule 6} | NP) * PROB(a | N) * PROB(flower | N) \\ &= 0.55 * 0.36 * 0.06 + 0.09 * 0.001 * 0.06 \\ &= 0.012 \end{aligned}$$

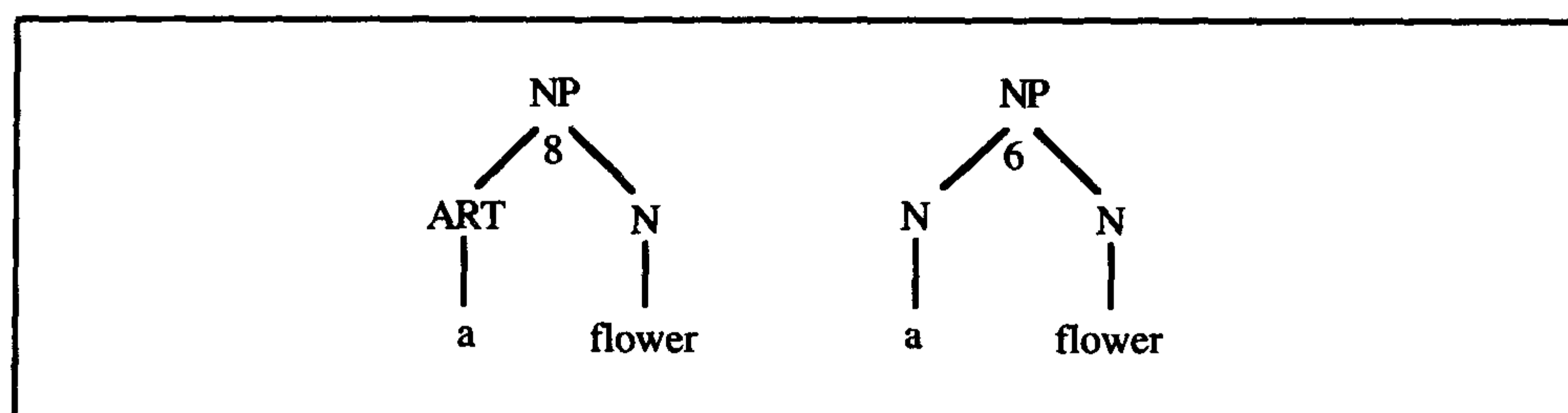


图 7.18 “a flower”分析为 NP 的两种可能途径

接下来,这个概率可以用来计算更大成分的概率。例如,图 7.19 给出词语“A flower wilted” (花儿枯萎了) 每种可能的分析树,将各分析树的概率相加,最终可以计算出成分 S 生成“a flower wilted”的概率。尽管存在三种可能的解析结果,不过,前两种结果的不同之处仅在于“a flower”派生为 NP 的方式不一样而已,而前面 $PROB(a \text{ flower} | NP)$ 的计算已经包含了这两种解析结果。因此,“a flower blooms”的概率为:

$$\begin{aligned} PROB(a \text{ flower blooms} | S) &= \\ &PROB(\text{Rule 1} | S) * PROB(a \text{ flower} | NP) * PROB(blooms | VP) + \\ &PROB(\text{Rule 1} | S) * PROB(a | NP) * PROB(flower blooms | VP) \end{aligned}$$

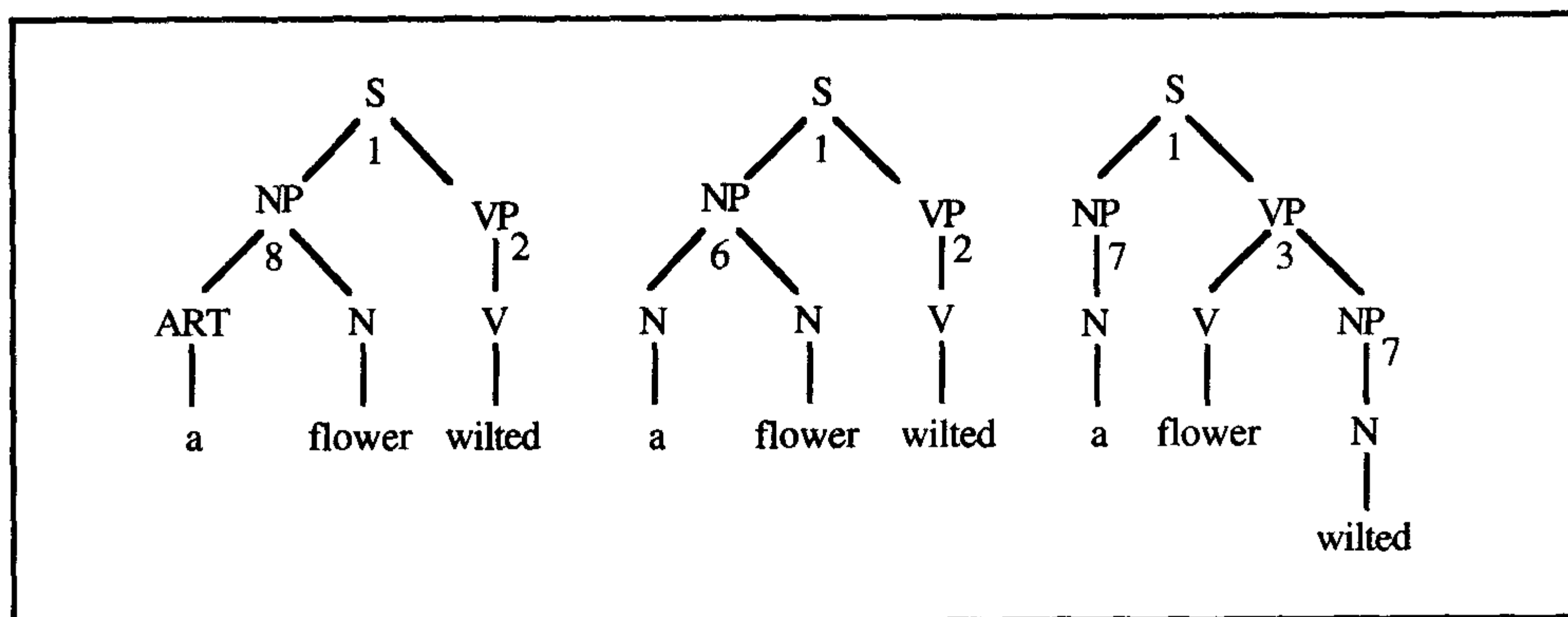


图 7.19 “a flower wilted”作为 S 生成的三种途径

采用这种方法,能够有效计算出语法生成给定句子的概率。这种方法只须记录每两个可能位置之间各个成分的值。实质上,它使用了与压缩 chart 图结构相同的优化方法,在第 6 章中讨论过压缩 chart 图结构(见第 6 章)。

不过,我们并不准备对此进行深入的讨论,因为在句法分析过程中我们感兴趣的是找到最有可能的句法分析结果,而不是给定句子的总体概率。例如,图 7.19 中给出了两种 NP 的解释,我们关心的是究竟采用哪一种解释。采用标准的 chart 分析算法,可以找到特定句法分析树的概率。其中,每个成分的概率可以通过其子成分的概率和使用规则的概率综合计算而来。比如,使用规则 i 生成一个包含 n 个子成分、类别为 C 的词条 E ,则有:

$$PROB(E) = PROB(\text{Rule } i | C) * PROB(E_1) * \dots * PROB(E_n)$$

对于词类来说,最好使用前向概率而不是采用词语生成概率。因为考虑到了句子的一些上下文,所以将产生更好的估计值。可以采用标准的 chart 句法分析算法,不过,向 chart 中添加条目的时候,还需要增加一个步骤,即计算该条目的概率。采用自底向上算法以及从示例语料库中统计得到的概率,可以分析输入“a flower”,图 7.20 给出了其完整的 chart 图。注意,它是一个 NP,这个最直观的解释具有最大的概率 0.54。实际上,还有许多其他的解释,只不过词语“a”作为名词以及词语“flower”作为动词的解析结果概率比较低罢了。并且,依据上下文无关概率,“flower”更偏向于作为动词,但是,在它跟着词语“a”的上下文中,前向算法强烈偏向于名词的解释。使用前向算法,计算出词法成分 ART416, N417, V419 以及 N422 的概率,而上下文无关的词法概率和直觉相差较大。

遗憾的是,尽管使用这些技术构建的句法分析器有用处,但是最终的结果并不如期望的那样有效。有些研究者发现,这些技术能正确识别分析出大约 50% 的句子。不可能做得更好了,原因是必需的独立假设太极端了。问题出自许多假象,而一个关键问题在于词项的处理。例如,上下文无关模型假定 VP 规则中使用特定动词的概率与所使用的规则相互独立。这意味着并不能在基本框架内处理某些规则中的词法优先原则,这个问题会接着影响许多其他的问题,比如附着决策。

例如,语法 7.17 表明,使用规则 3 的机会为 39%,规则 4 的机会为 22%,规则 5 的机会为 24%。这意味着不管使用什么词语,一个形式为 V NP PP 的输入序列中,PP 总会附着在动词之上。图 7.21 给出了这棵句法分析树的片段。任何将 PP 附着在动词上的结构,该部分的概

率均为 0.22;而将 PP 附着在 NP 上的结构概率为 $0.39 * 0.25$,即 0.093。因此,不管使用的是
什么词语,句法分析器总会将 PP 附着在动词短语上。实际上,在语料库中,有 23 个实例都属
于这种情况,即 PP 附着在 NP 上,而概率句法分析器将其混淆了。即使对那些很少采用
_np_pp作为补语的特定动词来说,情况也是如此。因为概率存在上下文无关假设,所以用到的
特定动词对于特定规则的使用概率没有任何影响。由于这种类型的问题,分析真实语法时,基
于该方法的句法分析器比非概率方法提高的程度非常有限。

NP425		0.14
1 N422		
NP424		
1 N417		
2 N422		0.00011
NP423		
1 ART416		
2 N422		0.54
S421		
1 NP418		
2 VP420		3.2×10^{-8}
NP418		
1 N417	0.00018	VP420
		0.00018
N417		
	0.001	N422
		0.999
ART416		
	0.99	V419
		0.00047

图 7.20 “a flower”完整的 chart 图

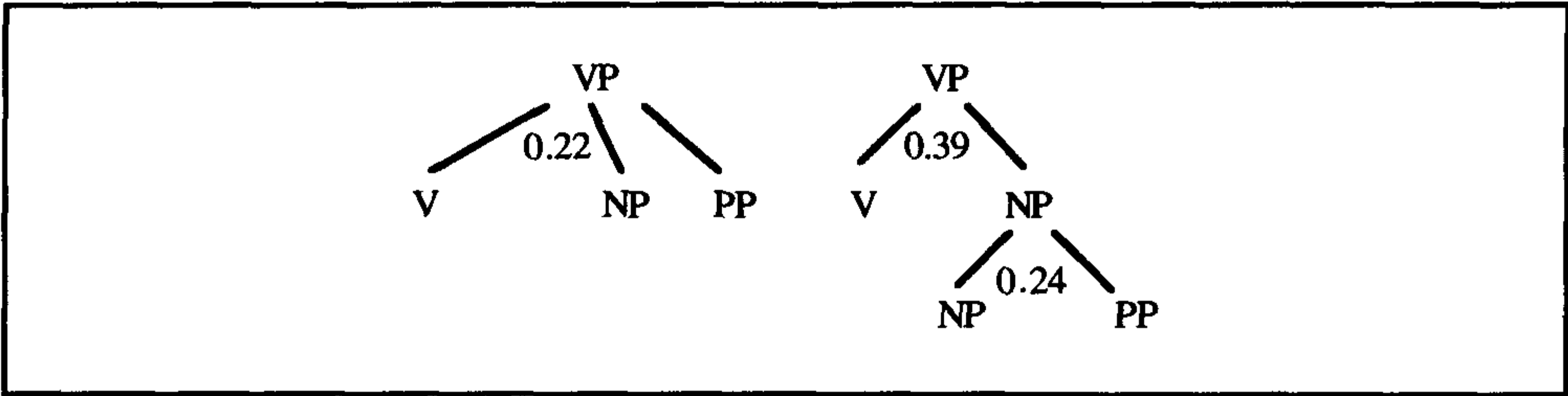


图 7.21 影响附着决策的两种结构

例如,语料库中的 84 个句子都存在 PP 附着问题。根据语法 7.17,采用标准的自底向上的
非概率算法,我们在三分之一的句子中找到的第一个完整 S 结构都是正确的。而使用概率的
句法分析,最高概率的 S 结构有一半是正确的。因此,纯粹的概率句法分析比猜测的要好,但仍
留有大量的提升空间。同时也要注意,在这个测试中,测试用的句子和第一个地方用来计算最高
概率的句子相同。所以,在那些非训练语料的新句子上进行测试,其性能很可能会更差。文献中
有些论文采用了真实语料并进行了综合训练,它们报告的准确性结果与这里给出的相似。

尽管纯粹的概率上下文无关文法自身存在缺陷,但是这里所研究的技术方法仍然很重要。我们尝试研究更多的上下文相关概率文法分析模式,这些方法可以进一步泛化并重新使用。

7.6 最佳优先句法分析

到现在为止,概率上下文无关文法在提高句法分析器效率方面还没有起到什么作用。我们可以研究一些算法来尝试优先找出高概率的成分,这称为最佳优先句法分析算法。我们希望能迅速找到最好的句法分析,而且不用遍历大部分包含较低概率的搜索空间。

实际上,第3章所有的 chart 句法分析算法都可以相当容易地进行改造,使之能够优先生成最可能的语法成分。其核心思想是将待处理表组织成为优先队列——在这个结构里,得分排名最高的语法成分总在队列的首位。然后,句法分析器总是将排名最高的语法成分从待处理表中移除,并将其添加到 chart 图中。

从表面上看,该算法需要进行的全部改进好像就是搜索策略的变化,但是,还有一个复杂的问题。以前的 chart 句法分析算法都依赖于一个事实,即句法分析器会系统地从左向右进行处理。在考虑句子的后一个成分之前,必须处理完前面出现的成分。而在改进算法中,情况并非如此。如果句子的最后一个词语得分最高,那么,它将第一个加到 chart 图中。由此导致的问题是,不能简单地在 chart 图中添加活动边(并且,要依靠算法后面的步骤对它们进行扩展)。实际上,chart 图可能已经包含了需要用来扩展特定活动边的语法成分。因此,对于现有的 chart 图,无论什么时候向其中添加活动边,都必须检查,确定它是否可以立刻进行扩展。除了需要修改步骤2使之检查 chart 图中已有的成分之外,该算法和前面给出的算法相同(见3.4节)。图7.22给出了完整的算法。

在位置 p_1 到位置 p_2 之间添加成分 C :

1. 在位置 p_1 到位置 p_2 之间,向 chart 图中添加 C 。
2. 对于在位置 p_0 到位置 p_1 之间任何一个形式为 $X \rightarrow X_1 \dots \circ C \dots X_n$ 的活动边,添加新的活动边 $X \rightarrow X_1 \dots C \circ \dots X_n$,它在位置 p_0 到位置 p_2 之间。

在 chart 图中,添加一条位于位置 p_0 和位置 p_2 间的活动边 $X \rightarrow X_1 \dots C \circ C' \dots X_n$:

1. 如果 C 是最后一个成分(即该边是完整的),则向待处理表中添加一个类型为 X 的新成分。
2. 否则,如果在 chart 图中存在一个位于位置 p_2 与位置 p_3 之间、类型为 C' 的成分 Y ,则在位置 p_0 与位置 p_3 之间,递归地添加活动边 $X \rightarrow X_1 \dots CC' \circ \dots X_n$ (当然,可能会进一步生成新的边,或者进一步创建成分)。

图 7.22 新的边扩展算法

采用最佳优先策略能极大地提高句法分析器的效率。例如,使用语法7.17以及从语料库中训练出的词典,最佳优先句法分析器生成65个成分之后,就可以正确地分析出句子“The man put a bird in the house”。对于同一个句子,自底向上的标准算法却要产生158个成分,才能得到相同的结果。如果标准算法修改成发现第一个完整的 S 解析后立即中止,仍然要为此句子生成106个成分。因此,最佳优先策略的效率显著提高。

尽管最佳优先句法分析器并不考虑每个可能的语法成分,但是它仍然可以保证找到概率最大的解释。为了进一步说明,可以假定句法分析器找到了一个概率为 p_1 的解释 S_1 。在这

里,概率得分的重要性质是一个成分的概率总是小于(或者等于)其任意子成分的概率。如果还存在一个得分为 p_2 的解释 S_2 ,并且 p_2 比 p_1 大,那么,构成它的所有子成分概率将为 p_2 或者更高。这意味着在 S_1 加入之前,所有的这些子成分都已经加入了 chart 图中。这也意味着构建 S_2 的边是完整的,因此, S_2 会在待处理表中。既然 S_2 的得分比 S_1 高,那么句法分析器会优先考虑 S_2 。

框 7.2 未登录词的处理

未登录词的处理是统计方法表现极优的另一个领域。在传统的句法分析中,未登录词会扰乱整个句法分析。不过,本章讨论过的技术方法可以给我们带来一些很有意思的想法。首先,如果有三元语法模型的数据,对于未登录词的语法类别,你或许已经具备了很强的预测能力。比如,有一个序列 $w_1 w_2 w_3$,其中最后一个词未知。前面两个词类分别为 C_1 和 C_2 ,那么,可以选择使 $PROB(C|C_1 C_2)$ 最大的词类 C 作为未登录词的词类。例如,如果 C_2 为词类 ART,那么 C 最可能的类别或许就是名词(也可能是形容词)。预测词类方法以及其他一些方法利用了词语形态知识。比如,如果未登录词以-ing 结尾,它很可能就是动词;如果它以-ly 结尾,很可能就是副词,依次类推。可以通过语料分析获取基于后缀的估计值,标准的方法是估计 $PROB(\text{词语为类别 } C | \text{以-ly 结尾})$ 等概率。Church(1988),de Marcken(1990) 和 Weischedel 等(1993)均讨论了处理未登录词的技术方法。

尽管最佳优先句法分析的思想在概念上很简单,但是,在实践中试图应用这项技术时,仍然存在很多问题。其中一个问题就是如果使用乘积来综合计算得分,那么,随着输入越来越多,对成分的评分会迅速下降。这看起来好像没有问题,但是在实践中,如果语法巨大,则概率下降会非常快。因此,这里的搜索会非常类似于广度优先搜索,即首先生成所有长度为 1 的成分,然后生成所有长度为 2 的成分,依次类推。因此,快速找到最好结果的期望就无法实现了。为了处理这个问题,有些系统采用了一些不同的函数来计算语法成分的评分。比如,可以采用任意子成分概率与规则使用概率的最小值,即:

$$\text{Score}(C) = \text{MIN}(\text{Score}(C \rightarrow C_1, \dots, C_n), \text{Score}(C_1), \dots, \text{Score}(C_n))$$

这可以得到高于(或者等于)第一个方法的得分。但是,如果存在某个单一的子成分,其得分排名非常糟糕,那么,无论所有其他成分的得分排名如何,任何一个包含它的成分都会被排除。遗憾的是,在同前面一样的 84 个句子上进行测试时,使用 MIN 函数会导致准确率下降很多。其准确率只有 39%,这比原始的搜索方法还要差一点。不过,其他一些研究者认为,这个方法在实践中性能要更好一些。或许也可以尝试其他的综合评分方法,比如取所有子成分的平均得分。

7.7 简单的上下文相关最佳优先句法分析器

最佳优先算法可以提高句法分析器的效率而不影响其准确率。本节将探讨另外一个简单的规则概率计算方法,该方法使用更多的上下文相关词语信息。其思想利用的观察规律为,语法成分中的第一个词语通常都是中心语,并且对其补语规则的概率影响非常大。这就意味着

可以采用与第一个词语相关的新方法来计算规则的概率,即 $PROB(R|C,w)$ 。可以采用下式进行估计:

$$PROB(R|C,w) = \frac{\text{Count}(\text{# 规则 } R \text{ 的次数,其中类别为 } C, \text{首词为 } w)}{\text{Count}(\text{# 类别为 } C, \text{首词为 } w \text{ 的次数})}$$

改进的效果是计算出的概率值与使用的特定词存在着直接的关系。例如,在语料库中,单数名词很少作为名词短语(起始规则为 $NP \rightarrow N$),而复数名词则很少作为名词修饰语(起始规则为 $NP \rightarrow N N$)。给定词语“house”和“peaches”,从这两条规则概率上的差别可以看出这点,如图 7.23 所示。如果使用的是上下文无关概率,则规则 $NP \rightarrow N$ 的概率为 0.14。因此,当输入包含一个复数名词时,这个值低估了该规则的概率;而当输入包含一个单数名词时,这个值又高估了规则的概率。

规则	the	house	peaches	flowers
$NP \rightarrow N$	0	0	0.65	0.76
$NP \rightarrow N N$	0	0.82	0	0
$NP \rightarrow NP PP$	0.23	0.18	0.35	0.24
$NP \rightarrow ART N$	0.76	0	0	0
规则	ate	bloom	like	put
$VP \rightarrow V$	0.28	0.84	0	0.03
$VP \rightarrow V NP$	0.57	0.1	0.9	0.03
$VP \rightarrow V NP PP$	0.14	0.05	0.1	0.93

图 7.23 基于首词的一些规则估计

更重要的是,上下文相关规则可以表达动词不同次范畴的优先级。例如,图 7.23 表明,动词为“put”时,规则 $VP \rightarrow V NP PP$ 的使用概率为 93%;而动词为“like”时,概率值仅为 10%。这种差别使得基于这些概率的分析器能取得比上下文无关概率分析器更好的效果。对同前面一样的 84 个句子进行测试时,上下文无关概率分析器能取得 49%的准确率,而上下文相关概率句法分析器的准确率为 66%,对其中的 14 个句子能正确地解决附着问题,而上下文无关分析器均处理失败。查找句子“The man put the bird in the house”的正确解释时,上下文相关概率句法分析器的效率也要比无关分析器好,只须生成 36 个成分。图 7.24 总结了这三种句法分析策略的最终结果。

策略	84 个 PP 附着问题 准确率	对“The man put the bird in the bouse” 生成的 chart 图的大小
完全句法分析	33%(取发现的第一个 S)	158
上下文无关概率	49%	65
上下文相关概率	66%	36

图 7.24 不同句法分析策略准确率和效率的总结

为什么上下文相关句法分析器能取得更好的效果呢？为了了解这个原因，我们来看看分析器如何在两棵句法分析树之间选择正确的附着关系，如图 7.21 所示。其中，处理的动词是“like”和“put”，它们所在的句子分别为“The man put the bird in the house”和“The man likes the bird in the house”。上下文无关概率句法分析器将两者分析为同一个结果，其中“put”例子的分析是正确的，而“like”错误。图 7.25 和图 7.26 分别给出了相关的部分 chart 图。在图 7.25 中，规则 $VP \rightarrow V NP PP$ 的第一个词为“put”时，概率为 0.93。由此，可以生成语法成分 VP6840，其概率为 0.54($0.93 * 0.99 * 0.76 * 0.76$)。另一个成分 VP6828 的概率为 0.003 8，VP6840 的概率比它要大得多。将动词改为“like”对这些概率影响很大，句法分析器会推翻原来解释为 V-NP-PP 的倾向，如图 7.26 所示。在这个例子中，规则 $VP \rightarrow V NP PP$ 的第一个词为“like”时，概率仅为 0.1，而规则 $VP \rightarrow V NP$ 的概率为 0.9。因此，这使得 VP6883 的概率为 0.1，并战胜了其他的候选结果。

VP6840		
1 V6812		
2 NP6815		
3 PP6822		
0.54		
VP6828		
1 V6812		
2 NP6827		
0.0038		
NP6827		
1 NP6815		
2 PP6822		
0.13		
V6812	NP6815	PP6822
0.99	0.76	0.76
put	the bird	in the house

图 7.25 VP“put the bird in the house”的 chart 图

VP6883		
1 V6848		
2 NP6873		
0.1		
VP6881		
1 V6848		
2 NP6851		
3 PP6858		
0.054		
NP6873		
1 NP6851		
2 PP6858		
0.13		
V6848	NP6851	PP6858
0.94	0.76	0.76
likes	the bird	in the house

图 7.26 VP“likes the bird in the house”的 chart 图

采用适当的上下文信息之后,人们会提出一个问题,即改进后的句法分析器的准确率又如何呢?它能取得 66% 的准确率,与其他策略相比,效果确实很好。但是,在 PP 附着选择上,仍然存在 33% 的错误率。那么,还需要另外增加什么样的信息来提高性能呢?显然,在规则的开始部分,可以采用二元或者三元语法模型,这样就可以将规则概率与输入的较大片段联系起来。如果存在足够多的训练数据,或许可以帮助我们进行最后的选择判断。

附着关系选择不仅依赖于动词,也依赖于 PP 中的介词。根据规则 $VP \rightarrow V NP PP$ 中的前一个语法类、中心动词和介词,可以设计出更复杂的评估体系。这要求我们准备更多的数据来获取可靠的概率统计,不过,这也会让可信度显著提高。然而,其中的困难是现在不能轻易地在规则中进行比较。规则 $VP \rightarrow V NP PP$ 可以采用动词和介词来评估,但是,规则 $VP \rightarrow V NP$ 缺少相应的介词。因此,还需要设计一个更复杂的评估测试办法。

一般来说,如果有足够的数据,词类的区分性越好,估计值的预测结果就会越准确。前面我们也提到了基于词语的统计需要大量的数据,但是,是否存在一部分词语可以单独处理并能取得更好的效果呢?当然,一些类似于介词这样的功能词好像比较适合于单独处理。这些词语的数目是固定的,它们对句子结构的分析有比较大的影响。类似地,其他一些封闭型的词语都可以单独处理,而不要作为同一个词类进行处理,这些词类有冠词、数词和连词等。可以合理地做出假设:我们能获取这些词的足够数据,并做出可靠的估计。

开放型词语则要复杂得多。例如,在数据约束方面,动词和名词扮演着关键的角色。但是,这些词实在是太多了,不可能都单独加以考虑。其中一个办法是只对那些常用的词语进行单独处理,处理方法和我们刚刚介绍的相同。另外一个办法是按照词语的相似性对它们聚类。聚类可以根据语义属性手工实现。比如,从本质上看,所有描述动作的动词可能具备相同的行为特征,因此可以将它们聚成同一个类。另外一种可选的方式是分析存在附着歧义的句子语料库,采用自动的技术方法学习有用的类别。

7.8 小结

在自然语言处理的歧义消解方面,基于统计的方法显示了很强的生命力。在此所使用的统计方法,在词性标注、词汇概率估计以及构建基于概率的句法分析算法等方面均取得了非常好的效果。在词性标注方面,采用二元或者三元概率模型的 Viterbi 算法能够达到超过 95% 的准确率。采用类似于 Viterbi 的算法,通过计算前向概率,可以计算出上下文无关的词汇概率。在使用上下文无关文法的 chart 句法分析器中,可以将这些数据作为其输入,这种句法分析器可以找到最可能的句法分析树。通过使用最佳优先策略,可以大大提高分析器的效率,最佳优先指的是将得分排名最高的成分第一个加入到 chart 图中。遗憾的是,上下文无关概率的框架常常找不到直观上正确的分析树。与上下文无关方法相比,我们给出的上下文相关概率可以显著提高句法分析器的准确率。这些技术方法目前仍然是一个研究热点,在不久的将来,应该可以看到更辉煌的进展。

7.9 相关工作与深入阅读材料

自然语言处理统计方法的文献还只是刚刚起步,所用到的技术方法都是最近十年内研究出来的。Charniak(1993)详细描述了这些技术方法,是这方面的优秀资源。而许多基本算法都

是在自然语言处理之外的领域研究出来的。比如, Viterbi 算法(Viterbi, 1967)最初是在马尔可夫模型中提出来的, 后来广泛用于各种不同的应用之中。而词类概率的估计方法主要基于 Baum(1972)提出的前向算法以及前向-后向算法。在 Rabiner(1989)以及 Waibel 和 Lee(1990)多篇文章中, 都很好地讨论了将这些技术方法如何应用于语音识别。

很多研究者都将这些技术应用于词性标注。在这里, 我们简略地列出一些研究者, 其中包括 Jelinek(1990), Church(1988), DeRose(1988)和 Weischedel 等(1993)。许多研究人员也开发了一些利用上下文无关规则概率的句法分析器。在 7.5 节中介绍过这种概率, Jelinek(1990)就归属其中。基于统计方法的一种主要的吸引力在于, 具备从已处理语料库中有效学习参数的能力。这些算法开始时会给出概率的初始估计, 然后分析处理语料库, 从中计算出更好的估计值。这个过程可以反复进行, 直到不会有进一步的提升为止。这些技术方法可以保证收敛, 但不一定要找到最优值。它们找到的是局部最优值, 因此, 这有点类似于多数梯度爬山算法(hill-climbing algorithms)。Pereira 和 Schabes(1992)给出了这种方法的例子, 其中, 语法参数的学习对象是仅仅进行过浅层分析的数据。

最佳句法分析算法使用了很长一段时间, 最早期的工作可以追溯到 Paxton 和 Robinson(1973)。不过, 最近人们才开始利用语料库中计算出来的数据[比如, Weischedel 等(1993)以及 Chitrao 和 Grishman(1990)]。Magerman 和 Weir(1992)就是一个很好的例子, 其中使用了三元语法模型的上下文相关方法。在《计算语言学》(*Computational Linguistics*)杂志的一本专刊(第 19 卷, 第 1-2 期, 1993)中, 可以找到最近统计模型方面比较出色的工作介绍。

介词短语附着好像是一个非常适合于统计方法的问题。Whittemore 等(1990)详尽地研究了传统的策略, 比如右关联和最小附着, 但发现它们的预测能力都不强, 而词汇优先模型好像很有潜力。Hindle 和 Rooth(1993)研究了一些从不完全句法分析语料库中采集数据的方法, 也就是说, 这个语料库并没有采用句法分析进行完整的标注。

Francis 和 Kucera(1982)详细阐述了 Brown 语料库, 自从发布以来, 该语料库一直作为测试平台被广泛地使用。宾州大学主持的语言数据联盟(LDC, Linguistic Data Consortium)是一个很好的语言数据资源。LDC 拥有许多规模庞大的语言数据库, 其中包括书面形式和口语形式, 而且风格和格式非常多。LDC 也包含了宾州树库, 宾州树库是已经分析标注好的大规模句子语料库。Marcus 等(1993)介绍了宾州树库。

在概率论方面, 现在有许多导论性质的书籍。Ross(1988)是其中比较好的一本书, 介绍了基本的概念, 并简明地讨论了马尔可夫链与信息论。

7.10 习题

1. 【易】根据条件概率的定义, 证明贝叶斯法则。同时, 请证明如果 A 与 B 相互独立, 则 $PROB(A|B) = PROB(A)$ 。
2. 【中】在估计硬币朝上的概率时, 假定可接受的误差区间(margin of error)为 0.4 到 0.6 之间。如果抛 5 次硬币, 请问获得可靠估计的机会是多少?
3. 【中】采用图 7.4 ~ 图 7.6 给出的数据和概率, 针对句子“Flower flowers like flowers”, 请手工模拟 Viterbi 算法。针对这个问题, 请画出如图 7.10 ~ 图 7.12 所示的转移网络, 并判别算法对每个词给出的词性。

4. 【中】根据本章中给出的二元语法模型和词语生成概率。采用前向算法,计算句子“The a flies like flowers”中的词语概率。(其中,词语“a”作为名词使用,这是很罕见的,它用在“the a flies”,“the b flies”等情形下。)记住,表中所有不存在的二元概率,全部设定为0.000 1。得到的结果是否合理?如果不合理,存在的问题是什么呢?怎样才能让结果固定。
5. 【中】使用我们提供的标注语料库,请编写代码完成二元统计与词法统计,并采用 Viterbi 算法实现一个词性标注器。在同一个语料库上测试你的算法,看看最终正确标注的句子有多少个?有哪些词类错误?如果可以得到更多的数据,你认为是否可以在这些错误标注的句子中获得更好的准确率?
6. 【中】在语法 7.17 的基础上,另外添加一条规则如下:

10. $VP \rightarrow V PP$

修正的规则概率如下所示(这里没有提到的规则均与语法 7.17 相同):

$VP \rightarrow V$	0.32
$VP \rightarrow V NP$	0.33
$VP \rightarrow V NP PP$	0.20
$VP \rightarrow V PP$	0.15

另外,下面是与图 7.4 不同的二元概率:

$$PROB(N|V) = 0.53$$

$$PROB(ART|V) = 0.32$$

$$PROB(P|V) = 0.15$$

- a. 针对“Fruit flies like birds”,请手工模拟(或者实现)前向算法,并生成词语概率。
- b. 画出“Fruit flies like birds”完整的 chart 图,要求给出每个成分的概率。
7. 【难】实现一个前向与后向概率派生词类概率的算法,请开发两个词法分析程序:其中一个仅使用前向算法,而另一个可以综合这两种方法。同时,请给出一个句子,要求说明两者的结果存在着本质的差异。
8. 【难】将 Viterbi 算法推广到三元统计模型。现在,位置 t 上的某个状态依赖于它前面的两个状态,因此,当前的问题并不符合马尔可夫假设的基本定义。通常,这也叫做二元马尔可夫假设。最好的方法是将其看成隐马尔可夫模型。其中,隐马尔可夫模型所有状态的标记都是词类对。因此,序列 ART N V N 可以由状态序列(\emptyset , ART), (ART, N), (N, V)与(V, N)生成。这些状态之间的转移概率就是三元概率。例如,状态(\emptyset , ART)到(ART, N)的概率就是词类 N 跟着 \emptyset 和 ART 后面的概率。采用我们提供的语料库,训练你的数据,并实现一个基于三元语法模型的词性标注器。在同一测试数据上,对其性能和二元语法模型进行比较。

第二部分 语义解释

正如引论中所述,判断一句话的意思要分两步来进行。首先,计算出意义的上下文无关的标记形式,称之为逻辑形式(logical form);然后,在上下文中对逻辑形式进行解释,以生成最终的意义表示。本书的第二部分主要讨论第一步,称之为语义解释。很多实际的系统并没有进行这种划分,而是在处理过程的前期就使用了上下文信息。但是,这种安排对于问题的分类和阐明不同的技术还是非常有用的。

语言学中通常也有类似的划分。对上下文无关的意义研究称为语义学,对上下文相关的语言的研究称为语用学。我们依据是否需要上下文信息来划分问题,这样下面一些问题的示例就可以用上下文无关的方法来解决。因此,我们在第二部分中阐述以下这些问题:

- 利用上下文无关的结构约束来排除候选词义。
- 识别每个词和短语在逻辑形式中扮演的语义角色,尤其要识别出谓词/参数和限定关系。
- 识别从句子结构中推导出来的互指约束。

下面的问题虽然也很重要,但并不在第二部分中进行研究,而是留待第三部分的上下文解释部分再加以讨论:

- 判断名词短语或其他短语的所指对象。
- 从多个候选的解释中选出一个解释并确定一组词义。
- 判断每个句子的意图。

第8章将更加详细地讨论逻辑形式和最终的意义表示形式之间的区别,并且提出一种逻辑形式语言,这种语言将在本书剩下的部分中使用。而第9章解决如何将逻辑形式和句法结构相关联的问题,并说明如何利用语法中的特征体系,采用逐条规则的方式来识别逻辑形式。第10章将讨论歧义消解这个重要问题,并且说明如何确认最通顺的词义和语义结构。第11章将讨论一些语义解释的变通方法,它们在现有系统和应用中已经证明有效。最后,第12章将讨论一组更高深的语义解释问题,包括辖域指定依存分析,这些问题是面向那些对语义问题有强烈兴趣的学生而专门挑选出来的。

第8章 语义和逻辑形式

本章介绍一些基本概念,它们构成了意义理论或语义的基础。这里引入了一层上下文无关的意义,称为逻辑形式,它可以直接从句子的句法结构中生成。由于逻辑形式必须上下文无关,因此,它不能包含任何需要用上下文对句子进行解释的分析结果。

8.1节介绍基本的意义概念和语义学,并描述了逻辑形式在语义处理中的角色。8.2节介绍词义和语义原语,并讨论词义的歧义问题。8.3节提出了基本逻辑形式语言,用它来表示句子的上下文无关意义。8.4节讨论用简洁的形式来表示某个常见歧义形式的问题。8.5节讨论了动词的表示形式并介绍了状态和事件变量的概念。8.6节则讨论了论旨角色(或称为格)。对于不同的动词意义,还阐释了如何用论旨角色刻画它们共有的一般语义特征。8.7节介绍浅层言语行为的概念,并讨论如何处理逻辑形式中的内嵌句。这一节将完成对逻辑形式语言的介绍,并在图8.7和图8.8中予以简要说明。8.8节是可选内容,它是为那些想要为逻辑形式语言自身定义一种模型论语义学的学生准备的。其中,描述了一种面向逻辑形式语言的模型论,并讨论了句子之间的各种语义关系,这些关系可用蕴涵和隐含来定义。

8.1 语义和逻辑形式简介

准确定义语义(semantics)和意义(meaning)的概念是一件非常困难的事,因为这些词的原生用法和理论用法有一些不同的使用目的。例如,动词“mean”有一种和语言无关的用法。假设你正在树林里散步,经过一个未燃尽的火堆。你可能会说“This fire means someone camped here last night”。通过这句话,你想表达的意思是,火堆直接或间接证实了你的结论。这和后面几章将要重点讨论的意义的概念相关但又有所不同。我们所指的意义与词的用法定义更接近,例如“‘Amble’ means to walk slowly”。这里采用其他词来定义某个词的意义。为了更准确,我们将研究更形式化的语言,通过这种语言不必借助自然语言本身就可以界定意义。但即便做到这一点,定义句子的意义仍很困难。例如,最近我去过一个机场,当我正走向登机门时,入口的一名警卫问我:“Do you know what gate you are going to?”我对这句话的解释是,问我是否知道自己要去哪里。于是我回答是。但这个回答是基于对警卫意思的错误理解上,当他再问:“Which gate is it?”就清楚地表明他想让我告诉他登机门号。因此,句子“Do you know what gate you are going to?”在不同的上下文中显然有不同的意思。

我们可以定义句子的上下文无关的意义吗?换句话说,是否存在一个层面,在这个层面上句子“Do you know what gate you are going to?”只有一个意思,但却可以表达不同的意图?这是一个很复杂的问题,但是尝试找出一种方法会带来很多好处。首先是模块化。如果能够找出这样一个层面,我们就可以详细研究句子的意义,而不必考虑句子众多复杂的用法。但是,如果句子不存在上下文无关的意义,我们就无法将研究语言和研究一般的人类逻辑推理以及上下文区分开。在后面几章我们会看到,很多约束的例子都以上下文无关的词意义为基础。因此,从现在开始,在表示上下文无关的意思时,我们用“意义”(meaning)这个词,而在上下文相关的

情况下,我们用“用法”(usage)这个词。上下文无关意义的表示称为逻辑形式(logical form)。将一个句子映射到它的逻辑形式的过程称为语义解释(semantic interpretation),将逻辑形式映射到最终的知识表示(KR, knowledge representation)语言的过程称为上下文解释(contextual interpretation)。图 8.1 给出了一个各解释阶段的简单例子,图中标记的具体意义将在以后给出。

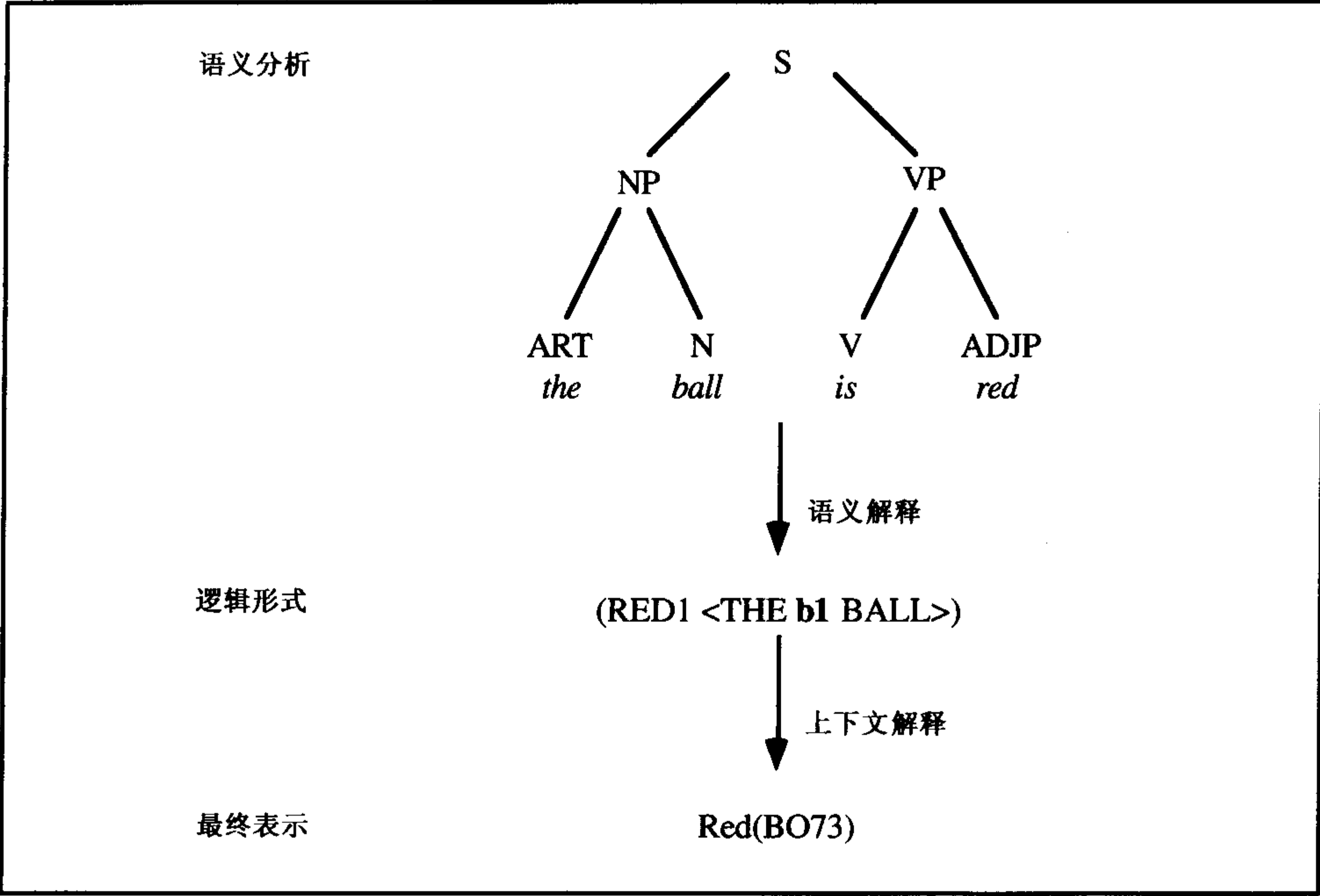


图 8.1 作为中间表示形式的逻辑形式

现在,我们假定知识表示语言等同于一阶谓词演算(FOPC, first-order predicate calculus)。在这个假设下,逻辑形式应该是什么样子呢? 在一些方法中,逻辑形式定义成话语的字面意义,这样逻辑形式语言就和最终的知识表示语言相同。假如从长期来看这个方案可行,则意味着知识表示形式要比现在 AI 系统里所用的表示形式复杂得多。例如,逻辑形式语言必须允许指示词(indexical term)存在,所谓指示词就是由上下文界定的词。代词“I”和“you”都是指示词,因为对它们的解释取决于说者和听者这种具体的上下文。事实上,大多数定指描述(例如,“the red ball”)都是指示词,因为所指的对象只能通过上下文才能确定。其他很多语言问题,包括对时态的解释以及判断量词的辖域,也都取决于上下文,因此无法在逻辑形式这一层面就确定下来。当然,这些问题在逻辑形式层面也可以看成歧义,但这样行不通,因为每个句子都会有很多候选逻辑形式。(例如句子“The red ball dropped”对于颜色是红的,形状是球形的每个物体都有不同的逻辑形式相对应。)

但如果逻辑形式语言不是知识表示语言的一部分,那它应该以什么形式呈现呢? 在过去十年里语言学家已经提出了一种看起来行之有效的方法,这个方法提出了“情境”(situation)的概念,并用它来解答前面的问题。所谓“情境”,就是客观世界环境的一个特定集合,这和英语中“situation”的直观意义非常吻合。例如,在上课时,你处的情境中有学生和老师,有老师所讲的特定话语,有提问,等等。教室里还会有物品,比如黑板和座椅,等等。如果用更形式化的语言来表达,可以认为情境就是一个对象集合以及这些对象之间的关系。一个简单的情境可能只包

含两个对象,一个球 B0005 和一个人 P86;也包括两者的关系,即这个人拥有这个球。我们可以把这个情境表示成集合{(BALL B0005),(PERSON P86),(OWNS P86 B0005)}。

语言基于它所传达的信息生成特定类型的情境,这个问题我们将在以后详细研究。现在,为帮助理解,请先思考下面的情况。在任意会话或文本中,假设存在篇章情境(discourse situation),它记录了到目前为止所传递的信息。对新句子,要根据这个情境进行解释,并且生成包含新句子所传递信息的新情境。根据这个观点,逻辑形式就是一个函数,它将话语产生的会话情境映射到由新话语的出现所产生的新的篇章情境中。举例来说,假设我们刚才表示的情境由前面那些描述球以及球的主人的句子产生,那么句子“The ball is red(这个球是红的)”就会产生一个新的情境。这个情境包含了以前的情境,还有 B0005 拥有属性“红”(RED)这种新信息: {(BALL B0005),(PERSON P86),(OWNS P86 B0005),(RED B0005)}。图 8.2 给出了用这种观点进行解释的过程,它将逻辑形式视为从情境到情境的函数。图 8.1 和图 8.2 所展示的结构有所不同,因为后者可以不包括用知识表示法表示的单一定指表达式,而这种表达式可以完整刻画句子的“意义”。而且,逻辑形式可加以调整变化以产生最新的情境。这样,其他不能直接用句子的语义内容刻画的隐含意义也能够从话语中推断出来。这个问题在我们以后讨论上下文解释的时候很重要。

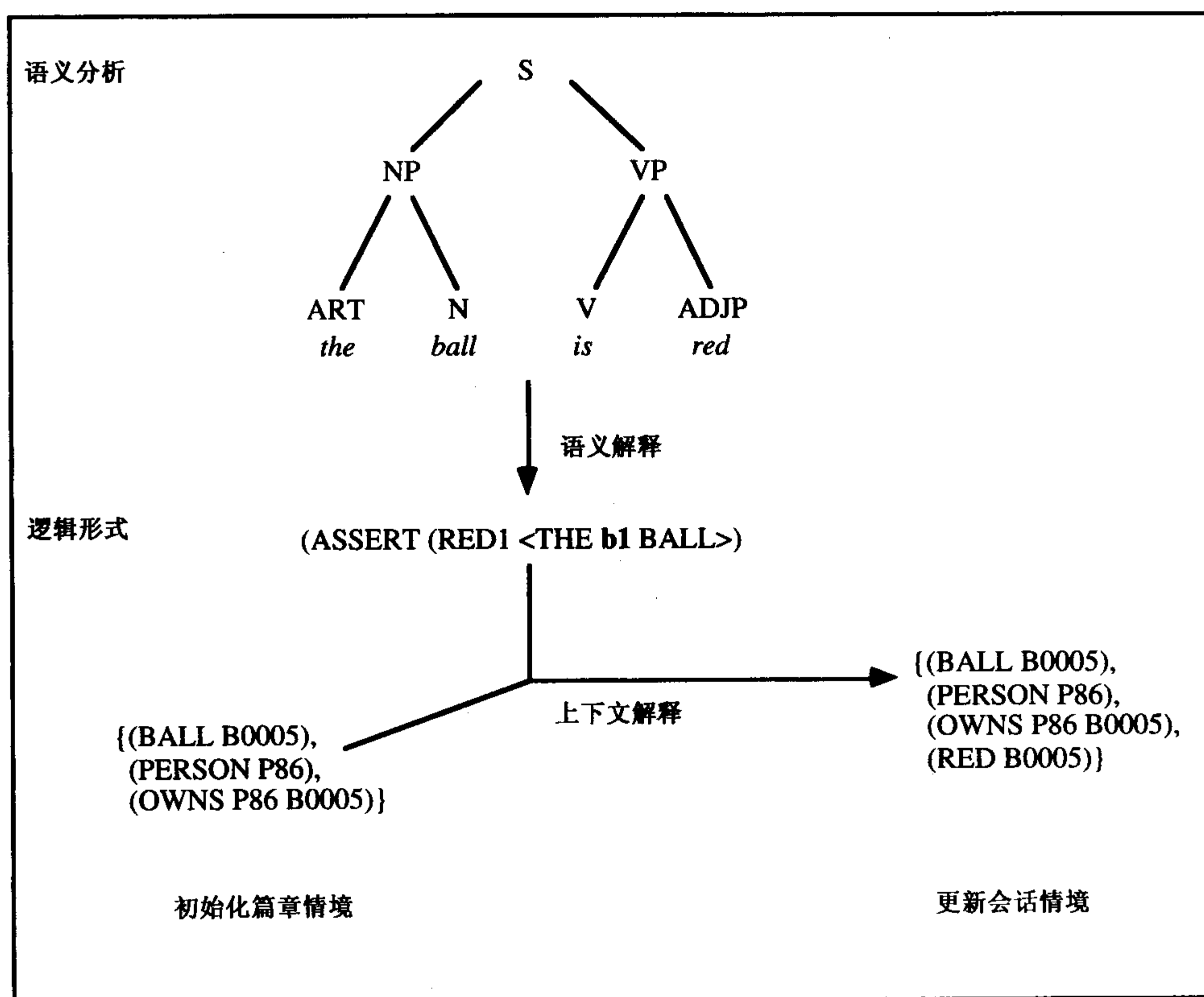


图 8.2 作为函数的逻辑形式

尽管很多语言现象都非常依赖上下文,但在语言中,仍存在相当多的上下文无关的语义结构,这些结构可以通过语义解释过程生成逻辑形式。很多语义知识是由那些可以在词典中找

到的信息组成,包括词的基本语义属性(即,它们是否涉及关系、对象,等等),每个词都有哪些不同意义,哪些意义可以合并形成更大的语义结构,等等。确定这些信息的形式并利用这些信息计算逻辑形式是本书第二部分的重点。

8.2 词义与歧义

要提出一种语义学和语义解释理论,首先需要提出一种结构模型,就像我们在研究句法时做的那样。对句法,我们首先引入了基本句法类的概念,然后提出一些方法用以解决将简单的类合并形成更大的结构的问题。对于语义,将遵循同样的策略。读者也许会认为基本语义单元是词或词素,但这样做将因为歧义的存在而面临太多的困难。例如,一部典型的词典中,动词“go”有40个以上的词条并不鲜见。其中,每个词条对应于这个词的一个不同意义。词典对于特定词义常常给出其同义词。对于“go”,可以找到多种同义词,例如“move”,“depart”,“pass”,“vanish”,“reach”,“extend”和“set out”。其中,很多词都标识出动词“go”的不同意义。当然,如果这些词真是“go”的某个意义的同义词,那么这些动词本身也拥有同样的意义。例如,“go”的一个意义就和“depart”的一个意义相同。

如果每个词都有一个或多个意义,那么即便有些词的意义相同,你还是会面临非常多的意义选择。幸运的是,这些意义可以组织到不同的类别集合里,而依据这些类我们将对整个客观世界进行划分。这一组对象类经过形式化表示之后称为本体(ontology)。为了处理自然语言,我们需要比形式逻辑著作中常见的定义更宽泛的本体。对象的分类长期以来就是一件吸引众多人关注的工作,早在亚里士多德(公元前384~前322)的著作中就已经提到了相关内容。亚里士多德提出的几大类包括物质(实体)、数量(例如数字)、性质(例如鲜红色)、关系、地点、时间、位置、状态、行为和影响。对这个列表,还可以添加一些其他的类,比如事件、思想、概念和计划。其中,最重要的两个类是行为和事件。事件是在客观世界中发生的事情,它在很多语义理论中都很重要,因为它提供一种结构,可以把句子的解释组织起来。行为是行为主体所做的事情,会引发某个事件。就像本体中的所有对象一样,行为和事件都可由代词指代,例如以下篇章片段:

We lifted the box. It was hard work. (我们抬起了箱子。这是一项艰巨的工作。)

这里,代词“it”指抬箱子这个行为(“lifting the box”)。情境是另一个很重要的类。正如前面所指出的那样,情境指某些情况的特定集合,可以认为它包含了事件的概念。在很多情况下,情境就像是某个地点某个时间的客观世界的缩影。例如,句子“We laughed and sang at the football game”(我们在足球赛时又笑又唱)描述在特定时间和特定地点发生的一组活动,可以用“the football game”这个情境加以描述。

毫无疑问,歧义在语义解释阶段依然是很严重的问题。如果一个词映射为一个以上的意义,我们就称这个词有语义歧义(semantic ambiguity)。但是这个问题比乍看上去要复杂,因为需要一种方法来判断哪些才是可用的意义。例如,直观上“kid”这个词是有歧义的,它既可以指小山羊,也可以指儿童。但我们又怎么知道它没有一个意义能同时包含这两种解释?另一方面,虽然我们知道马(horse)实际上可以分为母马(mare)、马驹(colt)、赛马(trotter),等等,但“horse”这个词却似乎没有歧义。那为何每次用这个词而我们又不能分辨它指的是母马还是马

驹的时候,它仍然不算有歧义呢?为了使语义歧义的定义更准确,人们曾经进行了一些语言学测试。其中一个较为成功的测试利用了一个性质,即特定的句法结构通常指的是同一个对象类。例如,句子“I have two kids and George has three”可能是说我和乔治都是放养山羊的农夫,也可能是说我们都有孩子,但不可能是两者的组合(我养山羊而乔治有孩子)。另一方面,即便我有的是马驹而乔治有的是母马,你也可以说“I have one horse and George has two”(我有一匹马,而乔治有两匹)。因此,这个测试提供了一个方法来检测我们对词义的直观知识。词“kid”是有歧义的,因为它有两个意义, BABY-GOAT1 和 BABY-HUMAN1;而“horse”是没有歧义的,虽然它也同时有 MARE1 和 COLT1 等意义。相反,它的一个意义 HORSE1 同时包含这两个意义。这说明了很重要的一点,即某些意义更为明确。这种性质常常称为“模糊性”(vagueness)。HORSE1 这个意义在某种程度上是模糊的,因为它没有区分母马和马驹。当然,如果考虑到 MARE1 这个意义没有指明是大母马还是小母马,那它也是模糊的。事实上,所有的意义都有一定程度的模糊性,因为它们总是可以表达得更为明确。

对动词语义也可以做一个类似的歧义测试。例如,句子“I ran last year and George did too”可能指我们两个都是参加选举的候选人,也可能是指我们两个都参加了跑步比赛,但是一般不会说这句话的意思是两者的混合。由于同时有 RUN1(运动意义上的)和 RUN2(政治意义上的)两个意义,因此 run 这个词是有歧义的。与之不同,动词“kiss”是模糊的,因为没有指明亲吻的部位。可以说“I kissed Sue and George did too”,即便我亲的是她的脸颊而 George 亲的是手。

除了词的歧义,语义级还有很多结构性歧义。有些类型的歧义是由底层的句法歧义引起的。例如,句子“Happy cats and dogs live on the farm”是有歧义的,因为未表明狗是否也快乐(也就是说,句子可能是指快乐的猫和快乐的狗,也可能是说猫很快乐,而狗怎么样则没有说明)。尽管这种歧义在语义层面呈现出来,但它实际根源于句法结构。也就是说,不能判断连词连接的究竟是两个名词短语(Happy cats)和(dogs),还是单个名词[Happy(cats and dogs)]。但其他形式的结构歧义则真正是语义层面的,即这种歧义是从单一的句法结构中产生的。一个常见的例子就是量词的辖域指定。例如,句子“Every boy loves a dog”究竟是说所有男孩喜欢的是同一条狗,还是说每个男孩喜欢的是不同的狗?每种情况下句法结构都是一样的,而区别就在于数量词的辖域作用到哪里。举例来说,这两种意思严格对应于下面的一阶谓词演算表达式:

$$\begin{aligned} \exists d. Dog(d) \& \forall b. Boy(b) \supset Loves(b, d) \\ \forall b. Boy(b) \supset \exists d. Dog(d) \& Loves(b, d) \end{aligned}$$

因此,尽管“Every boy loves a dog”只有一种句法结构,但它的语义结构却是有歧义的。如果考虑到模糊性,那么各个量词也有一些不同。量词“all”是精确的,它很明确地表示某集合中的每一个成员,而一些量词,比如“many”,在“Many people saw the accident”(很多人看到了这场事故)中就是模糊的,因为没有说明到底指多少人。

你可能会认为指示词也是有歧义的,比如“you”,“I”和“here”,因为它们的解释取决于上下文。但这并非我们此处所讨论的歧义的意思。需要注意的是,不能因为名词短语“the dog”可能会指很多种狗,就认为“dog”这个词有歧义。“the dog”的语义意义是能够准确判断出来的。同样,从代词“I”指代的对象来看,可能也是不确定的,但它有惟一的明确的意义。

短语的指代对象上下文相关,这超出本章的讨论范围,但关于指代对象有一些上下文无关的约束必须加以说明。分析句子“Jack saw him in the mirror”。虽然没有说明看到的是谁,你也能知道所说的不会是杰克看到了他自己。要表达这种意思,可以使用反身代词,即“Jack saw

himself in the mirror”。因此,句子的结构会产生特定的指代约束。这个话题我们会在第12章加以探讨。

关于上下文无关意义的一个非常重要的现象是词义之间存在共现约束。通常正确的词义可以根据结构和句子其他部分的意义确定下来。考察动词“run”,它有一个意义指的是慢跑时的动作,还有一个意义是指开动某台机器所做的动作。第一个意义通常用做不及物动词(比如“Jack ran in the park”),而第二个意义通常只能是及物动词(比如“Jack ran the printing press for years”)。其他情况下句法结构相同,词的各种可能的意义要用特定的方法进行组合。例如,分析“Jack ran in the park”和“Jack ran in the election”这两句话。此二者的句法结构相同,但由于修饰部分的意义不同,因而动词“run”的意义必须从几个可能的选项中进行挑选。语义解释最重要的任务之一就是利用这样的约束来减少每个词的候选意义的数目。

8.3 基本逻辑形式语言

上一节介绍了基本语义单元,也就是词义。本节则介绍一种语言,通过它可以把基本单元组合起来表达更复杂的意思。这种语言与一阶谓词演算很相似,尽管其中有很多表达式使用了等价形式,例如基于网络的表达式,但其基本思想一样。词义将作为表达式的原子(atom)或者常量(constant)。这些常量可以根据所描述事物的类型进行分类。例如,描述客观世界对象,包括诸如事件和情境这种抽象对象的常量,称为词项(term)。描述关系和属性的常量称为谓词(predicate)。谓词后跟适当数量的词项作为参数就构成语言中的命题(proposition)。例如,对应句子“Fido is a dog”的命题由词项 FIDO1 和谓词常量 DOG1 构成,其形式可以写成:

(DOG1 FIDO1)

只跟一个参数的谓词称为一元谓词或属性;像 LOVES1 这样可跟两个参数的称为二元谓词;可跟 n 个参数的称为 n 元谓词。对应于句子“Sue loves Jack”的命题包含一个二元谓词 LOVES1,其形式可写成:

(LOVES1 SUE1 JACK1)

在英语中,可以找到对应于逻辑形式中不同类型常量的各种词类。专有名词,比如“Jack”,有词项的词义;普通名词,例如“dog”,有一元谓词的词义;而动词,比如“run”,“love”和“put”,都有对应于 n 元谓词的词义。这里, n 的取值取决于动词的次范畴需要多少个词项。

值得注意的是,虽然这里逻辑形式是用谓词-参数形式表示的,但其他大多数意义表示形式也有同样的区别。例如,网络表示形式有对应于语义的节点和表示谓词-参数结构的边。用语义网络形式表示句子“Sue loves Jack”的意义,可以得到图 8.3 所示的两种形式之一。对于大多数应用来说,这些表示形式体系都是等价的。

更复杂的命题要用一类称为逻辑运算符的新常量来表示。例如,运算符 NOT 能构造一类命题,这类命题表示某个命题非真。对应于句子“Sue does not love Jack”的命题是:

(NOT(LOVES1 SUE1 JACK1))

英语中还有把两个或更多命题合并成一个复杂命题的运算符。一阶谓词演算中存在诸如析取(\vee)、合取($\&$)、蕴涵(\supset)以及其他形式的运算符(一阶谓词演算中有 16 种真值函数二元

运算符)。英语也有类似的运算符,包括 or, and, if, only if, 等等。自然语言中的连词常常涉及句子间更复杂的关系。例如,连词“and”可能和逻辑运算符“&”对应,但它常常还涉及时序问题。例如,在“I went home and had a drink”中,回家发生在喝饮料之前。另一方面,连词“but”和“and”很相似,但是其第二个参数是听者根据第一个参数所不期望发生的事情。这种命题的一般形式是(连词 命题 命题)。例如,句子“Jack loves Sue or Jack loves Mary”的逻辑形式是[OR1 (LOVES1 JACK1 SUE1)(LOVES1 JACK1 MARY1)]。逻辑形式语言中同时存在对应于词义的运算符和类似于“&”这种直接来自一阶谓词演算的运算符。来源于逻辑的运算符用于连接那些在句子中没有显式连接在一起的命题。

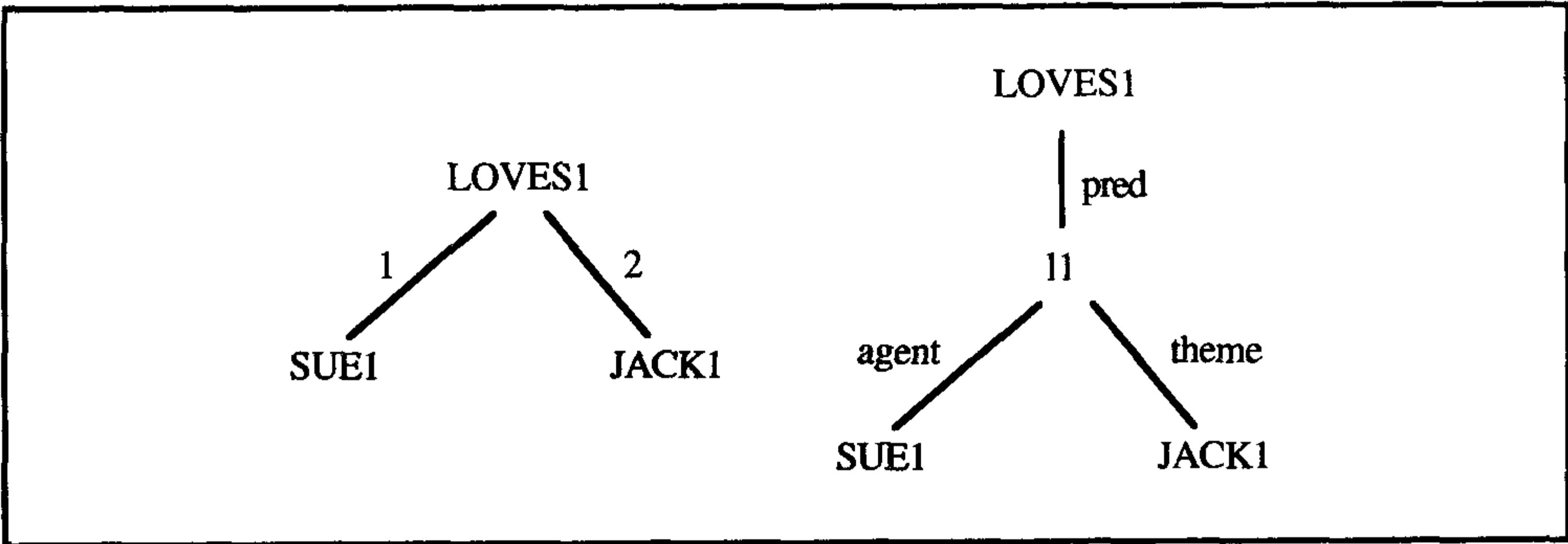


图 8.3 “Sue loves Jack”的两种可能的网络表示形式

利用刚才定义的简单命题,我们只能定义英语中一组非常有限的子集的意义,也就是那些由简单动词形式和专有名词组成的句子。为了说明更复杂的句子,必须定义其他的语义结构。其中一个重要结构就是量词。在一阶谓词演算中,只有两个量词: \forall 和 \exists 。英语中的量词范围大得多,包括 all, some, most, many, a few, the, 等等。要使用量词,变量要以一阶逻辑的形式引入,但同时还存在一个很重要的差异。在一阶逻辑中,变量只在其量词辖域内起作用。因此,出现在两个不同公式中的同一个变量 x 的两个实例[比如,在公式 $\exists x, P(x)$ 和 $\exists x, Q(x)$ 中]要当做彼此之间毫无关联的两个完全不同的变量来处理。自然语言的情况则完全不同。例如,分析下面两个句子“A man entered the room”(一个男人进了房间)和“He walked over to the table”(他跨过了桌子)。第一个句子在会话中引入了一个新的对象,也就是某个男人。你可能想按逻辑中存在量词的用法来处理这句话的意思。但问题是,在第一句话中引入的“the man”在第二句话中是用代词“He”来指代的。因此,引入之后变量好像还继续存在。考虑到这种情况,每次引入一个篇章变量,就赋予它一个以前未用过的独一无二的名称。在正常的情况下,后面的句子会指向这个词项。

自然语言的量词有严格的作用范围,因此比一阶谓词演算中的量词更复杂。在一阶谓词演算中,范式 $\forall x. P_x$ 为真,当且仅当对于论域中每种可能的对象 P_x 为真(也就是说, x 可能是语言中的任何一个词)。在自然语言中,这种句子是很罕见的。更可能出现的情况是,我们也许会说“all dogs bark”或者“most people laughed”,它们有一种通常称为广义量词的结构。这些量词一般用于如下形式的句子中。

(量词变量:限制命题 主体命题)

例如,句子“Most dogs bark”有如下逻辑形式:

(MOST1 d1:(DOG1 d1)(BARKS1 d1))

这表示 d1 中的大多数对象都满足(DOG1 d1),同时也满足(BARKS1 d1)。需要注意这和下面公式的意义迥然不同:

(MOST d2:(BARKS1 d2)(DOG1 d2))

这个公式表达的是“Most barking things are dogs”(大多数会吠的东西都是狗)的意义。

有一类很重要的广义量词对应于冠词“the”和“a”。句子“The dog barks”(狗会吠)的逻辑形式为:

(THE x:(DOG1 x)(BARKS1 x))

这个逻辑形式为真,仅当上下文中有一条可以惟一确定的狗,而且这条狗吠了。显然,在客观世界的任何自然环境中都会有很多狗,因此对于句子的理解而言,借助上下文来确定正确对象就是关键因素。由于这个判断过程需要用到上下文,所以将这个问题的讨论推迟到第三部分进行。这里,只要能找到写出逻辑形式的方法就够了。与冠词(或者不带冠词的没有任何限定的名词短语)相对应的特殊量词集如图 8.4 所示。

量词	用途	例子
THE	定指性指代	the dog
A	非定指性指代	a dog
BARE	无修饰成分的单数 NP(不可数名词)或	water, food
BARE	无修饰成分的复数 NP(泛指)	dogs

图 8.4 一些常见的量词

名词短语越复杂,产生的约束越复杂。例如,句子“The happy dog barks”(快乐的狗会吠)涉及到合取约束,即(THE x:(&(DOG1 x)(HAPPY x))(BARKS1 x))。该式为真仅当上下文中存在惟一的 x 使(&(DOG1 x)(HAPPY x))为真,而且 x 吠了。

为了处理像“the dogs bark”这种短语中的复数形式,需要引入另一种结构。这里,引入一种新的常量,称为谓词运算符,它以谓词作为参数并生成一个新的谓词。复数形式需要用到谓词运算符 PLUR。如果 DOG1 这个谓词对任何一条狗为真,那么(PLUR DOG1)这个谓词对于任何一群狗都为真。因此,句子“The dogs bark”的意义可以表示为:

(THE x:((PLUR DOG1) x)(BARKS1 x))

复数名词短语也可能导致另外一种歧义。请注意,“The dogs bark”的直观解读是,有一群狗而且其中的每条狗都叫了。这种解读称为“个体性解读”(distributive reading),因为谓词 BARKS1 被分配给集合中的每个成员。相反,考察句子“The dogs met at the corner”(狗在墙角相遇了)。在这个例子中,说每条狗相遇了是没有意义的,相反,理解成这群狗聚在一起才是对的。这种情况称为“整体性解读”(collective reading)。有些句子可以同时用两种方法解释,因此会产生歧义。例如,句子“Two men bought a stereo”(两个人买了一台立体声音响)既可以解释成两个人分别买了一台立体声音响(个体性解读),也可以说两个人合起来买了一台立体声音响(整体性解读)。

最后要介绍的一种结构是情态运算符。这种运算符在表示像“believe”和“want”这样的动词、时态以及其他一些结构时需要用到。情态运算符和逻辑运算符很相似,但也有一些重要的区别。具体地说,对情态运算符辖域内词的解释和通常情况下不同,这会影响到我们从命题中得到的结论。例如,假设有些人把 Jack 叫做 John,那么这两个语义是等价的,即 JACK1 = JOHN22。对一个简单命题,使用哪个常量都没有影响:如果(HAPPY JOHN22)为真,那么(HAPPY JACK1)为真,反之亦然。即使是在由逻辑运算符组成的复杂命题中这也成立。如果(OR (HAPPY JOHN1)(SAD JOHN1))为真,那么(OR (HAPPY JACK1)(SAD JACK1))也为真,反之亦然。但是,如果同样的命题放在像“believe”这样的情态运算符辖域内,就不能互相转换了。例如,如果“Sue believes that Jack is happy(Sue 相信 Jack 很高兴)”,即:

(BELIEVE SUE1(HAPPY JACK1))

那么“Sue believes John is happy(Sue 相信 John 很高兴)”就未必是真的,即:

(BELIEVE SUE(HAPPY JOHN22))

因为 Sue 可能不知道 JACK1 和 JOHN22 是同一个人。因此,当等价词出现在情态运算符辖域内时,不能随意替换。在情态上下文中这种现象常称为替换失效(failure of substitutivity)。

自然语言中一类很重要的情态运算符是时态运算符 PAST, PRES 和 FUT。到目前为止,所有的例子都没有考虑时态的作用。但借助这些新的运算符,可以表示出“John sees Fido”, “John saw Fido”和“John will see Fido”之间的差异,即如下命题:

(PRES(SEES1 JOHN1 FIDO1))

(PAST(SEES1 JOHN1 FIDO1))

(FUT(SEES1 JOHN1 FIDO1))

我们知道这些是情态运算符,因为它们都有替换失效现象。例如,考察运算符 PAST,假定两个常量,设为 JOHN1 和 PRESIDENT1,它们现在是等价的,说明 John 现在是总统。但以前 John 并不是总统,因此 JOHN1 并不等同于 PRESIDENT1。已知这些情况,而且存在这样一个事实,即“John 以前见到过 Fido”,(PAST(SEES1 JOHN1 FIDO1))。不能据此得出结论,说总统以前见到过 Fido,因为 John 那时还不是总统。还要注意的是命题和它的否定形式在过去的某个时刻(在不同的时间)可以都为真。因此,“John was happy”(John 曾经很高兴)和“John was not happy”(John 曾经不高兴)这两个句子可能都为真,也就是说,(PAST(HAPPY JOHN1))和(PAST(NOT (HAPPY JOHN1)))都可以为真。

这一节介绍了基本逻辑形式语言的特点。下一节讨论各个引申部分,使这种语言可以更方便地表示歧义并刻画语义规律。

8.4 逻辑形式中的歧义表示

上一节定义了很多用于确定句子逻辑形式的结构,如果你只对逻辑形式的性质感兴趣,请到此为止。但用于计算机处理的表达形式还有另一个很重要的难题,即对歧义的处理。一个典型的句子有多个可能的句法结构,其中每一个都会有多种可能的逻辑形式。而且,句子中的词也有多个意义。简单地列举出所有可能的逻辑形式是行不通的。与之相反,应当用一种方

法排除常见的歧义并在逻辑形式内部表示出来。而且,应当开发一种技术,能根据句子其他部分以及上下文提供的约束逐步解决这些歧义。很多研究者把这种歧义表示看做从逻辑形式中切分出的一层表示形式,通常称为准逻辑形式(quasi-logical form)。

框 8.1 引入广义量词的必要性

对于标准的存在量词和全称量词,在标准一阶谓词演算范式中有若干范式与这些广义量词的形式等价。举例来说,范式

$$(\text{EXISTS } x : P_x Q_x)$$

等价于:

$$\exists x . P_x \& Q_x$$

而全称量词形式

$$(\text{ALL } x : P_x Q_x)$$

等价于:

$$\forall x . P_x \supset Q_x$$

这些广义量词形式可以简单地看成缩写形式。但其他量词在标准一阶谓词演算中没有等价形式。要说明这一点,让我们试着用量词的标准语义来定义“Most dogs bark”的意义。显然, $\forall x . \text{Dog}(x) \supset \text{Bark}(x)$ 语义过强,而 $\exists x . \text{Dog}(x) \& \text{Bark}(x)$ 的语义又太弱。可以定义一个新的量词 M 使得 $Mx . P_x$ 的某种范式拥有正确的真值条件吗? 假设我们定义论域中一半以上的项满足 P_x 时, $Mx . P_x$ 为真。分析 $Mx . \text{Dog}(x) \& \text{Bark}(x)$ 。该式为真仅当论域中一半以上的个体满足 $\text{Dog}(x) \& \text{Bark}(x)$ 。但即便所有的狗都叫了,这也未必为真。更明确地说,要求论域中一半以上的个体都是狗! 那么可能 $Mx . \text{Dog}(x) \supset \text{Bark}(x)$ 能奏效。如果论域中一半以上的个体不是狗或者不叫,那么该式为真。而在有 11 个对象的模型中,如果 10 只海豹叫了,而 1 只狗没叫,则该式也为真! 还可以对 P_x 试试别的范式,但没有一种简单的方法能提供令人满意的解决方案。

逻辑形式中的歧义可能多数源自大多数词都有多个意义这一原因。有些词义有不同的结构属性,因此可以根据前后句子的上下文排除掉。但很多时候词的不同意义有相同的结构约束。目前,表示这些意义的惟一方法就是为句子中的每种可能的词义组合分别构建其逻辑形式。为了避免逻辑形式的数目激增,可以借用和处理句法结构中多个特征值相同的方法来处理。即,任何只能用一个原子语义的地方,都用候选原子语义的集合来表示。例如,名词 ball 至少有两种意义: BALL1,比赛中使用的一个东西; BALL2,和跳舞有关的一种社交活动。因此,句子“Sue watched the ball”在脱离上下文时是有歧义的。然而,一个逻辑形式就可以表示这两种可能的意义:

1. (THE **b1**:({BALL1 BALL2} **b1**)(PAST(WATCH1 SUE1 **b1**)))

上式是两种可能的逻辑形式的缩写形式,即:

2. (THE **b1**:(BALL1 **b1**)(PAST(WATCH1 SUE1 **b1**))), 和
 3. (THE **b1**:(BALL2 **b1**)(PAST(WATCH1 SUE1 **b1**)))

逻辑形式中最复杂的歧义形式之一就是量词和运算符的相对辖域的界定。在 8.1 节中, 我们看到像“Every boy loves a dog”这样的句子是有歧义的, 因为它随量词辖域的不同而有两种解释。任何一种上下文无关的方法都无法解决这种问题, 因此歧义必定会在句子的最终逻辑形式中表现出来。与其一一列举所有可能的辖域, 并因此导致解释的数量随辖域指定结构的数量呈指数级增长, 还不如在逻辑形式语言中引入其他缩写形式, 从而将这些解释合并到一起。更明确地说些, 缩写的逻辑形式根本就不包含辖域指定信息。相反, 像广义量词这样的结构应该像对于词那样进行句法分析, 并出现在句法结构要求它们出现的位置上。用尖括号将其括起来, 表明它们代表其辖域的缩写形式。例如, 句子“Every boy loves a dog”的逻辑形式就可以用一个简单但有歧义的形式表示为:

(LOVES1 < EVERY **b1**(BOY1 **b1**) > < A **d1**(DOG1 **d1**) >)

它是下面两个逻辑形式的有歧义的缩写形式:

(EVERY **b1**:(BOY1 **b1**)(A **d1**:(DOG1 **d1**)(LOVES1 **b1** **d1**)))

和

(A **d1**:(DOG1 **d1**)(EVERY **b1**:(BOY1 **b1**)(LOVES1 **b1** **d1**)))

然而, 在这个例子中缩写的好处并不明显。考虑这样一个句子, 有 4 种辖域结构, 因而就有 24(4 的阶乘)种可能的排列, 而有 5 种辖域结构的句子就有 120 种排列。缩写方法能把所有这些形式压缩成一种表示形式。第 12 章将利用一些启发式技术来判断运算符的辖域。但现在, 我们可以合理地假设, 上下文无关的辖域约束不需要表示出来。

如果广义量词约束是一个简单一元谓词的命题, 那么要再缩写一次把变量删除。因此, 表达式 < EVERY **b1**(BOY **b1**) > 常常可缩写为 < EVERY **b1** BOY >。

自然语言中很多结构都受辖域影响。具体来说, 所有的广义量词, 包括“the”都受辖域支配。例如, 在“At every hotel, the receptionist was friendly”中, 几乎在任何上下文中, 首选解读都会认为定指性指代“the receptionist”在“every hotel”的辖域内; 也就是说, 每个宾馆都有各自的接待员。

另外, 诸如否定和时态这样的运算符也受辖域影响。例如, 句子“Every boy didn't run”是有歧义的, 因为它有两种解读。一种是有些男孩不跑而有些跑, 即:

(NOT(EVERY **b1**:(BOY1 **b1**)(RUN1 **b1**)))

而另一种解读是所有男孩都不跑, 即:

(EVERY **b1**:(BOY1 **b1**)(NOT(RUN1 **b1**)))

这两种解读可以用一个逻辑形式表示:

(< NOT RUN1 > < EVERY **b1** BOY1 >)

其中, 不在辖域内的一元运算符(比如, NOT, PAST, PRES, 等等)和谓词放在一起。

最后,让我们再来看看两个需要更深入研究的结构,专有名词和代词。到现在为止,我们都假定每个专有名词只有一个意义,这个意义表示论域内的某个对象。虽然在介绍逻辑形式的基本概念时,这种抽象方法很有用,但它并不能作为这种现象的一般化处理方法。事实上,专有名词必须根据上下文进行解释:“John”这个名字在不同情境中可能指向不同的人。我们借助篇章变量对上述方法进行修正以解决这类问题。这个变量有一个属性,而这个属性有特定的名称。我们用一个特殊的函数来引入这种结构,即:

(NAME <变量> <名称>)

它产生一个特定对象,该对象在当前的上下文中有这个名字。因此,“John ran”(John 跑了)的逻辑形式就是[<PAST RUN1> (NAME j1 “John”)]。

前面对专有名词的说明也适用于代词和其他指示词,比如“here”和“yesterday”,我们可以用一个特殊的表达式函数(PRO <变量> <命题>)来进行处理。例如,“Every man liked him”(每个人都喜欢他)的准逻辑形式是:

(<PAST LIKE1> <EVERY m1 MAN1> (PRO m2(HE1 m2)))

HE1 是“he”和“him”的语义,而且形式上也是谓词,只要对象满足前述的约束即可。在这种情况下只要是男的,它就为真。对广义量词,当约束条件是简单一元谓词时,pro 形式常常会缩写。例如,“he”的逻辑形式常写成(PRO m2 HE1)。

对给定的句子,本章描述的这些结构大大减少了初步计算所得的逻辑形式的数量。但缩写形式并不能刻画所有的歧义。因此,即便是单一的句法结构,仍有一些句子需要把所有可能的逻辑形式列举出来。

8.5 动词与逻辑形式中的状态

到目前为止,动词都映射到对应的意义,这些意义在逻辑形式中充当谓词。该方法能处理各种不同的形式,却失去了一些普遍性。此外,它还有一些很难处理的性质。

分析下面的句子,它们都用到动词“break”:

John broke the window with the hammer. (John 用锤子打破了窗户。)

The hammer broke the window. (锤子打破了窗户。)

The window broke. (窗户破了。)

直观上,这几句话说的是同一件事,只是细节上有所区别。因此,只要把每个例子中的动词 break 映射到同一个意义上就很完满了。但这样有一个问题,这个动词的三种用法似乎说明了动词与数相关的几种不同意义。第一个表明约翰、窗户和锤子三者之间的关系,第二个则是窗户和锤子之间的二元关系,第三个则是只和窗户有关的一元关系。看来,需要提供 break 的三个不同的意义 BREAK1, BREAK2 和 BREAK3 来表示它们的数之间的不同,并且生成如下的逻辑形式:

1. (<PAST BREAK1> (NAME j1 “John”) <THE w1 WINDOW1>
<THE h1 HAMMER1>)
2. (<PAST BREAK2> <THE h1 HAMMER1> <THE w1 WINDOW1>), 与

3. ($\langle \text{PAST BREAK3} \rangle \langle \text{THE w1 WINDOW1} \rangle$)

而且,要保证每个谓词都能正确地解释各自的含义。这些表达式需要一些定理来保证只要 1 为真,2 就为真,且只要 2 为真,3 就为真。这些定理常称为意义公设(meaning postulate)。但为每个动词都指定这样的约束条件相当困难。

Davidson(1967)提出一种新的表示方法来处理这种情况,以及更复杂的副词修饰形式。他建议,在本体中引入事件,按下述方法来处理像“John broke it”这种句子的意思(已转换成我们的表示方法):

$$(\exists e_1: (\text{BREAK } e_1 (\text{NAME } j_1 \text{ "John"}) (\text{PRO } i_1 \text{ IT1})))$$

这是一个断言,即 e_1 代表约翰打破了某扇窗户这个事件。现在,“John broke it with the hammer”的意思可以表示成:

$$(\exists e_1: (\& (\text{BREAK } e_1 (\text{NAME } j_1 \text{ "John"}) (\text{PRO } i_1 \text{ IT1})) (\text{INSTR } e_1 \langle \text{THE h1 HAMMER} \rangle)))$$

其优点在于,可以通过增加更多和事件有关的谓词把新的修饰语(比如“with the hammer”,“on Tuesday”或“in the hallway”)不断加入到基本表达式中去。因此,只要定义动词“break”的一种意义就可以处理这几种情况。

20 世纪 70 年代早期,类似的想法引发了对格语法的研究。虽然一开始提出的格语法已被大家抛弃,但它的很多概念仍在影响现在的理论。其中一个仍有影响的提法是动词及其参数之间存在一组抽象的语义关系。这些关系一般称为论旨角色(thematic role)或者格角色(case role)。虽然不同的研究者有各自的角色集,但集合所需元素的数目都很少。直观上看,“John”,“the hammer”和“the window”在每句话中扮演同样的语义角色。“John”是行为者(主体角色),“the window”是对象(论题角色),而“the hammer”则是“砸”这个动作中所用的工具(工具对象)。实际上,这提出了一种与语义网络所用的表示法相似的句子意义的表达式,语义网络中的所有对象都用一元或二元关系的词表示。具体来说,用刚才提到的三种语义网络,“John broke the window”的意义可以表示为:

$$(\exists e (\& (\text{BREAK } e) (\text{AGENT } e (\text{NAME } j_1 \text{ "John"})) (\text{THEME } e \langle \text{THE w1 WINDOW} \rangle)))$$

由于这种结构很常见,我们专门引入一个新的符号来表示它。缩写形式如下的断言:

$$(\exists e: (\& (\text{Event-p } e) (\text{Relation}_1 e \text{ obj}_1) \dots (\text{Relation}_n e \text{ obj}_n)))$$

可以表示为:

$$(\text{Event-p } e [\text{Relation}_1 \text{ obj}_1] \dots [\text{Relation}_n \text{ obj}_n])$$

举例说明,句子“John broke the window”的准逻辑形式用这种缩写方法表示为:

$$(\langle \text{PAST BREAK1} \rangle e_1 [\text{AGENT}(\text{NAME } j_1 \text{ "John"})] [\text{THEME } \langle \text{THE w1 WINDOW1} \rangle])$$

可以想见,除了事件动词之外,其他动词也有类似的参数。分析句子“Mary was unhappy”(Mary 不高兴了)。它用一元谓词可以表示为:

(< PAST UNHAPPY > (NAME j1 “Mary”))

那么,如何处理像“Mary was unhappy in the meeting”(Mary 在会议中不高兴了)这种句子中的修饰成分呢? 根据前面的讨论,可以把事件概念加以推广,使状态也包含在内,然后应用同样的方法处理。例如,可以把 UNHAPPY 按谓词处理,断言其参数处于不高兴的状态。用 THEME 角色把 John 包含进来,可以写成:

(∃ s < PAST UNHAPPY > s)(THEME s(NAME j1 “Mary”))

当然,也可以用和前面定义事件同样的缩写方法,将“Mary was unhappy in the meeting”表示为:

(< PAST UNHAPPY > s [THEME(NAME j1 “Mary”)]
[IN-LOC < THE m1 MEETING >])

有人认为,把事件变量作为基本谓词的参数仍不足以刻画自然语言要表达的意义。比如, Hwang 和 Schubert(1993a)允许事件由任意句子定义。但目前,有了事件和状态的概念,就足以阐明本书中的重要思想。

很多情况下,在公式中明确使用事件和状态变量非常麻烦,并且会干扰对其他方法的阐述。因此,我们选用那些最适于表示的方法来进行表示。例如,“Mary sees John”的逻辑形式有时写成:

(PRES(SEES1 i1 [AGENT(NAME j1 “Mary”)]
[THEME(NAME m1 “John”)]))

当然,它和下式等价:

(PRES(∃ i1(& (SEES1 i1)(AGENT i1(NAME j1 “Mary”))
(THEME i1 (NAME m1 “John”)))))

有时,也写成谓词参数形式:

(PRES(SEES1(NAME j1 “Mary”)(NAME m1 “John”)))

很多著作都在争论是否有必要进行论旨角色分析,或者在谓词-参数类型的表达式中是否也能用其他方法得到想从论旨角色中得到的属性。虽然很难说论旨角色必不可少,但这种表示方式在很多语义表示语言中确实非常有用。

8.6 论旨角色

本节探讨基于论旨角色(或格角色)标记体系的理论。上一节中引出该理论的例子包括三个句子:

John broke the window with the hammer.(John 用锤子打破了窗户。)

The hammer broke the window.(锤子打破了窗户。)

The window broke.(窗户破了。)

三个句子中,“John”,“the hammer”和“the window”扮演相同的语义角色。在打破窗户这个动作中,“John”是行为者,“the window”是对象,“the hammer”是工具。我们引入 AGENT, THEME 和

INSTR 这些关系来表示这些直观的意义。但是否可以更精确定义这些关系呢?在自然语言体系中还有哪些有用的论旨角色呢?本节将一一探讨这些问题。

也许最容易定义的论旨角色就是 AGENT 角色。在句子中,描述行为发出者的名词短语就扮演 AGENT 角色。进一步说,这个角色把对该行为的目的、意愿或后果归于所述的行为主体。我们可以对 AGENT 关系做一个测试,即向主动语态的句子中增加类似于“intentionally”或“in order to”这样的短语。若产生的句子结构规范,则主语 NP(名词短语)就充当主体角色。下面的句子是合理的。

John intentionally broke the window. (John 故意砸破了窗户。)

John broke the window in order to let in some air. (为了透气,John 砸破了窗户。)

而下面的句子不合理:

* The hammer intentionally broke the window.

* The window broke in order to let in some air.

因此,只有前面两句中的“John”是 AGENT 角色。

并非所有有生命的 NP 都是 AGENT 角色,即使做主语的时候也是这样。例如,一般情况下不能说:

* John intentionally died.

* Mary remembered her birthday in order to get some presents.

当然,向第一个句子中添加短语“intentionally”,会使句子产生模棱两可的含义(“John killed himself”),而这扭曲了句子“John died”(John 死了)的原义。

描述正在发生的某些变化或者充当行为对象的 NP 则扮演 THEME 角色。它通常对应于句法的宾语(OBJECT),而且对任意的及物动词 X,它是问题“what is Xed”的答案。例如,句子“The gray eagle saw the mouse”(苍鹰看见了老鼠),名词短语“the mouse”是 THEME,同时也是问题“what was seen?”的答案。对不及物动词而言,用做主语但非主体角色的名词短语是 THEME 角色。因此,句子“The clouds appeared over the horizon”(云雾出现在地平面上)中,名词短语“the clouds”充当 THEME 角色。下面这些例子中斜体的 NP 就是 THEME 名词短语:

The rock broke. (岩石碎了。)

John broke *the rock*. (John 打碎了岩石。)

I gave John *the book*. (我把书给了 John。)

有一些角色与位置或抽象位置有关。首先,必须区分前面提到的表示位置或场所的关系与表示运动或路径的关系。AT-LOC 关系表示某个对象在哪里或某个事件在哪里发生。比如,在下面的例子中:

Harry walked *on the road*. (Harry 在路上散步。)

The chair is *by the door*. (椅子在门边。)

“On the road”表示步行发生在何地,而“by the door”说明椅子所放的位置。

其他一些短语描述位置、运动的方向或路线的改变:

I walked *from here to school* yesterday. (昨天我从这里走到学校。)

It fell *to the ground*. (它坠向地面。)

The birds flew *from the lake along the river gorge*. (鸟从这个湖沿着河谷飞去。)

这里,至少有三种不同类型的短语,包括描述事物从哪里来的(FROM-LOC 角色)短语,比如“from here”;描述去向(TO-LOC 角色)的短语,如“to the ground”;描述轨迹或路线的(PATH-LOC 角色)短语,如“along the gorge”。

这些位置角色可以推广到有任意状态值的角色(称为 AT 角色),以及有任意状态变化的角色(FROM, TO 和 PATH 角色)。因此,AT-LOC 是 AT 角色的一种特殊情况,依次类推。在分析抽象的占有关系时,能看到这些角色的其他例子:

I threw the ball <i>to John</i> .	(TO-LOC 角色)
I gave a book <i>to John</i> .	(TO-POSS 角色)
I caught the ball <i>from John</i> .	(FROM-LOC 角色)
I borrowed a book <i>from John</i> .	(FROM-POSS 角色)
<i>The box</i> contains a ball.	(AT-LOC 角色)
<i>John</i> owns a book.	(AT-POSS 角色)

与此相似,可以定义 AT-TIME, TO-TIME 和 FROM-TIME 这些角色,比如下面的句子:

I saw the car at *3 o'clock*. (AT-TIME 角色)

I worked *from one until three*. (FROM-TIME 和 TO-TIME 角色)

对一般的状态改变,这些角色依然适用,比如下面几句话中提到的温度:

The temperature remains *at zero*. (AT-VALUE)

The temperature rose *from zero*. (FROM-VALUE)

因此,通过类似的方法在句子中将其角色化,可以表示出一般的值及其在多个维度上的变化。

根据目前的分类,有些句子中名词短语的角色还不易确定。为解决这个问题,需要引入一类角色,比如:

John believed that it was raining. (John 相信下雨了。)

从句“that it was raining”充当 THEME 角色,因为充当被相信的对象。“John”不可能是主体角色,因为相信某事并无主观意图可言。因此,必须引入一种新的角色,称为体验者(EXPERIENCER)。这个角色由有生命的对象担当,该对象正处于所描述的某个心理状态或正经历某种心理过程,比如感觉。如前面及下面的句子所示:

John saw the unicorn. (John 见过白鲸。)

还有一种角色叫 BENEFICIARY 角色,由特定事件的受益人充当,比如:

I rolled on the floor *for Lucy*. (为了 Lucy 我在地上打滚。)

Find *me* the papers! (给我找到那些报纸!)

I gave the book to Jack *for Susan*. (我替 Susan 把书交给 Jack。)

上面最后一个例子证明有必要清楚区分 TO-POSS 角色(即 to Jack)和 BENEFICIARY 角色。

INSTR 角色描述的是某事件中所用的工具、材料或力,如:

Harry broke the glass *with the telescope*. (Harry 用望远镜打碎了玻璃。)

The telescope broke the glass. (望远镜打碎了玻璃。)

I used *some flour* to make a cake. (我用面粉做了一个蛋糕。)

I made a cake *with some flour*. (我做了一个用面粉做的蛋糕。)

对于某些动词,当主体角色未指定时,工具角色有时会用做形式主语。自然力在这里也归入工具类,虽然你的分析可能与此不同。下面同样是工具角色的例子:

The sun dried the apples. (太阳晒干了苹果。)

Jack used *the sun* to dry the apples. (Jack 借助阳光晒干了苹果。)

AGENT 角色和 INSTR 角色可以归入一个更宽泛的角色,称为因果主体(CAUSAL-AGENT)角色。

如果要分析某些句子,还需要引进其他角色。比如,一些句子描述了两个人共同完成一个动作的情境:

Henry lifted the piano *with Jack*. (Henry 和 Jack 把钢琴抬了起来。)

要处理这个句子,必须引入协同主体(CO-AGENT)角色以表示介词短语“with Jack”。

还有一些更复杂的情况出现在涉及交换或其他复杂交互的句子中。举例来说,考察下面的句子:

Jack paid \$1 to the man for the book. (Jack 付了 1 美元给那个人买下这本书。)

Jack bought the book from the man for \$1. (Jack 用 1 美元从那个人手里买下了这本书。)

这两个句子都描述了这样的情境,John 给那个人 1 美元,同时作为交换得到一本书。但是,第一个句子里,1 美元是 THEME 角色,而“the book”则没有对应的角色。第二个句子里情况相反,“the book”是 THEME,而 1 美元则没有对应的角色。要处理这种情况,必须加入 CO-THEME 角色以表示交换中的第二个对象。

这个问题更一般化的解决方法是将这句话按两个事件来分析。主要事件就是目前你所考察的那个事件,而次要事件是当前发生的事件。在这种分析方法中,可以这样分析第一个句子(主要事件是 Jack 付钱,次要事件是 Jack 得到书):

Jack: 主要事件与次要事件中均为 AGENT 角色

\$1: 主要事件中的 THEME

the man: 主要事件中的 TO-POSS 角色,次要事件中的 FROM-POSS 角色

the book: 次要事件中的 THEME 角色

我们对这种可能的情况不做更深入的分析,因为涉及很多与本节剩下部分无关的内容。

图 8.5 列出了到目前为止出现的大部分角色及其层次关系。

正如我们前面所看到的,动词可以按其论旨角色来分类。但要对其准确分类,必须区分与该动词有“亲密”关系的角色以及没有“亲密”关系的角色。例如,几乎所有的过去时态动词都允许副词“yesterday”充当 AT-TIME 角色。因此,与任何一个动词的性质比较而言,这个角色显然是动词短语的一个更普遍的性质。但其他角色(即,那些把动词细分为次范畴的句子成分所充当的角色)好像也是动词本身的性质。例如,动词“put”根据介词短语(PP)可进行细分,而这

个 PP 必定充当 TO-LOC 角色。在动词的次范畴划分中,后一种角色十分重要,称为动词的内在角色。

角色与子角色	其他常用名	定义
CAUSAL-AGENT		引发事件的对象
AGENT		主动引发
INSTR		在引发事件中所用的力/工具
THEME	PATIENT	被事件影响到的事物
EXPERIENCER		参与感知或处于某种生理/精神状态中的人
BENEFICIARY		该动作指向的人
AT		某个维上的状态/值
AT-LOC	LOCATION	当前位置
AT-POSS	POSSESSOR	当前所有者
AT-VALUE		当前值
AT-TIME		当前时间
TO		状态变化的终值
TO-LOC	DESTINATION	最后的位置
TO-POSS	RECIPIENT	最后的所有者
TO-VALUE		最后的值
FROM		状态变化的初始值
FROM-LOC	SOURCE	初始位置
FROM-POSS		初始的所有者
FROM-VALUE		初始值
PATH		移动的路线
CO-AGENT		动作的协同主体
CO-THEME		交换中的协同论题

图 8.5 一些可能的语义角色

前面的例子给出了一种方法,用于判断给定动词的已知角色是不是其内在角色:如果该角色必不可少,就是内在角色。但有些内在角色似乎可有可无,因此,还需要寻找别的方法。观察发现,所有动词后跟任何已知的内在角色时,最多只能接一个 NP。由此,可以设计另一种测试方法。若要求接多个 NP,则必须通过连词将它们连接起来。因此,可以说:

John and I ran to the store.(我和 John 跑向商店。)

但不能说:

* John I ran to the store.

与此相似,可以说:

I ran to the store and to the bank.(我跑向商店和银行。)

但不能说:

* I ran to the store to the bank.

因此,动词“run”的 AGENT 角色和 TO-LOC 角色都是内在角色。

动词通常最多有三个内在角色,其中至少有一个必定会在使用该动词的句子中得以体现。

有时,某个特定角色必定会出现(比如“put”的 TO-LOC 角色)。通常,对任何有被动形式的动词而言,THEME 角色是必定要有的,而 AGENT 角色往往是可选的。

各种角色的实现方式均存在句法约束。图 8.6 给出了在不同句子里如何实例化角色的示例。

角色	实现
AGENT	做主动句的主语,做被动句中 介词 by 的引导
THEME	做及物动词的宾语,做不及物动词的主语
INSTR	做无主体的主动句的主语,介词 with 引导
EXPERIENCER	在无主体的主动句中做主语,由有生命的物体充当
BENEFICIARY	做及物动词的间接宾语,介词 for 引导
AT-LOC	介词 in, on, beyond 等引导
AT-POSS	所有格名词短语, 如果句子没有主体,做主语
TO-LOC	介词 to, into 引导
TO-POSS	介词 to 引导,做某些动词的间接宾语
FROM-LOC	介词 from, out of 等引导
FROM-POSS	介词 from 引导

图 8.6 主要角色的常见实现

下面例句中的动词及其参数(无论 NP, PP 还是内嵌句 S)以斜体字表示,各句按照角色进行分类,并按出现顺序介绍如下:

Jack <i>ran</i> .	只有 AGENT
Jack <i>ran</i> with a crutch.	AGENT + INSTR
Jack <i>ran</i> with a crutch for Susan.	AGENT + INSTR + BENEFICIARY
Jack <i>destroyed</i> the car.	AGENT + THEME
Jack <i>put</i> the car through the wall.	AGENT + THEME + PATH
Jack <i>sold</i> Henry the car.	AGENT + TO-POSS + THEME
Henry <i>pushed</i> the car from Jack's house to the junkyard.	AGENT + THEME + FROM-LOC + TO-LOC
Jack <i>is tall</i> .	THEME
Henry <i>believes</i> that Jack is tall.	EXPERIENCER + THEME
Susan <i>owns</i> a car.	AT-POSS + THEME
I <i>am</i> in the closet.	THEME + AT-LOC
The ice <i>melted</i> .	THEME
Jack <i>enjoyed</i> the play.	EXPERIENCER + THEME
The ball <i>rolled</i> down the hill to the water.	THEME + PATH + TO-LOC

8.7 言语行为与内嵌句

句子各有其使用目的。每个句子的语气都表明说话人和言语表达内容之间不同的关系。这些问题会在后面的章节里进行更详细的阐述。目前,我们需要扩展逻辑形式语言以刻画这

些不同的关系。每一大类句子都有一个对应的运算符,该运算符以句子的解释作为参数,从而生成所谓的表层言语行为(surface speech act),用于描述如何以所述命题来更新会话情境。我们采用如下的新运算符来表示句子语气:

ASSERT——命题是一个断言

Y/N-QUERY——命题是一个查询

COMMAND——命题描述一个要执行的动作

WH-QUERY——命题描述一个要确认的对象

对于陈述句,如“The man ate a peach”(这个人吃了一个桃子),完整的逻辑形式(LF)是:

(ASSERT(< PAST EAT > e1 [AGENT < THE m1 MAN1 >]
[THEME < A p1 PEACH1 >]))

对一般疑问句,如“Did the man eat a peach?”(这个人吃了个桃子吗?)LF 是:

(Y/N-QUERY(< PAST EAT > e1 [AGENT < THE m1 MAN1 >]
[THEME < A p1 PEACH1 >]))

对命令句,例如“Eat the peach”(吃桃子),LF 是:

(COMMAND(EAT e1 [THEME < THE p1 PEACH1 >]))

对于特殊疑问句,还需要对逻辑形式语言做几项补充,比如“What did the man eat?”(这个人吃了什么?)首先,需要一种方法来表示含有特殊疑问词的名词短语的含义。这里,定义一个新的量词 WH,该词代表问题中的一个或多个对象。因此,像 what 这样的名词短语可以表示成 < WH o1 ANYTHING > ,“which man”可以表示成 < WH m1 MAN1 > ,而“who”可以表示成 < WH p1 PERSON > 。最后,对于“how many”和“how much”这种疑问形式,我们引入量词 HOW-MANY 和 HOW-MUCH。请注意,特殊疑问词受辖域影响,合理的做法是将它们按量词来处理。疑问句“Who is the leader of every group?”(每组的领导者是谁?)有歧义,因为它既有可能问的是所有团体共同的领导人,也可能问的是每个团体各自的领导人。

因此,句子“What did the man eat?”的逻辑形式表示为:

(WH-QUERY(< PAST EAT > e1 [AGENT < THE m1 MAN1 >]
[THEME < WH w1 PHYSOBJ >]))

内嵌句(如关系从句)这一复杂约束已经存在于名词短语结构之内了,因而不需要任何新的标记来表示。例如,句子“The man who ate a peach left”(吃桃子的那个人离开了)的逻辑形式是:

(ASSERT
(< PAST LEAVE > i1
[AGENT < THE m1(& (MAN1 m1)
(< PAST EAT1 > e2
[AGENT m1]
[THEME < A p1 PEACH >])) >]))

图 8.7 给出了贯穿于本章的逻辑形式语言语法的形式定义。图 8.8 给出了定义准逻辑形式语言的语法所需的附加规则。

```

UTTERANCE → (ASSERT PROPOSITION) |
              (Y/N-QUERY PROPOSITION) |
              (COMMAND PROPOSITION) |
              (WH-QUERY PROPOSITION)
PROPOSITION → (n-ARY-OPERATOR PROPOSITION1 ... PROPOSITIONn) |
              (QUANTIFIER VARIABLE : PROPOSITION PROPOSITION) |
              (n-ARY-PREDICATE TERM1 ... TERMn) |
              (EVENT-STATE-PRED VARIABLE [ROLE-NAME TERM]1 ...
              [ROLE-NAME TERM]n)
TERM → VARIABLE |
      (NAME VARIABLE NAME-STRING) |
      (PRO VARIABLE PROPOSITION)
1-ARY-OPERATOR → NOT | PAST | PERF | PROG | ...
2-ARY-OPERATOR → AND | BUT | IF-THEN | ...
QUANTIFIER → THE | SOME | WH | ∃ | ..
VARIABLE → b1 | man3 | ...
1-ARY-PREDICATE → TYPE-PREDICATE | HAPPY1 | RED1 | ...
TYPE-PREDICATE → EVENT-STATE-PRED | (PLUR TYPE-PREDICATE) | MAN1 | ...
EVENT-STATE-PRED → RUN1 | LOVE3 | GIVE1 | HAPPY | ...
2-ARY-PREDICATE → ROLE-NAME | ABOVE1 | ...
ROLE-NAME → AGENT | THEME | AT-LOC | INSTR | ...
NAME-STRING → "John" | "The New York Times" | ...

```

图 8.7 逻辑形式语言语法的形式定义

```

TERM → <QUANTIFIER VARIABLE PROPOSITION>
TERM → <n-ARY-OPERATOR TERM1 ... TERMn>
n-ARY-PREDICATE
    → <m-ARY-OPERATOR n-ARY-PREDICATE1 ... n-ARY-PREDICATEm>
n-ARY-OPERATOR → {n-ARY-OPERATOR1 ... n-ARY-OPERATORm}
QUANTIFIER → {QUANTIFIER1 ... QUANTIFIERm}
n-ARY-PREDICATE → {n-ARY-PREDICATE1 ... n-ARY-PREDICATEm}
TYPE-PREDICATE → {TYPE-PREDICATE1 ... TYPE-PREDICATEm}
EVENT-STATE-PRED → {EVENT-STATE-PRED1 ... EVENT-STATE-PREDm}
ROLE-NAME → {ROLE-NAME1 ... ROLE-NAMEm}

```

图 8.8 定义准逻辑形式所需的附加规则

8.8 定义语义结构:模型论

到目前为止,语义这个词都用来反映句子意义的表示,并且是通过逻辑形式语言来表示

的。在形式语言理论里,语义还有另外一种含义,即逻辑形式语言自身的语义。其主要目标是通过语义对象的模型论性质对它们的类加以区分。也就是说,用语义单元到集合论的映射来定义语义单元。虽然这些技术在逻辑学中应用得比较广泛,但同样可以应用于自然语言。本节是可选章节,对理解后续内容并非必不可少。如果你对一阶谓词演算及其模型论语义学不太熟悉,请在学习本节之前阅读附录 B。

定义语义性质的基础是模型的思想(简单地说,模型就是一组对象、它们的性质和它们之间关系的集合)以及如何将所研究的语言 and 对象与对象间的关系关联起来的方法。模型可以看做是处理句子的特定上下文表示。可以对模型必须具备的性质增加种种约束。比如标准的逻辑模型,即 Tarskian 模型就是完备的,因为它把语言中所有合法的词都映射到论域中,并对每个命题赋以真值或假值。但对各种不完备的模型而言,可能并非所有命题都为真或都为假。这些模型类似于上一节所描述的情境。实际上,情境的数学理论就是可以在这里使用的形式模型。后面再遇到与此相关的问题时,我们再对其做深入探讨。

对上下文无关意义的研究而言,模型论是一个极好的方法,因为句子的含义并非由某一个特定模型来定义,而更可能是用它和任何可用模型关联的方式来定义的。换句话说,句子的含义是由句子在模型中所具有的性质来定义的。

用形式化的语言来讲,模型 m 是一个二元组 $\langle D_m, I_m \rangle$ 。其中, D_m 是解释域,即原始对象集合,而 I 则是解释函数。要处理自然语言,解释域必须包含用来指代各种事物的对象,包括实体对象、时间、场所、事件和情境。解释函数将词义和更大的结构映射到在该论域中定义的结构。举例来说,下面几段话描述了解释函数基于一些词类解释意义的方法:

名词短语的意义——指向特定对象;解释函数将名词短语的意义映射为 D_m 的一个元素。

普通单数名词(如“dog”, “idea”, “party”)的意义——确定论域中的对象类别;解释函数将其映射为 D_m 的元素的集合(即 D_m 的子集)。

动词的意义——确定 D 内对象的 n 元关系集。变元(arity)的多少依动词而定。举例来说,“run”的锻炼的意义 RUN1,就映射为一个一元关系的集合($\langle X \rangle$, 其中满足 X runs);而“loves”通常的意义(LOVES1)映射为一个二元关系的集合($\langle X, Y \rangle$, 满足 X loves Y);“put”常用的意义(PUT1)映射为一个三元关系的集合($\langle X, Y, L \rangle$, 满足 X puts Y in location L)。

现在,我们可以在逻辑形式语言中定义命题的真假值概念了,这也和任意确定的模型 m 相关。对于模型 m ,一个形式为 $(V_n, a_1 \dots a_n)$ 的命题为真,当且仅当由所有 a_i 的解释组成的 n 元组属于 V_n 的解释集。也就是说,元组 $\langle I_m(a_1), \dots, I_m(a_n) \rangle$ 属于集合 $I_m(V_n)$ 。按照习惯记法,把模型 m 下命题 P 为真记做:

$$m \models P$$

有时,这也读做“ m 支持 P ”。举例来说, m 支持(RUN1 JACK1),仅当 $I_m(JACK1)$ 属于集合 $I_m(RUN1)$;而 m 支持(LOVES1 JACK1 SUE1),仅当 $\langle I_m(JACK1), I_m(SUE1) \rangle$ 属于集合 $I_m(LOVES1)$ 。

否定语义取决于所用模型的性质。在标准 Tarskian 的语义中(此模型是完备的),否定的定义是不存在使命题为真的必要关系。也就是说,模型 m 支持(NOT P)当且仅当 m 不支持 P 。在其他模型中,这个定义可能过强,因为一个命题也可能既非真又非假。这种模型必须为每个

关系的名称维护两个不相交的集合,即关系为真的元组和关系为假的元组,而不在这二者之内的序列既非真又非假。

一阶谓词演算中量词的语义也相当简单。例如,命题 $\forall x. P(x)$ 在模型 m 中为真当且仅当对 D_m 里 x 的所有值, $P(x)$ 均为真。另一方面,命题 $\exists x. P(x)$ 在模型 m 中为真,当且仅当 D_m 里 x 至少有一个值使 $P(x)$ 为真。而对于自然语言,有广义量词与之对应。每个量词的真值条件指定了满足这两个命题的对象之间所需的关系。举例来说,考察命题 $(\text{MOST1 } x(P(x))(Q(x)))$ 。该命题在模型 m 中可为真,当且仅当 $I_m(P)$ 中超过一半的对象在 $I_m(Q)$ 里。例如, $(\text{MOST1 } x(\text{DOG1 } x)(\text{BARKS1 } x))$ 在模型 m 中为真,当且仅当集合 $\{x | (\text{DOG1 } x)\}$ 里超过半数的元素同时也在集合 $\{y | (\text{BARKS1 } y)\}$ 里,即仅当 m 里有超过半数的狗在叫。

8.8.1 句子间的语义关系

有了语义理论,就可以更准确地推断句子间的特定关系了。比如,如果知道某个句子 S 为真,那另外一些句子也必然为真。比如,你知道“A red box is on the table”(红盒子在桌上),那你同时也知道“A box is on the table”(盒子在桌上)。这两个句子间的关系称为蕴涵。我们说句子“A red box is on the table”蕴涵了句子“A box is on the table”。蕴涵可以由表达这些句子的模型来进行形式化定义。具体而言,句子 S 蕴涵了句子 S' 当且仅当每个能表达 S 的模型都表达 S' ,即:

S 蕴涵 S' , 当且仅当对任意的模型 m , 若 $m \models S$, 则 $m \models S'$

相反,若不存在同时支持两个句子的模型,则称这两个句子矛盾。即,不存在任何情境,使两个句子同时为真。稍微修改一下前面的例子,我们知道对于句子“A red box is on the table”而言,句子“*There is no box on the table*”是矛盾的,因为没有一个模型能同时支持这两个句子。

因为蕴涵要在所有的模型中都成立,所以能找到的蕴涵关系似乎不会很多。但事实并非如此,因为模型间共有的性质比我们根据第一印象认定的数量多得多。具体来说,每一种模型都拥有语言中所有的上下文无关性质。思考上面的例子。尽管各个模型中盒子的集合与红色物体的集合可以千差万别,但上述蕴涵关系并不依赖于所指的实际物体。事实上,它仅依赖于对“red”和“box”的含义的一般性解释、两个集合 X 和 Y 的集合论属性以及 X 中的 X 和 Y 的交集部分(就此而言,也包含在 Y 中)。因此,只要模型将名词短语“A red box”解释为从集合 $RED1$ 中与集合 $BOX1$ 的交集中选择的一个对象,则这个对象必然也在集合 $BOX1$ 中。这就是所有的必要条件。因此,每个模型都会包含该蕴涵关系。进一步说,8.2 节里曾提出,词义可以基于集合的包含关系组织成一个层次关系。这就在语义上定义了一个附加的上下文无关结构,因此会出现在所有的模型中。由此,因为每个模型都会将 $MARE1$ 映射为 $HORSE1$ 的一个子集,所以句子“A mare ran away”(母马跑了)就蕴涵了句子“A horse ran away”(马跑了)。

尽管蕴涵是一个很有用的性质,但在通常所理解的自然语言中,句子间的关系比蕴涵要弱,这种关系称为隐含(implication)。句子 S 隐含句子 S' , 如果 S 为真则表明 S' 也为真,或者说使 S' 可能为真。例如,我可以说“I used to walk to school everyday. In fact, I still do!”所以句子“I used to walk to school everyday”隐含我现在不走路去学校,但并不蕴涵后者。因为这两个句子并不违反常理,因此必须有一个模型使两者同时为真。而且,在这个模型中,“我再也不走路去学校”不为真。但在通常的上下文中,与该上下文相关的模型也可能完全支持“我再也不走路去学校”。

学校”。语言理解的一个难题源于以下事实,即推理大多源自隐含而非蕴涵。因此,某个时刻所做的结论以后可能需要重新分析乃至撤销。

8.8.2 情态运算符与可能的世界语义

为使情态运算符也有语义,还要给模型理论加入更多的结构。具体来说,性质的真假值由模型的集合共同定义,而并非由单独一个模型给出。除此之外,这个模型的集合可能还有一个复杂结构,称为模型结构。

分析过去时态运算符 PAST。假设每个模型 m 都表示时间轴上某个点的世界状态,那么,模型结构就是描述世界的一个可能历史模型的集合。在这个集合里,模型通过运算符“ $<$ ”互相联系,“ $m_1 < m_2$ ”表示 m_1 描述的世界状态在 m_2 之前。现在,一个命题的真假值由一个确定的模型结构 Ω 给出,其模型之间有“ $<$ ”关系。例如,参照模型结构 Ω 的模型 m ,可以定义表达式(PAST P)的语义:

$$m \models_{\Omega} (\text{PAST } P) \text{ 当且仅当 } \Omega \text{ 中存在其他的模型 } m', \text{ 使 } m' < m, \text{ 且 } m' \models_{\Omega} P$$

换句话说,(PAST P)在模型 m 中为真,当且仅当该模型结构里有另一个模型刻画了 m 之前的某一时间,而那时 P 为真。这种分析称为可能世界语义。

框 8.2 外延解读与内涵解读

本章中讨论的语义框架基本上指的是表达式的外延意义。也就是说,含义是由词汇在客观世界中代表的意义所定义的。例如,词义 DOG1 从它所指示对象的集合[即 $I_m(\text{DOG1})$]中获得意义。与之相似,诸如“the dog”或“John”这样的词通过它在论域里所指的对象获得含义。如果两个词指同一个对象,那么它们在语义上就没有区别。外延涵盖了语言中很多领域,但仍然不足以解释许多表达式。分析 Montague(1974)的一个经典例子:

The temperature is rising. (温度正在升高。)

句中的名词短语“the temperature”的含义是什么? 如果指示一个确定值,比如 30°C , 那么就不能再升高,因为句子“ 30°C is rising”没有意义。相反,短语“the temperature”一定是一个指向时间的函数。假如是函数 Temp, 那么给定一个时间和地点,就得到该温度。对于这个含义,谓词“is rising”对任何随时间推移而增加数值的函数都为真。因此,词“the temperature”似乎有两个不同的含义。外延是特定地点和时间上的实际值,比如“ 30°C ”;而内涵是一个时间和地点的函数,输入时间和地点得到一个数值。一些动词短语,如“is rising”,要求有内涵;而另外一些,如句子“The temperature is over 90°C ”,要求有外延,因为说一个函数的属性高于 90°C 是没有意义的。

因为内涵处理起来非常复杂,所以在本书中只涉及对表达式外延解读的处理。

8.9 小结

本章提出了一种上下文无关语义表示体系,称为逻辑形式。这种表示体系的目的在于简化由句法结构计算语义结构的过程。从更广泛的意义上来说,是将上下文的作用分离出来,从而使句子的处理模块化。逻辑形式语言使用了很多一阶谓词演算的概念,包括词项、谓词、命题和逻辑运算符。它对基本一阶谓词演算的重要扩展包括:

- 广义量词,反映两个集合间的关系,对应于“each”,“every”,“some”,“most”,“several”等词汇。
- 情态运算符,判定待考察命题的各种不同语态,对应于像“believe”和“hopes”这样的词,以及时态运算符。
- 谓词运算符,将一个谓词映射为一个新的谓词。比如,用一个复数形式运算符,就能将描述拥有性质 P 的单个个体的谓词,变为描述个体集合的谓词,且该集合中每个个体都有性质 P。

除此之外,逻辑形式语言可以用一种有效的方式表示很多常见的歧义形式。即,在任何只允许一个含义出现的地方列举多个含义以供选择。

逻辑形式语言的一个重要特点是它对谓词使用了事件和状态变量。这样,不必为该动词论元的每种组合形式都引入不同的谓词,就能表示额外的副词修饰语。我们还介绍了一种基于论旨角色的表示方法。尽管各个系统中的角色不尽相同,但这种表示方法在自然语言系统里十分普遍。

8.10 相关工作与深入阅读材料

这里阐述的逻辑形式语言参考了很多资料。早期影响最大的是 LUNAR 系统(Woods, 1978)的表达式语言,其中很多内容是逻辑形式语言中许多细节的思想渊源。近期影响较大的包括 Schubert 和 Pelletier(1982),Hwang 和 Schubert(1993a),以及 Moore(1981)提出的逻辑形式语言,还包括核心语言引擎(Alshaw, 1992)中的准逻辑形式。

和很多逻辑形式语言一样,这里提出的语言高度依赖于某种技术,即为处理可选参数及其他动词修饰语而对本体引入事件。这种技术在 Davidson(1967)那篇里程碑式的论文发表之后产生了很大影响,Davidson 对这种方法进行了很多论证。Moore(1981),Alshaw(1992)和 Hobbs 等(1987)提供了这种基于事件系统的范例。

对论旨角色的研究源自 Fillmore(1968)介绍的对格语法所做的研究工作,其中引入了 6 种格。这些思想随后经语言学和哲学的许多学者加以调整[例如 Jackendoff(1972),Dowty(1989),Parsons(1990)]。类似的技术可以在许多计算系统里找到。在这些系统里,论旨角色常常与底层的知识表示联系在一起(例如 Charniak, 1981)。

用逻辑形式对辖域歧义进行表示的思想已经在很多系统中得到应用。比如, Woods(1978)和 Cooper(1983)提出逻辑形式可以表示为两部分,一部分是谓词-参数结构,另一部分是量词信息。Schubert 和 Pelletier(1982),McCord(1986),Hobbs 和 Shieber(1987)以及近期的很多系统[如核心语言引擎(Core Language Engine, Alshaw, 1992)]都用特殊句法对辖域形式进行表示。这种方式不仅对歧义的表达很重要,而且在组合语义解释(compositional semantic interpretation)理论的定义里也扮演了关键角色,这一点将在第 9 章进行讨论。

语言学中关于语义的一些很好的综合性知识可以在 Chierchia 和 McConnell-Ginet (1990) 里面找到。这本书讨论了语义的本质, 以及像歧义与模糊性、指示性、符号语义、广义量词、内涵及其他很多问题。McCawley (1993) 也是非常好的资料, 其中介绍了有关语义的逻辑基础知识。关于语义表示的数学基础, 在 Partee 等 (1993) 中可以找到更详细的描述。Barwise 和 Cooper (1981) 讨论了广义量词。本章许多有关语义的想法都是首先在符号逻辑里发展起来的, 也有很多优秀文章 (比如 Thomason, 1970, Barwise 和 Etchemendy, 1987) 对其进行了论述。自然语言的语义当前所用的形式受 Montague (1974) 研究的影响很大。

当前, 形式语义的很多工作都是在情境语义的框架下进行的。本章讨论的情境的概念就源于这项研究。情境语义是 Jon Barwise 与其同事提出来的, 该理论的最初形式在 Barwise 和 Perry (1983) 中做了描述。Devlin (1991) 中可以找到非常好的有关情境的讨论及形式化描述。

8.11 习题

1. 【易】说明下列句子是否有歧义并阐述理由。尤其要说明它们是否会因多个可能的句法结构、词义、语义结构或者这些因素的共同作用而产生歧义。给出每种含义的描述。

A man stopped at every truck stop.

Several people ate the pizza.

We saw her duck.

2. 【中】对以下每个词, 讨论是否有歧义或模糊性。请至少给出一个语言学测试来论证你的答案。讨论你在解答过程中遇到的任何难题。

face—as a noun, as in what's on your head, the front of a clock, and the side of a mountain

bird—as a noun, as in animals that fly (such as a sparrow) and china figures of such animals

record—as a noun, as in where a school keeps your grades or where the FBI writes down your movements

3. 【易】判断下面句子中“can”这个词的含义, 确定该含义是否是词项、属性(一元谓词)、n元谓词、逻辑运算符、广义量词、谓词运算符或者情态运算符。论证你所做的分类, 讨论其他可行的解释。对每种含义给出一个逻辑形式的例子。

The yellow can fell to the ground.

He can see it.

He wants to can the tomatoes.

4. 【易】扩展以下有歧义的逻辑形式, 列举出歧义形式所产生的全部逻辑形式。

({ RUN1 RUN2 } [AGENT (PRO h1 HE1)])

(SEES1 s1 [AGENT (PRO h1 HE1)]

[< EVERY b1 { BALL1 BALL2 } >])

(GIVES1 I1 [AGENT < EVERY m1 MAN1 >]

[THEME < A g1 GIFT1 >])

(< NOT {RUN1 RUN2} >

[AGENT < EVERY m1(&(MAN1 m1)(HAPPY m1)) >])

5. 【中】确定下列句子的准逻辑形式。若句子有歧义,请用有歧义的逻辑形式或者用列举若干逻辑形式的方法,从而给出所有可能的表示形式。

George ate a pizza at every road stop.

Several employees from every company bought a pizza.

We saw John in the park by the beach.

6. 【中】确定下列句子中每个名词短语的角色。对每个句子给出其可能的逻辑形式。

We returned the ring to the store.

We returned to the party.

The owner received a ticket.

7. 【中】

- a. 给出令下列每个逻辑形式为真的形式语义模型。

(IN1 JOHN1 ROOM1)

(BOY1 JOHN1)

(EVERY b1:BOY1(EAT b1 PLZZA1))

尤其要定义出该模型中对象的集合,并给出把其中的每一个符号映射为该论域中的一个元素或元素集合的解释函数。通过确定其解释证明这些逻辑形式都为真。

- b. 你的模型是否也使下式为真?

(EAT JOHN1 PIZZA1)

能否建立一个相反的模型(若你的模型为真,则该模型为假,反之亦然)?如果可以,请给出该模型。如果不行,为什么?

8. 【中】对下列每个句子列表,说明其第一个句子是否蕴涵或隐含其后的句子,或者它们之间没有语义关系。用一些语言学测试证明你的回答。

- a. John didn't manage to find the key.

John didn't find the key.

John looked for the key.

The key is hard to find.

- b. John was disappointed that Fido was last in the dog show.

Fido was last in the dog show.

Fido was entered in the dog show.

John wanted Fido to win.

Fido is a stupid dog.

第9章 句法和语义的衔接

本章介绍一种特殊的方法,可以将逻辑形式和句法结构联系起来。借助这种方法,可以在句法分析的过程中计算出逻辑形式,这个过程称为语义解释。反过来,它的一种变形可以从特定的逻辑形式中生成句法树,这个过程称为语义实现。要让句法和语义完全耦合,其每个成分都必须有合乎语法的意义表达形式。成分的意义与其子成分的意义之间的关系在语法中可以用特征来确定。因为每个句法规则都有一个对应的语义解释规则,所以我们常常把这种方法看做语义解释的逐条引用形式。

9.1节解释组合性的概念,并引入 λ 演算作为构建组合理论的工具。9.2节和9.3节则探讨了语言中的一些基本结构,并针对英语中的一个子集提出了一种语法,这种语法可以在分析过程中计算其每个成分的逻辑形式。9.2节和9.3节所用的逻辑形式是一种谓词变量结构。9.4节阐述如何用语义角色生成逻辑形式,并简单说明要使用层次词典减少确定每个词条意义所需的工作量。9.5节论述了如何将语义解释和语义差异关联起来,并阐明了解决一些简单疑问句的方法。9.6节研究另一种用于计算逻辑形式的方法,这种方法使用附加特征而非 λ 表达式,这样就能表示出可逆语法。9.7节是选学部分,讨论的是语义实现,讲述如何根据已知的逻辑形式和一种可逆语法来生成一个句子。

9.1 语义解释与组合理论

在语义解释中,我们经常做的一个基本假设是将它看做一个组合过程。这表示,一个成分的意义是从其各个子成分的意义中推出来的。组合理论有一些很吸引人的性质。具体来说,解释可以从它的各个分解步骤中增量式地构造出来。例如,句法的上下文无关语法模型就是句法的组合理论。规则基于子成分类别来使用,而无须考虑其内部结构。例如,只要出现了NP,无论其形式如何,都可以应用规则 $S \rightarrow NP VP$ 。只要向语法中添加另一个NP规则,比如 $NP \rightarrow PRO$,就可以处理一类新的句子,即任何把代词放在可接受的NP位置上的句子。这在语义解释中,是我们非常需要一个特性。

在语言学中,组合理论(compositionality)常常定义成一个严格的规则,即某一个子成分的意义必须是一个函数,这个函数能够把其他所有子成分的意义映射成新产生的成分的意义。在计算方法中,这种要求常常较为宽松,只须实现一种思维过程,通过对成分的逐个分析把意义构造出来。这里,成分的意义是由某个明确定义的计算函数(也就是程序)生成的,该函数以各个子成分的意义作为输入。

组合理论模型使语法更易于扩展和维护。但语义解释组合理论的构建远不是第一眼看上去那么简单。首先,句法结构和逻辑形式结构之间存在结构上的不一致性。例如,我们都知道一个与量化句有关的经典问题。考虑“Jill loves every dog”(Jill爱每只狗)这句话。很明显,其句法结构将词聚成如下短语:((Jill)(loves(every dog)))。但以谓词变量形式表示这句话的无歧义逻辑形式,可得到如下形式:

(EVERY *d* : (DOG1 *d*) (LOVES1 *l1* (NAME *j1* "Jill") *d*))

根据上面的结构可以断言,对应于每条狗 *d*,都存在一个包含“Jill 爱 *d*”的事件 *l1*。逻辑形式的各个部分和句法分析的各个成分之间没有简单的一一对应关系。比如,短语“every dog”是 VP loves every dog 的子成分,而它的语义解释[广义量词短语(EVERY *d* : (DOG1 *d*) ...)]却包含了这个动词短语的意义。更糟糕的是,对“every dog”的解释好像分成了两部分,一部分解释是量词结构在谓词的作用范围之外,另一部分是作为谓词的参数。因此,很难在孤立的情况下表示“every dog”的意义,因而也就很难用它来表达句子的意义。这是在构建组合理论的过程中最难处理的问题之一。值得注意的是,非辖域逻辑形式结构的引入为解决此问题提供了一条途径。如果将语义解释的目标定义为生成一个非辖域逻辑形式,那么这个句子的非辖域写法就是:

(LOVES1 *l1* (NAME *j1* "Jill") <EVERY *d* DOG1>)

它在结构上和句法结构更接近。

组合理论的另一个难题是习惯用语的存在。例如,可以说“Jack kicked the bucket”,意思是杰克死了。这个解释好像既和动词“kick”的意义没什么关系,也和桶沾不上边。所以这句话的意义不能从它的子成分的意义推导出来。解决此类问题的一个方法是把语义赋给整个短语,而不是通过组合理论来表示其意义。到目前为止,我们都假定基本单位是词(或语素)。习惯用语表达式表明完整的短语也有基本(即非派生)意义,而且这种情况可以推而广之。在前一个例子中,动词短语“kick the bucket”有一个基本意义和动词“die”接近。观察发现,根据其组合意义找出的适合句子的特定解释不能直接解读习惯用语,这也从一方面支持了这种方法。例如,被动句“The bucket was kicked by Jack”(桶被 Jack 踢了)就不表示 Jack 死了。

有趣的是,“Jack kicked the bucket”是有歧义的。一种意义从直接组合每个词的意义得到,即(KICK1 *k1*(NAME *j1* "Jack") <THE *b* BUCKET>);另一个则从词 Jack 的意义和短语 kick the bucket 的基本意义组合得到,即(DIE1 *d1*(NAME *j1* "Jack"))。

解决这一问题的另一种方法是给习惯用语中的词引入新的意义。例如,“kick”的一个意义可能是 DIE1,由宾语的种类 BUCKET1 标记出来。虽然习惯用语是语言中非常有趣而且也非常重要的一个方面,但在下面几章并没有用多少篇幅来介绍它们。就本书的目的而言,读者可以假定基本意义总是和词联系在一起的。

如果语义解释的过程满足组合性,我们一定可以给任意语法成分赋予某种语义结构。例如,对于每个动词短语,如果可用在任何以 VP 为子成分的规则中,就一定可以给它赋予某种固定形式的意义。考虑最简单的情况,VP 由不及物动词组成,比如句子“Jack laughed”。一种看法是动词短语“laughed”的意义是一元谓词,对于任何在过去某个时间大笑的对象都为真。但这种方法能放之四海而皆准吗?换句话说,每个 VP 都有一元谓词的意义吗?让我们分析句子“Jack kissed Sue”及其逻辑形式:

(KISS1 *k1* (NAME *j1* "Jack") (NAME *s1* "Sue"))

动词短语“kissed Sue”的意义是什么?用上面的方法来解释,它是一个一元谓词,对于任何亲吻 Sue 的对象都为真。但到目前为止,还没有一种方法用来表达这种复杂的一元谓词。 λ 演算为此提供了一种形式化方法。举例来说,下面的表达式是一个后面只跟一个变量的谓词:

$(\lambda x (\text{KISS1 } k1 x (\text{NAME } s1 \text{ "Sue"})))$

可以将 x 看成变量,对于任何对象 O 此谓词都为真。因此,如果用 O 替换表达式中的 x ,就会得到一个真命题。和其他谓词一样,可以用一个 λ 表达式和一个参数来构造命题。在逻辑形式语言中,下面是一个命题:

$((\lambda x (\text{KISS1 } k1 x (\text{NAME } s1 \text{ "Sue"}))) (\text{NAME } j1 \text{ "Jack"}))$

这个命题为真,当且仅当 $(\text{NAME } j1 \text{ "Jack"})$ 满足谓词 $(\lambda x (\text{KISS1 } k1 x (\text{NAME } s1 \text{ "Sue"})))$;而后者为真,当且仅当下式为真:

$(\text{KISS1 } k1 (\text{NAME } j1 \text{ "Jack"}) (\text{NAME } s1 \text{ "Sue"}))$

我们通常说,最后一个表达式是把 λ 表达式 $(\lambda x (\text{KISS1 } x (\text{NAME } s1 \text{ "Sue"})))$ 应用于参数 $(\text{NAME } j1 \text{ "Jack"})$ 上得到的。这种操作称为 λ 归约。

前面讲过,为了建立紧密的句法-语义耦合,需要引入诸如 λ 表达式这样的新概念。为此,你可能想要抛弃这种方法,转而研究其他的语义解释方法。但是当语法变得越来越庞大,可以解释越来越复杂的现象时,组合理论却变得越来越有用。举个例子,用组合理论,即使句法结构不同,也可以把动词短语联结在一起,比如下面这个句子:

Sue laughs and opens the door. (Sue 大声笑了并且打开了门。)

句中有两个 VP。其中一个为“laughs”,它的语义解释是一元谓词,对任何大笑的人为真,即 $(\lambda a (\text{LAUGHS1 } l2 a))$;而另一个为“opens the door”,它也是一元谓词,对任何开门的人为真,即:

$(\lambda a (\text{OPENS1 } l2 a <\text{THE } d1 \text{ DOOR1}>))$

这两个一元谓词可以合起来形成一个复杂的一元谓词,对既大笑又开门的人为真,即:

$(\lambda a (& (\text{LAUGHS1 } l2 a) (\text{OPENS1 } o1 a <\text{THE } d1 \text{ DOOR1}>)))$

这才是 VP 的正确形式,并且还可以和其他 VP 类成分组合起来。例如,可以应用到逻辑形式为 $(\text{NAME } s1 \text{ "Sue"})$ 的主语 NP 上来表示原句的意义。

$(& (\text{LAUGHS1 } l2 (\text{NAME } s1 \text{ "Sue"}))$
 $(\text{OPENS1 } o1 (\text{NAME } s1 \text{ "Sue"}) <\text{THE } d1 \text{ DOOR1}>))$

让我们探讨另一个例子。名词短语中的介词短语修饰成分可以用多种方法表示。例如,孤立地看名词短语“The man in the store”中的短语“in the store”,也许看不出它的意义。而借助一种特殊的方法,可以在名词短语中查找位置修饰成分,并且将其融合到解释中去。但是,如果要解释诸如“The man is in the store”或“The man was thought to be in the store”这样的句子,这种方法就不起作用。如果介词短语有自己独立的意义,在这个例子里,一元谓词的形式就是:

$(\lambda o (\text{IN-LOC1 } o <\text{THE } s1 \text{ STORE1}>))$

这种解释像名词短语的修饰成分(添加新的限制)或是句子的谓词一样容易使用。名词短语“the man in the store”的逻辑形式如下所示:

$<\text{THE } m1 (\text{MAN1 } m1) (\text{IN-LOC1 } m1 <\text{THE } s1 \text{ STORE1}>)>$

而句子“The man is in the store”的逻辑形式如下所示:

(IN-LOC1 <THE m1 MAN1> <THE s1 STORE1>)

这只是两个简单的例子。如果用组合规则方法来表示语义,还会面临其他具有普遍性的问题。

框 9.1 λ 演算与 λ 归约

λ 演算是一门功能强大的基于简单原语集合的语言。 λ 演算的公式由如下形式的等式组成:

$$\langle \text{表达式} \rangle = \langle \text{表达式} \rangle$$

就我们的目的而言,本系统中最重要的公理是:

$$((\lambda x Px)a) = P\{x/a\}$$

其中, Px 是包含 x 的任意公式, $P\{x/a\}$ 是一个公式,其中的每个 x 都替换成 a 。根据这个公理,可以定义两个基本操作,即 λ 归约(将公理中的左侧变为右侧)和 λ 提取(将公理中的右侧变为左侧)。通常, λ 归约是我们更关心的一个操作,因为公式变得更简单。事实上,因为 λ 归约仅仅是用更简单的对等形式替换公式,所以这个操作从形式上并不一定能解决语义解释问题。但是,如果不使用 λ 归约,答案即便正确,往往也难以理解。

通常,每个大的句法短语都会对应于特定的语义结构。VP 和 PP 映射到一元谓词(可能是用 λ 表达式构建的复杂表达式),句子映射到命题,而 NP 映射到词项。小的语法类映射到表达式上,这个表达式在构造大的语法类时用来定义其角色。由于同一句法类中的每个成分都映射到同一类型的语义结构,因而可以用相同的方法处理它们。例如,不必知道 VP 的具体结构,只要其意义是一元谓词,就能用于构造更大成分的意义,而后者包含这个 VP。

9.2 带语义解释的简单语法和词典

本节构建一个简单的语法和一部简单的词典,用它们来说明在句法分析中如何用特征来计算逻辑形式。为了使例子简单易懂,逻辑形式使用的是上一节所用的谓词-参数结构,而不是论旨角色表达式。这样,就可以用相同的方法来处理所有次范畴结构相同的动词。9.4 节讨论如何对这个框架加以推广,使之能够识别论旨角色。

这里要做的主要扩展是给每个词项和语法规则增加 SEM 特征。例如,有一条语法规则是:

$$(S \text{ SEM } (?semvp ?semnp)) \rightarrow (NP \text{ SEM } ?semnp) (VP \text{ SEM } ?semvp)$$

给定带 SEM(NAME m1 "Mary") 的 NP 子成分,并给定 SEM 如下的 VP 子成分:

$$(\lambda a (SEES1 \text{ e8 } a (NAME \text{ j1 } "Jack")))$$

分析这条规则的呈现形式。新成分 S 的 SEM 特征如下:

$$((\lambda a (SEES1 \text{ e8 } a (NAME \text{ j1 } "Jack"))) (NAME \text{ m1 } "Mary"))$$

该表达式可用 λ 归约简化为:

$$(SEES1 \text{ e8 } (NAME \text{ m1 } "Mary") (NAME \text{ j1 } "Jack"))$$

这是该句的理想逻辑形式。图 9.1 显示了在给定每个成分的 SEM 特征条件下该句的句法分析树。

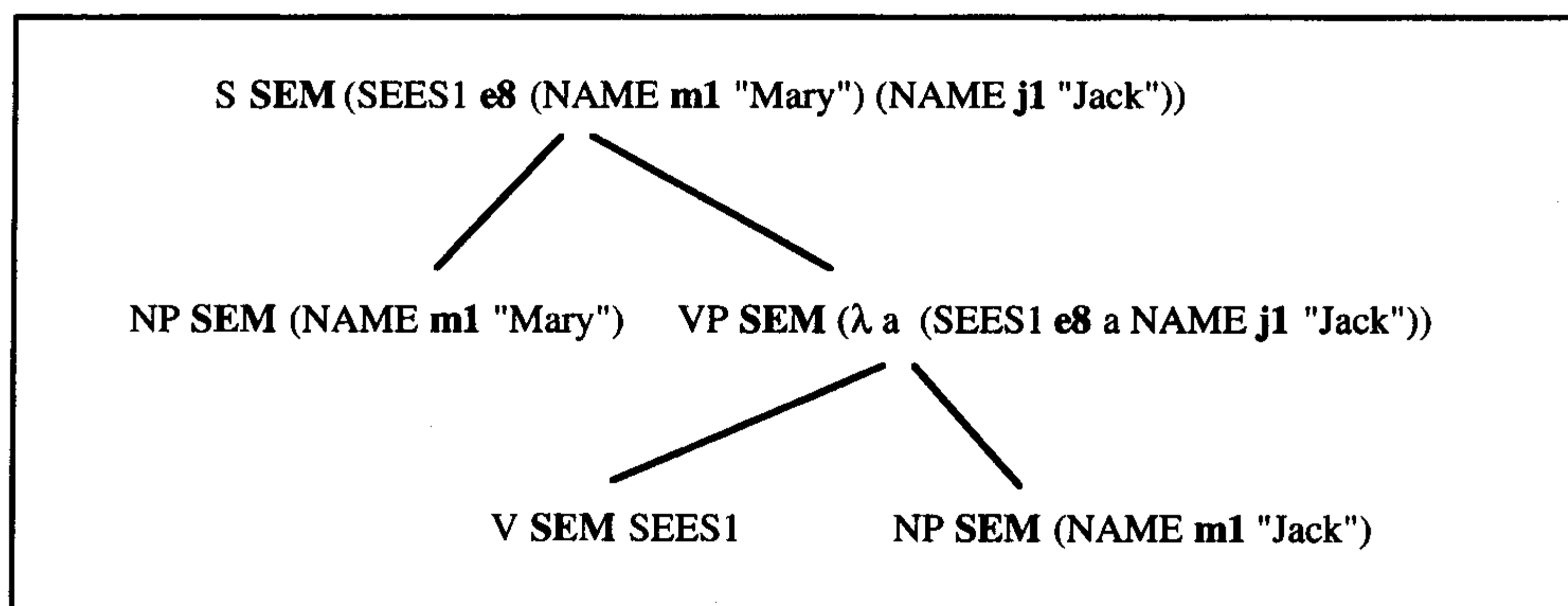


图 9.1 包含 SEM 特征的句法分析树

在词典中, SEM 特征用来标明每个词可能的意义。通常, 对于一个词每种可能的次范畴, 都有一个不同的词义, 因为这些词义对应于数量不同的谓词。样例词典如图 9.2 所示。如果一个词因不同的句法结构而有不同的 SEM 形式, 则需要有多个义项。例如, 动词“decide”有两个义项, 一个对应于 SUBCAT 为空的情况, 一种对应于 SUBCAT 为 `_pp:on` 的情况, 这里动词有一个附加的参数。需要注意的是, “fish”这个词也有两个义项, 因为其 SEM 取决于它是单数还是复数。

```

a (art AGR 3s SEM INDEF1)
can (aux SUBCAT base SEM CAN1)
car (n SEM CAR1 AGR 3s)
cry (v SEM CRY1 VFORM base SUBCAT _none)
decide (v SEM DECIDES1 VFORM base SUBCAT _none)
decide (v SEM DECIDES-ON1 VFORM base SUBCAT _pp:on)
dog (n SEM DOG1 AGR 3s)
fish (n SEM FISH1 AGR 3s)
fish (n SEM (PLUR FISH1) AGR 3p)
house (n SEM HOUSE1 AGR 3s)
has (aux VFORM pres AGR 3s SUBCAT pastprt SEM PERF)
he (pro SEM HE1 AGR 3s)
in (p PFORM {LOC MOT} SEM IN-LOC1)
Jill (name AGR 3s SEM "Jill")
man (n SEM MAN1 AGR 3s)
men (n SEM (PLUR MAN1) AGR 3p)
on (p PFORM LOC SEM ON-LOC1)
saw (v SEM SEES1 VFORM past SUBCAT _np AGR ?a)
see (v SEM SEES1 VFORM base SUBCAT _np IRREG-PAST + EN-PASTPRT +)
she (pro AGR 3s SEM SHE1)
the (art SEM THE AGR {3s 3p})
to (to AGR - VFORM inf)
  
```

图 9.2 包含 SEM 特征的一部小词典

分析语法 9.3, 该语法能处理非常简单的句子和动词短语, 并计算其逻辑形式。请注意, 除了 SEM 特征之外, 这里还引入了另一个新特征 VAR, 用于记录和成分相对应的篇章变量。它有助于解决特定形式的修饰成分, 这一点会在后面讲到。从词中构造出来一个词法成分时, 句法分析器会自动生成一个 VAR 特征, 且将该特征作为中心特征传给树。这样, 可以保证篇章变量总是惟一的。

1. (S SEM (?semvp ?semnp) → (NP SEM ?semnp) (VP SEM ?semvp)
2. (VP VAR ?v SEM (λ a2 (?semv ?v a2))) → (V[_none] SEM ?semv)
3. (VP VAR ?v SEM (λ a3 (?semv ?v a3 ?semnp))) →
(V[_np] SEM ?semv) (NP SEM ?semnp)
4. (NP WH - VAR ?v SEM (PRO ?v ?sempro)) → (PRO SEM ?sempro)
5. (NP VAR ?v SEM (NAME ?v ?semname)) → (NAME SEM ?semname)
6. (NP VAR ?v SEM <?semart ?v (?semcnp ?v)>) →
(ART SEM ?semart) (CNP SEM ?semcnp)
7. (CNP SEM ?semn) → (N SEM ?semn)

S, VP, NP, CNP 的中心特征: VAR

语法 9.3 带 SEM 特征的简单语法

对词语形态学中的派生规则也需要加以修改, 使之能够处理 SEM 特征。例如, 将单数名词转换为复数名词的规则可以接受单数名词的 SEM 特征, 并且给它加上 PLUR 运算符。

(N AGR 3p SEM (PLUR ?semn)) →
(N AGR 3s IRREG-PL - SEM ?semn) +S

这里, 用类似的方法把过去时和现在时的非辖域时态运算符添加了进去。修改后的词语形态规则如语法 9.4 所示。除了加入 SEM 特征, 这些规则和语法 4.5 原来的规则完全一样。

- L1. (V VFORM pres AGR 3s SEM <PRES ?semv>) →
(V VFORM base IRREG-PRES - SEM ?semv) +S
- L2. (V VFORM pres AGR {1s 2s 1p 2p 3p} SEM <PRES ?semv>) →
(V VFORM base IRREG-PRES - SEM ?semv)
- L3. (V VFORM past AGR {1s 2s 3s 1p 2p 3p} SEM <PAST ?semv>) →
(V VFORM base IRREG-PAST - SEM ?semv) +ED
- L4. (V VFORM pastprt SEM ?semv) →
(V VFORM base EN-PASTPRT - SEM ?semv) +ED
- L5. (V VFORM pastprt SEM ?semv) →
(V VFORM base EN-PASTPRT + SEM ?semv) +EN
- L6. (V VFORM ing SEM ?semv) →
(V VFORM base SEM ?semv) +ING
- L7. (N AGR 3p SEM (PLUR ?semn)) →
(V AGR 3s IRREG-PL - SEM ?semn) +S

语法 9.4 带语义解释的词语形态规则

我们前面讨论过规则 1。规则 2 和规则 3 处理及物和不及物动词,并构造正确的 VP 解释。每条规则都接受动词的 SEM(即?semv)并且构造可应用于主语的一元谓词。动词意义的参数包括一个事件变量(由 VAR 特征来记录)、主语以及用于表示次语类成分的任何附加参数。如果已知代词的意义?sempro,规则 4 就能构造代词正确的 SEM 结构,而规则 5 对专有名词进行同样的操作。规则 6 定义了一个表达式,其中包含一个非辖域定量表达式,该表达式是由量词?semart、篇章变量?v 以及一个限制量词的命题(通过把一元谓词?semcnp 应用于会话上得到)组成。例如,假设篇章变量?v 为 **m1**,名词短语“the man”将“the”的 SEM(即操作符 THE)和“man”的 SEM(即 MAN1)组合成表达式 $\langle \text{THE } \mathbf{m1}(\text{MAN1 } \mathbf{m1}) \rangle$ 。规则 7 用一个单独的 N 构造简单的 CNP。由于一般名词的 SEM 已经是一元谓词,因此,这个值只作为 CNP 的 SEM。

只须对标准的 chart 分析器做两处简单的改进,就可以用它来处理语义解释:

- 当词法规则实例化时,就将 VAR 特征设置成一个新的篇章变量。
- 无论何时,只要构建一个成分,就要尽可能用 λ 归约对 SEM 进行简化。

通过这两处改变,现有的分析器也能够在分析的过程中计算逻辑形式。我们看看“Jill saw the dog”(Jill 看到了狗)的分析过程,其分析树如图 9.5 所示。通过查找词典,“Jill”这个词分析成人名。这样,就产生了一个新的篇章变量 **j1**,并设置为 VAR 特征。规则 5 用这个成分构造 NP。由于 VAR 是中心特征,则 NP 的 VAR 特征就是 **j1**,而用一种简单的方法也可以从等式中构造出 SEM。用“saw”的词项可以生成成分 V,其 SEM 为 $\langle \text{PAST SEES1} \rangle$,而 VAR 为 **ev1**。规则 6 将“the”和“dog”两个词项的 SEM 和名词的 VAR 组合起来,构造成一个 NP,其 SEM 为 $\langle \text{THE } \mathbf{d1}(\text{DOG } \mathbf{d1}) \rangle$ 。然后,这个 NP 再按规则 3 与动词的 SEM 及其 VAR 组合,形成 VP,其 SEM 为:

$(\lambda x (\langle \text{PAST SEES1} \rangle \mathbf{ev1} \ x \ \langle \text{THE } \mathbf{d1}(\text{DOG } \mathbf{d1}) \rangle))$

上式再和主语 NP 组合形成句子最终的逻辑形式。

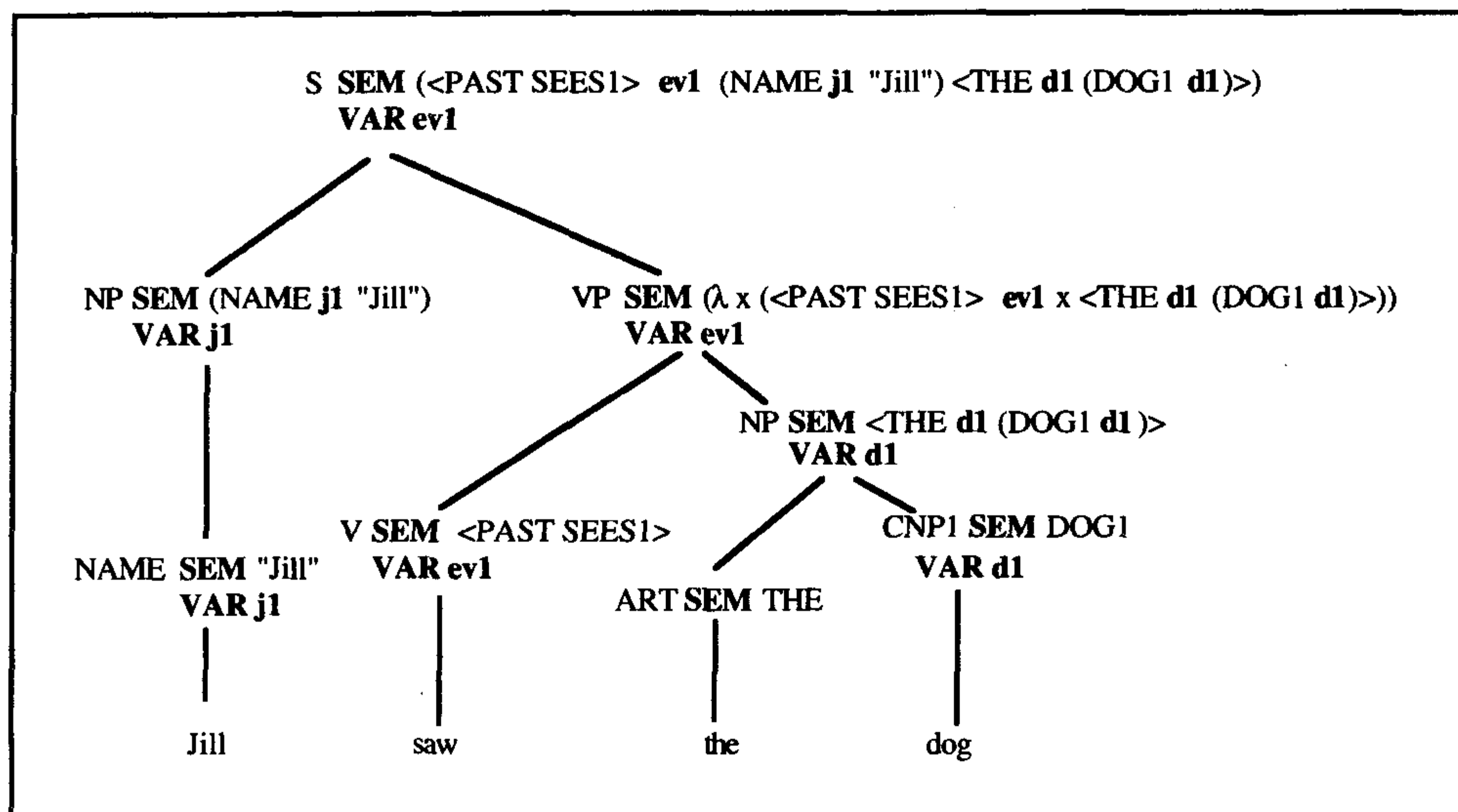


图 9.5 句子“Jill saw the dog”含 SEM 和 VAR 特征的句法分析过程

本节完整地介绍了基本的语义解释过程。借助两个新特征和在上述分析器基础上的两处细小扩展,我们定义了一个可在分析过程中构造逻辑形式的语法。采用本书中任何基于 chart 的分析策略,这个方法都有效。

9.3 介词短语与动词短语

虽然上一节介绍了对句子进行语义解释所需的全部内容,但其中只讲到了最简单的解释方法。本节将阐述一些语法规则的其他例子,这些规则能够处理更复杂的现象。具体地说,它更详细地说明了动词短语和介词短语的解释过程。首先,分析助动词的处理规则:

$$\begin{aligned} &(\text{VP SEM } (\lambda a1 \text{ } (?semaux \text{ } (?semvp a1)))) \rightarrow \\ &\quad (\text{AUX SUBCAT } ?v \text{ SEM } ?semaux) \\ &\quad (\text{VP VFORM } ?v \text{ SEM } ?semvp) \end{aligned}$$

这条规则为新的 VP 在适当的位置插入了一个情态运算符。新 VP 的 SEM 等式比较复杂,因此,要更仔细地研究它。如果 ?semaux 是一个情态运算符,比如 CAN1,而 ?semvp 是 λ 表达式,比如 $(\lambda x(\text{LAUGHS1 } e3 \text{ } x))$,那么,根据助动词规则,动词短语“can laugh”的 SEM 就是:

$$(\lambda a1 (\text{CAN1 } ((\lambda x (\text{LAUGHS1 } e3 \text{ } x)) a1)))$$

上式可简化为:

$$(\lambda a1 (\text{CAN1 } (\text{LAUGHS1 } e3 \text{ } a1)))$$

如果将这种类型的 SEM 等式看做对主语变量的“提升”,通过这种“提升”,将变量放到了 CAN1 运算符之外,这样可能会比较容易理解。以 VP 的解释 $(\lambda x(\text{LAUGHS1 } e3 \text{ } x))$ 作为起点,这个等式构造了一个新的公式。该公式在包含 CAN1 运算符的同时,还将主语的 λ 变量放在公式的外面。注意,和所有的 VP 一样,新的 SEM 是一个可应用于主语的一元谓词,因此,助动词规则可以递归使用,用来分析更复杂的助动词序列。

分析介词短语时很重要的一点是,要认识到它们在句子中扮演双重语义角色。其中一种分析认为 PP 是名词短语或动词短语的修饰成分。另一种分析认为 PP 的用途在于区分中心词的次范畴,介词充当的是参数位置的标识而非独立的谓词。

首先,让我们分析 PP 充当修饰成分的情况。在这些情况下,无论 PP 最终修饰的是什么,它的 SEM 都可以作为一元谓词来使用。因此,下述规则恰好可以用于构造一个 PP 修饰成分:

$$(\text{PP SEM } (\lambda y \text{ } (?semp y \text{ } ?semnp))) \rightarrow (\text{P SEM } ?semp) (\text{NP SEM } ?semnp)$$

给定一个介词短语“in the corner”,如果 P 的 SEM 是 IN-LOC1, NP 的 SEM 是 $\langle \text{THE } c1 (\text{CORNER1 } c1) \rangle$,那么 PP 的 SEM 就是一元谓词:

$$(\lambda y (\text{IN-LOC1 } y \text{ } \langle \text{THE } c1 \text{ CORNER1} \rangle))$$

现在,思考名词短语“the man in the corner”的解释。一个能把 PP 修饰成分集成进来的规则是:

$$\begin{aligned} &(\text{CNP SEM } (\lambda n1 \text{ } (& \text{ } (?semcnp n1) \text{ } (?semp n1)))) \rightarrow \\ &\quad (\text{CNP SEM } ?semcnp) (\text{PP SEM } ?semp) \end{aligned}$$

已知 CNP man 的 SEM 是一元谓词 MAN1,介词短语“in the corner”是 $(\lambda y (\text{IN1 } y \text{ } \langle \text{THE } c1 \text{ CORNER1} \rangle))$,简化之前 CNP 的新 SEM 是:

$$(\lambda n1 \text{ } (& \text{ } (\text{MAN1 } n1) \text{ } ((\lambda y (\text{IN1 } y \text{ } \langle \text{THE } c1 \text{ CORNER1} \rangle)) n1)))$$

它的子表达式 $((\lambda y(\text{IN1 } y < \text{THE } c1 \text{ CORNER1 } >))n1)$ 可以简化为 $(\text{IN1 } n1 < \text{THE } c1 \text{ CORNER1 } >)$ 。因此,整个表达式就变为:

$(\lambda n1 (& (\text{MAN1 } n1) (\text{IN1 } n1 < \text{THE } c1 \text{ CORNER1 } >))))$

这是一个一元谓词,对任何一位角落里的男人都为真,这也是我们所期望的解释。将这个表达式和量词(比如用规则 6)组合起来就形成如下的 SEM:

$< \text{THE } m2 ((\lambda z (& (\text{MAN1 } z) (\text{IN1 } z < \text{THE } c1 \text{ CORNER1 } >))) m2) >$

它可以简化为:

$< \text{THE } m2 (& (\text{MAN1 } m2) (\text{IN1 } m2 < \text{THE } c1 \text{ CORNER1 } >)) >$

PP 也可以修饰动词短语,比如“Jill can cry in the corner”(Jill 会在墙角叫喊)中的动词短语“cry in the corner”。引入 PP 修饰成分的句法规则是:

$\text{VP} \rightarrow \text{VP PP}$

你可能认为可以按名词短语的方式来处理 SEM 特征等式,只是更复杂一些罢了。让我们思考下面这种情况。VP 子成分“cry”的逻辑形式为:

$(\lambda x (\text{CRIES1 } e1 x))$

而前面已经列出 PP 的逻辑形式。整个动词短语“cry in the corner”的逻辑形式是:

$(\lambda a (& (\text{CRIES1 } e1 a) (\text{IN-LOC1 } e1 < \text{THE } c1 \text{ CORNER1 } >)))$

问题是,VP 子成分的一元谓词要应用于主语,而 PP 修饰成分的一元谓词则必须应用于 VP 生成的篇章变量。因此,用规则 9 的方法并不能得到正确的答案。相反,从 PP 构造出的 SEM 必须应用到篇章变量上。换句话说,正确的规则是:

$(\text{VP VAR ?v SEM } (\lambda x (& (?semvp x) (?semppp ?v)))) \rightarrow$
 $(\text{VP VAR ?v SEM ?semvp}) (\text{PP SEM ?semppp})$

使用这条规则,可以得到动词短语“cry in the corner”的句法分析树,如图 9.6 所示。

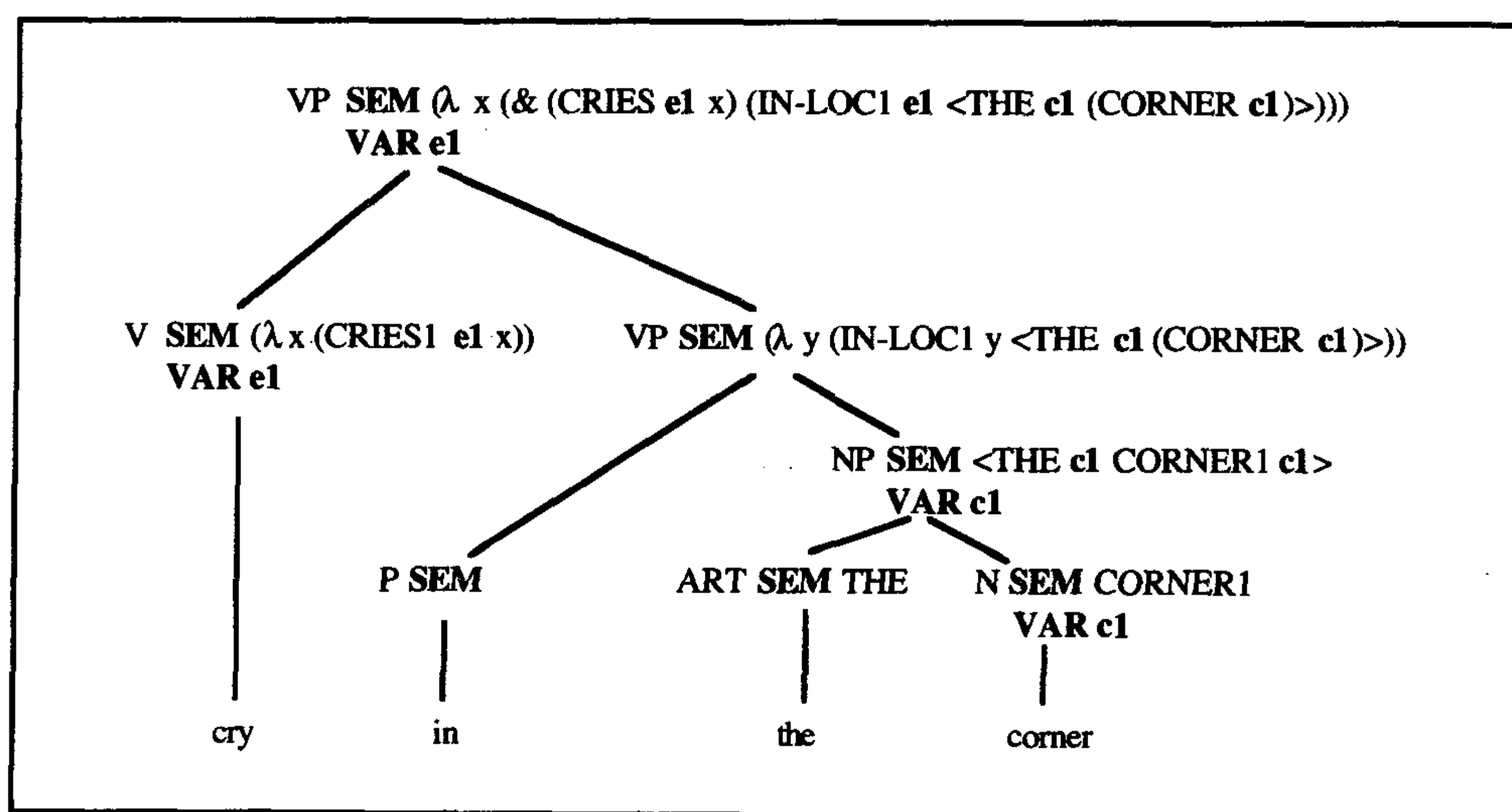


图 9.6 用 VAR 特征表示 VP 的 PP 修饰成分

介词短语也可以作为动词短语的次范畴成分,而且我们必须就事论事。确切地说,要由动词决定如何解释介词短语。例如,如果孤立分析介词短语“on a couch”,它可能表示某个物体或某件事的位置;但如果加上动词“decide”,则可能表示某个正在考虑的对象。我们可以举例说明这两种解读之间的区别,分析句子“Jill decided on a couch”两种解读的不同意义:

- Jill made a decision while she was on a couch. (Jill 在长椅上做了决定。)
- Jill made a decision about a couch. (Jill 做了一个有关长椅的决定。)

第一种情况下将“on a couch”看做状语 PP,我们前面已经讨论过这种情况。那么,第二种情况的语义解释等式是什么呢?正确的句法规则是:

$VP \rightarrow V[_{pp: on}] NP PP[on]$

最终可得到 VP 的逻辑形式是:

$(\lambda s (DECIDES-ON1 \ d1 \ s \ <A \ c1 (COUCH \ c1) >))$

需要注意的是,在这种情况下单词“on”的语义似乎没有什么作用。很多系统把区分次范畴的 PP 按照两种不同的介词短语来处理。我们引入一个新的二元特征 PRED,其中 + 表示介词短语应该视为一个谓词,也就是说,应该将它解释成一个参数,即词项。借助这种二元特征,可以为介词短语定义两种规则。也就是语法 9.7 中的规则 8 和规则 9。规则 8 仅限于处理带 + PRED 值的介词短语,此时 PP 充当修饰成分。规则 9 则用于解决所有次语类介词短语,它们带有 - PRED 特征。在这种情况下,PP 的 SEM 就是宾语 NP 的 SEM。语法 9.7 总结了本节提出的所有其他规则。

8. $(PP \ PRED \ + \ SEM \ (\lambda x \ (?sempp \ x \ ?semnp))) \rightarrow$
 $(P \ SEM \ ?sempp) (NP \ SEM \ ?semnp)$
 9. $(PP \ PRED \ - \ PFORM \ ?pf \ SEM \ ?semnp) \rightarrow$
 $(P \ ROOT \ ?pf) (NP \ SEM \ ?semnp)$
 10. $(VP \ VAR \ ?v \ SEM \ (\lambda ag1 \ (& \ (?semvp \ ag1) \ (?sempp \ ?v)))) \rightarrow$
 $(VP \ SEM \ ?semvp) (PP \ PRED \ + \ SEM \ ?sempp)$
 11. $(VP \ VAR \ ?v \ SEM \ (\lambda ag2 \ (?semv \ ?v \ ag2 \ ?sempp))) \rightarrow$
 $(V[_{np_pp: on} \ SEM \ ?semv) (PP \ PRED \ - \ PFORM \ on \ SEM \ ?sempp)$
 12. $(VP \ SEM \ (\lambda a1 \ (?semaux \ (?semvp \ a1)))) \rightarrow$
 $(AUX \ SUBCAT \ ?v \ SEM \ ?semaux) (VP \ VFORM \ ?v \ SEM \ ?semvp)$
 13. $(CNP \ SEM \ (\lambda n1 \ (& \ (?semcnp \ n1) \ (?sempp \ n1)))) \rightarrow$
 $(CNP \ SEM \ ?semcnp) (PP \ PRED \ + \ SEM \ ?sempp)$
- PP 的中心特征: PFORM
 VP, CNP 的中心特征: VAR

语法 9.7 动词短语中 PP 的处理规则

图 9.8 给出了动词短语的“decide on a couch”两种解读,依据的是语法 9.3 和语法 9.7 定义的语法。图的上半部给出的是“与长椅有关的决定”这种解读,如前所述,PP 的特征是 - PRED,它的 SEM 是 $\langle A \ c1 \ COUCH1 \rangle$ 。图的下半部给出了“决定在长椅上做出”这种解读,PP 的特征是 + PRED,它的 SEM 是 $(\lambda x (ON-LOC1 \ x \ \langle A \ c1 \ COUCH1 \rangle))$ 。

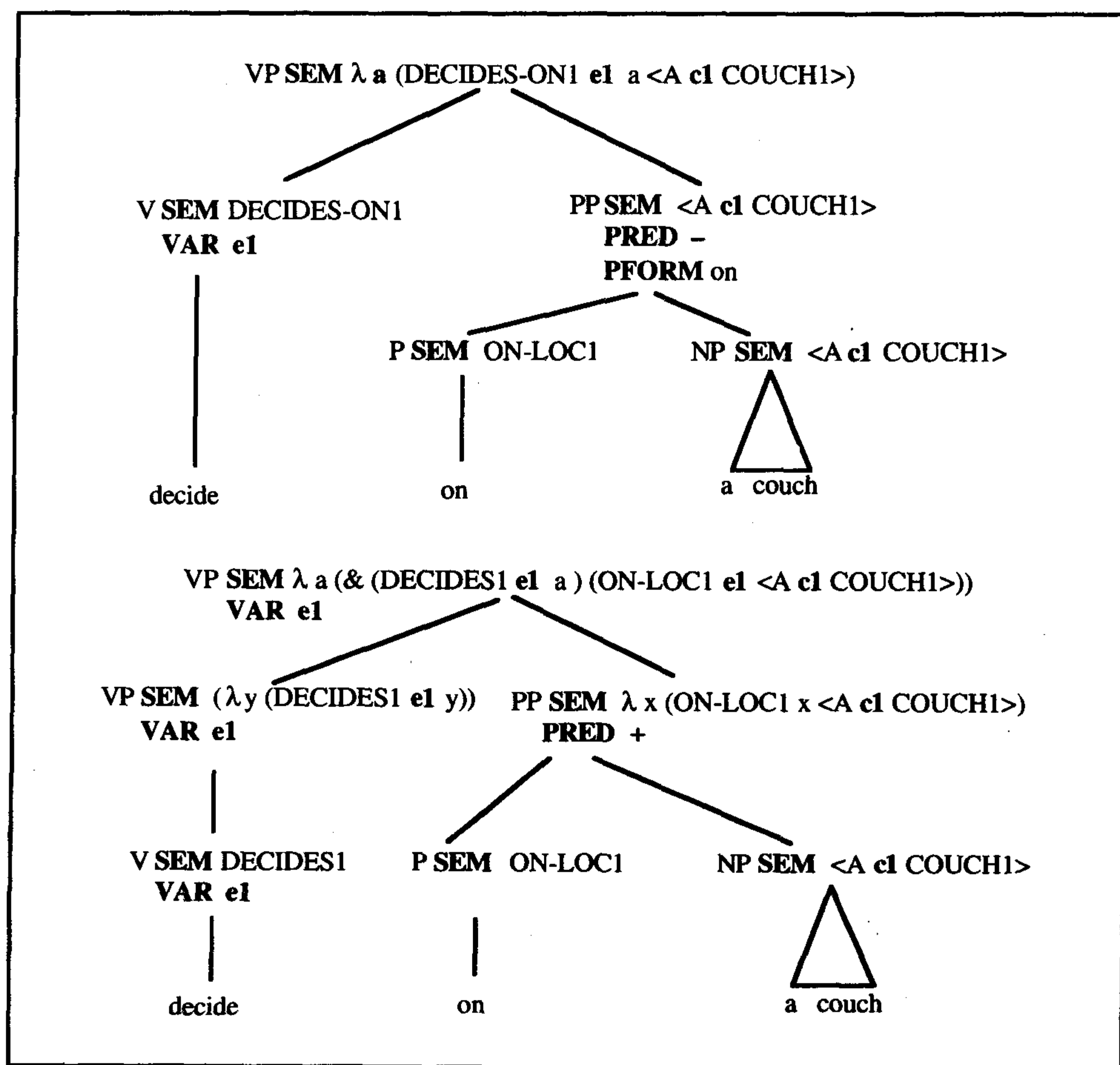


图 9.8 动词短语“decide on a couch”的两种可能的分析树

9.4 词汇化语义解释与语义角色

到目前为止,词条的语义形式都由每个词的所有意义组成,错综复杂的语义解释是用语法规则表示的。这种策略虽然合情合理,但是有很多研究人员使用与此不同的方法。这种方法用词条来表示复杂结构,而且语法规则更简单。考虑动词“decide”,假设分析它的不及物意义 DECIDES1。其 SEM 是意义 DECIDES1,用规则 2 构造得到 λ 表达式(λy (DECIDES1 e1 y))。另外有一种方法将该词条的 SEM 定义为(λy (DECIDES1 e1 y)),用规则 2 的 SEM 等式就是这个动词的 SEM。类似可得,词条表示及物意义的 SEM 是表达式(λo (λy (DECIDES-ON1 e1 y o)))。规则 3 的 SEM 等式将该谓词应用于宾语的 SEM,从而得到正确的 SEM,该 SEM 已在前面列出。

这里,在语法规则的复杂性和词条的复杂性之间进行折中。到目前为止,我们所研究的语法好像用一部简单的词典更适合。但如果逻辑形式更加复杂,人们就倾向于用另一种方法取而代之。举例来说,让我们想想如何定义一个语法来生成基于论旨角色的逻辑形式。首先思考一下,如果词典只包含原子词义,会产生什么。因为早期语法只有一条涵盖所有及物动词的规则,因而新语法需要用及物动词的 THEME 角色对它们加以区分,而且每个动词都要用一条

单独的规则。例如动词“see”和“eat”，它们都有及物形式，在这种形式中主语充当 AGENT 角色，宾语充当 THEME 角色。另一方面，动词“break”也有一种意义，在这种意义中主语充当 INSTR 角色，宾语充当 THEME 角色，比如“The hammer broke the window”。要处理这两种情况，需要对词条添加一个新特征，叫做 ROLES，用来判断正确的形式，而且对每一个词都添加一个单独的语法规则。这样就得到如下的表达式：

$$\begin{aligned} &(\text{VP VAR ?v SEM } (\lambda a (?semv ?v [\text{AGENT } a] [\text{THEME ?semnp}]))) \rightarrow \\ &\quad (\text{V ROLES AG-THEME SEM ?semv}) (\text{NP SEM ?semnp}) \\ &(\text{VP VAR ?v SEM } (\lambda a (?semv ?v [\text{INSTR } a] [\text{THEME ?semnp}]))) \rightarrow \\ &\quad (\text{V ROLES INSTR-THEME SEM ?semv}) (\text{NP SEM ?semnp}) \end{aligned}$$

对应于动词的其他角色组合，要添加额外的规则。

显然，这种方法很麻烦，因为要向词典中添加论旨角色信息（使用 ROLES 特征）。如果在词典中以恰当的形式来表示，可能会更简单。比如，如果词条是：

$$\begin{aligned} \text{see:} & \quad (\text{V VAR ?v SEM } (\lambda o (\lambda a (\text{SEES1 ?v} [\text{AGENT } a] [\text{THEME ?o}])))) \\ \text{break:} & \quad (\text{V VAR ?v SEM } (\lambda o (\lambda a (\text{BREAKS1 ?v} [\text{INSTR } a] \\ & \quad \quad \quad [\text{THEME ?o}])))) \end{aligned}$$

那么语法规则：

$$\begin{aligned} &(\text{VP SEM } (?semv ?semnp)) \rightarrow \\ &\quad (\text{V SEM ?semv}) (\text{NP SEM ?semnp}) \end{aligned}$$

就可以涵盖这几种情况。考察动词短语“see the book”，其中“see”的 SEM 如上所述，而“the book”是 $\langle \text{THE } b1(\text{BOOK1 } b1) \rangle$ 。VP 的 SEM 是：

$$((\lambda o (\lambda a (\text{SEES1 } b1 [\text{AGENT } a] [\text{THEME } o]))) \langle \text{THE } b1 (\text{BOOK1 } b1) \rangle)$$

上式可以用 λ 归约简化为：

$$(\lambda a (\text{SEES1 } b1 [\text{AGENT } a] [\text{THEME } \langle \text{THE } b1 (\text{BOOK1 } b1) \rangle]))$$

对动词短语“break the book”用同样的规则，将上述“break”的 SEM 应用到“the book”的 SEM，这样就产生归约逻辑形式：

$$(\lambda a (\text{BREAKS1 } b1 [\text{INSTR } a] [\text{THEME } \langle \text{THE } b1 (\text{BOOK1 } b1) \rangle]))$$

9.4.1 层次词典

词的数量太多是导致词典越来越复杂的主要原因。即使词条很简单，制定一部词典也是一件很困难的事。因为动词后可接的每种补语结构都有对应的语义解释，所以即便是只定义最常见意义的语义解释，也是一件单调乏味的事情。例如，动词“give”可接受以下形式：

I gave the money. (我给了钱。)

I gave John the money. (我给了 John 钱。)

I gave the money to John. (我把钱给了 John。)

give 的词条包括如下几条：

$$\begin{aligned} &(\text{V SUBCAT } _np \\ &\quad \text{SEM } \lambda o \lambda a (\text{GIVE1 } * [\text{AGENT } a] [\text{THEME } o])) \end{aligned}$$

```

(V  SUBCAT _np_np
  SEM  $\lambda r \lambda o \lambda a$  (GIVE1 * [AGENT a] [THEME o] [TO-POSS r]))
(V  SUBCAT _np_pp:to
  SEM  $\lambda o \lambda r \lambda a$  (GIVE1 * [AGENT a] [THEME o] [TO-POSS r]))

```

如果对每个动词重复同样的步骤,这项工作将是一个沉重的负担。幸运的是,可以利用英语中动词的某些一般性规律更有效地完成这件事。例如,有一大类动词(包括大多数及物动词)都使用同样的语义解释规则来处理 `_np SUBCAT` 形式。这一大类包括“give”,“take”,“see”,“find”,“paint”等几乎所有描述行为的动词。层次词典的思想是将动词意义按某种方法组织起来,通过这种方法,可以将它们的共性准确地刻画出来。这种方法建立在一种称为继承的技术之上。在这种技术中,在整个层次体系里,词义继承或直接获得其上若干层的抽象类的性质。例如,基于动词的 SUBCAT 和 SEM 性质,可以建立一个很有用的词法层次体系,接近这个层次顶端的部分定义了普通动词类的抽象动词意义。举例来说,抽象类 INTRANS-ACT 定义这样一类动词,它们接受 SUBCAT“_none”,并且有如下的语义解释规则:

```
 $\lambda s$  (?PREDN * [AGENT s])
```

其中,?PREDN 是一个由动词确定的谓词名称。还需要在每个词的词条中说明实际的谓词名称,除此之外,诸如“run”,“laugh”和“sit”这样的不及物动词的语义解释都已经给出了完整的定义。还有一种常见形式是描述动作的简单及物动词,包括上面所列的那些动词。这种形式叫做 TRANS-ACT,它有一个 SUBCAT“_np”和一个 SEM“ $\lambda o \lambda a$ (?PREDN * [AGENT a][THEME o])”。

可以为所有的普通动词形式定义相似的类,并且构造一个继承结构,将动词语义与它们后面可接的形式关联起来。图 9.9 展示了一个词法层次,这个层次表示四种不同的语义,等价于下面这些不带层次的词条:

run(不及物的“运动”意义, RUN1):

```

(SUBCAT _none
  SEM  $\lambda a$  (RUN1 * [AGENT a]))

```

run(及物的“操作”意义, OP1):

```

(SUBCAT _np
  SEM  $\lambda o \lambda a$  (OP1 * [AGENT a] [THEME o]))

```

donate(接受及物和“to”形式):

```

(SUBCAT _np
  SEM  $\lambda o \lambda a$  (DONATE1 * [AGENT a] [THEME o]))
(SUBCAT _np_pp:to
  SEM  $\lambda o \lambda r \lambda a$  (DONATE1 * [AGENT a] [THEME o] [TO-POSS r]))

```

当然,还包括:

give(上述所有形式):

```

(SUBCAT _np
  SEM  $\lambda o \lambda a$  (GIVE1 * [AGENT a] [THEME o]))
(SUBCAT _np_pp:to
  SEM  $\lambda o \lambda r \lambda a$  (GIVE1 * [AGENT a] [THEME o] [TO-POSS r]))
(SUBCAT _np_np
  SEM  $\lambda r \lambda o \lambda a$  (GIVE1 * [AGENT a] [THEME o] [TO-POSS r]))

```

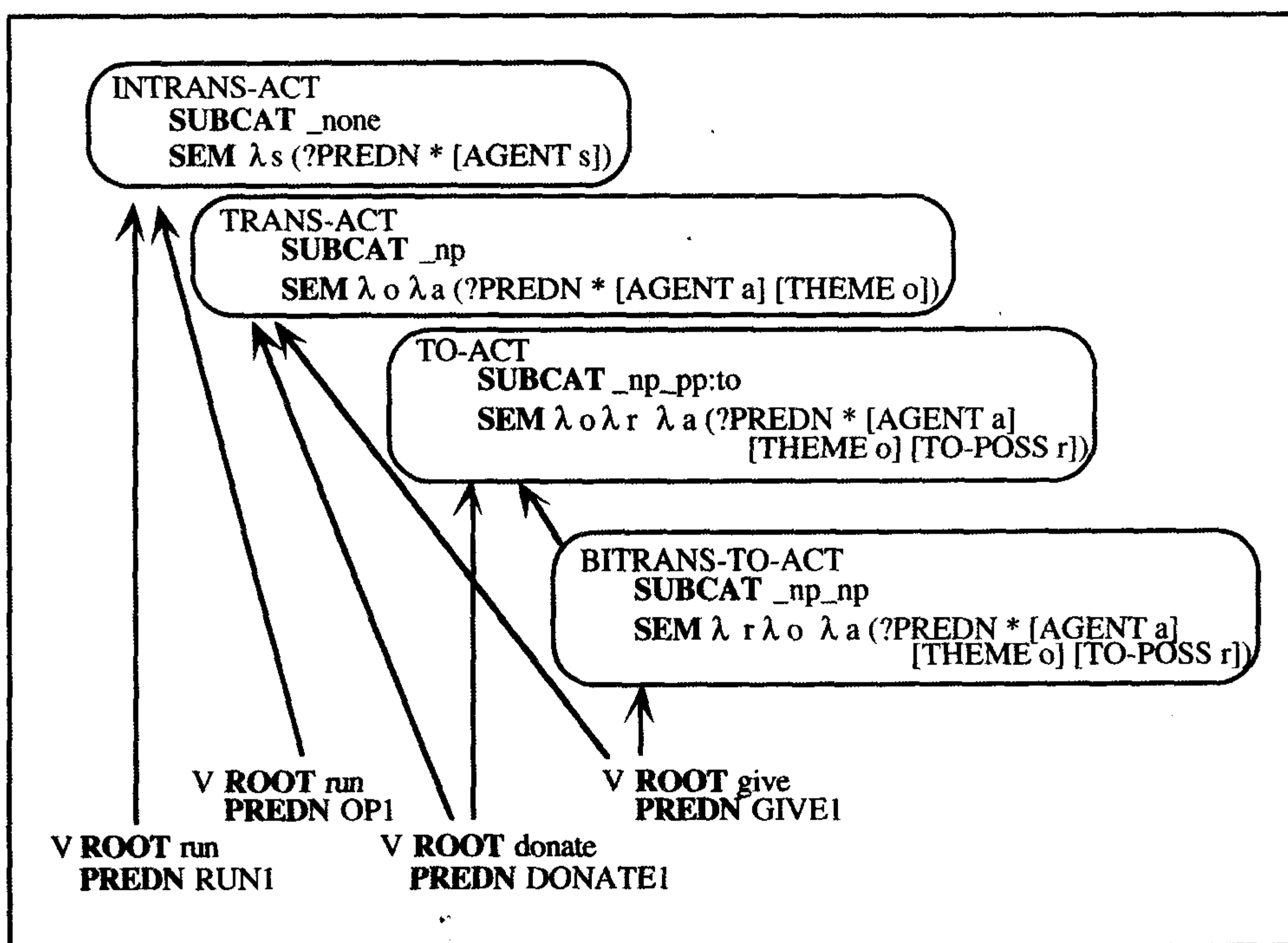



图 9.9 基于 SUBCAT 和 SEM 特征的动词词法的部分层次体系

如果为每个词条再添加一个称为 SUP(表示超类, superclass)的特征,就可以构造词法层次体系。这种特征以抽象类列表作为值,而成分继承了这些抽象类的属性。如果写一个程序,在存取词典时搜索这个层次体系从而找到所有相关特征值,就将是一件比较简单的事了。

```
give:  (V  ROOT give
        PREDN GIVE1
        SUP (BITRANS-TO-ACT TRANS-ACT))
```

9.5 简单疑问句的处理

到目前为止,我们探讨的语法都只是处理简单的陈述句。如果要扩展语法使之能处理其他类型的句子,需要增加一些规则来解释特殊疑问词、倒装句,还需要用缺位传播来解决特殊疑问句。我们目前设计的方法可以直接解决这个问题,而且不需要对分析器做任何额外的扩展。惟一要做的事情是添加一个 S 规则(这个规则是最早在第 5 章为解决疑问句而提出来的),并构造恰当的 SEM 特征等式。

这里,需要说明的地方就是 SEM 特征和 GAP 特征相互作用的机理。回想一下,很多特殊疑问句都用 GAP 特征来构造恰当的结构,比如:

Who did Jill see? (Jill 看到了谁?)

Who did Jill want to see? (Jill 想看到谁?)

Who did Jill want to win the prize? (Jill 希望谁赢得比赛?)

第 5 章为解决这些问题而引入的规则是:

```
(S INV -) → (NP WH Q AGR ?a) (S INV + GAP (NP AGR ?a))
```

也就是说,特殊疑问句包含一个 WH 特征为 Q 的 NP,后面紧跟一个 NP 为空的倒装句,而这个空 NP 就是第一个 NP。要使语义解释起作用,就要给 GAP 特征添加 SEM 特征。通过这种方法,特征传递给了 S 结构。因此,当发现缺位时就可以在正确的位置使用特征。修订后的规则是:

(S INV – SEM (WH-query ?sems)) →
 (NP WH Q AGR ?a SEM ?semnp)
 (S INV + SEM ?sems GAP (NP AGR ?a SEM ?semnp))

语法 9.10 给出了新的规则来处理这种问题。规则 14 已经讨论过了。规则 15 处理倒装句,可用于处理一般疑问句以及这里讨论的特殊疑问句。规则 17 可以接受由疑问代词(比如“who”和“what”)构造的名词短语。

14. (S INV – SEM (WH-query ?sems)) →
 (NP WH Q AGR ?a SEM ?semnp)
 (S INV + SEM ?sems GAP (NP AGR ?a SEM ?semnp))
15. (S INV + GAP ?g SEM (?semaux (?semvp ?semnp))) →
 (AUX AGR ?a SUBCAT ?s SEM ?semaux)
 (NP AGR ?a GAP – SEM ?sempp)
 (VP VFORM ?s GAP ?g SEM ?semvp)
16. (NP WH Q VAR ?v SEM <WH ?v (?sempro ?v)>) →
 (PRO WH Q SEM ?sempro)

语法 9.10 处理简单特殊疑问句的规则

特殊疑问词的词条也需要用 SEM 特征扩展。例如,“who”的词条是:

(PRO WH {Q R} SEM WHO1 AGR {3s 3p})

谓词 WHO1 对于可作为此类问题答案的任何对象都为真,包括人和其他有生命的主体。

为了理解 SEM 特征和 GAP 特征相互作用的机理,我们研究疑问句“Who did Jill see?”的句法分析树的演化过程,如图 9.11 所示。使用规则 16,“who”分析为名词短语:

(NP WH Q AGR 3s SEM <WH p1 (WHO1 p1)>)

这个成分可以作为规则 14 的起点,因此我们需要如下成分来完成这个规则:

(S INV +
 GAP (NP AGR 3s SEM <WH p1 (WHO1 p1)>)
 SEM ?sems)

可用规则 15 对其进行改写,“did”和“Jill”这两个词提供了前两个成分。所需的第三个成分是:

(VP VFORM base
 GAP (NP AGR 3s SEM <WH p1 (WHO1 p1)>)
 SEM ?semvp)

这是一个带空 NP 的 VP。语法 9.3 中的规则 3 用于“see”这样的及物动词。GAP 特征用于填充动词空的宾语,经过实例化得到 ?semnp to <WH p1 (WHO p1)>。因此,新 VP 的 SEM 是:

($\lambda a3$ (SEES1 s1 a3 <WH p1 (WHO1 p1)>))

中心词“who”的 SEM 已经位于句子中的恰当位置。于是,我们可以像通常那样完成分析。这种将 SEM 传进 GAP 的技术是一种完全通用的方法,可用来处理第 5 章讨论的所有类型的疑问句。

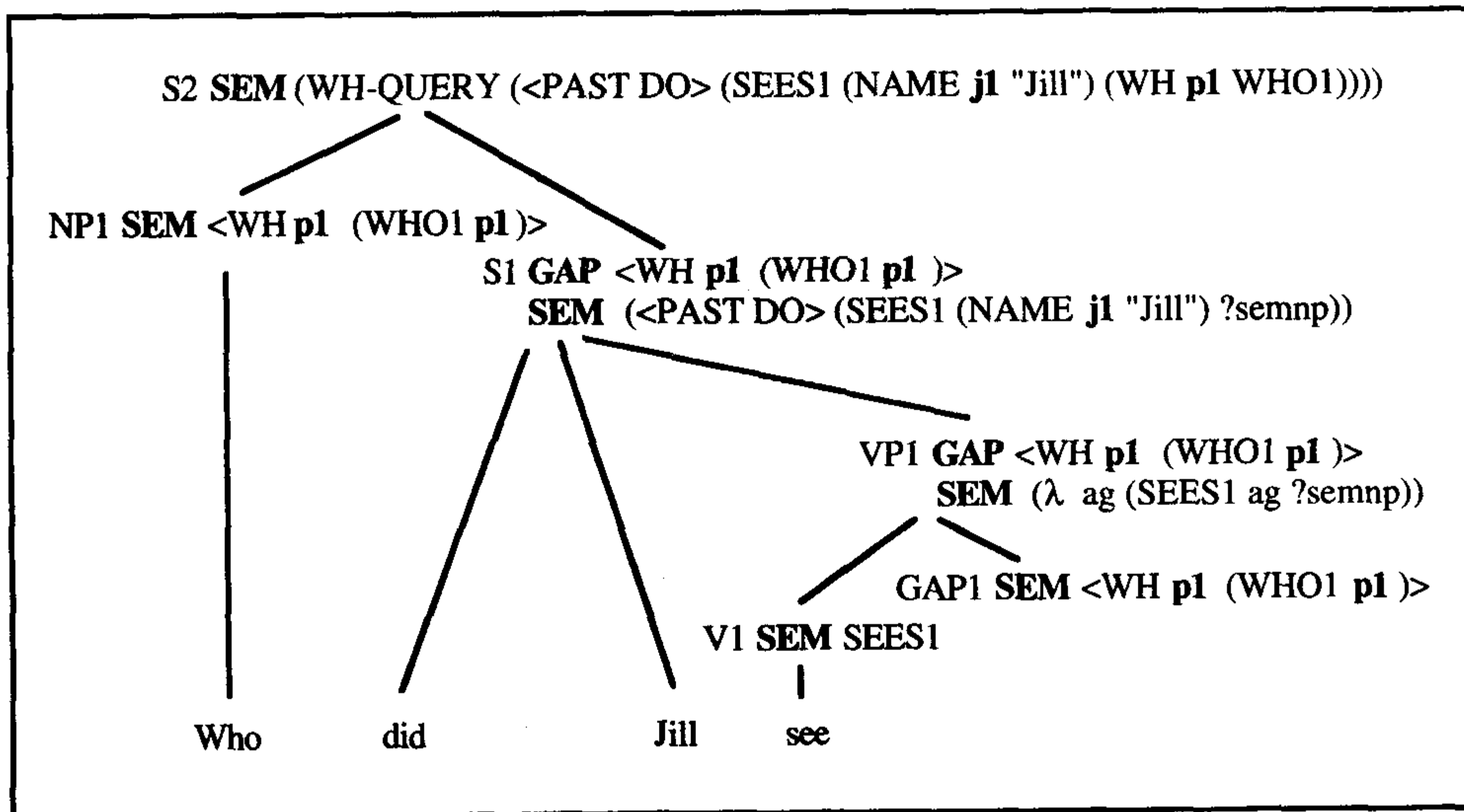


图 9.11 “Who did Jill see?”的句法分析树

9.5.1 介词短语的特殊疑问句

疑问句也可以用介词短语作为起始,比如“In which box did you put the book?”(你把书放到哪个盒子里了?)“Where did you put the book?”(你把书放到哪里了?)和“When did he disappear?”(他什么时候消失的?)这种问题的语义解释取决于 PP 是动词的次范畴成分还是 VP 的修饰成分。很多这种疑问句都可以用和规则 14 相似的规则来处理,而规则 14 与 NP 特殊疑问句相对应,即:

(S INV – SEM (WH-query ?sems)) →
 (PP WH Q PRED ?p PTYPE ?pt SEM ?sempp)
 (S INV + SEM ?sems GAP (PP PRED ?p PTYPE ?pt SEM ?sempp))

要正确处理 “where” 这样的疑问词,需要下述规则:

(PP PRED ?pd PTYPE ?pt SEM ?sem) →
 (PP-WRD PRED ?pd PTYPE ?pt SEM ?sem)

特殊疑问词需要两个词条,分别对应一个 PRED 值:

(PP-WRD PTYPE {LOC MOT} PRED – VAR ?v
 SEM <WH ?v (LOC1 ?v)>)

(PP PRED + VAR ?v SEM (λ x (AT-LOC x <WH ?v (LOC1 v)>)))

这些规则对现有语法进行扩展,这样可以解决很多此类问题。图 9.12 给出了疑问句“Where did Jill go?”的句法分析树的一部分。

需要注意的是,要想处理以 +PRED 介词短语开头的疑问句,首先要找到一种解决缺位传播相关问题的方法,这个问题最早在第 5 章提到过。具体来说,规则 VP → VP PP 通常只是把

GAP 传递给 VP 子成分,即非词法中心语。因此,我们好像无法构造一个修饰动词短语的空 PP。对于句法规则而言这也是一个问题,而这一层的任何解决办法也都能用于语义级。

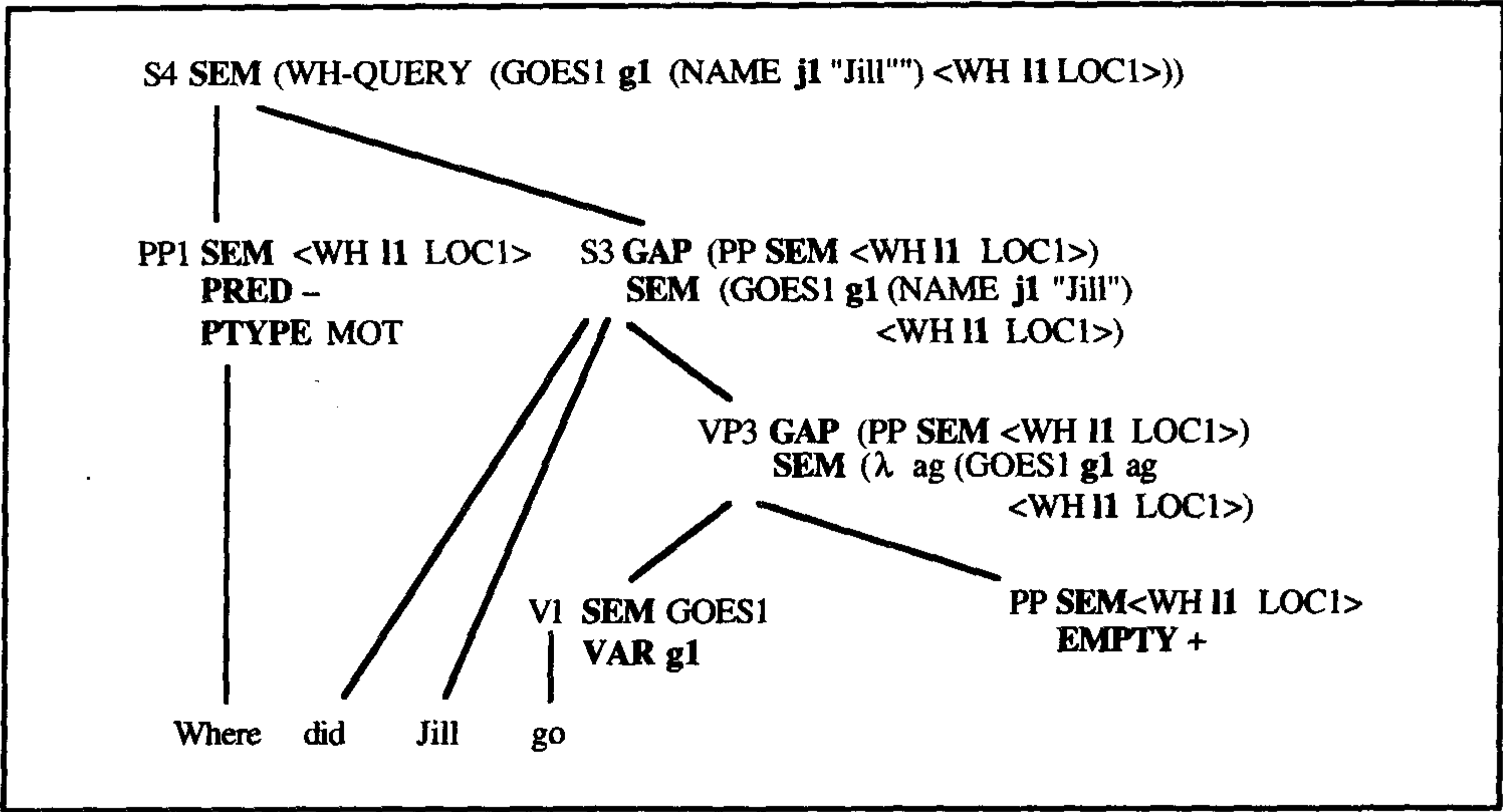


图 9.12 “Where did Jill go?”的句法分析树

9.6 特征合一的语义解释

目前,我们用 λ 表达式和 λ 归约改进了语义解释,因为这提供了一个说明框架和语义解释的很好的分析技术。但是,很多系统并没有显式地应用 λ 表达式,而是直接用特征值和变量来进行语义解释。其基本思想是在参数位置上引入新的特征,而这些位置用 λ 归约在早期就可以表示出来。例如,要替换语法 9.3 的规则 1,即:

$$(S\ SEM\ (?semvp\ ?semnp)) \rightarrow (NP\ SEM\ ?semnp)\ (VP\ SEM\ ?semvp)$$

我们引入了一个新特征 SUBJ,该规则变为:

$$(S\ SEM\ ?semvp) \rightarrow (NP\ SEM\ ?semnp)\ (VP\ SUBJ\ ?semnp\ SEM\ ?semvp)$$

主语的 SEM 作为 SUBJ 特征被传递给 VP 成分,而 VP 的 SEM 等式则将主语插入到恰当的位置上。要处理这种情况,我们引入语法 9.3 的规则 3 的新形式,即:

$$(VP\ VAR\ ?v\ SUBJ\ ?semsubj\ SEM\ (?semv\ ?v\ ?semsubj\ ?semnp)) \rightarrow (V[_none]\ SEM\ ?semv)\ (NP\ SEM\ ?semnp)$$

图 9.13 给出用这条规则构造句子“Jill saw the dog”的 SEM 的方法。将它和图 9.5 中用语法 9.3 构造的分析进行比较,我们发现,两者的不同主要表现在对 VP 的处理上。这里,插入主语的 SEM 是完整的命题;而在前面的例子中,SEM 是一个 λ 表达式,该表达式在后面用于构造 S 的主语。

语法 9.14 是用这种技术重新表示语法 9.3 之后的形式。除了改变了规则 1,2,3 之外,我们对规则 6 和规则 7 也做了修改。规则 7 用 VAR 值构造了一个完整的命题(和以前的语法中的一元谓词相对应),我们对规则 6 进行了适当改变以便和规则 7 的变化相对应。

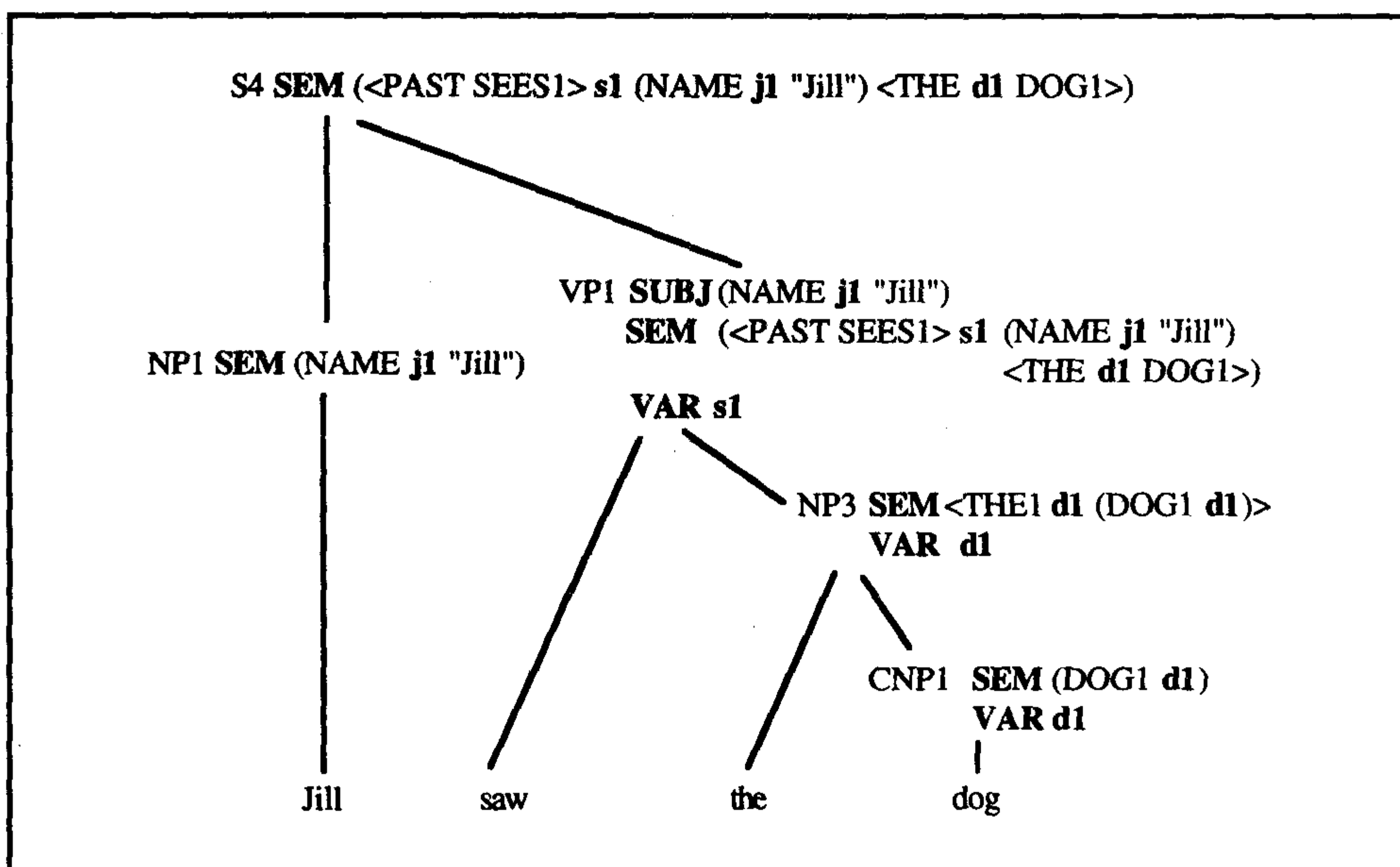


图 9.13 “Jill saw the dog”使用 SUBJ 特征的分析树

1. (S SEM ?semvp) →
(NP SEM ?semsubj) (VP SUBJ ?semsubj SEM ?semvp)
 2. (VP VAR ?v SUBJ ?semsubj SEM (?semv ?v ?semsubj)) →
(V[_none] SEM ?semv)
 3. (VP VAR ?v SUBJ ?semsubj SEM (?semv ?v ?semsubj ?semnp)) →
(V[_np] SEM ?semv) (NP SEM ?semnp)
 4. (NP VAR ?v SEM (PRO ?v ?sempro)) → (PRO SEM ?sempro)
 5. (NP VAR ?v SEM (NAME ?v ?semname)) → (NAME SEM ?semname)
 6. (NP VAR ?v SEM <?semart ?v ?semcnp>) →
(ART SEM ?semart) (CNP SEM ?semcnp)
 7. (CNP VAR ?v SEM (?semn ?v)) → (N SEM ?semn)
- S, VP, NP, CNP 的中心特征: VAR

语法 9.14 带 SEM 特征的简单语法

这种方法的一个优点是不需要引入新的特殊机制来处理语义解释。具体而言,就是不需要加上一个 λ 归约的步骤。一切都用特征合一起来完成。另一个优点是,这种形式确定的语法是可逆的,因此它既可以分析句子又可以产生句子,这一点我们将在下一节讨论。但不是所有的 λ 表达式都能用这种技术排除。例如,要处理“Sue and Sam saw Jack”中的并列主语短语,动词短语的意义就必须是一个 λ 表达式。如果借助表示 SUBJ 特征的变量将主语插入到 VP 中,那么,该变量就必须能和 Sue 与 Sam 的 SEM 都合一,但这不可能。

框 9.2 完全用特征表示语义

对本章方法更彻底的修改是用一组特征表示整个逻辑形式。例如,不用前面讨论的逻辑形式,句子“Jill saw the dog”的 SEM 是:

```
(PRED SEES1
  VAR s1
  TNS PAST
  AGENT (NAME j1 "Jill")
  THEME (SPEC THE1
    VAR d1
    RESTRICTION (PRED DOG1
      ARG1 d1)))
```

对每个语义槽,这种逻辑形式用附加特征可以很容易地构造出来。例如,为计算这种逻辑形式,我们改写语法 9.14 前三条规则,新规则如下:

1. (S VFORM past SEM ?semvp) →
 (NP SEM ?semnp) (VP TNS past SUBJ ?semnp SEM ?semvp)
2. (VP TNS ?tns SUBJ ?semsubj SEM ?semv) →
 (V[_none] VAR ?v SUBJ ?semsubj TNS ?tns SEM ?semv)
3. (VP TNS ?tns SUBJ ?subj SEM ?semv) →
 (V[_np] VAR ?v SUBJ ?subj OBJVAL ?obj TNS ?tns SEM ?semv)
 (NP SEM ?obj)

需要注意的是,所有参数都向下传递给词法规则,而词法规则负责组合正确的语义形式。例如,“saw”的词条是:

```
(V[_np] SUBJ ?s OBJVAL ?o VAR ?v TNS past
  SEM (PRED SEES1 VAR ?v TNS PAST AGENT ?s THEME ?o))
```

当该词条和规则 3 中的 V 合为一体时,就可以构造适当的 SEM 结构,并使之趋向于我们期望的变量?semv。

框 9.3 关系从句

5.4 节描述的关系从句句法利用了关系从句和特殊疑问句之间很强的对应关系。这种相似性在语义级也延续了下来。关系从句的逻辑形式是一个一元谓词,而引入它们的规则如下:

```
(CNP SEM (λ x (& (?semcnp ?x) (?semrel ?x))) →  
  (CNP SEM ?semcnp) (REL SEM ?semrel))
```

这条规则接受两个一元谓词(一个来自 CNP,而另一个来自 REL),并将二者组合起来得到由这两个谓词结合而成的一个新的一元谓词。

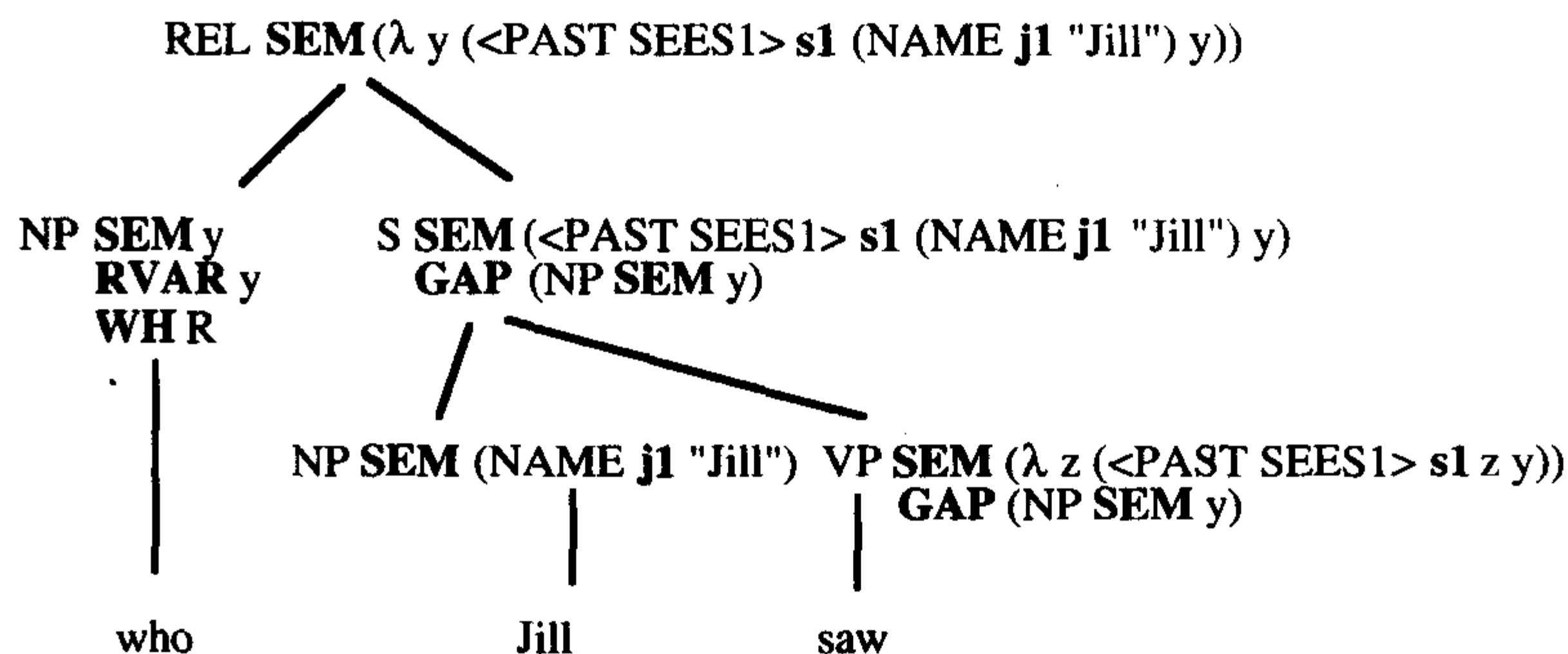
现在,让我们看看其中关于 REL 类的一条规则。这里所需的语言学知识是引入关系从句的特殊疑问词,而关系从句对应于一元谓词的参数。因此,在构造 REL “who Jill saw”的 SEM 时,不能像疑问句那样将 <WH p1(WHO1 p1)> 作为“who”的 SEM,而是要用一个变量来表示。当处理完这个内嵌句之后,再将 λ 包括在内,从而得到关系从句的 SEM,对于这个例子就是:

$(\lambda y (<\text{PAST SEES1}> (\text{NAME } j1 \text{ "Jill"}) y))$

在下面的规则中,称为 RVAR 的特征标识了特殊疑问词变量。

$(\text{REL SEM } (\lambda ?v ?sems)) \rightarrow$
 $(\text{NP WHR RVAR } ?v \text{ AGR } ?a \text{ SEM } ?semnp)$
 $(S [fin] \text{ INV - GAP } (\text{NP AGR } ?a \text{ SEM } ?semnp) \text{ SEM } ?sems)$

下面的分析树片段给出了该关系从句的推导过程。



9.7 用逻辑形式生成句子

直观上,只要你有了一个可供分析的语法,那么将它反过来使用,生成句子就是一件易如反掌的事。句子生成器的输入是语法成分及其作为逻辑形式的 SEM 特征,这样,就能借助语法将这个成分分解成一系列意义正确的词法成分。但是,并非所有的语法都可逆。事实上,由于使用了 λ 归约,语法 9.3 是不可逆的。要弄清原因,我们不妨分析一个例子。假设你想生成一个有如下意义的句子:

$(<\text{PAST SEES1}> s1 (\text{NAME } j1 \text{ "Jill"}) <\text{THE } d1 (\text{DOG1 } d1)>)$

语法 9.3 只有一个 S 规则,如果你想将规则 1 中的 SEM 值同这个逻辑形式合一,不要指望能成功。模式($?semvp ?semnp$)可匹配任何由一元谓词和一个参数构造的命题,但这里所指的逻辑形式是一个拥有三个参数的命题。问题在于,这里用 λ 归约对原逻辑形式进行了转化,即:

$((\lambda a (<\text{PAST SEES1}> s1 a <\text{THE } d1 (\text{DOG1 } d1)>)) (\text{NAME } j1 \text{ "Jill"}))$

λ 归约有一个逆操作,称为 λ 提取,用来寻找匹配的结果。但这样也有问题,现在逻辑形式有三种可能的 λ 提取,即:

$(\lambda e (<\text{PAST SEES1}> e (\text{NAME } j1 \text{ "Jill"}) <\text{THE } d1 (\text{DOG1 } d1)>))$
 $(\lambda a (<\text{PAST SEES1}> s1 a <\text{THE } d1 (\text{DOG1 } d1)>))$
 $(\lambda o (<\text{PAST SEES1}> s1 (\text{NAME } j1 \text{ "Jill"}) o))$

规则 1 中没有任何信息提示哪个才是正确的提取,但实际上只有第二种才能产生正确的句子。

另一方面,使用特征的方法是可逆的,比如语法 9.14,因为该方法在特征中保留了判断构造逻辑形式所需的信息。对于这个问题,用例子很容易解释。

在很多方法中,句法分析和句法实现是相似的过程。两者均可视为构造一个句法树的过程。句法分析程序以词为起点,尝试寻找一个能表示这些词的树,由此得到这句话的逻辑形式。而句法生成器以逻辑形式作为起点,尝试寻找一个能表示该逻辑形式的树,由此得到实现该形式的词。两者的相似表明标准的句法分析算法对于句法实现也适用。举个例子,假设用标准的自顶向下算法,根据是否与期望的 SEM 结构匹配,从语法中挑选规则,句子中的词就自左向右地选择出来了。问题在于,这种方法的效率可能非常低。例如,假设用语法 9.14 来实现带 SEM 的 S:

```
(<PAST SEES1> s1 (NAME j1 "Jill") <THE d1 (DOG1 d1)>)
```

规则 1 基本上与之匹配,因为它的 SEM 特征有一个变量,它的值为?semp。因为没有其他 S 规则,因此这条规则就被实例化。现在,让我们尝试用标准的自顶向下算法来生成这两个子成分,即:

```
(NP SEM ?semsubj)
(VP SUBJ ?semsubj
 SEM (<PAST SEES1> s1 (NAME j1 "Jill") <THE d1 (DOG1 d1)>))
```

但是,NP 的 SEM 不受此限。句法生成器要继续进行上面的过程,必须随机生成名词短语,并且利用每个名词短语来尝试实现 VP,向前回溯直到找到合适的组合。遗憾的是,如果语法规模很大,该算法可能会陷入无限循环中。显然,这种策略不可行。

避免这种问题的一个方法是按不同的顺序来扩充成分。例如,选择哪个词做主语,取决于动词及动词短语的结构,比如是主动语态还是被动语态,等等。因此,先扩充动词短语,然后再生成适当的主语,可能是一种比较有效的做法。事实上,一种比较好的策略是,先扩充中心成分,然后再填入其他成分。图 9.15 给出了一种简单的实现算法。这种算法对一个成分列表进行操作,它和第 3 章介绍的基本自顶向下句法分析器很相似。其过程是不断改写列表中的成分,直到列表全部由词法成分组成,表中的每一点都可生成一个词。

初始化:令 L 为包含你想要生成的成分列表。
重复以下步骤直到 L 不再包含非词法成分:
1. 如果 L 包含一个非词法中心成分 C,
2. 那么,用语法中的某条规则改写 C。C 中任何需要改写的变量在整个列表中都应实例化。
3. 否则,优先选择 SEM 特征必不可少的非词法成分。如果存在这样的非词法成分 C,就选择它。用语法中的一条规则改写 C。C 中所有需要改写的变量应全部实例化。

图 9.15 中心成分驱动的实现算法

推广这个算法很容易,只须利用特定语法,借助回溯技术,可以遍历所有可能的实现方式。这种方法有效的原因是,首先扩展中心成分,算法可以很快到达词法一级,并选择最能影响整个句子结构的词。一旦选好中心词,成分中剩下的结构基本上都可以判断出来了。

让我们分析用语法 9.14 实现的算法,其初始输入是:

```
(S SEM (<PAST SEES1> s1 (NAME j1 "Jill") <THE d1 (DOG1 d1)>))
```

我们基于语法中的规则 1 改写了 S 成分,生成如下的成分列表:


```
(NP SEM ?semsubj)
(VP SUBJ ?semsubj
  SEM (<PAST SEES1> s1 (NAME j1 "Jill") <THE d1 (DOG1 d1)>))
```

非词法中心成分以斜体字表示。接下来对 VP 进行扩展。只有规则 3 符合 SEM 结构。根据匹配的结果,以下变量必然会出现:

```
?semv ← <PAST SEES1>
?v ← s1
?semsubj ← (NAME j1 "Jill")
?semnp ← <THE d1 (DOG1 d1)>
```

在改写 VP 之后,得到成分列表如下。实例化列表中的所有变量,得到:

```
(NP SEM (NAME j1 "Jill"))
(V[_np] SEM <PAST SEES1>)
(NP SEM <THE d1 (DOG1 d1)>)
```

由于没有非词法中心成分,算法现在先挑选出含有必不可少的 SEM 的非词法成分,比如第一个 NP。只有规则 5 与之匹配,这样就得到词法成分(NAME SEM "Jill"),并输出成分列表:

```
(NAME SEM "Jill")
(V[_np] SEM <PAST SEES1>)
(NP SEM <THE d1 (DOG1 d1)>)
```

接下来再挑选剩下的 NP。规则 6 与之匹配,这样就生成子成分(ART SEM THE)和(CNP SEM DOG1),并输出成分列表:

```
(NAME SEM "Jill")
(V[_np] SEM <PAST SEES1>)
(ART SEM THE)
(CNP SEM DOG1)
```

接下来选择 CNP 成分,并利用 SEM DOG1 将其改写为普通名词,这样算法就结束了。现在的成分列表是一串词类:

```
(NAME SEM "Jill")
(V[_np] SEM <PAST SEES1>)
(ART SEM THE)
(N SEM DOG1)
```

利用词法成分来生成句子“Jill saw the dog”还是很简单的。

本例用到的语法很简单,因此,句子只有一种实现方法。如果用更大规模的语法,会有更多的形式可供挑选。举例来说,如果只确定了 SEM 特征,那么句子生成器就会随机选择可以和动词搭配的主动句或被动句。对于其他情况,实现生成器可能还要在不同的次语类结构之中进行挑选。例如,同样的逻辑形式既可以用“Jill gave the dog to Jack”来实现,也可以用“Jill gave Jack the dog”来实现。根据词义的数量多少以及不同的词是否有公共意义这两个条件,句子生成器可能会从不同的逻辑形式词汇实现中随机挑选一个。例如,某个逻辑形式既可以用“Jill gave the money to the Humane Society”(Jill 把钱给了慈善社团)实现,也可以用“Jack donated the money to the Humane Society”(Jack 把钱捐给了慈善社团)实现。

这些不同的形式在上下文中有不同的效果,但这些差异在逻辑形式中无法体现。如果一定要实现为某种形式,需要指定除 SEM 特征之外的其他特征。例如,可能需要为主动语态设定 VOICE 特征,这样才肯定能得到主动语态句。

9.8 小结

本章引入新的 SEM 和 VAR 特征来表示语义信息,提出了语义解释的逐条规则模型。这种表示形式主要采用 λ 表达式,其中的每个成分都有规范的语义形式。通过对原有方法的两个小扩展,我们引入了惟一的 VAR 值,并且提供 λ 归约操作。于是,在构造句法结构的过程中,任何 chart 分析算法都能计算出逻辑形式。接着,我们讨论了这种方法的一种变体形式,即采用附加特征而非 λ 表达式。这种表示法能产生可逆的语法,而这种语法既可以用来进行语义解释,也可用来进行语义实现。中心成分驱动的句子实现算法可根据特定的逻辑形式生成句子。其过程是首先扩充中心成分,得到整个句子的结构,然后再填充其他成分的细节。

9.9 相关工作与深入阅读材料

过去 20 年里,很多与语法有关的工作都深受 Montague(1974)的影响。Montague 提出,自然语言的语义可以像定义形式语言语义一样按组合规则来定义。他引入了 7 个关键概念,对我们而言,最有影响的莫过于引入了 λ 演算。Chierchia 和 McConnell-Ginet(1990)是一篇介绍 Montague 式语义的优秀文献。Dowty 等(1981)和 Partee 等(1993)做了更详细的阐述。

本章描述的组合语义解释器源自多本著作,其中主要参考了 Rosenschein 和 Shieber(1982), Schubert 和 Pelletier(1982)以及 GPSG(Gazdar 等,1985)。组合语义解释的其他重要参考资料包括 Hirst(1987), McCord(1986), Saint-Dizier(1985)和 Hwang 和 Schubert(1993b)。Pereira 和 Shieber(1987)以及 Moore(1989)也讨论了 λ 归约中使用特征合一的问题。关于这项技术的一个很好的例子可以参考 Alshawi(1992)。

自然语言生成方面有很多著作,其中需要解决两个问题:首先确定要说哪些内容,然后用句子来表达该内容。9.7 节中的算法只解决了实现部分,假设逻辑形式已经确定。而在确定逻辑形式之前,在生成句子中面临的许多困难问题(比如判定内容、判定主谓词以及指代方式)必须要先解决。关于句子生成方面比较好的综述文章,请参考 McDonald 发表在 Shapiro(1992)标题为 *Natural Language Generation* 的文章。关于近期比较好的论文集,请参考 *Computational Intelligence* 7,4(1991)专刊以及 Dale 等(1990)的论文。其中介绍的中心成分驱动算法是 Shieber 等(1990)所描述算法的简化版。

9.10 习题

1. 【易】用 λ 归约简化以下表达式:

$$\begin{aligned} & ((\lambda x (P x)) A) \\ & ((\lambda x (x A)) (\lambda y (Q y))) \\ & ((\lambda x ((\lambda y (P y)) x)) A) \end{aligned}$$

2. 【易】利用本章定义的解释规则,并且定义所需的其他规则,给出句子“The man gave the apple to Bill”的详细解释过程。具体来说,要分析每个成分并给出其 SEM 特征。
3. 【中】本题主要是用指定的语义解释规则来分析动词“roll”的不同语义,即对于下列例句:

We rolled the log into the river.

The log rolled by the house.

The cook rolled the pastry with a large jar.

The ball rolled around the room.

We rolled the piano to the house on a dolly.

- 确定这些句子中动词“roll”的不同语义,并给出每个意义的非正式定义(可以使用任何一部词典)。尝试说明,不同的意义怎样使各个句子产生不同的结果。指出动词“roll”的词条。
 - 列出本章中用来处理这些句子的 VP 规则,并添加所需的新规则。
 - 根据练习 3b 中列出的规则,假设对于 NP 存在恰当的规则,用每个成分的逻辑形式绘出分析树。
4. 【中】绘出一棵句法分析树,给出以下疑问句中成分的语义解释。给出词条,使这些词的词条能够表示句子中未在本章中定义的词的 SEM 特征,对本章虽未指定但分析又需要的附加规则,给出其定义。

Who saw the dog?

Who did John give the book to?

5. 【中】思考以下规则,它们用来解决那些可接不定式补语的动词。

- $(VP \text{ VAR } ?v \text{ SEM } (\lambda \text{ ag } (?semv \text{ ag } (?semvp \text{ ag })))) \rightarrow$
 $(V \text{ SUBCAT } _vp\text{-}inf \text{ SEM } ?semv)$
 $(VP \text{ VFORM } inf \text{ SEM } ?semvp)$
- $(VP \text{ VAR } ?v \text{ SEM } (\lambda \text{ ag } (?semv \text{ ag } (?semvp \text{ ?semnp })))) \rightarrow$
 $(V \text{ SUBCAT } _np_vp\text{-}inf \text{ SEM } ?semv)$
 $(NP \text{ SEM } ?semnp)$
 $(VP \text{ VFORM } inf \text{ SEM } ?semvp)$
- $(VP \text{ VFORM } inf \text{ ?v SEM } ?semvp) \rightarrow$
 $TO (VP \text{ VFORM } base \text{ SEM } ?semvp)$

注意,根据构造的语义结构,我们能识别内嵌 VP 的主语。对于_{vp-inf}动词,规则 1 表示的是,该动词的隐含主语就是句子主语;而对于_{np_vp-inf}动词,规则 2 表示的是,该动词的隐含主语是主动词的宾语。根据这些规则及本章所描述的规则,你认为哪种逻辑形式是下列句子生成的?

Jill wants to cry.

Jill wants Jack to cry.

Who does Jill want to see?

6. 【中】思考下面的句子：

John is happy.

John is in the corner.

John is a man with a problem.

每句话都给出 John 的某种属性。这表明短语“happy”，“in the corner”和“a man with a problem”应该有相似的句法结构，可能会是一元谓词。在某些方法中，PRED 特征被扩展到除 PP 之外的其他形式，可以用它来刻画这三种情况的相似之处。使用这种方法，当这三个短语都有 + PRED 特征时，在这种上下文中它们的语义解释是什么？写出正确处理每个句子所需的规则，并添加到本章所描述的语法中。

第 10 章 歧义消解

上一章讨论的技术并未解决歧义消解问题。例如,“bridge”这个词作为名词至少有 4 种不同的意义,包括跨越水体的高架结构、一种牌类比赛、牙科设备以及一种表示沟通的抽象概念。要区分这些不同的解读,需要引入不同的意义,比如 STRUCTURE1, GAME27, DENTAL-DEV37 和 CONNECT12。任何一个句子都只接受这些意义中的某一个。例如,句子“The bridge crosses the river”只能用 STRUCTURE1 这个意义来解释。但是,上一节中的语义解释机制却能接受每一种解读,因为上一节中没有提供方法来定义哪种意义组合才是符合要求的。本章将介绍一些方法和技术,可用来解决歧义问题。

10.1 节讨论用选择限制和类型层次来确定可能的语义组合,并提出了限制传播算法来排除那些不可能的语义。10.2 节则在分析过程中用这些技术筛选出语义候选成分。10.3 节讨论语义网络表示方法,对简单类型层次进行了泛化。10.4 节讨论用词语搭配进行语义识别的一些统计方法,而 10.5 节将选择限制的概念推广为统计选择。10.6 节讨论基于结构的方法和基于统计的方法相结合出现的一般问题。

10.1 选择限制

词义可以通过不同方法关联起来,这些方法基于它们所描述对象的类别。某些意义是不相交的,也就是说,没有可以同时属于两个类的对象。DOG1(dog 的标准意义)和 CAT1(cat 的标准意义)是不相交意义。而某些意义则可以是别的意义的子类。例如,DOG1 类既可以是 MAMMAL1(哺乳动物)类的子类,也可以是 PET1 类(家庭宠物)的子类。还有一些意义有重叠,比如 MAMMAL1 和 PET1。这些常识在歧义消除方面会起到一定的作用。

这些子集之间的关系定义了一种语义上的抽象层次体系。这种关系非常重要,因为它提供一种简洁直观的方式根据更一般的对象类来表示约束关系。例如,像“purple”(紫色)这样的形容词只在修饰一个实体时才有意义,比如一座房子或一件衣服。谈论像“purple idea”(紫色的想法)或“purple event”(紫色的事件)这样的短语是没有意义的。这种现象可以用一个条件来表述,即“purple”必须修饰一个实体。在这个条件下,所有实体的意义都可以接受。另一方面,修饰语“precise”(准确的)只在修饰一种思想或一个动作(或者人类行为的一般特征)时才有意义,而修饰语“unfortunate”只在修饰某些事件或情况时才有意义。图 10.1 给出了类型层次体系最上层的一部分,这一体系对自然语言非常有用。但图中给出的不是全部,而前面几章中提到的本体的所有对象都要在这个层次中找到相应的位置。还需要注意的是,层次体系不一定是树结构;也就是说,意义可能会有多个父类。例如,MALE 和 FEMALE 这两个类应该和 ANIMATE/VEGETATIVE 在同一层级,并且和 LIVING 这个单独子树的所有子类相连。在这种情况下,MALE PERSON 这个意义既是 PERSON 的子类,又是 MALE 的子类。

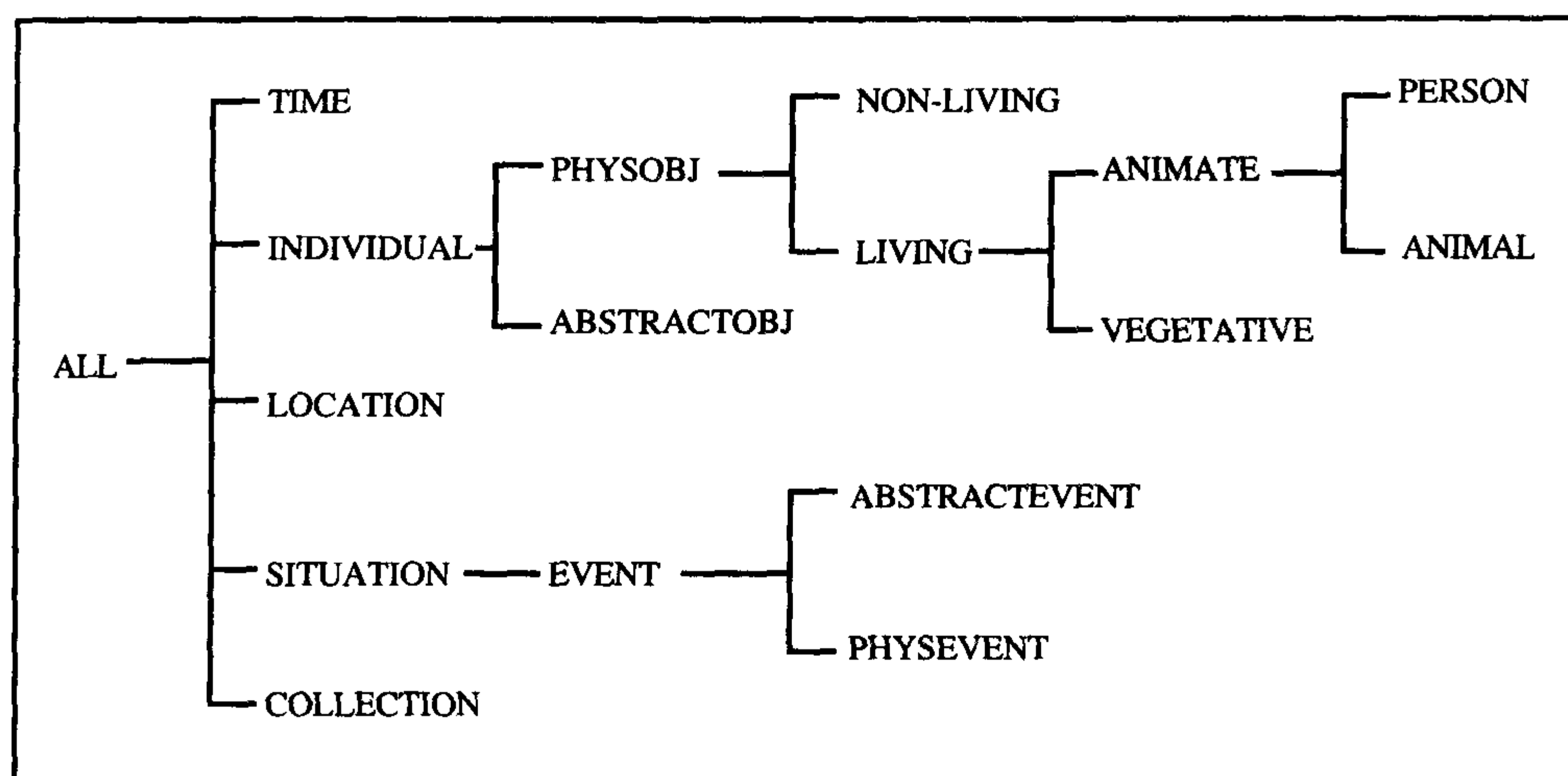


图 10.1 词义层次体系

根据这样一个层次体系,可以开始探讨谓词对其参数所加的约束。分析动词“read”,它有两个主要参数:AGENT,必须是具有阅读能力的对象(例如,类型 PERSON 中的某个对象);THEME,必须是某个包含文字的对象(例如书、报纸、标签、路牌,等等)。要正确处理此类动词,我们先引入一个新类型,记做 TEXTOBJ,位于 NONLIVING 的下层,它是 BOOK1, ARTICLE/TEXT 等的超集。根据这些限制可以为句子中的词选出恰当的意义。比如,名词“dishwasher”有两种词义,既可以指机器(DISHWASH/MACH1)也可以指工人(DISHWASH/PERS);名词“article”既可以指文章(ARTICLE/TEXT)也可以指演讲的一部分(ARTICLE1)。这些词义放入类型层次体系中,如图 10.2 所示。由于这两个词都有歧义,所以“The dishwasher read the article”这句话有可能出现四种不同的语义,每种代表一个不同的意义组合。但其中只有一种解读有意义,即:

```
(READS1 [AGENT <THE d1 DISHWASH/PERS>]
        [THEME <THE p1 ARTICLE/TEXT>])
```

用一种称为选择限制的方法,语义解释器可以得到消歧之后的结果。这种方法能确定可共现(co-occur)意义的合法组合,并排除由语法分析器构造的那些前后不一致的形式。

要将这项技术和语义解释器结合起来,需要一种可以将逻辑形式内在的类型信息提取出来的机制。通过分析一元和二元谓词逻辑形式中的每个词,可以得到这种机制。例如,在应用选择限制之前,考察“The dishwasher read the article”的逻辑形式:

```
(READS1 r1 [AGENT <THE d1 {DISHWASH/MACH1
                           DISHWASH/PERS}>]
            [THEME <THE p1 {ARTICLE/TEXT ARTICLE1}>])
```

分解这些符号,找到一元和二元关系如下:

```
(READS1 r1)
({DISHWASH/MACH1 DISHWASH/PERS} d1)
({ARTICLE/TEXT ARTICLE1} p1)
(AGENT r1 d1)
(THEME r1 p1)
```

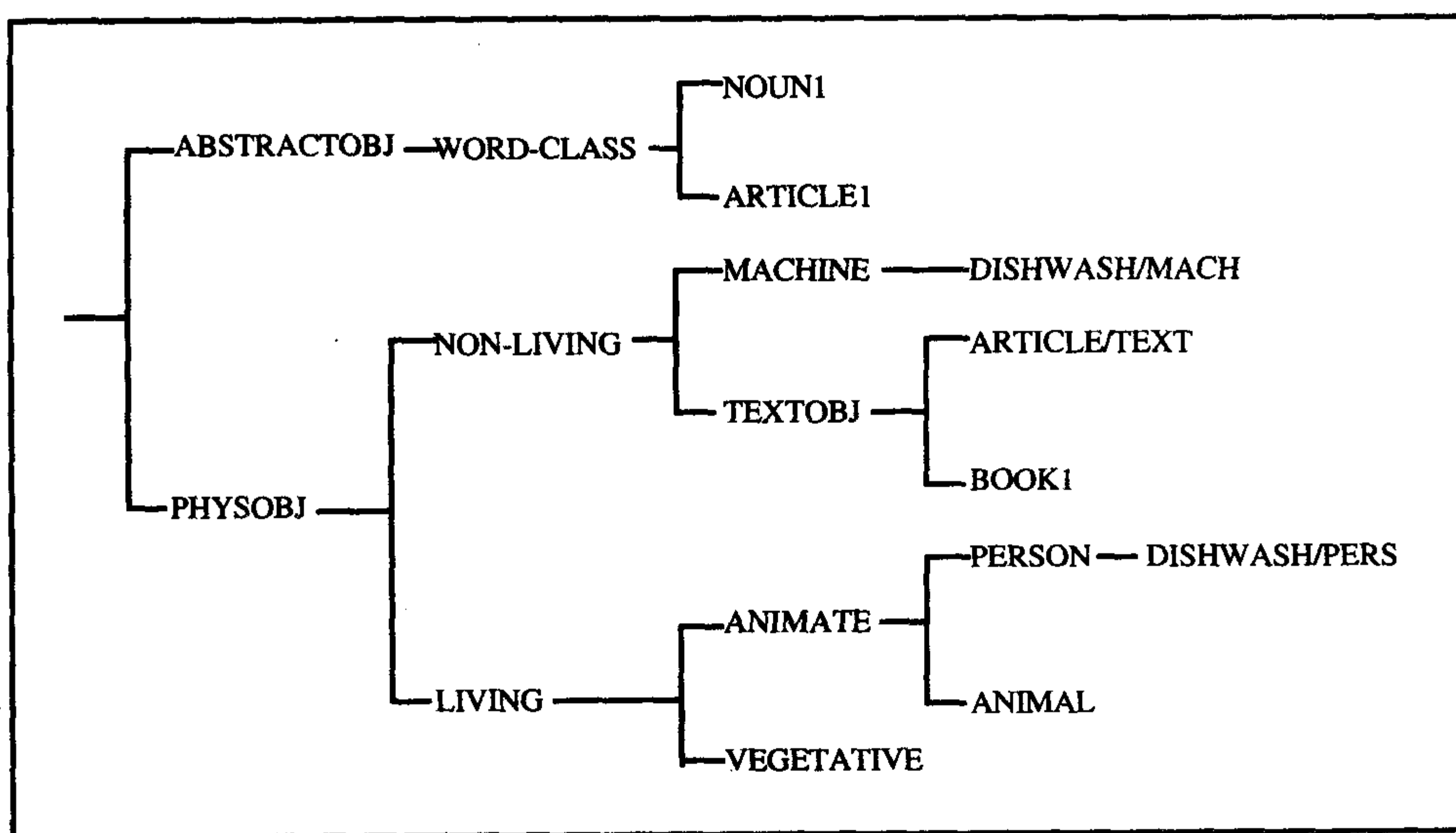


图 10.2 词义层次体系片段

查找可能组合可以看做约束满足问题。在这个例子中,约束条件包括一元谓词间的语义关系(子集、交集,等等)(如图 10.1 和图 10.2 所示的类型层次体系表示了这种关系)和二元关系中各参数语义类型的选择限制。READS1 选择限制如下所示:

(AGENT READS1 PERSON)——AGENT 必须是人。

(THEME READS1 TEXTOBJ)——THEME 必须是一个 TEXTOBJ 对象。

假设这些就是词义 READS1 惟一可接受的关系,那么可以排除前面提到的逻辑形式中的几个可能词项。要让 (AGENT *r1* *d1*) 合法, *d1* 必须是人。那么, *d1* 的一元约束就可以由 (*{* DISHWASH/MACH1 DISHWASH/PERS *}* *d1*) 简化为 (DISHWASH/PERS *d1*)。与此相似, *p1* 的解释也可以简化为 (ARTICLE/TEXT *p1*)。只要将这些约束传到逻辑形式中,我们最终可以如愿以偿地得到一个无歧义的解读。

当然,在很多情况下问题要复杂得多。例如,动词“read”有两种意义:READS1 和 READS2,后者表示理解某人的意图,比如“Jill can read John’s mind”。READS2 的选择限制是:

(AGENT READS2 PERSON)

(THEME READS2 MENTAL-STATE)

借助这个意义,“The dishwasher read the article”(洗碗工读文章)的初始逻辑形式可表示为:

(*{* READS1 READS2 *}* *r1*)

[AGENT <THE *d1* { DISHWASH/MACH1 DISHWASH/PERS }>]

[THEME <THE *p1* { ARTICLE/TEXT ARTICLE1 }>]

多出来的意义并没有影响最终结果,因为 READS2 需要一个 MENTAL-STATE 作为 THEME,而“article”的两个词义都不是精神状态的子类。因此,这里没有符合 READS2 词义的解释组合。

我们还需要将这项技术扩展到专有名称、代词和形容词。这里,需要假设我们知道专有名称所指向的类型信息。例如,“John”的类型可能是 MALE,即 MALE 型的有生命对象。未知名称默认具有 INDIVIDUAL 属性。代词短语的类型由代词的词义决定,因此, SHE1 应该是 FEMALE 的子类,而 IT1 应该属于非 PERSON 类。形容词和动词相似,因为它们对其参数施加限

制。通过一种方法,对每个形容词修饰语引入状态变量表达式和一种新的主题关系 MOD,从而可以处理这类修饰语。例如,对于短语“happy dishwasher”(快乐的洗碗工),我们不用谓词参数的形式(HAPPY1 d1),而是使用一元关系(HAPPY-STATE h1)和二元关系(MOD h1 d1)。通过这种方法,可以像动词那样处理形容词。于是,从“The happy dishwasher read the paper”中推导出的关系集就是:

```
(READS1 r1)
({DISHWASH/MACH1 DISHWASH/PERS} d1)
({ARTICLE/TEXT ARTICLE1} p1)
(HAPPY-STATE h1)
(AGENT r1 d1)
(THEME r1 p1)
(MOD h1 d1)
```

形容词“happy”的选择限制是:

(MOD HAPPY-STATE ANIMATE)——HAPPY-STATE 必须修饰有生命的对象。

逻辑形式一产生,就可以用选择限制检查它所包含的一元和二元的语义关系集。如果没有能满足全部约束条件的解释,就认为解释异常,并丢弃该成分。反之,通过排除不可能的意义,能够得到简化的逻辑形式,并且可以作为该成分的 SEM。在本节剩下的部分,我们会更详细地探讨约束满足算法。

算法如图 10.3 所示。为实现该算法,需要词义匹配过程。如果两个类型没有相交部分,则匹配失败。否则,该过程将返回两者相交的类型。如果其中一个是另一个的子类,则该子类就是结果。已知如图 10.2 所示的层次体系,Match(PERSON, DISHWASH/PERS)的结果就是 DISHWASH/PERS。在其他情况下,两种类型相交,返回两种类型共有子类的词义。例如,通过适当的定义,Match(MALE, PERSON)的结果就是 MALE-PERSON1。

初始化步骤

给变量 i 的可能词义列表赋予 types(variable_i)。

迭代步骤

对每个二元关系(rel variable₁ variable₂)进行如下迭代:

1. 对 types(variable₁)的每个意义 sense₁:

a. 找到 sense₂ 与 types(variable₂)的某个词义相交的所有选择限制(rel sense₁ sense₂)。

b. 如果没找到,从 types(variable₁)中删除该意义 sense₁。

2. 如果 types(variable₂)中的某个意义与步骤 1 中的至少一个约束不符,就将该意义删除。

终止步骤

如果上一步迭代过程中变量类型还有变化,则继续执行一次迭代。

否则,如果对任意 i, types(variable_i)都为空,则失败。

图 10.3 一个简单的约束满足算法

这个算法一开始列出每个篇章变量的所有可能词义,然后对其二元关系进行迭代。如果某个一元约束不满足一个二元约束条件,就将其排除。如果算法对原来的结果有改动,就重新检查一遍二元约束条件。这个过程一直进行下去,直到不再有改动为止。

举个例子,考虑用这个算法分析句子“The dishwasher read the article”(洗碗工读文章)。初始化步骤产生如下类型:

```
types(r1) =  READS1, READS2
types(p1) =  ARTICLE/TEXT, ARTICLE1
types(d1) =  DISHWASH/PERS, DISHWASH/MACH1
```

这个问题中的二元关系是(AGENT **r1** **d1**)和(THEME **r1** **p1**)。第一步迭代如下:

对(AGENT **r1** **d1**),我们对 **r1** 的意义进行迭代:

READS1——我们找到选择限制 (AGENT READS1 PERSON), PERSON 只和 DISHWASH/PERS 匹配(结果为 DISHWASH/PERS)。

READS2——我们找到选择限制 (AGENT READS2 PERSON), PERSON 只和 DISHWASH/PERS 匹配(结果为 DISHWASH/PERS)。

这样,类型(**d1**)就变成 (DISHWASH/PERS);也就是说,由于不满足任何一个二元约束条件,DISHWASH/MACH1 被删除了。

对于(THEME **r1** **p1**),我们对 **r1** 的意义进行迭代:

READS1——我们找到选择限制(THEME READS1 TEXTOBJ),因为 TEXTOBJ 可以和 ARTICLE/TEXT 匹配(结果为 ARTICLE/TEXT)。

READS2——我们没发现相匹配的选择限制;也就是说,无法满足(THEME READS2 MENTAL-STATE)。

这样,类型(**r1**)就变成(READS1);也就是说,READS2 被排除了,类型(**d1**)变成了(ARTICLE/TEXT),因为 ARTICLE1 被排除了。

由于发生了变化,要再次进行迭代。第二次迭代过程如下:

对(AGENT **r1** **d1**),**r1** 只剩下一个词义:

READS1——我们找到选择限制(AGENT READS1 PERSON)。

对(THEME **r1** **p1**):

READS1——我们找到选择限制(THEME READS2 TEXTOBJ)。

由于这次没有变化,整个过程就结束了。最后得到的类型为:

```
types(r1) =  READS1
types(p1) =  ARTICLE/TEXT
types(d1) =  DISHWASH/PERS
```

这和设想的一样。

对未知对象类型的细化,选择限制同样很有用。例如,除了不用于指人,代词“it”本身对它所指对象的类型并没有多少限制。但是,动词的选择性限制却对它所指对象的类型给出明显的提示。分析句子“He reads it”的处理过程,假定动词取 READS1 这个意义:

```
(READS1 r3 [AGENT (PRO i1 HE1)] [THEME (PRO n1 (IT1 n1))])
```

则对象的一元和二元约束条件为:

```
(READS1 r3), (AGENT r3 i1), (THEME r3 n1), (MALE i1), (IT1 n1)
```

在对意义 READS1 应用选择限制之后,“he”的类型被限定为 MALE-PERSON1 类(MALE 和 PERSON 类的交集),“it”的类型被限定为 TEXTOBJ 类(IT1 和 TEXTOBJ 类的交集)。在上下文相关处理中,遇到需要确定代词所指的情况时,这些额外的信息是非常有用的。为了保存这些信息,可以将其加入逻辑形式中。这样,在应用选择限制后,这个句子的逻辑形式变为:

(READS1 r3 [AGENT (PRO i1 (& (MALE i1) (PERSON i1)))]
[THEME (PRO n1 (& (IT1 n1) (TEXTOBJ n1)))]])

选择限制是一项重要的歧义消解技术,几乎每个计算系统都以某种方式在使用这项技术。但这个方法也招来不少批评,而这些批评意见都值得认真加以研究。同时,我们也应该思考:尽管存在种种问题,为何它在计算系统中仍然发挥着如此巨大的作用? 主要问题在于,语义的好与不好不能简单地以是非标准来判断,相反,它是一个连续的尺度。如果一定要对规范做非此即彼的判断,会错误地排除许多实际上可行的解释。另一种可能是,由于约束非常抽象,以至于我们基本上无法排除任何组合。考虑下面这些句子:

1. I ate the pizza.(我吃了比萨。)
2. I ate the box.(我吃了盒子。)
3. I ate the car.(我吃了轿车。)
4. I ate the thought.(我吃了思想。)

显然,第一句没有任何问题,而第四句则有严重问题。但中间两句呢? 如果需要指明“吃”这个事件的 THEME 角色应满足的选择限制条件,你可能会引入 FOOD1(食物)这样的类别。这样,第一句就可以接受,而第二句就不合乎语法。但显然,第二句的语法并没有错误。虽然不合情理,但我还是可以吃盒子的,所以第二句似乎也是一个好的句子。但是,哪一个类既包含 FOOD 又包含盒子(以及纸、帽子等可吃的东西)? 我们或许会考虑 PHYSOBJ,这样第一、第二、第三句就都可以成立了。但是这样做,我们就无法判断以下事实:相对于那些非食物的物品,尤其是轿车这样的金属物体,人们更愿意吃食物。具体而言,在句子“I ate some chips”中,“chip”这个词即可指食物(薯片),又可以指不是食物的物体(木屑),这样就无法消除歧义。这表明,选择限制对歧义消解的作用是有限的(这个例子中,只能区别出抽象体和实体)。

选择限制同样和命题的上下文密切相关。例如,在否定性上下文中,甚至可以违反这些约束。举个例子,虽然 EATS1 的 AGENT 类型只能是 FOOD1,但句子“I could not eat a car”(我不能吃轿车)是完全正确的。“My car drinks gasoline”(我的轿车喝汽油)使用了隐喻,选择限制显然也不适用于这种情况,它们只是表明这里需要进行隐喻分析。尽管存在这些问题,选择限制在实际系统中还是非常有用的。一个原因是,当系统在受限领域内运行时,诸如“eat box”(吃盒子)这样不常见的解释不会出现,因而选择限制(**THEME EATS1 FOOD**)能很好地发挥作用。当然,随着涵盖范围的扩大,可以处理这些复杂情况时,这种方法就会遇到问题了。

把这个方法和这些问题有机协调起来的一个途径是,将选择限制看做优先选择的手段而非绝对必需的要求。在这种情况下,满足所有选择限制的解釋就优于和限制相冲突的解釋。但如果所有解釋都和某些限制冲突时,我们就优先选择和限制冲突最少的解釋。可以通过以下策略来实现这种方法:对满足选择限制条件程度不同的解釋进行排序,并从中选出和约束冲突最少的一个。

10.2 用选择限制进行语义筛选

至少有两种方法可以将选择限制加到句法分析器中。其中一个顺序模型,它先运行句法分析器,然后检查发现的所有 S 解释。另一个是增量式模型,它检查句法分析器提供的每个候选成分,并将语义不规范的成分剔除。增量式模型的效率远远高于顺序模型。本节用一个简单例子来检验其效率。具体来说,我们先假定词义层次体系是一个简单的树结构。这样,对模型进行一般化并不困难。然而,如果不做这种假设,会使例子更复杂,从而掩盖了基本的问题。

考虑下面的句子,它的句法结构有两种 PP 附着歧义:

He booked a flight to the city for me.

如果语法允许 PP 既可以跟在 VP 之后,又可以跟在 CNP 之后,那么这个句子有 5 种不同的结构:介词短语“to the city”既可以修饰动词“booked”又可以修饰名词“flight”,介词短语“for me”既可以修饰名词“city”,又可以修饰名词“flight”,还可以修饰动词“booked”。有 5 种途径可以将这些可能组成合乎句法的结构。但是,如果考虑到句子的语义,就只存在一种可能的解读:航班到那个城市,而他是为我预定的。这种直观知识是通过动词“book”和名词“flight”以及“city”的选择限制来表示的。

语法 10.4 是一个小型语法,可以处理 PP 附着的各种不同形式。其中,使用了 9.6 节阐述的技术,但不要求使用 λ 归约。这里,只列出了和语义解释相关的部分特征。规则 7 和规则 8 引入了 PP 修饰语。特征 ARGVAR 用于将一个篇章变量传递给 PP,这样可以构造正确的命题。除了这几点,这个语法和语法 9.14 都很相像。

1. (S SEM ?semvp) \rightarrow (NP SEM ?semsubj) (VP SUBJ ?semsubj SEM ?semvp)
 2. (VP SUBJ ?semsubj SEM ?semv) \rightarrow (V[_none] SUBJ ?semsubj SEM ?semv)
 3. (VP SUBJ ?semsubj SEM ?semv) \rightarrow
(V[_np] SUBJ ?semsubj OBJ ?semnp SEM ?semv) (NP SEM ?semnp)
 4. (NP VAR ?v SEM (PRO ?v (?sempro ?v))) \rightarrow (PRO SEM ?sempro)
 5. (NP VAR ?v SEM <?semart ?v ?semcnp>) \rightarrow (ART SEM ?semart) (CNP SEM ?semcnp)
 6. (CNP VAR ?v SEM (?semn ?v)) \rightarrow (N SEM ?semn)
 7. (CNP SEM (& ?semcnp ?sempp)) \rightarrow
(CNP VAR ?v SEM ?semv) (PP PRED + ARGVAR ?v SEM ?sempp)
 8. (VP SEM (& ?semvp ?sempp)) \rightarrow
(VP VAR ?v SEM ?semvp) (PP PRED + ARGVAR ?v SEM ?sempp)
 9. (PP PRED + ARGVAR ?v1 SEM (?semp ?v1 ?semnp)) \rightarrow
(P SEM ?semp) (NP SEM ?semnp)
- S, VP, NP, CNP 的中心特征: VAR

语法 10.4 可处理 PP 附着歧义的小型语法

图 10.5 给出了一些词条以及在词义检查中要用到的词义层次体系。谓词 HE1 和 ME1 分别表示代词 he 和 me 的语义约束,就本例而言,可以近似地将它们放在意义 PERSON 下。和本节相关的选择限制如下:

(AGENT BOOKS1 PERSON1)
 (THEME BOOKS1 FLIGHT1)
 (BENEFICIARY ACTION1 PERSON1)
 (DESTINATION FLIGHT1 CITY1)
 (NEARBY PHYSOBJ PHYSOBJ)
 (NEARBY ACTION PHYSOBJ)

这些限制条件要求订票这个动作的主体必须是人,而论题必须是航班。第三个限制条件表明任意动作和人之间都有受益关系,第四个限制条件表明航班和城市之间存在目的关系。最后两个限制条件表示两个实体之间以及行为和实体之间可以有相邻关系,比如“*He sighed near the boat*”(他在船边叹了口气)。

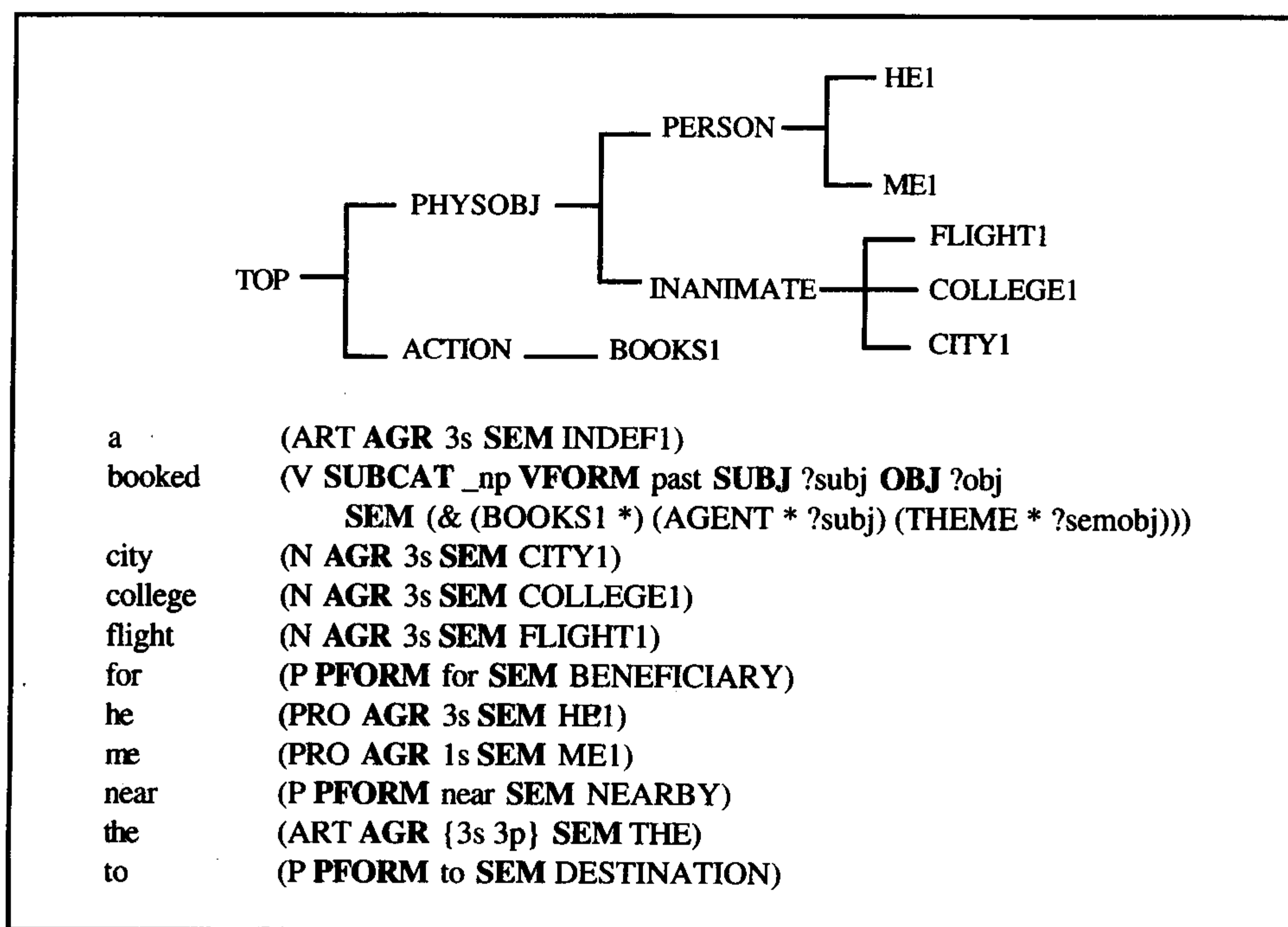


图 10.5 一部小型词典及语义层次体系

以上就是分析本例所需的全部信息。当一个词条加入 chart 时,语义筛选就加入到 chart 分析器中。在添加之前,系统需要先做一次检查,目的是保证当前逻辑形式的关系不违反选择限制。如果 SEM 值已经全部实例化,就可以直接应用上一节给出的方法。如果 SEM 值包含变元,则检查不含变元的 SEM 中的关系。如果 SEM 满足限制,这些成分就按通常的方法加入 chart。如果不满足限制,就将其丢弃。由此,不仅可以从 chart 中排除这些成分,而且这样一来,任何由这些成分构成的更大成分也不再需要句法分析器进行解析了。因此,即便在自底向上的句法分析技术中,这项技术也可以显著提高效率。

举例来说,已知语法 10.4,考虑用自底向上的 chart 句法分析器分析句子“*He booked the flight to the city for me*”。如果不使用语义筛选,句法分析器将找到五种不同的解释,在 chart 中产生 52 个成分。使用语义筛选时,它只找到一种解释,同时只产生 33 个成分。思考由句法分析器产生并用语义筛选排除的第一个成分:


```
(VP SEM (BOOKS1 v258
      [AGENT ?semsubj]
      [THEME <INDEF1 v260 (FLIGHT1 v260)>]
      [DESTINATION <THE v263 (CITY1 v263)>])
  VAR v258
  SUBJ ?semsubj)
```

在构造动词短语“book the flight”和介词短语“to the city”的句法成分时,这个成分由规则 8 产生。但是它和谓词 DESTINATION 的选择限制冲突,即无法将“预订”这个动作和城市关联起来;换句话说,(DESTINATION BOOKS1 CITY1)不满足任何选择限制。具体的分析过程如下所示。变量的一元约束条件为:

```
v258 BOOKS1
v260 FLIGHT1
v263 CITY1
```

根据这些条件,由 SEM 产生的关系,其形式为 (THEME BOOKS1 FLIGHT1) 及 (DESTINATION BOOKS1 CITY1)。后者不满足任何选择限制。这里要注意的是,即使主语不确定,也可以排除这个成分。

在更复杂的句子中,这种节省就显得更加可观。例如,在下面的句子中:

He booked a flight to the city near the college for me.

使用顺序方法的句法分析器将产生 14 种不同的解释和 116 个 chart 成分。而使用增量方法的句法分析器只产生 3 种解释和 63 个 chart 成分。这三种解释的语义看上去都很合理——第一种解释中,城市靠近大学;第二种解释中,飞往城市的航班靠近大学;第三种解释中,订票这个动作的发生地点靠近大学。在没有句子上下文信息的情况下,第一种解读似乎更为自然,但在特定的上下文中,其他两种解读也有可能。由于在本书的第三部分才探讨上下文,所以在此之前,最好还是使用上下文无关技术。

10.3 语义网络

上一节中,我们看到语义类型层次体系对于歧义消解非常重要。本节将这些思想加以推广,提出一种新的词法知识表示方法,称为语义网络。通过属性继承,语义网络简化了词典的构建,同时还提供了词义之间更为丰富的语义关系集,这些都有助于歧义消解。

语义网络是由带标记的链和带标记的节点组成的图。节点表示词义或抽象意义类型,链表示意义间的语义关系。图 10.6 所示的简单抽象层次体系给出了实体类型和抽象对象之间的某些语义关系。s 边表示子类关系。这幅图实际上是前面两节提到的类型层次体系的另一种表示方法。

语义关系的选择限制以边的形式存储在网络结构中。因此,根据图 10.7 中的网络,可以说所有的动作都有一个由有生命的对象充当的施事格。这里,引入了一个新的节点类型,即存在节点,用框表示,代表一个特定的值。在这种情况下,我们将施事格限定为 ANIMATE 类中的对象。还有很多表示法可以表示格取值的其他信息,例如它们是否必不可少,以及它们代表一个填充物还是一组填充物。

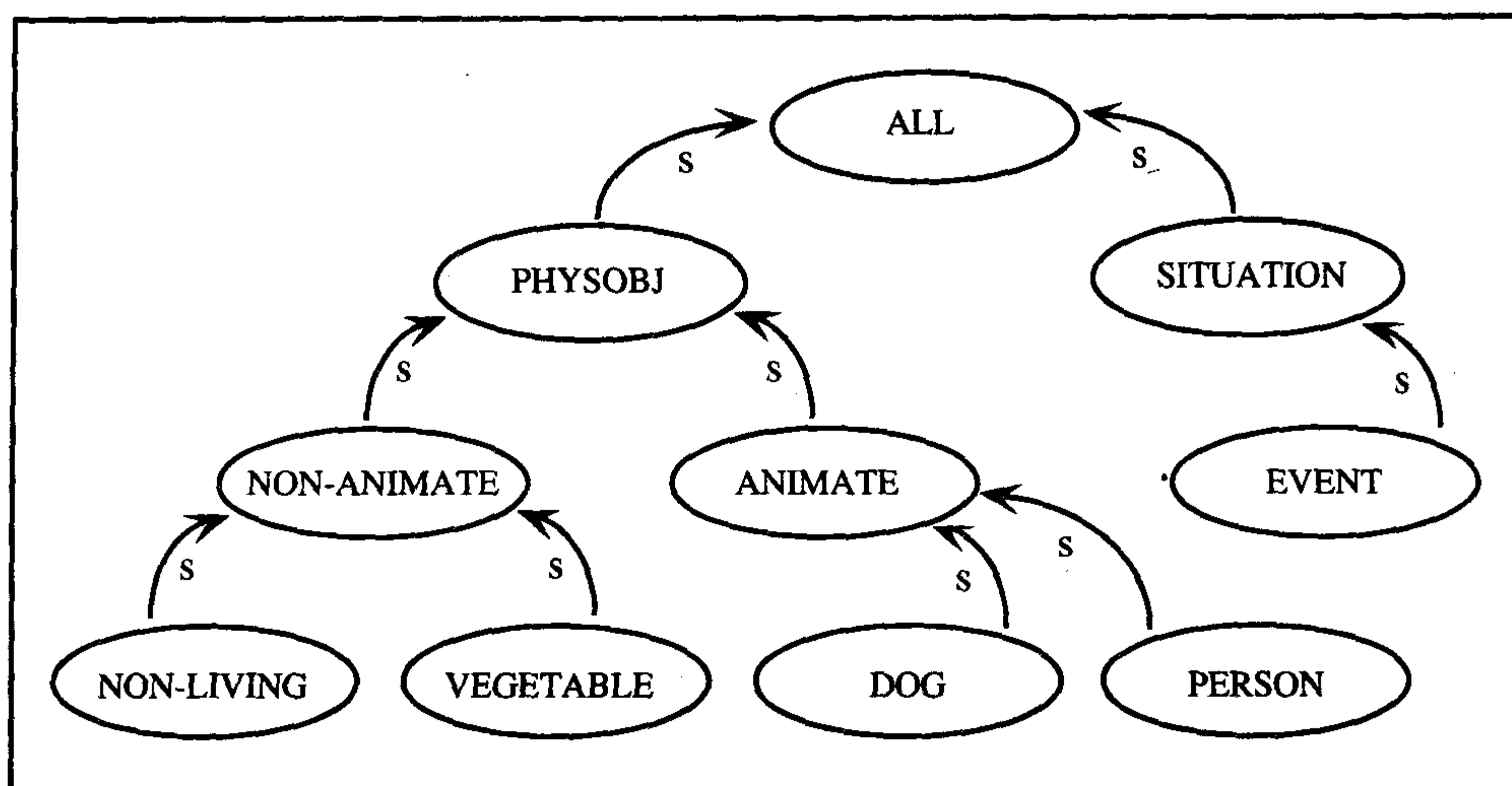


图 10.6 类型层次体系的一部分

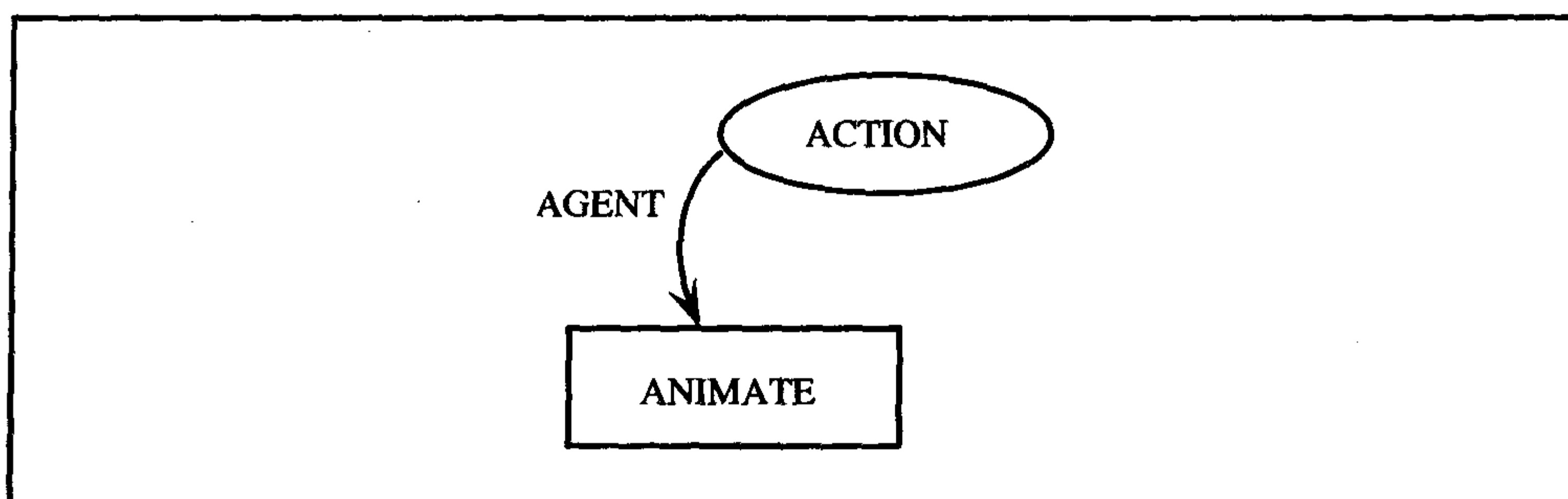


图 10.7 所有的动作都有一个有生命的主体

语义网络最重要的特点就是属性继承。例如,给定图 10.8 所示的语义网络,RUNS1 这个动作类继承了这样的属性,其每个实例都要由 ANIMATE 对象充当 AGENT。通常,对多数节点而言,继承可以通过对图的搜索来完成。为找到某个特定角色 R 对应的约束条件,先要检查特定节点(例如 RUNS1)以找到 R 关系,如果找不到,就沿 s 链上移(例如,在节点 ACTION 上)。如果还没有找到 R 关系,就继续沿 s 的层次关系上移,直到找到一个 R 关系或搜索到层次体系的顶端为止。

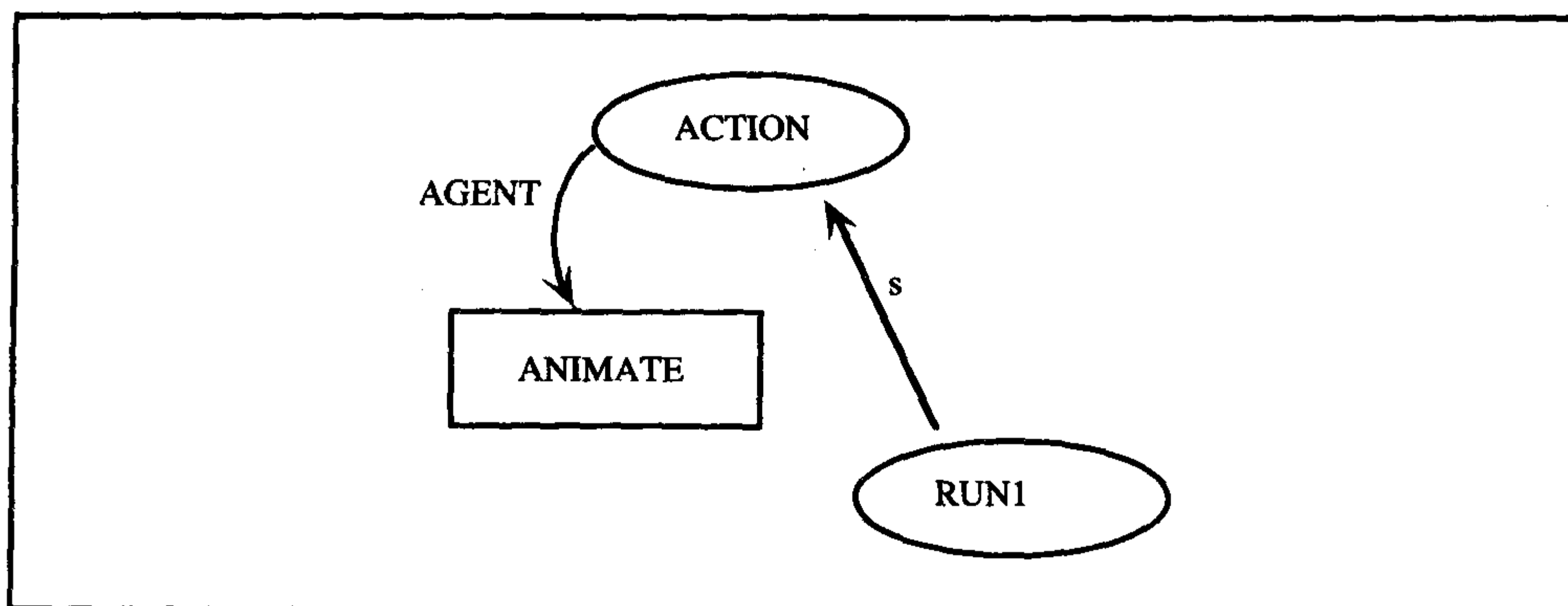


图 10.8 表示角色继承的网络

对于类型众多的动词而言,继承层次体系对于选择限制的表示非常有用。图 10.9给出了作为 ACTION 子类的动词词义集合的选择限制。借助这种继承机制可以发现,通过对 ACTION 类的继承,动作类 TRANSFER-ACTION 得到了 AGENT, AT-TIME 和 AT-LOC 这三种语义关系。它从 OBJ/ACTION 类里继承了 THEME 和 INSTR;同时,拥有为 TRANSFER-ACTION 直接定义的 TO-POSS。需要注意的是,这种表示法定义了所有可能的语义关系,而不只是用来区分动词子类的语义关系。

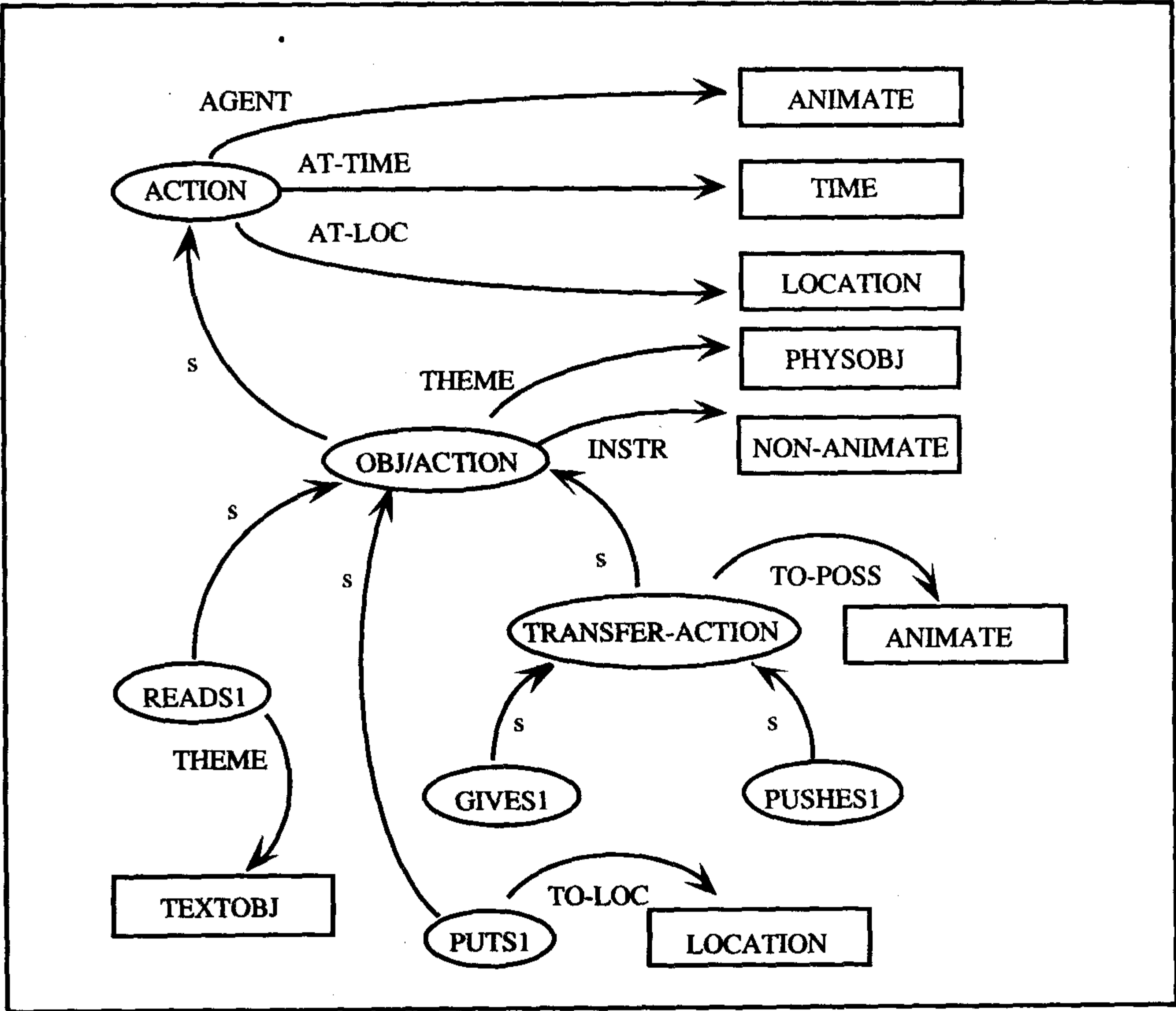


图 10.9 带角色的动作层次体系

通常,一个动词对其参数所加的限制比它所继承的严格。例如,词义 READS1 位于 OBJ/ACTION 之下,于是它可以继承 AGENT, AT-TIME, AT-LOC, THEME 和 INSTR 这些语义关系。但是,我们想让主题 THEME 属于 TEXTOBJ 类别。上文曾经简要介绍过继承算法,在找到第一个信息时就停止,因而它使用的是该动词最明确的信息。所以,这个新的 THEME 约束覆盖了继承来的更一般的约束。

除了子类型和参数关系之外,语义网络还可以表示其他形式的词法结构的一般知识。整体-部分层次是另一种重要的层次体系,在这种层次体系中,实体与其组成部分相关联。在英语中,这种关系往往用介词 of, 名词-名词修饰关系或者所有格形式来表示,比如:

- The desk drawer (抽屉是桌子的一部分)
- The man's head (脑袋是人的一部分)
- The handle of the drawer (把手是抽屉的一部分)

为了识别这种关系,并消除由这些关系引起的歧义,需要表示出语义网络中对象的结构信息。这个问题可以通过引入一种新链(part-of 链)来解决。这样,可以表示屋子是由房间和门组成的,门上有把手,如图 10.10 所示。在这幅图中,部件节点的类型并没有放在节点的位置上,相反,它用一条 isa 边来表示,大多数语义网络表示形式中都可以采用这种表示法。

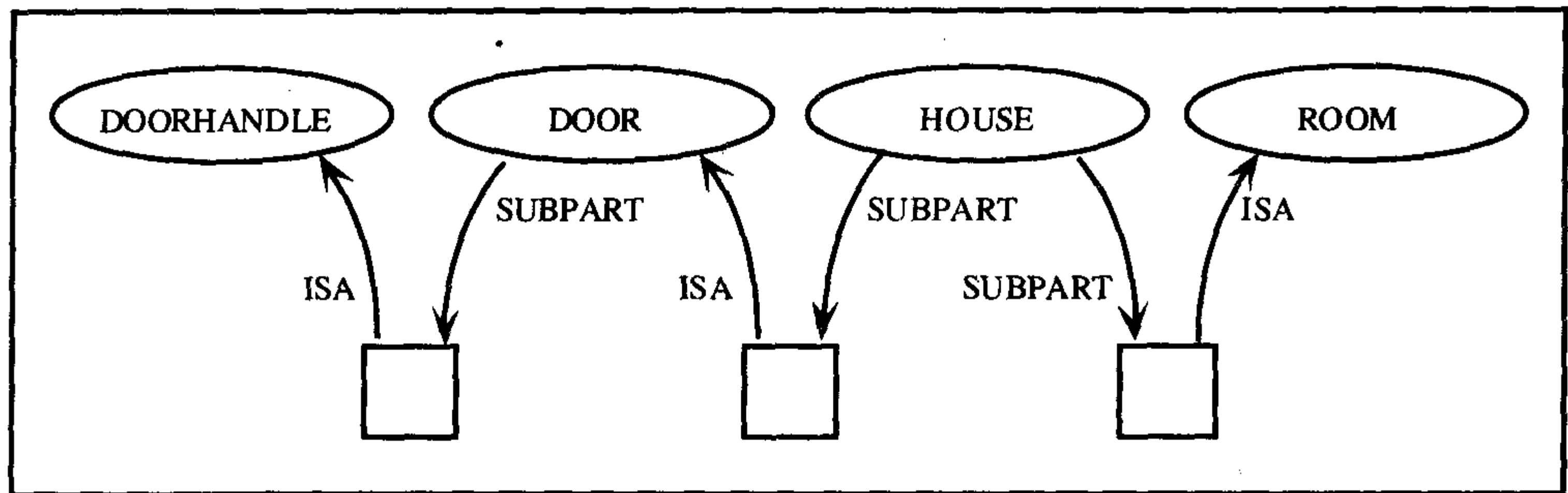


图 10.10 一些组成关系

一个完整的表示体系必须能指明其组成部分是惟一的对象(如门的把手)还是对象(屋子里的房间)集合。例如,人的身体由头、躯干、胳膊和腿等部分组成,而我们需要表示惟一的头连在躯干上,有两条胳膊连在躯干上,等等。这些都构成了 BODY 子类的一部分。此外,这个体系还必须能表示出空间及其他关系,这些关系存在于各组成部分之间。不过,对这个问题细节的深入讨论已经超出了知识表示的范围。

除了能表示属性继承之外,语义网络还有其他用途。例如,我们在后面将会讨论一个很有用的性质,和两个不同词义的语义接近度相关。一开始讨论这个问题时,我们评价语义接近度的方法是在层次类型中找到这两个词义的距离。例如,对图 10.6 所示层次体系稍做扩展,就可以发现 DOG1 和 CAT1 密切相关,它们有公共父类 ANIMATE 类(每一个都只有一个链)。另一方面,DOG1 和 CARROT1 语义就稍微远了一点,因为它们有公共父类 PHYSOBJ,每一个都有几个链。DOG1 和 EVENT 的距离就更远了。这个方法有一些难点,因为它在一定程度上取决于语义网络创建人的观点,不同的结构会产生不同的答案。于是,有人提出一种新的衡量方法,即通过最小公共父类的大小来衡量两个词义间的接近度。用这种方法,程序设计人员的每种分类结果都不会带来太大的影响。基本上,包含两种语义的公共集越小,两个词义之间就越紧密。这种方法用于上例可以得到同样的结论,ANIMATE 类(包含 DOG1 和 CAT1 的最小公共集)包含元素的数量显然比 PHYSOBJ 类(包含 DOG1 和 CARROT1 的最小公共集合)少。同样,PHYSOBJ 类(包含 DOG1 和 CARROT1 的最小公共集合)包含元素的数量显然也比 ALL 类(包含 DOG1 和 EVENT 的最小公共集合)少。

当然,使用类型层次体系只表示了语义接近度的一种意义,还有很多其他结构可能更适于表达这个概念。例如,你可能认为 DOG1 类同 DOG-SHOW1 事件之间有语义关系,但是类型层次体系并没有将它表示出来。对于这种关系,我们需要一种衡量接近度的方法,这种方法能表示哪些对象参与了某事。根据以前提出的概念,这里引入一个新的语义角色,比如 PARTICIPANT,它把事件和这个事件所涉及的对象关联起来,这样可能会有效。不过,本节阐述的层次体系已经足以说明下面的问题。

10.4 统计词义消歧

选择限制只提供了一种粗糙的分类,它只有两种形式,即接受和不接受。结果导致很多词义歧义情况无法解决。为了更好地对人类排歧的过程建模,必须更多地使用预测技术,相对于较少出现的词义,这些技术会优先选择常见词义。本节探讨如何使用统计方法将常见词义形式化。通常,在准确计算统计结果所需信息的数量和获取有用信息的能力之间,我们要找到一个平衡点。为此,必须付出巨大努力。

最简单的技术基于简单一元统计方法。给定一个标记正确的语料库,可以收集每个词不同意义用法的信息。例如,我们看到单词“bridge”使用了 5845 次:

STRUCTURE1 5651 次

DENTAL-DEV37 194 次

根据这些数据,我们每次都会猜测“bridge”是 STRUCTURE1 的意思。如果我们的训练数据有代表性,那么这个技术每次给出正确答案的概率是 97% (5845 次猜测中 5651 次准确)。更一般的情况下,对英语更大范围的使用,可以估算出这种简单的策略大约有 70% 的准确率。当然,我们希望借助一些上下文信息取得更好的结果。现在,考虑一下很少出现的词义 DENTAL-DEV37。尽管它在整个语料库中非常罕见,但是在某些特定文本(那些关于牙科和口腔正畸学的文本)中却是最常见的词义。如果研究出一种技术,能够可靠地识别这种上下文,就可以选出这个词义,从而取得比上述简单算法更好的效果。可以证明,一个文本中的实词能很好地表明该文本的话题。例如,在关于牙科的文章中,诸如“teeth”,“dentist”,“cavity”,“braces”,“orthodontics”等单词会经常用到。如果出现了这些单词,我们希望优先选择 DENTAL-DEV37。

此类信息和词语搭配(即哪些单词经常一起出现)有很大关系。我们可以考虑二元概率、三元概率或者更大范围的组合,比如前后 5 个词、整个句子甚至更大范围的上下文(某些研究工作中使用多达近 100 个词)。对每个词所考察的文本长度称为窗口。

一种解决方法是调整词类标记技术,在标记中使用词义而不是句法类。对于这种方法,需要一个语料库,库中每个词都标记好语义。然后,可以计算一元统计概率(例如,词 w 的词义为 S 的概率)、二元统计概率、三元统计概率,等等。当然,建这样一个语料库需要很大的工作量,不过现在已经有人在做这种语料库了。但是,有两个因素导致直接使用标记这种方法不可行。第一,词义的数量比句法类多得多;第二,为得到合理的结果,必须用到范围更大的上下文,而不能用简单的二元或三元模型。这两个问题表明,要获得足够的训练数据实际上是不可能的,需要另想他法。

基本的解决思路是,计算某个词 w 的词义相对于文本中以词 w 为中心的一个窗口内的词的概率。已知以词 w 为中心的窗口大小为 n ,这个窗口中的词可表示成如下形式:

$$w_1 w_2 \dots w_{n/2} w w_{n/2+1} \dots w_{n-1}$$

我们要计算出使下式取最大值的词 w 的词义 S :

$$\text{PROB}(w/S | w_1 w_2 \dots w_{n/2} w w_{n/2+1} \dots w_{n-1})$$

为计算这些概率,我们用贝叶斯法则改写上式,并且和前面几次一样,要做独立性假设。公式可改写为:

$$\frac{\text{PROB}(w_1 \dots w_{n-1} | w/S) * \text{PROB}(w/S)}{\text{PROB}(w_1 \dots w_{n-1})}$$

因为上式分母不随词义变化,所以分母可以忽略。进一步假定,每个词 w_i 的出现独立于窗口中的其他词,则 $\text{PROB}(w_1 \dots w_{n-1} | w/S)$ 近似改写成:

$$\prod_{i=1, n-1} \text{PROB}_n(w_i | w/S)$$

其中, $\text{PROB}_n(w_i | w/S)$ 是当词 w 的词义为 S 时,词 w_i 在以词 w 为中心、大小为 n 的窗口中出现的概率。合并上述公式,最佳词义 S 就是使下式取最大值的那个词义:

$$\text{PROB}(w/S) * \prod_{i=1, n-1} \text{PROB}_n(w_i | w/S)$$

由于假定各个事件的出现不相关,最终可以证明窗口越大,所需的数据就越少,因为这样从每个窗口中能收集更多的数据。给定一个语料库,通过每个词大小为 n 的窗口来收集数据,可以计算词的每个词义出现的次数,并且记录窗口中所有的词。考虑一下图 10.11 中为词“bridge”假设的信息,这里使用窗口的大小为 11,语料库中有一千万个词。

	对于 <i>bridge</i> / STRUCTURE1	对于 <i>bridge</i> / DENTAL-DEV37	在任何 窗口中
<i>teeth</i>	1	10	300
<i>suspension</i>	200	1	2000
<i>the</i>	5500	180	500 000
<i>dentist</i>	2	35	900
出现的总次数	5651	194	501 500

图 10.11 “bridge”的几个意义在伪语料库中的相关数据

根据 $\text{PROB}_n(w_i | w/S)$ 的定义,我们可以按下式计算它的值:

$$\text{PROB}_n(w_i | w/S) = \frac{\text{Count}(\text{在以 } w/S \text{ 为中心的窗口中 } w_i \text{ 出现的次数})}{\text{Count}(w/S \text{ 作为窗口中心的次数})}$$

给定如图 10.11 中所示的数据,可得如下计算结果:

$$\text{PROB}_n(\textit{teeth} | \textit{bridge}/\text{STRUCTURE1}) = 1/5651 = 1.77 * 10^{-4}$$

$$\text{PROB}_n(\textit{teeth} | \textit{bridge}/\text{DENTAL-DEV37}) = 10/194 = 0.052$$

$$\text{PROB}_n(\textit{suspension} | \textit{bridge}/(\text{STRUCTURE1}) = 200/5651 = 0.035$$

$$\text{PROB}_n(\textit{suspension} | \textit{bridge}/\text{DENTAL-DEV37}) = 1/194 = 5.15 * 10^{-3}$$

$$\text{PROB}_n(\textit{the} | \textit{bridge}/\text{STRUCTURE1}) = 5500/5651 = 0.97$$

$$\text{PROB}_n(\textit{the} | \textit{bridge}/\text{DENTAL-DEV37}) = 180/194 = 0.93$$

$$\text{PROB}_n(\textit{dentist} | \textit{bridge}/\text{STRUCTURE1}) = 2/5651 = 3.54 * 10^{-4}$$

$$\text{PROB}_n(\textit{dentist} | \textit{bridge}/\text{DENTAL-DEV37}) = 35/194 = 0.18$$

词义的上下文无关概率很容易计算出来:

$$\text{PROB}(\textit{bridge}/\text{STRUCTURE1}) = 5651/501500 = 0.113$$

$$\text{PROB}(\textit{bridge}/\text{DENTAL-DEV37}) = 194/501500 = 3.87 * 10^{-4}$$

这两个概率给出了不考虑任何上下文(也就是说,窗口大小为 1)的情况下每个词义的可能性。需要注意的是,在只包含单词“the”的窗口中,词义的概率非常接近上下文无关概率。

$$\begin{aligned} & \text{PROB}_n(\text{the} \mid \text{bridge/STRUCTURE1}) * \text{PROB}(\text{bridge/STRUCTURE1}) \\ & = 0.97 * 0.113 = 0.109 \\ & \text{PROB}_n(\text{the} \mid \text{bridge/DENTAL-DEV37}) * \text{PROB}(\text{bridge/DENTAL-DEV37}) \\ & = 0.93 * 3.87 * 10^{-4} = 3.6 * 10^{-4} \end{aligned}$$

这表明单词“the”对两个词义几乎没有区分能力,这正是我们期望的结果,因为“the”与名词搭配太常见了。通常,功能词(function word)太过常见,因此它们对于词义排歧没有任何帮助。在本例中,像“teeth”这样的内容词(content word)才对排歧有非常大的作用。事实上,对只包含“dentist”的窗口进行概率计算,将原来的优先选择次序反了过来:

$$\begin{aligned} & \text{PROB}_n(\text{dentist} \mid \text{bridge/STRUCTURE1}) * \text{PROB}(\text{bridge/STRUCTURE1}) \\ & = 3.54 * 10^{-4} * 0.113 = 4 * 10^{-5} \\ & \text{PROB}_n(\text{dentist} \mid \text{bridge/DENTAL-DEV37}) * \text{PROB}(\text{bridge/DENTAL-DEV37}) \\ & = 0.18 * 3.87 * 10^{-4} = 6.97 * 10^{-5} \end{aligned}$$

当然,使用更大的窗口,内容词会有更多的机会对结果产生巨大的影响。例如,分析句子“The dentist put a bridge on my teeth”(牙医在我的牙齿里放了一个牙桥)。除了“dentist”和“teeth”之外,其他单词都是常用词,它们和两个词义共现的频率差不多。因此,对于每个词义的相对概率都没有什么影响。但是,词“dentist”和“teeth”出现在同一个窗口中,它们共同作用,从而导致优先选择“bridge”的不常见语义。事实上,DENTAL-DEV37这个词义的计算结果是 $3.6 * 10^{-6}$,显然要比STRUCTURE1的 $7.08 * 10^{-7}$ 这个结果大很多。对这些计算结果进行归一化,就能将其用于计算真正的概率。更确切地说,如果单词“bridge”只有这两个词义,那么在这个窗口中使用DENTAL-DEV37这个词义的概率是 $3.6 * 10^{-6} / (3.6 * 10^{-6} + 7.08 * 10^{-7}) = 0.84$ 。

10.4.1 搭配与互信息

这个领域中的许多研究工作都用到了搭配信息,用于衡量两个词出现在同一个文本窗口中的概率有多少。计算这种概率的一种方法是分析相关性统计信息(其中 n 是窗口大小):

$$C_n(wS, w') = \frac{\text{PROB}(w/S \text{ 和 } w' \text{ 出现在同一窗口中})}{\text{PROB}(w/S \text{ 在窗口中}) * \text{PROB}(w' \text{ 在窗口中})}$$

值得注意的是,这和上面只在分母中所用的计算方法不同。如果 K 是语料库中窗口的数目,那么上面每个概率值都可以用 $\text{Count}(\text{出现在窗口中的事件次数})/K$ 来计算。用这种计算方法替换 $C_n(wS, w')$ 所用的每个概率值,并进行简化后,得到公式:

$$C_n(wS, w') = \frac{K * \text{Count}(w/S \text{ 和 } w' \text{ 在窗口中共现的次数})}{\text{Count}(w/S \text{ 在窗口中的次数}) * \text{Count}(w' \text{ 在窗口中的次数})}$$

在我们的样例语料库中, K 是 10 000 000。根据图 10.11 中的数据, C_n 的计算如下所示:

$$\begin{aligned} C_n(\text{bridge/STRUCTURE1}, \text{teeth}) &= (10^7 * 1) / (5651 * 300) = 5.9 \\ C_n(\text{bridge/DENTAL-DEV37}, \text{teeth}) &= (10^7 * 10) / (194 * 300) = 171.9 \\ C_n(\text{bridge/(STRUCTURE1}, \text{suspension)}) &= (10^7 * 200) / (5651 * 2000) = 17.7 \\ C_n(\text{bridge/DENTAL-DEV37}, \text{suspension}) &= (10^7 * 1) / (194 * 2000) = 2.5 \\ C_n(\text{bridge/STRUCTURE1}, \text{the}) &= (10^7 * 5500) / (5651 * 500,000) = 1.94 \\ C_n(\text{bridge/DENTAL-DEV37}, \text{the}) &= (10^7 * 180) / (194 * 500,000) = 1.84 \\ C_n(\text{bridge/STRUCTURE1}, \text{dentist}) &= (10^7 * 2) / (5651 * 900) = 3.9 \\ C_n(\text{bridge/DENTAL-DEV37}, \text{dentist}) &= (10^7 * 35) / (194 * 900) = 200 \end{aligned}$$

如果比值接近 1,则两个词基本上是随机共现;如果比值大于 1,则两个词的共现就不是随机事件。请注意,两个词义与“the”这个词的相关值分别是 1.94 和 1.84。这表明两个词义与“the”共现的概率比随机出现稍高一点,但“the”不能表明要优先选择哪一个,不选哪一个。

为了更好地区分基于比值的统计信息,这一领域的研究工作中常常用比值的对数来表示,如果比值小于 1,结果将是负数。这能更直观地表示优先选择与非优先选择。对于本节中描述的词的比值,这种衡量方法称为两个词的互信息,记为 $I_n(w_1, w_2)$ 。

$$I_n(w_1, w_2) = \log C_n(w_1, w_2)$$

以“bridge”的两个语义为例,其互信息量为:

$$I_3(\text{bridge/STRUCTURE1, teeth}) = 1.77$$

$$I_3(\text{bridge/DENTAL-DEV37, teeth}) = 5.14$$

$$I_3(\text{bridge/STRUCTURE1, the}) = 0.66$$

需要注意的是,彼此之间无关联且随机共现的词的互信息量接近于 0。如果两个词是负相关的,即它们共现的概率比随机发生还低,那么它们的互信息量就是负数。如果用互信息量比较不同的词义,我们要把窗口中不同词的分数相加,而不是相乘,因为它们是对数值。

10.5 统计语义优选

本章第一节讨论了选择限制及其在消歧中的应用。这种方法的问题在于,约束非此即彼的性质使得它不能表示出语义选择的优先程度。例如,“bridge”这个词有表示建筑的意义,也有表示牙科设备的意义。而在句子“I painted the bridge”中,显然指的是建筑这个意义。但是 PAINTS1 的选择限制要求其 THEME 必须是一个实体,而“bridge”的两个意义都属于这个定义的范畴,都满足约束条件。因此,现有的模型在选择更为常见的解释时无能为力。另一个尚未解决的问题是结构歧义,如 PP 附着问题。本节将研究一种语义选择模型,这有助于解决上述问题。

一般的思路是统计语义角色在语义之间出现频度的信息。这样,在解释一个句子时,会选择最常见的语义组合。我们从 10.1 节讨论的选择限制开始谈起。这里,限制描述成逻辑形式中篇章变量之间的一元和二元关系。借助一个标注好语义信息的语料库,我们能收集到所有这些一元和二元关系频度的统计信息。现在,先假定有足够的信息,根据这些数据,可以准确地算出每个关系的值。本节后面的部分将对这个假设放宽条件。

分析例句“I painted the bridge”,它的初始逻辑形式如下:

(PAINTS1 p1 [AGENT (PRO i1 I1)]
[THEME <THE b1
{STRUCTURE1 DENTAL-DEV37}>])

一元和二元关系如下所示:

(PAINTS1 p1), (I1 i1), ({STRUCTURE1 DENTAL-DEV37} b1),
(AGENT p1 i1), (THEME p1 b1)

我们用每个子成分出现概率的乘积来计算每个特定解释的出现概率;也就是说,如果一个逻辑形式(LF)包含 n 个关系 R_1, \dots, R_n ,那么:

$$PROB(LF) = \prod_{i=1, n} PROB(R_i)$$

当然,只有当所有关系的出现互不相关时,这项计算才正确。显然,这个假设并不总是正确的,但是这项技术依然很有用。

让我们再前进一步,研究计算二元关系概率的方法。通过比较某些三元组(reln head arg)出现的次数和它的中心成分出现的次数,可以算出该三元组的概率。一开始,一个合理的选择是选择标准的最大似然估计:

$$PROB((reln\ head\ arg) | head) \approx \frac{Count((reln\ head\ arg))}{Count(head)}$$

现在,考虑如何应用这种方法。“I painted the bridge”有两种可能的解读,分别对应“bridge”的两个词义。由于其余的关系在这两种情况下完全相同,所以,两种解释的概率差就简化成(THEME PAINTS1 STRUCTURE1)和(THEME PAINTS1 DENTAL-DEV37)的概率差。如果已知给桥(跨越水面的建筑物)喷漆是一种常见合理的行为,而且在数据库中出现过若干次,而给牙科设备喷漆是一种不太可能发生的行为,而且可能从未在数据库中出现过,因此优先选择前者。

当然,对结构上有歧义的句子,可能要比完全不同的语义结构。但是,我们可以用同样的方法来比较这些结构。考虑句子“He saw the man with a telescope”和“He saw the man with a hat”。这两个句子的逻辑形式如图 10.12 所示。

1. He saw the man with a telescope.
 - 1a. (SEES1 p1 [EXPERIENCER (PRO h1 HE1)]
[THEME <THE w1 MALE-PERSON1>]
[INSTR <A b1 TELESCOPE1>])
 - 1b. (SEES1 p1 [EXPERIENCER (PRO h1 HE1)]
[THEME <THE w1 MALE-PERSON1
([WITH1 ... WITHn] w1 <A b1 TELESCOPE1>)>])
 2. He saw the man with a hat.
 - 2a. (SEES1 p1 [EXPERIENCER (PRO h1 HE1)]
[THEME <THE w1 MALE-PERSON1>]
[INSTR <A w2 HAT1>])
 - 2b. (SEES1 p1 [EXPERIENCER (PRO h1 HE1)]
[THEME <THE w1 MALE-PERSON1
([WITH1 ... WITHn] w1 <A w2 HAT1>)>])

图 10.12 两个歧义句的逻辑形式

即使语义结构不同,它们也有很多相同的二元关系。例如,句子 1a 和 1b 有相同的三元组 (EXPERIENCER SEES1 MALE1) 和 (THEME SEES1 MALE-PERSON1)。它们的区别仅仅在于其中一个包含 (INSTR SEES1 TELESCOPE1), 假设这是一个常见模式; 而另一个包含 (WITHi MAN1 TELESCOPE1), 其中, WITHi 作用于所有可用“with”指示的语义关系 (比如 part-of, accompanied-by, wearing, 等等)。后者也经常出现, 因此两种解释似乎都合乎情理。对句子 2, 对这两个 LF 的比较最后简化为关系 (INSTR SEES1 HAT1) 和 (WITHi MAN1 HAT1) 概率的比较。假设前者是一种很罕见的关系, 而后者在 WITHi 表示 ACCOMPANY (共现) 关系时经常出现, 因为人戴帽子是合理的。所以, 在第二个例子中, 应该选择 2b 的解释, 同时“with”的解释歧义亦被消除。

虽然在第一种情况下,共现统计信息无法确定哪种解释更好,但概率模型却可以让你量化句子的确定度(或者说是模糊度)。另外,可以将这些技术组合在一起形成一种消歧方法,并且可以应用于已构造好逻辑形式的每个主成分。其返回值能用于调整最佳优先句法分析方法所得成分的最后结果。

然而,这个方法的首要问题是如何才能获得一组准确的统计信息。如果读者还有印象,在开始讨论词的句法类时,我们就已经提出了这个问题。在处理词义时,这个问题变得更严重,因为每个词都有太多不同的意义。所以,必须使用概率计算方法,因为语料库中从未出现的合法三元组会很常见。

如果有一个预先定义的抽象层次体系,那么该层次体系能给未注意到的三元组提供计算方法。基本思路是不只为所用到语义的三元组统计数据,也为这些语义抽象类的三元组统计数据。考虑图 10.13,它表示的是父类的对象对应到其每个子类的概率是多少。根据该图,SMALL-PHYSOBJ 对应到 DENTAL-APPLIANCE 的概率 $PROB(DENTAL-APPLIANCE|SMALL-PHYSOBJ)$ 是 0.005。

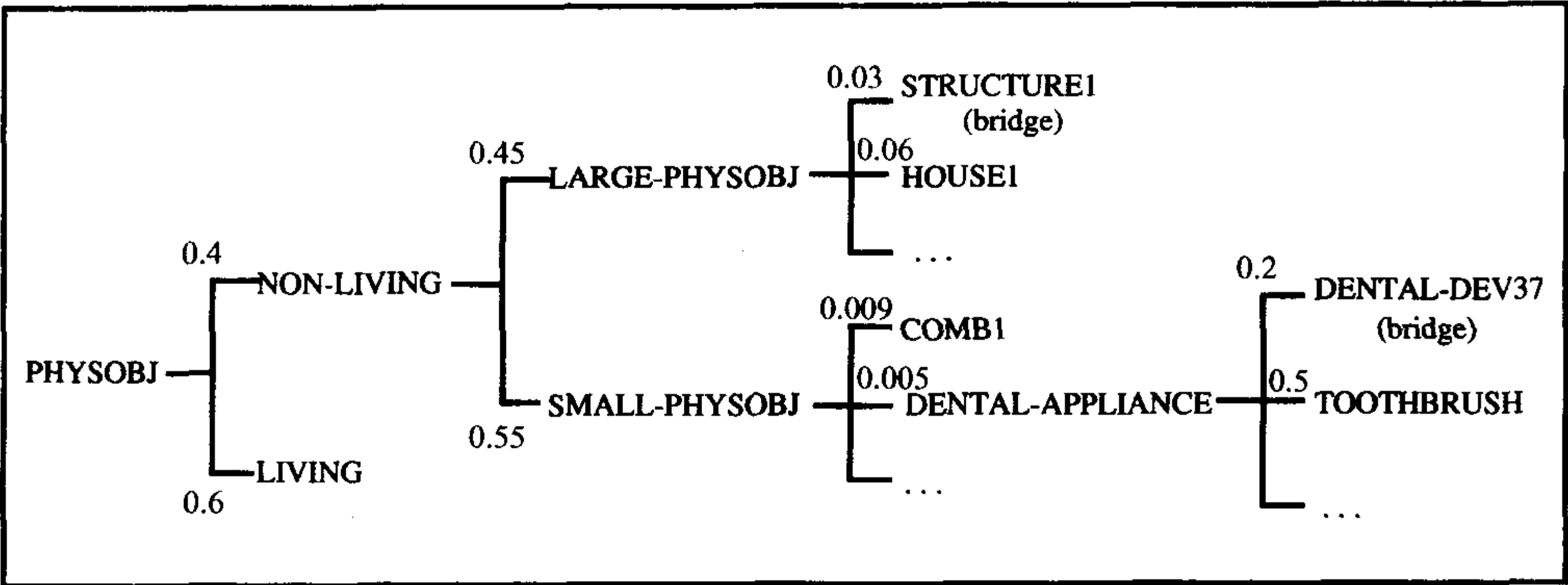


图 10.13 概率类型层次体系

已知这样一个层次体系,如果观测到一个三元组 (THEME PAINTS1 STRUCTURE1),我们不仅记录该三元组的出现,还记录 STRUCTURE1 对应于抽象实体的记录,如 (THEME PAINTS1 LARGE-PHYSOBJ) 和 (THEME PAINTS1 PHYSOBJ)。因此,在分析含有 1000 个 PAINTS1 实例的扩展语料库后,可以获得图 10.14 所示的数据和概率计算结果。当然,还有许多其他三元组虽然被观测到了,但在这里没有列出来。不过,这四个三元组足以说明这个例子。当遇到一个以前从未出现的三元组时,比如 (THEME PAINTS1 DENTAL-DEV37),可以用父类的信息对该三元组进行计算。

三元组	出现的次数	<i>PROB</i>
(THEME PAINTS1 NON-LIVING)	960	0.96
(THEME PAINTS1 SMALL-PHYSOBJ)	300	0.3
(THEME PAINTS1 LARGE-PHYSOBJ)	650	0.65
(THEME PAINTS1 STRUCTURE1)	10	0.01

图 10.14 一些语义关系的统计信息

再次分析句子“I painted the bridge”。因为“bridge”的 DENTAL-DEV37 这个词义从未在训练数据中出现过,所以三元组 (THEME PAINTS1 DENTAL-DEV37) 的概率值是 0,表明这种解读不可能发生,但这和我们的直觉不符。考虑 DENTAL-DEV37 的抽象类,可以计算其概率。因为 DENTAL-DEV37 是 DENTAL-APPLIANCE 的子类,而且可能有 (THEME PAINTS1 DENTAL-APPLIANCE) 的计算结果,如果没有,再考虑更抽象的一层。因为 DENTAL-APPLIANCE 属于 SMALL-PHYSOBJ,那么看看是否有 (THEME PAINTS1 SMALL-PHYSOBJ) 的估计结果。我们有这种三元组的概率,概率值是 0.3:

$$PROB((THEME PAINTS1 SMALL-PHYSOBJ) | PAINTS1) = 0.3$$

因为 DENTAL-DEV37 属于 SMALL-PHYSOBJ,所以这个计算结果有一定的相关性。但是,0.3 的结果明显太过夸张了,不能直接使用,因为 DENTAL-DEV37 仅仅是 SMALL-PHYSOBJ 的一个小的子集。如果考虑任意一个 SMALL-PHYSOBJ 是 DENTAL-DEV37 的概率,则可以修正这个结果。使用图 10.13 的数据可得:

$$\begin{aligned} PROB(DENTAL-DEV37 | SMALL-PHYSOBJ) &= \\ &PROB(DENTAL-DEV37 | DENTAL-APPLIANCE) * \\ &PROB(DENTAL-APPLIANCE | SMALL-PHYSOBJ) \\ &= 0.005 * 0.2 = 0.001 \end{aligned}$$

因此,对 $PROB((THEME PAINTS1 DENTAL-DEV37) | PAINTS1)$ 的一个估计值是:

$$\begin{aligned} &PROB((THEME PAINTS1 SMALL-PHYSOBJ) | PAINTS1) * \\ &PROB(DENTAL-DEV37 | SMALL-PHYSOBJ) = 0.0003 \end{aligned}$$

这个例子使用平滑技术来调整估计值。通过引入额外的乘法因子或综合其他信息等方法,可以得到更好的结果。

10.6 组合消歧方法

本章已经讨论了两种不同的方法,这些方法用来消除词义歧义和语义关系歧义。基于选择限制的方法先假设了一些严格的限制条件,这些条件规定词的哪个意义可以作为其他词义的参数。这些限制都基于每个词义的语义因素。这种方法可以推而广之,也就是说,限制条件可以表示优选,和限制条件冲突最少的解释就是首选解释。

另一种方法基于对语料库的统计分析。我们优先选择在语料中出现频率最高的词义搭配。在纯统计方法中,这个结果和句子的基本意义没有内在联系。而附着决策的选择条件是词典条目的哪些模式在哪些结构中出现最频繁。

看起来,它们似乎是两种截然不同的方法,许多研究人员也倾向于认为这是一种非此即彼的选择。事实上,这两种方法可以看做是相辅相成的,我们期望有一种方法同时使用这两种技术,并且其结果优于只用一种技术的方法。10.5 节描述了同时采用这两种方法的技术。它使用从语料库中获得的统计度量信息,为了记录其语义关系,又将语义组织成类型层次体系。借助这些信息,可以将统计的抽象语义类信息用于计算那些以前从未出现过的三元组的概率。因此,类型层次体系的语义知识为处理稀疏数据提供了一种很有吸引力的方法。

如果有足够多的数据,读者可能希望统计方法最终能表示出基于语义的方法的所有信息;也就是说,常见用法模式一般能描述词和词之间的语义关系。如果这种设想为真,那么只用统

计数据就能建立一个与类型层次体系等价的方法。然而,上述设想不能成立,一个明显问题就是新词(或者说,相对于某人而言的新词)。我们总会不断碰到新词。给出一个词的语义,即使以前从未在上下文中遇到过这个词,我们也能马上在适当的环境中使用这个词。举一个简单的例子,假如我们知道“preef”是一种帆船,就能理解句子“We sailed the lake in my preef yesterday”的含义,即“in my preef”修饰的是动词短语而不是“the lake”。相反,如果“preef”的意义是一个庄园或其他种类的大片土地,那么与前面相反,我们会选择附着结构。这些选择是依据词“preef”的新的语义知识做出的。纯统计的方法不可能做出合理的选择,因为 PP 结构“in my preef”以前从未遇见过,也就没有关于“preef”的足够观测样本,因而无法将它和更抽象的类(如“boat”或“estate”)联系在一起,即使我们的数据规模已经足够大。

当然,如果像 10.5 节那样,假定存在一个语义层次体系可以将统计信息组织起来,就可以有效运用统计方法确定正确的附着结构。例如,再来分析新词“preef”,这里定义它是一种船。在这种情况下,我们会比较船作为航行活动发生地点的概率以及湖位于一艘船里的概率。但是,这恰好说明了一个观点,即综合运用两种方法才是有效的技术。如果用第 7 章介绍的 PP 附着技术改进这种组合方法,可将其进一步推广。一个简单的方法是假设这两种方法产生的结果互不相关,只将这两种方法结果的乘积作为最终结果。显然,还可以研究其他组合结果的有效方法,从而更好地利用这两种方法之间的相似之处。

10.7 小结

歧义消解是计算系统研究中面临的一个极为重要的问题。本章探讨了几种不同的歧义处理方法。消歧一般以语义的层次体系表示作为解决问题的出发点,并经常使用语义网络记录歧义信息。借助这个层次体系,可以定义选择限制,并利用这些约束条件减少候选词义。另外,在给定前后词的条件下,可用统计方法表示每个词的词义优选;基于语义关系的相对频度,也可以用统计方法表示特定解释的优选。

10.8 相关工作与深入阅读材料

词义消歧曾被认为是计算语言模型研究中最难的问题之一。早在 20 世纪 60 年代,Bar-Hillel(1960)就提出,这个问题是机器翻译之类的自然语言应用系统开发中的主要障碍。Katz 和 Fodor(1963)提出将选择限制应用到语言学中,从此,各种使用语义特征的方法在几乎所有的计算模型中得到应用。语义选择最早的倡导者之一是 Wilks(1975),Wilks 开发了一个系统,使用语义模板解释经过基本语法分析的句子表示形式。这个系统运用了基于最小化语义约束条件冲突的启发式技术。

解决歧义问题的系统通常综合运用多种技术,而不仅仅是选择限制。其中,许多技术都直接映射到基本表示形式而非逻辑形式,但是这种知识表示形式中的约束条件和选择限制很相似。Hayes(1977)和 Hirst(1987)综合运用了选择限制以及从语义网络中算出的语义相关度。BORIS 系统(Dyer, 1983)结合了自顶向下的期望和选择限制这两种方法来理解复杂文本。

约束满足首先被 Waltz(1975)引入到 AI 中解释视觉场景,Freuder(1982)和 MacWorth(1977)详细说明了这种方法的形式化基础理论。Mellish(1985)用约束满足技术来解决语义消歧问题,并用于识别名词短语的指代对象。

Quillian(1968)引入了自然语言的语义网络,用来表示语义以及词之间的概念联系。语义网络的发展推动了很多对网络表示方法的研究[例如, Woods(1975), Findler(1979), Sowa(1984; 1991)]。由于表示方法本身的吸引力,语义网络成为表示知识和支持一般推理的通用框架。但这里提到的系统更接近于它的最初目的——获取词和词之间的概念联系。因此,我们并不试图用这些网络定义任何一般的推理过程,甚至没有说明如何表示现实世界中的特定事实。这些内容将在第 13 章进行讨论。最近,在创建综合性英语词典数据库方面的一个成果是 WordNet(Miller, 1990),可以通过普林斯顿大学的 ftp 获得。

直到最近,统计模型才用于处理歧义问题。对词典编纂来说,词的搭配问题是一个首先需要关注的领域。信息论中提出了互信息的概念(Fano, 1961),并成功应用于词义排歧。现有工作成果的一个好例子是 Yarowsky(1992)。最近,关于语料库运用研究的一个好的资源是《计算语言学》(*Computational Linguistics*)1993 年的第 19 卷第一期和第二期的专刊。

另一种排歧方法要使用语义网络中的扩展激活方法。在这种方法中,相关的词义可以互相激活。这个方法的例子能够在 Hirst(1987)中找到。与此相关的方法是连接模型(connectionist model),这个模型从神经网络的相互作用角度出发描述了完整的计算过程。这种方法的例子可参见 Cottrell 和 Small(1983)以及 Pollack 和 Waltz(1985)。分布式连接模型比较好的入门文章是 Rumelhart 和 McClelland(1986)。

关于附着问题有很多文献。Ford, Bresnan 和 Kaplan(1982)认为动词的语义选择很可能决定了附着的选择。Crain 和 Steedman(1985)认为基于名词短语使用与否的解释筛选在这一领域中起到了一定的作用。Hirst(1987)描述了一个使用词法优选和词义似然性的系统。Jensen 和 Binot(1987)使用一个在线字典来寻找介词短语附着的常见模式。Dahlgren(1988)提出了一组基于不同命题和介词对象语义类的优选规则。最近,使用类似于 10.5 节所描述的统计方法的工作可以在 Grishman 和 Sterling(1992)中找到。Hindle 和 Rooth(1993)给出了如何从未标注文本中统计 PP 附件成分信息的方法。Resnik(1993)探讨了一些技术,可以利用 WordNet 搜集用于各种句法歧义的信息。

10.9 习题

1. 【易】根据图 10.9 的层次体系列出动词语义 PUTS1 的所有选择限制。
2. 【易】用 10.4 节描述的 C_n 函数,计算单词“bridge”的每个词义在 5 个词的窗口“the suspension bridge the construction”中的值。
3. 【易】用 10.5 节描述的概率计算技术和图 10.13、图 10.14 中的数据,计算三元组 (THEME PAINTS1 HOUSE1) 的概率,假设该三元组以前未见过。
4. 【中】实现一个程序计算词“happy”和“dog”的互信息值,要求用 3 个词的窗口,这里窗口允许跨句切割,给定下面一个小语料库:

I saw a happy dog by the river. I was never happy before I met my dog. It was sad that the dog left. A happy cat is a well-fed cat. Cows sit in pastures all day and eat grass. My dog is happy most of the time.

这些词共现的概率比随机出现的概率高吗? 在计算概率和互信息值的时候,如果做了任何假设,都要加以描述。

5. 【中】定义一个数据结构,用来表示一个带格值约束的简单类型网络。已知该层次体系,写一个程序,要求给定该层次类型中的任何类型,计算出格值约束的全集。在图 10.9 中测试该网络,得到 PUTS1 的格值约束。请在文档中说明你的数据结构与算法。
6. 【中】实现一个系统,要求它能维护一个类型层次体系,并允许测试任何两种类型是否兼容。具体地说,该系统应允许通过下述函数添加类型信息:

(Add-Subtype T1 T2)——断言 T1 是 T2 的子类型

该信息可以用于下面的函数:

(Test-Intersection T1 T2)

如果不知道两类型是否相交,则返回空;如果两类型相交,则返回相交部分的名称。具体地说,如果 T1 是 T2 的子类型,则返回 T1;如果 T3 是 T1 和 T2 的子类型,则假定 T3 是相交部分,并返回 T3。给出数据结构的说明文档,并验证你的系统可以处理的输入/输出范围。

7. 【中】为了准确解释下述句子,请对 10.2 节中的语法、词典、语义层次体系和选择限制进行扩展:

He gave the book to the college.

He knows the route to the college.

说明用你的数据采用自底向上的 chart 分析器为何不会产生错误的解释。

8. 【中】10.5 节的消歧技术仅仅基于二元关系概率。如何对它进行扩展,使之也能表示一元关系? 详细说明你的算法,并演示怎样用它处理下面的句子:

He painted the suspension bridge at night.

可以假设除了“bridge”之外其他词都没有歧义,请补充 10.4 节和 10.5 节没有提及而你的方法又需要的数据。

9. 【难】以你解决第 5、第 6 两题的方法作为组成部分,实现 10.1 节描述的约束满足算法,并在歧义句生成的选择限制范围内进行测试。这些歧义句如下:

The dishwasher read the article.

He painted the bridge.

使用一个类型层次体系,使之包括图 10.1 和图 10.2 所示的两个层次体系,以及图 10.9 显示的动词层次体系。对它进行必要的扩展,使之包含名词“bridge”的 4 个不同词义,以及在你的测试句中使用的其他词的意义,用以上这些来测试你的程序。另外构造两个包含歧义词的句子,用来验证你的方案在上述两个句子中没有说明的特性。通常,你的程序应该详细说明并全面测试。

你不需要用句子的完整逻辑形式作为测试的起点。相反,输入的可以是篇章变量的一元或二元类型关系。因此,“The dishwasher read the article”的输入就是一元关系列表:

```
(( (DISHWASH1 DISHWASH2) d1) ((READS1 READS2) r1)
  ((ARTICLE1 ARTICLE2) al))
```

以及二元关系列表:

```
(( (AGENT r1 d1) (THEME r1 al))
```

第 11 章 语义解释的其他策略

前面的章节描述了语义解释的方法,借助这些方法,句法结构采用逐条使用规则的形式驱动语义解释。但还有别的方法可用来构造系统,这些方法具有其他优点。其中一些技术可以针对特定的应用需求较快地开发出一个系统,而其他技术则提供一些有吸引力的方法来生成健壮性更强的系统,这些系统在处理不符合语法的情况时也不会出错。本章分析用其他方法构造真实文本中语义解释的若干例子。这些技术涵盖的范围从松散耦合的句法和语义一直到以语义驱动为主并只使用最少句法信息的技术。

第 9 章讨论了以句法驱动的逐条使用规则的语义解释方法,在理论方面很有吸引力。这种方法假设句法和语义之间存在紧密联系,并且可以同时用两者来生成句子的解释。该方法也是那些关注形式语义的研究者最常使用的,借助这种方法,语言学的知识可以很容易地与计算系统集成起来。但如果你调研过现有的大多数计算系统,会发现其中采用了各种各样的方法,而这些方法和不同的工程目的相关。到底是面向长期研究,以解决语言的综合语法为目标,还是面向短期目标,目的是为特定的应用需求开发一套系统呢?一般来说,我们发现长期研究通常采用基于逐条规则的句法驱动的方法,而短期目标倾向于使用其他技术。开发者实际采用的方法主要考虑如下两个重要方面:

- 构造一个句法驱动的逐条使用规则的系统非常复杂,由此产生的很多理论问题将会减慢系统的开发速度。
- 纯句法驱动的系统在处理实际的语言问题时会有很多问题,它们常常是因为错误、比喻以及其他现象引起的不符合语法的现象。因此,对于一个句子来说,如果找不到一个覆盖整句的分析树,句法驱动的系统也许就不能生成任何分析结果。

现在,有很多不同的方法来处理这些问题。其中最极端的一种是完全抛开句法,构造一个直接从句子本身生成语义解释的系统。与此相对应的另一种方法是,仍然采用句法驱动的方法,但对句法处理进行推广,使之可以健壮性更强地处理不符合句法的情况。折中的方法包括,采用部分句法分析来生成一种结构,并将其作为后续语义解释的输入。还有的方法将句法的概念推而广之,使之可以基于语义结构而非句法结构进行分析。本章将针对每种方法给出一些例子。

我们先从句法处理的方法讲起,而后介绍那些越来越不依赖于句法的方法。11.1 节描述一些方法,在这些方法中,句法分析器生成表示抽象句法结构的表达式,以此作为语义解释的输入。11.2 节描述了语义句法,这是一种用语义概念而非句法概念编写语法的方法。11.3 节介绍采用浅层句法分析和模板驱动的语义分析的方法。11.4 节则介绍直接用语义解释句子的方法。

11.1 语法关系

这种方法的思路是用句法分析器产生一个输出,该输出对实际句子进行抽象,忽略细节,但保留了对语义来说很重要的一组语法关系或句法依存。然后,语义解释器在一个单独的解釋过程中生成意义表达式,这个过程用语法关系作为输入。

我们在前面的章节中介绍了基于 ATN 的语法,读者应该已经看到了这种分析的框架。语法关系通常包括逻辑主语 (LSUBJ, logical subject)、逻辑宾语 (LOBJ, logical object)、间接宾语 (IOBJ, indirect object) 等关系,也包括基于介词短语的关系。图 11.1 列出了几个简单句及其用语法关系表示的表达式。每种关系都是(篇章变量 关系 值)的形式,其中,值可以是其他篇章变量或 SEM 结构。

Jack bought a ticket.	(s1 PRED BUYS1) (s1 TNS PAST) (s1 LSUBJ (NAME j2 "Jack")) (s1 LOBJ <A t1 TICKET1>)
A ticket was bought by Jill.	(s2 PRED BUYS1) (s2 TNS PAST) (s2 LSUBJ (NAME j2 "Jill")) (s2 LOBJ <A t1 TICKET1>)
Jill gave Jack a book.	(s3 PRED GIVES1) (TNS s3 PAST) (s3 LSUBJ (NAME j1 "Jill")) (s3 LOBJ <A b1 BOOK1>) (s3 IOBJ (NAME j2 "Jack"))
Jill gave a book to Jack.	(s4 PRED GIVES1) (TNS s4 PAST) (s4 LSUBJ (NAME j1 "Jill")) (GIVES1 LOBJ <A b1 BOOK1>) (GIVES1 TO (NAME j2 "Jack"))
Jill thinks that Jack stole the book.	(s5 PRED THINKS1) (s5 LSUBJ (NAME j1 "Jill")) (s5 LOBJ s6) (s6 PRED STEALS1) (s6 TNS PAST) (s6 LSUBJ (NAME j2 "Jack")) (s6 LOBJ <THE b1 BOOK1>)

图 11.1 基于语法关系的表示形式

值得注意的是,主动语态/被动语态之间的区别用语法关系表示时无法得到体现,因为 LSUBJ 总被视为动作的发出者,而 LOBJ 则是施事的对象,无论它是作为主语(在被动句中)还是宾语(在主动句中)。

对上下文无关文法进行扩展,使之可以在句法分析中生成这种类型的表示形式并不难。一种方法是将每个语法关系表示成一个特征。这样,产生的特征结构很容易转化为三元组。以下特征:

(PRED BUYS1 LSUBJ (NAME j2 "Jack") LOBJ <A t1 TICKET1>)

可以很容易地转成为三元组:

**(s1 PRED BUYS1) (s1 LSUBJ (NAME j2 "Jack"))
(s1 LOBJ <A t1 TICKET1>)**

接下来,语义解释器可以作为一个单独的处理过程,接受语法依存表达式,并且生成语义表达式。我们假定它生成的是以前章节中用到的基于格与角色 (case-role) 的逻辑形式。用模式将语法角色和表达式中所用的格角色关联起来,是一种常见的构造这种表达式的技术。每个动词都可以定义自己的映射。因为语法角色已经刻画了部分语义结构,所以这种映射常常是直截了当的。例如,图 11.2 给出了用于表示图 11.1 中动词所需的模式,其中用到了图 11.3 所示的动词层次体系。

模式		逻辑形式
1. (<VAR> PRED <PRED>)	→	(3 1)
2. (<VERB> AT <LOC>)	→	(AT-LOC 1 3)
3. (<ACTION-VERB> LSUBJ <ANIMATE>)	→	(AGENT 1 3)
4. (<ACTION-VERB> LOBJ <PHYSOBJ>)	→	(THEME 1 3)
5. (<GIVE-VERB> TO <ANIMATE>)	→	(TO-POSS 1 3)
6. (<GIVE-VERB> IOBJ <ANIMATE>)	→	(TO-POSS 1 3)
7. (<ATTITUDE-VERB> LSUBJ <ANIMATE>)	→	(EXPERIENCER 1 3)
8. (<ATTITUDE-VERB> LOBJ <VAR>)	→	(THEME 1 3)
9. (PAST <VAR>)	→	(PAST 1)

图 11.2 用来解释语法关系的一些模式

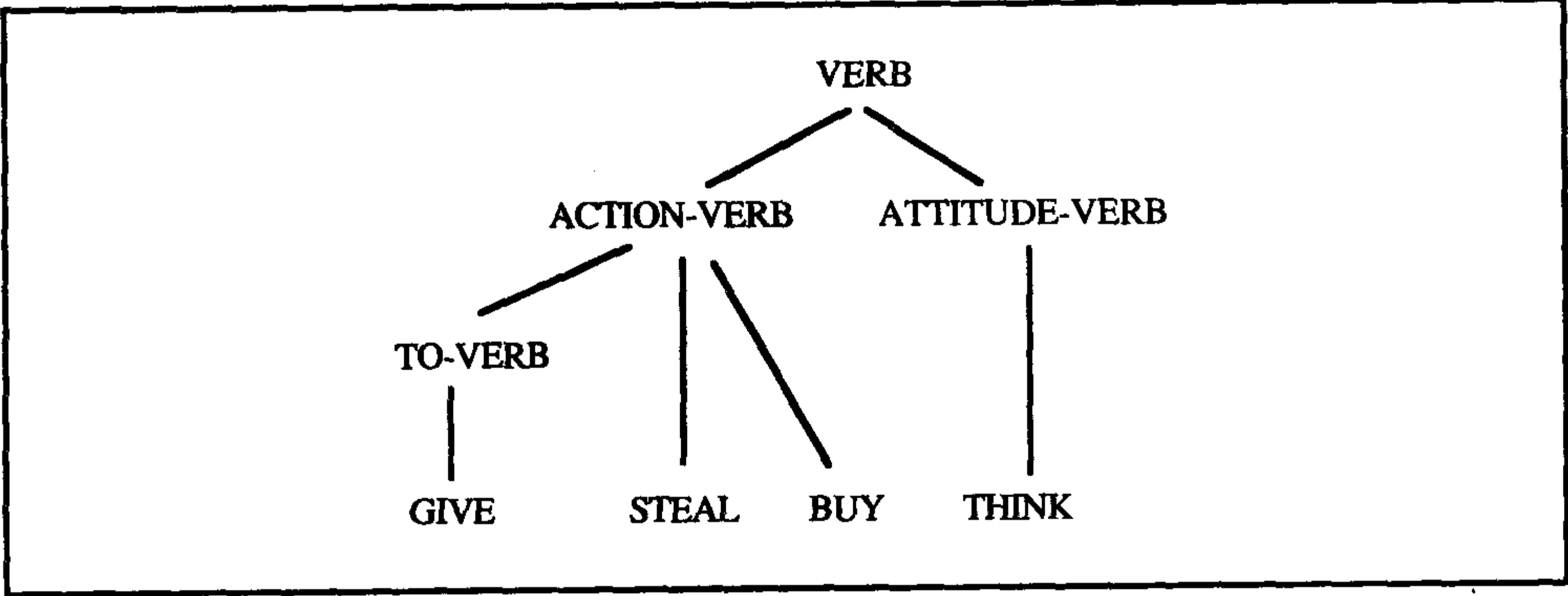


图 11.3 用于模式匹配的动词层次体系

规则由模式构成,其中 < T > 匹配类型 T 的任意元素。右侧给出了语义解释,其中数字 n 表示模式中第 n 个元素的值。例如,规则 2 包含模式(< VERB > AT < LOC >),它匹配第一个元素类型为 VERB,第二个为原子 AT,第三个为类型 LOC 的三元组。并且,会生成一个 SEM 结构,该结构包含了 AT-LOC 和动词的关系,并以位置作为参数。对于以上模式,动词必须按图 11.3 所示进行归类。该层次体系将动词分为 ACTION-VERB(例如,“give”,“buy”,“steal”),ATTITUDE-VERB(包括那些描述态度的动词,例如,“believe”,“think”)和 TO-VERB(包括那些意义涉及到将某个东西传给某人的动词,例如,“give”,“donate”,“throw”)。

假设存在一种机制用来检查模式匹配之后词汇的类型,那么通过寻找和每个词匹配的模式并将结果集放到逻辑形式中,可以解释由语法生成的一组依存关系。例如,考虑句子“Jack bought a ticket”(Jack 买了票),忽略了时态信息之后,产生如下结构:

(s1 PRED BUYS1)
(s1 LSUBJ (NAME j2 “Jack”))
(s1 LOBJ <A t1 TICKET1>)

规则 1 匹配第一个三元组,规则 3 匹配第二个三元组,规则 4 匹配第三个三元组,由此生成的三个逻辑形式片段如下:

(BUYS1 s1)
 (AGENT s1 (NAME j2 "Jack"))
 (THEME s1 <A t1 TICKET1>)

这三个片段合在一起组成了完整的逻辑形式,用本书介绍的缩写形式可将其改写为:

(BUYS1 s1 [AGENT (NAME j2 "Jack")] [THEME <A t1 TICKET1>])

与之相似,句子“Jill thinks that Jack stole the book”(Jill 认为 Jack 偷了书)可以解析成下列依存关系,该依存关系用如下模式生成的语义变换表示:

(s5 PRED THINKS1)	→	(THINKS1 s5)
(s5 LSUBJ (NAME j1 "Jill"))	→	(EXPERIENCER s5 (NAME j1 "Jill"))
(s5 LOBJ s6)	→	(THEME s5 s6)
(s6 PRED STEALS1)	→	(STEALS1 s6)
(s6 TNS PAST)	→	(PAST s6)
(s6 LSUBJ (NAME j2 "Jack"))	→	(AGENT s6 (NAME j2 "Jack"))
(s6 LOBJ <THE b1 BOOK1>)	→	(THEME s6 <THE b1 BOOK1>)

将这些语义变换合起来可生成如下缩写后的逻辑形式(忽略时态):

(THINKS1 s5 [EXPERIENCER (NAME j1 "Jill")]
 [THEME (STEALS1 s6
 [AGENT (NAME j2 "Jack")]
 [THEME s6 <THE b1 BOOK1>])])

这里我们看到,句法和语义过程相互独立且顺序执行。这种假设使得表示起来更加方便,但对于本方法却并非必须的。使句法和语义分析像下面介绍的那样并发执行是有可能的。每当句法分析器构建了一条语法关系,就会传递给语义解释器,而语义解释器可以对其进行处理。如果语法关系没有语义解释,则该信息返还给句法分析器,句法分析器会抛弃生成该关系的句法分析(或将其降级),因为它在语义上是异常的(或者不可靠的)。因而,可以在生成成分的过程中进行语义过滤。这种技术的优点是将句法和语义紧密耦合起来,而且保持了各个模块的独立性。

本节没有涉及如何解释名词短语的问题。很多系统都用一个特殊的处理过程来解释名词短语(有时还需要用上下文判断其指代对象),而处理的结果则插入到依存关系中。也可以对通用方法进行扩展使之可以处理名词短语,比如引入形容词及其修饰的名词之间的依存关系,等等。而语义解释器必须进行扩展以便能递归地对子成分进行语义解释,并用生成的结果构造现有成分的逻辑形式。

基于语法关系的方法是一种很有吸引力的方法,因为表达式提供了句法处理和复杂语义解释过程之间的易用接口,使得它们可以相互独立地进行处理。这就使映射规则可以比逐条使用规则的方法所使用的规则复杂得多。例如,映射过程中可以进行扩展类型检查、推理以及用于消除句子语用歧义的篇章处理。当然,语法关系本身也可以用逐条规则的方法由句法分析器构造组合出来。对这种方法的一种理解是,基于语法关系的表达式是此处使用的逻辑形式的另一种形式,而映射规则进行上下文解释并将其转换为最终的意义表示语言。

11.2 语义语法

在构造特定的应用系统时,常常会用很多技术来提高句法分析和语义解释的效率和性能。为了简化或取消那些通用的歧义处理过程,这些技术利用了应用预设的语境。本节介绍一种技术,用来构造为这种应用量身定制的一种语法。

为表示尽可能多的语言现象,一般英语语法都包含很多结构,而这些结构对于当前特定的应用来说却不是必要的。特定的结构只和特定的语义上下文一起出现。在这样的环境下,我们用特定语义激活的规则来代替语法中的通用句法规则。考虑这样一种应用,它支持对班机航线数据库的查询。下列与班机有关的名词短语会出现在这个论域中:

the flight to Chicago
the 8 o'clock flight
the first flight out
flight 457 to Chicago

要处理这类名词短语,通用语法必须包含如下规则(例子用圆括号括起来)。

NP → DET CNP	(the flight)
CNP → N	(flight)
CNP → CNP PP	(flight to Chicago)
CNP → N PART	(flight out)
CNP → PRE-MOD CNP	(8 o'clock flight)
NP → N NUMB	(flight 457)

现在,对于这个论域中的城市,我们找到如下类型的名词短语:

Chicago
the nearest city to Dallas

这些名词短语可由我们刚才建立的通用语法进行处理,但还需要添加一条处理专有名词的规则。现在的问题是,必须对规则加以限制使其适用于适当的类别。例如,即使规则表明以下名词短语合法,它们也不会出现在这个论域中(但有可能出现在其他论域中):

* the city to Chicago
* the 8 o'clock city
* the first city out
* city 567

要用通用语法处理此类情况,需要在词语上加入选择限制和特征,以限制可能产生的句法结构和修饰成分。但在一个受限论域中,引入像 FLIGHT-N 这样基于语义特征的新的特定词类(也就是那些表示航班意义的名词)常常会更加简单。借助这种词类,通用语法可以改写成如下形式(圆括号中给出了例子)。

FLIGHT-NP	→ DET FLIGHT-CNP	(the flight)
FLIGHT-CNP	→ FLIGHT-N	(flight)
FLIGHT-CNP	→ FLIGHT-CNP FLIGHT-DEST	(flight to Chicago)
FLIGHT-CNP	→ FLIGHT-CNP FLIGHT-SOURCE	(flight from Boston)
FLIGHT-CNP	→ FLIGHT-N FLIGHT-PART	(flight out)
FLIGHT-CNP	→ FLIGHT-PRE-MOD FLIGHT-CNP	(8 o'clock flight)
FLIGHT-NP	→ FLIGHT-N NUMB	(flight 457)
CITY-NP	→ CITY-NAME	(Boston)
CITY-NP	→ DET CITY-CNP	(the city)
CITY-CNP	→ CITY-N	(city)
CITY-CNP	→ CITY-MOD CITY-CNP	(nearest city to
	CITY-MOD-ARG	Dallas)

当然,还需要很多其他规则,而这些规则也要用语义类。例如,FLIGHT-DEST 类可表示用于特定班机目的地的介词短语:

- FLIGHT-DEST → to CITY-NP
- FLIGHT-DEST → for CITY-NP

更高层的句法结构同样可以根据这些类进行定制,比如规则:

TIME-QUERY → When does FLIGHT-NP FLIGHT-VP

用论域的主要语义类表示的语法称为语义语法。显然,句法语法和语义语法之间并非泾渭分明;相反,这两个极端情况之间还存在连续区间。虽然语义语法要比对等的句法语法大得多,但由于上下文受限,所以通常情况下定义规则反而更容易,而且不需要很复杂的特征就可以判断出哪些形式与哪些语义类相符。

可以对语义语法进行扩充,使之用一般的方法就可以生成逻辑形式。通常,解释规则都直接明了,因为我们直接从规则结构就能知道所需的绝大多数语义信息。然而,还有一种方法只须用自身的分析树作为逻辑形式。因为它已经包含了语义信息,所以转换成另外一种表示形式并没有什么益处。例如,图 11.4 给出了查询“*When does the flight to Chicago leave?*”的分析树。

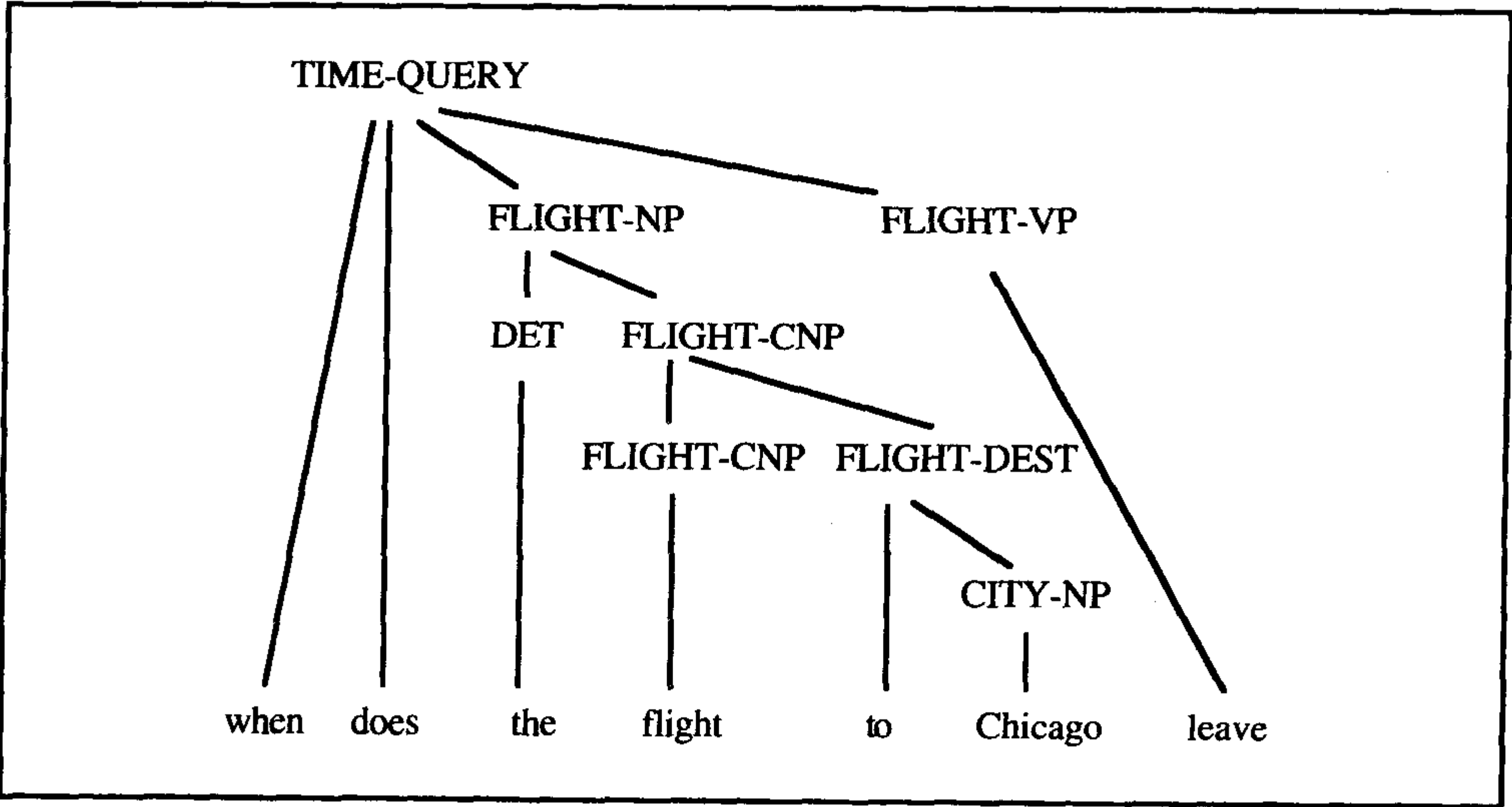


图 11.4 “When does the flight to Chicago leave?”的分析树

采用 LISP 范式,该树可表示成:

```
(TIME-QUERY
  (FLIGHT-NP
    (DET the)
    (FLIGHT-CNP
      (FLIGHT-CNP flight)
      (FLIGHT-DEST to (CITY-NP Chicago))))
  (FLIGHT-EVENT leave))
```

这种结构至少可以和句子的完全逻辑形式一样容易地转化成数据库查询。所以,进一步的语义分析没有什么意义。语义语法将句法、语义和选择限制以一种简单的合一框架进行了整合。为此,语义语法对于受限应用领域的快速开发十分有效。但是,不足之处在于它们无法很好地移植到新的论域,也就不能很好地解决更广泛领域的应用问题。通常,即使一个论域的大部分句法语法都适用于另一个论域,新的论域也需要构建全新的语义语法。

11.3 模板匹配

对特定的应用,任务通常都足够明确,这样就可以使用面向特定目标的技术。该技术利用了论域结构。例如,你所期望的系统是,能够对新闻报道描述的某个股票交易创建文摘。所需的信息可用固定格式表示,比如谁从谁那里以什么价格买了什么。由于系统除了所关心的特定信息以外不需要理解其他信息,因此只须采用基于局部信息的信息提取技术。这种技术虽然只用于解决特定的问题,但生成的系统常常比基于通用目标的技术更有效。因为对于后者,每个句子都必须完全解析,并且在提取信息之前解释。本节将研究一些这样的技术。

设计针对受限领域系统的基本思想是,可以指定一些简单的模板,它们指明了该领域中的关键信息。这些模式可靠地指出了那些用于填充任务模板的信息。例如,在股票交易领域,介词“to”通常指出股票的买进方。在不同的应用中,可以找出不同的解释。例如,在航线日程表查询领域,介词“to”通常指出航班的目的地。

我们通常按如下方式定义需求:领域由一组模板表示,每个模板表示一种可能的输入。在某个领域中系统必须生成一组文摘,例如报告发生在南美的恐怖袭击事件,那么所需的信息就按图 11.5 所示的模板摘出来即可。其基本思想是定义输入模式,这些模式标识出模板中不同的槽。比如,劫持人质的模式为:

```
take <HUMAN> hostage →
(TERRORIST-INCIDENT HUMAN-TARGET 1)
```

它表示任何一个句子,如果其中包含一个由动词“take”后跟一个表示人的短语,而后再跟由单词“hostage”组成的词串,则会包含一个满足 TERRORIST-INCIDENT 模板中 HUMAN-TARGET 槽的值。

要使本方法可用,必须至少将输入解析到可以识别名词短语的程度(比如,能匹配模式中 < HUMAN > 的名词短语),并且要能生成词的规范形式(这样“take”的所有形式都能匹配我们的模式)。在这里,使用第 6 章所述的浅层句法分析技术可以取得不错的效果。

恐怖袭击事件	
DATE	日期
LOCATION	城市/州/国家
TYPE	比如,爆炸
STAGE of EXECUTION	比如,完成、计划
INSTRUMENT	比如,炸弹、枪
PERPETRATOR NAME	比如,“FMLN”
PHYSICAL TARGET	比如,汽车、房屋
HUMAN TARGET	比如,总统
NATIONALITY TARGET	比如,圣萨尔瓦多
EFFECT	比如,没有造成伤亡

图 11.5 恐怖袭击事件摘要的简化模板

当然,其他模式匹配输入的其他部分,并且给出模板中其他槽的值。一旦所有可能的模式都得到匹配,分析的最后一步就是将模板的各个部分合并起来。本节剩余的部分将仔细探讨这个方法的方方面面。

首先,考虑浅层句法分析。如果我们引入一个句法分析器来寻找句子的浅层句法分析结果,那为何不用完全句法分析器来获取尽可能多的信息? 我们有如下几个理由说明在这些应用中这种方法不适用。首先,完全句法分析器计算开销大,尤其是在处理通常包含 30 个或 30 个以上词汇的句子的文本理解任务中,系统每天要处理数以百万计的词汇。其次,现在还不可能为真实的领域构造完整的语法,因此很多句子还无法解析。另外,即使句子能够解析,对于不同的解释,其解析结果也有歧义。

浅层句法分析器利用了这样一个事实,即句子的特定部分可以被相当可信地解析出来。因此,语法只需要处理这些部分,而句子剩下的部分直到模式匹配阶段才进行分析。在第 6 章曾经提到,有两大结构可以正确地检测并解析出来,分别是动词词组,包括的词从第一个助动词开始直到中心动词,其中还包括副词修饰语;以及名词词组,包括的词从名词的前置修饰语到中心词,但不包括名词补足语。另外,分析器还能正确识别功能词,比如介词、语助词、连词,还有一些其他的重要类别,包括专有名词(常以大写字母开头表示)。为了能进行基于语义类的模式匹配,分析器需要计算它所生成短语的语义类型。这些类型通常根据中心词的特征得到。语法 11.6 和语法 11.7 给出了动词词组和名词词组的简单语法,这些词组可用来计算模式匹配中可用的特征 TYPE。图 11.8 给出了例子中需要用到的一个小型词典。

已知这些语法,第 6 章所述的浅层句法分析算法就能找到所有可能的名词词组和动词词组,筛选出完全被同类型的成分包含的 NG 和 VG 成分。未识别词或无法分析的词则被忽略。图 11.9 给出了下列句子生成的所有成分:

Guerrillas attacked Merino's home in San Salvador five days ago with explosives.

下一阶段的分析使用 chart 的语义模式。最简单的模式类型由一串特征组成,通常是 TYPE 特征,这些特征串可以从输入的任何地方进行匹配。如果匹配成功,将会生成模板的一部分。

1. (VG TYPE ?typ LEX ?l) →
(VG1 VFORM {pres past} TYPE ?typ LEX ?l)
 2. (VG TYPE ?typ LEX (?l1 ?l2)) →
(MODAL LEX ?l1) (VG1 VFORM base TYPE ?typ LEX ?l2)
 3. (VG1 TYPE ?typ LEX (?l1 ?l2)) →
(ADV LEX ?l1)
(VG1 TYPE ?typ LEX ?l2)
 4. (VG1 TYPE ?typ LEX (?l1 ?l2)) →
(AUX ROOT have LEX ?l1)
(VG1 VFORM pastprt TYPE ?typ LEX ?l2)
 5. (VG1 TYPE ?typ LEX (?l1 ?l2)) →
(AUX ROOT be LEX ?l1)
(VG1 VFORM {pastprt ing} TYPE ?typ LEX ?l1)
 6. (VG1 TYPE ?typ LEX ?l) → (V TYPE ?typ LEX ?l)
- VG, VG1 的中心特征: VFORM

语法 11.6 动词词组的简单语法

7. (NG LEX ?l) → (PRO LEX ?l)
 8. (NG LEX ?l) → (N TYPE {TIME LOC} LEX ?l)
 9. (NG LEX ?l) → (N AGR 3p LEX ?l)
 10. (NG LEX (?l1 ?l2)) → (DETP AGR ?a LEX ?l1) (N AGR ?a LEX ?l2)
 11. (NG LEX (?l1 ?l2 ?l3)) →
(DETP AGR ?a LEX ?l1) (ADJP LEX ?l2) (N AGR ?a LEX ?l3)
 12. (NG LEX ?l) → (NAMEP LEX ?l)
 13. (NG LEX ?l) → (DATEP LEX ?l)
 14. (DETP LEX ?l) → (ART LEX ?l)
 15. (DETP LEX (?l 's)) → (NG LEX ?l) 's
 16. ADJP → ADJ
 17. ADJP → (V VFORM pastprt SUBCAT _np)
 18. (NAMEP LEX ?l) → (NAME LEX ?l)
 19. (NAMEP LEX (?l1 ?l2)) → (N TITLE + LEX ?l1) (NAMEP LEX ?l2)
 20. (DATE LEX (BEFORE-NOW ?l1 ?l2)) →
(NUMB LEX ?l1) (DATEUNIT LEX ?l2) AGO
- (NG LEX ?L) 的中心特征: TYPE, LEX
NAMEP 的中心特征: TYPE

语法 11.7 名词词组与名称的简单语法

ago	(AGO)
attacked	(V VFORM past TYPE attack)
days	(DATEUNIT LEX days)
explosives	(N AGR 3p TYPE WEAPON LEX explosives)
five	(NUMB LEX 5)
Guerrillas	(N AGR 3p TYPE HUMAN-GROUP LEX Guerrillas)
home	(N AGR 3s TYPE LOC LEX home)
in	(P TYPE IN)
Merino	(NAME TYPE person LEX Merino)
San-Salvador	(NAME TYPE LOC LEX San-Salvador)
with	(P TYPE WITH)

图 11.8 一部样例词典

(NG TYPE HUMAN-GROUP LEX Guerrillas)
(VG TYPE ATTACK VFORM past LEX attacked)
(NG TYPE LOC LEX (Merino home))
(P TYPE IN)
(NG TYPE LOC LEX San-Salvador)
(NG TYPE DATE LEX (BEFORE-NOW 5 days))
(P TYPE WITH)
(NG TYPE WEAPON LEX explosives)

图 11.9 由浅层句法分析器找出的成分

例如,已知下面的模式:

<IN> <LOC> → (LOCATION 2)

无论从哪里开始,都能匹配这种输入,该输入由类型为 IN 的成分后接类型为 LOC 的成分组成。如果相互匹配,标记为 LOC 成分的输入就作为模板中 LOCATION 槽的值。用于分析例句的 4 个模式如图 11.10 所示。这些模式与 chart 进行匹配,生成如下的浅层模板:

(INCIDENT ATTACK
PERP “Guerrillas”
TARGET “Merino’s home”)
(LOCATION “San Salvador”)
(DATE “five days ago”)
(INSTRUMENT “explosives”)

本例的浅层模板中没有冲突,因此,最终的分析结果模板包含了浅层模板表示的所有信息。

当然,并非所有句子都那么容易分析。当领域越来越复杂时,也不得不从更多的结果中选择模式。例如,如果上述系统也要处理武器交易的文章,那么模式 < WITH > < WEAPON > 就不总是指攻击的工具,还可能指装载的货物。比如,句子“A truck loaded with explosives was stopped at the border”就属于这种情况。要进行处理,需要更复杂的模式以包含待分析的动词。例如,用下述规则,模式可以跳过一些成分,比如:

<ATTACK> ... <WITH> <WEAPON> → (INSTRUMENT 3)

该模式可以匹配任何由“attack”类型的动词后跟介词短语组成的句子。而其他模式,比如:

<CONVEYER> <LOAD, pastprt> <WITH> <WEAPON> →
(CARRIER 1 CARGO 4)

能处理其他的意义。需要注意,除了类型限制之外,模式也可使用其他特征。比如,我们可能不想用同样的模式来匹配句子“The guerrillas were attacked by the police”和“The guerrillas attacked the police”。当模式更复杂时,该模式可以区分出关系从句与主句之间的信息,因为每个从句都描述单独的事件。例如,句子“The weapons, which were smuggled in by the guerrillas, were destroyed yesterday”应分析为两个事件,而不是由游击队走私武器和武器被摧毁这两个事件组合成的一个事件。某些系统允许用任意有限状态自动机确定模式,这给了程序员相当大的权力来处理这种复杂的情况。一旦完成这一步,除了从预分析的成分开始处理之外,系统和语义语法方法看起来已经很相似了。

P1	<HUMAN> <ATTACK> <LOC>	→	(INCIDENT ATTACK PERP 1 TARGET 2)
P2	<IN> <LOC>	→	(LOCATION 2)
P3	<DATE>	→	(DATE 1)
P4	<WITH> <WEAPON>	→	(INSTRUMENT 2)

图 11.10 一些样例模式

框 11.1 消息理解系统(Message-Understanding System)的评测

正如本节一开始提到的那样,基于模式技术的一个主要应用是从文本中进行信息提取。在这种应用中,系统需要处理海量文本(比如美联社新闻电讯)并提取出适用于和特定话题(比如股票交易)相关的任何文本的模板。

研究者已经对面向这种需求的系统进行了评测,并且在一系列年度会议(MUC大会)上进行报告。测试的系统综合使用各种模式匹配技术和通用技术以生成文章的分析结果。此类系统可以在几个方面进行测评,包括:

- 系统召回率——所识别的关于这个话题的文章有多少? 所能填充的模板有多少?
- 系统准确率——生成不正确模板或模板中不正确条目的频度是多少?
- 系统效率——处理一组文章要多长时间?

在 1993 年的评测中,系统有 1500 篇训练文本,所有文本都标记了指示最佳答案的模板。然后,系统在 100 个以前未出现的消息上进行测试。最佳系统取得了 62%的召回率和 53%的准确率。基于模式的系统比通用系统的效率高得多,每分钟能够处理几千个词。要了解 MUC 评测的更多信息,请参考 Chincor 等(1993)。

当然,实用系统还要进行额外的处理而不是简单地用词填充槽。例如,要对日期进行推理,用文章的日期计算短语“five days ago”指的是哪一天。另外,还要在分析中构造更为丰富的名词短语语义表达式,这些分析结果用做模板中槽的值。如果模板按照动词及其论旨角色而非事件组织,那么,还要用同样的技术来构建句子的逻辑形式。

基于这些技术的系统健壮性很强,能为几乎所有的输入生成某种解释。另一方面,这些系统的分析很浅,因此很容易发生错误。但根据目前研究的情况来看,在可预见的将来,这类技术在实用系统中还将扮演重要角色。研究中要解决的一个很大的问题是,如果这些问题越积越多,会发生什么情况。如果需要对范围很广的话题提取信息,或者要求对那些很难用简单模板表示的内容提取更详细的摘要,该怎么办?

有些系统尝试在两种情况下都取得最好效果。首先,使用完全句法分析器和语义解释器,只要分析失败,就用模式匹配技术来提取可以从句子中提取的信息。这种技术的优点是能够更仔细地处理某些句子,而在完全解析失败时仍保持系统的健壮性。其缺点与计算效率有关。如果每天要处理数百万个句子,浅层分析和模板匹配可能才是真正可行的技术。

11.4 语义驱动的分析技术

上一节我们使用了有限句法预处理器来提供语义解释的输入。也可以不经过句法处理阶段,除了使用基本的词语形态分析之外,其他一切都用语义驱动的模式来完成。本节简要介绍这种技术。

在这种系统中,语法和语义信息存储在词条中。具体地说,词典中包含了一些重要信息,比如词语的各种可能意义,其中包括动词和形容词的格框架信息;还有一些规范,根据这些规范可以进行语义消歧,并且可以将词语和其他词组合起来集成到更大的语义结构中。这里,我们分析一个系统,其中,信息都存成一组模式-行为规则,对和第 6 章所述的确定型分析器相似的缓冲区进行处理。缓冲区包含输入成分,并由模式-行为规则进行更新。只要我们输入一个新词,该词的词典条目中的规则就激活。例如,和词“booked”相关的规则如图 11.11 所示。每个规则包含一个对语义类进行检测的模式以及出现在缓冲区中的各个词。若某个模式匹配成功,和这次匹配相关的缓冲区元素就被移除,用动作部分指定的结构将其替换。除此之外,规则的动作部分还可以激活或者停用附加的规则。例如,图 11.11 中的规则 BOOK.1 和带 ANIMATE 类型条目的缓冲区后跟包含词“book”的缓冲区相匹配,用(RESERVING * [AGENT 1])条目替换这两个缓冲区。其中,* 可实例化为新的篇章变量。正如上一节所述,解释规则中的数字用来指示和模式匹配的缓冲区条目。因此,1 指的是和该匹配相关的第一个缓冲区的值。操作符(^)用于将两个浅层描述合并起来构造一个新的解释。

考虑句子“John booked me a flight to Chicago”如何用上述规则进行解释。现在,假定所有的 NP 在本次解析开始前都已经预先处理好了。一开始,分析器只有一条激活规则 S.end(检查句子是否处理到结尾了)、前两个填好词的缓冲区(NAME j1 HUMAN “John”)以及“booked”。激活 booked 的规则,然后是 BOOK.1 匹配成功。用下面的值替换前两个缓冲区:

(RESERVING r1 [AGENT (NAME j1 “John”)])

下一个缓冲区用结构(PRO m1 HUMAN me)填充。这一次,规则 BOOK.3 匹配成功。其动作是将第一个缓冲区的值和新的结构合并起来,生成下面的结构作为缓冲区 1 的新值:

```
(RESERVING r1 [AGENT (NAME j1 "John")]
  [BENEFICIARY (PRO m1 ME1)])
```

下一个输入是名词短语“a flight to Chicago”的分析结果 < INDEF1 f1 FLIGHT(TO-LOC f1(NAME c1 “Chicago”)) >。规则 BOOK.2 匹配成功,所得结果再次和第一个缓冲区合并,生成分析结果如下:

```
(RESERVING r1 [AGENT (NAME j1 HUMAN “John”)
  [BENEFICIARY (PRO m1 HUMAN “me”)]
  [THEME <INDEF1 f1 FLIGHT
    (TO-LOC f1 (NAME c1 “Chicago”))>]])
```

当读入最后的句号时,规则 S.end 触发,表示分析结束。

```
BOOK.1
  <ANIMATE> “book” → (RESERVING * [AGENT 1])
BOOK.2
  <RESERVING> <TRANSPORT> → 1 ∧ (RESERVING * [THEME 2])
BOOK.3
  <RESERVING> <ANIMATE> → 1 ∧ (RESERVING * [BENEFICIARY 2])
BOOK.4
  <RESERVING> “for” <ANIMATE> → 1 ∧ (RESERVING * [BENEFICIARY 2])
S.end
  <ANYTHING> “.” → pop
```

图 11.11 样例词典

要对这个解释器进行扩展,使它能分析名词短语,需要引入一个和确定型分析器中的注意力转移相似的机制。举例说明,a 的规则是:

```
ART.1 “a” → INDEF1
```

在前面的例子中,如果名词短语“a flight to Chicago”没有进行预处理,分析器将会到达下面这个位置:

```
Buffer 1: (RESERVING r1 [AGENT (NAME j1 “John”)
  [BENEFICIARY (PRO m1 ME1)])
```

```
Buffer 2: a
```

现有的活动规则(BOOK.2, BOOK.4 和 ART.1)中没有一个能匹配这种情况。但是,如果允许模式在第一个以外的其他位置进行匹配,规则 ART.1 就能匹配成功。我们接受这种匹配,从而生成了一个注意力转移。激活的规则临时从活动表中删除,分析继续进行,好像缓冲区 2 是第一个缓冲区一样。现在,分析器又到了另一个位置,在这里要用图 11.12 所示的“flight”,“to”和“Chicago”的规则来构造 NP 的分析。当 NP 完成之后,分析器的原始状态又被一个称为弹出的新动作恢复。我们跟踪这次分析,它从前面所示的词 a 开始进入第二个缓冲区的那一点并

继续进行下去。规则 ART.1 触发,生成一个注意力转移,结果规则 BOOK.2 和 BOOK.4 临时失效,缓冲区 2 作为匹配的第一个缓冲区。当输入词“flight”时,得到如下结果:

```
Buffer 1: (RESERVING r1 [AGENT (NAME j1 "John")]  
          [BENEFICIARY (PRO m1 ME1)])  
Buffer 2: INDEF1  **patterns start matching here**  
Buffer 3: flight
```

活动规则是 FLIGHT.1 和 FLIGHT.2。FLIGHT.1 匹配成功,用 < INDEF1 f1 FLIGHT > 替换了缓冲区 3 的内容。下一步,词“to”被送到缓冲区 4,但没有模式能与之匹配。下一步是输入“Chicago”,规则 CHICAGO.1 触发,立刻用词(NAME c1 “Chicago”)替换该词。现在,缓冲区看起来是下面这种情况:

```
Buffer 1: (RESERVING r1 [AGENT (NAME j1 "John")]  
          [BENEFICIARY (PRO m1 ME1)])  
Buffer 2: <INDEF1 f1 (FLIGHT f1)>  **patterns start matching here**  
Buffer 3: to  
Buffer 4: (NAME c1 "Chicago")
```

规则 FLIGHT.2 匹配成功,用值 < INDEF1 f1(FLIGHT f1 [TO-LOC(NAME c1 “Chicago”)]) > 替换缓冲区 2。接下来,句号送到缓冲区 3,规则 NP.end1 触发,执行一次弹出,将分析器重置到 NP 开始之前的状态。规则 BOOK.2 和 BOOK.4 被重新激活,分析按前面所示的那样继续进行。

ART.1	“a” → INDEF1 (活动规则 NP.end1 和 NP.end2)
FLIGHT.1	<DET> “flight” → <1* (FLIGHT *)>
FLIGHT.2	<FLIGHT> “to” <LOC> → <?a ?f (FLIGHT ?f [TO-LOC 3])>
MAN.1	<DET> “man” → <1 * (MAN *)>
CHICAGO.1	“Chicago” → (NAME c1 “Chicago”)
ME.1	“me” → (PRO * HUMAN “me”)
NP.end1	“.” → pop (标志一个NP结束的句点)
NP.end2	<VERB> → pop (标志主语NP结束的动词)

图 11.12 用于名词短语分析的一些规则

虽然在这种框架下可以很快构建用于处理特定句子集的分析器,但如果将这些系统作为分析的通用模型,则问题会随之而来。因为所有的规则都是按各个词进行编排索引的,因此很难用一种简便的方法来刻画语言学的一般规律。例如,规则 BOOK.1 将找到的第一个名词短语识别成动作 RESERVING(“booked”的一个意义)的 AGENT。但这只是应用于所有动作动词的一般性规则的一个例子而已。当规则需要从词典条目中存取时,必须对每个行为动词指定一条这样的规则。显然,构造这么大的语法是一项繁重的工作。

更重要的是,这些系统只能用局部句法信息,因为它们维护的惟一状态只包括特定的当前格框架结构和当前的输入。因此,要对词进行排歧,最好考察其前后的几个词。实际上,当这些规则越来越复杂时,它们适用的情况就越来越少,从而需要添加同样复杂的规则来处理简单的句法变量。例如,动词“booked”早前的定义只考虑它作为简单过去式的句子主动词的用法。

实际上,“booked”还可以作为过去分词,因此可以用在被动句中,或者可以引入关系从句,比如“The flight booked by the travel agent leaves at three”。解决这些缺陷的方法通常是恢复那些能用于帮助解释复杂句的句法成分。通过先后运行解释器和句法分析器,可以维护句法上下文。但是,分析器的动作是由语义分析器发出。由语义分析器确定的句法操作是否是下一个可能的步骤,对于这个问题,句法分析器只是简单地给出答案。如果不是,语义解释器就尝试寻找另一个不同的分析。

但是,我们还不清楚语义分析器是否能用一种简单的处理方法来处理复杂句。当句法分析器控制处理过程时,句法成分可用于在激活语义分析器之前消除这些复杂问题。如果用这里介绍的控制策略,语义分析器本身也必须能处理这些复杂问题,而且只有如此才能得到与句法成分相吻合的分析结果。

11.5 小结

除了第9章讨论的逐条使用规则方法之外,还有很多不同的策略可用于将句法和语义信息组合起来。这些方法通常用模式匹配技术对句子进行语义解释,区别在于句法分析进行到何种程度。一种方法是用完全句法分析来计算句法和语义之间一种称为语法形式的中间表示法。该表达式随后由一个单独的语义解释过程进行解释。另一种方法是用句法预处理器生成句子的浅层分析,然后对输出进行语义解释。还有一种方法无需任何句法处理,直接将语义模式和输入进行匹配。有一种稍微不同的方法使用直接由语义结构确定的语法,得到一个语义语法。

每种方法各有其优点。通常,语法用得越少,系统与特定领域越相关。这使得我们可以比较快地构造一个健壮性强的系统,但很多细节部分在解释句子的过程中可能会丢失。除此之外,这类系统通常无法很好地推广到新的领域。而在某些应用中,领域相关的模式匹配方法在可预见的未来可能是取得较好效果的惟一方法。

框 11.2 概念分析:语义驱动的分析器

很多语义驱动的分析器都是在耶鲁自然语言理解小组工作的基础上发展起来的。Birnbaum 和 Selfridge(1981)所述的分析器可能是介绍这种方法的最好文献。该分析器由一个基于格的表示方法驱动,该方法称为概念依存。其关键思想是特定的词(尤其是动词)标记了包含格的意义结构,而分析器的主要任务是识别这些词,并用句子的其余部分来填充格的值。虽然分析器的细节部分在形式上和本节描述的语义驱动的分析器有所不同,但进行的操作类型却很相似。系统由一组和词条相关的称为请求的模式-动作规则组成。当读入一个词的时候,它的请求是活动的。通常,一个请求用于测试两种信息,既可以检查输入中的特定词或短语,也可以检查缓冲区里结构的特定语义属性,称为 C-LIST。可接受的行为包括显式出现在表达式中的行为。更具体地说,可以向 C-LIST 添加新项,填充 C-LIST 中某结构的槽(通过合并结构执行的一项功能),激活或者停用其他请求。

11.6 相关工作与深入阅读材料

RUS 系统(Bobrow 和 Webber, 1980)描述了一个基于 ATN 的构造语法关系的语法。它允许在激活语义解释器的边上执行某种操作,该解释器能用增量式方法构造句子基于格框架的语义解释。语义解释器可以接受也可以拒绝不同的边,因此,影响到句法分析的行为。Woods (1980)提出了这种交替进行的解释器的更形式化分析,Woods 用到了一种称为层叠 ATN 语法的形式化方法。另一个用到交替语义处理的基于 ATN 的系统在 Ritchie(1980)中有更详细的描述。

基于语法关系的表示方法,Hirst(1987)给出了语义解释的方法范例。其中,综合了多种方法将语法关系解释成基于框架的表达式。另一个好的例子是 KERNEL 系统(Palmer 等 1993)。语法关系的语言学处理就是词汇功能语法(LFG, lexical functional grammar)(Kaplan 和 Bresnan, 1982)。在词汇功能语法中,句法规则使用语法关系的特征,再用逐条规则的形式应用于语义解释。

在 20 世纪 70 年代,语义语法作为构造处理受限领域且健壮性强的系统的一项技术提了出来。这项技术首先用于 SOPHIE 系统(Brown 和 Burton, 1975),该系统用于调试电路的演示。语法有终结类型,比如 REQUEST, TRANSISTOR, JUNCTION/TYPE, 等等。每条规则都和一个可生成执行相应操作的 LISP 代码的函数相关。LIFER 系统(Hendrix 等, 1978)应用相同的技术来处理数据库查询应用,采用和各个数据库相关的概念为相应的数据库构造不同的语法。

在文本摘要中,最早应用浅层理解技术的是 FRUMP 系统(DeJong, 1982)。随着大家开始关注消息理解应用,出现了很多这方面的文献。11.3 节所述的技术基于 FASTUS 系统(Appelt 等, 1993)。该系统使用一系列有限状态机来识别名词和动词词组,实现了信息提取的模式。另一个好的例子是 SCISOR 系统(Jacobs 和 Rau, 1990)。

Wilks(1975)提出了一个模板驱动的语义解释和最小句法处理相结合的出色例子。系统的语义信息包括一组定义了系统中各种语义关系的语义模板,以及每个词条的语义公式。系统必需的操作是将这些模板和句子的表达式相匹配,表达式中基本的短语结构已经提取出来了。11.4 节中的语义驱动的分析器仿照 Birnbaum 和 Selfridge(1981)(参见框 11.2)提出的概念分析器构造,而概念分析器本身也是更早的解释器(Schank, 1975; Riesbeck 和 Schank, 1978)的变体。Lytinen(1986)描述了一种有趣的语义驱动的分析器,它用句法成分对分析结果进行验证。

11.7 习题

1. 【易】用图 11.1、图 11.2 和图 11.3 中所示的数据,给出句子:

A ticket was bought at the theater.

用语法依存表达式生成的逻辑形式。

2. 【易】用 11.2 节所述的语义语法生成查询“*When does the 8 p.m. train from Boston arrive in Chicago?*”的分析树。为了使语法能表示该句子,写出那些需要添加的附加规则。
3. 【中】扩展 11.1 节中所述的基于语法关系的方法,使之可以处理受益人的情况,比如:

Jack bought me a ticket.

Jack bought a ticket for me.

给出每个句子的分析结果,以及由你的模式所生成的最终的逻辑形式。

4. 【中】完成 11.2 节所描述的语义语法,使之可以处理该节中所有名词短语的例子。再给出 4 个名词短语(两个规范的和两个不规范的),它们的结构和你的语法不同但也要求能正确处理。设计一组特征,在记录了纯句法语法时也能记下在你的语义语法中发现的限制。用这些特征给出一个句法语法,使其等价于你的语义语法。在构造一个新的领域时哪一个更易使用? 哪种语法更适用于新的领域?
5. 【中】扩展语法 11.6 和语法 11.7,给出一组基于模式的解释规则,要求能从下面关于恐怖袭击活动的应用的多个句子中提取必要的信息。可以假定图 11.5 给出的模板能确定这个应用中需要的所有信息。对每个句子,用和图 11.9 相似的格式,指出由你的语法生成的成分序列。对每句话给出由你的模式构造的输出。每个句子视为一个单独的输入,而不是一个段落的一部分。
 - a. Guerrillas used explosives to attack San Salvador yesterday.
 - b. No one was injured when the guerrillas bombed the government buildings.
 - c. In San Salvador several terrorists were thwarted in their plan to bomb the home of the president.

给出在定义这些模式时面临的至少一个难题,并且描述你的解决方案可处理的范围及其局限性。

6. 【中】实现一个消息理解系统中的模式匹配部分,要求能够解释习题 5 中的那些句子,假定已给出预分析成分的输入序列。用习题 5 中的句子,加上你挑选的用于验证功能范围的其他 3 个句子来测试你的系统。
7. 【中】设计一组 11.4 节所述的那种模式-操作规则,要求这些规则能像习题 5 所述的那样完成同样的任务,但不需要用到任何句法处理。需要解决的最困难的问题是什么? 哪些比用句法处理器容易? 基于直接句法解释的方法有什么优点吗?

第 12 章 辖域指定与名词短语的解释

本章讨论语义解释的其他问题,其中大部分都和名词短语的解释有关。更具体地说,本章讨论的问题包括量词与运算符的辖域指定,基于句子结构的互指约束生成,以及形容词短语和其他名词短语中的修饰语解释方面的各种问题。

12.1 节讨论量词辖域判定方面的问题。12.2 节进一步探讨并分析定指描述的特征,它有时和量词短语比较相似。而 12.3 节提出一种技术,用于在句法分析过程中计算可能的辖域的排序结果。12.4 节介绍一些一般性问题,和代词及其他指代性名词短语之间的互指约束相关,并描述了一种在分析中生成这种约束的技术。12.5 节分析对形容词短语和其他名词修饰语进行语义解释的问题。12.6 节讨论了一类特殊的名词短语,与关系名词和名词化有关。最后,12.7 节简要描述了前面没有谈到的其他重要问题。

12.1 辖域指定现象

本节讨论语言中的辖域指定方面的问题。辖域歧义与句中的量词、逻辑运算符、情态运算符和副词有关。不太严格地说,可以将每种有辖域指定现象的结构视为一个运算符。判断逻辑形式中的量词和运算符的辖域是一个非常复杂的问题,不仅要用到词法、句法和语义信息,还深受上下文的影响。但是,也有一些约束和解释的选择不需要考虑上下文。

辖域指定是语义解释中广泛存在的问题。它的存在可以用一些由辖域指定现象引起歧义的句子来说明。任何包含两个或多个量词的句子都可能有歧义,例如:

A dog entered with every man.

至少描述了三种不同的情境:(1)有很多狗,每一条都有一个人陪伴;(2)一条狗走了进来,同时所有的人也一起走了进来;(3)一条狗进来了很多次,每一次都跟一个不同的人一起进来。这几种解读是由两个量词的相对辖域以及和每个事件相关的隐含存在量词导致的。逻辑运算符和量词的辖域也可能导致歧义,比如:

We didn't see every dog.

既可以说成我们一只狗都没有看见,即:

(EVERY **d1** : (DOG **d1**) (NOT (SEE1 (PRO **w1** WE1) **d1**)))

也可以说成至少有一只狗我们没看到,即:

(NOT (EVERY : **d1** (DOG **d1**) (SEE1 (PRO **w1** WE1) **d1**)))

由逻辑连词引起歧义的例子是:

Everyone thought that Fido or Fifi would win.

一种理解是大家认为 Fido 会赢或者大家认为 Fifi 会赢,另一种理解是大家认为要么 Fido 要么 Fifi 会赢。其中的差别就在于全局量词与析取命题之间的辖域判定。

时态运算符和副词也有可能造成歧义,比如:

He will feed the hungriest dog tomorrow.

这句话的意思既可能是说虽然狗现在很饿,但以后再喂它(可能那时它已经不饿了);也可能是说明天要喂狗的时候它才很饿(可能现在不饿)。句子:

A fat dog always loses the race.

用来说明副词的辖域判定歧义。它可能说输家总是太胖的狗,也可能说某条大胖狗跑得太慢以至于它总是输掉比赛。

12.1.1 对量词进行分类

名词短语在语言中有多种不同的功能,在分析辖域指定问题时,区分这些功能非常重要。我们至少需要分析三大类。和定指性指代相关的那一类是指,听者原则上应能确定的对象或集合。定指性指代与“the”这样的限定词以及所有格限定词一同出现,前者如“the dog”(个体)或“the fat men”(特定集合),后者如“their ideas”或“John’s book”。第二类量词是存在或不定指量词,通常在会话中引入新的个体或集合。非定指性指代的典型例子是“a”这样的限定词(比如“a good book”)以及由“some”,“several”这样的量词构成的不确定集合。这里有一个存在量词的实验,测试它们是否能出现在如下形式的句子中:

There are Q men who like golf.

其中,Q可以是任何存在(复数)量词,比如“some”,“many”,“a few”,“several”,“two”,“seven”,“no”,等等。注意,诸如“all”,“each”,“every”和“most”这样的量词在这种上下文里是不合规范的。这几个量词构成第三类,即全称量词,它们指的是一个集合的全部或接近全部成员。

这些和集合相关的量词的另一个重要的区别性特征是整体性/个体性。整体性解释将集合视为一个整体,比如“The men met at the corner”(这些人在拐角相遇)。其中,相遇的是人的集合。单纯说这个集合中的某一个人相遇是没有意义的。个体性解释则涉及到集合中的所有成员,比如“Each man bought a suit”(每人买了一套衣服)。其中,每一个人各自都买了一套衣服,而不是他们合起来作为一个整体买了一套衣服,但在“The men bought a suit to give to Harry”(这些人买了一套衣服送给 Harry)中后一种解释更为合理。某些量词只支持某一种解读(比如,“each”必然是个体性的),但大多数都是有歧义的,也就是说,既可以是个体性的,又可以是整体性的。分析以下和集合相关的四个句子:

Each man lifted the piano.

Every man lifted the piano.

All the men lifted the piano.

The men lifted the piano.

第一个句子只有个体性解读,即每个人都能自己抬起钢琴。而最后一个句子则明显是整体性解读,即这一组人一起抬起钢琴。其他例子都不外乎这两种解读,可以用任何一种来解释。

对个体性或整体性解读的选择影响到可能的辖域指定判定,因为其他量词可能只在个体性解读中才需要名词短语来判定。例如,分析句子“Each man lifted a piano”。这里指的可能是

很多不同的钢琴,每一架对应一个人。我们可以说名词短语“a piano”取决于名词短语“each man”,或者说“each man”的辖域比“a man”大,从而得到量词以一阶逻辑判定辖域的方法。只有辖域更大的名词短语是个体性的时候,这种决定关系才可能成立。如果第一个名词短语是整体性的,比如:

Together, the men lifted a piano. (这些人一起抬起一架钢琴。)

就不存在这种决定关系,短语指的就是一架钢琴。当然,很多结构还是有歧义的,比如:

A piano was lifted by each man.

对它的理解模棱两可,既可以理解成有很多架钢琴,也可以理解成一架钢琴被抬过很多次。需要注意的是,这两种情况下名词短语“each man”都是个体性的。但是名词短语“a piano”只在第一种解读中才由它来决定。第二种情况下钢琴还是同样的钢琴,但是每一个都被分派给一组“抬起”事件(每一个事件中由一个人抬)。

12.1.2 进行辖域判定

量词辖域判定问题通常伴随完全辖域问题一同出现,后者的目的是寻找嵌套形式(在一阶谓词演算中也可以找到)中的运算符。但需要注意的是,这种完全辖域指定形式并不一定取决于某个变量辖域以外的变量。例如,一个不定指名名词短语序列,比如“a man lifted a piano”,无论辖域如何确定,表达的意思都一样。必须用一阶谓词演算语法的表示形式来进行排序。还有一种可用的表示方法,其中,只有必要的依存关系才表示出来,没有经过排序的变量彼此不相关。这种表示方法的优点是,句子不会比它实际表达的意思有更多的歧义。但为了表示起来更简单一些,我们在本节的剩余部分假定辖域指定问题就是构造量词的完整序列。

要更仔细地分析辖域指定现象,比较有用的方法是定义成分的局部论域。成分 C 的局部论域是包含 C 的最近的 S 或 NP 所包含的成分集合。分析图 12.1 中的树。由顶端的 S 成分定义的局部论域包含 NP₁ 和 NP₂,而由成分 NP₂ 定义的局部论域包含 NP₃ 和 NP₄。用于定义这个论域的 S 或 NP 成分称为该论域所包含成分的支配成分。

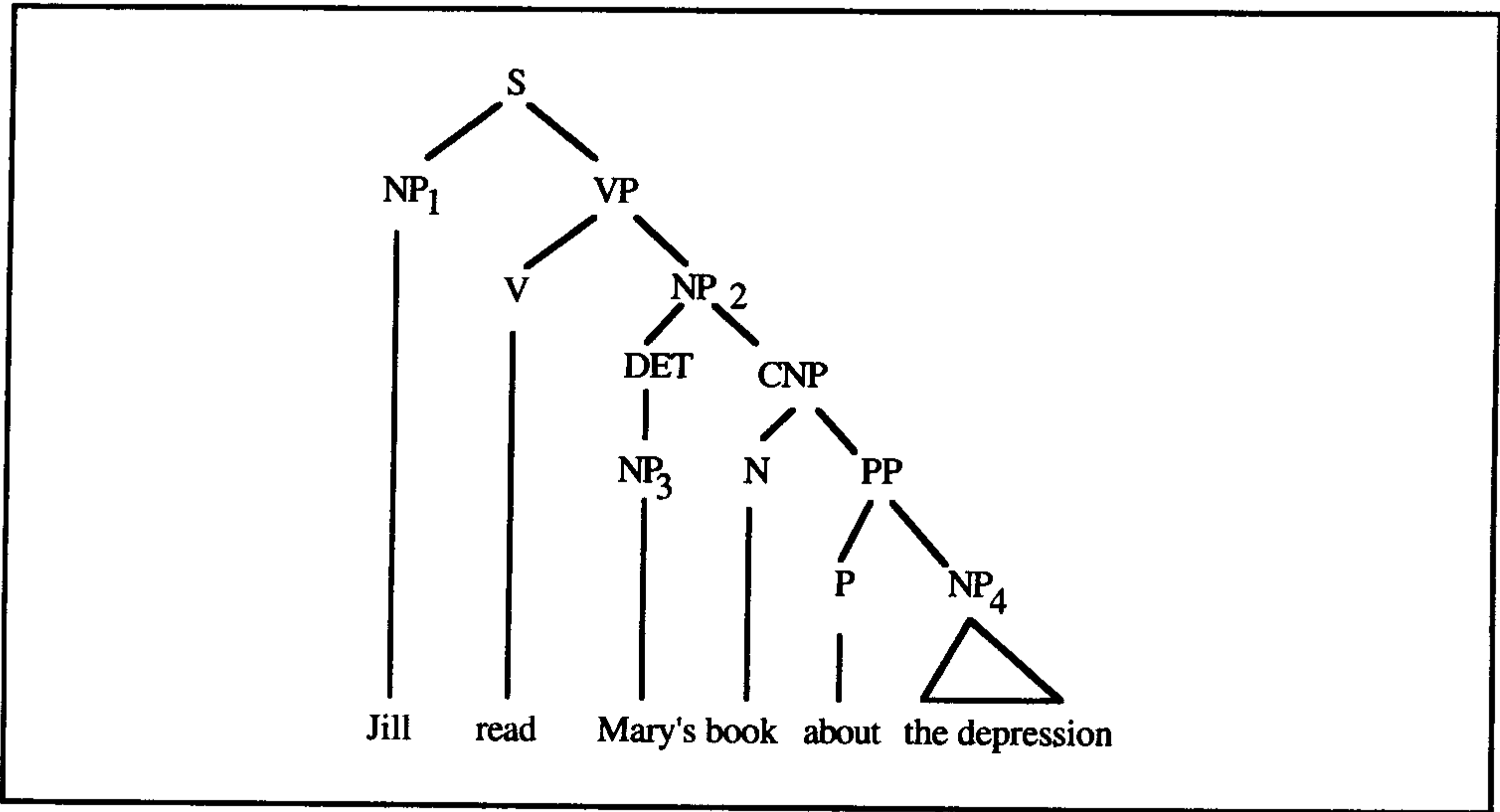


图 12.1 用于论证局部论域的分析树

如果两个成分属于同一个局部论域,则它们横向相关。如果第一个成分是第二个成分所在局部论域的支配成分,则它们纵向相关。这个术语也可以扩展到量词;即,如果包含两个量词的成分横向相关,就说这两个量词横向相关。

12.1.3 横向辖域判定

处理横向辖域判定的一种常见方法是比较每一对运算符,看看是否存在一种模式,其中一个辖域比另一个更大。这种选择通常表示成量词的“强度”,“较强”的量词通常比“较弱”的量词辖域大。很多研究人员提出如下所示的量词强度的层次关系:

each > every > all, some, several, a

在问答系统中很重要的一点是,特殊疑问词也可以赋予一个强度。多数问答系统在“each”和“every”之间插入一个特殊疑问词,如下句所示:

Who saw every dog?

Who saw each dog?

第一句理解成找一个人,他看到了所有的狗;第二句理解成找一组人——一个或更多的人,他(们)是否看到了每一条狗。

然而,最终可以证明这种单一的优选策略在很多情况下很不准确,因为没有考虑量词间的结构关系。例如,首选可能随着与量词共现的主语发生改变。分析下面的句子:

Every man saw a dog.

A man saw every dog.

第一句的首选解读是每个人看见一条不同的狗(即,“every”的辖域蕴涵“a”)。而第二句的首选解读指的是一个人(即“a”的辖域蕴涵“every”)。因此,句中量词的位置也影响到辖域的判定。有人提出了下面这种基于位置的辖域范围的一般优选策略,不过这种优选方法可能不适用于特定的量词:

前置成分

> 表层主语

> 后置状语

> 直接/间接宾语

12.1.4 横向辖域歧义解析

生成完全辖域逻辑形式的算法通常可用一项操作来完成,这项操作将量词从非辖域形式中取出,放到其完全辖域形式中。这样,一个过程包括从公式中取出定量表达式,做标记,最后用原来包含该词的介词把它再包起来。通过改变量词取出的次序,能生成不同的辖域。例如,已知无辖域形式:

(<PRES LOVES1> <EVERY m1 MAN> <A d1 DOG>)

通过将 d1 取出,得到首选解读,即:

(A d1 : (DOG d1) (<PRES LOVES1> <EVERY m1 MAN> d1))

然后,取出 **m1** 得到:

(EVERY **m1** : (MAN1 **m1**)
(A **d1** : (DOG **d1**) <PRES LOVES1> **ll m1 d1**))

最后,取出时态运算符,得到完全辖域逻辑形式:

(PRES (EVERY **m1** : (MAN1 **m1**)
(A **d1** : (DOG1 **d1**) (LOVES1 **ll m1 d1**))))

另一方面,如果先取出的是 **m1**,然后取出的是 **d1**,会得到一种不同的解读。我们的问题是研究一种算法,用权值来选择首选解释。

有时,用别的表示法解决问题更容易,因为这种方法只是简单地记录量词的次序。更确切地说,用一个列表来表示属于无辖域逻辑形式的量词序列。因此,完全辖域指定逻辑形式中的辖域先后次序可以记为[**m1 d1**]。

各种文献中提出了很多算法,这些算法试图用启发式方法将量词强度和句法角色优选组合起来。例如,首先按句中出现的先后次序列出连词,这样得到一个初始序列[$V_1 \dots V_n$]。然后,如果变量 V_{i+1} 的量词比变量 V_i 更强,就交换变量。这种交换一直进行到不再有交换发生为止。

12.1.5 纵向辖域关系

在分析纵向关系时,这种辖域判定的过程更复杂。具体来说,嵌套成分中的量词可以纵向提升到其支配运算符之上,然后根据所在的新层次确定辖域。因此,纵向辖域指定算法可能会改变那些需要用横向辖域指定算法确定辖域的量词集。量词是否能提升取决于句法结构是否会产生纵向辖域关系。例如,句子:

Some man rewarded a boy who gave each dog a bone.

有两个限定性从句,用逻辑形式表示为:

(<PAST REWARDS1> **r1**
 <SOME **m1** MAN1>
 <A **b1** (& (BOY1 **b1**)
 (<PAST GIVES1> **g1 b1**
 <EACH **d1** DOG1> <A **b2** BONE1>)))>

利用句法结构,可得到两种局部论域,一个包含 **m1** 和 **b1**,另一个包含 **d1** 和 **b2**。而且,**d1** 和 **b2** 都与其支配成分 **b1** 中的运算符纵向相关。

完整的关系从句通常称为辖域孤岛(scope island),因为它们往往不允许量词移出从句。例如,在这个例子中,最直观的解读指的是只有一个男孩,而非每条狗对应于一个不同的男孩。如果总是有这种约束,那么只须运行两次横向算法,就可以确定这类句子的辖域。一次确定从句的辖域,一次确定主句的辖域。

但从句并非总是辖域孤岛,比如“The dogs that won each race are hungry”,该句的一种解读是参加每场赛跑的都是不同的狗(即,“each”的辖域蕴涵了“the”)。在这种解读中,内嵌的量词必须从局部论域中提升出来放到紧邻的更高的论域中。在这个更高的论域中,内嵌的量词和同级的量词进行横向论域判定。更明确地说,暂时忽略时态,该句的无辖域逻辑形式为:

(HUNGRY1 h1 <THE d1 (& ((PLUR DOG1) d1)
(RUNS-IN1 r1 d1 <EACH r2 RACE1>)))>)

最上面一层的局部辖域只包含 **d1**, 而关系从句的局部辖域包含 **r2**, 它也和 **d1** 横向相关。如果 **r2** 的量词没有提升出 **d1** 的量词辖域, 则最可能的完全辖域的解释是:

(THE d1 : (& ((PLUR DOG1) d1)
(EACH r2 : (RACE1 r2) (RUNS-IN1 r1 d1 r2))
(HUNGRY h1 d1)))

如果提升了量词, 会在最上面一级进行横向辖域指定。在这种情况下, 生成解释:

(EACH r2 : (RACE1 r2)
(THE d1 : (& ((PLUR DOG1) d1)
(RUNS-IN1 r1 d1 r2))
(HUNGRY1 h1 d1)))

需要注意的是, 如果量词提升到紧邻的一个横向层级, 其辖域必然比用于支配它的量词更大。否则, 会产生一个不合法的表达式。例如, 如果 **r2** 的量词提升到 **d1** 的量词之上, 但横向辖域比 **d1** 还小, 会生成如下结果:

??? (THE d1 : (& ((PLUR DOG1) d1)
(RUNS-IN1 r1 d1 r2))
(EACH r2 : (RACE1 r2)
(HUNGRY1 m1 d1)))

注意, 变量 **r2** 的一个实例出现在它的量词辖域之外。显然, 这种表达式是无法解释的。

到目前为止, 例子中的局部论域都是由 S 成分定义的, 即主句和关系从句。局部论域也可以由 NP 定义, 当它们有介词短语修饰成分时, 两者会关联起来。分析句子:

The man in every boat rows.

句子的无辖域逻辑形式(忽略时态)是:

(ROWS1 r1
<THE m1 (& (MAN1 m1) (IN m1 <EVERY b1 BOAT1>)))>)

这里有两个局部论域, 一个只包含 **m1**, 另一个只包含 **b1**, 而且 **b1** 和 **m1** 纵向相关。如果 **b1** 的量词没有提升, 就会受到定指描述的限制, 最终的逻辑形式为:

(THE m1 : (& (MAN1 m1)
(EVERY b1 : (BOAT1 b1) (IN m1 b1)))
(ROWS1 r1 m1))

这里表示有一个人, 他在每艘船上都待过。如果提升了量词, 则它必然蕴涵了 **m1** 的量词, 生成的逻辑形式如下:

(EVERY b1 : (BOAT1 b1)
(THE m1 : (& (MAN1 m1)
(IN m1 b1))
(ROWS1 r1 m1)))

在这种解释中, 有很多不同的人, 每艘船上一个人。如果不考虑上下文, 这似乎是本句的首选解读。

通过提升不同的内嵌量词到相邻更高的层级,不同的纵向嵌入关系会有不同的最优解。基于内嵌量词被纵向提升的可能性,有人提出了内嵌结构的先后次序:

所有格 > PP 修饰成分 > 简化关系从句 > 关系从句

但要准确分析纵向关系,还需要在逐个例子的基础上观察每个结构,看看不同的运算符和每个结构之间如何相互作用。首先,让我们更仔细地研究 PP 修饰成分。当个体量词(比如“every”)内嵌到 PP 修饰成分时,我们很想提升它,例如:

A man in every boat was singing.

我们非常倾向于个体性解读,即很多人在歌唱。但如果内嵌量词不是个体性的(比如“a”),这种优选就不起作用了。例如:

Every man in a boat was singing.

我们非常倾向于认为“every man”要蕴涵“a boat”;即,不能理解成一艘船上的每个人都在歌唱,而另一艘船上的每个人都不唱歌。使用横向关系时,我们统计数据,用概率值来表示不同量词提升到各个不同的内嵌结构之外的可能性。

还有一点很重要,就是要记住上下文对于解释的优选有很大的影响。因此,任何基于纯结构优选的算法都必须和上下文解释结合起来。有一种技术是一次生成一种可能的序列,从基于结构属性的最可能的序列开始。如果解释在上下文中不恰当,就不大可能被采用。还有一种更一般性的方法是研究特定技术,将基于结构的概率和根据上下文信息得到的概率结合起来,生成一种整体上可能性最大的解释。

12.2 定指描述与辖域指定

定指描述由带量词“the”的名词短语组成。在某些情况下,这些短语就像定量表达式一样。而在其他情况下,它们像名称或者与辖域指定问题无关的结构一样。例如,句子:

The child entered with each dog.

好像不存在“each”辖域蕴涵“the”的解读。事实上,要理解这个句子,需要用迭代解读来进行解释。在迭代解读中有很多进入事件,但所有的事件只和同一个孩子有关。这里只有一个孩子,就像包含专有名词的句子一样,比如:

Jill entered with each dog.

该句不能理解成有很多人叫“Jill”。因此,看上去“the”在横向辖域歧义中毫无疑问地拥有最广的辖域。

但定指描述在特定结构中确实表现出辖域歧义,比如下面的句子:

The owner of every house showed us the plumbing.

In every house, the owner showed us the plumbing.

两个句子的首选解读都是每个房子都有不同的主人。通常,世界知识强制选择量词辖域蕴涵“the”的解释。比如,“The kitchen in every house had a stove”(每栋房子的厨房都有一个炉子),只

有存在很多不同的厨房,每栋房子都有厨房的情况下,句子才有意义。因此,在一些结构中,比如 PP 修饰成分的全称量词和前置状语修饰语,定指描述常常取决于其他量化参数。

要处理这种情况,我们常常将定指名词短语看成在两种应用中有歧义。第一种是指代用法,要求在上下文中寻找一个指代对象。另一种将“the”看成存在量词(存在惟一性约束)。要了解这些解读,让我们考虑一种情况。假设 Jack 有一个老板叫 Sam。句子“Jack has always been afraid of the boss”就有两种解读。在指代解读中,句子的意思是 Jack 总害怕 Sam(Sam 在这个时候刚好是老板)。在存在解读中,Jack 害怕的只是老板,而无论老板是谁。如果这句话是在去年某个时候说的,那时 Jack 的老板是另一个人,比如 Sue,这两种解读的区别就能体现出来了。用指代解读,Jack 去年怕的是 Sam。用存在解读,他去年怕的是 Sue。

只在少数情况下,可以只依靠句法和语义信息得到正确的解释。这些例子都是存在解读。例如,无论何种情况下,只要定指名词短语在全称量词的辖域内,都应该有一个存在解读。与此相似,谓词的使用,比如“He seems the best for the job”,也是存在解读。

12.3 句法分析过程中进行辖域指定的方法

现在,有很多不同的策略可用于判定辖域。例如,分析器按原来的方式运行,但我们写了一个新的解释程序,以无辖域的逻辑形式作为输入,并生成一个完整的指定了辖域的逻辑形式。那些直接使用逻辑形式的有效方法往往不考虑不同句法结构的影响。另一种方法是修改在语法中进行语义解释的方法,并要求分析器在分析句子的同时计算各种候选(或可能)的辖域。本节将探讨第二种方法,并且实现一个简单的辖域判定算法,这个算法和很多数据库查询程序中所用的方法类似。

我们对语法中的语义分析做一些改变,这样量词结构和 SEM 结构可以分别存到不同的特征里。例如,在老方法里诸如“the woman”这样的名词短语的 SEM 是 $\langle \text{THE } w1(\text{WOMAN1 } w1) \rangle$,在新方法里 QS(表示量词 Quantifiers)特征设定为 $\langle \text{THE } w1(\text{WOMAN1 } w1) \rangle$,而 SEM 设为篇章变量 $w1$ 。构造子成分的 SEM 形式的技术和以前一样,惟一要做的扩展是定义如何构造 QS 特征。一旦同一局部论域中所有的量词都收集到一起,就可以对其进行排序,并构造完整的指定了辖域的 SEM 形式。可以对分析器进行修改,每当构造了一个 S 或 NP 成分,就触发辖域判定算法。而这个方法将留给语法设计者更大的灵活性。我们定义一个新的二元特征 SCOPEPOS,只要它为+,分析器就触发程序,对量词进行排序。这个程序判断要将哪个量词提升到上一层横向上下文中,并对剩下的量词进行排序,然后将它们插入到 SEM 中。

例如,思考疑问句“When does each plane fly?”一开始,这个疑问句经过分析,构造出一个无辖域的表达式(忽略表示时间的时态运算符):

```
(S SCOPEPOS +
  QS(<WH t1 (TIME t1)> <EACH p1 (PLANE1 p1)>)
  SEM (& (FLIES1 f1 p1) (AT-TIME f1 t1)))
```

因为这个表达式有 SCOPEPOS 特征,所以激活了量词辖域判定算法。最高层的 S 不需要提升量词,然后经过排序,生成的新成分之一可能是:


```
(S SCOPEPOS -
  QS nil
  SEM (EACH f1 : (PLANE1 p1)
    (WH t1 : (TIME t1)
      (& (FLIES1 f1 p1) (AT-TIME f1 t1))))))
```

如果存在几种可能的解释,辖域判定程序会构造出多个新的 S 成分并添加到 chart 里。

再举一例,思考对关系从句的处理,例如名词短语“The flights that each man took”。作为关系从句的内嵌句,其初始形式是:

```
(S SCOPEPOS +
  QS (<EACH m1 MAN1>)
  SEM (TAKES1 t1 m1 x))
```

其中,x 是表示关系代词的变量。如果将关系从句视为绝对辖域孤岛,则只有一种可能的解释,即将量词插入到 SEM 中,生成成分:

```
(S SCOPEPOS -
  QS nil
  SEM (EACH m1 : (MAN1 m1) (TAKES1 t1 m1 x)))
```

如果量词可以提升,会得到另一个可能的成分,即:

```
(S SCOPEPOS -
  QS (<EACH m1 MAN1>)
  SEM (TAKES1 t1 m1 x))
```

可以用同样的方法处理由 NP 支配的局部论域的量词辖域判定问题。要根据修饰成分的形式分别处理辖域判定问题,需要引入新的特征来收集内嵌量词,包括出现在 PP 修饰成分(QSPP 特征)中的量词,以及提升到关系从句之外的量词(QSREL 特征)。对各种不同的情况,我们用不同的辖域指定策略来处理。需要注意的是,那些没有提升到 NP 上下文之外的量词要插入到量词约束范围的周围,而不要插入到 SEM 里面,对句子的处理也是如此。

例如,在判定辖域之前,考虑从“the flights to each city”中构造的 NP 成分:

```
(NP SCOPELOC +
  QS <THE f1 (& (FLIGHT1 f1) (DEST f1 c1))>
  QSPP <EACH1 c1 CITY1>
  SEM f1)
```

如果量词 EACH1 没有提升,则新的 NP 成分是:

```
(NP SCOPELOC -
  QS <THE f1 (& (FLIGHT1 f1)
    (EACH c1 (CITY1 c1) (DEST f1 c1)))>
  SEM f1)
```

如果量词提升,新的成分就是:

```
(NP SCOPELOC -
  QS (ORDERED <EACH c1 CITY1>
    <THE f1 (& (FLIGHT1 f1) (DEST f1 c1))>)
  SEM f1)
```

需要注意的是,QS 的值包含一个额外的结构,根据提升的成分指定约束的次序。在这种情况下,EACH 必须在 THE 的辖域之外。

一元运算符,比如否定和时态,都可以用这种方法进行处理。运算符置于 QS 特征内,激活辖域判定算法的时候再指定其位置。例如,动词短语“saw a dog”(看见一条狗)可以分析为:

(VP QS (<PAST> <A d1 DOG1>)
SEM (λ ag (SEES1 s1 ag d1)))

这个表达式传递给 S 结构(代表句子“The man saw a dog”),其形式为:

(S SCOPELOC +
QS (<THE m1 MAN1> <PAST> <A d1 DOG1>)
SEM (SEES1 s1 m1 d1))

对这个表达式可用通常的方法进行辖域判定,并在算法判定最佳的地方插入 PAST 运算符。

要让这种方法也能判定运算符的辖域,必须修改词典条目和词语形态规则,使之可以正确地设置 QS 记录。例如,将一般过去时态动词的规则设成 QS 记录,如下所示:

(V QS <PAST> SEM ?semv) \rightarrow
(V SEM ?semv IRREG-PAST -) +ED

12.3.1 一个例子

让我们看看这种技术在简单数据库查询程序中的应用。数据库的查询中广泛用到量词,因此,用这个领域来做测试是个不错的选择。在 S 支配的论域中,对横向排序进行选择时,通常建议使用加权方法。对每个量词赋予一个特定的权值,这个权值标识辖域的大小。然后,将所有横向相关的量词根据权值进行排序。所加的权值需满足如下次序:

时态运算符 > *the* > *each* > *wh* > 其他量词 > 否定

如此一来,诸如“When does each plane receive maintenance?”(什么时候各架飞机进行维护保养?)这样的查询可以解释为查找每架飞机分别进行维护的时间列表,而“When does every plane receive maintenance?”(什么时候所有的飞机都进行维护保养?)则可以解释为查找所有飞机同时进行维护的时间。在一句话中,根据权值相同的量词出现次序的先后进行辖域指定。时态运算符的辖域最大,因为如果一个句子的时态运算符的辖域很小,则通常不会出现在数据库查询中。否定运算符的辖域最小。这表示为如下所示的形式:

(S QS (<WH f1 (PLUR FLIGHT1)> <NOT> <A m1 MEAL1>)
SEM (SERVES1 s1 f1 m1))

分析得到的指定辖域的完整形式是:

(WH f1 : ((PLUR FLIGHT1) f1)
(A m1 : (MEAL1 m1)
(NOT (SERVES1 s1 f1 m1))))

对 NP 支配论域中的纵向关系,我们用同样的策略进行处理,即 PP 修饰成分中的所有量词的辖域比它们所修饰的量词短语的辖域都大;而关系从句中,除了定指量词之外的所有量词的辖域都不超出从句范围(也就是说,关系从句是强辖域孤岛)。

对于真正的数据库查询系统而言,这里尝试的方法虽然过于简单,但可以用来说明这些技术。使用同样的框架很容易对其进行扩展。

语法 12.2 是一个小型语法,能够解析某些包含横向关系的典型疑问句。读者想必在第 9 章已经看到了大部分规则的变体。惟一真正的改变就是加上了 QS 特征。规则 1 是标准 S 规则,只是 S 标记上了 + SCOPELOC。因此,一遇到从 NP 和 VP 集中起来的所有量词,就会触发辖域判定算法。规则 2 处理诸如“when”和“where”这样以特殊疑问词开始的疑问句。规则 3 处理特殊疑问句中需用到的 S 的倒装形式。规则 4 是及物动词的标准规则,但增加了 QS 特征。规则 5 处理以限定词开始的名词短语。注意,这里用 QSPP 和 QSREL 来存储内嵌量词,而 QS 存储支配成分(也就是新的 NP)的量词。规则 6 可处理不带修饰成分的 CNP 结构,规则 7 处理疑问句中诸如“when”和“where”这样的 PP 单词。

1. (S SCOPELOC + INV - QS (?qsnp ?qss) SEM (?semvp ?semnp)) →
(NP QS ?qsnp SEM ?semnp) (VP QS ?qsvp SEM ?semvp)
 2. (S SCOPELOC + QS (?qspp ?qss) SEM (& (?sempv ?v) ?sems)) →
(PP WH Q QS ?qspp SEM ?sempv)
(S INV + VAR ?v QS ?qss SEM ?sems)
 3. (S INV + QS (?qsaux ?qsnp ?qsvp) SEM (?semaux (?semvp ?semnp))) →
(AUX QS ?qsaux SEM ?semaux)
(NP WH Q QS ?qsnp SEM ?semnp)
(VP QS ?qss SEM ?semvp)
 4. (VP VAR ?v QS (?qsv ?qs) SEM (l a3 (?semv ?v a3 ?semnp))) →
(V[_np] QS ?qsv SEM ?semv) (NP QS ?qs SEM ?semnp)
 5. (NP SCOPELOC + VAR ?v SEM ?v
QSPP ?qspp QSREL ?qsrel QS <?semdet ?v (?semcnp ?v)>) →
(DET SEM ?semdet)
(CNP QSPP ?qspp QSREL ?qsrel SEM ?semcnp)
 6. (CNP SEM ?semn) → (N SEM ?semn)
 7. (PP WH Q QS ?qs SEM ?sem) → (PP-WRD WH Q QS ?qs SEM ?sem)
- S, VP, NP, CNP 的中心特征: VAR

语法 12.2 一个带 QS 和 SEM 特征的简单语法

图 12.3 中给出了“*When does the flight leave each city?*”的分析树。这个句子包含一个值得重视的局部论域,该论域包含所有的量词和时态运算符。这里,请注意运算符如何沿着树传递给 S 结构。在该结构中,辖域判定算法生成了完整的辖域指定形式。

语法 12.4 列出了一些补充规则,这些规则用于处理 PP 变体和关系从句。前面已提到,介词短语修饰语中的量词总得到提升,而关系从句中的量词总无法提升。规则 8 是处理增加了 QS 特征的谓词中介词短语的标准规则。规则 9 处理 CNP 的 PP 变体,并设置 QSPP 特征。规则 10 处理关系从句并设置 QSREL 特征,而规则 11 处理真正的关系从句。

使用该语法,疑问句“*When does the flight to each city leave?*”生成如下逻辑形式:

```
(EACH c1 : (CITY1 c1)
  (THE f1 : (& (FLIGHT1 f1) (DEST f1 c1))
    (WH t1 : (TIME t1)
      (& (<PRES LEAVES1> l1 f1 c1) (AT-TIME f1 t1))))
```

具体地说,问题是关于很多不同航班的,每个航班飞往一个城市。这个辖域判定结果是对 PP 修饰语进行处理得到的。与此相反,疑问句“*When did the flight that each man took leave?*”问的是所有人搭乘的一个航班。这种解读是对关系从句进行处理的结果。

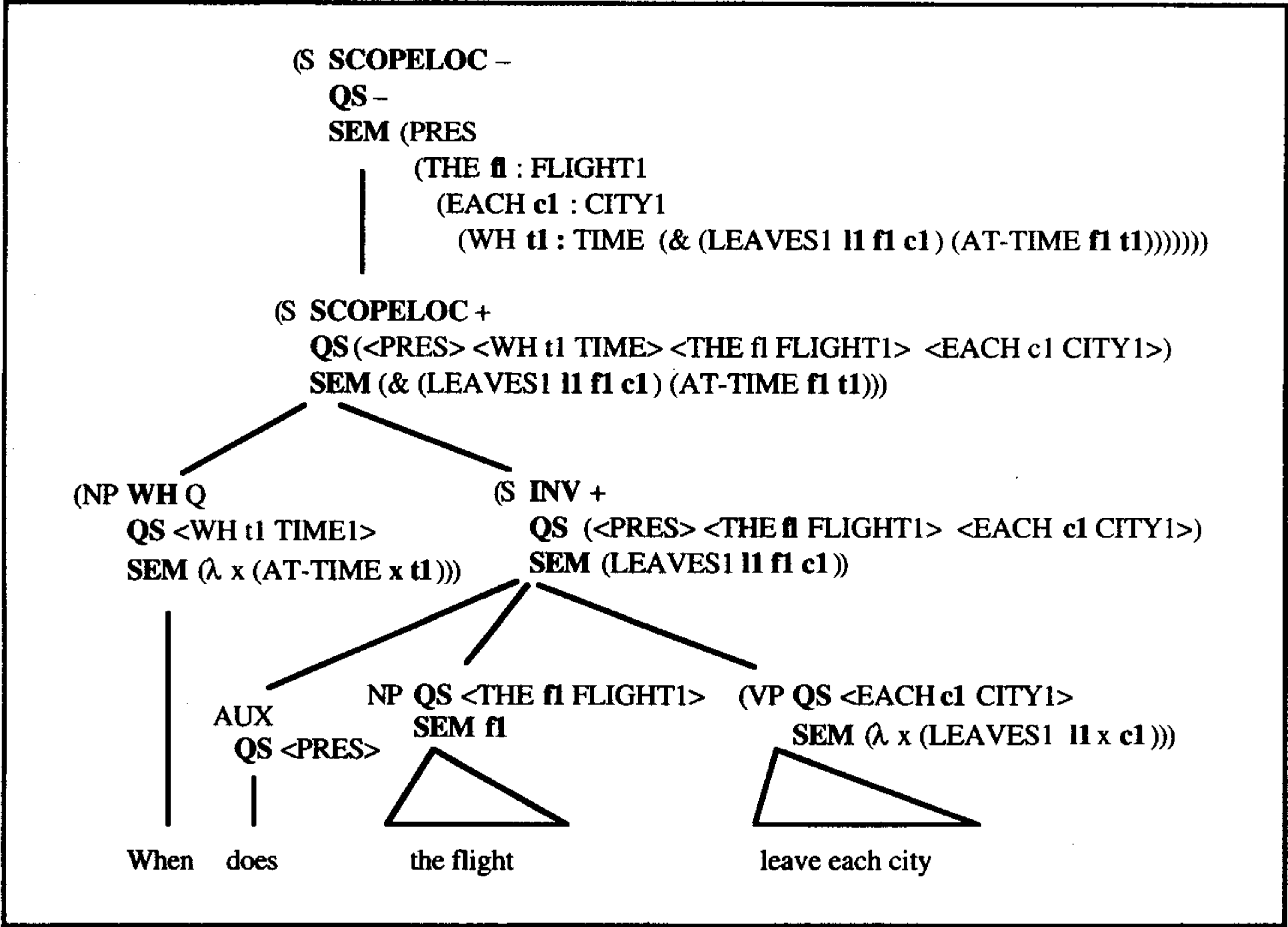


图 12.3 “When does the flight leave each city?”的分析树

- 8. (PP PRED + QS ?qs SEM (λ x (?semp x ?semnp))) →
(P SEM ?semp) (NP QS ?qs SEM ?semnp)
 - 9. (CNP SEM (λ n1 (& (?semcnp n1) (?sempp n1)))
QSPP ?qspp QS ?qscnp QSREL ?qsrel) →
(CNP QS ?qscnp QSREL ?qsrel SEM ?semcnp)
(PP PRED + QS ?qspp SEM ?sempp)
 - 10. (CNP QSPP ?qs QSREL ?qsrel SEM (λ n2 (& (?semcnp n2) (?semrel n2)))) →
(CNP QSPP ?qs SEM ?semcnp) (REL QS ?qsrel SEM ?semrel)
 - 11. (REL QS ?qs RVAR ?v SEM (λ ?v ?sems)) →
(NP WH R VAR ?v SEM ?semnp)
(S QS ?qs GAP (NP SEM ?semnp) SEM ?sems)
- NP, CNP 的中心特征: VAR

语法 12.4 处理简单 PP 变体和关系从句的规则

12.4 互指与绑定约束

名词短语的互指存在明显的句法约束,这些约束大致表明它们指向同一个对象。这是语

言学的一个比较活跃的研究领域。在此,只讨论一些基本的结果。具体地说,考虑以下例句,名词短语的下标表明它们互指,而星号和往常一样,表示不正确的解释。

- 1a. * When Jack_i arrived at the party, she_i was drunk.
- 1b. When Jill_i arrived at the party, she_i was drunk.
- 2a. * He_i said Jack_i wants to leave.
- 2b. Jack_i said he_i wants to leave.
- 3a. * Jill_i saw her_i in the mirror.
- 3b. Jill_i saw herself_i in the mirror.
- 3c. * Jill_i thought that Jack saw herself_i.
- 3d. Jill_i thought that Jack saw her_i.
- 4a. * Jack_i thought the tired man_i was dying.
- 4b. Jack_i thought he_i was dying.

如果存在互指,第一个短语通常称为先行词,而第二个则称为回指对象。这些例句用以说明句内回指——两个互指表达式出现在同一个句子中。回指的其他主要形式是篇章回指或句间回指,将在第 14 章进行讨论。要理解它们的差别,让我们分析句子“Jack found his hat”的歧义问题。在这个句子中,代词“his”既可能指 Jack(句内),也可能指前面一个句子中提到的其他人(句间)。因为目前不考虑上下文,所以这里只研究句内的情况。

在下面几段中,我们提出一些初步的约束,并对问题进行探讨。这将促使我们在下文提出更好的约束集。句子 1a 和 1b 可以用一个显而易见的约束来说明,两个互指的名词短语,在性别、数以及人称上必须一致。句子 1a 是不规范的,因为 Jack 和 she 在性别上不一致。

要解释句子 2a 和 2b,一个很容易让人想到的约束就是先行词必须出现在指代它的代词之前。因此,2a 不合法,因为代词先出现。遗憾的是,这个约束并不合理,我们可用下面的例子来说明:

In its_i dreams, the goldfish lives in the ocean.

Before she_i went to the party, Jill_i bought some wine.

Most of her_i friends think that Jill_i will win.

我们随后将提出一个更好的约束。

从句子 3a 到 3d 的例子给出了反身代词使用约束的用法。最终可以看到,以前定义的局部论域的概念为此提供了一个好的起点。反身代词必然指代同一局部论域中的一个短语,而非反身代词和同一局部论域中的短语必然不能互相指代。句子 3a 是不规范的,因为 Jill 和“her”在同一局部论域中。而 3c 不规范是因为 Jill 和“herself”不在同一局部论域中。下面的例子给出了反身代词不指代句子主语的一种用法:

Jill talked to Jack_i about himself_i.

但这里所说的约束无法解释下面这样的句子为何不规范:

* Jill talked to himself_i about Jack_i.

最后,句子 4a 和 4b 表明在某些情况下必须使用代词。这里有一个前提约束,即同一句话中的非代词性名词短语不能互指。但这个约束太宽泛了,如下句所示:

After Jill had been questioned for hours, Sue took the tired witness out to lunch.

借助语言学中的其他概念,可以得到一组更好的约束。前几节中局部论域的定义就是一个开始。除此之外,我们还使用一个称为 C-统制(C-command)的概念:

成分 C 对成分 X 进行 C-统制,当且仅当:

1. C 不支配 X。
2. 支配 C 的第一个分支节点也支配 X。

要理解这个定义,让我们分析图 12.5 中的句法树。支配 NP₁ 的第一个分支节点是 S,因此, NP₁ 对 NP₂ 和 NP₃ 进行 C-统制。另一方面,支配 NP₂ 的第一个分支节点是 VP 节点,因此, NP₂ 对 NP₃ 进行 C-统制。在第二棵树中, NP₄ 对 NP₅, NP₆ 和 NP₇ 进行 C-统制,而 NP₆ 对 NP₇ 进行 C-统制。

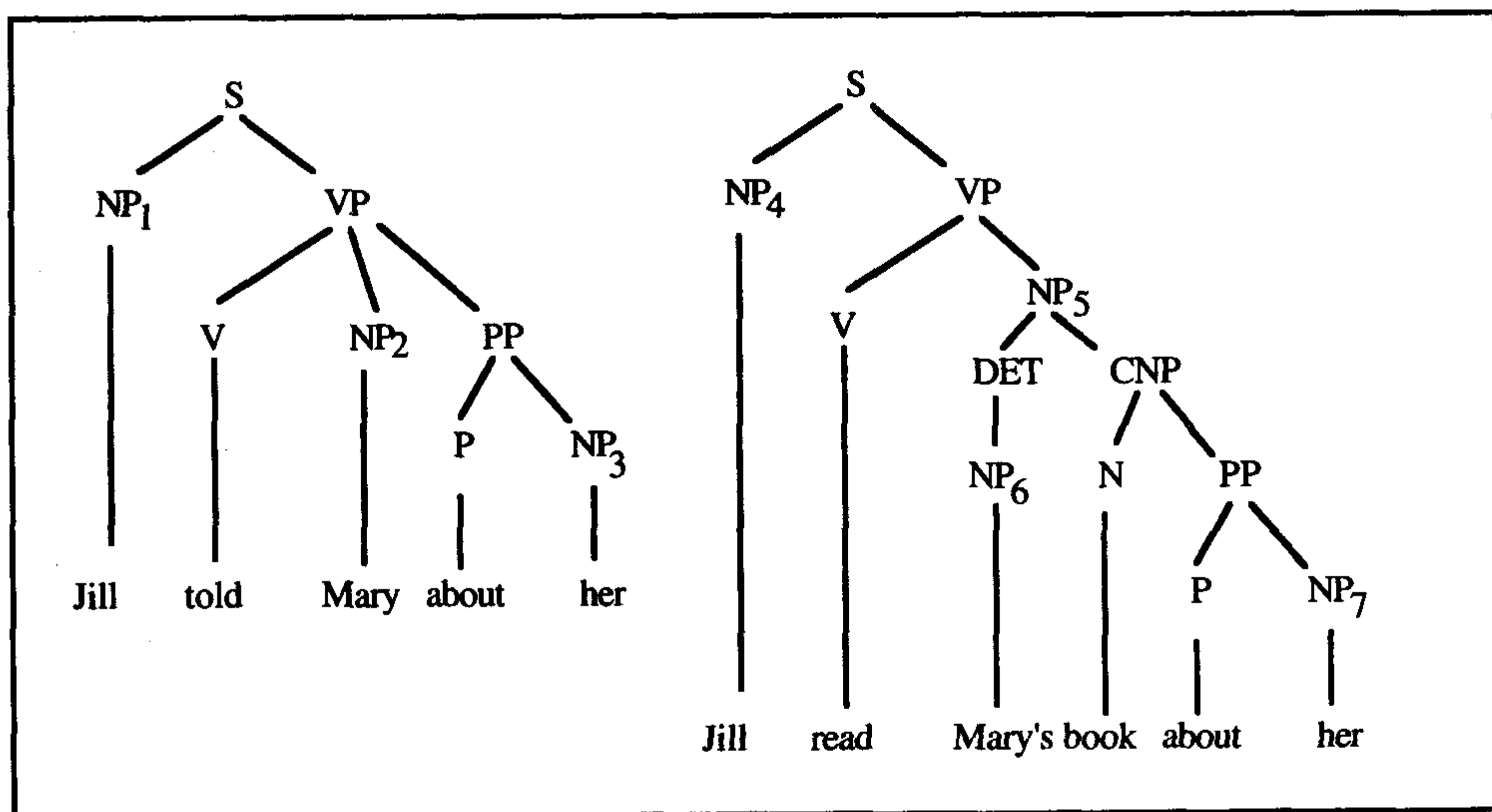


图 12.5 C-统制关系的示范句法树

现在,反身代词的约束可以用 C-统制关系重新表述如下:

1. 反身代词指代的 NP 必须 C-统制它并且在同一个局部论域中。
2. 非反身代词必然不能指代在同一局部论域中 C-统制它的 NP。

要探讨这些约束的含义,让我们分析图 12.5 中的句子及某些变体。在左边的句子中,代词“her”既不能指 Jill 也不能指 Mary,因为两者都 C-统制它,而且两者都在包含该代词的最小 S 中。但在“Jill told Mary about herself”中,反身代词既可以指 Jill,也可以指 Mary。在“Jill told herself about Mary”中,反身代词只能指 Mary,因为 Mary 和代词虽然在同一局部论域中,但没有 C-统制它。在右边的句子中,代词“her”可以指代 Jill,因为虽然 NP₄ 对 NP₇ 进行 C-统制,但它和 NP₇ 不在同一局部论域中。另一方面, NP₇ 指代 Mary,则这个代词必然是反身代词。

这些约束为由反身代词引出的问题提供了合理的解释,但我们还需要对句子 2a, 2b, 4a 和 4b 做出解释。这些句子可以用非指代性名词短语的约束进行处理:

3. 非指代性名词短语不能和 C-统制的 NP 互指。

这个约束解释了前面几个例子。句子 2a 是不规范的,因为代词“*He*”C-统制 *Jack*。句子 4a 不规范也是基于同样的原因,即主语 *Jack* 对名词短语“*the tired man*”进行 C-统制。注意,对句子“*After Jill_i had been cross-examined for hours, Sue took the tired witness_i out to lunch*”,两个名词短语之间没有 C-统制关系,因此可以互指。

12.4.1 和全称量词的相互影响

代词和量词之间也会相互影响。分析句子:

Every man_i thought he_i would win the race.

这个句子表示每个人都希望自己在跑步比赛中获胜。本例中,代词“*he*”处于全称量词的辖域内,指的是量词作用于的每个个体。这种现象称为代词的绑定变量解释(*bound variable interpretation*)。这种解读出现在很多结构中,比如:

Every cat_i ate its_i dinner.

Each man_i shot himself_i.

通常,绑定变量解释必须被全称量词 C-统制:

4. 绑定变量约束:非反身代词可以绑定为一个全称量化 NP 的变量,仅当 NP 对该代词进行 C-统制。

非全称定量表达式,比如存在量词,尤其是定指量词 *THE*,都不需要这条约束。并且,在任何名称和其他指代项可用的情况下都可以作为代词的先行词。

图 12.6 总结了前面讨论的一些规则。现在还存在这些约束的一些反例,表明需要做进一步的提炼才能处理所有的情况。比如,下面的句子似乎违反了反身代词约束:

He saw a book about himself.

因为反身代词“*himself*”的局部论域是名词短语“*the book about himself*”。注意,在下面的句子中:

He saw John's book about himself.

反身代词和 *John* 互指,而并非和句子的主语互指。一种可能的解决方法是对局部论域的概念进一步细化,使其只包含名词短语形式。比如,所有格限定成分,排除只有“简单”限定词结构的名词短语。此类策略在语言学著作中已有深入的探讨。

1. 反身代词指代的 NP 必须 C-统制它并且在同一局部论域中。
 2. 非反身代词必然不能指代在同一局部论域中 C-统制它的 NP。
 3. 非代词性 NP 不能和 C-统制它的 NP 互指。
 4. 绑定变量约束:非反身代词可以绑定为一个全称量词辖域内的 NP 的一个变量,仅当 NP 对该代词进行 C-统制。
 5. 两个互指的名词短语必然在数、性别和人称上一致。

图 12.6 绑定约束小结

框 12.1 驴子句

一些更复杂的结构并不遵循本章描述的一般模式。这类句子常称为“驴子句”(donkey sentence),我们用一组例子来说明这个问题。分析句子:

Every man who owns a donkey_i beats it_i.

这里,自然的解读是将“it”作为指代“a donkey”的绑定变量,但这个代词本身可以取一组值,因为它在全称量词 NP“every man”的辖域内。之所以会出现这个问题,是因为代词没有落在它所绑定的名词短语的辖域内。要理解这个问题,让我们分析无辖域的逻辑形式:

(PRES(BEAT

[AGENT < EVERY m1 MAN1(OWNS m1 < A d1 DONKEY1 >) >]

[THEME d1]))

如果 d1 的量词提升到 m1 之外,那么所有人只共同拥有一头驴,这并非本意。但如果 d1 还在 m1 的辖域内,量词就会落入关系从句内,因而不能作为 it 的绑定变量解释。这个例子启发了很多人的工作[例如,Kamp(1981)和 Heim(1982)]。

12.4.2 计算互指约束

要计算互指约束,需要知道 C-统制各个成分的成分以及在同一局部论域中的成分集合。有了这些信息,就可以为各种不同类型的名词短语生成互指约束。互指约束可以用三个新的谓词表示成逻辑形式的附加限制。EQ-SET 表示第一个参数必然等于第二个参数中的某一项。比如:

(EQ-SET b1 (g1 d1))

表示 b1 必然等于 g1 或 d1。另一方面,NEQ-SET 表示第一个参数必然不等于第二个参数中的任何一项,比如:

(NEQ-SET b1 (g1 d1))

表示 b1 不等于 g1,也不等于 d1。最后,谓词 BV-SET 列出 C-统制名词短语,其中包含非反身代词形式的所有可能的绑定变量解释。(反身代词形式可能的绑定变量解释已经包含在代词的 EQ-SET 中了。)

这些谓词可以将约束放到互指关系中,但还不能定义所有的可能性。例如,对于非反身代词,只能基于句法结构说明代词和哪些名词短语不能互指。更具体地说,根据一致性约束,可以排除其他的句内先行词。由于先行词可能出现在代词之后,因而没有一种简单的方法可以在分析过程中生成所有可能的句内指代对象。进一步说,因为一致性检查也需要在篇章理解中完成,因此,最好的方法是把这些情况留给后续的上下文过程来处理。但是,当代词明确符合这些限制时,可以取一种绑定变量解读。

例如,下面是为句子“Every boy thought he saw him”中的名词短语生成的逻辑形式:

Every boy: <EVERY **b1** (BOY1 **b1**)>
 he: (PRO **h1** (& (HE1 **h1**) (BV-SET **h1** (**b1**))))
 him: (PRO **h2** (& (HE1 **h2**) (NEQ-SET **h2** (**h1**))
 (BV-SET **h2** (**b1**))))

换句话说,**h1** 可能有一个绑定变量解释,和 **b1** 互指,也可能有一个会话句内解读。代词 **h2** 也可能有一个绑定变量解释或是会话解读,但无论如何,都不会与 **h1** 互指。这个限制成功地表达了由句子的句法结构加在它上面的绑定约束。

再举最后一个例子,分析句子“The woman who owns every record bought it for herself”。反身代词必然指的是“the woman”,但代词“it”不能有“every record”的绑定变量解释,因为这个量词短语没有 C-统制它。该句的逻辑形式是:

every record: <EVERY1 **r1** (RECORD1 **r1**)>
 The woman who owns every record:
 <THE **w1** (& (WOMAN1 **w1**)
 (OWNS **o1** **w1** <EVERY1 **r1** (RECORD1 **r1**)>))>
 it: (PRO **i1** (IT1 **i1**))
 herself: (PRO **h1** (& (SHE1 **h1**) (EQ-SET **h1** (**i1** **w1**))))

注意,**i1** 没有约束。但是它不能与 **r1** 互指,因为这样会产生一个绑定变量解释。如果约束中没有明确提到,这种情况是不允许的。还要注意,**h1** 必然和 **i1** 或 **w1** 互指。如果分析器检查了一致性约束,**i1** 就会被排除而 **h1** 将和 **w1** 互指。

修改一个语法使之能够记录 C-统制关系和局部论域,这是比较简单的。此外,需要用两个新特征,即 CC,所有不在当前局部论域中的 C-统制成分的篇章变量;以及 LCC,所有在局部论域中的 C-统制成分的变量。

通过分析以下两种现象,可以推导出等式来:

- S 和 NP 结构定义了新的局部论域,因而“重置”了 LCC 变量。LCC 特征的值成为子成分的 CC 值的一部分。
- 在 S 或 NP 的子成分中生成的 LCC 或 CC 的值不需要回传到成分的外部。

12.5 形容词短语

形容词通常对 NP 中所指的对象引入一些限制,但在很多情况下,分析会稍微复杂。形容词可以分为两类,交集型形容词和非交集型形容词。之所以称为交集型形容词是因为可以用集合的交集对其进行分析。例如,要生成绿球的集合,只须对球的集合和绿色物体的集合取交集即可。这类形容词可以视为谓词,它们是对名词短语所加的限制。例如,“the green ball”的逻辑形式是:

<THE **b1** (& (BALL1 **b1**) (GREEN **b1**))>

非交集型形容词有好几类。第一类包括诸如“large”,“slow”和“bright”这样的形容词,它们和所修饰的名词短语相关。例如,一条游得很慢的海豚比一只爬得很快的蜗牛要快很多。与之相似,对一个人的同一种跑步速度,如果他是大学教授,在大家眼里可能是一个跑得很快的人;但如果这个人是运动员,大家可能会认为他跑得很慢。因此,不能简单地认为在纯粹意义上运动速度很慢的事物,“slow”作为谓词都为真。这些结构可以通过谓词修饰成分用逻辑形

式来处理,以前,我们曾用这些修饰成分来处理复数名词。谓词修饰语是从一个谓词到新谓词的函数。从语义上说,可以称其为从集合到集合的函数。如果 DOLPHIN1 是海豚集合的谓词表达式,则 SLOW1 以 DOLPHIN1 这样的集合作为输入,并生成包含那些游得慢的海豚的子集(相对于海豚的集合)。因为总能产生子集,所以这些形容词常称为子集型形容词。用第 8 章提出的谓词修饰语的概念,“the slow dolphin”的逻辑形式可写为:

<THE d1 ((SLOW1 DOLPHIN1) d1)>

语法 12.7 列出了一个小型语法,给出了交集型形容词和子集型形容词的处理方法。“slow”的词典条目是:

(ADJ ROOT slow ATYPE SUBSECTIVE SEM SLOW1)

规则 2 生成 CNP “slow dolphin”的 SEM 如下:

(SLOW1 DOLPHIN1)

可用在标准 NP 规则中构造“the slow dolphin”的 SEM。

1. (CNP SEM (λx (& (?semadjp x) (?semcnp x)))) \rightarrow
(ADJP ATYPE INTERSECTIVE SEM ?semadjp)
(CNP SEM ?semcnp)
2. (CNP SEM (?semadjp ?semcnp)) \rightarrow
(ADJP ATYPE SUBSECTIVE SEM ?semadjp)
(CNP SEM ?semcnp)
3. (ADJP ATYPE ?a SEM ?sem) \rightarrow (ADJ ATYPE ?a SEM ?sem)

语法 12.7 形容词修饰成分的一些解释规则

另一类非交集型形容词和它们所修饰的集合不存在简单的关系。这类形容词包括“average”,“toy”和“alleged”等。平均分未必是真实的分数,玩具枪也不是真枪,而受指控的疑凶也可能不是凶手。相反,这些修饰语以一种不同的方式和它所修饰的属性相关联。再举一例,形容词“former”,在修饰“astronaut”这样的属性时产生一个集合,集合个体是那些以前是宇航员但现在不是的人。形容词“toy”修饰“gun”产生一个看起来或操作起来像枪的对象的集合。对这些结构给出完全的语义解释已经超出了本书的范围,但可以考虑某一类特殊的计算应用需求,比如数据库查询系统。这类形容词定义了一个量,这个量是对它所修饰的集合计算所得的结果,比如“average”。

要处理此类结构,必须在逻辑形式中引入一个显式集合构造运算符。该运算符定义如下:

(SET($\lambda x P_x$))是所有满足谓词($\lambda x P_x$)的对象的集合。

给定此运算符,诸如“average”这样的形容词的规则可按如下方式进行处理:

(CNP SEM (?semadj (SET ?semcnp))) \rightarrow
(ADJ ATYPE SET-PROPERTY SEM ?semadj) (CNP SEM ?semcnp)

已知此规则,名词短语“the average grade in the class”的逻辑形式可表示为:

<THE a1 ((AVERAGE (SET (λ g (& (GRADE1 g)
(IN1 g <THE c1 CLASS1>)))))) a1)>

形容词短语可能也有修饰语和补语,虽然用做名词前修饰语的形容词带补语的情况比较罕见。但是,“light red”这样的形容词短语却很常见。这类形容词修饰语可以看做谓词修饰语,照此处理不会有太大的问题。因而,简单形容词修饰语的规则可写成如下形式:

(ADJP SEM (?semadv ?semadjp)) →
(ADV ATYPE PREMOD SEM ?semadv)
(ADJP SEM ?semadjp)

如果形容词修饰语“light”的 SEM 是 LIGHT1,则形容词短语“light red”的逻辑形式是以下谓词:

(LIGHT1 RED1)

按照通常的做法,该形容词短语要和其他成分合并。比如,“the light red bus”的逻辑形式是:

<THE b1 (& (BUS1 b1) ((LIGHT1 RED1) b1))>

12.5.1 形容词比较级

比较级形式是一类对于计算应用来说很重要的形容词短语。其结构比较复杂,这里只讨论一些简单的例子。比较级形容词结构会出现在很多形式中,比如:

He is *larger than the boy is*.

He is *larger than the boy*.

The *more brilliant woman* won the prize.

A woman *more brilliant than the first speaker* has won the prize.

I am *happier than I have ever been*.

这些例子要和

I own *more horses than George does*.

Jack gave *as much money as he could*.

这样的结构区分开来。这些也称为比较级结构,但其实是名词短语。其中的“more...than”和“as much...as”结构应看成量词而非形容词短语。这种例子在框 12.2 中加以探讨。

形容词比较级短语结构都隐含地标识了某一个尺度,在这个尺度上不同的对象可以进行比较。例如,“happier”标识了喜悦的尺度。在这个尺度上,形容词“happy”标识了尺子的一端,而“sad”标识了另一端。对象 X 比对象 Y 更高兴,只要 X 在尺度上比 Y 更高(也就是说,更靠近“happy”)。需要注意的是,“happier”不一定就“happy”。X 和 Y 可能都难过,但 X 要比 Y 稍好一些。当然,相对于所用的尺度,形容词可能会有歧义。例如,句子“Canada is larger than the U.S.”,如果比的是面积就为真,如果比的是人口就为假。不同的形容词比较级也可能指同一个尺度。例如,“larger”和“smaller”指的是同一个尺度,区别只在于对象在这个尺度上比较的方向不同。

借助这些背景知识,我们将形容词的比较级定义成二元谓词,从而在某一个特定尺度上对两个对象进行排序。已知尺度 S,谓词(MORE S)是一个二元谓词。在尺度 S 上,当第一个参数

比第二个参数大时,谓词为真。“heavier”的 SEM 是(MORE WEIGHT-SCALE),“higher”的 SEM 是(MORE HEIGHT-SCALE),而“happier”的 SEM 是(MORE HAPPY-SCALE)。很多比较级结构都带一个补语,由引导补语从句的“than”后跟一个 S/ADJP 或 NP 构成。如果后跟 NP,其规则是:

$$\begin{aligned} &(\text{ADJP SEM } \lambda x \text{ (?semadjp } x \text{ ?semnp)}) \rightarrow \\ &\quad (\text{ADJP ATYPE COMPARATIVE SEM ?semadjp}) \\ &\quad (\text{THAN}) \\ &\quad (\text{NP SEM ?semnp}) \end{aligned}$$

其中,(ADJP ATYPE COMPARATIVE)规定要包含“more brilliant”和“happier”这样的结构。已知该规则,形容词短语“happier than John”的逻辑形式可表示为:

$$\lambda x . ((\text{MORE HAPPY-SCALE}) x (\text{NAME j1 "John"}))$$

12.6 关系名词与名词化

诸如“sister”,“boss”和“author”这样的名词和其他普通名词(比如“dog”或“idea”)有很大的不同。具体来说,这些名词并不直接指某一类个体;相反,定义为相对于其他个体的关系。例如,当集合中所有人都是某人的姐妹时,谈论“sister”的意义是没什么意义的。相反,每次用到名词“sister”,其明确或者隐含的意义是我们所想到的某人的姐妹。有一些标准结构可以表示这种对象。这些结构包括所有格,比如“John’s sister”;以及使用“of”的 PP 补足语成分,比如“the author of this book”。这表明应为这类名词定义一套通用的子类和语义解释规则集。

一种表示方法用非关系类型和使用子类 NP 的二元关系来定义每个关系名词。例如,“author”的意义可表示成:

$$\lambda b \lambda p (& (\text{PERSON } p) (\text{AUTHOR-OF } b \text{ } p))$$

其中,第一个变量指提到的书,第二个指作者本人。这样,就可以用类似于语法 12.8 中的规则 1 的规则,对 CNP“author of the book”给出其逻辑形式:

$$\lambda p (& (\text{PERSON } p) (\text{AUTHOR-OF } \langle \text{THE } b1 \text{ BOOK1} \rangle p))$$

所有格形式只须借助一个特殊的关系名词的规则就能进行类似的处理,这项规则改写了以前所述的所有格的标准规则。在这种情况下,需要引入某些特征来表示所有格 NP 子类,从而以内嵌 NP 的 SEM 作为其 SEM,就像 PP 子类以其 NP 补足语作为 SEM 特征一样。如果用 PRED 特征来表示,那么适用于带关系名词的所有格的规则就是语法 12.8 中的规则 2 和规则 3。语法还要能处理没有出现子类成分的情况,比如名词短语“the author”。要处理这种情况,需要引入指代成分来表示缺失的对象,这种情况用规则 4 来处理。在这条规则中,REL-N1 是一个新的意义,用来表示插入指代元素的形式。

已知语法和上述“author”的意义,可得到名词短语“the book’s author”的逻辑形式:

$$\langle \text{THE } a1 (& (\text{PERSON1 } a1) (\text{AUTHOR-OF } \langle \text{THE } b1 \text{ BOOK1} \rangle a1)) \rangle$$

而名词短语“the author”的逻辑形式是:

$$\langle \text{THE } a2 (& (\text{PERSON } a2) (\text{AUTHOR-OF } (\text{PRO } b2 \text{ REL-N1}) a2)) \rangle$$

诸如“murderer”,“actor”和“thief”这样通常以后缀 er 或 or 结尾的词,也属于这一类,通常根据人的行为对人进行区分。因此,可以说“murderer”谋杀了一个或多个人,“actor”表演节目,而

“thief”偷东西。还有一些名词,标识出 AGENT 格之外的其他格。例如,名词“recipient”在传递这个动作中标识 TO 格,“donation”在捐赠这个动作中标识 THEME 格,更一般的情况下,带后缀 -ee 的词,比如“addressee”和“rentee”在两个人的交互动作中标识由非 AGENT 的有生命对象充当的格。这些名词同时也对标识动作中其他格的成分进行区分。例如,在“the murderer of John”中,名词短语 John 直观上应充当谋杀这个动作的 THEME 格。因此,对谋杀这个动作中“murderer”的分析看起来很有代表性。

1. (CNP RELATIONAL – SEM (?semcnp ?sempp)) →
 (CNP RELATIONAL + SEM ?semcnp)
 (PP PRED – PTYPE of SEM ?sempp)
2. (NP POSS + PRED – SEM ?sem) → (NP POSS – SEM ?sem) 's
3. (NP RELATIONAL – VAR ?v SEM <THE ?v ((?semcnp ?semnp) ?v)>) →
 (NP POSS + PRED – SEM ?semnp)
 (CNP RELATIONAL + SEM ?semcnp)
4. (CNP RELATIONAL – SEM (?cnp (PRO x REL-N1))) →
 (CNP RELATIONAL + SEM ?cnp)

语法 12.8 简单关系名词短语的规则

框 12.2 数值量词及其他

数值量词在数据库查询程序中扮演重要角色。大多数情况下它们都很简单,但也有一些复杂的情况需要加以说明。数值量词属于存在量词,在下面的名词短语中出现:

Three men rowed the boat. (三个人划船。)

At least three men rowed the boat. (至少三个人划了船。)

More than three men rowed the boat. (超过三个人划了船。)

可以通过引入更复杂的量词来处理这种例子,比如 (EXACTLY 3), (AT-LEAST 3), (MORE-THAN 3), (LESS-THAN 3), 等等。因而,名词短语“Three men”的逻辑形式是:

<(EXACTLY 3) m1 (PLUR MAN1)>

诸如“more men than were at the concert”(比音乐会上更多的人)和“less money than I expected”(比我期望的更少的钱)这样的名词短语更复杂。一种解决方法是假定限定结构不连续;例如,第一个例子中的量词是“more ... than were at the concert”。这个解释使用刚定义的运算符 MORE-THAN。但在这种情况下,我们将集合的基数设为其参数,而不是一个简单的数。因此,“more men than were at the concert”的逻辑形式应该是:

<(MORE-THAN (SIZE (λ x (& ((PLUR MAN1) x) (AT x <THE c1 CNC1>))))
 m1 (PLUR MAN1))>

更具体地说,以 er 结尾的名词的意义可以系统地从其动词形式推导出来。例如,假设动词“murder”的 SEM 为:

$\lambda o \lambda a (\text{MURDER1} * [\text{AGENT } a] [\text{THEME } o])$

可以用同样的形式表示关系名词“murderer”。根据语法 12.8 中的规则,“the murderer of John”的逻辑形式可表示为:

$\langle \text{THE } p1 (\text{MURDER1 } m1 [\text{AGENT } p1] [\text{THEME } (\text{NAME } j1 \text{ “John”})]) \rangle$

并非所有的 -er 形式都可定义为动作动词及其施事者。例如,“owner”也用动词“own”来定义,但它并没有一个施事角色。动词的条目需要进行扩展以便对 -er 形式给出正确的解释。

另一类重要的名词包括动词本身的名词化形式,比如“destruction”,它既可以指摧毁动作,也可以指摧毁动作的结果。这类名词可带 of 形式的 PP 补语来表示 THEME 角色,比如名词短语“the destruction of the city”。与之相似,也可用介词“by”引入 AGENT 格,比如“the destruction of the city by the Huns”,对它的分析指向一个特定的破坏动作:

$\langle \text{THE } d1 (\text{DESTROY1 } d1 [\text{AGENT } \langle \text{THE } h1 (\text{PLUR HUN}) \rangle] [\text{THEME } \langle \text{THE } c1 \text{ CITY} \rangle]) \rangle$

所有格形式既可以表示 AGENT 格(the Huns' destruction),又可以表示 THEME 格(the city's destruction)。

12.7 其他语义问题

本节简要介绍语义解释中的其他几个问题。这些问题虽然也很重要,但在本书中无法再详细阐述,包括不可数名词、泛指、意愿运算符和名词-名词修饰语。如果对这些问题进行详细讨论,可能需要一到两章。

12.7.1 不可数名词

到目前为止,例子中出现的名词短语基本上都是可数名词短语,比如“the dog”,“a few clowns”,等等。之所以称为可数名词短语,是因为它们所指的个体是可数的。因而可以说你有一条狗、有三条狗或者比别人的狗多。包含可数名词的单数名词短语指的是个体,而复数指的是集合。与此相反,不可数名词短语,比如“some water”和“the hot sand”,并不指某个个体或集合,而是大量出现的物质。不可数名词可以用度量单位来衡量,比如“three quarts of water”或“five pounds of sand”。如果意义不变,不可数名词不能以复数形式出现,也不能出现在带量词“each”和“every”的名词短语中。

特别要注意的一点是,虽然名词被分成可数名词和不可数名词,但大多数都可以被包含它们的名词短语强制变形。比如,因为英语中只能将单个不带修饰成分的名词短语(比如“sand”,“very clean water”)看成不可数,如果在这种结构中使用了一个可数名词,会将它的意义强制变为不可数。比如,“We ate deer for dinner”,那么 NP “deer”就是不可数名词,意思是鹿肉。相反,虽然“water”是不可数名词,但名词短语“many waters”将其意义强制变为可数,代表湖的集合。

12.7.2 泛指

句子的另一种复杂形式涉及泛指用法。用到泛指的句子对世界做出一般性断言,比如下面的句子都对狮子做出一般性结论:

Lions are dangerous. (狮子是危险的。)

The lion is a dangerous animal. (狮子是一种危险的动物。)

A lion can be dangerous. (狮子是危险的。)

在泛指解释中,名词短语既不指个体,也不对个体的集合进行量化。即便某些狮子是温顺的,句子“Lions are dangerous”也为真。事实上,即便类里的大多数对象都不具备这个性质,泛指句也可为真。例如,Carlson(1979)就指出,即便大多数海龟是公的,从来不下蛋,说海龟能下大约100个蛋也是成立的。

更确切地说,泛指看起来是对所述对象的类别或种属做论断。对如何解释此类事实的讨论将推迟到后面讨论知识表示和推理的章节中再讲。而在此,需要在逻辑形式语言中引入不同的特征。尽管现在有几种不同的理论来解决此类问题,但其中大部分都需要在本体中引入类的概念。类和个体或个体的集合不同,所指的是物种、物质或个体的类型。这样,无须确认某个种类的个体是否都具有某种特定的性质,该类就可以具有这种性质。当然,常见的情况是,如果类有某种性质,那么属于这个类的所有或绝大多数个体也都会具有这个性质。例如,如果知道“Dogs have four legs”(狗有四条腿),就可以推断大多数狗都有四条腿;但是,这并不能排除可能有三条腿的狗。类还可能拥有某些属性,它的个体反而不具备。例如,说“Dodos are extinct”(渡渡鸟灭绝了)并不意味着某只渡渡鸟灭绝了。相反,灭绝只能是“dodos”这个类(这个例子中是物种)的性质。

需要注意的是,并不是所有不带修饰的复数名词都要作为泛指来理解。这种情况很常见,比如句子“Large dogs have large noses”(大型犬有大鼻子)。这种对大型犬的性质所下的论断不是说每条狗有很多大鼻子(集合解读),或者说它们拥有某种叫鼻子的物体(泛指解读),而是说它们有一个特殊的鼻子(存在解读)。因而,不带修饰的复数名词需要根据其功能做出不同的解释。

除了不带修饰语的复数形式之外,很多其他的句法元素也能表示泛指意义。事实上,正如本节一开始所述,泛指句可能也会使用单数定指或不定指名词语。泛指句的另一种常见标志是使用一般现在时。例如,考察“Large dogs run over the hill”(大狗爬山)和“Large dogs ran over the hill”(大狗翻过了山峰)的区别。第一句话很可能是泛指句,而第二句可能只是对某个特定事件的描述。但是,这两句话都有泛指和非泛指两种解读。用来表示一段时间的时间修饰语也可以表示泛指,比如“Large dogs ran over the hill in those days”(大狗们在那几天里翻过了山峰),是表示过去时间的泛指句。其他副词,比如“often”和“rarely”,也常常表示泛指。

12.7.3 意愿运算符与辖域解析

诸如“believe”或“want”这样的动词以限定从句作为补语,并以一种复杂的方式与定指或非定指描述进行交互。举例来说,句子“Jack believes a man found the ring”有歧义,一种意义是 Jack 知道某个人,他相信这个人找到了戒指;还有一种意义是 Jack 所相信的事情是戒指被某个人找到了。这些歧义通常可以用不同的辖域表示出来。要理解这个问题,需要更仔细地研究这些句子的逻辑形式。

正如第8章所述,这类动词的语义很复杂,因为它们不支持等价替换这样一个常见的推理规则。例如,如果 John 是个子最高的人,而且已知 John 吻了 Sue,就可以推断个子最高的人吻了 Sue。但类似的推论对表示相信的句子却并不有效。比如,已知 Sam 相信 John 吻了 Sue,不能得出结论说 Sam 相信个子最高的人吻了 Sue,因为 Sam 可能不知道 John 是个子最高的人。更糟的是, Sam 可能认为 George 个子最高。在这类运算符的辖域内等价项无法互相替换,因此常称之为指代不透明(referentially opaque),而一般的谓词常叫做指代透明(referentially transparent)。

不透明运算符又引入一种新的辖域歧义形式,即量词可能在此运算符的辖域内,也可能在它的辖域外。这种形式的辖域指定可用于表示“John believes a man kissed Sue”中的歧义,其无辖域逻辑形式是:

(PRES (BELIEVE b1
[EXPERIENCER (NAME j1 “John”)]
[THEME (PAST (KISS k1 <A m1 MAN1> (NAME s1 “Sue”))))))

第一种解读是 John 知道这个人是谁,可以解释成“*There is a particular man whom John believes kissed Sue*”。其形式为:

(A m1 : (MAN m1)
(PRES (BELIEVE b1
[EXPERIENCER (NAME j1 “John”)]
[THEME (PAST (KISS k1 m1 (NAME s1 “Sue”))))))

这个形式中包含一种称为实体信念(*de re belief*)的现象,也就是相信某一个特定的人。第二种情况则可解释为 John 只是相信有一个人吻了 Sue,尽管他不知道是谁。其形式如下:

(PRES (BELIEVE b1
[EXPERIENCER (NAME j1 “John”)]
[THEME (PAST (A m1 : (MAN m1)
(KISS k1 m1 (NAME s1 “Sue”))))))

这个形式包含命题信念(*de dicto belief*),也就是相信某一个命题。

关于这种现象还有很多别的例子,其中有些相当复杂。例如,很早以前我参加一个聚会,有一个人走进厨房说“I’m waiting for a phone call”。这句话后来导致某些理解上的混乱,因为我按照通常的解释理解成他在等某一个人打来的电话。但后来证明那个说话者只是想听到电话铃响,因此任何电话都可以。这个例子相当有趣,因为原句所要表达的信息本该可以不带歧义地由相当特殊的句子“I’m waiting for any phone call”传递出去。而这个句子之所以特殊只是因为表示其正确意义所需的上下文非常罕见。

12.7.4 名词 – 名词修饰语

带名词 – 名词修饰语的 NP, 比如“car paint”, “soup pot handle”, “water glass”和“computer evaluation”是公认的分析难题。句法形式基本没什么帮助,而类似于用于分析多个形容词语义约束的一般语义约束也不存在。这里,有两个主要问题。一是判断到底谁修饰谁,这个问题和多形容词的问题类似。另一个问题是找出修饰名词和被修饰名词之间的语义关系。名词的关联有几种常见的模式。由两个名词 X 和 Y 的序列组成的 NP 有以下几种常见关系:

- Y 是 X 的一部分(pot handles)。

- Y 用于某些和 X 相关的活动(car paint)。
- Y 以 X 为材料制成(stone wall)。

因此,我们说“pot handle”是一个把手,是壶的一部分;“car paint”是一种用于喷车的涂料;而“stone wall”是由石头垒成的墙。

和某些格相关联或者标识特定动词的格的名词,可以用这些格来进行分析。例如,对于名词短语“the contest winner”(比赛优胜者),名词“winner”标识 WIN 这个动作的 AGENT 格,而“contest”可以认为充当 THEME 格。因而,这个 NP 可以分析成:

<THE p1 (& (PERSON p1) (WIN w1 p1 <THE c1 CONTEST>))>

另一方面,名词短语“the car evaluation”(汽车评测)指的是一个评测事件,而“car”可以认为充当 THEME 格。如果修饰语可以充当多个格,NP 就会有歧义。例如,“the computer evaluation”既可能指对计算机的评测,也可能指用计算机进行的评测。

在最终的分析中,很多名词-名词修饰语不能用基于推理的策略来解析,而是直接从上下文中取得信息。

在逻辑形式中,如果没有有力的证据表示更明确的特定关系,那么用歧义谓词修饰符 N-N-MOD 来表示,即虽然存在某种关系,但在没有上下文的情况下无法判断是什么关系。例如,诸如“the smoke box”这样的名词短语没有明显的修饰关系,其逻辑形式是:

<THE b1 ((N-N-MOD SMOKE1 BOX1) b1)>

这就需要上下文解释来判断所述对象的类型和该对象的其他性质。

12.8 小结

本章研究了很多问题,这些问题在以前章节的讨论中被忽略了。其中,很多问题和运算符的辖域指定及指代约束相关。我们在书中提到,量词的选择和句法结构对句子的辖域识别和解释的优选都有很大关系。此外,代词和其他指代表达式之间的句内互指存在很重要的结构约束。本章还探讨了形容词修饰语并提出必须区分几种不同类的形容词,这样才能得到正确的解释。最后,我们讨论了具有复杂语义结构的名词,用关系或相关动词形式对其进行了定义。

12.9 相关工作与深入阅读材料

判断量词辖域已经成了开发自然语言数据库查询系统人员关注的焦点,本书中描述的很多启发式方法已经用在 Woods (1978), Pereira (1983), Saint-Dizier (1985), Grosz 等 (1987) 和 Alshawi (1992) 的工作中。VanLehn (1978) 做了一系列实验来测试书中提到的优选策略。虽然在某些情况下,人们给出了一些确定无误的优选结果,但在很多情况下还是上下文的作用主导了这个过程。Hobbs 和 Shieber (1987) 描述了一种算法,用于生成所有可能的辖域,只排除那些从语言学的角度来看在任何上下文中都不能接受的结果。辖域的更一般解决方案,包括对副词和命题运算符的处理,在 McCord (1986) 中有详细阐述。

最早在语义解释中将量词结构和语义形式分开的系统之一是 LUNAR 系统 (Woods, 1978)。它属于 ATN 框架,自那时起到现在已在很多方法中得到运用。Cooper 在形式语义的框架内发展了一套理论,该理论使用一种称为量词存储的技术,这种技术使用了同样的思想。Pereira 和

Pollack(1991)提出了一个有趣的方法,对这种基于存储的方法进行了形式化,并且不需要对会话进行假设。

现在,基本上还没有专门面向句内回指方面的著作。本书提到的其中一种算法在 Hobbs(1978)中有所介绍。这个算法有效运用了本章提到的约束,根据反身代词约束来排除候选项,然后严格按照先行词和代词的近似度对剩下的候选项进行启发式排序(也就是说,选择最邻近的先行词)。

关于量化、回指和语言学中绑定理论方面的著作很多。最佳入门是 Chierchia 和 McConnell-Ginet(1990,第3章)这样的教材。May(1985)是讲述用句法转化处理量词辖域判定的好书(他称之为量词提升)。Reinhart(1983)是最早基于绑定约束进行回指分析的著作之一。*Government and Binding* 是更一般性的介绍基础语言学理论的参考书,具体内容可以在 Chomsky(1981)中找到。使用这种方法的计算语言学方面的工作可参见 Berwick 和 Weinberg(1984)以及 Stabler(1992)。

Chierchia 和 McConnell-Ginet(1990)中有对形容词短语形式语义的一般性探讨。更详细的讨论可以参考 Kamp(1975)和 Klein(1980)。McCawley(1993)中有对形容词比较级的精彩综述。对形容词的计算意义的解释可以在 Ballard(1988)和 Alshaw(1992)中找到。

de Bruin 和 Scha(1988)讨论了关系名词。Finin(1980)对名词-名词修饰语进行了分析。Hobbs 等(1993)用一个很模糊的谓词 NN 来表示名词-名词修饰语,并用上下文信息对其进行解析。

类的概念及其在泛指句中的应用最早由 Carlson(1979;1982)引入,从那时起有很多著作对这个领域进行了探讨。Schubert 和 Pelletier(1987)对其中的几种方法进行了很好的综述。Pelletier(1979)是关于不可数名词的一部出色的文集,McCawley(1993)则对处理不可数名词和泛指的一些方法给出了很好的综述。

语义解释中,我们没有展开讨论的另一个重要领域是描述地点的短语的解释(主要是介词短语)。这个领域的一本很好的参考书是 Herskovits(1986)。

12.10 习题

1. 【易】通过分析其辖域歧义,对下面每个句子给出其所有不相同的逻辑解释。但如果两个不同的辖域有相同的解释,那么只须列出其中一个。

Each man gave a boy a present.

Some man from a club stole every present.

Most people from all companies ate some peaches.

2. 【易】对下面各句中带索引 i 或 j 的代词,给出其所有先行词。注意,要考虑到 12.4 节中提到的所有约束。

Jack₁ realized that he_i would see Sam₂ at the party₃.

After Jack₁ left the room₂, Sam₃ saw him_i run to his_j car₄.

Even though Sam₁ apologized, Jack₂ wants to kill him_i.

3. 【易】把下面的形容词分成三类:交集、子交集或两者皆非。证明你的结论。对于对应句子中充当宾语的名词短语,给出合理的逻辑形式。

fake	I bought a fake turtle.
sluggish	I bought a sluggish turtle.
expensive	I bought an expensive turtle.
dead	I bought a dead turtle.
smelly	I bought a smelly turtle.

4. 【中】对下述句子绘出和图 12.3 所示一样详细的分析树。

When does the flight to each city leave?

When did the flight that each man took leave?

注意,要使用语法 12.2 和语法 12.4 以及 12.3 节所述的辖域指定策略。

5. 【中】对下面的句子,给出每个名词短语的语义分析,要包含互指约束。

Jack saw the man beside him.

Sue saw Jill's book about herself.

Every man gave his book to him.

6. 【中】

- a. 写一段程序,在已知逻辑形式的条件下生成所有可能的量词辖域,并且要求和 12.1 节阐述的约束相一致。因此,给定下面的表达式作为输入:

```
(PRES (SIT s1 [AGENT <A m1 MAN1>]
          [AT-LOC <THE k1 (& (KITCHEN k1)
                              (IN k1 <(EVERY h1 HOUSE)>))>]))
```

程序应输出:

```
(m1 (k1 h1))
((k1 h1) m1)
(m1 h1 k1)
(h1 m1 k1)
(h1 k1 m1)
```

构造一组测试样例来验证该算法的功能。

- b. 对你的程序进行扩展,使之能排除逻辑冗余的辖域,从而生成一个较小的解释集。

确认你用来验证冗余的等式,并证明这些可互相替换的形式真正逻辑等价。

7. 【中】思考句子“In every house, a man fixed his sink”的无辖域形式:

```
(PAST (FIX1 f1
        [AGENT <A m1 (MAN1 m1)>]
        [THEME <THE s1 (& (SINK1 s1)(POSS-BY s1 (PRO h1 HE1)))>]
        [AT-LOC <EVERY h2 HOUSE1>]))
```

绘出这个句子的句法树,并对每个代词形式给出其所有可能的先行词。在对这句话最直观的解读中真正的互指关系是什么? 这个定指描述看成指代性的还是存在性的?

8. 【中】12.5 节讨论了形容词的用法,比如“average”,这些形容词是某些集合的函数。“average”也可以当做名词,比如“in the class average”或“the average of the best students”。分析这些名词的正确关系,并给出它们的一组语义解释规则。

第三部分 上下文和世界知识

本书的第三部分主要讨论语言的上下文处理,主要包括知识表示、篇章结构的推理和语用的推理。术语“篇章”指任何形式的多个书面句子或者多个口头句子所组成的一段话。在计算机应用中,篇章最重要的形式是文本(如文章和书)和对话(即通过口头或计算机键盘进行的两个或多个人的交流)。很多篇章模型都适用于书面语和口语,所以,书面的和口头的句子将同等对待,说话者和听者也可以分别对应于作者和读者。

将上下文分为情景上下文和篇章上下文是有用的。就一个会话而言,情景上下文包括特定的场景(时间、地点和对每一个会话者都可见的对象)和通用的场景(涉及会话者对于世界的背景知识)。篇章上下文包括局部篇章上下文(或称为局部上下文,指的是前面句子中的详细信息)和全局篇章上下文(指的是整个会话的信息,包括正在阐述的话题)。下面的表格给出了一个特定会话中上下文信息的例子。在这个会话中,对话双方为 John 和 Mary, Mary 刚刚说了“My coffee is cold”(我的咖啡凉了)。

	特定/局部上下文	通用/一般上下文
情景上下文	Mary 是说话者, John 是听者。现在是下午三点钟。他们在学生会里。他们每人有一杯咖啡,放在位于两个人之间的桌子上,……	他们在美国。总统是比尔·克林顿。有一个月亮围绕地球在运转。鸟一般会飞……
对话上下文	最近一个句子的分析树是(S(NP my coffee)(VP is cold))。My coffee 指 Mary 身边的那杯咖啡。Mary 断言那杯咖啡是凉的。	John 和 Mary 一直在讨论他们的物理课上的一个项目。然后,他们开始讨论学生会的食物质量。

第 13 章介绍了知识表示和推理的问题,为讨论知识和推理怎样影响语言解释奠定了基础。第 14 章探讨了上下文解释的中心问题,即主要利用局部篇章上下文来确定名词短语的指代,例如代词和定指性描述的指代问题。第 15 章主要介绍情境上下文的问题,探讨利用世界知识和特定情境来解释语言的方法。第 16 章描述与全局篇章上下文相关的问题,集中于确定篇章的结构。最后,第 17 章讨论会话 agent 的问题,探讨怎样从说话者的内容来确定说话者的意图,并且这些意图是怎样通过语言手段来实现的。

第 13 章 知识表示和推理

在前面的章节中,很多问题没有解决的原因在于需要结合上下文知识。上下文知识包括两个重要的方面:(1)世界通用知识;(2)与发生语言交流的场景相关的特定知识。为了分析这两类上下文,需要给出表示知识和进行推理的形式化描述,这方面的研究称为知识表示。

对于不同的研究者,知识表示代表不同的事情。对于一些研究者来说,知识表示强调的是用来表述知识的语言的结构,例如逻辑、语义网络、框架或其他专门设计的表示形式。对于另一些研究者来说,知识表示指的是句子内容,例如需要什么谓语以及句子是怎么组织在一起的。这两种不同的观点都是重要的。实际上,在很多时候,这两种观点的分歧只是由于强调知识表示的一个方面,而忽视了另一方面。

在这里,我们不详细讨论知识表示的形式化问题,而是详细探讨知识和推理是怎样帮助语言理解的。因此,我们采用了一种基于一阶谓词演算的抽象表示形式。这样做可以在引入很少的新的知识和记法的情况下讨论相关的表示问题。这并不是说,在一个系统中要实现下面的知识表示只能直接用逻辑公式并采用定理证明技术作为推理的模型。实际上,只要具有相当的描述能力,下面的知识表示可以是语义网络、描述逻辑、基于框架的系统、连接模型或其他任何形式。很多知识表示系统都满足这一条件。

13.1 节讨论知识表示的常见问题,后面两节将详细探讨贯串于后续章节的知识表示的抽象语言。其中,13.2 节讨论基于一阶谓词演算的简单表示方法。13.3 节讨论类似于框架的知识表示方法,作为对固定模式对象和情境进行信息聚类 and 默认表示的手段。13.4 节讨论了从逻辑形式语言映射到知识表示语言中的一个重要问题,即自然语言中的复杂量词的表示。后面的几节是可选的,然而又是知识表示中的重要问题。13.5 节研究语言中的时间信息的表示问题,这种信息表现为时态类和体态类。13.6 节讨论自动演绎中的基本概念。13.7 节讨论了利用程序语义学的方法,并给出该方面在问题回答系统中的应用。13.8 节讨论了开发综合推理系统中的一些问题,在此过程中,推理技术的差别是依赖于需求信息的不同而变化的。

要阅读这一章,读者必须对一阶谓词演算有基本的了解,至少要达到附录 A 的水平。

13.1 知识表示

在任何知识表示系统中,都有两类重要的知识:(1)世界知识,(2)当前情境的特定知识。目前,我们已经探讨了一些一般性的世界知识,例如类型的层次体系、部分/整体的关系,等等。这包括世界的一般约束信息和语言中的一些术语的语义定义。通用知识主要描述的是某一类事物的信息,而不是描述某一特定个体的信息。例如,OWN1 表示人和对象之间的一种联系,而不是指某个特定的人(比如 John)拥有一辆特定的车。第二类知识,即某一个体的信息,在语言理解中也是很重要的,是我们称之为所理解句子的特定场景的主要组成部分。实际上,所有的知识表示系统或多或少都支持这两种层次的推理。

世界知识对于解决语言解释的很多问题是非常必要的,其中最重要的问题之一是歧义消解。例如,下面两个句子中介词短语附着的歧义消除完全依赖于读者对于 reading(阅读)和 evolution(进化)所需时间的背景知识:

I read a story about evolution in ten minutes.

(我花十分钟读了一篇关于进化的文章。)

I read a story about evolution in the last million years.

(我读了一篇关于近一百万年来的进化的文章。)

当前情境的特定知识对很多问题都很重要。例如,确定名词短语的指代,以及根据在当前情境中是否有意义来进行词义排歧。

在非正式的场合,知识表示有时候可以指智能系统中知识和信念的表示。但是,因为我们将后面对知识和信念这两个术语进行准确的定义,所以这里采用更中性的术语来介绍。知识表示包括命题库和一系列的推理技术。其中,命题库称为知识库,推理技术用于在给定当前知识库的前提下推导出新命题。当知识库中的原始命题正确时,如果只推导出正确的新命题,那么推理技术是可靠的。然而,后面我们将看到,不是所有有用的推理技术都是可靠的。

用于定义知识库中的命题的语言称为知识表示语言(KRL, knowledge representation language)。KRL 可以与逻辑形式语言相同,但实际上,它们经常不相同。这两种语言是从不同的需求推导出来的。逻辑形式语言必须具有很强的表达能力,以便于简化语义解释过程并能够有效地消除歧义。而知识表示语言必须支持特定领域内的有效的和预测性的推理。换言之,知识表示语言要很容易定义命题集合和建立计算模型。其中,命题集合可以从特定知识库中推理得到,计算模型用来在合理的时间内完成推理过程。

例如,有关量词处理的情况。在逻辑形式语言中引入了很多量词,这些量词与英文中量词的不同含义相对应。这样,就可以实现更精确的消除技术,例如能够利用实际量词之间的细微差别(比如 each 和 every)来消除量词辖域指定的歧义。而目前,大多数知识表示语言只有很少的量词,通常情况下只有一个量词——这个语言可以接受全称量词。这样,推理机制就很容易定义。通过保持两种语言独立性并在两种语言之间定义一种映射函数,可以同时拥有这两种语言的优势。当然,这一方法的成功与否取决于映射函数是否能够有效地定义。

要想得到一种最好的方法,同时能够满足强大表达能力的逻辑形式和高效的知识表示的需要,需要做更实质性的研究工作。在目前的知识条件下,分别维护一种逻辑形式和一种知识表示语言可能是最好的折中选择。

给定一个知识库中的所有公式,或者给定表示某个命题意义的所有公式,如果公式 P 一定为真,我们称这个知识库(或这个命题)蕴涵 P。为了理解语言,需要得到的结论往往都不是这个句子的蕴涵,而是句子的隐含意义。隐含意义一般指可以从句子中推导出来的结论,但是在特定情况下可能被明确地否认。例如,句子“Jack owns two cars”蕴涵了 Jack 有一辆车(即,不能否认这个事实),但是,可以隐含 Jack 没有三辆车的意思。不过下文中,你也可以继续说:“In fact, he owns three cars”(实际上, he 有三辆车)。知识表示系统必须能够支持这两种形式的推理。

13.1.1 推理的类型

我们需要不同的推理形式来理解自然语言。推理技术可以分为演绎形式和非演绎形式。演绎形式的推理根据蕴涵的逻辑概念来判断。给定一系列事实,演绎推理过程将只得出从逻辑上完全遵循所有这些事实的那些结论。非演绎形式的推理可以分为几类。例如,根据实例学习一般规律的推理技术(归纳推理)和从结果来推导原因的推理技术(溯因推理)。

通过下面的公理,可以比较溯因推理和演绎推理:

$$A \supset B$$

演绎推理将利用这个公理从 A 推导出 B;而诱导式推理利用这个公理从 B 推导出 A,因为 A 是 B 的原因。

很多系统允许利用默认信息。默认规则是运行有很多例外情况的推理规则,因此是可以推翻的。如果用符号 $A \Rightarrow B$ 来表示默认信息,默认推理规则可以表述如下:如果 $A \Rightarrow B$,且 A 为真,并且无法证明 $\neg B$,则可以得到结论 B。有人认为,默认规则可以给泛指句提供很好的解释。比如句子“Birds fly”可以用下面的一阶谓词演算公式表示为:

$$\forall x \text{ BIRD}(x) \Rightarrow \text{FLIES}(x)$$

可以得到这样的结论:对于任意一只鸟 B,如果不能证明它是 $\neg \text{FLIES}(B)$,就可以推理得到 $\text{FLIES}(B)$ 。换言之,任意一只特定的鸟被认为是会飞的,除非明确说它不会飞。

可废止的规则在知识表示方面引入了一系列复杂问题。如果没有这些规则,大部分表示都是单调的,这是因为增加断言将只增加公式的数目。具体来说,在一个单调的表示系统中,如果知识库 KB1 蕴涵一个结论 C,并且如果往 KB1 中增加了一个新的公式,形成了一个新的知识库 KB2,那么,KB2 仍将蕴涵 C。但是,如果利用默认规则,这种表示方法就不成立。例如,包括下列公式的知识库:

$\text{Cat}(\text{Sampson})$	Sampson is a cat. (Sampson 是一只猫。)
$\text{TabbyCat}(\text{Sampson})$	Sampson is a tabby cat. (Sampson 是一只虎斑猫。)
$\forall c. \text{Cat}(c) \Rightarrow \text{Purrs}(c)$	Cats purr. (猫会咪咪叫。)

给定这一知识库,因为没有信息与 $\text{Purrs}(\text{Sampson})$ 相冲突,所以可以利用默认规则得出结论 $\text{Purrs}(\text{Sampson})$ 。另一方面,如果增加一个“no tabby cats purr”(虎斑猫不会咪咪叫)的事实,那么这一扩展的知识库将不再蕴涵 Sampson 会咪咪叫。

除了默认规则外,还有另外一些有用的技术,也能够引入非单调性的结论。例如,封闭世界假设(CWA, closed world assumption)认为知识库包含了特定谓词的完整信息。例如,对于 CWA 支持的特定谓词 P,如果一个涉及 P 的命题从知识库得不到证明,那么此命题的反命题将假定为正确的。我们分析一个航空时刻表的数据库查询的应用场景,知识库中包含了航班信息(如 FDG100 航班从 Rochester 到 Boston),但是没有明确包括否定信息(比如 FDG100 不飞 Chicago 或者没有 FDG455 航班)。只有推理过程中认为封闭世界假设成立,才能得出这样的结论。

框 13.1 非单调逻辑的语义表示

我们可以利用最小模型的概念为很多非单调的结构提出一个模型论语义学。例如,考虑谓词 P 的封闭世界假设。我们可以对知识库中的所有模型定义一个顺序:

$$m_1 <_P m_2 \text{ iff } I_{m_1}(P) \subseteq I_{m_2}(P)$$

也就是说,对于谓词 P ,当且仅当使 $P(x)$ 为真的所有模型 m_1 中的对象是使 $P(x)$ 为真的所有模型 m_2 中的对象的子集,模型 m_1 小于模型 m_2 。基于这样的顺序,与谓词 P 相关的最小模型将包括这些模型 $\{m \mid \text{there is no } m' <_P m\}$ 。给定知识库 K ,可以证明在对谓词 P 做了封闭世界假设的情况下从知识库 K 中推导出来的结论,与 K 的最小模型(关于 P)推导出来的结论是完全一样的。

对封闭世界假设进行形式化的另一个方法是增加一个公理到知识库中并仍保持其封闭性。这个公理称为谓词完备性公理。考虑包含下面谓词的知识库:

$$P(A), P(B), Q(C), Q(A)$$

关于 P 的谓词完备性公理为:

$$\forall x. P(x) \equiv (x=A \vee x=B)$$

从这个公理,可以得到这样的结论: P 只对 A 和 B 才为真。正如我们所愿,这样会得到 $\neg P(C)$ 。可以看到知识库中扩充了谓词完备性公理以后的模型的集合,与初始知识库中的关于 P 的最小模型集合相同。谓词完备公理也能够处理更复杂的知识库。例如,如果知识库也包含公理:

$$\forall s. Q(s) \supset P(s)$$

关于 P 的谓词完备性公理为:

$$\forall x. P(x) \equiv (x=A \vee x=B \vee Q(x))$$

然而,谓词完备理论不能应用于所有知识库,因为它不能处理那些 P 的正例出现多次的公理。(McCarthy, 1980)提出了一个更广义的概念,称为“界限”(circumscription),它能够产生一个合适的封闭公理,但需要求助于二阶逻辑(即对谓词进行量化)。

13.1.2 推理技术

知识表示系统中有两种推理技术,程序性推理和描述性推理。大部分系统把这两种方法在某种程度上结合起来,这些做法从完全的描述性表示逐步过渡到完全的程序性表示,从而具有连续统一性。这种连续统一的描述性一端是基于逻辑的定理证明器。在这一推理方法中,知识库是一个公理的集合,而推理过程通过演绎性的定理证明算法来实现。在一个严格的描述性推理系统中,重点是给知识表示中的表达式赋予某种形式的语义,而这些表达式是独立于推理模块的。

另一方面,程序性推理系统强调的是表示的推理方面,在极端情况下,知识库中的表达式没有被赋予任何其他意义,完全独立于程序是操作这些表达式的方式。基于程序性的表示系

统的一个例子可以是这样的系统:没有任何关于数学知识的外在表示(如 Peano 算术公理)的情况下,利用计算机内嵌的算术函数来计算算术表达式的值。实际上,程序性系统在定义完备的领域的特定推理任务中是非常有效的。但更多的情况是,由于形式化的缺乏,导致很难对其进行分析。

考虑这样的例子。第 10 章介绍了语义网络的继承技术。这一推理过程可以是程序性的或描述性的。一个完全的描述性推理方法中,将关于子类型和角色的事实刻画为公理,通过标准的演绎推理得到继承的属性。比如,给定图 13.1 中的简单网络,下面的一阶谓词演算公理可以表示这一信息:

1. $\forall x. ACTION(x) \supset \exists a. AGENT(x, a) \& ANIMATE(a)$
2. $\forall a \exists x. ACTION(x) \& AGENT(x, a) \supset ANIMATE(a)$
3. $\forall x. OBJ/ACTION(x) \supset ACTION(x)$
4. $\forall x. OBJ/ACTION(x) \supset \exists o. THEME(x, o) \& PHYSOBJ(o)$
5. $\forall o \exists x. OBJ/ACTION(x) \& THEME(x, o) \supset PHYSOBJ(o)$

利用这些公理,可以证明类 OBJ/ACTION 继承了 AGENT 的角色。换言之,对于任何对象 A,如果 $OBJ/ACTION(A)$ 为真,则可以证明 A 具有 agent 的功能;即根据公理 1 和公理 3 可以得到:

$$\exists a. AGENT(A, a) \& ANIMATE(a)$$

正如第 10 章所述,上面例子的一个程序性版本是这样:从指定的节点 OBJ/ACTION 开始,寻找该节点的所有角色,然后顺着 S 边到达父类 ACTION,再找到附属于它的角色。这一过程得到的所有角色的集合就是答案。这样,任何 OBJ/ACTION 都具有 AGENT 角色,此功能是从类 ACTION 继承来的。

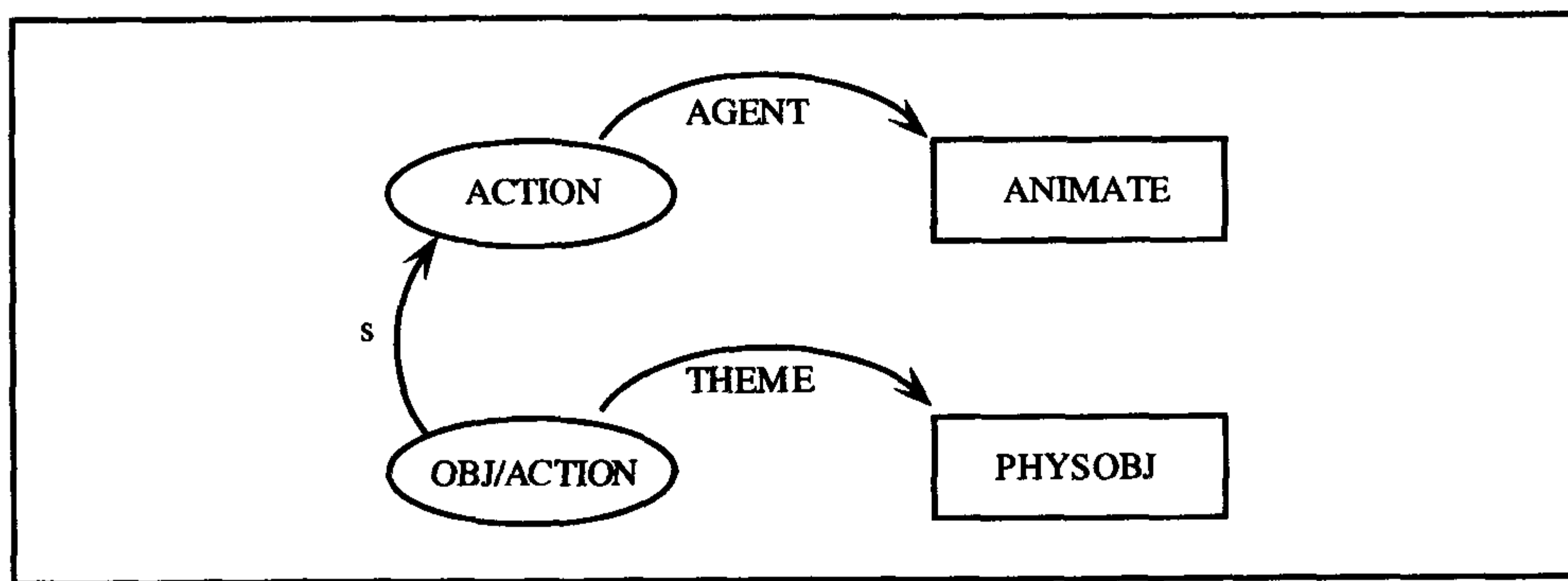


图 13.1 简单继承的例子

虽然这两种技术得到相同的结果,但是前者利用的是演绎技术,而后者利用的是遍历整幅图的方法。前者具有较严格的定义,而后者可能执行效率更高。类似于这个例子,在可以证明这两种技术能够得到相同结果的情况下,可以采用这两种方法的长处,即严格定义的语义和高效的推理过程。

13.2 基于一阶谓词演算的知识表示

本书中所使用的知识表示语言(KRL)将是一阶谓词演算的一个扩充版本。需要注意的

是,知识表示语言与推理技术没有必然的联系,即确定了表示语言后,对推理方法没有限制。后面,我们将看到 KRL 可以用于演绎推理,也可以应用于程序性推理技术。

前面已经介绍了一阶谓词演算的语法,这里就不再详述。我们将重点介绍怎样扩展标准一阶谓词演算以便能够表达自然语言句子,以及这一语言与逻辑形式语言的区别。一种语言主要包括的术语有常量(如 John1)、函数[如 father(John1)]、变量(如 x, y)。需要注意的是,逻辑形式语言中不使用常量,而是用篇章变量来表示所有的东西,以保持表示的上下文无关性。而在知识库中,常数用来表示一个特定的个体,例如,逻辑形式项(NAME j1 "John")表示这个短语的指代对象的名字为"John"。特定上下文中的实际人名可以用知识库中的常数 John1 来表示。

在知识表示语言中,利用受限的量化方法是比较方便的,这就使它与逻辑形式语言中的通用量词符号类似。限定紧跟在变量后面,并用冒号隔开。如第 8 章所述,对于存在量词和全称量词来说,这一符号可以看做一种缩写,而且没有扩展这一语言的表达能力。因此,

$$\begin{aligned} \exists x : \text{Man}(x) \text{ Happy}(x) \text{ is equivalent to } \exists x . \text{Man}(x) \ \& \ \text{Happy}(x) \text{ and} \\ \forall x : \text{Man}(x) \text{ Happy}(x) \text{ is equivalent to } \forall x . \text{Man}(x) \supset \text{Happy}(x) \end{aligned}$$

我们也需要 equality 谓词($a = b$),表示项 a 与项 b 具有相同的意义。给定一个简单的包括常数 a 的命题 P_a ,如果 P_a 是真命题并且 $a = b$,那么 P_b 也为真命题。除了 a 被 b 代替外,命题 P_b 与命题 P_a 相同。

很多知识表示系统中没有明显地利用量词。虽然系统中包含变量,但是这些变量起的作用更像全称量化的变量。例如,知识库中的公式($P \ ?x \ A$)在意义上与一阶谓词演算中的公式 $\forall x . P(x, A)$ 对应。可以通过 Skolem 化的方法处理存在量词变量,即用一个没有被使用过的新的常量来代替这个变量。例如,公式 $\exists y \ \forall x . P(x, y)$ 可以用知识库中的公式($P \ ?x \ \text{Sk1}$)来表示。其中,Sk1 是一个没有使用过的新常量,表示已经存在的一个对象。量词的辖域依赖关系用一个称为 Skolem 函数的新函数来表示。例如,公式 $\forall y \ \exists x . P(x, y)$ 可以用知识库中的 ($P \ (\text{Sk2} \ ?y) \ ?y$) 来表示。其中,Sk2 是一个新的函数,它对每一个不同的 $?y$ 产生一个(可能)不同的对象。在表示中常常把这两种方法结合在一起,其中,全称量词仍然存在,而存在量词被 Skolem 化了。例如,公式 $\forall y \ \exists x . P(x, y)$ 可以写为 $\forall y \ P(\text{Sk1}(y), y)$ 。可以证明这些不同的表示方法是等价的。

假定基本的表示语言是一阶谓词演算,这样做并没有对知识表示的方式带来很多限制。特别是没有限制谓词是什么样的,所以在选择什么样的谓词方面有很大的余地。一种方法是对每一个词义定义一个谓词,这是逻辑形式语言采用的基本方法;另一种方法是,可以有一个预先给出的谓词集合,称为原语,然后使用这些原语来定义每一个词义。这两种方法各有一些优点。通过对每一个词义定义一个谓词,我们可以抓住语义相近的词义项的微小区别。例如,在知识库中,我们定义了 SAUNTERS1 来表示走路很慢,而且是在一种无忧无虑的心情下。这样,句子"Jack sauntered down the street"与句子"Jack walked down the street"的意义就有一些区别。当然,为了得到这种表达能力,要付出一些代价,因为我们要使用大量非常相似的公理来定义这些谓词。例如,谓词 SAUNTERS1 和谓词 WALKS1 的定义就有很大的重叠。

另一方面,在利用原语的方法中,差别微小的这些语义将简化为一个(或几个)刻画基本动作的谓词,如 MOVE-BY-FOOT(步行),因此推理规则的定义就建立在原语的基础上。这一方法的优点是单词之间的共同点可以刻画得非常简洁。但是,如果不使用推理规则,就很难抓住这

些语义之间的细微的差别。当然,要刻画“sauntering”(漫步)和“walking”(走路)之间的不同,需要再定义一个新的原语,比如一个关于情绪的原语。如“sauntering”可以定义为 MOVE-BY-FOOT和 CAREFREE-STATE(无忧无虑)。不过,语义分解得越复杂,使用原语表示的优势就越不明显,因为为了处理实际出现的各种情况,原语的数目会急剧增加。更严重的是,如果推理规则建立在原语谓词的复杂组合上而不是一个谓词上,推理过程会变得非常复杂。

对这两种不同方法的优缺点,我们可以探讨很多种折中的方法。例如,采用类型层次体系可以获得很多基于原语谓词表示的优点。如果我们断言 SAUNTERS1 和 WALKS1 都是更抽象的行为 MOVE-BY-FOOT 的子类,就可以从 MOVE-BY-FOOT 继承大部分共同属性,而不需要额外的公理。同时,我们仍然可以自由地为 SAUNTERS1 增加其他公理来刻画其特殊属性。

这一方法也允许我们处理不完整知识。假设系统只知道 SAUNTERS1 是走路的一种形式,不知道这个单词的任何其他信息,但仍然可以利用从 MOVE-BY-FOOT 继承得到的信息来进行这个单词所需要的大部分推理。另外,系统知道 SAUNTERS1 与 WALKS1 有一些不同,虽然系统并不知道原因。如果在以后的步骤中,系统获得了“sauntering”的其他信息,这个信息就可以采用递增的方式添加到系统中。

除了层次结构的关系以外,一个知识表示系统应该能够充分利用其他方法的优点来定义词义。有时,可以给出一个项的完整定义。例如,可以定义谓词“father”为“male parent”,即:

$$\forall x. FATHER(x) \equiv \exists y PARENT(x, y) \& MALE(x)$$

但是,大部分单词都不能被准确定义。例如,没有属性集能够准确定义自然界的类别(如狗、猫、椅子等)。它们可以表示为类型的层次体系,并且可以使用公理来表示一些可以表示出来的必要条件,但是不可能给出一个完备的定义。从一阶谓词演算的公理来看,意味着这些定义只表示单向的隐含。例如,DOG1 的一个公理可以表示为:

$$\forall x. DOG1(x) \supset CANINE(x) \& DOMESTIC-PET(x)$$

其中,CANINE 本身被定义为 MAMMAL 的一种类型,依次类推。这些公理刻画了一条狗的最重要的属性,但是并没有完整地定义这一概念。例如,某人有一只宠物狼,它满足狗的所有属性,但它不是狗。

从生成句子的角度看,知识表示语言中从单词中抽象出来的谓词越多,基于意义生成句子就越难。例如,假定下面的公式:

$$\forall p : ((MaleHuman\ p) \& \exists c. Parent(p, c)) . \\ MoveByCar(p, L1) \& Building(L1) \& Used-for-teaching(L1)$$

从这一公式很自然地对应于句子“All fathers drove to the school”(所有的父亲都开车去学校)。要生成这样一个句子,系统必须能够实现公式 $((Male\ p) \& \exists c. Parent(p, c))$ (字面上可表示为“male humans who have a child”,即单词“father”)和命题 $MoveByCar(p, L1) \& Building(L1) \& Used-for-teaching(L1)$ (字面上可表示为“moved by car to a building used for teaching”,即短语“drove to school”)。很明显,这要求有具体单词 father 和 drive 的意义的大量知识,以及在 KRL 中的公式和这些谓词之间进行复杂的匹配。如果没有谓词能够与 KRL 中的这些词义相匹配,匹配过程将十分复杂。如果引入了一些这种谓词,层次体系结构将为我们查找这个公式的实现方法提供帮助。例如,对抽象谓词“MaleHuman”,我们在抽象层次体系中考虑这个谓词下的所有谓词,看看哪一个能够更准确地匹配所需的意义。在这个例子中,“Father”是一个合适的选择,因为它不但蕴涵了“MaleHuman”,还蕴涵这个意义的另一方面,即 $\exists c. Parent(p, c)$ 。

设计知识表示方法的技巧是选择谓词的集合,以便能够使层次关系更有效。通常,最好的表示方法对应于对语言的可行的概括,这种表示同时有助于使用知识库中的表达式来进行句子的解释和生成。

13.3 框架:表示固定模式的信息

自然语言理解所需要的很多推理技术常常需要对当前讨论的对象和情境的一般情况做一些假设。这些信息常常表示为框架结构。用最抽象的形式来说,框架就是描述一些典型的对象或情境的一组事实或对象,以及对情境进行推理的特定的推理策略。这里,表示的情境包括可见的场景、复杂物理对象的结构以及执行某一特定行为的典型方法。基于框架的系统通常提供一些常见的推理机制,例如默认推理、通过层次结构的自动属性继承以及附属程序。在一些系统中,所有推理都是通过与框架有关的专门推理过程来实现的;而另一些系统中,框架主要是描述性的,通过一种更统一的推理过程来解释。无论哪种方法,关键的理念是通过信息的聚类来刻画常见的对象和情境的属性。

可以给框架中的一些主要对象赋予名字,称之为槽或角色。例如,房子的框架可能包括这些槽:“kitchen”(厨房)、“living room”(起居室)、“hallway”(走廊)、“front door”(前门),等等。同时,框架还指定了这些槽和此框架所表示的对象之间的关系。例如,房子框架的“kitchen”槽在物理上位于房子内部,还包括一些做饭的用具等。我们可以将每一个槽看做一个函数,其输入为由这个框架所描述的一个对象,输出为合适的槽值。这样,房子框架的一个实例 H1 包括一个厨房的实例(H1 的“kitchen”槽)和所有其他槽的实例。

下面这个例子给出了个人计算机框架的定义:

Define Object Class *PC*(*e*):
Roles: *Keyb*, *Disk1*, *MainBox*
Constraints: *Keyboard*(*Keyb*), *DiskDrive*(*Disk1*), *CPU*(*MainBox*)

这个结构说明 PC 类型的所有对象都具有类型为“keyboard”(键盘)、“disk drive”(硬盘驱动器)和 CPU 的槽,这三个槽可以分别用函数 Keyb, Disk1 和 MainBox 来表示。这与很多语义网络系统中的表示方法相同。实际上,也可以很容易地将这一结构用语义网络的符号来表示,如图 13.2 所示。

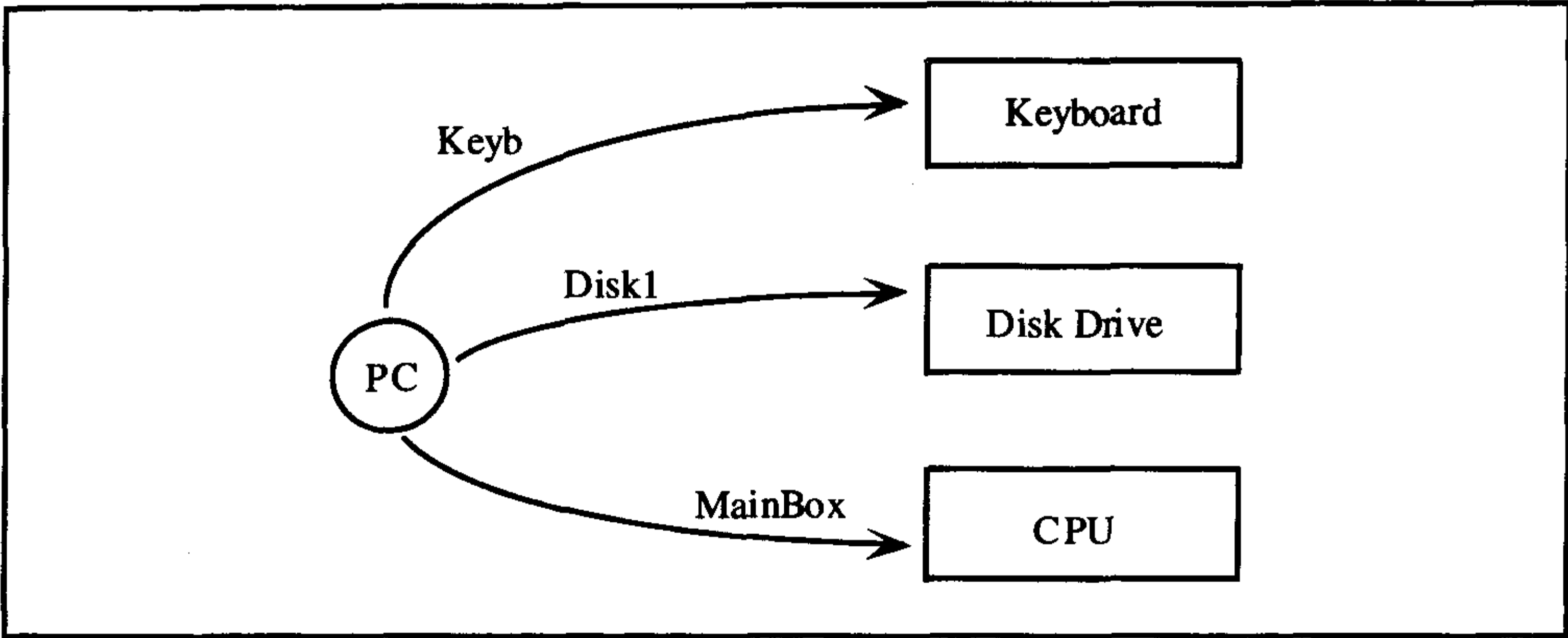


图 13.2 定义 PC 槽的一个语义网络

框 13.2 概念依存:一种基于原语的表达方法

早期的几个有影响的语义表示都是基于小规模的原语集合,利用它们来支持一些推理技术。最有影响的表示方法是 Schank 提出的概念依存理论(Schank, 1975; Schank 和 Riesbeck, 1981)。这一表示方法主要集中于行为动词,并定义了行为的类型。具体来说,主要的行为类型包括下面的三种转变:

ATRANS——抽象的转变(如所有者的变化)

PTRANS——物理的转变

MTRANS——心理的转变(如在说话中)

还有一些基于身体活动原语为:

PROPEL(进行强制)

MOVE(移动身体的一个部位)

GRASP

INGEST

EXPEL

以及心理的行为:

CONC(构思或思考)

MBUILD(实施推理)

基于这些原语、框架角色的集合以及一些因果联系,就能够进行语义表示。早期的研究认为,这种表示方法足以描述所有行为动词的语义。但是,在后面的研究(如 15.5 节)中,这些原语仅作为基本单位用于构造更复杂的结构来刻画动词的语义。研究发现,推理技术只能建立在这种更复杂的结构的基础上,而不是建立在原语的基础上。这样,基于原语的语义表示方法不存在任何优势。

假定 PC 的一个实例 PC3 具有的子部分为: KEYS13, DD11 和 CPU00023,用框架可以表示为:

(PC3 isa PC with Keyb = KEY13, Disk1 = DD11, MainBox = CPU00023))

这一定义可以看做一阶谓词演算公式 $PC(PC3) \ \& \ Keyb(PC3) = KEY13 \ \& \ Disk1(PC3) = DD11 \ \& \ MainBox(PC3) = CPU00023$ 的简写形式。

13.3.1 带约束的槽

一般来说,要描述一个 PC,我们不仅仅需要 PC 的结构模块这些简单的知识。例如,PC 框架可能包括这些槽值之间是怎样相互联系的更多信息。例如,我们可能想断言每一个槽是一个子部分,并说明这些子部分是怎样联系起来的:“keyboard”和“disk drive”是怎样与 CPU 通过合适的设备连接起来的。要达到这一点,需要定义 CPU 为框架结构,其槽包括“KeyboardPlug”,“DiskPort”和“PowerPlug”,等等。下面的例子扩充了这些符号,重新定义了 PC 类。这样,

“keyboard”和“disk”都作为一个模块与 CPU 连接起来。

Define Object Class *PC*(*p*):

Roles: *Keyb*, *Disk1*, *MainBox*

Constraints: *Keyboard*(*Keyb*) & *PART-OF*(*Keyb*, *p*) &
CONNECTED-TO(*Keyb*, *KeyboardPlug*(*MainBox*)) &
DiskDrive(*Disk1*) & *PART-OF*(*Disk1*, *p*) &
CONNECTED-TO(*Disk1*, *DiskPort*(*MainBox*)) &
CPU(*MainBox*) & *PART-OF*(*MainBox*, *p*)

有了这个定义, *PC* 的实例 *PC4* 的槽的数值为 *KEY14*, *DD12* 和 *CPU07*, 可以表示为:

(*PC4* isa *PC* with *Keyb* = *KEY14*, *Disk1* = *DD12*, *MainBox* = *CPU07*)

它包括下面的这些信息:

PC(*PC4*) & *Keyb*(*PC4*) = *KEY14* & *PART-OF*(*KEY14*, *PC4*) &
CONNECTED-TO(*KEY14*, *KeyboardPlug*(*CPU07*)) &
Disk1(*PC4*) = *DD12* & *PART-OF*(*DD12*, *PC4*) &
CONNECTED-TO(*DD12*, *DiskPort*(*CPU07*)) &
MainBox(*PC4*) = *CPU07* & *PART-OF*(*CPU07*, *PC4*)

因为框架是表示某一类对象知识的方法, 所以很自然地, 框架信息应该通过层次体系来继承。例如, 如果定义 *PC* 的子类“*PC-With-Second-Disk*”, 那么这个类应该继承 *PC* 的所有槽。如果为子类“*PC-With-Second Disk*”定义一个新槽“*Disk2*”, 那么所有的实例将有四个槽: “*Keyb*”, “*Disk1*”, “*MainBox*”和“*Disk2*”。

需要注意的是, 一个框架中的信息应该可以看做默认条件。例如, 可能有一个 *PC* (我们将这个 *PC* 标记为 *PC5*) 的“*keyboard*”与主机没有连接起来。违背了这个属性的这一事实不能说明 *PC5* 就不是 *PC*, 而只是说明它不是一个典型的 *PC*。

基于框架的表示方法除了刻画其子部分外, 还可以用于刻画情境的更多信息。自然语言理解中的很有用的例子之一是行为的表示。在后面的章节中可以看到, 在那些出现行为的情境中, 情景的知识对解释语言是非常重要的。特别是关于因果关系(一个行为有什么样的结果以及行为的发生要什么样的条件)的知识是很重要的。还可以扩展这个框架以便允许框架的实例能够与其他命题或事件建立联系。对行为来说, 下面的关系是有用的:

前提——使行为发生的典型属性

结果——由行为引起的典型属性

分解步骤——执行行为的典型方法(一般用一系列的子行为来定义)

例如, 图 13.3 给出了买东西这一行为的定义。这一行为涉及四个对象, 分别是买方、卖方、物品与物品价格对应的钱。另外, 定义还说明只有买方有足够的钱、卖方有东西要卖(前提)的条件下, 买东西这一行为才能够发生。最终买方拥有物品、卖方有了钱(结果)。买东西这一行为可以分解为两个子行为, 即买方给卖方钱, 卖方给买方物品(分解步骤)。虽然这都是看似很平常的日常信息, 但是这些知识对于理解句子中所描述的行为和状态之间的联系是非常关键的, 对歧义的消解也很关键。

The Action Class **BUY(b)**:
Roles: *Buyer, Seller, Object, Money*
Constraints: *Human(Buyer), SalesAgent(Seller), IsObject(Object)*
Value(Money, Price(Object))
Preconditions: *OWNS(Buyer, Money)*
OWNS(Seller, Object)
Effects: \neg *OWNS(Buyer, Money)*
 \neg *OWNS(Seller, Object)*
OWNS(Buyer, Object)
OWNS(Seller, Money)
Decomposition: *GIVE(Buyer, Seller, Money)*
GIVE(Seller, Buyer, Object)

图 13.3 BUY 的定义和步骤

13.4 处理自然语言中的量词

一旦定义了基本的知识表示语言(KRL),就可以考虑将逻辑形式语言映射为 KRL。这两者之间的一个明显区别是量词的处理。逻辑形式包括一些与英语中的量词对应的量化成分,而 KRL 只允许全称量词和存在量词。乍一看,这两者之间的差异似乎是不可协调的,但是通过扩展 KRL 的本体以允许集合作为对象,可以明显减少两者之间的差异。

集合是一些对象的汇集,其中每一个对象可以看做一个单元。一般意义上的集合包括有限集合(如由 *John* 和 *Mary* 构成的集合)和无限集合(如由大于 7 的所有数字构成的集合),我们在知识表示语言只涉及有限集合。一个集合可以用大括号中的成员列表来表示。例如, $\{John1, Mary1\}$ 指由 *John1* 所指对象和 *Mary1* 所指对象构成的集合。集合中的元素是没有顺序的,即 $\{John1, Mary1\} = \{Mary1, John1\}$ 。我们也允许用常数来表示集合,这样, *S1* 就表示由公式 $S1 = \{John1, Mary1\}$ 定义的集合。完全集合理论还允许集合作为其他集合的元素。在知识表示语言中,我们将不用这样的集合。集合一般用一些属性来表示,形式如 $\{y \mid P_y\}$,表示满足表达式 P_y 的所有对象的集合。例如,男人的集合为 $\{y \mid Man(y)\}$ 。另外,我们引入下面的谓词建立集合之间以及集合和个体之间的关系:

$S1 \subset S2$ 当且仅当 $S1$ 中所有的元素也在 $S2$ 中

$x \in S$ 当且仅当 x 是 S 的一个元素

在知识表示语言中使用类似集合的对象,我们可以对句子“Some men met at three”生成下面的解释:

$\exists M: M \subset \{x \mid Man(x)\} . Meet1(M, 3PM)$

即存在男人集合的一个子集 M ,他们三点钟见面。根据惯例,我们用大写字母表示集合中的所有变量。原则上,集合可以出现在任何一个个体允许出现的地方。不过实际上,一些特定的动词在特定参数位置上只与集合或只与个体搭配。例如,动词“meet”的施事者应该是至少包括两个元素的集合,单个对象的见面是没有意义的。另一些动词与个体搭配,而与集合不搭配;还有一些动词允许与集合和个体搭配。

让我们分析对应于集合的整体性解读和个体性解读的不同公式。句子“Some men bought a suit”有两个不同的解读,其逻辑表达式如下(不考虑时态运算符):

$$\begin{aligned} &(\text{SOME } m1 : (\text{PLUR } \text{MAN1}) \\ &(\text{A } s1 : \text{SUIT1} \\ &(\text{BUY1 } m1 s1))) \end{aligned}$$

整体性解读可以映射为:

$$\exists M1 : M1 \subset \{z \mid \text{Man}(z)\} \exists s : \text{Suit}(s) . \text{Buy1}(M1, s)$$

也就是说,存在一起买一套衣服的男人集合的一个子集。个体性解读为一些男人中的每个人都买了一套衣服,表示为:

$$\begin{aligned} &\exists M2 : M2 \subset \{z \mid \text{Man}(z)\} \forall m : m \in M2 \\ &\exists s : \text{Suit}(s) . \text{Buy1}(m, s) \end{aligned}$$

需要注意的是,整体性解读和个体性解读均与一个共同的核心意义相关,即男人的子集。唯一的差异是将这个集合当做一个单位还是表示集合中的每一个元素。

基于集合的表示方式也可以用来确保不止一个人买了一套衣服。为此,我们引入一个新的函数返回这个集合的势。对任何集合 S , $|S|$ 表示集合 S 中的元素个数。用算术运算符,可以对集合中元素个数的约束进行表示。例如,“Three men entered the room”(三个人进入了这个房间)的意义如下(仍然忽略了时态信息):

$$\begin{aligned} &\exists M : (M \subset \{y \mid \text{Man}(y)\} \ \& \ |M| = 3) \\ &\forall m : m \in M . \text{Enter1}(m, \text{Room1}) \end{aligned}$$

通过改变这个约束为 $|M| \geq 3$,可以得到“At least three men entered the room”(至少三个人都进入了这个房间)的意义。

一些含义模糊的量词也可以用集合赋予一个近似的意义。例如,定义“most”为集合中超过一半的元素个数,那么“Most men laughed”(大部分人都笑了)的意义如下:

$$\begin{aligned} &\exists M : (M \subset \{y \mid \text{Man}(y)\} \ \& \ |M| \geq \frac{|\{y \mid \text{Man}(y)\}|}{2}) \\ &\forall m : m \in M . \text{Laughed}(m) \end{aligned}$$

在一个实际的篇章中,量词项的解释往往与前面定义的某个集合有关。例如,句子“Most men laughed”指上述集合中的大部分人,而不是指世界上的大部分人。换言之,这个句子不是说所有男人中的大部分都笑了,而是指在某个特定上下文(比如,一个给定的房间)中的大部分人都笑了。我们将在第 14 章中继续讨论这个问题。

在知识表示中引入集合作为明确对象后,我们就能够以一种直觉上令人满意的方法来描述一些定量化的结构。这里讨论的是对一阶谓词演算的扩展,实际上,在其他表示方法中也需要有类似的描述能力来刻画相同的现象。例如,在语义网络表示方法中,要有节点能够表示集合,能够表示这些节点上的集合中数量的约束,还能够对这些集合进行量化以得到个体性的解释。

◦ 13.5 时间和动词的体态类

支持自然语言的任何知识表示系统的一个中心模块是对动词和时间的处理。很多语言涉

及时间,包括隐含在时态中的时间信息、句子的体态以及由一些时间副词所表示的明显的时间信息(如“for five minutes”,“yesterday”,“at 3 o'clock”,“after they had left”)等。

在逻辑形式语言中,有好几种方式可以处理时间信息。一些模态运算符可以表示时态(如“PAST”,“PRES”,“PROG”,“FUT”)和时间连词(如“BEFORE”,“DURING”),所有谓词都可以用时间作为其参数。要处理这些语言现象,需要进一步扩充一阶谓词演算以便能够表示时间。

时间有不同的类型,例如时间点和时间段等。时间点表示的是瞬间时间,往往与世界中的变化相联系(如灯被打开了或某人找回了丢失的钢笔);时间段表示事件发生的一段时间,并且有一定的持续期(如五分钟的时间),而时间点则没有。很多谓词只能定义在时间段上,例如,考虑某谓词断言“John drove his car to work at a certain time”(John 在一个特定的时间开车去上班)。这个谓词只有在一个时间段上才为真,而不是时间点,因为开车到一个地方需要一段时间。

因为在时间点和时间段之间有不同的关系,所以需要对这些两种情况进行区分。例如,两个时间段可能有重叠,而时间点则不能重叠。另外,两个时间段可能相接,一个结束而另一个开始。一个时间点或时间段也可能在另一个时间段内部,而一个时间点内则不能包括任何时间。要表达时间关系,需要下面的谓词:

$t_1 < t_2$	时间点/时间段 t_1 在时间点/时间段 t_2 的前面
$t_1 : t_2$	时间段 t_1 紧挨着时间段 t_2 , 或者时间点 t_1 是时间段 t_2 的开始, 或者时间点 t_2 是时间段 t_1 的结束
$t_1 \subseteq t_2$	时间点/时间段 t_1 包含在时间段 t_2 中

如前所述,一些谓词只能在时间段上为真,一些谓词只能在时间点上为真,也有一些谓词在时间点和时间段上都为真。谓词的分类对应于动词短语的不同体态类。

句子所描述的命题至少分为 3 类,包括描述状态(称为状态性命题),描述正在进行的活动(称为活动性命题),定义已经完成的事件(称为完成性命题)。状态性命题描述世界的一个瞬间或时间无限延长的某个属性(如下面的句子):

Jack is happy. (Jack 很高兴。)

I believe the world is flat. (我相信世界是平的。)

状态性命题描述的情形缺少一个准确定义的结束点,因此不能出现在一些特定的语言上下文中。例如,不能出现在正在进行的形式中:

* Jack is being happy.

* I am believing that the world is flat.

活动性命题描述出现在一个时间段内的活动。活动经常用进行时来表达,如:

Jack is running. (Jack 在跑。)

The door was swinging to and fro. (这扇门在来回扇动。)

描述状态和活动的句子一般不允许时间性修饰成分(如“in five minutes”),但是可以允许持续性修饰成分(如“for five minutes”)。

完成性的句子描述那些已经结束的事件,如:

Jack fell asleep. (Jack 睡着了。)

Jack climbed the mountain. (Jack 爬上了山。)

这两个句子中,事件在某一个时间点(称为“终点”)结束。同时,在这个终点,某一个结果属性开始。例如,第一个句子中,事件结束后 Jack 处于睡着了的状态;第二个句子中,事件结束后 Jack 位于山顶。

描述完成性命题的句子可以包含时间性的修饰成分(如“in an hour”),例如下面两句:

They climbed the mountain in two days. (他们在两天内攀登到了山顶。)

Jack fell asleep in an hour. (Jack 在一个小时内睡着了。)

完成事件可以细分为两个子类,一类描述状态的变换(称为达成类, achievement class),另一类涉及到达一个终点的活动(称为成就类, accomplishment class)。例如,前面的例子描述成就类的情况,而下面的例子则描述达成类的情况:

Jack recognized the man. (Jack 认出了这个人。)

Helen woke up. (Helen 醒了。)

根据不同的时间性参数,命题可以分为四类。具体来说,状态性命题可以在一个时间点或时间段上为真。例如,可以说在一个时间点或一个长期的时间段内一个球是红的。状态性命题是同质的,即一旦这个命题在一个时间段内为真,那么在此时间段的子时间段内也为真。

达成性的句子对应于描述状态转移的命题,如句子“Jack reached the summit”(Jack 到达了山顶)或者“Helen closed the door”(Helen 关上了这扇门)。第一个命题描述 Jack 到达山顶后状态的转换,第二个命题描述门被关闭后状态的转换。这两个命题都不能描述一个时间段,但是它们的定义可能包括结果状态的信息,如:

$$\forall a1, l1, t1. Reach(a1, l1, t1) \supset \exists T1. t1 : T1 \& At(a1, l1, T1)$$

即,如果一个行动主体 $a1$ 在时间点 $t1$ 到达地点 $l1$,那么 $a1$ 位于 $l1$ 将有一段时间,这段时间的起点是 $t1$ 。

描述过程的命题与活动型动词对应,如“Jack ran”。过程谓词只能出现在一个时间段内,往往是同质的。但这里的同质不如状态性命题中的严格,如“Jack was running between 2 and 3 o'clock”(2点到3点之间 Jack 在跑步),即使他在中间停顿了5分钟,这个描述也是没有问题的。这样,就存在可推翻的隐含,但不是蕴涵,即如果过程 P 在时间段 $T1$ 出现,那么它也可能在时间段 $T1$ 内的某一个时间段 $T2$ 内出现。

成就型的句子往往具有将几个不同的形式结合起来的一种更复杂的结构,如“Jack ran to the store”(Jack 跑到了商店)。我们处理复杂结构的方法是将逻辑形式谓词映射为知识表示语言中的复杂命题。具体来说,“Jack ran to the store”(Jack 跑到这个商店)的逻辑表达式可以映射为以下公式,表明跑步这个过程达到的终点状态为“在商店”,即:

$$\exists T1, T2. Running(Jack1, T1) \& At(Jack1, Store1, T2) \& T1 : T2$$

图 13.4 总结了体态类的一些相互区别的属性。

体态类	是否可以在一个时间点为真?	是否可以在一个时间段为真?	时间性修饰 <i>in</i>
状态短语	YES	YES	NO
活动	NO	YES	NO
达成	YES	NO	YES
成就型	NO	YES	YES

图 13.4 体态类的不同属性

13.5.1 时态的表示

时态运算符可以直接用时态逻辑表达式来表示,不需要模态运算符。基本理念是将时态运算符映射为一些时间性关系,这种关系涉及到一些以当前时间为参照的指示性区间。对于下面的例子,我们假定 NOW1 表示当前时间。这样,可以将 PAST 运算符映射为一个公式,对在这之间的时间进行量化。这样,句子“John was happy”将对应为知识表达式:

$$\exists T1. T1 < NOW1. Happy(Jack1, T1)$$

同一个句子如果用简单现在时来表达,“John is happy”将映射为:

$$\exists T1. NOW1 \subseteq T1. Happy(John1, NOW1)$$

简单将来时的句子“John will be happy”将映射为:

$$\exists T1. T1 > NOW1. Happy(Jack1, T1)$$

值得注意的是,时态的解释是有歧义的,因为一些简单现在时的句子可以指将来,如“The flight arrives at noon”。而一些将来时的句子也可以表示现在,如“Jack will be in class by now”。这里将不考虑这些问题。

即使没有歧义,也存在一些困难的问题。例如,有两种方法来表明过去的某一个事件为真,分别对应于过去时和过去完成时。例如:

Helen saw the books.(Helen 看见了这些书。)

Helen had seen the books.(Helen 已经看见过这些书了。)

这两种解释之间的区别何在? 如果是孤立的句子,很难说明它们之间的区别。不过,可以将它们放到具体的上下文之中,如:

When Jack opened the door, Helen saw the books.

(当 Jack 打开门时, Helen 看见了这些书。)

When Jack opened the door, Helen had seen the books.

(当 Jack 打开门时, Helen 已经看见过这些书了。)

在第一个句子中,“seeing”行为的发生是在 Jack 开门的同时或刚打开门以后;而第二个句子中的“seeing”行为要早于 Jack 开门的时间。Reichenbach(1947)首先提出了这一不同,并且提出了参照时间的概念。在这两个例子中,主要子句的参照时间是 Jack 开门的时间。简单过去时中事件的发生时间与参照时间相同;而在过去完成时中,事件的发生时间要早于参照时间。具体来说,Reichenbach 提出了一个关于时态的理论,这个理论给出了三类时间的信息:

- S——说话时间
- E——事件/状态的发生时间
- R——参照时间

参照时间可以借助于前面描述的时间性副词来确定,也可以由篇章上下文来确定(在第 15 章讨论这个问题)。对于简单过去时,参照时间与事件时间相同,即 $E = R$ 。这三种形式是由 R 和 S 之间关系的变化产生的:

- Jack sings 简单现在时: $S = R, E = R$
- Jack sang 简单过去时: $R < S, E = R$
- Jack will sing 简单将来时: $S < R, E = R$

另一方面,完成时的事件时间要早于参照时间。与前面一样,是根据参照时间和说话时间如何联系来区分的。这样,有下面的结果:

- Jack has sung 现在完成时: $S = R, E < R$
- Jack had sung 过去完成时: $R < S, E < R$
- Jack will have sung 将来完成时: $S < R, E < R$

这一分析也能够解释后事时态,其中 $R < E$:

- Jack is going to sing 后事现在时: $S = R, R < E$
- Jack was going to sing 后事过去时: $R < S, R < E$
- Jack will be going to sing 后事将来时: $S < R, R < E$

这三个时间之间的顺序关系如图 13.5 所示。

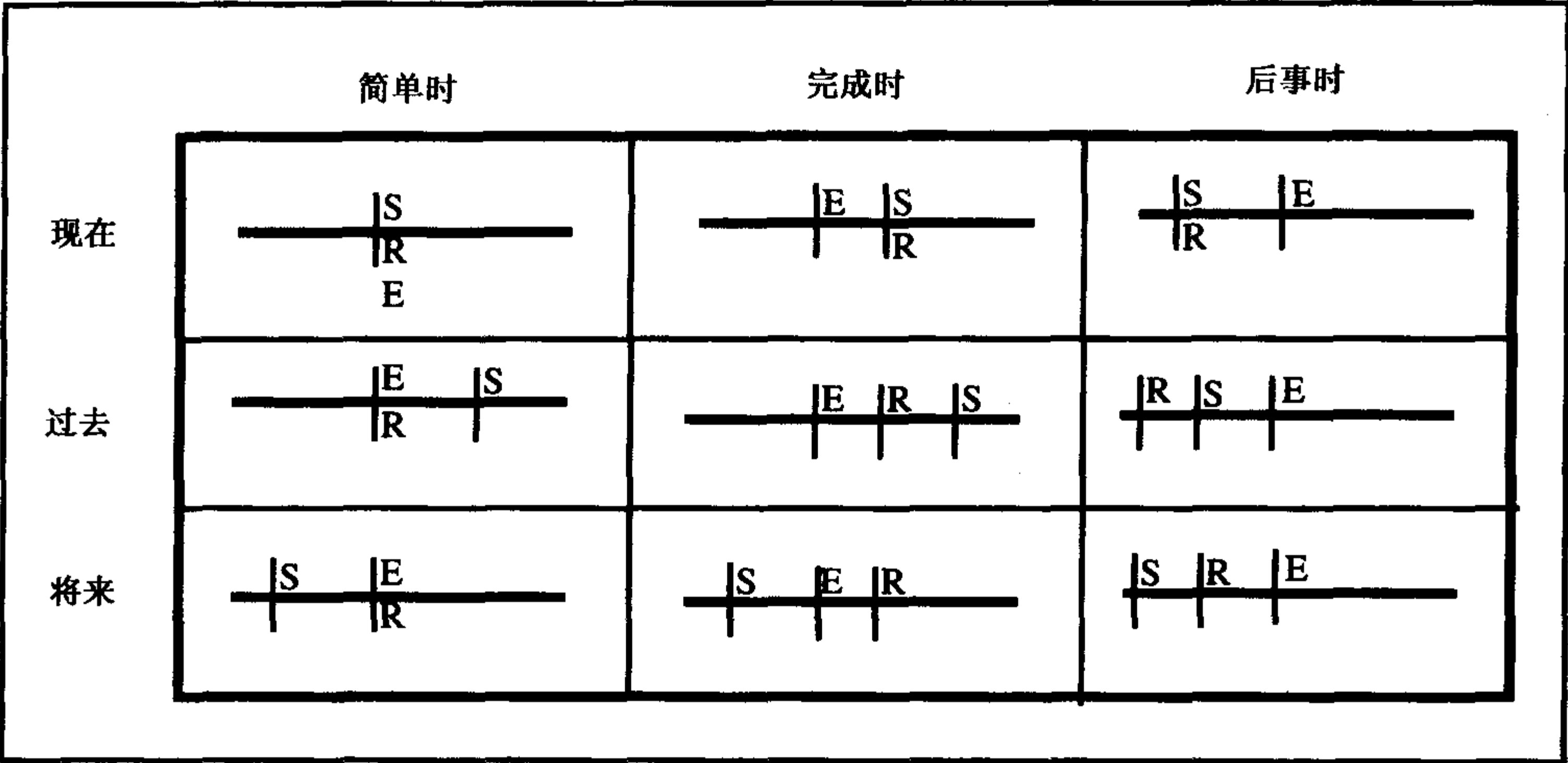


图 13.5 用时态表示的时间结构

◦ 13.6 基于逻辑的表示方法中的自动演绎

我们在前面几节介绍了抽象的知识表示语言,本书的后面部分将继续使用这种语言。下

面几节讨论的重点是知识表示系统中的推理策略。本节介绍知识表示中的几种自动推理技术。如前所述,推理系统可以分为两大类:(1)基于演绎证明技术的描述性方法;(2)程序性方法。本节考察一些纯粹的推理技术。

如果你以前研究过一些证明方法(如一阶谓词演算中的自然演绎),就会知道证明方法的过程非常复杂,这是因为对任何给定公式都有很多不同的方法可以证明。此外,有很多句法手段表示同样的内容。大多数自动推理的研究工作在开始研究这个问题前都尝试降低这种复杂性。更具体地说,我们通常用一种范式来表示符合完全一阶谓词演算句法的一个受限子集的公式。很多逻辑等价而句法结构不同的公式都有相同的范式。例如,以下公式:

$$\begin{aligned} &\neg P \& Q \\ &\neg(P \vee \neg Q) \\ &\neg(Q \supset P) \end{aligned}$$

都是逻辑等价的。在本节将讲到的基于合取范式(CNF, conjunctive normal form)的表达式中,这些公式都映射到同一个公式。第二种技术是对变量的处理。一般来说,对于量化的变量,这些变量的很多特定实例都可用于构造一个证明。这样,在搜索一个证明的过程中,很可能选中错误的实例,导致以后需要推倒重来。合一技术通过对给定的一项推理计算最可能的通用答案,从而部分地避免了这个问题。如果读者对合一技术还不太熟悉,请参考附录 A 和附录 B。

从最简单的对数据库基于模式的检索,到类似于 PROLOG 的霍恩(Horn)子句系统,再到通用的定理证明系统,从系统的总体框架角度看,很多推理系统与自动证明系统类似。其间的差异来源于每个系统能表示和推理的表达式的形式。为了更清楚地了解这一点,让我们先为一阶谓词演算构造一个完整的通用范式。在常数、函数和原子命题这一层次,合取范式与一阶谓词演算相同。在合取范式中,一个文字命题对应于一个原子命题,或者原子命题的否定形式,如下所示:

Person(John1)
Car(Carl)
Owns(John1, Carl)
¬Happy(John1)

一个子句就是如下所示的文字命题的析取,断言 Helen 要么拥有一部轿车,要么就不高兴:

Owns(Helen1, Car2) ∨ ¬Happy(Helen1)

在很多系统中,子句用隐含运算符的形式来表示,前面的子句可以记为:

(Owns(Helen1, Car2) <- Happy(Helen1))

一般地,一个子句记为:

(P1, ..., Pn <- Q1, ..., Qm)

它的意思是如果 $Q1, \dots, Qm$ 为真,那么 $P1, \dots, Pn$ 中至少有一个为真。如果将一个子句视为析取公式,那么这些 Qi 都是否定文字命题,而 Pi 都是肯定文字命题。我们可以证明一阶谓词演算中所有的公式都有一个等价的子句形式。图 13.6 给出了用标准逻辑运算符来表示的一些命题公式的例子,及其在合取范式中的等价形式。量化在基于子句的系统中用变量和 Skolem 化处理,这部分内容我们已经在 13.2 节中讨论过。

一阶谓词演算中的公式	等价的子句形式
P	$(P \leftarrow)$
$\neg P$	$(\leftarrow P)$
$P \& Q$	两个子句: $(P \leftarrow)$ 和 $(Q \leftarrow)$
$P \vee Q$	$(P \vee Q \leftarrow)$
$Q \vee \neg P$	$(Q \leftarrow P)$
$P \supset Q$	$(Q \leftarrow P)$
$\neg(P \supset Q)$	两个子句: $(P \leftarrow)$ 和 $(\leftarrow Q)$
$(P \& Q) \vee R$	两个子句: $(P \vee R \leftarrow)$ 和 $(Q \vee R \leftarrow)$

图 13.6 子句形式的公式

这种表达式的一个很有用的约束是霍恩(Horn)子句,其左侧只有一个文字命题,也就是说,只有一个肯定文字命题。在一个受霍恩子句约束的知识库中,PROLOG 中使用的回溯链策略能够为任何从知识库中逻辑推导出来的公式找到证明。

用子句来推理在很大程度上利用了合一算法。首先,分析一个很有限的知识表示,其知识库中只有文字命题,这与只允许量化表达的关系数据库很相似。我们的直觉是,如果从知识库中能够找到与文字命题 P 合一的一个公式,那么这个文字命题 P 就能从这个知识库中推导出来。让我们分析在这样一个系统中是如何处理变量的。一阶谓词演算公式 $\forall y \exists x P(x, y)$ 对应于文字命题 $P(Sk2(?y), ?y)$ 。如果知识库包含这个文字命题,想像一下如果以后我们想知道 $\forall w \exists z P(w, z)$ 是否从知识库中推导出来,将会发生什么事。如果将其转化为子句形式,会得到一个像这样的文字命题 $P(Sk3(?w), ?w)$ 。该命题和数据库中的文字命题不能合一,因为这里 Skolem 函数使用了不同的名字。在模式检索系统中,如果这个文字命题用做查询,则可以通过改变量词的解释来处理这种情况。这样,作为一个查询, $\forall y \exists x P(x, y)$ 就可以映射到一个文字命题 $P(?z, Sk4)$,它可以和已经存在于知识库中的同一个公式的文字命题合一。

乍看上去,这种改变查询中量词解释的技术似乎是随心所欲的,但实际上是沿用了以前用过的基本证明策略。更具体地说,总是有两种通用的方法来证明公式 P 。第一种是用推理规则直接从知识库中来证明 P ,第二种是证明 P 和 KB 不一致(因而 P 必然能从 KB 中推导出来)。后一种方法称为反证法,这种方法被认为是自动演绎中最有用的技术。它构成了前面所讲的模式匹配技术、霍恩子句证明法和基于消解的通用定理证明系统的基础。所有这些都看做一条称为消解规则的推理规则的特定实现。一个比较简单的消解规则的例子和假言推理比较相似。更具体地说,给定一个子句:

$(Q \leftarrow P)$ (即, P 蕴涵 Q)

和子句:

$(P \leftarrow)$ (即, P 为真)

根据这个消解规则,可得到 $(Q \leftarrow)$ (也就是说, Q 为真)。消解规则的另一种简单应用是检查冲突。已知 $(P \leftarrow)$ (即 P 为真)和 $(\leftarrow P)$ (即 P 为假),消解规则给出一个空子句 (\leftarrow) ,这表示知识库不一致。

通过利用合一法对两个子句中的变量进行实例化使得这些 x 相同,从而将消解规则推广为一阶谓词逻辑形式。例如,分析一个知识库,其中包含若干子句,这些子句断言所有的狗都会叫以及 Fido 是条狗:

$(Bark(?x) \leftarrow Dog(?x))$
 $(Dog(Fido1) \leftarrow)$

根据消解规则,可推断 Fido 会叫。用 $Fido1$ 替换两个子句中的 $?x$,可以从两个子句中消掉文字命题 $Dog(Fido1)$,从而得到结果子句:

$(Bark(Fido1) \leftarrow)$

借助消解规则,我们可以再次研究反证法。已知一个用子句形式表示的前后一致的知识库,通过对 P 取反,将其转化为子句形式并添加到知识库中。然后,看看是否能用消解规则推导出空子句,从而可以判断是否能从知识库中推出公式 P 。因为这表示知识库现在前后不一致,所以 P 肯定能从知识库中推导出来。可以证明,如果确实能从知识库中推出 P ,那么可以找到证明,在这个意义上消解策略是完备的。但是,其逆命题在一般情况下却并不成立。如果不能从知识库中推出 P ,那么这种证明方法不可能证明这个事实。通过限制可用的子句形式,可以得到不同的性质。例如,借助一个只包含霍恩子句的知识库,我们可以在任何情况下判断公式 P 是否是从知识库中推出的。

在基于演绎的系统中,一种常见的引入默认机制的技术称为失败证明 (proof by failure)。我们引入一种新的称为 UNLESS 的运算符,对于作为其参数的公式,这个运算符递归地调用定理证明程序。如果对定理证明程序的递归调用结束时没能证明公式为真,那么 UNLESS 公式为真。例如,“cats purr”(猫会咪咪叫)的默认规则可以表示成如下公理:

$\forall c. Cat(c) \& Unless(\neg Purr(c)) \supset Purr(c)$

也就是说,除非能明确证明猫不会咪咪叫,否则可以得出结论:猫会咪咪叫。因为递归调用证明程序的潜在开销很大,因此这种技术通常只用于受限证明系统中,例如在 PROLOG 类型的霍恩子句表达中。在很多与封闭世界假设密切相关的系统中,另一种常用的技术是失败否定 (negation as failure) 规则。在这个规则中,如果命题不能证明为真,它就为假;也就是说,对于任何一个命题 P , $Unless(P) \supset \neg P$ 。当然,如果在一个系统中失败否定规则作为默认规则,我们必须很小心。例如,在一个使用失败否定的系统中,前面的默认规则表示,只有能断定猫咪咪叫的时候才能断定猫咪咪叫——这显然不是一条有用的规则。

子句、合一以及反证法的概念提供了几乎所有现代知识表示系统的形式基础;也就是说,任何与变量进行模式匹配的系统都可以看成通用技术的一个特例。当然,这并不意味着匹配就能涵盖知识表示系统做的所有推理,但它的确是每个系统的关键部分。

。13.7 程序语义学与问答系统

基于程序的技术通常在数据库查询程序中使用。在这种应用中,逻辑形式语言和数据库语言的表达能力有很大的差别。用上一节的形式方法来表示,知识库(即数据库)只包含肯定文字命题,通常不包含变量。在这种情况下,一般做法不是将逻辑形式语言转成本节先前描述的扩展的一阶谓词演算,而是将逻辑形式当成用查询语言构造的表达式来处理。每个逻辑

形式语言结构对应一个执行适当查询的特定过程。例如,查询“Does every flight to Chicago serve breakfast?”(是否每一趟飞往 Chicago 的航班都提供早餐?)的逻辑形式语言为:

(EVERY f1 : (& (FLIGHT f1) (DEST f1 (NAME c1 “Chicago”)))
(SERVE-BREAKFAST f1))

可以解释为如下过程:

1. 在数据库找出所有目的地为 CHI(数据库中表示 Chicago 的符号)的航班。
2. 对找到的每个航班,检查是否提供早餐。如果都提供,返回是;否则返回不是。

本节说明了如何将逻辑形式的表达式解释为过程的方法,这种方法常称为程序语义学(procedural semantic)。

为了更清楚地了解这个过程,我们先分析图 13.7 所示的简单数据库查询系统。这个数据库由一组不含变量的肯定文字命题组成。时间用国际表示法来表示;比如 1700HR 表示下午 5:00。关系(ATIME f c t)表示航班 f 在时刻 t 到达机场 c,而(DTIME f c t)表示航班 f 在时刻 t 离开机场 c。这个数据库系统提供一个基于文字命题的模式匹配的简单接口,其中,查询中可能包含变量。假定现在有两个数据库查询函数:

(Test < literal >₁, ..., < literal >_n)——如果对于文字命题中出现的变量存在某种约束,以便每个文字命题都能在数据库中找到,则返回真。

(Retrieve < var > < literal >₁, ..., < literal >_n)——和前面的 Test 类似。但如果成功,就返回能提供问题答案的那个指示变量的每个实例。

例如,已知图 13.7 的数据库,查询:

(Retrieve ?x (FLIGHT ?x) (ATIME ?x CHI 1000HR))

将返回列表(F2),因为 F2 是?x 的惟一约束,使得这两个文字命题都在数据库中。

(FLIGHT F1)	(ATIME F2 CHI 1000HR)
(FLIGHT F2)	(ATIME F3 CHI 900HR)
(FLIGHT F3)	(ATIME F4 BOS 1700HR)
(FLIGHT F4)	(DTIME F1 BOS 1600HR)
(AIRPORT BOS)	(DTIME F2 BOS 900HR)
(AIRPORT CHI)	(DTIME F3 BOS 800HR)
(ATIME F1 CHI 1700HR)	(DTIME F4 CHI 1600HR)

图 13.7 一个简单的航班时刻表数据库

用逻辑形式语言表示的所有表达式必须用一种特定的方法来解释,按照这种方法,这些表达式最终都简化为这个数据库的两种查询形式。简化的方法是通过将逻辑形式映射为对数据库执行适当查询的过程。因此,回答一个问题可以通过两个步骤来完成:将逻辑形式转换为一个程序,然后执行该程序,计算得到答案。

首先,分析转换步骤。对于任何逻辑形式表达式 E,在数据库查询语言中 E 的转换将表示成 T(E)。表达式的转换随结构的不同而有所变化。例如,像(NAME c1 “Chicago”)这样的表达

式可以转换成合适的数据库常量,在这种情况下就是 CHI。但除此之外,符号 **c1** 必须和称为符号表的结构中的常量 CHI 一起保存。这样,如果在逻辑形式中的其他部分再发现 **c1**,就可以用值 CHI 来替换它。

某些逻辑形式关系可以直接转换成数据库关系,而其他关系会转换成更复杂的表达式。例如,数据库中没有用到逻辑形式关系 DEST;相反,是将航班的目的地表示成 ATIME 关系,其中既包含航班目的地,也包含到达时间。因此,逻辑形式关系 (DEST **f1**(NAME **c1** "Chicago")) (其中 **f1** 已经和变量 ?f 关联在一起) 被转换成数据库关系:

(ATIME ?f CHI ?t)

因为在 DEST 关系中没有包括时间信息,所以在转换过程中将其解释为一个不受约束的变量。一般来说,逻辑表达式中的每一个关系的转换都必须进行指定。

程序语义学方法在解释逻辑连词和量词的时候更有趣,这两种词在关系数据库中很显然没有对应的结构。逻辑运算符可解释成如下形式:

合取: (& R_1, \dots, R_n)——转换成形式为 (CHECK-ALL-TRUE $T(R_1), \dots, T(R_n)$) 的一个程序。在执行程序时,依次查询每个 $T(R_i)$,确认为真,并将变量的约束传递给后面的查询。如果存在一组变量约束,用其查询每个 $T(R_i)$ 都为真,那么该程序成功;否则失败。

析取: (OR R_1, \dots, R_n)——转换成形式为 (FIND-ONE-TRUE $T(R_1), \dots, T(R_n)$) 的一个程序。在执行时,依次查询每个 $T(R_i)$,直到其中一个 R_i 为真,这样该程序成功;如果没有 R_i 为真,该程序失败。

否定过程对数据库中的所有关系做了一个封闭世界假设,并使用失败证明 (proof by failure):

(NOT R)——可转换成形式为 (UNLESS $T(R)$) 的一个程序。只有查询 $T(R)$ 失败了,该程序才成功。

最复杂的转换常常与量词有关。每个量词都转换成对数据库进行适当操作的程序。由于数据库语言的限制,通常只支持复数量词的个体性解读。这里,分析问答系统中比较重要的三个量词: THE, EACH 和 WH。

(THE $x: R_x P_x$)——转换为程序 (FIND-THE ?x $T(R_{?x}) T(P_{?x})$)。首先,进行一次查询以找到所有满足 $T(R_{?x})$ 的 ?x,也就是说 (Retrieve ?x $T(R_{?x})$)。如果只找到一个答案,就用该答案替换整个表达式中的 ?x,并执行 $T(P_{?x})$ 以提供整个表达式的答案。如果查询 $T(R_{?x})$,但没找到符合条件的对象,则与假设存在冲突,该冲突可由问答系统以一种特殊的方式进行处理,比如,通知用户没有这种对象。如果找到多个答案,系统的设计者必须决定哪一个才是最佳答案。一些系统能处理这种情境,并对每个值执行 $T(P_x)$;而其他系统将这种情况看成失败。

(EACH $x: R_x P_x$)——转换为程序 (ITERATE ?x $T(R_{?x}) T(P_{?x})$),也是先执行一次查询以找出所有满足 $T(R_{?x})$ 的 ?x。然后,对每个找到的值重复执行 $T(P_{?x})$,只有所有的查询都成功,该程序才成功。

(WH $x: R_x P_x$)——转换为程序 (PRINT-ALL ?x $T(R_{?x}) T(P_{?x})$),查找所有满足 R_x 和 P_x 的转换的对象,即 (Retrieve ?x $T(R_{?x}) T(P_{?x})$),然后打印结果。确定打印答案的最佳格式,尤

其是确定是否还要提供附加信息,这是一个很复杂的问题。这里,假定只是简单地打印找到的答案。

这里介绍的方法足以解释一些使用了图 13.7 中的数据库的例子。查询“Which flight to Chicago leaves at 4PM?”的逻辑形式为(经过辖域指定之后):

```
(WH f1 : (& (FLIGHT f1) (DEST f1 (NAME c1 "Chicago")))
  (LEAVE t1 (NAME t1 "4PM")))
```

可以将其转换为如下形式的查询:

```
(PRINT-ALL ?f (FLIGHT ?f) (ATIME ?f CHI ?t) (DTIME ?f ?s 1600HR))
```

这里,如前所述,DEST 关系映射为 ATIME 关系,LEAVE 谓词映射为 DTIME 关系。需要注意的是,这个逻辑形式中没有指定出发地点,所以我们将出发地点当做变量来处理。在真实的应用中,出发城市应根据上下文或者根据默认规则来确定。在图 13.7 所示的小数据库中,只有一个航班符合现有的描述,即 F1,因此它是这个例子的答案。

下面,考虑涉及迭代的更复杂的例子,比如查询“Give the departure time of each flight to Chicago”,其逻辑形式为:

```
(EACH f1 : (& (FLIGHT f1) (DEST f1 (NAME c1 "Chicago")))
  (THE t1 : (DEPART-TIME f1 t1)
    (GIVE-SPECIFY1 g1)))
```

上式可转换成下面的查询:

```
(ITERATE ?f1 (CHECK-ALL-TRUE (FLIGHT ?f1) (ATIME ?f CHI ?t1))
  (FIND-THE ?t1 (DTIME ?f1 ?city ?t1)
    (PRINT ?t1)))
```

在这个例子中,动词“give”的解释仅仅是打印出其参数,例如离开时间。这个表达式的执行步骤如下:

1. ITERATE 步骤的第一部分是找出满足条件的所有?f1。只有(FLIGHT ?f1)和(ATIME ?f CHI ?t1)都在数据库中,CHECK-ALL-TRUE 步骤对?f1 才成功。这一步骤返回航班 F1, F2 和 F3。
2. 这一步的第二部分是对这 3 个值中的每一个都执行(FIND-THE ?t1(DTIME ?f1 ?city ?t1) (PRINT ?t1))。分析对第一个值 F1 的执行过程,表达式为:

```
(FIND-THE ?t1 (DTIME F1 ?city ?t1) (PRINT ?t1))
```

转换 FIND-THE 后的程序首先执行查询(Retrieve ?t1 (DTIME F1 ?city ?t1))。这一步返回惟一的答案,即时间 1700HR。FIND-THE 程序的第二步对这个值执行 PRINT,这样就打印出 1700HR。第二次和第三次迭代分别打印值 1000HR 和 900HR。

很多自然语言数据库查询系统都使用了程序语义学技术。对无法用数据库系统所提供的受限语言来表示其意义的那些结构来说,这种技术提供了一种刻画其合适特性的便利方法。由于数据库应用本身的性质,这些技术的局限性在实际应用中并没有表现出来。例如,在某些信息中,使用量词的整体性解释的那些查询是必要的,而数据库系统通常无法描述这样的信息。另外,由于数据库不能包含析取信息,所以有限的析取查询形式也不构成问题。

程序语义学技术还可以和基于霍恩子句的数据库一同使用。大多数结构的程序性定义都可以用霍恩子句公理进行定义,这样它还多了一种能力,即递归地调用证明程序,以执行查找满足某些文字命题集合的所有对象这样的任务。对其进行扩展使之能处理有限集,这种表示方法就能够借助 13.4 节描述的表示技术用程序来处理很多的量词。由于具有额外的表达能力,作为一种支持自然语言查询的系统,基于霍恩子句的数据库是传统关系数据库的一种很好的泛化形式。

框 13.3 LUNAR:一个自然语言数据库查询系统

在讨论程序语义学时,我们已看到 LUNAR 系统的核心模块的大部分。LUNAR 开发于 20 世纪 70 年代。作为一个数据库的前端查询系统,该数据库包含阿波罗月球之旅带回来的岩石样本。它是第一个证实了在真实应用领域中具有广泛覆盖度的自然语言系统。目前,这个领域中的很多通用技术都源于该系统,而且有些技术在该系统中被提升到高级阶段。该系统利用一个 ATN 分析器(参见 4.6 节)来产生基于语法关系的表示,语义解释模块随后用递归的模式匹配技术生成采用语义表示形式的表达式,如 11.1 节所述。在语义表达式的其余部分中,量词信息被单独处理,然后用类似于 12.3 节所述的启发式方法进行排序。作为结果的最终语义表达式是用一种语义表示语言来表示的,这种语言与我们使用的完全指定辖域的逻辑形式类似。然后,用本节所述的程序性语义方法执行这种最终语义表达式。LUNAR 可处理的一些查询样例包括:

Give me all lunar samples with magnetite.

(给出带有磁铁矿物质的所有月球岩石样本。)

In which samples has apatite been identified?

(已经确定在哪些样本中含有磷灰石?)

What is the specific activity of A126 in soil?

(土壤中 A126 的特定活动是什么?)

What is the average concentration of olivine in breccias?

(角砾岩中的橄榄石的平均密度是多少?)

In which breccias is the average concentration of titanium greater than 6 percent?

(在哪些角砾岩中,钛的平均密度大于%6?)

要了解 LUNAR 的更多信息,请参见 Woods(1970;1977;1978)。

13.8 综合的知识表示方法

即使知识表示语言是一阶的,在定理证明中通用的搜索策略对实际系统而言也常常效率很差。然而,将推理看做定理证明,这样带来的理论上的简洁性,仍旧具有很多有吸引力的特点。综合的知识表示系统试图在保持定理证明系统的理论框架的同时,在一些任务上,还获得程序性推理在高效率方面的优势。

在开始时,合一和反证法的思想可以应用于很多系统。但是,一个综合的系统并不能完全依赖于这些技术。相反,特定形式的推理可以用更有效的专用技术来实现。例如,分析一个知识表示系统中类型层次结构的实现。我们先前看到的类型层次结构可以表示为公理[例如, $\forall x. DOG(x) \supset MAMMAL(x)$]、图或者语义网络。这些技术形式上可能是等价的,但实际上会产生完全不同的计算性质。

将这些技术综合在一起的一种方法是,假定一种与 13.2 节讨论的受限量词逻辑相似的类型化逻辑。类型层次结构可用语义网络结构预先定义。其中,“DOG”是“MAMMAL”的子类型,而后者又是“ANIMAL”的子类型,而“Fido1”预先定义成集合“DOG”的一个成员。已知这些知识,该知识库表示断言“All animals have a mother”的结果如下:

$(MOTHER(?x:ANIMAL, Sk1(?x)) \leftarrow)$

其中,符号 $?x:ANIMAL$ 表示一个类型为“ANIMAL”的变量。通过扩展合一算法,可以推出这种表达式。这样,只要两个项类型兼容,它们就可以合一。也就是说,只要“Fido1”是集合“ANIMAL”的成员, $?x:ANIMAL$ 与“Fido1”就可以合一。这个条件可以用图 13.8 所示的语义网络一步一步来检查。这样,对于查询“Whether Fido has a mother”(Fido 是否有妈妈)[即 $MOTHER(Fido1, ?y)$],只用一个合一步骤就可以得到证明。

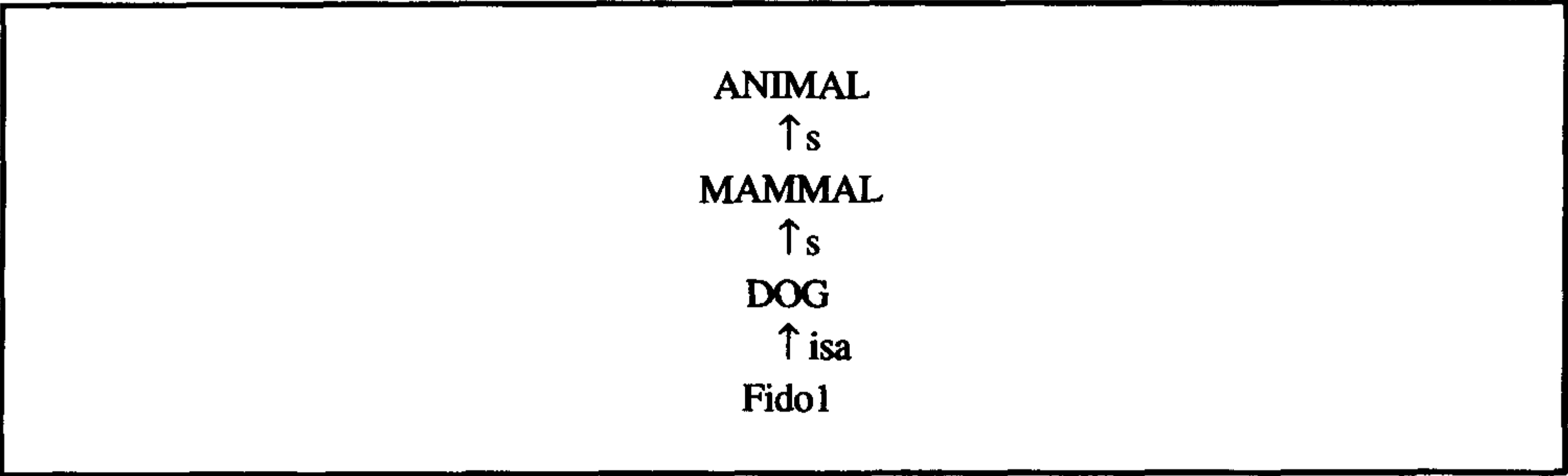


图 13.8 一个小型的层次体系

程序性的方法使我们可以写出高度优化的程序,这些程序比用公理的方法完成相同的工作要快得多。综合的表示方法可以用类型信息的语义网络对信息进行更直觉的表示,并允许其他非演绎算法在语义网络上执行。

另一个非常高效的专用推理系统与等价性推理相关。直接将等价性作为公理放到定理证明系统中是很难的。这是因为,对同一个对象而言如果只是使用两个不同的名字,将很难表示出两个不同但等价的公式。但是,有一种有效的算法,能够记录基于等价类的基项之间的等价信息。如果这些技术能够集成到合一程序中,就不需要直接将等价性作为公理写到系统中。相反,扩展的合一算法将用等价程序来检查两个项是否等价并进而合一。

其他几种专用的推理系统可以这样实现:在系统中集成某些专门定义的程序,这些程序用来确定一些特定谓词的真值。这种技术称为附属程序。例如,思考一下时间的推理。虽然可以用时间公理化来驱动时间性的推理,但这样的系统效率很低。有些专门的推理技术能有效地管理时间信息,因此可以作为特殊的谓词集成到综合系统中。为了理解这一点,我们分析命题在一个推理系统中充当的角色。一般有三种不同的操作可应用于命题:

断言为真(即,将其添加到 KB 中)

查询它是否为真(即,激活定理证明程序来加以验证)

撤销它(即,将其从知识库中删除)

虽然每个操作都是用定理证明系统中的术语来定义的,但这并不表示它是完成这些操作的惟一途径。事实上,可以定义任意过程来执行每一个任务。例如,考虑一个特定的时间推理系统,它维护一个时间关系图,并使用图搜索技术来构建时间关系。假定知识库中有一个谓词“BEFORE”,表示一个时间在另一个之前。当命题(BEFORE t_1 t_2)加到知识库中时,这个特定的时间推理过程就被激活,并将信息加到时间图中去。在查询同样的命题时(或者是直接由用户提出的查询,或者是一个复杂证明过程的其中一步),系统就调用该特定的时间推理过程来进行构造。这样,这个专用的时间推理程序就完全集成到定理证明程序中,并且一旦需要时间性信息时就会调用。当然,要完全集成起来,这种专用的推理还要能处理变量并返回与合一等价的结果。例如,定理证明程序需要构造(BEFORE t_1 ? x),那么时间推理程序就要返回一个对? x 的约束。除此之外,当需要寻找另一个答案时,还要能处理回溯。

为将专用的推理算法集成到统一的框架中,综合推理系统提供了一种很有吸引力的方法。

13.9 小结

自然语言理解需要有对世界知识进行表示和推理的能力。虽然知识表示的方法之间有很大的区别,但是,自然语言理解所使用的知识表示方法至少都要具备下面这些通用能力:

- 完整的逻辑运算符和逻辑量词处理能力,如一阶谓词演算所需要的这种能力
- 能够对这个领域内出现的对象和情境的默认的、固定模式的信息进行表示的方法
- 对有限集的明确表示和推理的方法
- 对时间性信息的表示和推理的方法

这些都具有代表性,但是不可能穷尽所有涉及的领域。例如,在一个完全通用的语言生成系统中,还需要研究语言中的空间信息以及心理状态的表示等。

本章提出了一种抽象的表示语言,这种语言将一阶谓词演算的技术和基于框架的方法结合在一起,可以满足前面提出的需求。可以利用不同的技术,将这种抽象的表示实现为不同的知识表示系统。任何知识表示系统必须支持基本的模式匹配的能力,因为它能为合一的概念和基于反证的推理提供一个形式化的基础。很多系统对部分或所有的推理任务也都指定了专门的过程。如果一个系统利用了很多技术,包括演绎性和程序性技术,则这样的系统称为综合的系统。

13.10 相关工作与深入阅读材料

在人工智能中,知识表示是一个非常广的研究领域,也是自然语言理解以外的很多其他问题的基础。Brachman 和 Levesque(1985)中的一些文章对这个领域做了很好的介绍。在这一领域中,当前研究的较好例子都可以在知识表示和推理会议(“Conferences on Knowledge Representation and Reasoning”)的论文集中找到[如 Brachman, Levesque 和 Reiter(1989); Allen, Fikes 和

Sandewall(1991); Nebel, Rich 和 Swartout(1992)]。Norvig(1992)是一篇很好的论文,这篇论文详细介绍了知识表示系统的一些实现技术。

Minsky(1975)引入了框架的概念,这一概念引出了很多关于知识表示的后续工作。基于这些想法的较早的知识表示系统之一是 KRL(Bobrow 和 Winograd, 1977)。Hayes(1979)用一阶谓词演算对 KRL 进行了分析。现在的大部分知识表示系统都是框架、语义网络和演绎逻辑的综合[如 Brachman 和 Levesque(1985)中描述的这些系统]。有一大类系统利用对象的类别描述来组织知识,称为项分类语言(term subsumption languages),在 Brachman 和 Levesque(1985)中有很好的例子。关于语义网络的很好的参考文献是 Sowa(1991)。

基于分解为原语集合的知识表示方法的最有力的支持者是 Schank(1975)和 Wilks(1975)。不过,目前大部分的表示系统都利用抽象作为表示中的组织工具,这些表示系统常常是基于语义网络和框架系统的。有很多文献对语言学中的分解提出了很多建议,如 Dowty(1979)和 Jackendoff(1990)。

利用集合中的列举法对量词进行处理在计算机系统[如 Woods(1977); Warren 和 Pereira(1982); Alshawhi(1992)]和语言学[如 McCawley(1993)]中都非常普遍。

时态和体态的研究在语言学、哲学和计算语言学中都很活跃。在计算语言学领域, *Computational Linguistics* (1988)有一个专刊专门讨论时态和体态的研究,是这个领域非常好的参考文献。在语言学领域, Dowty(1979; 1986)和 Bach(1986)是很好的入门资料。最近的工作包括 Parsons(1990)和 Pustejovsky(1991)。时态研究的经典文献是 Reichenbach(1947),也有很多研究把时态当做模态运算符来处理[如 Prior(1967)]。McCawley(1993)详细介绍了处理时态和体态的一些语言问题。Allen(1984)描述了一种时态逻辑,这种时态逻辑明显包括三种不同体态类中的一些谓词。Davis(1990)介绍了基于逻辑的表示方法,其中专门探讨了时间、空间以及这个世界的很多其他方面的表示方法。

大多数演绎技术都是从 Robinson(1965)所介绍的消解理论证明的研究中演变发展来的。Robinson 介绍了消解规则,并证明了消解反证技术(resolution refutation proof technique)是完备的。失败证明(proof by failure)技术首先用于人工智能编程语言 PLANNER(Hewitt, 1971),其形式化的工作由 Clark(1978)完成。关于默认逻辑的很多工作来自于 Reiter(1980)的研究。Etherington 和 Reiter(1983)利用这一形式体系形式化地定义了类型层次体系中带例外的继承。很多文献讨论了利用基于最小模型的语义技术的知识表示, Genesereth 和 Nilsson(1987)详细讨论了这个问题。

程序语义学在早期系统中应用广泛,较出名的包括 Winograd(1973)和 Woods(1978),它仍然是数据库查询系统中一种很常用的技术。CHAT-80 系统(Warren 和 Pereira, 1982)利用类似的基于 PROLOG 的表示方法实现了一个精致的、非常强大的查询工具。Webber(1992)对问答系统(QA, question answering)的一些技术进行了简要的阐述。目前,问答系统的一个典型例子是 TEAM 系统(Grosz 等, 1987)。TEAM 的一个目标是可移植性,即可以方便地移植到不同的数据库,而不需要重新编写语法和语义解释器。为此,TEAM 利用了与第 8 章中的方法类似的上下文无关的逻辑表达式。

13.11 习题

1. 【易】对下列句子的语义,给出可接受的基于逻辑的表示形式(把重点放在量词的解释上)。如果一个句子有整体性/个体性的歧义,请给出两种解释。

Several men cried.

Seven men in the book met in the park.

All but three men bought a suit.

2. 【易】对篇章 a 和篇章 b,请说明第一个句子是否蕴涵或隐含其后面的每一个句子,或者是否有语义联系。用一些语言学测试来证明你的答案。

- a. John didn't manage to find the key.

(John 没有设法去找这把钥匙。)

John didn't find the key.

(John 没有找到这把钥匙。)

John looked for the key.

(John 找过这把钥匙。)

The key is hard to find.

(这把钥匙很难找到。)

- b. John was disappointed that Fido was last in the dog show.

(John 对 Fido 在狗展中排在最后很失望。)

Fido was last in the dog show.

(Fido 在狗展中排在最后。)

Fido was entered in the dog show.

(Fido 参加了狗展。)

John wanted Fido to win.

(John 希望 Fido 取胜。)

Fido is a stupid dog.

(Fido 是一条笨狗。)

3. 【易】对下列句子中的提示动词短语进行分类,说明是描述状态、活动、达成还是成就。用一些能够说明这些语言特性的例子来证明你的答案,并讨论分类过程中遇到的一些难题。

Jack ran to the store. (Jack 跑着去了商店。)

Jack was running to the store. (Jack 正在向商店跑去。)

Jack hated running. (Jack 痛恨跑步。)

Jack runs every day. (Jack 每天都跑步。)

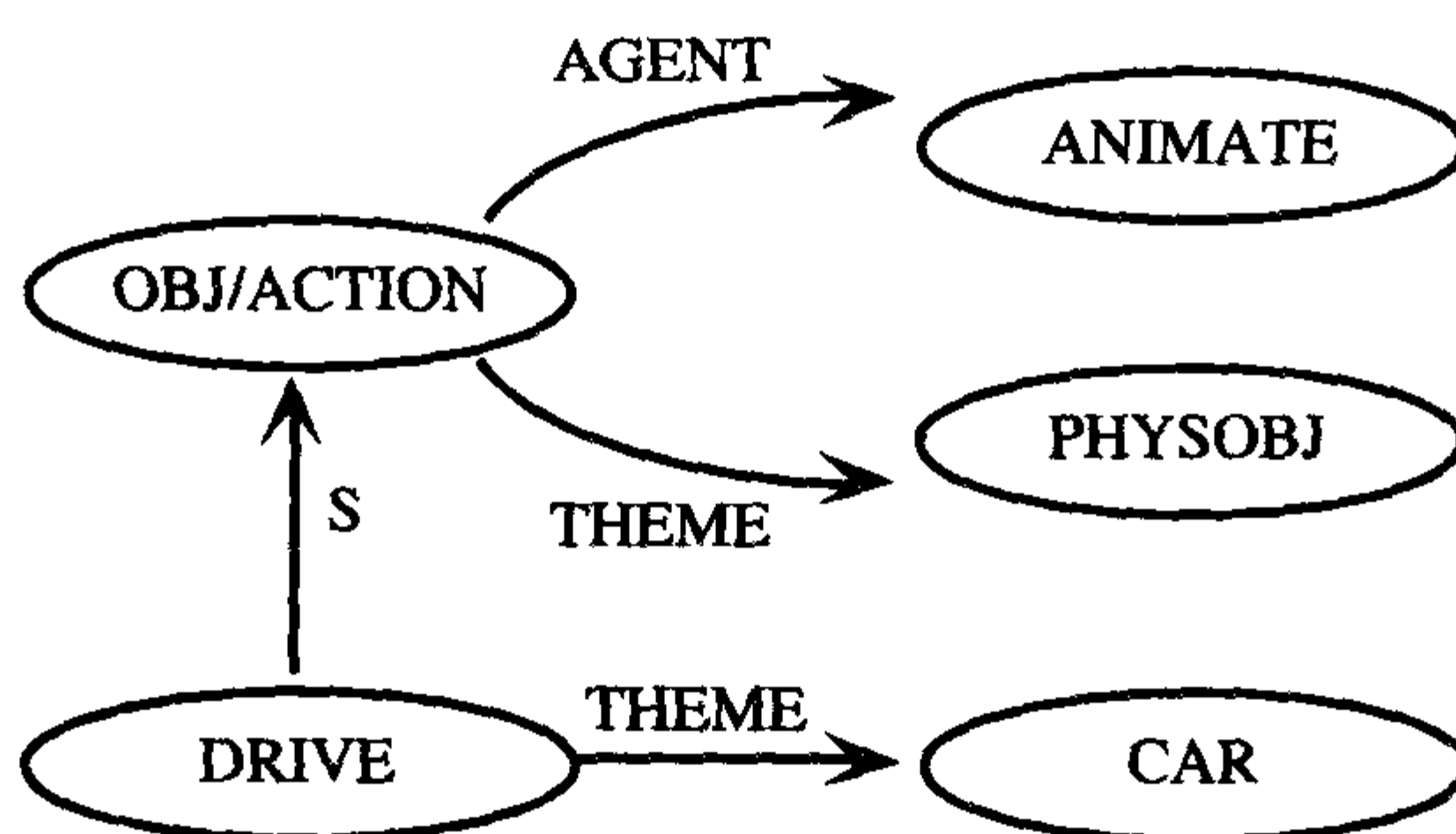
Jack stopped running when he broke his leg. (当 Jack 摔坏腿时,他就停止了跑步。)

4. 【中】分解语义学的一个经典例子是用因果运算符和谓词 DIE 来表示动词“kill”。具体来说,“John killed Sam”的语义可以表示为:

$CAUSE(John1, DIE1(Sam))$

请问,这一分解是否完全刻画了动词“kill”的语义? 考虑“John killed Sam”和“John caused Sam to die”这两个句子的语义在所有情境中是否完全一样? 给定你在这个问题上的立场,在知识库中是将“kill”的所有实例都分解为这个形式,还是使用一个语义假设并在知识库中保留“KILL”这个谓词? 论证你的回答。

5. 【中】利用 13.1 节介绍的将继承网络转换为逻辑的方法,给出下面这个简单网络的一些公理。



需要注意的是,“drive”行为的定义对 THEME 角色的类型给予了更严格的约束。这一公理化操作是否合适? 即,作为结果的公理是否一致,并且对任何“drive”行为 D,是否都满足 $D, \exists o. THEME(D, o) \wedge CAR(o)$? 在通用类 OBJ/ACTION 上的 THEME 角色的定义与“drive”行为的定义是否有冲突? 在回答这些问题时,请明确指定在每个证明中需要对层次化体系所做的任何假设。

6. 【中】利用基于框架的表示方法,请对下列句子中的动词“drive”的语义所对应的行为类 DRIVE 给出定义:

i. I drove to school today. (今天,我驾车到了学校。)

具体来说,给出的定义要足够详细以便下面的每一个句子都能够从句子 i 推理得到。

ii. I was inside the car at some time. (在某些时间,我在车内。)

iii. I had the car keys. (我有这辆车的钥匙。)

iv. The car was at school for some time. (在某些时间,这辆车在学校。)

v. I opened the car door. (我打开了这辆车的门。)

对每一个句子,请详细讨论所需的知识是怎样表示的(作为前提、结果、分解等)? 有什么一般性原则来证明它是句子 i 的结论? 确定能够从句子 i 推理得到的除此之外的三个结论,并讨论得到每一个结论所需的知识是怎样表示的? 任何定义是否都应该认为是默认为知识? 如果是,为什么? 如果不是,请给出一个在利用默认为知识的情况下可以得到的进一步的结论。

7. 【中】给定局限于如下基本文字命题的一个固定的数据库,利用封闭世界假设,考虑一个在这个数据库上的问答系统:

TYPE(FORD,COMPANY),TYPE(GM,COMPANY)
TYPE(CAR1,AUTO),...,TYPE(CAR9,AUTO)
MADE-BY(CAR1,FORD),COLOR(CAR1,WHITE)
MADE-BY(CAR2,FORD),COLOR(CAR2,WHITE)
MADE-BY(CAR3,FORD),COLOR(CAR3,BLACK)
MADE-BY(CAR4,FORD),COLOR(CAR4,WHITE)
MADE-BY(CAR5,GM),COLOR(CAR5,RED)
MADE-BY(CAR6,GM),COLOR(CAR6,WHITE)
MADE-BY(CAR7,GM),COLOR(CAR7,BLUE)
MADE-BY(CAR8,GM),COLOR(CAR8,RED)
MADE-BY(CAR9,GM),COLOR(CAR9,BLUE)

- a. 给出英语量词“most”的适合于数据库查询的程序语义。讨论遇到的任何复杂问题和需要做的任何假设。
- b. 对句子“Most cars made by some company are white”,对应量词辖域指定的不同,给出两个不同的解释。对于每一种解释,请给出回答问题过程的非形式化的跟踪信息。这两种情况的答案分别是什么?
- c. 要选择 b 中的最可能的解释,需要运用哪些领域的知识? 利用 b 中的分析作为例子,描述歧义消解的过程以便能够选择最可能的解释。

第 14 章 局部篇章上下文和指代

本章主要讨论与指代和局部篇章上下文相关的一些问题,包括指代、省略、VP 回指和“one”型回指^①。为了详细探讨回指处理的问题,引入了一个简单的称为历史记录列表的全局性篇章结构的模型。这样,就可以探讨一系列与指代有关的问题,而不需要等到探讨完全局性篇章上下文之后才来讨论指代。

14.1 节探讨局部上下文的概念,并且介绍了篇章实体的概念,用来作为指代表达式的指代对象。14.2 节逐步阐明了历史记录列表表示方法,并且探讨了定指性名词短语的解释。14.3 节详细研究了代词指代问题,并且采用篇章中心这一概念逐步提出用结构化的优先表示来解决代词的指代问题。14.4 节和 14.5 节均探讨了定指性描述。其中,14.4 节研究了单数的定指性描述,14.5 节研究了复数的描述和集合。14.6 研究了省略问题,其中局部上下文用来补充当前句子丢失的信息。最后,14.7 节研究了表层的回指,它是介于代词指代和省略中间的一种现象。

14.1 定义局部篇章上下文和篇章实体

首先,我们分析局部上下文,包括先前句子的句法和语义结构,以及句子涉及的一些对象,这些对象可以是后来的代词和其他定指性名词短语的先行词。对于很多情况来说,局部上下文是很有帮助的。如,下面的句子可能包含代词的先行词:

1a. Jack_i lost his wallet_j in his car. (Jack 把钱包忘在汽车里了。)

1b. He_i looked for it_j for several hours. (他找了它几个小时。)

另外,在下面的例子中,局部上下文定义了这样的结构,这些结构对解释采用动词短语省略的句子是非常有用的:

2a. Jack forgot his wallet. (Jack 忘了带钱包。)

2b. Sam did too. (Sam 也是。)

这些动词短语省略一般用来指上一个句子所描述的事件,如:

3a. Jack forgot his wallet. (Jack 忘了带钱包。)

3b. He looked for someone to borrow money from. (他找人借钱。)

3c. Sam did too. (Sam 也是。)

在这个篇章中,句子 3c 不是指 Sam 忘记带钱包。

但是由于连词的存在,利用句子作为局部上下文的基本单元存在一些问题。比如,可以修改篇章 3 以便使句子 3a 和句子 3b 结合为一个句子,但是这并没有改变可能的解释:

^① 例如在篇章“John had a blue t-shirt, Mary had a red one”中,这里的“one”指代上文中的“t-shirt”。——译者注

4a. Jack forgot his wallet, so he looked for someone to borrow money from.

(Jack 忘了带钱包,所以他找人借钱。)

4b. Sam did too. (Sam 也是。)

很难把句子 4b 解释为 Sam 忘记了带钱包。与其相对,一个包含连词的句子支持两个并列成分之间的 VP 省略,比如:

5. Jack forgot his wallet, and Sam did too. (Jack 忘了带钱包, Sam 也是。)

一种更复杂的情况是,下面的例子涉及两个动词短语的并列。在这种情况下,由连词所连接的两个并列成分就可以作为一个整体,构成支持省略的上下文,比如:

6a. Jack forgot his wallet and lost his credit cards. (Jack 忘记带钱包并且丢了信用卡。)

6b. Sam did too. (Sam 也是。)

这种情况下,句子 6b 是指 Sam 也忘记带钱包并且丢了信用卡。

基于上面的例子,一种很自然的想法是局部上下文来自于前面的主要子句而不是前面的那个句子。例如,因为连词“and”能够连接主要子句,所以句子 5 可以解释为,前面第一个连词为后面的句子提供了局部上下文。VP 连词出现在惟一的主要子句中,所以 6b 的局部上下文是句子 6a 中的所有信息。从属子句作为主要子句的一部分,不能创建新的局部上下文,如下面的例子:

7a. Jack forgot his wallet when he went out to the movies.

7b. Sam did too.

句子 7b 可以解释为 Sam 丢了钱包(或当他去看电影时丢了钱包),但并不是说他去看电影。

局部上下文的一个很重要的方面是代词的可能先行词的列表,我们称之为篇章实体(DE, discourse entity)列表。篇章实体列表是定义在知识库中的常数的集合,这一集合表示最近的主要子句中涉及的,并且可以被下文中的代词所指代的对象。有时候,前面的句子并没有明确涉及一个篇章实体,只是隐含地介绍。考虑到这样的情况,我们将常常讨论一个句子所引发的对象,这里所指的对象包括明确提及的和隐含提及的。在后面的章节中,将讨论这两种情况。

当我们说一个代词有某一先行词时,实际上意味着这个代词和先行词是指同一个对象。值得注意的是,这个代词及其先行词可以指同一个对象 X,即使说话者和听者双方都没有认定 X 为一个具体的对象。简单说来就是二者指同一个对象,但不关心具体对象是什么。考虑下面的篇章片段:

8a. John bought a car_i yesterday. (John 昨天买了一辆车。)

8b. It_i was very expensive. (它很贵。)

即使说话者和听者都没有看见这辆车或者没有确认它,这句话的意思也是很容易理解的。这种不定指性描述词“a car”证实车的存在,代词“it”用来增加这辆车的信息。这一语句很好理解,即使谁都不知道这辆车的情况。那么,当不知道要指代的对象时,在知识库中应怎样表示这个常数呢?面对这样的问题,一些研究人员讨论了完全不同层次的表示篇章实体的方法。不过在这里,我们采用了一种简单方法,也就是使用 Skolem 函数来表示篇章实体。需要记住的是,Skolem 函数(或 Skolem 常量)只是引入到表示语言中的一个新的术语。在可能的情况

下,我们将逻辑形式中引入的篇章标记作为新的 Skolem 常量。这样,如果“a car”的逻辑形式为 $\langle A\ c1\ CAR \rangle$,那么产生的 Skolem 常数就是 C1。共指用等式来表示。这样,如果一个代词“it”的逻辑形式为 $(PRO\ i1\ IT1)$,那么,它与 C1 之间是共指关系这个事实就表示为 $(I1 = C1)$ 。

14.1.1 生成篇章实体

与子句的解释类似,通常,每一个名词短语都要生成一个篇章实体。不同的名词短语对各自生成的篇章实体赋予不同的约束。一个不定指性名词短语往往生成一个新的篇章实体,在知识库中不需要进一步标识。而一个专有名称通常描述一个已经在知识库中定义的与该名称相关的对象。一个定指性名词短语(包括代词)一般指一个篇章中已经涉及的一个对象,通常是局部上下文中的篇章实体。复数名词短语产生一系列对象。一个包含连词的复杂名词短语引发一个包含所有并列成分的集合。例如,名词短语“John and Mary”引发三个篇章实体: $John1, Mary1$ 和集合 $\{John1, Mary1\}$ 。

这一节的后半部分将探讨由不定指性名词短语所引发的篇章实体。这类短语非常重要,因为不定指性名词短语引入了很多新的篇章实体,并且提供了处理定指性指代的更多背景。为了计算出篇章实体的集合,我们首先将逻辑表达式转换为 12.4 节中的量词表示形式,使用明确的集合,将所有自然语言的量词化简为全称和存在量化表示形式。对于这个简单的例子来说,我们假定所有的专有名称和定指性名词短语都已经用知识库中表示所指代对象的常量替换掉了。

对于复数名词短语,无论是解释为一个整体,还是解释为多个个体,都要引发同一个集合。解释为一个整体时,这个集合直接在命题中用做一个参数;而解释为多个个体时,这个集合作为全称量词所指的范围。要定义这个集合,修饰语的转换相应地也必须分为两类,分别是对整个集合的约束(SR)和对集合中每一个个体的约束(IR)。例如,下面句子中的名词短语“Three boys”的转换:

9. Three boys lifted Fido.(三个男孩抬起了 Fido。)

将引入一个 SR,表示这个集合有三个元素,以及一个 IR,断言其中每一个元素都是一个孩子。图 14.1对一些不定指性名词短语的转换进行了总结。

例子	篇章实体 (DE)	集合约束 (SR)	个体约束 (IR)
a man	MI	无	$MANI(MI)$
three women	WI	$ WI = 3$	$WI \subseteq \{w \mid WOMANI(w)\}$ 或等同于 $\forall w. w \in WI \supset WOMANI(w)$
some black cats	CI	$ CI > 1$	$CI \subseteq \{c \mid CATI(c) \ \& \ BLACKI(c)\}$

图 14.1 不定指性名词短语的转换

14.1.2 全称量词辖域内的不定指性名词短语

当一个不定指性名词短语出现在一个表示个体的全称量词的辖域内时,问题就复杂了。例如,在一个全称量词辖域内的单个不定指性名词短语引发的是一个集合而不是一个个体。比如下面的例子:

10a. Three boys_i each bought a pizza_j. (三个男孩每人买了一张比萨。)

10b. They_i ate them_j in the park. (他们在公园中吃掉了它们。)

值得注意的是,“They_i ate it_j in the park”不是句子 10a 的合理延续句。需要什么属性来定义句子 10a 中的“a pizza”所生成的篇章实体呢? 集合 $\{x \mid \text{Pizza}(x)\}$ 太笼统,句子 10b 中的“them_j”也不能指所有比萨的集合。较好的办法是,“a pizza”为一个新的篇章实体 $P1$,表示 $\{x \mid \text{Pizza}(x)\}$ 的子集。基于这一解释,句子 10b 中的“them_j”指上一个句子中提到的比萨的集合。但是这一方法也会丢失 $P1$ 内容的信息,即 $P1$ 由是句子 10a 中提到的男孩所买的比萨组成的集合。 $P1$ 的表达可以从句子 10a 的最初形式中推理得到:

$$\forall b: b \in B1.$$

$$\exists p: \text{Pizza}(p) . \text{Buy}(b, p) \text{ where } |B1| = 3 \ \& \ B1 \subseteq \{x \mid \text{Boy}(x)\}$$

Skolem 化的形式为:

$$\forall b: b \in B1 . \text{Pizza}(\text{sk4}(b)) \ \& \ \text{Buy1}(b, \text{sk4}(b))$$

其中, $\text{sk4}(b)$ 是新函数,用于产生每个男孩所买的比萨。这样,集合 $P1$ 可以定义为由这个新函数产生的比萨的集合:

$$P1 = \{x \mid \text{Pizza}(x) \ \& \ \exists y: y \in B1 . x = \text{sk4}(y)\}$$

这个公式准确表示了这些男孩买的那些比萨。这个句子的完整分析过程如图 14.2 所示。

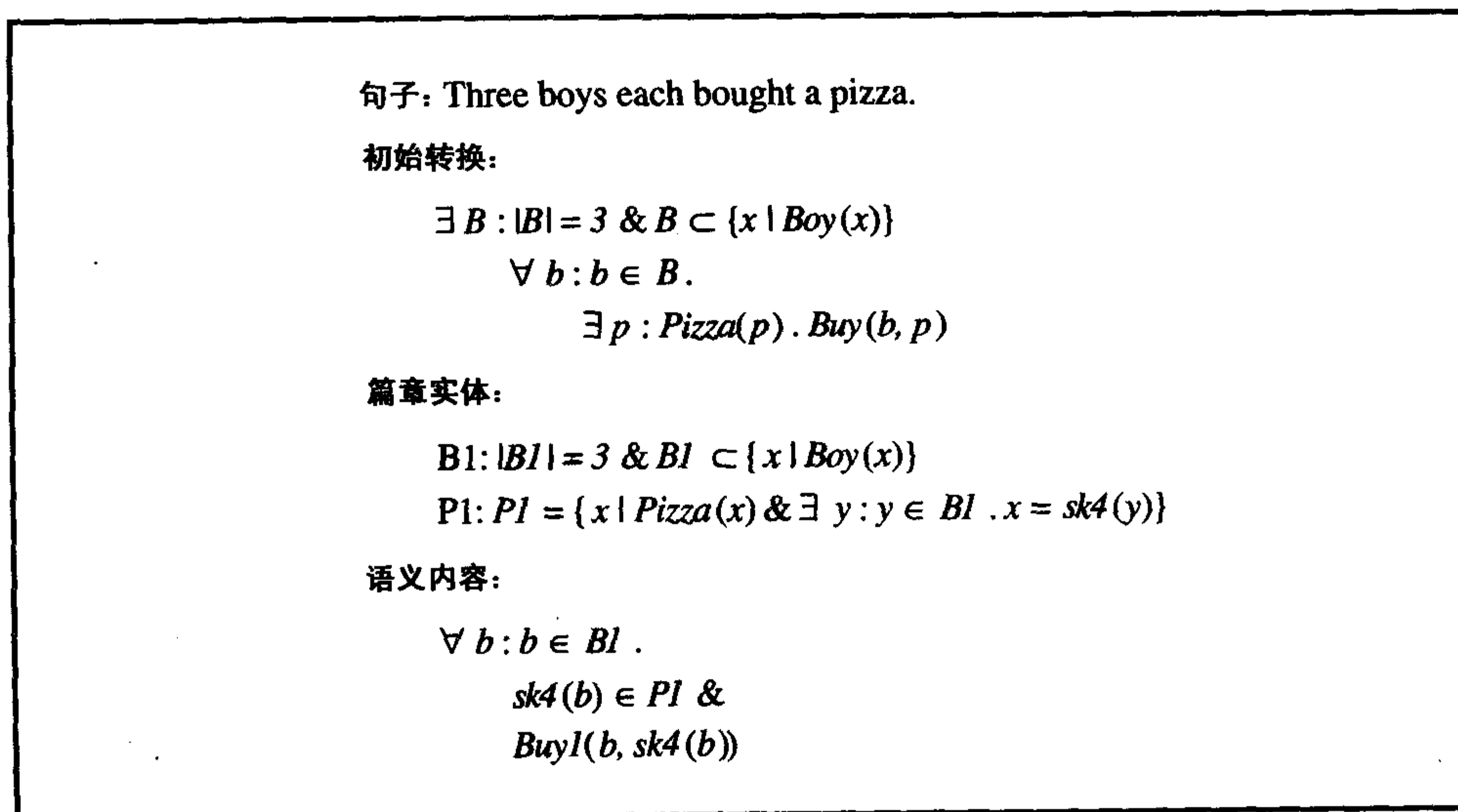


图 14.2 由句子“Three boys each bought a pizza”所生成的篇章实体

14.2 一个基于历史记录列表的简单回指模型

这一节将探讨识别代词的先行词的一种简单方法。最简单的方法是历史记录列表,它是在前面句子基础上生成的篇章实体的列表。根据前文的介绍,历史记录列表就是一个结构的序列,这些结构对应于前面局部上下文的篇章实体。列表中首先列出来自于当前局部上下文的实体(即前一个子句生成的实体),然后是前一个句子的局部上下文中的实体,依次类推。

在第 12 章,我们已经了解了代词怎样指代同一个句子中的对象,并推导出了一系列共指约束。这些约束说明,在同一个句子中,每个代词的先行词可能或者不可能是哪些对象。这些约束也影响了句子之间的情况。例如,即使在不同句子之间出现的同一个对象,自反性约束也存在,如下面的篇章:

11a. Jack saw Sam at the party. (在晚会上 Jack 看见了 Sam。)

11b. Sam gave him a drink. (Sam 给了他饮料。)

在这个例子中,自反性约束表明“him”和 11b 中的“Sam”不能互指。这也禁止了代词“him”与句子 11a 中的由“Sam”引发的篇章实体互指。

代词的可能的先行词并不限于在局部上下文中出现,但是局部上下文对解决代词指代问题非常重要。代词的大部分先行词都出现在同一个句子中或局部上下文中。提及的先行词离当前篇章越远,被某一代词指代的可能性就越小。

一旦定义了由句子生成篇章实体的算法后,历史记录列表的概念就非常简单。历史记录列表包括最近产生的所有篇章实体。一些系统允许最近一个或两个局部上下文,而另外一些允许历史记录列表无限制增长。给定一个历史记录列表,搜索一个先行词的算法描述如下:检验最近的局部上下文,发现满足该代词的所有约束的先行词。约束可以来自于任何来源。例如,自反性约束将限制一些对象成为先行词;性别和数的约束排除了另外一些对象;而施加选择性的约束条件所产生的约束可能会进一步引入其他约束。如果当前局部上下文中没有发现先行词,则移到下一个最近的局部上下文中并进行搜索。这一算法执行了所谓的“最近约束”(recency constraint),该约束规定先行词应该是满足所有约束的最近提及的对象。例如,下面关于帆船比赛的场景:

12a. The companies had a lot of money and spent lavishly on their boat.

(这些公司有很多钱,并花了很多钱在船上。)

12b. The boys, in contrast, built their boat on a tight budget.

(相反地,这些小孩却只有很少的钱来造船。)

12c. They knew they would win the race easily.

(他们知道自己能够很轻易地获胜。)

尽管单纯基于语义也可能是那些公司认为他们自己能够赢得比赛,但是句子 12c 中的“they”很可能还是指那些孩子。

句子 12c 的历史记录列表如图 14.3 所示。为了寻找代词“they”的先行词,可以寻找满足代词约束条件的历史记录列表中的第一个对象。在这种情况下,答案是满足 $THEY1(x)$ 的一个对象 x ,也就是任何一个复数的对象。第一个被检验的实体 B2 被选择为一个指代对象。同

样的方法也可以应用到定指性描述中。例如,如果句子 12c 为“*They knew the boat would win easily*”,那么“*the boat*”所指的是满足约束 $BOAT(x)$ 的第一个对象 $B3$ 。

句子	生成的篇章实体
句子 12b	$B2: B2 \subseteq \{x \mid Boy(x)\}$ $B3: Boat(B3) \ \& \ BuiltBy(B3, B2)$ $B4: Budget(B4) \ \& \ Limited-Funds(B3)$
句子 12a	$C1: C1 \subseteq \{c \mid Company(c)\}$ $M1: Money(M1)$ $B1: Boat(B2) \ \& \ OwnedBy(B2, C1)$

图 14.3 由篇章 12 生成的历史记录列表

在很多计算方法中,历史记录列表是一种基本的方法。我们必须进一步改进这一方法以便能够处理特殊情况。下面的两节将详细解释代词和定指性描述。在第 16 章中,在对全局的篇章结构进行更详细的讨论时,我们会再次考虑历史记录列表。

14.3 代词和中心

最近约束对不同的局部上下文赋予了一个优先顺序。那么,在一个单独的局部上下文中,不同的篇章实体是否有优先顺序呢?答案是肯定的。在主要子句中起关键作用的对象看起来要比附属短语和从属子句中的对象的优先顺序要高。在一些情况下,这些优先顺序会干扰逻辑上的可能解释。例如下面的篇章[摘抄于 Wilks(1975)]:

Jack drank the wine on the table. It was brown and round.

(Jack 喝掉了桌子上的酒。它是棕色的且是圆的。)

尽管最近约束和语义约束提示我们“it”可能是指“table”,但是大多数人认为很难确定“it”的所指对象。并且,很多人认为第二个句子是异常的,或者至少是让人感到滑稽的,因为“it”似乎是指代“wine”(酒)。

在主要子句中的主要变量之间的优先顺序更微妙,例如下面的篇章:

13a. Jack saw Sam at the party. (在晚会上 Jack 看见了 Sam。)

13b. He went back to the bar to get another drink. (他回到酒吧,又喝了一杯。)

虽然句子 13b 中的“he”从逻辑上可能指 Jack 或 Sam,但是在这个句子中,更倾向于指 Jack。这种优先顺序并没有强大到足以凌驾于语义和上下文的影响之上。但是,下面的例子:

14a. Jack saw Sam at the party. (在晚会上 Jack 看见了 Sam。)

14b. He clearly had drunk too much. (他明显喝多了。)

句子 14b 中的“he”被解释为指代 Jack 显得有些牵强,因为如果这样解释,将很难看出这两个句子是如何联系在一起的。

另一个因素也会影响解释。该因素基于篇章焦点或中心的概念,而这些理论的直观含义

是篇章会围绕篇章所要阐述的对象来组织。一些句子会围绕一个对象(我们称之为中心)来展开论述,然后会转移到新的对象。另一个关键的直观含义是一个句子的中心倾向于被代词化,这会影响代词的解释。因为一旦确定了一个中心,后续的代词就会强烈倾向于继续指向这个中心。例如:

15a. Jack left for the party late. (Jack 很晚才去参加晚会。)

15b. When he arrived, Sam met him at the door. (当他到达时, Sam 在门口碰到了他。)

15c. He decided to leave early. (他决定早点离开。)

从语义上理解,句子 15c 中的“*He*”指 Jack 或者 Sam 都是可能的。从结构上更倾向于 Sam,因为在句子 15c 中,“*he*”处于主要子句的中心地位。然而,根据中心理论,可知 Jack 是先行词,因为句子 15b 中的代词是指 Jack,所以是句子 15b 的中心。在句子 15c 中,没有任何信息表明中心有变化。

更准确地说,在中心理论中采用了两个互相影响的结构:

- 局部上下文中的篇章实体,我们称之为潜在的下一个中心(或前瞻性中心, *forward-looking centers*)。这些篇章实体按照结构优先考虑顺序排列为一个列表。首先是主语,其次是直接宾语、间接宾语和句子中的其他篇章实体。列表中的第一个被称为优先考虑的下一个中心,记为 C_p 。
- 另一个中心 C_b (回顾式中心, *backward-looking center*)表示当前句子的论述点。 C_b 是潜在的下一个中心之一,而且通常是被代词化的。

中心和代词化之间的约束可以表述如下:

中心约束 1: 如果局部上下文中的任何对象被当前句子中的一个代词所指代,那么该句子的中心一定也被代词化了。

中心约束 2: 中心必定是局部上下文中被代词所指代的最优先考虑的那个篇章实体。

中心约束 3: 从一个句子到另一个句子的过渡过程中,中心倾向于继续而不是被更换。

需要注意的是,根据约束 1,如果一个句子中只有一个代词,毫无疑义它就是中心。根据约束 2,如果下一个句子也包含一个代词,这两个代词很自然地都指向同一个对象。那么,它们是共指的。例如,考虑下面的句子:

16a. Jack₁ saw him₂ in the park₃. (Jack 在公园里看见了他。)

16b. He₄ was riding a bike₅. (他正在骑自行车。)

假定 DR_1 , DR_2 和 DR_3 分别代表“Jack”, “him”和“the park”的篇章实体。 DR_2 是句子 16a 的中心(即 C_b),潜在的下一个中心列表的优先顺序分别为 DR_1 , DR_2 和 DR_3 。这个列表的第一个元素 DR_1 是优先考虑的下一个中心(C_p)。给定这个局部上下文,现在考虑句子 16b。单纯从语义上来说,“*he*”可以指 DR_1 或 DR_2 。不过,考虑中心约束,更倾向于 $DR_4 = DR_2$,即继续同一个中心。按照这一解释, DR_4 将是句子 16b 的中心,也是优先考虑的下一个中心。

如果一个句子包含多个代词,情况就更复杂了,因为到底哪一个代词和这个中心对应是有歧义的。例如下面的句子中,中心约束没有表明优先顺序,句子 17b 中的“*he*”将根据句子内的信息来解释,即被解释为指代“Jack”。

17a. While Jack₁ was walking in the park₂, he₁ met Sam₃.

(当 Jack 在公园中走路时,他遇到了 Sam。)

17b. He₄ invited him₅ to the party₆. (他邀请他参加晚会。)

根据约束 1, 句子 17a 的中心无疑是 DR₁, 也就是 Jack, 因为这句话只包含一个代词。然而, 句子 17b 包含两个代词, 因此中心约束可以是 DR₄ = DR₁ 或者 DR₅ = DR₁; 即句子 17b 中的 He₄ 无论是指代“Jack”或者“Sam”都没有优先顺序。这可能是问题也可能不是问题, 完全依赖于个人感觉而定。但常常是, He₄ 更倾向于指代“Jack”。

为了探讨这个概念, 需要注意到, 基于当前中心和优先考虑的下一个中心, 两个句子的关联关系有 4 种不同的组合。如图 14.4 所示, Cb₁ 和 Cb₂ 是两个句子的中心, Cp₂ 是第二个句子的优先考虑的下一个中心。当这两个句子的中心保持不变时, 这一转移是延续或保持, 具体取决于这个中心是否是优先考虑的下一个中心。如果中心漂移了, 根据这个新的中心是否是优先考虑的下一个中心, 存在两种情况。

	Cb₂ = Cp₂	Cb₂ ≠ Cp₂
Cb₁ = Cb₂	延续	保持
Cb₁ ≠ Cb₂	转移到优先考虑的中心	转移到非优先考虑的中心

图 14.4 中心变化的类型

改进的约束 3 中应用了这些特性:

中心约束 3'——延续优先于保持, 保持优先于转移到优先考虑的对象, 转移到优先考虑的中心优先于转移到非优先考虑的中心。

让我们再研究一下篇章 17 的解释, 请参见图 14.5 所示的总结。因为“Jack”和“Sam”都倾向于被句子 17b 中的代词指代, 约束 2 需要的中心是“Jack₁”。但是这不能确定 He₄ 的指代对象, 因为它有两个可能的先行词, DR₁ (局部上下文的中心, 表示中心的延续) 和 DR₃ (表示转移到倾向的下一个中心)。约束 3' 倾向于解释 1, 即 He₄ 指代“Jack”。

句子 17a: While Jack₁ was walking in the park₂, he₁ met Sam₃。

篇章实体: Cp: DR₁ = Jack₁, 其他: DR₃ = Sam₁, DR₂ = Park₇

篇章中心: Cb: Jack₁

句子 17b: He₄ invited him₅ to the party₆。

解释 1: (延续)

篇章实体: Cp: DR₄ = Jack₁, 其他: DR₅ = Sam₁, DR₆ = Party₁

篇章中心: Cb: Jack₁

解释 2: (保持)

篇章实体: Cp: DR₄ = Sam₁, 其他: DR₅ = Jack₁, DR₆ = Party₁

篇章中心: Cb: Jack₁

图 14.5 中心的延续与中心的保持

中心的优先考虑顺序与句内成分的可能解释是如何相互影响的呢？要确定什么方法是最合适的，还要取决于对各种可能算法的进一步研究和评价。目前，一些算法总是倾向于句子内的指代对象，而另外一些算法则相反。一个有趣的组合是倾向于任何将代词当做中心的解释，如果失败，则倾向于句子内的解读而不是句子间的解读。无论策略是什么，很重要的一点是，更一般的上下文因素最终决定了最好的解释。任何只取决于结构属性的算法都是不完善的。这说明，如果一个基于结构的代词消歧算法为每一个代词都按优先考虑的顺序产生一个有序的可能指代对象序列，然后让通用推理系统来做最后的决策，那么这种算法将是非常有用的。

14.3.1 寻找可能的先行词

下面的例子说明了一个算法。该算法基于局部篇章上下文、共指约束和刚才讨论的中心约束的一个子集，对每个代词产生可能指代对象的一个有序列表。遗憾的是，由于共指约束，不存在一种方法可以独立地处理每一个代词。例如，在句子 17b 中，如果两个代词独立解释，每一个首选的指代对象将是局部上下文的中心，即“Jack”。但是，因为自反性约束的缘故，我们知道两个代词都不是指 Jack。除了对每一个代词的指代对象都列出所有组合情况外，在这个阶段，还没有什么办法可以避免这个问题。在下面的算法中，将跳过这一问题。两个代词最可能指代对象“Jack”，我们将求助于推理系统中的共指约束和解释过程。这样，推理系统在最后决策时，可能需要考虑中心约束 3。这个算法的 3 个步骤如下：

1. 对每一个代词产生可能先行词的有序列表。
2. 用一般的推理来选择合适的先行词。
3. 用步骤 2 的结果来定义句子的 Cb，并将其用做下一个句子的局部上下文的一部分。

我们将在第 15 章讨论步骤 2。图 14.6 给出了步骤 1 的算法。考察篇章 17 中的算法，句子 17a 所生成的局部上下文如图 14.5 所示。句子 17b “He invited him to the party”的逻辑表达式如下：

```
(PAST (INVITE1 i1 [AGENT (PRO h1 (& (HE1 h1) (≠ h1 h2))])
      [THEME (PRO h2 (& (HE1 h2) (≠ h1 h2))])
      [PURPOSE <THE p1 PARTY1>]))
```

考虑代词(PRO h1 (& (HE1 h1) (≠ h1 h2))), 候选的初始列表包括 Cb(Jack1), Cp(Jack1 again), 句子内的先行词(为空)和其他篇章实体(Sam1, Park3), 最后的列表为(Jack1 Sam1 Park3)。只有前两个满足 h1 上的约束, 所以逻辑表达式更新为:

```
(PRO h1 HE1 (≠ h1 h2) (REF-LIST h1 (Jack1 Sam1)))
```

第二个代词的处理过程类似, 逻辑表达式更新为:

```
(PRO h2 HE1 (≠ h1 h2) (REF-LIST h2 (Jack1 Sam1)))
```

这一信息将传递给第 2 步中的通用推理系统。如果它与直觉相一致, 则第 2 步认定 h1 的先行词为“Jack1”, h2 的先行词为“Sam1”。基于这样的结果, 我们可以确定当前句子的中心, 并且用做下一个句子的局部上下文。该算法的第 3 步如图 14.7 所示, 这一步实现了前面提到的中心约束理论。在前面的例子中, 这一算法的执行结果是“Jack1”被确定为新的 Cb。

对于每个具有从 S 和 R 中导出的约束 CR_x 的代词($PRO\ p\ S\ R$):

1. 构造一个可能的先行词的排序列表,其中(按顺序)包含局部上下文的 C_b 和 C_p ,句内处理中得到的可能指代对象,以及局部上下文中的其他篇章实体。
2. 删除 L 中任何一个使 CR_R 为假的指代对象 R。
3. 将($REF-LIST\ p\ L$)加入到代词逻辑形式的约束中。

图 14.6 为非反身代词确定可能先行词的算法

如果句子中不含有代词,那么新的 C_b 就是 NIL(空)。否则,执行以下步骤:

1. 构造一个包含所有代词的指代对象的列表 L。
2. 如果旧 C_b 在 L 中,那么新 $C_b =$ 旧 C_b (延续或者保持)。
3. 否则,从 L 中的旧篇章实体列表选择一个排名最靠前的实体,并把它设置为新 C_b (漂移)。

图 14.7 识别新的 C_b 的算法

14.4 定指性描述

定指性描述指向一个在上下文中可以惟一确定的对象。为了确定指代对象,必须利用四种类型的上下文中的任何一种。一个定指性描述可能单指一般或特定场景中的对象,也可能指局部或全局上下文中的对象。定指性名词短语的存在性用法和指代性用法有重要的区别。在前面,我们已经讨论了这两种解读,但需要提醒的是,存在性解读表示是确保存在惟一的满足描述的对象;指代性解读则使用描述来指代前面已经知道的对象,如名称和代词等。

对于指代性解读,听者必须确定指代对象,而不仅仅是接受指代对象的存在。例如,如果我们住在一个奶牛厂,而且事先并不知道刚好有一头牛病了,我们就不能简单地以“The cow is sick”来开始对话,除非前面的上下文告诉了我们是在谈论的是哪一头牛。作为断言有一头牛病了的一种方式(就像存在性解读一样),这个句子是不合适的。在指代性解读中,定指性描述必须指已经存在于上下文中的对象,否则这个句子被认为是有缺陷的。

需要注意的是,定指性描述不必确定完全意义上的对象,因为篇章上下文可以用于约束可能的指代对象。事实上,一个指代性的名词短语可以指一个对话双方都不确认的对象,例如:

18a. Helen bought a car_i and a boat yesterday. (Helen 昨天买了一辆车和一艘船。)

18b. She paid too much for the car_i. (她在车上的花费太高。)

即使说话者或听者不能确定是现实世界中的哪一辆车,仍然可以理解这句话的意思。定指性描述“the car”单指知识库中由前面句子引发的一个对象,即 Helen 买的那辆车。这个例子虽然说明定指性描述和代词之间有一定的不同,但实际上它们也有很大程度上的相似性。例如,在大部分情况下,定指性描述在紧邻的局部上下文中并没有先行词。另外,如第 12 章所述,定指性描述不能与同一个句子中的多种成分发生共指,相反,它们通常都指代前面的历史列表中引入的对象。

框 14.1 生成指代表达式

在自然语言生成系统中,篇章模型对于指代项的选择起着决定性作用。让我们考虑这样的自然语言生成系统,其语义表示系统中包括常数,但没有指代的模式。例如,从下面两个语义中生成句子:

1. *Think*(*W1*, *Won*(*W1*)) & *Woman*(*W1*)
2. *Think*(*W2*, *Won*(*W1*)) & *Woman*(*W1*) & *Woman*(*W2*) & *W1* ≠ *W2*

在没有先验的篇章上下文的情况下,对应语义 1,语言生成系统可能生成这样的句子“A woman thought she won”;对应语义 2,系统可能生成句子“A woman thought a woman won”。值得注意的是,第二个句子不能用来表达语义 1,因为它不符合 12.4 节中的互指约束,即名词短语“a woman”的两个实例不能指代同一个对象。

给定前面句子的上下文后,中心约束理论也能够影响句子的生成方式。如果系统由语义表达式

3. *Guess-Number*(*W1*) & *Name*(*W1*, “Jane”)

刚生成了这样的句子“Jane guessed a number”,然后系统必须生成语义 1,那么结果如何呢?根据前文中的信息,语义 1 最可能生成的句子为“She thought she won”。另一个可能的句子“Jane thought she won”也可以接受但不自然。而句子“Jane thought that Jane won”和“Jane thought that the woman won”则不正确。另一方面,如果前文为:

- 4a. Jane guessed a number.
- 4b. She picked the one Sue had suggested.

如果下一步需要实现语义 2,其中 *W2* 表示 Sue,那么句子“Sue thought she won”可以接受;而句子“She thought Jane won”就不能接受,因为它不符合中心约束理论。McKeown(1985)是将篇章约束应用到语言生成系统中的典型例子。

定指性描述可以指代前面篇章上下文中没有引入的对象。这些对象在这一篇章的某些假设场景中是惟一的。例如,常见的定指性描述“the moon”一般指围绕地球转的月球,虽然在合适的上下文中也可能有其他的卫星并用这个定指性描述来指代。定指性描述也可以指谈话者已经知道,但没有提到的对象。例如,当我向一个房子走去时,我可以让人打开门(指我们面前的这扇门)。

通过定义上下文惟一性这个概念,可以为成功的定指性指代规定约束条件。对于一个描述 D_x 、一个场景 S (包括特定场景和一般场景)和篇章上下文(包含一个局部篇章上下文 C_1, \dots, C_n 的序列),一个对象 O 在上下文中是惟一的,当且仅当:

1. 存在一个数 k , 满足:
 - a. 对所有的 $i < k$, 在 C_i 中不存在使 D_x 为真的对象 x , 并且
 - b. 在 C_k 中 O 是惟一使 D_x 为真的对象。
2. 否则, 场景 S 中存在惟一的使 D_x 为真的对象 O 。

这个定义提出了一个指代性名词短语处理算法,该算法是 14.2 节中给出的基本历史记录列表搜索算法的一种改进。给定一个形式为 $\langle \text{THE } x D_x \rangle$ 的定指性描述,其中 D_x 为表示这个描述的公式,测试局部上下文中的每一个篇章实体 r ,以检查 D_x 是否为真。如果 D_x 对唯一的一个 r 为真,那么它就是指代对象。如果发现多个 r ,那么这个描述在某种程度上是有缺陷的。如果没有满足 D_x 的篇章实体,那么,在上一个篇章上下文中重复这一过程。如果在任何局部上下文中都没有发现指代对象,那么测试 D_x 是否对 S 中的唯一一个实体为真。

和代词解释一样,指代对象的最后决定将由通用推理系统来实现。这样,前面的算法将用来提出一些可能的指代对象(具有一定的顺序)。与代词的处理方法类似,这一信息可以采用附加约束的形式添加到逻辑表达式中。

要确定复杂名词短语的指代对象,必须同时解决名词短语中的所有指代项。例如,名词短语“the cow in the field”。要确定这个短语的指代对象,首先需要确定名词短语“the field”的指代对象。但是,如果按照这个顺序查找指代对象,又有一些问题。例如,考虑这样的情形:两头母牛(Cow) $C1$ 和 $C2$,两个牧场(Field) $F1$ 和 $F2$;更进一步,母牛 $C1$ 在牧场 $F1$,母牛 $C2$ 在畜棚(Barn) $B1$ 。这一情形可以用下面的公式来描述:

$Cow(C1), Cow(C2), Field(F1), Field(F2), Barn(B1)$
 $In(C1, F1), In(C2, B1)$

名词短语“the cow in the field”很明显指 $C1$,但是短语“the field”是有歧义的,可能是 $F1$ 或 $F2$ 。这样,我们就不能为“the field”确定一个唯一的指代对象,所以整个 NP 是不可解释的。要避免这个问题,必须同时搜索这二者,以找到一个满足这个公式的对象:

$\exists c: Cow(c) \exists f: Field(f) . In(c, f)$

对这个查询,只有一个解决方案,即 c 是指 $C1$, f 是指 $F1$ 。

14.4.1 存在性解读和间接指代

存在性解读要确定由这个描述所定义的惟一对象。但是,仅仅惟一是不够的。正如例子“The cow is sick”,即使我们接受你的牧场有一头牛病了,这个句子也是有缺陷的。而存在性解读可以通过根据指代的其他对象来定义一个对象的方法引入这个对象,如:

19. The winner in the 10K race was American.

对话双方均不需要知道是谁赢得了一万米田径比赛,他们只需要知道比赛的惟一获胜者是一个美国人。在这个例子中,需要的一般知识是比赛只有一个获胜者。在其他情况下,惟一性只有根据定指性名词短语的用法得到。让我们判断一下句子“The drawer of my dresser is stuck”(我的衣柜的抽屉卡住了)和“A drawer in my dresser is stuck”(我的衣柜的一个抽屉卡住了)的不同。听者可能对“dresser”一无所知,要理解这两个句子,听者必须假定已经存在一个“dresser”。另外,给定第一个句子,听者可能推理认为“dresser”只有一个“drawer”。如果给定第二个句子,听者可能推理得到“dresser”不止一个“drawer”。这些例子说明定指性描述不总是指代篇章历史中的对象,它们也可以引入一些对象。这个过程称为适应性,因为听者通过引入对象和句子属性来适应谈话者。

这里,我们为惟一存在性定义一个新的量词:

$$\exists! x : R_x \cdot P_x$$

仅当只有一个 x 满足 R_x 并且 P_x 也为真,这个公式才为真。基于此,假定“the 10K race”指对象 *Race10K*,那么句子 19 可以转换为:

$$\exists! w : \text{Win}(w, \text{Race10K}) \\ \text{American}(w)$$

由存在性的定指性名词短语引发的篇章实体的生成与不定指性名词短语类似。惟一的区别在于指代对象是由描述来准确定义的。具体来说,句子 19“the winner in the 10K race”中的篇章实体是指篇章实体 $W1$,它是由 $\text{Win}(W1, \text{Race10K})$ 惟一定义的。这与“A runner in the 10K race was American”中的不定指性名词短语相反。其中,篇章实体 $R1$ 的定义为:

$$\text{RunIn}(R1, \text{Race10K}) \& \text{American}(R1)$$

一些存在性表达并不明显地包含可以用来确定其惟一性的指代项。在这些情况下,对象和对象间的关系必须从篇章上下文中推理得到。例如下面的篇章:

20a. My club held a raffle.(我的俱乐部举办了一次抽奖活动。)

20b. The winner won a car.(中奖者得到了一辆小汽车。)

在名词短语“the winner”中,没有补足语来说明要定义的内容。但是,如果要像句子 19 一样,给这个词语项“winner”赋予类似的语义解释,那么这个语义解释应该是:

$$\exists! w : \text{Win}(w, *PRO*)$$

其中, $*PRO*$ 是一个必须由上下文确定的回指项。我们可以通过创建一个不存在上下文惟一性的例子来给出这个分析的证据,例如:

21a. Both my club and John's club held raffles last week.

(上星期我和 John 的俱乐部都举办了抽奖活动。)

21b. * The winner won a car.(中奖者获得了一辆小汽车。)

句子 21b 要么是不合法的,因为没有合适的指代对象;要么是很蹩脚的,因为它强迫限定了一种解释,即同一个人在两场抽奖活动中都中奖了。

对于某些较少解释为其他实体的函数的词语,也有可能出现间接性指代的例子。例如,考虑下列句子:

22a. Jack brought a pencil to class.(Jack 带了一支铅笔到班上。)

22b. But he found that the lead was broken.(但他发现笔芯断了。)

问题出在句子 22b 中的名词短语“the lead”上。由直觉判断是指代 Jack 带到课堂上的铅笔的笔芯。这一类型的指代可以通过关系 R 和回指项 $*PRO*$ 来刻画。要正确解释这个名词短语,必须先要找到 pro-形式的先行词,并判定关系 $*R*$ 。这个名词短语“the lead”将映射为以下形式的存在性解读:

$$\exists! l : \text{Lead}(l) \& *R*(l, *PRO*)$$

其中, $*R*$ 和 $*PRO*$ 需要根据上下文来确定。在这个例子中,解析后的形式如下。其中,“*Pencil1*”指句子 22a 生成的篇章实体:

$\exists l: \text{Lead}(l) \ \& \ \text{SubpartOf}(l, \text{Pencil})$

需要注意的是,因为单词 lead 是有歧义的,比如铅笔芯、钓鱼坠或者一小块金属,所以要确定单词的正确词义,也需要确定这个关系。

尽管几乎不可能确定用于 $*R*$ 的一个固定的关系集合,不过还是有一些比较常用的关系。例如,部分关系就很常见,如篇章 22 中。部分关系也可以用于涵盖涉及典型容纳物的情况,如以下的句子:

When we entered the kitchen, we noticed that *the stove* had been left on.

(当我们进到厨房时,我们注意到炉子还开着。)

这里,我们必须利用这一信息,即厨房中一般都有炉子。

另一类常见的例子涉及事件和角色,如:

The day after we sold our car, *the buyer* returned and wanted his money back.

(我们卖掉汽车的第二天,买主回来想要回他的钱。)

这种情况通常可以识别出来,因为描述中使用了一个角色名词,其语义解释的逻辑形式中可能已经有了一个 pro-形式。

在利用基于框架的知识表示系统中,一种方法是允许任何槽关系都能够用于这种情况。要分析间接指代形式,系统将检查历史记录列表中的对象定义,以确定新的名词短语是否描述一个能够填充槽的对象。这一表示能够处理前面讨论的很多例子,因为部分关系可以作为槽来处理。格角色将与表示一个事件(这个事件由动词短语来描述)的框架中的槽相对应。这一技术提供了限制要搜索的可能关系的方法,但是可以看出这一技术很难处理所有的例子。如句子“*When we entered the kitchen, we saw that the gas had been left on*(当我们进入厨房的时候,我们看到煤气还开着)”。在这种情况下,名词短语“*the gas*”就是作为这个厨房的一部分的炉子所使用的煤气。可以看到,处理这样的情况需要一些非常复杂的推理能力。

◦ 14.5 定指性指代和集合

当描述中涉及到集合,或指代集合中的多个对象时,就会出现更复杂的情况。我们已经遇到指代集合的一些不定指性描述,诸如“*three men*”和“*some men*”。这些描述的定指形式是“*the three men*”和“*the men*”,可以解释为直接或间接指代。

考虑这样的例子。描述集合的名词短语中的修饰成分可能是描述集合本身的属性,也可以描述集合中每一个个体的属性。这样,句子“*The three men who left the party*”(那三个离开晚会的人)描述一个包含 3 个元素的集合,其中的每一个元素都是一个人。这一定指性名词短语的存在性解释如下所示:

$\exists M: |M|=3 \ \& \ M = \{m \mid \text{Man}(m) \ \& \ \text{Leave}(m, \text{Party})\}$

这一集合是惟一的,这个事实意味着正好有三个人离开了晚会;否则,将有几个可能的集合都适合这个描述。与不定指性描述相比,“*Three men who left the party*”的解释如下:

$\exists M: |M|=3 \ \& \ M \subset \{m \mid \text{Man}(m) \ \& \ \text{Leave}(m, \text{Party})\}$

这一集合不一定是惟一的,因为可能有三个人以上的人已经离开了晚会。

没有补足语的复数名词短语一般都涉及间接指代,常常指已经引入的集合的一个子集。例如:

23a. Some boys and girls came to the party.(一些男孩和女孩来到这个晚会。)

23b. The boys left at 8PM.(男孩们在晚上 8 点离开了。)

名词短语“The boys”描述的不是这个局部上下文中直接出现的集合,而是间接描述了一个集合,即前面提到的集合中的所有男孩的集合。要接受这样的解释,句子 23b 将转换为:

$Left(B1, 8PM)$ 其中 $B1 = \{x \mid Boy(x)\} \cap *PROSET*$

其中, $*PROSET*$ 是由上下文确定的集合。如果用 S2 表示句子 23a 中的“Some boys and girls”所生成的篇章实体,可以化简为:

$Left(B1, 8PM)$ 其中 $B1 = \{x \mid Boy(x)\} \cap S2$

考虑另一个篇章的例子:

24a. Jack, Mary, Sam and Helen went to the beach.(Jack, Mary, Sam 和 Helen 去海边。)

24b. The boys were too chicken to go in the water.(男孩们很胆小,不敢下水。)

在句子 24b 中,“The boys”指一个集合,这个集合包含 Jack 和 Sam。这是由男孩的集合和由并列名词短语所引入的集合($Jack \mid Mary \mid Sam \mid Helen \mid$)的交集定义的集合。

这种技术可用于处理很多情况,不过有时甚至连指代对象集合也没有在局部上下文中出现,这种情况下还是会有问题,如:

25a. Jack met Sam at the beach.(Jack 在海边遇见了 Sam。)

25b. The two boys then went to the store.(然后两个男孩去了商店。)

这里,句子 25a 中没有明确地谈到一个集合。一种建议是集合 {Jack Sam} 通过对这个情景的推理变为相关。这种情况下,可以利用一般知识:两个人相遇后,他们一起构成了新的集合。但是这一方法对其他例子并不奏效,如:

26a. Jack lived in San Francisco, while Sam lived in New York.

(Jack 住在旧金山,而 Sam 住在纽约。)

26b. The two boys never met.(这两个男孩从来没有遇见过。)

用一般的方法来处理这种情况会出现问题。

14.5.1 集合元素的指代

当一个集合在篇章上下文中时,描述可能是指集合中的元素。很多例子利用特别的方法来表示这一特性。例如单词“one”可以用来间接指代一个集合,如名词短语“one”和“a large one”,它们是不定指性描述词。“the one I saw”和“the large one”是定指性描述。考虑以下例子:

27a. At the zoo, a monkey scampered between two elephants.

(动物园里,一只猴子在两头大象之间来回奔跑。)

27b. One snorted at it.(一头大象对它喷了一个响鼻。)

27b'. The large one snorted at it.(较大的那头大象对它喷了一个响鼻。)

框 14.2 规划定指性描述

自然语言生成中的一个重要问题是规划定指性名词短语。语言生成系统首先要确定是利用代词形式、专有名词还是定指性描述。我们已经在框 14.1 中涉及了一些这方面的问题。如果要生成定指性描述,可能会有很多问题,因为有很多不同的方法来描述对象。给定篇章上下文后,分析这个描述是否提供了足够的信息来保证指代对象是上下文惟一的,这样,一些候选描述可以被删除。但即使有这样的约束,还是有很多的候选。另一个约束是简洁性。如果一个生成系统只是简单给出描述中所涉及对象的所有信息,生成的句子仍可能是不能理解的。生成的句子应该是简洁且上下文惟一的。例如,特定上下文中涉及两个红的盒子,一大一小。要描述那个小盒子,“the small box”就是很恰当的描述。虽然“the small red box”也很明确,但它包括了不必要的信息。而另一方面,“the red box”也不是一个很好的描述,因为它没有惟一地描述其指代对象。

在很多情况下,有很多方法来描述满足上述两个约束的对象。例如,一个知识库中包含下面的信息:

*SellEvent(s1) & Agent(s1) = M1 & Recip(s1) = J1 & Theme(s1) = P1 &
 Person(M1) & Named(M1, "Mary") & LivesIn(M1, CITY1) &
 Person(J1) & Named(J1, "John") & President(J1, COMP1) &
 Plans(P1) & Secret(P1) & Named(COMP1, "XTRA Corp")*

假定上文中已经生成句子“Mary sold the secret plans”来表示事件 *s1*,那么我们考察命题 *Recip(s1) = J1* 的可能实现:

1. *John is the president of XTRA Corp.*
2. *The buyer was the president of XTRA Corp.*
3. *The buyer was John.*

从逻辑上来说,这三种句子都表达了这些内容,也惟一确定了这些术语项。例如,句子 1 用惟一确定的名字 *John* 来表达 *Recip(s1)*,用惟一确定的描述“the president of XTRA Corp”来表达项“*J1*”。但是句子 1 不能提供足够的信息将这个句子与上文联系起来,所以不是一种很好的表达方式。句子 2 和句子 3 就好些,因为它们通过“the buyer”(其中,“the buyer”确定了事件“卖”的一个角色)与上文建立了联系。名词短语“John”和“the president of XTRA Corp”都是指 *J1*,它们看似等价而实际上不一样。至于这两种表达方式中的哪种更恰当,与说话者的目标有关系。例如,如果说话者是试图揭露工业间谍之类的活动,那么句子 3 就不是很恰当,因为它没有提供关键信息——是“Mary”把计划卖给另一家公司。Dale(1992)是生成定指性描述的系统典型例子。

如果假定它转换为下面的公式,我们可以得到“one”的合适特性。

$\exists x: x \in *PROSET* \dots$

其中, $*PROSET*$ 的指代是确定的。在句子 27b 中,先行词是句子 27a 中大象的集合,即 ELS1,句子 27b 的形式为:

$$\exists x: x \in ELS1 .$$

$$SnortAt(x, monkey1)$$

句子 27b' 中的定指性指代为间接指代, 形式为:

$$\exists! x: x \in ELS1 \ \& \ Large(x)$$

$$SnortAt(x, monkey1)$$

如果没有明确的信息说明这一指代对象是由上下文中的集合定义的, 情况会变得更加复杂。看下面的例子:

28a. Jack used two scuba tanks on his trip. (Jack 在旅途中用了两个氧气罐。)

28b. He preferred the 1600psi tank. (他喜欢 1600psi 的那个。)

在这种情况下, 篇章上下文中没有这样一个额定为 1600psi 的氧气罐对象。准确一点说, 这个名词短语从前面提到的集合中选择一个元素, 并给它赋予了一个额外的属性, 这个属性使这个对象成为惟一的。这样, 假定篇章实体“two scuba tanks”为 $TNKS1$, 句子 28b 可以表示为:

$$\exists! t: t \in TNKS1 \ \& \ Rated(t, 1600psi)$$

$$Prefer(Jack1, t)$$

因为这些情况包含了听者事先不知道的一些属性, 所以这些情况很难检测到。

14.5.2 全称量词和集合

全称量词“each”和“every”显示了一种有趣的特性。它们在句法上是单数的, 并且支持句子内的单数代词, 例如在句子“*When each boy_i saw Helen, he_i was angry*”中。但在句子间这些量词引发一些集合, 这个集合整体作为一个篇章实体。

29a. Each boy_i in the park saw Helen. (公园中每个男孩都看见了 Helen。)

29b. * He_i was angry at her. (他对她很生气。)

29b'. They_i were angry at her. (他们对她很生气。)

需要注意的是, 句子 29b 中的“he”不能指代句子 29a 中的男孩们之一, 而句子 29b' 中的“*They*”可以指代 29a 中的男孩们。所以, 由“*Each boy in the park*”所引发的篇章实体是集合 $\{x \mid Boy(x) \ \& \ In(x, Park1)\}$ 。

这些名词短语经常涉及对一个集合的隐含的定指性指代, 这个集合限制了量词的范围。例如, 孤立地看, 句子“*Each girl saw Helen*”中为“*each girl*”所生成的篇章实体为集合 $\{x \mid Girl(x)\}$ 。不过, 这种解释几乎不可能, 因为它宣称每一个女孩都看见了 Helen。相反, “*each girl*”的范围通常是由上下文定义的, 如在以下篇章中:

30a. Several girls arrived at the party. (一些女孩来到晚会。)

30b. Each girl saw Helen. (每个女孩都看见了 Helen。)

在处理句子 30a 时, 名词短语“*several girls*”将创建一个常量 $G1$, 即女孩的集合。将 $G1$ 放在句子 30b 中的局部上下文中, “*each girl*”很自然地解释为 $G1$ 中的所有女孩, 句子 30b 可以初始转换为:

$$\forall g: g \in G1 . See1(g, Helen1)$$

这一分析将名词短语“each girl”解释为“each of the girls”,将“every girl”解释为“every one of the girls”。女孩的集合是根据指代决定的,然后作为量词的范围。在其他例子中,集合用间接指代来定义。例如,在“Several boys and girls arrived at the party”所产生的上下文中来考虑句子 30b,其中篇章实体 *BG1* 是为男孩和女孩所引发的集合而创建的。在这种情况下,与句子 23b 类似,范围是由集合的交集创建的。句子 30b 可以初始转换为:

$$\begin{aligned} \exists! G3 : G3 &= \{x \mid Girl(x)\} \cap BG1 \\ \forall g : g \in G3 &. See1(g, Helen1) \end{aligned}$$

14.6 省略

正如 14.1 节所介绍,省略主要涉及一些句法不完全的句子。这些省略通常可以从前面的子句得到补充和恢复。省略常与连词一起出现,例如:

31. Helen saw the movie and Mary did too. (Helen 看了电影, Mary 也看了。)

其中,我们可以知道 Mary 也看了电影。其他的情况则涉及句子对,例如:

32a. Some think that Jack will win the race next week. (有些人认为 Jack 将赢得下周的赛跑。)

32b. But he never will. (但他从不这么认为。)

重要的是,在问答系统这一应用场合中,省略可能出现在一系列的问句中。例如在下面的 A 和 B 进行对话的场合中:

33a. A: Did Sam find the bananas? (Sam 找到那个香蕉了吗?)

33b. B: Yes. (是的。)

33c. A: The peach? (那个桃子呢?)

这里, A 的第二个句子只有在 A 的第一个句子的基础上才可以理解,即理解为“Did Sam find the peach?”的省略形式。

框 14.3 非 NP 引发对象的指代

本章中关于指代解释的所有例子都涉及由名词短语引发的先行词。有很多例子指代这个句子中的其他对象。在大多数情况下,指代表达式中的内容或选择性约束都明确指明指代对象是某一种类型。一般来说,从局部上下文中就可以找到被指代对象。例如,我们要分析的例子是紧随句子“Jack went to New Orleans last year”(Jack 去年去了 New Orleans)的下文。

可以像下面的句子一样,指代前面的这个事件:

The trip changed his life. (这次旅行改变了他的生活。)

It changed his life. (它改变了他的生活。)

也可以指代 Jack 所完成的行为,如下面的句子:

He does *that* every year. (他每年都这么做。)

Sam did *it* last year as well. (Sam 去年也这么做了。)

你还可以指代“Jack 去过 New Orleans”这样的事实,如下面的句子:

That really surprised Helen. (这确实让 Helen 吃惊。)

你也可以用特殊的代词“then”和“there”来指代事件的时间和地点,如下面的句子:

Mardi Gras was on *then*. (那时 Mardi Gras 也在。)

He loves to go *there*. (他喜欢去那儿。)

目前,有两种办法来处理这些情况。可以扩充篇章实体的集合以包括任何行为、事件、时间、地点或子句中隐含描述的命题,也可以开发从局部上下文中推导出所指代对象的技术。由于表层形式和选择性约束往往表明了这种指代表达式的作用,所以这两种方法是同样有效的。

14.6.1 关于省略的句法约束

对省略进行分析的一个基本假设是,省略子句的完整句法结构与前面子句的结构具有对应关系。一旦这两个子句之间的对应关系确定了,前面子句的语义解释可以利用省略子句所包含的新信息来补充,从而可以产生新的解释。例如句子 31,初始解释在图 14.8 中。比较这两个句子的句法结构,可以发现这两个主语“Helen”和“Mary”有对应关系。这就决定了下面的语义形式的转换。首先,抽取第一个子句的语义形式中的“*Helen1*”,结果如下:

$\lambda p\ Seel(p, Movie24)$

然后,将这一形式应用于新的信息,即解释 Mary,这样就产生了省略子句的语义形式:

Seel(Maryl, Movie24)

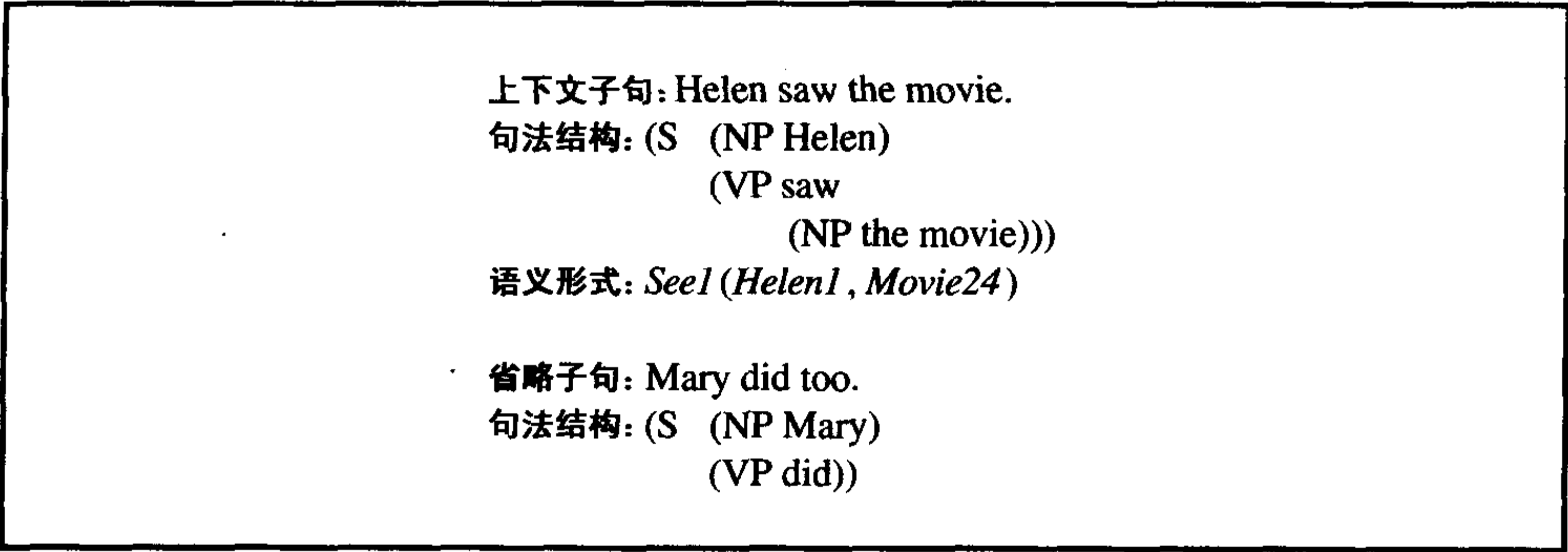


图 14.8 动词短语省略的一个简单例子

这一技术对于更复杂的情况也是有效的。例如,文献中经常提及的一个经典问题就涉及如何处理下面篇章中的歧义:

- 34a. Jack kissed his wife. (Jack 吻了他的妻子。)
- 34b. Sam did too. (Sam 也是。)

句子 34b 可以指 Sam 吻了 Jack 的妻子(称为误读)或者是指 Sam 吻了自己的妻子。句法结构的匹配过程如前所述,可以发现 Jack 和 Sam 的对应关系。假定句子 34a 的语义形式是:

Kiss1(Jack1, wifeOf(Jack1))

要生成句子 34b 的语义形式,这一语义形式需要被抽象。但是,从这个表达式中抽象 Jack1 的方法有两种,相对应产生如下两种形式:

$\lambda p \text{ Kiss1}(p, \text{wifeOf}(\text{Jack1}))$ 和
 $\lambda p \text{ Kiss1}(p, \text{wifeOf}(p))$

将 Sam1 应用到第一个形式中,结果为 *Kiss1(Sam1, wifeOf(Jack1))*;将 Sam1 应用到第二个形式中,产生了第二个结果 *Kiss1(Sam1, wifeOf(Sam1))*。从常识来说,这两种方式都是可以解释的。

14.6.2 基于语法的算法

这一方法的症结在于寻找这两个子句之间的结构对应关系,它受句法因素的影响很大。该算法的输入是局部上下文(前面一个句子)的语法结构和省略语句的部分句法结构。需要注意的是,虽然不一定能够得到完全的分析,但是自底向上句法分析器还是能够产生分析结果的。

对一些类型的结构(如 VP 省略),对应关系是非常明确的。如果一个动词短语包含助动词“do”(但没有补足语)和修饰语(如 too, as well 和 also),则预示着动词短语的省略,互相对应的是两个句子中的主语。对于其他情况,寻找对应关系就更有问题。例如,考虑对话 33,省略子句中的 NP 和前面子句中的 NP 对应。

潜在的对对应关系可以通过搜索一个从输入片段中推导出的模式来发现。为了尽可能发现句法结构上平行的片段,用输入片段中包含除内容词(如名词和形容词)外所有信息的模式来搜索,功能词(如介词和冠词)将保留。如果这一匹配没有产生任何潜在的目标片段,可以用更一般的模式(例如可以删除数的限制,删除特定冠词、介词,删除修饰语,等等)来匹配。考虑对话 33 的处理。这个算法的输入如图 14.9 所示。

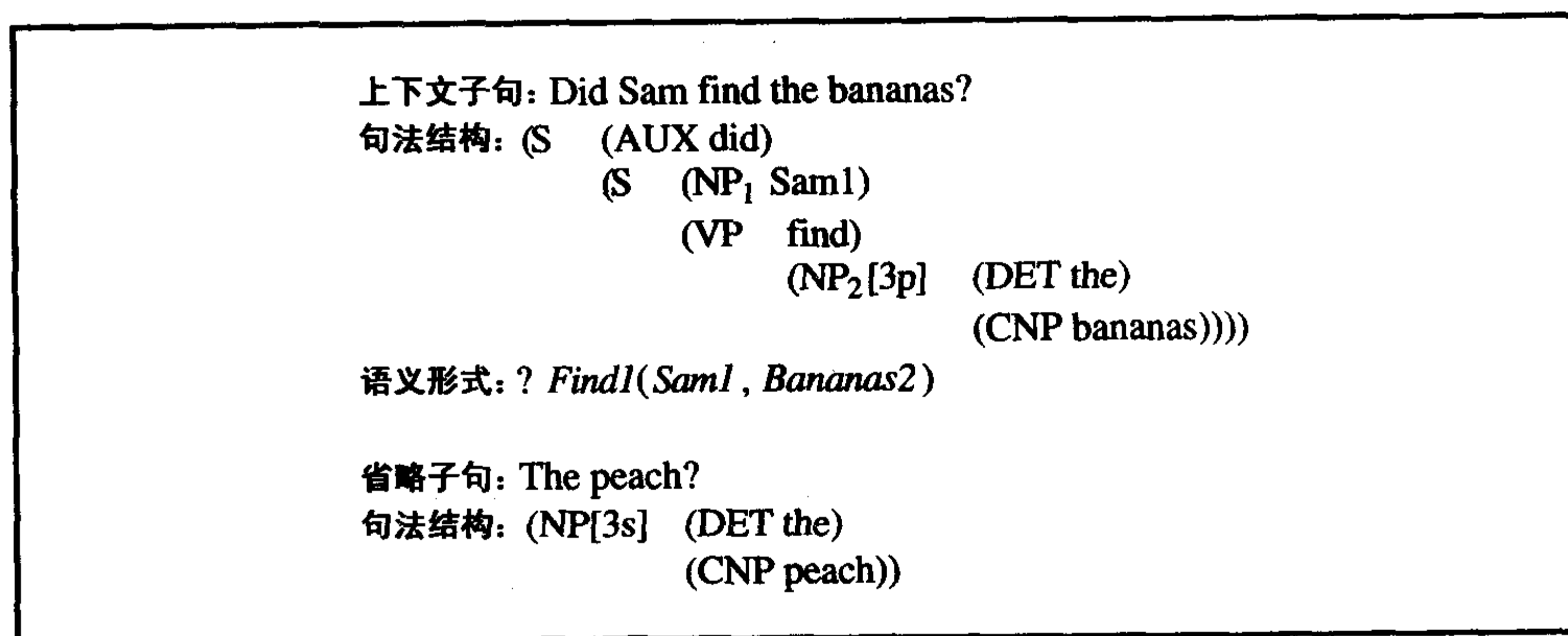


图 14.9 省略的例子

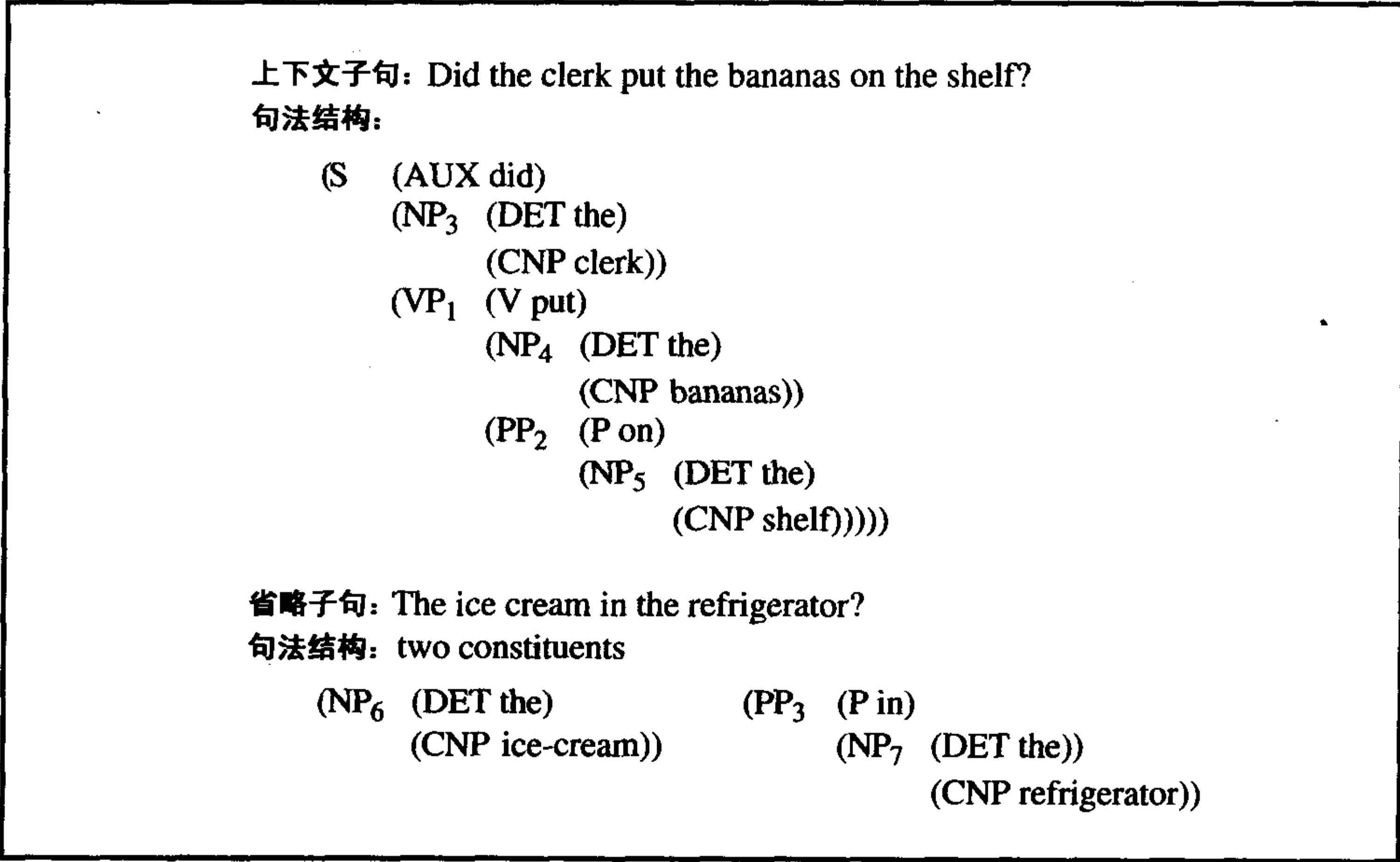
初始模式是(NP[3s](DET the)(CNP))。搜索匹配的上下文子句的句法结构,没有发现对应的内容。如果放宽模式中数的约束,即(NP(DET the)(CNP)),就能够发现香蕉(the bananas)

和桃子(the peach)之间的匹配,也可以推导出语义形式。如果模式匹配不上,可以继续放宽约束,尝试模式(NP(DET)(CNP));如果失败,再继续尝试(NP)。可以定义一系列逐步放宽的模式,第一个匹配的模式将抓住最重要的结构属性。

包含一系列成分的省略形式更复杂。考虑以下的例子:

- 35a. A: Did the clerk put the bananas on the shelf? (职员把香蕉放到架子上了吗?)
- 35b. B: Yes.(是的。)
- 35c. A: The ice cream in the refrigerator? (把冰激凌放到冰箱里了吗?)

句子 35c 的正确解释涉及两个独立的成分而不是单个的成分,即名词短语“the ice cream”和介词短语“in the refrigerator”(这将匹配为动词“put”的补语的一部分)。可以扩展模式匹配过程以便搜索每一个片段,同时,要满足额外的约束:输入中不同成分的顺序与问题中目标成分的顺序一样。另外,目标片段的序列应该是同一个成分的子部分。例如,在对话 35 中,A 的初始句法形式、输入片段以及由输入片段生成的模式序列如图 14.10 所示。



A: Did the clerk put the ice cream in the refrigerator? (职员是否把冰激凌放到冰箱里了?)

B: No.(没有。)

A: The TV dinners? (电视快餐呢?)

这种情况下,根据输入片段生成的模式能够匹配名词短语“the clerk”,“the ice cream”和“the refrigerator”。我们只能利用语义信息来选择“the ice cream”作为合适的目标片段。类似地,如果是“The manager?”,则可以推断“the clerk”作为合适的目标片段;如果是“The freezer?”,可以推断出“the refrigerator”作为目标片段。

由前面的例子可以总结出,计算输入片段和目标片段的语义相似度,然后选择最相似的候选目标片段。当考虑词义消歧时,第 10 章介绍的方法完全可行。例如,假定系统有图 14.11 所示的分类体系。

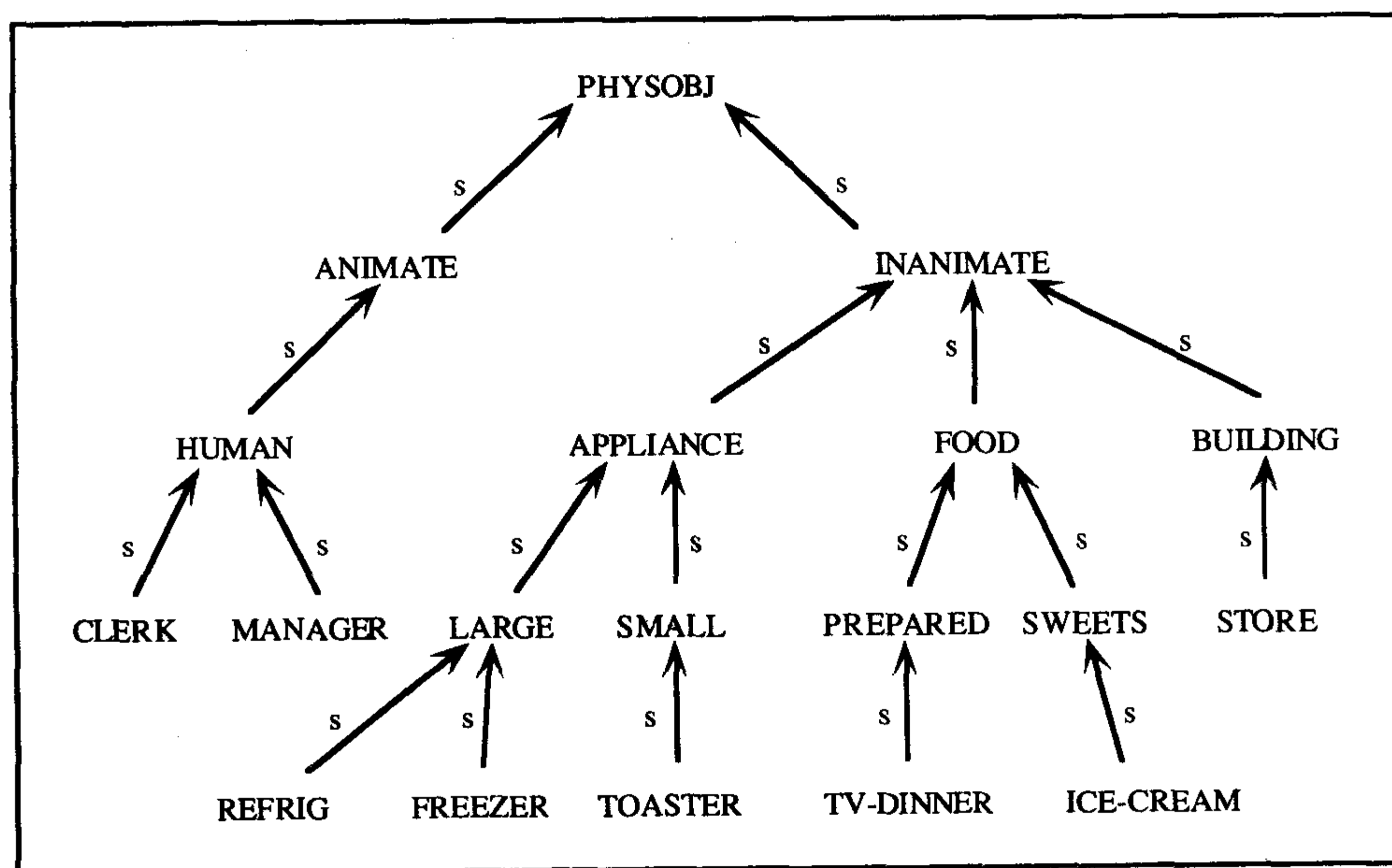


图 14.11 类型的层次体系

可以采用最简单的相似度计算方法来计算两个语义之间的距离。考虑输入片段是“The TV dinners?”, 候选目标片段是“the clerk”, “the ice cream”和“the refrigerator”。可以通过计算层次体系中的两个节点之间的步数来估计它们之间的语义相似度。结果如下所示:

TV-DINNER 与 CLERK: 7(通过 PHYSOBJ)

TV-DINNER 与 ICE-CREAM: 4(通过 FOOD)

TV-DINNER 与 REFRIG: 6(通过 INANIMATE)

这样,“the ice cream”是首选目标片段,我们得到了合适的分析。

当然,这样根据语义相似性来做推断并不一定完全正确。如果语义解释器不能很好地分析结构,可以尝试其他的可能。考虑下面的对话:

A: Did you see the clerk in the store? (你在商店里看到那个职员了吗?)

B: Yes.(是的。)

A: The toaster oven? (烤箱呢?)

这一情况有下面的语义相似校验:

TOASTER 与 CLERK: 7(通过 PHYSOBJ)

TOASTER 与 STORE: 5(通过 INANIMATE)

这样,语义相似性检查将“the store”作为目标片段,由此可以解释为“Did you see the clerk in the toaster oven?”我们希望语义解释规则足够丰富,以便说明这种解释是行不通的。然后,处理省略的算法可以给出下一个可能的候选,将“the clerk”作为目标片段,构建的解释为“Did you see the toaster oven in the store?”

14.7 表层回指

存在一类指代表达式,它们引入了与篇章上下文中的对象相关的新对象。这些情况称为表层的回指,之所以这样称呼是因为它们主要是根据前面句子中的表层信息来消除歧义的。与之相对应,我们前面介绍的例子主要是指篇章上下文中的对象,称之为深层指代。表层回指的例子如下:

36a. Tell me John's grade in CSC271.

36b'. Give me *it* in MTH444 as well.

36b''. Give me *Mike's* in MTH444 too.

句子 36b' 和 36b'' 中的对象在句子 36a 中并没有涉及。但是,句子 36a 中的上下文可以让我们知道,后面两句指的是其他科目的分数(分别为 John 的 MTH444 分数和 Mike 的 MTH444 分数)。

需要注意的是,存在特定的句法信息来说明表层回指的出现,其中一大类是缺少中心词的名词短语,例如“Mike's in MTH444”。另外一类是有补足语的代词,例如“it in MTH444”。另外,也有很多情况涉及副词的情况,如“too”,“also”,“as well”,它们明确说明了与篇章上下文中的某些对象具有一定的相似性。

研究人员已经提出了一些方法来处理表层的回指问题。方法之一是利用句法/语义结构来处理省略,其基本想法是表层的回指短语一般是不完整的成分,处理过程如下。首先,搜索前面句子的句法树以发现结构匹配的成分,然后,将前面句子的信息与当前句子的新信息合并在一起,建立新的短语;之后分析新短语,即产生这个句子的解释。例如,对句子 36b'' 中缺少中心词的短语,我们用这项技术对其进行分析。短语“Mike's in MTH444”可能产生下面的名词短语成分:

```
(NP (DET (NP (NAME m1 "Mike"))
      (s))
      (CNP (CNP *missing*)
            (PP (P in)
                  (NP (NAME m2 "MTH444"))))))
```

这个成分用来生成上一节描述的一个模式,然后用这个模式与前面句子的所有子成分匹配。在篇章 36 中,只有一个成分能够匹配成功,即“NP John's grade in CSC271”。一旦指定了这个成分,就将它复制一份,并用这个新信息来代替旧信息。新的名词短语如下所示:


```

(NP (DET (NP (NAME m1 "Mike")))
    (s))
(CNP (CNP GRADE1)
    (PP (P in)
        (NP (NAME m2 "MTH444")))))

```

可以将这个新的名词短语插入到当前分析树中,这样会产生正确的解释。同样,可以处理利用介词的情况,即用介词补上丢失的成分。

基于句法结构的方法存在的问题是,这些方法对短语的结构很敏感。例如,给定句子 36a 生成的上下文,前面介绍的方法对句子“Give me it for Mike as well”就不能正确处理,因为这种方法不能发现所有格限定词“John’s”和介词短语补语“for Mike”之间的对应关系。

其他方法一般是基于语义的。这种方法假定丢失的信息可以通过前面提及的一些集合来补充。为此,必须允许对涉及的集合进行一定的抽象化。句子 36a 将生产一个 John 的 CSC271 课程成绩的集合,即 $\{g \mid \text{Gradg}(g, \text{John}, \text{CSC271})\}$ 。这可以抽象为任何人的成绩,即 $\{g \mid \exists p: \text{Person}(p): \text{Grade}(g, p, \text{CSC271})\}$,或者 John 的任何科目的成绩 $\{g \mid \exists c: \text{Course}(c). \text{Grade}(g, \text{John}, c)\}$,或者任何人的任何科目的成绩 $\{g \mid \exists p: \text{Person}(p) \exists c: \text{Course}(c). \text{Grade}(g, p, c)\}$ 。表层的回指将从这些集合中选出其中之一,并基于短语中的新信息将其加入到必要的属性中。为此,需要一种方法将其他的修饰成分结合到这个集合中。例如,再考虑“Give me it for Mike as well”。一种合适的抽象结果是课程 CSC271 的成绩的集合,即 $\{g \mid \exists p: \text{Person}(p) \text{Grade}(g, p, \text{CSC271})\}$ 。现在,我们必须把修饰语“for Mike”解释为限制这个人必须是“Mike”。但是,如果没有利用句子 36a 中的词汇信息,即句子 36a 中的中心词是“grade”,就不清楚这个过程到底是怎样的。所以,在这种情况下,纯粹基于集合抽象的方法是很难行得通的。

14.8 小结

在篇章理解中,前面子句对后面子句的解释有很大影响。前面子句的信息为后面子句确定了局部篇章上下文。局部上下文的一个重要部分是由这个子句引发的篇章实体的集合,它们是后面子句中那些代词的最可能的先行词。由单数名词短语引发的篇章实体表示个体,而由复数名词短语引发的篇章实体表示集合。在全称量词的范围内,单数名词短语也引发一个集合,即由全称量词变量的每一个值生成的所有对象的集合。

历史记录列表是由前面子句生成的局部上下文的序列。发现定指性名词短语(包括代词)的先行词的最简单方法是遍历历史记录列表,寻找第一个(或最近的)满足约束条件的篇章实体。局部篇章上下文内代词的先行词的倾向性受篇章中心的影响。如果一个句子包含至少一个代词,那么其中一个代词必须指代中心。用来限制中心怎样从一个句子转移到另一个句子的约束条件确定了代词的优先考虑的解释。

局部上下文也包含前面子句的句法和语义结构,这对于解释省略和表层的回指是很重要的。这些语言现象可以通过基于模式匹配的方法发现前面子句和当前子句中片段的句法结构的相似性来处理。一旦在前面子句中发现了对应部分,就可以把省略句中的信息代到前面子句的结构中,然后可以解释这个修正后的结构,从而构建新的解释。

14.9 相关工作与深入阅读材料

早期的自然语言处理系统利用线性历史记录列表和启发式技术来处理常见的代词和动词短语的省略问题[例如 Winograd(1972)和 Woods(1977)]。这些系统主要依赖语义过滤来确定最可能的指代对象。Webber(1983)第一个从计算的角度研究了一个句子所引发的篇章实体,其研究重点是全局范围内的定指性和不定指性短语。14.1 节的内容是基于 Webber 的工作,Ayuso(1989)又对这一工作进行了改进。

语言中关于回指的很多工作都集中于处理句子内约束的情况。也有一些研究是集中于句子间的情况,其中主要介绍了篇章实体的概念[例如,Karttunen(1976)称之为篇章指导对象]。在形式语言模型中采用这些想法的最有影响的工作是 Kamp(1981)和 Heim(1982)。Kamp 的篇章表示理论(DRT, discourse representation theory)采用篇章实体为代词提供统一解释,无论这些代词表现为有界回指(bound anaphora)、句子内回指还是句子间回指。这一工作对后续的语言学和计算语言学的研究产生了巨大影响[例如 Harper(1992)]。

Sidner(1983)提出了焦点(focus)和焦点对代词解释的影响的计算模型。其基本假设是基于论旨角色的启发式优先信息来选择焦点。一般来说,最优先考虑的篇章实体被选做代词的先行词,除非这样做会引起一些语义或语用上的冲突。这一模型发展成为定指性名词短语的中心模型(centering model)[由 Grosz, Joshi 和 Weinstein(1983)提出]。基于这个中心模型激发了很多研究工作,14.3 节的内容是基于中心约束改进的算法[Brennan, Friedman 和 Pollard(1987)]。Walker(1989)对 Brennan 等人提出的算法与基于句法结构和最近上文的 Hobbs 算法(Hobbs, 1978)进行了比较。比较的结果很有意思,因为 Hobbs 算法总是倾向于句子内的解读,而中心理论总是倾向于句子间的基于中心的解读。这两种算法的性能相似,都能够识别书面文本中 90% 的先行词。但是对于对话而言,则只能正确处理 50%。

在语言学中有很多不同的理论,刻画了与篇章焦点特性相关的概念。这些理论一般将一个句子中的信息分为两部分:包含新信息的部分和提供背景信息的部分。一些有影响的理论对话题和评论的区别、主题和韵律的区别、已知信息和新信息的区别做了阐述[例如, Halliday(1967), Chafe(1975), Clark 和 Haviland(1977), Prince(1981)]。但是这些理论间有一个很重要的不同点,计算模型通常基于一种说话者的认知模型,也就是说,所谓中心,是指影响推理的注意力焦点;而在语言学中,同样的术语可能指一个句子的结构属性。在计算模型和语言学中,同一词汇的意义很多时候是一致的,但有时候会截然不同。

在哲学和语言学中,定指性描述的研究非常广泛。Russell 和 Whitehead(1925)介绍了定指性描述的概念,它是作为存在性运算符的特例。Donellan(1966)介绍了定指性描述在指示性(referential)用法和归属性(attributive)用法中的不同。Lewis(1979)和 Stalnaker(1974)提出了适应性(accommodation)的概念,它是 Kamp(1981)和 Heim(1982)中的工作的核心模块。要了解更多的信息,请参考 McCawley(1993)或 Chierchia 和 McConnell-Ginet(1990)。大部分的计算系统利用本章介绍的线性历史记录列表来处理定指性描述,并允许指代知识库中惟一描述的对象,例如 Winograd(1972)和 Woods(1977)。Grosz(1977)研究表明,在大范围的篇章中,线性历史记录列表是不足以处理很多情况的。这方面的问题将在第 16 章中讨论。

很多计算模型都采用了对省略进行基于句法的分析。其中,一旦发现相应的结构,就建立新的句法结构,然后重新进行语义解释。很多系统也采用基于语义相似的技术,特别是那些为数据库提供自然语言接口的系统[其中,省略很常见,例如 Hendrix 等(1978), Bates 等(1986)和 Grosz 等(1987)]。14.6 节对省略的分析主要介绍了这方面的工作,并且主要采用了 Dalrymple, Shieber 和 Pereira(1991)中描述的方法。

LUNAR 系统(Woods, 1977)是第一个研究表层回指的计算系统。对于这些情况经常出现的问答系统来说,这些技术是很重要的。Webber(1983)较详细地分析了这些现象,并描述了怎样处理这些情况。Hankamer 和 Sag(1976)分析了深层回指和表层回指的不同,并论证区分它们之间的不同对于处理很多语言现象都是必要的。

14.10 习题

1. 【中】给出下面两个句子引发的篇章实体及其语义形式。对每一个句子,假定前面的篇章介绍了比赛 R1 的背景知识。

In every race, the winner was American.

In every race, some runner was American.

2. 【中】句子“Two boys ate three pizzas”有两个可能的解释:“The two boys together ate three pizzas”或者“the two boys each ate three pizzas”。对每一种解释,指定产生的篇章实体和句子的意义。并请分析出现的任何问题。
3. 【中】利用中心理论来解释为什么篇章 a 是一致的,而篇章 b 则难于理解。
 - a1. Sue got up late.
 - a2. Helen didn't have time to drive her to school.
 - a3. She had to walk.
 - b1. Helen had to be at work early.
 - b2. She didn't have to time drive Sue to school.
 - b3. She had to walk.
4. 【中】对下面篇章中的每一个句子,给出其中心和优先考虑的下一个中心。基于 14.3 节描述的 4 个约束和优先关系,阐述采用哪个中心理论来预测每一个代词的指代。基于你的分析,把每一个转移分为下面几类中的一种:中心的延续、保持或转移。讨论预测的解释是否与你的直觉一致。如果不一致,哪一个约束或优先关系可能导致这个问题?
 - a. Sue got up late the day that Helen was supposed to drive her to school.
 - b. So she had to walk.
 - c. Mary saw her walking down the street.
 - d. She stopped to pick her up.
5. 【中】在 12.6 节中,有研究人员提出利用关系名词来解决语义分析的问题。请说明这一表示机制是怎样与历史记录列表机制一起来确定名词短语的指代对象的。具体来说,

对于下文中的每一个名词短语,给出合理的逻辑表达式和历史记录列表的当前状态,并讨论确定指代的过程。

The museum received a large grant from a local corporation.

The donation will allow the recipient to stay open through the summer.

6. 【中】要处理由并列名词短语产生的集合子集的指代问题,一个建议是将这个集合的所有子集都作为篇章实体,例如句子:

Jack, Sam, and Helen went to the beach.

将生成集合 $\{Jack1\}$, $\{Sam1\}$, $\{Helen1\}$, $\{Jack1\ Sam1\}$, $\{Jack1\ Helen1\}$, $\{Sam1\ Helen1\}$ 和 $\{Jack1\ Sam1\ Helen1\}$ 。现在,给定句子:

The boys were too chicken to go in the water.

合适的指代对象已经被定义为一个篇章实体。那么,这是解决这个问题的合理而又通用的方法吗?请与 14.5 节介绍的方法进行比较,并阐述这种方法和该方法相比更好或更坏的理由。

7. 【中】对下面的每一个句子给出其语法结构,并详细说明第二个句子中的指代问题是怎样解决的。

I bought a ticket for the early show on Tuesday.

Jack bought one for the late show.

8. 【中】利用语义倾向性技术和图 14.11 中的层次体系,请详细描述第三个句子的解释是怎样构建的?讨论所产生的结果。

A: Did Jack give the toaster to the manager?

B: No.

C: The clerk?

第 15 章 使用世界知识

如果不考虑所有说话者共享的关于世界的日常知识,语言就不可能被理解。前面的章节已经涉及了一些通用的世界知识。例如,关于谓词参数关系的类型约束的知识可以用于支持第 11 章中介绍的词义消歧。但是,这些技术都非常有限,并且没有将语言与语言所描述的真实世界中不断变化的情境联系起来。这一章,将着重阐述通用的世界知识是怎样用于解释语言的。

15.1 节讨论了连贯性的概念,这个概念为本章所讨论的这些技术提供了研究的动机和理由。15.2 节描述了使用世界知识的一般框架,引入了期望的概念。其中,期望生成于前面的篇章。如果一个句子与期望相匹配,则匹配的结果就是在这个篇章的前后文之间建立了一种联系,使得整个篇章呈现出较好的连贯性。15.3 节进一步完善了这个想法,表明定指性指代中的一些问题也可以在这个框架内处理。其后的几节探讨了生成期望的一些方法。15.4 节讨论了使用行为和因果关系的知识来生成期望。15.5 节讨论基于脚本的方法,其中系统是由大规模的行为描述所驱动的,该行为描述刻画了这个篇章所讨论的不同情境。15.6 节和 15.7 节描述了基于规划的技术。其中,因果关系和意图的知识可以用来生成期望。15.8 节进一步介绍了有关使用 agent 的问题求解行为知识来生成期望的一些更复杂的问题。

15.1 使用世界知识:保持前后连贯性

在本书所探讨的很多例子中,一个句子的解释最后要取决于上下文的处理。本章尝试描述使这个过程自动化的一些技术。然而,这需要对当前上下文中一个句子的含义有很好的理解。很明显,在逻辑的连贯性和表意性之间有一定的联系。在任何上下文中,前后不连贯的解释都没有意义。但经常有这种情况,有很多逻辑上连贯的解释,其中只有一部分是有意义的。所以,连贯性更多是指表意性而不是逻辑的连贯性。某些解释看上去是连贯的,而另外一些解释则完全不连贯。

如果一个篇章中的一个句子与其他句子的关系很容易被确定,那么这一篇章就是连贯的。如果一个篇章由不相关的句子组成,它就是不连贯的。要理解一个篇章的意义,必须确定每一个句子是怎样与其他句子相关联的,以及每一个句子是怎样与整个篇章相关联的。正是在这一连贯性假设的基础上,解释才能进行下去。例如下面的例子:

1a. Jack took out a match. (Jack 取出一根火柴。)

1b. He lit a candle. (他点燃了一根蜡烛。)

根据句子 1a 和 1b 连贯性的假设,我们可以得到关于句子 1b 的一些结论,这些结论可以分为以下几类:

指代——“He”指的是 Jack1(由中心理论得到)

歧义消解——“lit”指点燃而不是照亮

隐含意义——点燃的手段是利用句子 1a 中的“match”(火柴)

需要注意的是,要确定句子 1a 和 1b 是怎样联系起来的,必须知道点燃蜡烛需要火柴。正是这一常识使我们能够理解句子之间的联系。在其他情况下,句子之间也许没有明显的关系,但是连贯性的假设仍然施加了一定的联系,例如:

2a. John took out a match. (John 取出一根火柴。)

2b. The sun set. (太阳落山了。)

虽然句子 2a 和 2b 没有明显的联系,但事实上它们之间存在一种时间上的关系,即太阳落山前(或落山时)Jack 拿出了火柴。虽然这两个句子之间只有很微弱的联系,但足以对句子 2a 和 2b 所描述的内容建立前后连贯的情境。

需要注意的是,很多来自于连贯性假设的结论是隐含的,而不是必然的。因此,这些结论有可能被后续的讨论所推翻,因此是可废止的。这样,要得到这些结论,其推理过程将会很复杂。本章探讨的技术都是基于这样一种做法,将当前句子可能的解释与前面篇章生成的期望进行匹配。下一节将详细讨论这个问题。

15.2 与期望进行匹配

我们假定,由一个篇章所创建的特定场景都可以表示为前面一些句子的内容,以及在解释前面这些句子时所进行的推理。这一信息可以用来生成一系列关于下文所要描述的可能要发生事件的期望。后面几节将探讨使用不同的方法来生成期望。这一节将探讨,在已经生成这些期望的前提下,将所有可能解释的问题与期望进行匹配的问题。

更严格来说,这个问题可形式化地表述如下:给定一些可能的期望 E_1, \dots, E_n 和这个句子的一些解释 I_1, \dots, I_k , 确定那些能够匹配的 E_i, I_j 对的集合。如果存在惟一的期望/解释对 E_i/I_j , 那么这个句子的解释就是 E_i 和 I_j 匹配的结果。如果有多个可能的匹配,则需要其他一些过程来确定哪一个匹配是最恰当的。

考虑篇章 1, 根据后面讨论的方法, 句子 1a 支持的期望是, Jack 将用火柴点燃某件东西。使用基于事件的逻辑, 这一期望将表述为事件 E1:

期望 1:

$LIGHT1(E1) \& Agent(E1) = Jack1 \& Instrument(E1) = Match33$

其中, *Match33* 是句子 1a 中的不定指性名词短语“a match”引入的新常数。句子 1b 有两种解释:(1)某人点燃了蜡烛;(2)某人照亮了蜡烛(如,他打开电灯把房子照亮了,蜡烛也被灯光照亮了)。这里假定“he”是指 *Jack1*。这两个不同的解释可以用两个事件 *E2* 和 *E3* 来刻画:

解释 1:

$LIGHT1(E2) \& Agent(E2) = Jack1 \& Theme(E2) = Candle1$

解释 2:

$ILLUMINATE1(E3) \& Agent(E3) = Jack1 \& Theme(E3) = Candle1$

这个常数“*Candle1*”是为不定指性名词短语“a candle”引入的。

匹配过程应该以某种方式使用期望来选择第一个解释,从而得出结论:Jack 用火柴点燃了蜡烛。以下部分将描述定义匹配过程的两种方法。首先,我们要确定,使用简单的演绎推理方

法是不够的。采用演绎推理方法,有两种做法可以用来选择一个合适的解释。我们可以看是否一个解释可以从一个期望得到证明,或者一个解释可以用来证明一个期望。但是这两种方法对篇章 1 都是无效的。

15.2.1 尝试 1:从期望中证明解释

这一方法将寻找蕴涵一个解释 I_j 的期望 E_i 。但是这一方法的能力非常有限,不能用于处理篇章 1,因为期望 1 不蕴涵任何解释。具体来说,这个期望没有对论题“*Candle1*”做出任何声明。实际上,因为“*Candle1*”是句子 1b 引入的 Skolem,所以没有期望能够预测其用途。

15.2.2 尝试 2:从解释中证明期望

这一方法正好与前面的方法相反。其目标是选择一个解释 I_j ,以保证存在一个期望 E_i ,使 I_j 蕴涵这个 E_i 。这种方法也不是很有效,因为要求输入的句子能够完全包含期望中的所有信息,所以这个期望没有贡献任何其他的信息。将这个方法应用到篇章 1 上也是无效的。具体来说,解释 1 不蕴涵期望 1,这是因为解释 1 不蕴涵这样的期望:采用的工具是句子 1a 中的火柴。

15.2.3 基于等价性的技术

尝试 1 和尝试 2 存在的问题是解释和期望都只包含各自的信息,因此都不蕴涵对方。我们之所以采用非单调的等价性技术,是因为我们直觉上认为,如果存在一些连贯的等价性假设以使 E_i 蕴涵 I_j ,那么这两个公式就是匹配的。例如篇章 1 中,加上两个等价性假设,这个期望就可以蕴涵解释 1:

期望: $LIGHT1(E1) \ \& \ Agent(E1) = Jack1 \ \& \ Instrument(E1) = Match33$
 假设: $Theme(E1) = Candle1 \ \& \ E1 = E2$

与之对照,没有等价性断言能够使得期望 1 蕴涵解释 2。

这一技术可以通过图匹配算法来实现如下。每一公式都表示为一个语义网络图,其中语义网络图的根节点表示这个事件。篇章 1 中的期望和两个解释可以表示为图 15.1 中的示图。图匹配算法如下:对任何节点 N , $Label(N)$ 表示标记这个节点 N 的项, $T(N)$ 表示节点的类型(可以沿着 isa 链找到); $LL(N)$ 表示从 N 节点出发的所有链接。对任意边 A , $END(A)$ 表示边 A 指向的节点。那么,当且仅当满足下面的三个条件时,节点 $N1$ 和节点 $N2$ 才匹配:

1. $T(N1)$ 和 $T(N2)$ 相同,或者一个是另一个的子类。
2. 没有明确说明 $Label(N1)$ 和 $Label(N2)$ 是指代不同的对象。
3. 对于 $LL(N1)$ 中的所有链接 $L1$:
 - a. 不存在从 $N2$ 出发,标记也为 $L1$ 的链接,或者
 - b. 存在一个从 $N2$ 发出的相同标记的链接 $L2$,且 $END(L1)$ 与 $END(L2)$ 能够递归地匹配。

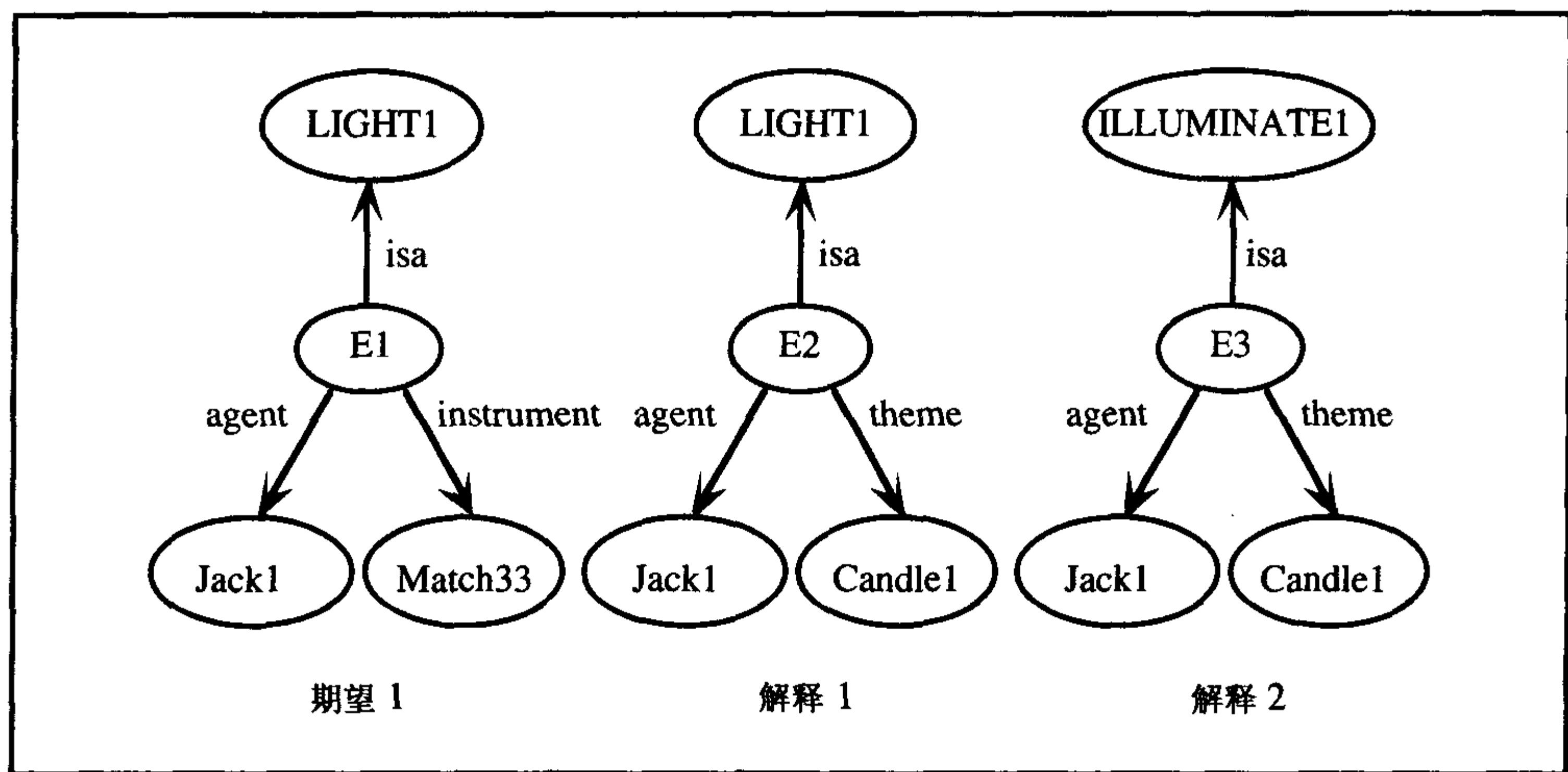


图 15.1 期望和当前句子的两种解释

这个算法与基于图的合一算法是一样的,区别仅在于我们是根据节点类型和等价性信息来判断节点是否匹配,而不是根据合一的特征列表。如果两幅图匹配,结果就是将所有匹配的节点整合在一起的图。例如,对于节点 E1 和节点 E3,因为类型不一致,所以互不匹配。而节点 E1 和节点 E2 能够匹配,合并以后的结果在图 15.2 中显示。这是句子 1b 所期望的解释。

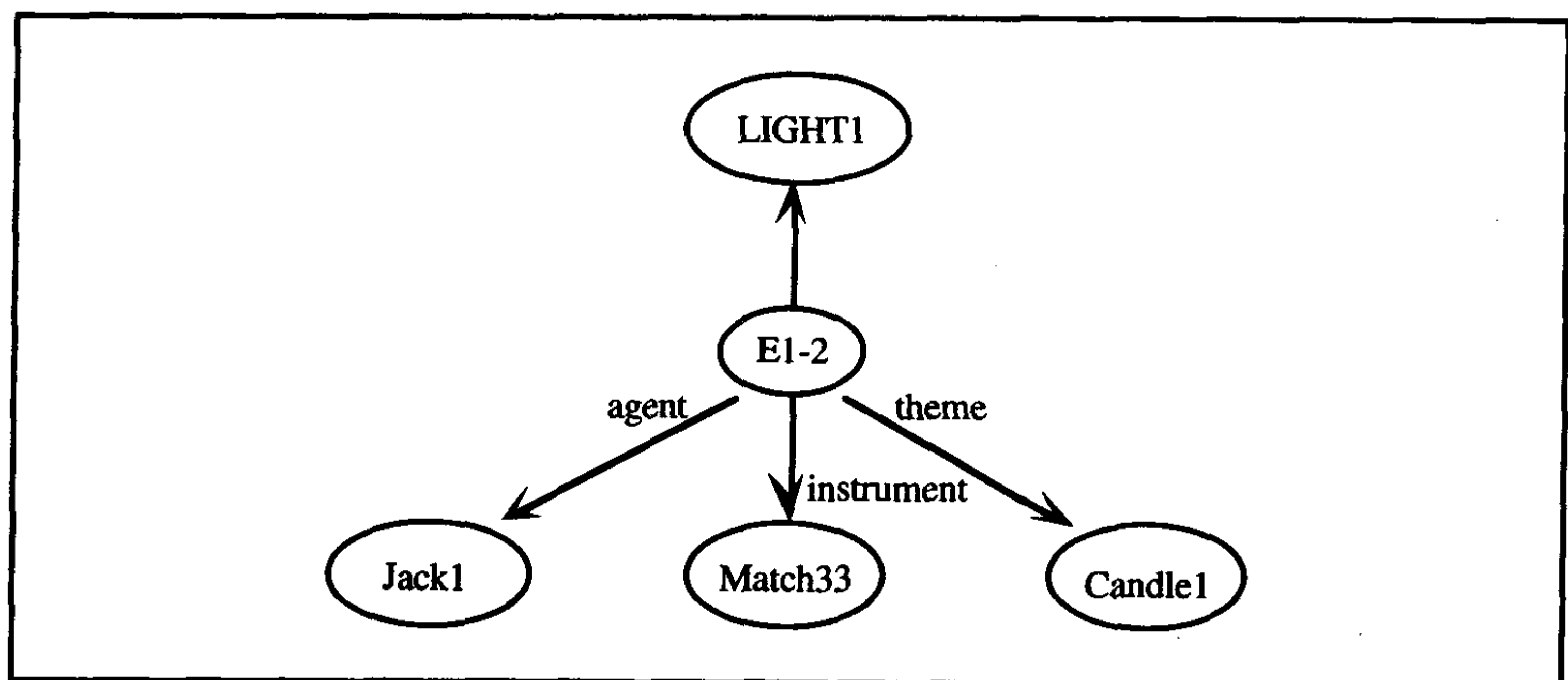


图 15.2 期望 E1 和期望 E2 匹配的结果

需要注意的是,这一方法并不完全可靠,因为我们不知道每一对匹配的节点是否不同。但是,新生成的图从整体上来说可能是不连贯的。假定一个知识库由三个常数 *Jack1*, *Sam1* 和 *Person1* 构成,其中 $Jack1 \neq Sam1$ 。图 15.3 中的两幅图对应于一个期望(即某人给自己刮脸)和一个解释(即 Jack 帮 Sam 刮脸)。匹配算法将成功匹配 E4 和 E5,这是因为不知道 *Person1* 和 *Jack1* 是否是不同的人,也不知道 *Person1* 和 *Sam1* 是否是不同的人。最后的结果将蕴涵 $Jack1 = Sam1$,而事实上不是这样的。为了处理这种情况,需要验证算法能否确保结果图的全局连贯性。

本质上,图匹配算法的实质,就是对两个事件是否可能相同进行启发式的验证。例如,对期望 E1 和期望 E2 进行匹配,就是基于类型信息和明确的区别性信息来验证 $E1 = E2$ 是否可

能成立的一种方法。通过检查所有发出的边,涉及 E1 和 E2 的格关系也得到了检查。虽然这一过程并不能保证匹配的全局连贯性,但可以发现最常见的不连贯的情况。事实上,因为一阶谓词演算的公式集合中这些公式的连贯性是不可确定的,所以可能根本就不存在保证结果连贯性的通用算法。但是,大部分知识表示系统提供了检查事实一致性的启发式工具。给定这样的知识表示系统,我们可以通过使用下面的方法,检查两个常量 C1 和 C2 的等价性是否可能成立:

1. 寻找知识库中涉及 C2 的所有命题。
2. 对于前面的每一个命题,用 C1 代替 C2,得到新的命题,然后将新命题加到知识库中。
3. 如果结果知识库在知识表示的限定条件下是连贯的,那么说明 $C1 = C2$ 可以接受。

当然,这一操作的时间代价可能很高。事实上,由于匹配的很多常数将是新的 Skolem 常数,因此对其相关信息不需要了解很多。在这种情况下,只需对少量命题进行连贯性检查。

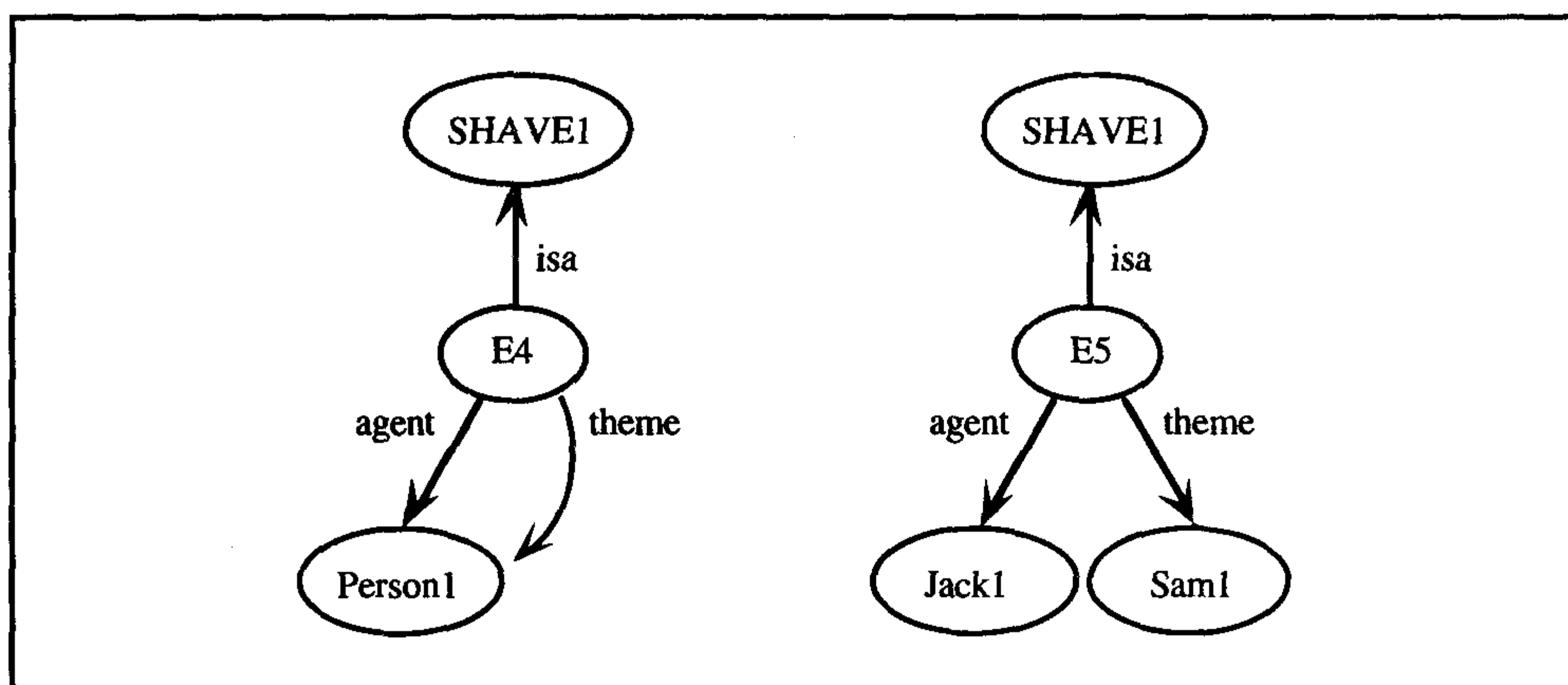


图 15.3 一个有问题的例子

15.2.4 基于溯因推理的技术

基于溯因推理的技术建立在一种直觉之上,即必须寻找一个解释,用其说明当前句子是正确的(符合逻辑的)。也就是说,必须使用特定的情境和期望来构建证据以证明当前句子是正确的。为了构造证据,需要做一些从知识库中不能直接得到的假设。

这一方法可以用更加形式化的方式描述如下。给定特定的情境 S (包括期望)和可能的解释 I_1, \dots, I_k ,对每一个解释 I_j ,我们分析对于每个 I_j ,必须将哪些假设 A_j 加到 S 中才能确保:

1. S 和 A_j 蕴涵 I_j ,
2. S 和 A_j 是连贯的。

选择需要的假设 A_j 的数目最少的那个解释 I_j 。当然,这需要有一些方法对假设进行度量,以便可以对最小集合进行定义。一般来说,简单计算每一个 A_j 中不同原子命题个数的方法并不会起作用,因为一些命题出现的次数可能比另外一些命题出现的次数少得多。

溯因推理理论是一种很泛化的方法,因此,如果不对假设定义某种度量,就不可能用于特定的用途。这种方法涵盖了一些更加专有的技术,如前面定义的基于等价性的技术。为了说

明这一点,可以利用下面的约束,将前面基于等价性的方法重新表示为溯因推理方法:

3. 对于单一的期望,惟一允许的假设就是等价性假设。
4. 优先选择具有最少等价性假设数量的那个假设 A_j 。

对于句子 1b,加入到期望 1 中的最小假设集合为 $E1 = E2$ 和 $\text{Theme}(E1) = \text{Candle1}$ 。这两个假设和期望 1 蕴涵解释 1。

很显然,如果能够增加其他类型的假设,使用溯因推理技术还可以完成更多的任务。不过,用这种方法对匹配进行形式化已经足够了。

15.3 指代和匹配期望

前面一节中的匹配算法假定知识库中的定指性名词短语的指代问题已经解决。在这一节中,我们把这一假设放宽,使用期望匹配算法来确定指代。这个方法还是比较简单的,大致如下:如果对于每一个定指性名词短语都生成一个新的 Skolem 常数,那么,当解释与期望进行匹配时,需要一个等价性假设来使这个匹配成立。在这种情况下,等价性假设就确定了指代对象。

再考虑句子 1b,这一次我们假定不知道代词所指代的对象。生成的 Skolem 常数为 $H1$,第一个解释为:

解释 1: $\text{LIGHT1}(E2) \ \& \ \text{Agent}(E2) = H1 \ \& \ \text{Theme}(E2) = \text{Candle1}$

解释 1 和期望匹配的过程如前所述,不同的是这次有 3 个节点合并成功,分别对应于下面的 3 个等价性假设: $E1 = E2$, $\text{Theme}(E1) = \text{Candle1}$ 和 $H1 = \text{Jack1}$ 。这样,在匹配的过程中就确定了代词的指代对象。在这个例子中,指代对象与给定中心约束得到的优先考虑的解释完全相同。

下面的两个例子说明了通用推理方法是怎样扩充中心技术的:

3a. Jack poisoned Sam. (Jack 对 Sam 下了毒。)

3b. He died within a week. (他一周内死了。)

在这种情况下,句子 3b 的最自然的解释是 Sam 死了。但另一方面,如果句子 3a 的下文是这样的:

4a. Jack poisoned Sam. (Jack 对 Sam 下了毒。)

4b. He was arrested within a week. (他一周内就被逮捕了。)

句子 4b 的最自然的解释就是 Jack 被判有罪。对于句子 3b 和 4b,中心理论或者预测它们有相同的先行词,或者对这两种情况无法做出选择。而期望匹配算法能够确定正确的指代。例如,根据“Jack poisoned Sam”这一信息,可以生成下面的这些期望:

期望 1: $\text{DIE}(E4) \ \& \ \text{Theme}(E4) = \text{Sam1}$

期望 2: $\text{IS-ILL}(E5) \ \& \ \text{Theme}(E5) = \text{Sam1}$

期望 3: $\text{ARREST}(E6) \ \& \ \text{Theme}(E6) = \text{Jack1}$

句子 3b(忽略了时态的表达)的解释为:

$$DIE(E7) \& Theme(E7) = H1$$

E7 将与期望 1 中的 E4 匹配,这样 H1 将指“Sam1”。而句子 4b 的解释为:

$$ARREST(E8) \& Theme(E8) = H2$$

E8 将与期望 3 中的 E6 匹配,那么 H2 将指“Jack1”。

这一方法也适用于定指性名词短语。例如,考虑下面的篇章:

5a. Jack poisoned Sam.(Jack 对 Sam 下了毒。)

5b. The villain was arrested within a week.(这个坏人在一周内就被逮捕了。)

虽然事先无法知道“Jack was a villain”(Jack 是坏人),相同的技术也有效。句子 5b 可以解释为:

$$ARREST(E9) \& Theme(E9) = V1 \& Villain(V1)$$

给定这样的知识:villain(坏人)是指一种类型的人,并且 Jack 是一个人,在 $V1 = Jack1$ 的假设下,这个解释将与期望 3 匹配。因此,我们就可以得到结果,Jack 是一个坏人。

为此,期望匹配算法可以是解决指代问题的有力工具,但是它不能代替前面介绍的指代消解的其他技术。举例来说,即使只使用语义信息就能确定明确的指代对象时,局部上下文的约束可能也是重要的。回忆一下以前的例子:

6. Jack walked over to the table and drank the wine. It was brown and round.

(Jack 走到桌边,喝了葡萄酒。它是棕色的,圆的。)

即使桌子(“table”)是惟一一个可以是棕色的并且是圆形的对象,这个句子也是很别扭并且有缺陷的。给定期望后仍有歧义的情况下,来自于局部上下文的优先关系也是很重要的,例如:

7a. Jack poisoned the man from whom George stole the jewels.

(Jack 对那个曾经被 George 偷过宝石的人下了毒。)

7b. He was arrested within a week.(在一周内,他就被逮捕了。)

在这种情况下,被逮捕的对象(即期望)可能是 Jack 和 George,因为他们两人都涉嫌非法活动。基于期望匹配算法, $H3 = Jack1$ 和 $H3 = George1$ 的可能性是相同的;而基于中心理论,则更倾向于前一个对象(即 $H3 = Jack1$)。通过分析句子 7a 的解释,可以看到来自于结构优先的更多证据,这种证据跟期望没有什么关系,这些证据改变了倾向的解释:

8a. George stole the jewels from the man that Jack poisoned.

(George 从 Jack 曾经下过毒的那个人处偷了宝石。)

8b. He was arrested within a week.(在一周内,他就被逮捕了。)

在这种情况下,被逮捕的是指 George,这与中心约束是一致的。

要把约束与期望匹配过程结合起来,我们只需要简单地增加下面的条件:如果有多个解释与期望匹配,那么将选择与来自于局部上下文的优先关系最一致的那个解释。

期望匹配是一种非常有效的方法,但是这种方法的有效程度完全取决于生成期望的过程。我们将在本章的后面部分讨论这个问题。

15.4 使用关于行为和因果关系的知识

很多篇章都涉及行为和因果关系。这一节将讨论下面两个问题:(1)行为的表示;(2)怎样使用这一信息来生成期望。

在深入探讨这些问题之前,我们首先考虑怎样定义行为(action)。几千年来,这个问题一直困扰着哲学家,所以这里不可能给出一个简单的答案,但我们还是可以澄清一些问题。首先,有意识的主体所执行的“action”(行为)的意义与物理学中的“action”(作用)的意义是不同的。在物理学中,力和惯性的概念为“action”(作用)提供了合理的理论。在这里,我们将主要讨论“action”的前一个意义(即“行为”)。

虽然先前我们把行为和事件分开来,但在这里我们关心的正是那些由正在执行的行为构成的事件。我们将这个事件简称为行为,虽然更准确地说,应该称之为行为执行的事件。

为了确定可以用来生成期望的因果关系,了解这些行为是很重要的。例如,如果你被告知 Jack 向商店走去(“Jack walked to the store”),那么你可能推测他将要在那个商店内(是这一行为的直接结果),并且可能买一些东西(是由关于正常意图性行为的常识所建议的一个行为)。要理解一个篇章,我们需要很多类型的因果关系,主要包括下面两类与状态有关的因果关系:

结果关系——每一个行为都有一些由此行为引起的结果,在行为结束后或进行中,这些结果常常继续保持一段时间。我们可以将结果分为两类,期望的结果(采取这些行为所希望得到的结果)和附带的结果(由这些行为导致的,但不是期望的结果)。如果我们尝试执行某一行为,但是失败了,它仍然可能有附带的结果。若没有出现我们所期望的结果,那么这个行为就是失败的。

前提关系——每一个行为都有其发生的一些条件,只有在这个行为开始前(或发生过程中)这些条件都被满足,这个行为才有机会成功。

生成期望的两个行为之间也有一些很重要的关系,主要包括:

使能关系——如果第一个行为的结果是第二个行为的前提条件,那么第一个行为就使第二个行为具备可行性。更常见的情况是,第一个行为只为第二个行为建立了一部分前提条件,其他条件是通过其他手段产生的。

分解关系——一个行为只是另一个行为的子部分或子步骤。例如,为了完成进入汽车这一行为,需要开锁、开门、进入。每一行为都是进入汽车的子步骤,如果这三个行为都顺利完成,进入汽车的行为才完成。

生成关系——如果在一定条件下执行第一个行为会导致执行第二个行为,我们称一个行为生成了另一个行为。例如,我的房间有供电系统和灯泡,那么打开开关这个行为将生成把灯打开的行为。有些研究人员将生成关系作为分解关系的一种退化情况来对待,即这个分解关系只有一个子步骤。

存在一些自然语言结构能够明确地确定每一种关系。例如,连词“by”可以表示生成关系(如“I turned on the light by flipping the switch”中)或使能关系(如“I bought some milk by going to the store”中)。连词“in order to”也表明了这两个关系,例如“I flipped the switch in order to turn on the light”和“I went to the store in order to buy some milk”。当然,动词“enable”和“cause”也表明了这些

关系,例如“The explosion caused the window to break”和“Being at the store earlier enabled me to buy many items on sale”这两个句子。

框 15.1 定义因果关系

有几个常识性的概念都提及了一个行为是怎样与其他行为和周围的环境联系起来的。它们一般都属于因果关系的范畴。如果一个行为导致出现了某种结果,那么这个行为与结果之间就可能有一定的因果关系。从直觉来说,因果关系可以表示为如果一个行为没有发生,这个结果也不会出现。虽然看似直观,但根据这几个简单的概念来定义因果关系却很困难。例如,你可能试图用一阶谓词演算中的原子形式来刻画,把玻璃杯扔在地上,它被摔碎了:

$DROP(Sam, Glass) \supset BROKEN(Glass)$

但是,如果 Sam 没有把玻璃杯扔在地上,该公式也为真。这是因为,给定逻辑隐含的定义,一个为假的条件也可能使隐含为真。但是,如果通过规定前提必须为真(即 Sam 确实把玻璃杯扔在地上)来扩充这个条件,那么它逻辑上等价于下面的合取公式:

$DROP(Sam, Glass) \& BROKEN(Glass)$

这一公式还是没有刻画出因果关系。因此,你可能会继续扩充信息:如果 Sam 没有把玻璃杯扔在地上,它就不会摔碎。这样就更接近因果的定义,但又引入了新的问题。首先,前面的表述称为反事实的陈述,这种陈述对于“如果把实际为真的某件事认为是假的,世界将会怎样”做了断言。一阶谓词演算刻画不了这种陈述,它需要模态算子和更复杂的语义模型(Lewis, 1973)。即使这样,还是不能准确定义因果关系的概念,因为存在这样的情况:不是 Sam 把杯子扔在地上,而是发生别的事件导致这个杯子碎了(例如,因为某人向 Sam 扔石头导致他把杯子掉在地上;或者不是 Sam 把杯子扔在地上,而是别人扔的石子把杯子打碎了)。由于这些问题,很多理论都把因果关系当做原语,而不试图进一步分解。

为了利用这些关系,知识表示系统必须能够表示行为和因果关系的一般知识。大部分的知识表示都围绕表示一个行为发生的每一个谓词来组织这种信息。图 15.4 表示了两个行为(BUY 和 PURCHASE-TICKET)的定义。行为“BUY”的定义与第 13 章的定义相同,其中“角色(role)”定义了行为中涉及的重要对象;“约束(constraint)”描述了角色取值的典型属性;“前提(precondition)”表示了行为发生前所必须具备的条件;“结果(effect)”表明了行为发生后状态的变化;“分解(decomposition)”表明用一系列的子行为来完成一个行为的典型方式。图中的行为定义详细描述了这几个不同层次的信息。“BUY”是一个通用行为,买的对象可以是任何物品,并且给出了钱和物品交换的细节。这一行为发生的前提是买方要有钱,卖方有物品要卖,行为的结果是钱和对象的所有者发生了变化。这一行为是通过两个“GIVE”行为来实现的。行为“PURCHASE-TICKET”描述了在售票处买票的特殊情形,并且只提供了交易的重要细节;“PURCHASE-TICKET”将“BUY”作为其子步骤之一。

```

行为类 BUY(e):
角色: Buyer, Seller, Object, Money
约束: Human(Buyer), SalesAgent(Seller), IsObject(Object),
      Value(Money, Price(Object))
前提条件: AT(Buyer, Loc(Seller))
           OWNS(Buyer, Money)
           OWNS(Seller, Object)
结果: ¬OWNS(Buyer, Money)
      ¬OWNS(Seller, Object)
      OWNS(Buyer, Object)
      OWNS(Seller, Money)
分解: GIVE(Buyer, Seller, Money)
      GIVE(Seller, Buyer, Object)

行为类 PURCHASE-TICKET(e):
角色: Agent, Clerk, Ticket, Booth, Money, Station
约束: Human(Agent), Clerk(Clerk), IsTicket(Ticket),
      TicketBooth(Booth), At(Clerk, Booth),
      Value(Money, Price(Ticket)), In(Booth, Station)
前提条件: OWNS(Agent, Money)
结果: OWNS(Agent, Ticket)
分解: GOTO(Agent, Booth)
      BUY(Agent, Clerk, Ticket)

```

图 15.4 BUY 和 PURCHASE-TICKET 的定义

我们有好几种方法来利用这一信息。首先,给定一个句子,描述一个行为的发生,前提和结果将给出这个句子的一些隐含信息。例如,给定句子“Jack bought a stereo at the mall”(Jack 在购物商场买了一台音响),那么,系统能够得到下面的信息:Jack 有一台音响,不过他已经没有了(买音响的)钱;这家商店曾经拥有一台音响,但现在也没有了。这些隐含的信息可以作为期望,因为说话者下一步可能会涉及这些条件中的某一个。但是,基于结果或前提的期望常常很明显,它们对于后续句子的解释并没有多大帮助。例如:

9a. Jack bought a stereo at the mall. (Jack 在购物商场买了一台音响。)

9b. He now owns it. (现在他拥有了这台音响。)

在这个例子中,句子 9b 是句子 9a 的很自然的结果。所以句子 9b 在一定程度上是多余的。

由“buy”行为触发的行为或“buy”作为其组成部分的那些行为,可以生成其他期望。潜在地,如果知识库中的一个行为的前提与“buy”行为的结果匹配,那么这个行为就是一个可能的期望。例如:

10a. Jack bought a stereo at the mall. (Jack 在购物商场买了一台音响。)

10b. Now he can disturb his neighbors late at night. (现在,他可以在深夜里打扰邻居们了。)

句子 10a 和句子 10b 之间的联系如下:买了一台音响就使 Jack 把音响的音量调到很大成为可能,然后就可能生成打扰邻居的行为。很显然,买音响这个行为可能引发很多行为。但是,我们不可能预先生成所有的期望。下面几节将介绍控制期望生成的一些方法。

15.5 脚本:理解固定模式的情境

控制期望生成的一个方法是储存一种大的信息单元,这种信息单元通常称为脚本。脚本给了我们感兴趣的领域内的常见情境和情节。我们不是使用前面的因果推理原则来生成期望,而是用脚本来表示期望。为了有效地控制推理,脚本必须涉及很多来自不同主体的行为,并且描述了这些行为之间的因果联系。用目前实现的行为表示方法足以表示脚本。图 15.5 显示了一个“TRAVEL-BY-TRAIN”行为(或脚本),描述一位乘客乘火车的典型场景,包括去火车站、买票、去相应站台和上火车等行为。这个脚本也包括大量的约束,这些约束详细描述了这些对象之间的关系。

TRAVEL-BY-TRAIN(e):

角色 : Actor, Clerk, SourceCity, DestCity, Train, Station, Booth, Ticket, Money

约束 : Person(Actor), Person(Clerk), City(SourceCity), City(DestCity),
TrainStation(Station), TicketBooth(Booth), In(Station, SourceCity),
In(Booth, Station), At(Clerk, Booth), DepartCity(Train, SourceCity),
Destination(Train, DestCity), TicketFor(Ticket, SourceCity, DestCity)
Value(Money, Price(Ticket))

前提条件 : Owns(Actor, Money)
At(Actor, SourceCity)

结果 : ¬Owns(Actor, Money)
¬At(Actor, SourceCity)
At(Actor, DestCity)

分解 : GoTo(Actor, Station)
Purchase-Ticket(Actor, Clerk, Ticket, Station)
GoTo(Actor, Loc(Train))
GetOn(Actor, Train)
Travel(Train, SourceCity, DestCity)
Arrive(Train, DestCity)
GetOff(Actor, Train)

图 15.5 乘火车旅行的脚本

在自然语言理解中使用基于脚本的知识会出现两个主要的问题。第一个问题是脚本选择,即这个系统怎么知道哪个脚本是相关的?第二个问题是确定脚本中的哪个部分是当前正在描述的内容。这一问题与剧作家使用剧本排戏有很大程度的相似性。首先,必须识别目前是哪一场戏,然后,必须找到与正在进行的表演对应的剧本中的剧情发展情况。我们称之为“当前点”(now point),因为它从脚本的角度表明了当前时间。

根据句子描述内容的不同,可以用不同的方法将句子与脚本结构进行匹配。例如,描述目标的句子可以用来确定相关的脚本,描述行为的句子用来更新脚本的进程。具体说来,如果没有任何脚本在进展中,我们有三种方法来引入这样一个脚本:

1. 把行为描述为目标,如在句子“Jack wanted to take the train to Rochester”(Jack 想要乘火车去 Rochester)中。在这些情况下,可以用期望的行为来命名这个脚本。
2. 把状态描述为目标,如在句子“Jack needed to be in Rochester”(Jack 要赶到 Rochester)中。在这些情况下,句子内容与脚本的结果相匹配,由此发现能够达到期望状态的脚本。

3. 描述进行中的行为,如在句子“Jack walked up to the ticket booth”(Jack 步行去售票口)中。在这些情况下,句子内容与脚本的分解相匹配以发现这一行为是实现目标的一个步骤。

一旦选择了一个脚本,当给定行为的描述时,这个脚本会更新,如在句子“Jack walked up to the ticket booth”(Jack 步行去售票口)中。在这种情况下,脚本的“当前点”会相应改变到下一个行为。

15.5.1 脚本和角色的实例化

现在,我们讨论脚本“TRAVEL-BY-TRAIN”是怎样用来分析下面的故事片段的:

11a. Sam wanted to take a train to Rochester. (Sam 想乘火车去 Rochester。)

11b. He purchased a ticket at the station. (他在火车站买了一张票。)

为了处理句子 11a,系统要能够认识到一个具有“agent wants action”形式的句子是要表述一个行为的目标——在这个例子中,目标为乘火车。假定具有合适参数的动词“take”可以用于确定“TRAVEL-BY-TRAIN”是相关的脚本,则可以创建脚本的实例 $T1$ 来描述当前情况。让我们详细分析这个例子。句子 11a 中描述的目标可以表示为:

$$\text{Travel}(E1) \ \& \ \text{Agent}(E1) = \text{Sam1} \ \& \ \text{Instrument}(E1) = \text{TR1} \ \& \\ \text{IsTrain}(\text{TR1}) \ \& \ \text{Dest}(\text{TR1}, \text{ROC})$$

将这个目标与 $T1$ 匹配。使用动词格角色和脚本角色之间的对应关系,“TRAVEL-BY-TRAIN”行为的一个实例将产生下面的等价性假设:

$$\begin{aligned} \text{Actor}(T1) &= \text{Agent}(E1) = \text{Sam1} \\ \text{Train}(T1) &= \text{Instrument}(E1) = \text{TR1} \\ \text{DestCity}(T1) &= \text{ROC} \end{aligned}$$

使用这些假设对 $T1$ 进行分解得到的结果如图 15.6 所示。现在,系统可以使用实例化的脚本来解释句子 11b。我们要寻找一些等价性假设以便这些分解的子步骤能够与新句子的语义内容相匹配。这种情况下,需要的等价性假设如下:

$$\begin{aligned} \text{PurchaseTicket}(E2) \ \& \ \text{Agent}(E2) = H1 \ \& \ \text{Theme}(E2) = \text{TIC3} \ \& \\ \text{At-Loc}(E2) &= \text{STAT3} \ \& \ \text{Ticket}(\text{TIC3}) \ \& \ \text{Station}(\text{STAT3}) \end{aligned}$$

其中, TIC3 是为“a ticket”而生成的新常数, $H1$ 是为代词“he”而生成的常数。这将与 $T1$ 分解中的第二步相匹配,用基于事件的形式表示如下:

$$\begin{aligned} \text{PurchaseTicket}(P1) \ \& \ \text{Agent}(P1) = \text{Sam1} \ \& \ \text{Clerk}(P1) = \text{Clerk}(T1) \ \& \\ \text{Ticket}(P1) &= \text{Ticket}(T1) \ \& \ \text{Station}(P1) = \text{Station}(T1) \end{aligned}$$

这一匹配将产生下面的等价性假设:

$$\begin{aligned} H1 &= \text{Sam1} \\ \text{TIC3} &= \text{Ticket}(P1) = \text{Ticket}(T1) \\ \text{STAT3} &= \text{Station}(P1) = \text{Station}(T1) \end{aligned}$$

匹配的结果是代词“he”指代“Sam1”,定指性描述“the station”的惟一指代对象被确定为指代“ $\text{Station}(T1)$ ”[即使前面的篇章并没有涉及 $\text{Station}(T1)$]。另外,可以从为脚本定义的这些约束中得到大量的隐含信息。我们可以得出火车站位于 Sam 将要离开的城市、句子 11b 中的车

票是指去 Rochester 的车票、Sam 去了火车站;我们还会有一些期望,包括 Sam 上火车以及旅行中涉及的其他行为。

```

GoTo(Sam1, Station(T1))
PurchaseTicket(Sam1, Clerk(T1), Ticket(T1), Station(T1))
GoTo(Sam1, Loc(TR1))
GetOn(Sam1, TR1)
Travel(TR1, SourceCity(T1), ROC)
Arrive(TR1, ROC)
GetOff(Sam1, TR1)

```

图 15.6 给定句子 11a, T1 的分解步骤的实例化

很明显,有必要对脚本结构进行一些泛化,以处理更真实的情况。脚本的分解结构有必要进行扩展,使之能够接受这些子行为之间的偏序关系。另一个重要的扩展是在脚本中要允许表示在不同的行为之间进行选择这种情况。这样,就可以使脚本根据行为执行过程中状态的不同来执行不同的行为路线。一些篇章在解释为什么执行一些行为或为什么一些行为成功时明确需要进行选择。例如,进入房间的脚本可能取决于门是否锁了而具有不同的行为顺序。如果门锁了,那么 agent 必须有钥匙来开门。即使具有这些概括能力,基于脚本的方法只能在篇章中描述的行为可以提前列举出来的那种有限的领域内有效。

15.6 使用层次化的规划

只有篇章按照脚本预测的方向发展,这个脚本才能成功刻画这个篇章的内容。要刻画该篇章中不同行为之间的未预料到的联系,需要更复杂的框架模型。基于规划的模型可以用来解决前面的不足。在该模型中,行为的表示方法不变,但是该模型定义的行为比脚本中的行为粒度要小。一个规划(plan)就是一系列的行为,这些行为通过等价性断言和因果关系相互关联,即如果执行某些行为将会达到某个目标。一个目标是一个状态(即 agent 试图使其为真)或 agent 想执行的一个行为。人工智能中典型的规划问题是将一个目标和可能行为的集合作为输入,产生一个试图达到预期目标的行为的序列。语言理解中的规划问题所需要的推理是不同的,需要根据其他 agent 的行为对其规划进行推断。这个过程一般称为规划识别或规划推理。规划推理的输入就是 agent 所追求的目标列表和行为的集合,其任务是建立一个规划,使所有行为按照一种有利于达到其中某个目标的方式来执行。通过强制规定所有行为与有限多个目标或单个目标相关联,基于规划的模型可以限制所生成的期望的数目。

基于脚本的系统可以看做一个简单的规划系统。每个脚本定义了一个目标和达到这个目标的方法。规划推理任务则要求选择与输入相匹配的合适的规划。而基于规划的系统则不局限于从定义好的规划中选择某一个规划,而是可以使用因果推理的方法在该领域中定义的这些行为中来建立新的规划。这样,基于规划的系统就有能力刻画更丰富的篇章内容。

我们先从一种称为层次化规划的简单规划开始讨论。层次化规划是从知识库中定义的一系列行为(称为行为库)中建立起来的。每一个行为定义的最重要的内容是描述其各个分解步骤之间的关系,因为这些关系定义了行为之间的层次关系。这个行为库中的一个子集被确定

为可能的目标。一个层次化规划定义为行为的一棵层次树,其中,根节点是可能的目标,父节点/子节点之间的关系定义了一种分解关系。一个规划所产生的结果就是这个层次树中每一个行为定义所导致的结果、将这个规划与输入所进行的匹配以及输入的行为与其他行为的分解步骤的匹配这两项所产生的等价性关系。图 15.7 给出了一个简单的层次化规划,使用它将“TRAVEL-BY-TRAIN”和“PURCHASE-TICKET”行为结合在一起,并且假设“TRAVEL-BY-TRAIN”行为是一个可行的目标。

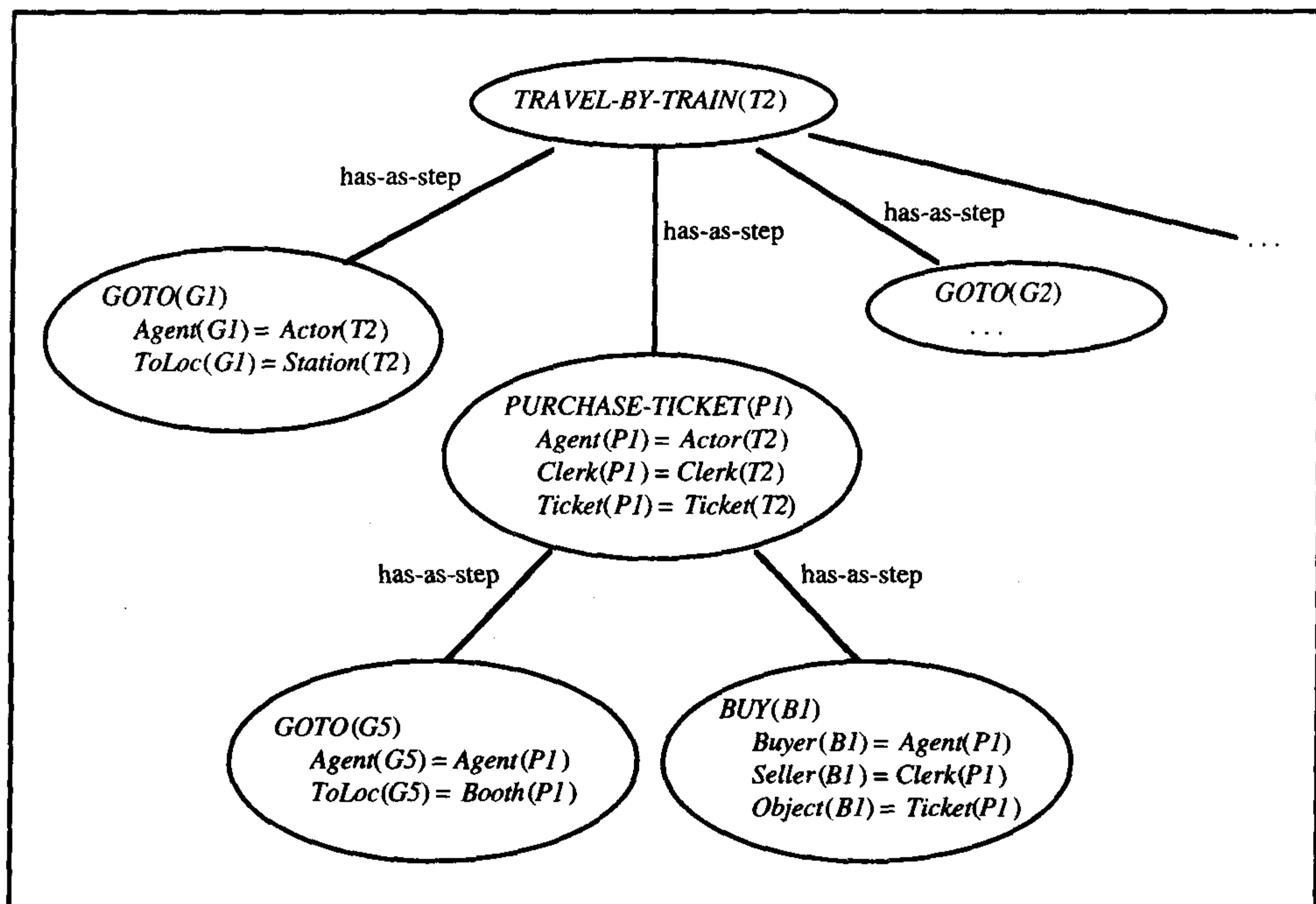


图 15.7 乘火车旅行的规划

假定图 15.7 中的这个规划已经建立起来,可以用它来解释下面篇章的连贯性:

- 12a. Sue wanted to take a train to Rochester. (Sue 想乘火车去 Rochester.)
 12b. She walked up to the ticket clerk. (她走向售票员。)

句子 12a 用于确定 Sue 的目标是“traveling to Rochester by train”。表述目标的句子必须与这个规划的层次树中根节点的行为相匹配。与前一节介绍的使用脚本的方法类似,这将与该规划中的根节点相匹配,并产生下面的等价性假设:

$Actor(T2) = Sue1$
 $Train(T2) = TR37$
 $DestCity(T2) = ROC$

句子 12b 的内容为:

$GoTo(E3) \ \& \ Agent(E3) = PRO1 \ \& \ ToLoc(E3) =$
 $LOC1 \ \& \ At(C1, LOC1) \ \& \ TicketClerk(C1)$

给定下面的等价性假设,这个句子可以与 P1 的第一个分解步骤相匹配:

$G5 = E3$
 $Agent(G5) = PRO1 (= Agent(P1) = Actor(T2) = Sue1)$
 $ToLoc(G5) = LOC1 (= Booth(P1))$
 $Clerk(P1) = C1$

有了这些等价性假设,这个规划就推导出了句子 12b 的内容。我们可以通过在对这些角色实例化以后分析 P1 的部分定义来观察这一点,如图 15.8 所示。这幅图中只给出了用来推导出句子 12b 的内容的相应公式。

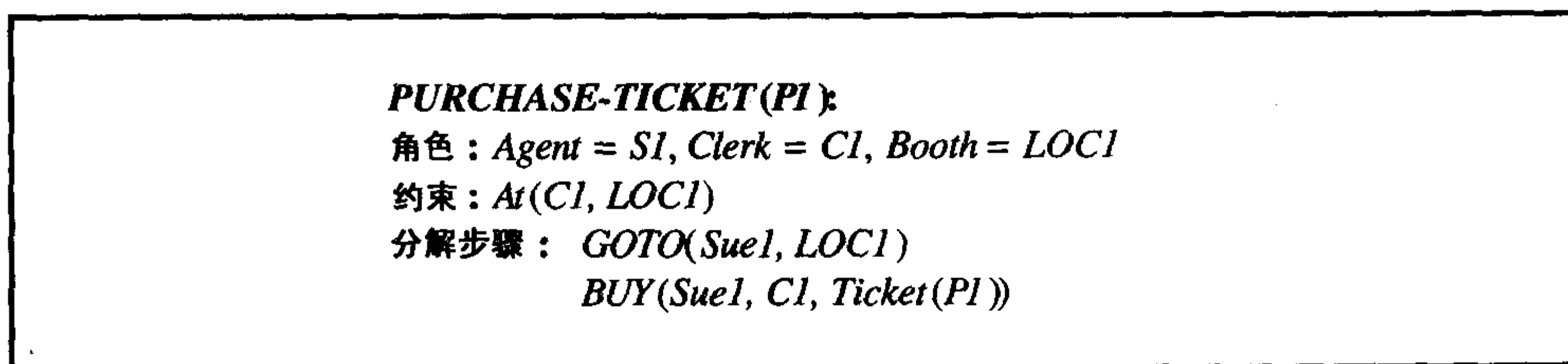


图 15.8 对行为 P1 的一部分进行实例化以推导出句子 12b

当然,这一方法的成功要取决于特定的算法,在给定输入的条件下,此算法能够构建一个规划。在这里,我们可以采用分解链的技术。分解链是指从分解关系中进行搜索直到发现一个行为的链。其中,这个行为链中的第一个行为与输入匹配,行为链中的最后一个行为与目标匹配。当目标已知时,自上而下的方法常常是很有效的。例如,在处理完句子 12a 以后,这个规划将包括单个节点“TRAVEL-BY-TRAIN(T2)”。我们可以采用自上而下的宽度优先的搜索策略来生成与句子 12b 的内容匹配的那些期望。这一策略将生成下面的有序的期望(按照已列出的顺序):

首先,检查 T2 的所有直接子步骤:

$GOTO(G1)$ ——由于 $LOC1 \neq Station(T2)$,匹配失败

$PURCHASE(P1)$ ——由于类型不匹配,匹配失败

$GOTO(G2)$ ——由于 $LOC1 \neq Loc(TR37)$,匹配失败

$GET-ON(G3), TRAVEL(T3), ARRIVE(A1)$ 和 $GET-OFF(G4)$ ——由于类型不匹配,匹配失败

然后,检查 G1 的子步骤,假定它无法分解。

然后,检查 P1 的子步骤:

$GOTO(G5)$ ——与上面定义的等式匹配。

这个例子忽略了几个困难的问题,即这些匹配是怎么搜索得到的。例如,为什么句子 12b 不与 T2 中的步骤 1: $GoTo(Actor, Station)$ 匹配(见图 15.5)? 问题在于关于地点推理的子系统。要进行有效推理,系统要能够确定职员的位置 $LOC1$ 不是火车站所在的位置,这些问题可以使用启发式信息来处理。至于怎样将地点的推理和基于规划的匹配技术结合在一起,还需要进行深入的研究。

另一个问题是,当没有定义目标时,该怎么办? 一种方法是在所有可能的目标中,采用自上而下的方法来搜索;另一种方法是使用自上而下的方法来匹配任何行为的子步骤,使用发现的行为再匹配其他行为的子步骤,并依次类推。这两种方法的复杂性都很大,因为可能的匹配会非常多。例如,对第一个句子 12b,怎样来识别目标呢? 因为在知识库中存在的很多行为都涉及“GOTO”这一步,除了“TRAVEL-BY-TRAIN”中包含“GOTO”行为外,知识库中可能存在

“GO-FOR-A-WALK”,也可能会到达最后的目标。那么,系统到底怎样确定选择哪一个呢?目前还没有很好的方法。一些系统利用启发式信息所包含的优先关系来确定。另一种方法是推迟决策,直到有其他的输入信息能够辅助我们的决策。

◦ 15.7 基于行为 – 结果的推理

虽然层次化规划中的分解链是减小搜索空间的一种有效方法,但仍不能处理很多篇章。问题在于不是所有的因果关系都能够很容易地表示为分解关系。这一节将探讨用其他方法对更一般的规划进行推理,来大大扩充系统能够处理的情境范围,而代价是搜索空间的扩大。

考虑一个涉及行为 – 结果关系的简单例子:

13a. Jack needed to be in Rochester by noon. (Jack 需要在中午前到达 Rochester。)

13b. He bought a ticket at the station. (他在火车站买了一张票。)

系统只有认识到第一个句子描述了执行“TRAVEL-BY-TRAIN to Rochester”这一行为的结果,并且第二个句子是该规划的一个步骤,才能够解释这段话。为了做到这一点,必须将状态和行为一样作为目标来处理,并且,规划推理算法必须不仅能够根据分解步骤从一些行为搜索到另一些行为,而且能够根据行为的结果从行为搜索到状态,反之也一样。

例如,从句子 13a 的目标状态为“*At*(Jack1, ROC)”开始分析。句子 13b 的逻辑表达式与期望不直接匹配。要发现它们之间的联系,系统必须首先把“TRAVEL-BY-TRAIN”行为的结果与目标状态相匹配,然后,使用分解链来发现与句子 13b 中的“PURCHASE”行为的匹配。最后的结果规划如图 15.9 所示,椭圆形节点表示行为,矩形结果表示状态。其中,使用了 3 种因果联系,如下:

has-as-step——将行为与该行为分解步骤中的一个动作联系起来

enables——将一个状态与将此状态作为前提条件的一个行为联系起来

has-effect——将行为与其结果联系起来

另外,与句子的逻辑表达式已经匹配了的那些节点都标记为黑体,其他节点标记为正常字体。例如,图 15.9 说明了“TRAVEL-BY-TRAIN”行为的一个结果是状态“*At*(Jack1, ROC)”。

如果两个行为是相关的(这里的相关不是指这两个行为都是其他某个行为的步骤,而是指一个行为使另一个行为发生,即一个行为的结果是另一个行为的前提),那么行为 – 结果的推理就更复杂了。例如:

14a. Jack bought a CD player. (Jack 买了一台 CD 播放机。)

14b. He played his favorite CDs all night long. (整个晚上,他都在播放他喜欢的那些 CD。)

假定在知识库中没有“buying a CD player and then using it to play CDs”(买了一台 CD 播放机然后用它来播放 CD)这样的行为,但是存在这两个单独行为的定义。那么,“playing CD”(播放 CD)这一行为的前提条件是 agent 有一台 CD 播放机,“buying”(买)这一行为的结果是 agent 拥有了这个物品。这样,第二个行为的前提和第一个行为的结果能够匹配,所以能够确定这两个行为之间的使能关系。结果规划如图 15.10 所示。

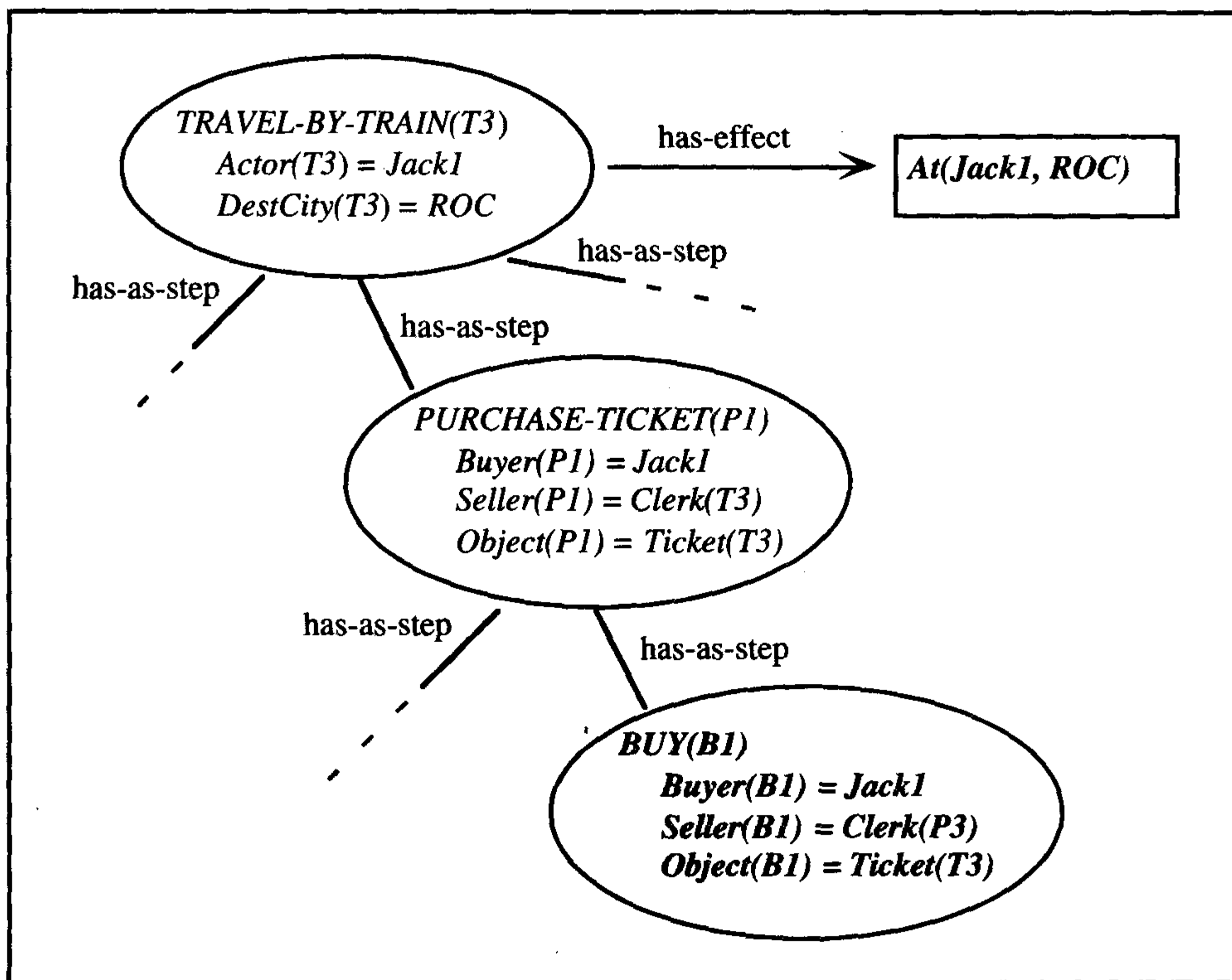


图 15.9 从篇章 13 中识别出来的规划

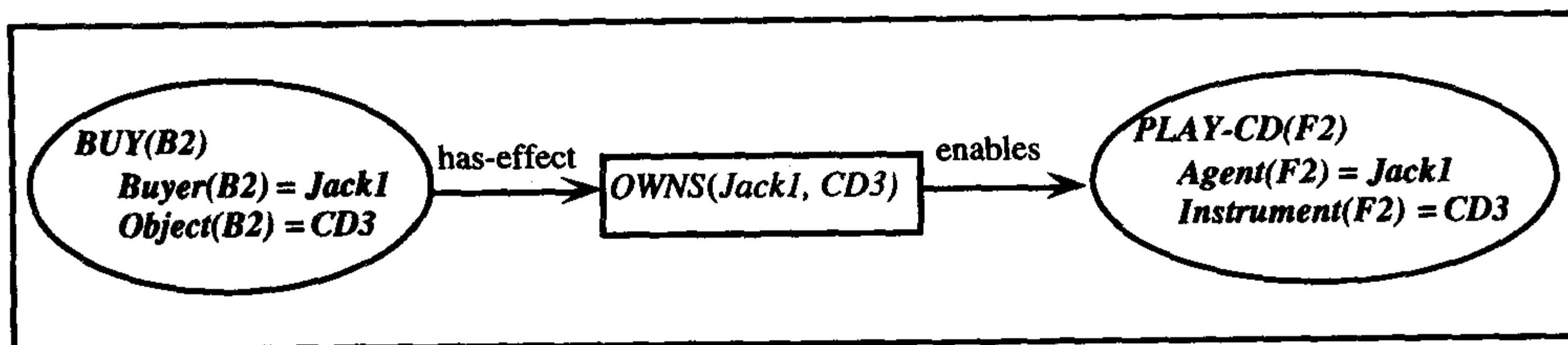


图 15.10 篇章 14 中涉及使能关系的规划

15.7.1 规划推理算法

这一节将用更简洁的形式来描述一个规划推理算法。我们将用来生成期望的规划结构称为“E-plan”。一般来说,系统必须维护很多的“E-plan”来处理歧义,虽然这个算法假定系统最开始只有一个“E-plan”。将“E-plan”扩展为一个集合是很简单的,新的算法仍然需要分解链,但同时检查中间的行为-结果和行为-前提的联系。

在描述这个算法之前,有必要先定义一些术语。具体来说,P 这个“E-plan”中已经与逻辑表达式匹配的节点称为已识别节点,而没有与逻辑表达式直接匹配的节点称为期望节点。期望节点还可以进一步分为:

A-set(P)——描述行为的期望节点

P-set(P)——描述能够使一个行为发生的那些状态的期望节点

E-set(P)——描述一个行为的结果的那些期望节点

这些节点是很有用的,因为它们包含规划中最可能匹配新句子的那些节点。同时,要注意的是 P-set 和 E-set 常常交叉,一个状态可能是一个行为的结果,也可能是另一个行为的前提条件。

例如,考虑图 15.11 中的规划,这个规划是从篇章 12 中推理出来的规划的一个片段。因为边标记可以根据上下文来确定,所以图中没有标出来。换言之,两个行为之间的边是“has-as-step”边,从状态到行为之间的边称为“enable”边,从行为到状态之间的边称为“has-effect”边。节点的标记说明了节点的内容,但是节点的角色值和等价性的细节没有标出来。节点“GOTO-CLERK1”与句子 12b “She walked up to the ticket clerk”(她走向售票员)的逻辑形式相匹配。给定这个规划, A-set 包括描述“PURCHASE-TICKET1”和“BUY-TICKET1”的那些节点, P-set 包括描述状态“OWNS-MONEY1”,“AT-LOC-CLERK1”和“CLERK-HAS-TICKET1”的那些节点, E-set 包括“AT-LOC-CLERK1”,“OWNS-TICKET1”,“CLERK-HAS-MONEY1”,“NOT-OWNS-MONEY1”和“NOT-CLERK-HAS-TICKET1”的那些节点。

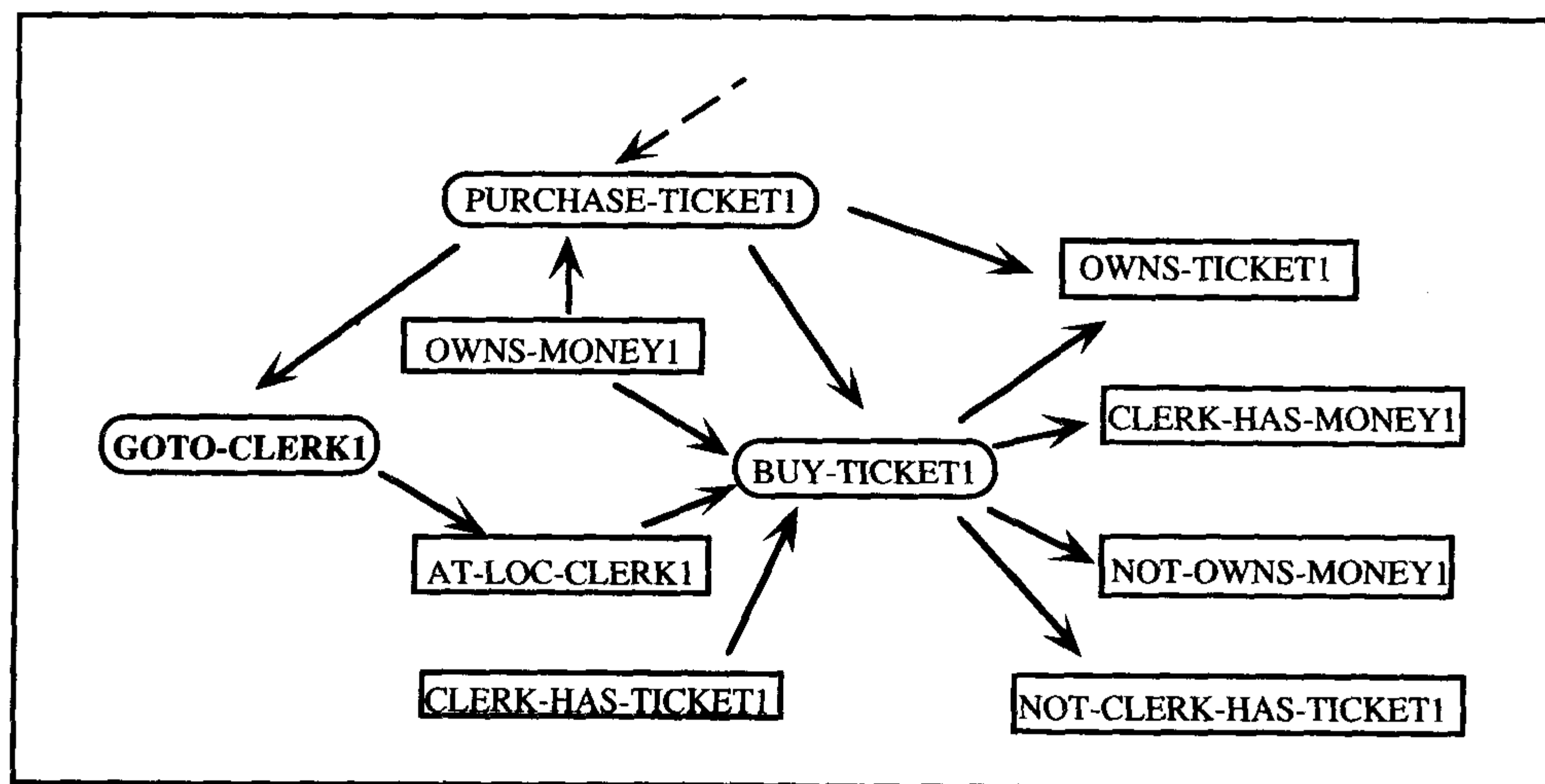


图 15.11 篇章 12 的“E-plan”的一个片段

给定一个规划 P, 扩展这个规划 P 涉及到将 A-set 中的所有行为和这些行为的分解都加到规划 P 中。这个动作可以采用一种宽度优先的方式用于前面描述的分解链推理。例如, 扩展图 15.11 中的这个规划, 将增加一个分解“BUY-TICKET1”、增加两个行为“GIVE-CLERK-MONEY1”和“GIVE-SUETICKET1”, 还会增加这些行为的前提和结果状态。行为“PURCHASE-TICKET1”也在 A-set 中, 但是由于它已经被分解了, 所以没有再扩展。

基于上面的这些概念, 我们可以对这个算法进行描述。一旦将一个行为加入到规划中, 其所有前提和结果都必须被包括进去, 并在前提、结果与行为之间建立合适的联系。另外, A-set, P-set 和 E-set 都必须用新的信息更新。根据这个新句子是描述了一个行为、状态还是目标的不同, 算法接受三类不同的输入。这个算法能够分别处理每一种情况。

首先, 考虑输入描述一个目标的情况。如果目前的“E-plan”为空且目标是一个行为, 则将行为增加到空的规划中, 并将其标记为目标; 如果目前“E-plan”为空且目标是一个状态, 那么就可以发现把这个状态当做目标的所有行为, 并对每一个发现的行为创建一个新的“E-plan”。现在, 考虑初始“E-plan”非空的情况。存在两种可能的解释, 一种解释是新目标是当前

“E-plan”的子目标(意味着新目标应该描述“E-plan”中的一个行为或一个结果);另一种解释为,新目标是一个比当前“E-plan”中的目标更高层次的目标,“E-plan”中的这个目标应该更新。这种情况描述起来有点复杂,更准确的算法描述如图 15.12 所示。

将把目标 G 结合到“E-plan”E 中:

1. 如果 E 为空的规划,那么
 - 1.1 如果 G 描述一个行为,将这个行为和这个行为的前提和结果增加到“E-plan”中,并将行为标记为结果。
 - 1.2 如果 G 描述一个状态,那么查找结果为 G 的所有行为 A_1, \dots, A_n 。用 1.1 描述的方法,为每一个 A_i 创建一个新的“E-plan”。
2. 如果 E 为非空的规划,那么
 - 2.1 试图将目标 G 结合到 E 中(用后面将要描述的结合行为和结合状态的方法)。
 - 2.2 如果 2.1 失败了,则将 OLDG 作为 E 的目标。用步骤 1 中的方法,构造一个可能的以 G 为目标的新的“E-plan”,然后尝试将 OLDG 结合到新的“E-plan”中。对于那些匹配成功的“E-Plan”,将 OLDG 和旧的“E-plan”结合到新的“E-plan”中。

图 15.12 将目标结合到“E-plan”中的算法

现在,考虑将行为 A 结合到“E-plan”中的算法。首先,如果“E-plan”为空,则使用图 15.12 描述的结合目标的算法将行为结合到“E-plan”;如果“E-plan”不为空,则可以用 3 种不同的方法检查规划中的匹配:

- 这个行为与 A-set 中的一个行为相匹配
- 这个行为有一个与 P-set 中的一个状态相匹配的结果
- 这个行为有一个与 E-set 中的一个状态相匹配的前提

对于发现的每一个匹配,通过增加行为,以及与“E-plan”中匹配的那部分的联系,可以生成一个新的“E-plan”。这样,一旦加入一个行为,其所有前提和结果也都将加入。如果使用这一方法没有发现匹配,则可以扩展“E-plan”中的行为,然后再试验一次。一般来说,可以限制“E-plan”被扩展的次数,否则,可能会有“E-plan”的组合爆炸问题。这里,可以假定只对 A-set 中的行为进行一次扩展。

用前面介绍的算法可以处理篇章 14。句子 14a“Jack bought a CD player”(Jack 买了一台 CD 播放机)描述了一个“buying”行为 B1;因为“E-plan”为空,B1 被增加到“E-plan”中并作为“E-plan”的目标。句子 14b“He played his favorite CDs all night long”(他整夜播放他喜欢的 CD)描述了一个播放 CD 的行为。把“PLAY-CD”行为与 A-set 中的期望直接匹配,结果没有匹配成功,并且“PLAY-CD”的任何结果都不与 B1 的前提相匹配。但是,“PLAY-CD”的前提与 B1 的结果“拥有 CD 播放机”匹配。这样,“PLAY-CD”行为可以结合到规划中,建立了一个“E-plan”,并且与图 15.10 所描述的类似。

下列例子中的篇章先描述一些行为,然后再引入一个目标。句子 15a 介绍了一个行为,然后句子 15b 描述为什么要执行这一行为。

15a. Jack bought a new stereo. (Jack 买了一台新的音响。)

15b. He wants to play music at his party. (他想在他的聚会上播放音乐。)

我们采用图 15.12 中的步骤 2.2 的方法来处理句子 15b。具体来说,句子 15b 建议一个新的“E-plan”,其目标是“playing music”(播放音乐)这一行为;句子 15a 描述的行为可以解释为触发这一行为。因此,产生图 15.10 所示的规划。

最后,考虑篇章 13,如下所示:

13a. Jack needed to be in Rochester by noon. (Jack 需要在中午前到达 Rochester。)

13b. He bought a ticket at the station. (他在火车站买了一张票。)

句子 13a 描述了一个目标,句子 13b 描述了分解链中的一个步骤。要把句子 13a 中描述的目标结合到“E-plan”中,系统需要寻找结果为 $At(Jack1, ROC)$ 的所有行为。行为“TRAVEL-BY-TRAIN”是其中之一,所以建立了一个“E-plan”,此“E-plan”有一个“TRAVEL-BY-TRAIN”的实例,与这个结果相关联,并且带有一些分解步骤,如图 15.13 所示。为了将句子 13b 结合到“E-plan”中,将行为“BUY”与“E-plan”匹配,结果是匹配不成功;然后,与“E-plan”中的结果和前提匹配,也不成功,所以将扩充“E-plan”。经过扩充后,行为“BUY”与行为“PURCHASE-TICKET”的第二步匹配,产生的结果规划与图 15.9 中所示的规划类似。

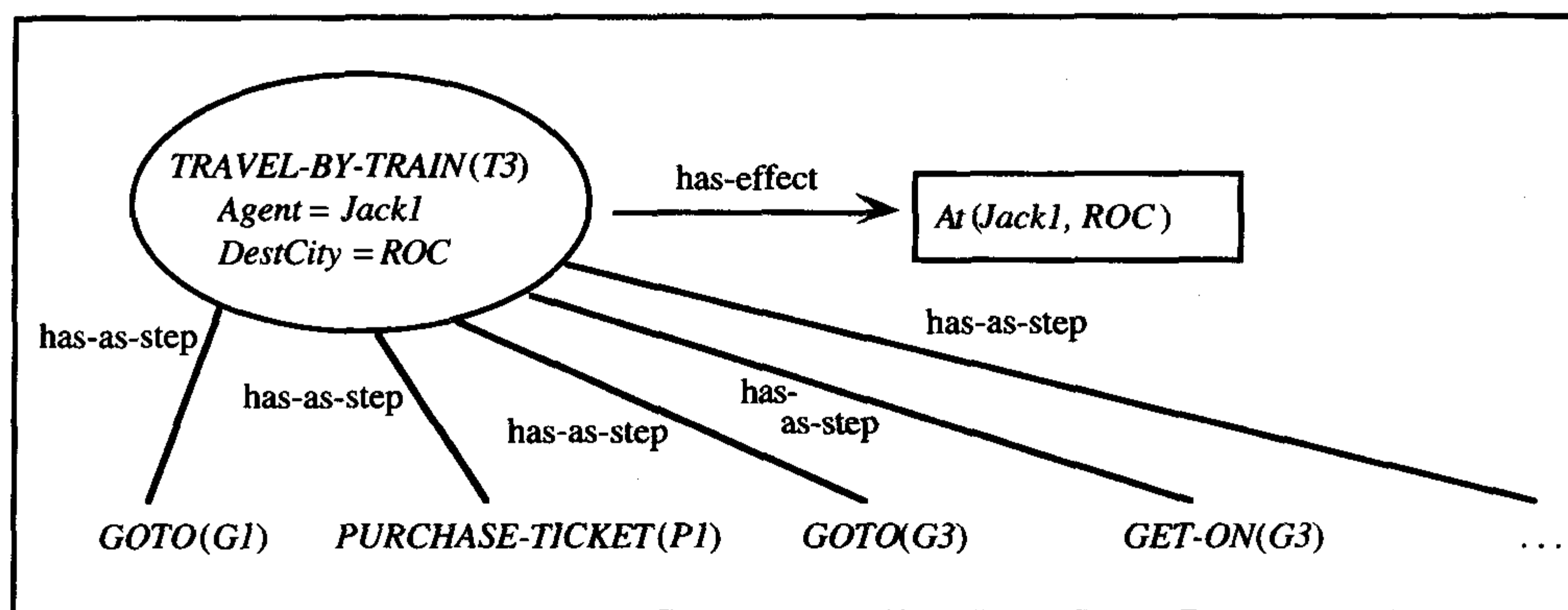


图 15.13 句子 13a 结合到“E-plan”后的规划

需要处理的最后一类输入涉及那些描述状态而非描述目标的句子。这一类中的描述主要用来提供背景信息或描述规划中行为的结果。如果当前“E-plan”为空,状态将被解释为背景信息,这是通过增加状态到规划中的“E-set”和“P-set”来实现的。虽然此状态不是规划中任何一个行为的结果或前提条件,但允许这个状态与后面加入的行为建立联系。例如:

16a. Sue had twenty dollars. (Sue 有 20 美元。)

16b. She bought a ticket to Rochester. (她买了一张去 Rochester 的票。)

在这种情况下,把结合行为算法应用到句子 16b 上,将会发现“having money”(有钱)使“buying the ticket”(买票)成为可能。如果“E-plan”不为空,则试图与 E-set 和 P-set 中的状态进行匹配。对于发现的每一个匹配,用新的等价性假设来更新规划,然后从 E-set 或 P-set 中把状态删除;如果没有发现匹配,状态将增加到 E-set 中,以便以后使用。

15.7.2 算法的一些局限

在很多情况下,前面描述的算法都能够发现句子之间的联系。但是,还有很多其他情况,

并不能很好处理。例如,不能实行自下而上的分解链,也就不能预测当前行为的超集(即当前行为是其子部分)。这样,就不能发现下面篇章中两个句子之间的联系:

17a. Sue bought a ticket to Rochester. (Sue 买了一张去 Rochester 的票。)

17b. She boarded the train at 4PM. (她在下午 4 点上了火车。)

为了发现这两个句子之间的联系,系统必须能够判断出这两个句子都是行为“TRAVEL-BY-TRAIN”的分解的一个部分,即“buying a ticket”(买票)是“purchasing a ticket”(购买车票)的一部分,“purchasing a ticket”(购买车票)又是“traveling by train”(乘火车旅行)的一部分;并且“boarding the train”(登上火车)是“traveling by train”(乘火车旅行)的子部分。由于不能判断两个句子之间的联系,一些明显的联系都丢失了。例如,这辆火车将开往 Rochester。很多规划识别算法广泛地使用自下而上的分解链,所以能够发现这之间的联系。由于行为库很大,这些技术将生成很多“E-plan”,搜索问题就成为一个重要问题。对于这个问题,可以采取一种策略,寻找能够包含“E-plan”中一个行为的那些行为,并把新找到的行为作为中间分解的一部分;如果找到,将这些行为加进去,以便把两个句子联系起来。

本节中应该讨论而没有涉及的另一类情况是描述非期望的状态和提出目标的那些陈述。例如,如果一个篇章包括句子“Jack was hungry”(Jack 饿了),你可能想把这个句子解释为“吃饭”这个目标。为了处理这些情况,知识库需要表示有关期望和非期望的状态,以及这些状态怎样将 agent 引导到到新目标的知识。我们将在下一节中介绍这些问题。

◦ 15.8 使用关于理性行为的知识

前面介绍的基于规划的表示只能表示很简单的情境。大部分的故事和对话都远远超出了这一表示形式。这一节,我们将对前面的工作进行扩展,以便能够处理更真实的文本。

前面描述的方法的最明显不足是被刻画的篇章需要在规划执行中不出现任何问题。然而在实际情况下,很多时间都是在描述解决问题的过程。很多内容描述人们为达到目标的各种尝试,例如他们怎样失败、他们设想如何才能成功、他们面对失败的反映、怎样改进存在的问题,最后怎样战胜困难。在涉及多个 agent 时,规划中也可能会有竞争和相互影响的情况。要刻画这些内容,需要使规划和实施的过程更直观。在当前的研究水平下,没有很具体的解决方案,但还是有人给出了一些基本想法。

考虑一个智能规划 agent 的可能表现行为的一个框架。虽然仍然比较简单,这个模型提供了足够的直觉来启发这些技术。这个模型描述了一个智能 agent,这个智能 agent 不断循环地执行以下步骤(这些步骤也可以同时执行):

1. 观察当前的状态(包括观察 agent 前面尝试的行为是否成功了)
2. 选择一个或多个目标
3. 开始实施要达到这些目标的一个规划
4. 在适当的时候执行规划中的行为

不同的规划框架在不同的粒度上制定和执行规划。经典的规划系统刚开始时要建立一个高层次的目标,然后在执行前建立规划。这个模型不适合我们的目的。灵活的或敏感的规划者通常选择一些小规模的目标,从而可以对当前的状态经常做出反映,并且能够立刻付诸于实施。这一框架能够使 agent 在执行过程中对环境变化做出反应,因而是更适于理解篇章的一个模型。

框 15.2 规划推理的形式化分析

要明确的一点是规划推理不是一个单调的过程,这是因为得到的结论是假设而不是蕴涵。这样,与演绎理论的形式化不同,因此对这个过程给出准确的形式化描述就很困难。不过,Kautz(1990)解决了分解链的规划推理算法的形式化描述问题。他给出了怎样把进行规划识别所依赖的知识库 KB 转换为一个新的知识库 KB'的方法,这样基于 KB 的规划推理基础之上的任何非单调结论都是 KB'的逻辑结果。Kautz(1990)给出的转换是建立在增加了 3 个基本假设的结果基础上。下面这 3 个基本假设是规划推理过程的基础:

1. 行为库是完备的,即你试图识别的规划是一个你了解的或能够构建的规划。
2. 每一个行为都是有目的的,即所有观察到的行为是某个规划的部分。
3. 高层目标尽可能少,即这个解释要尽可能简洁。

Kautz 给出了在引入这 3 个假设后怎样生成一个新的知识库的方法。规划推理的执行,可以通过在新建立的知识库上,从观察到的现象开始进行演绎推理来进行。换言之,在修改后的知识库中,观察到的现象将蕴涵所有可以通过推理生成的可能规划所构成的集合。

为了实现这样的模型,需要引入一些新的特性。首先,要将目标表示为一个层次化的体系。在最顶层是最概括的目标(提供了 agent 行为的最终解释),下面是更具体的目标(它们同时也是实现高层目标的一种手段),最后是最低层次的目标,即 agent 正在执行的行为的目标,我们称之为活动目标(active goal)。要达到这个活动目标,agent 规划一个或多个行为。行为序列中正在执行的行为,称为当前行为(current action)。

我们探讨在给定这个模型下,一个 agent 是如何工作的。如果有一个活动目标(例如,开门),可能涉及拧动门把手、把门打开这样的行为。如果发现门被锁了,可能要放弃当前行为,然后执行其他一系列行为来达到当前目标。例如,可能要敲门,然后等待反应;也可能放弃进入房子的目标,而试图去完成另一个目标。需要注意的是,没有一个规划能够涉及 agent 的所有这些行为,但是给定了上述模型,我们能够对这些行为的顺序给出一个合理的解释。

因为一个 agent 的行为不一定完全遵循定义好的规划,所以规划识别模块必须能够在这种关于意图性行为的更通用的模型基础上来识别这些行为。为了更准确起见,我们定义一个新的结构“iplan”(intentional plan 的简写),“iplan”包括下面的模块:

- 这些目标的一个层次结构表示,其中叶子目标节点表示活动目标。
- 一系列行为,其中最后的行为达到活动的目标,而有一个行为标记为当前行为。

调整前面的规划推理算法可以模拟这个分解链,再用其他推理模块来填充目标层次结构并对激活这个目标的一系列行为做出规划。agent 的行为是用对“iplan”的一系列“理性更新”(或称为“合理更新”)来定义的,这里更新表述的是两个“iplan”之间的关系。直观地说,我们称两个

“iplan” IP_1 和 IP_2 是通过更新关系互相关联的。如果一个带有 IP_1 的智能 agent, 从外部世界接收到观察 O 以后, 有可能采用 IP_2 描述的一系列目标。

让我们看一些理性更新的例子。根据行为类型的不同, 可以把这些更新大致分为 3 类, 虽然这里的分类体系并不完备。在每一种情况下, IP_1 是旧的“iplan”, IP_2 是根据对 IP_1 的修改做出的合法的更新。

15.8.1 规划更新

这些更新来自于 agent 的规划行为, 不涉及对这个 agent 的观察。

1. **目标/行为**——如果一个 agent 在 IP_1 的目标为 G , 但是在 IP_1 没有行为能够达到目标 G , 那么可以通过增加一个行为或行为的层次树来建立 IP_2 。其中, 如果是增加行为的层次树来建立 IP_2 , 这个层次树的根必须是目标 G 。
2. **分解**——如果一个 agent 具有层次化的行为结构, 且 IP_1 的当前行为是行为 C , 但是行为 C 不能直接执行, 那么可以通过将行为 C 分解为更小的行为来构造 IP_2 。
3. **前提条件不满足**——如果当前行为 C 需要一个前提条件且此前提条件不为真, 那么或者在 IP_2 中用一个新的当前行为来扩展行为层次结构以达到这个前提条件; 或者放弃行为 C , 引入新的行为来达到当前的活动目标。

15.8.2 基于执行的更新

这种更新来源于这个 agent 对行为执行成功还是失败的观察。

4. **执行成功**——如果 agent 的观察 O 表明当前行为 C 已经成功执行了, 那么 IP_2 的新的当前行为将是执行序列中的下一个行为; 如果没有下一个行为, 就选择一个新的活动目标。
5. **执行失败**——如果 agent 的观察 O 表明当前行为 C 失败了, 那么 IP_2 就标记当前行为是失败的。
6. **行为失败 1**——如果当前行为失败, 那么在 IP_2 中可能再试一次同样的行为; 或者尝试新的行为以达到活动目标。

15.8.3 目标更新

这种更新来源于 agent 观察到一些需要改变其目标结构的现象。

7. **非期望的状态**——如果 agent 观察到状态 O , 且 O 隐含着非期望的条件 U , 那么可以通过增加 $\neg U$ 到 IP_1 的已经存在的目标结构中来创建 IP_2 。这可能需要放弃 IP_1 中已经存在的一些目标。
8. **行为失败 2**——如果当前行为失败, 则 IP_2 中的当前活动目标被删除, 并激活一个新的目标。

分析下面的例子:

18a. When Jack got home, he tried the door(当 Jack 回到家时, 他尝试开门,)

18b. but found it was locked. (但是, 他发现门是锁着的。)

- 18c. He knocked, (他敲门,)
 18d. but there was no one home. (但是,家里没有人。)
 18e. He walked around the house to the back door. (他绕过房子去后门。)

图 15.14 总结了分析过程,其中的活动目标和当前行为用粗体表示。为了处理这一情况,知识库将包括回家的知识,例如通过门才能进入房子。我们首先假定初始目标来自于 IPLAN-18a 中的子句 18a, IPLAN-18a 具有目标层次结构“*Inside-House(Jack1)*”和子目标“*EnterViaDoor(Jack1, Door1)*”,其中,后者是活动目标。达到这个目标的行为序列包括“*OpenDoor(Jack1, Door1)*”和“*WalkThrough(Jack1, Door1)*”,其中,前者是当前行为。子句 18b 表明,通过声明一个前提条件的失败来说明这个行为失败了,并产生了 IPLAN-18b。当然,“*OpenDoor*”这一行为的合适的定义需要门没锁上这一前提。子句 18c 可以解释为对一个失败行为的反映,这是达到活动目标的另一个方法。子句 18d 表明这一尝试也失败了,结果为 IPLAN-18d。最后,子句 18e 解释为放弃活动目标,试图采用另一种方法来达到更高层次的目标“*Inside-House(Jack1)*”。



图 15.14 篇章 18 的“iplan”的跟踪

语言中有很多表达结构,它们能够明确指明这个模型中需要定义的概念和行为。例如,篇章 18 说明了连词“but”的两种用法,但是每一种情况都表示其前提条件不具备。其他的例子也很容易找到,如:

- 19a. Sue had to find a way to get enough money for the ticket.
 (Sue 不得不找一种方法来得到买票所需的钱。)

19b. She tried to use the automatic teller machine, but it was broken.

(她尝试用自动取款机,但是机器坏了。)

19c. She eventually gave up and walked to the bus station and took a bus.

(最后,她放弃了。她走到公共汽车站,乘了一辆公共汽车。)

其中的短语“find a way”,“try”和“give up”都与“iplan”模型中定义的这些行为直接相关。

然而,“iplan”模型只是一个开始,很多其他复杂情况都需要更通用的模型。例如,有些规划涉及一些重复的动作(如用锤子钉钉子)和周期性的时间(如每天都要走到学校)的行为,这些规划都不能用目前的方法来表示。比如,系统可能需要识别主体买了公交车票,以便能够每天都搭车。买公交车票的行为是整个公交旅行的使能条件。如果买公交车票这一行为只与“TAKE-BUS”行为的这一次出现有联系,那么这个故事的重要方面都丢失了。还有一些不涉及重复的目标,也很难表示。例如,去剧院这一行为的目标说明什么?你可能会说目标是看话剧,放松自己。你必须对这一目标进行明确的表示,才能理解由于钱包被偷了,你的好心情荡然无存这样的场景。为了处理这一篇章,我们需要一种理论使这种情况的分析变得更有价值。

因为人们往往不止有一个目标,所以情况会很复杂。当考虑描述多个目标的情境时,系统需要理解这些目标之间是有冲突的或是一致的。例如,一个句子描述某人正犹豫是准备明天的考试还是出去看电影。系统需要能够表示这样进退两难的局面,并能对像目标为什么会冲突这样的事件进行推理。这将需要理解一个人是怎样试图避免冲突并达到目标的。另一个篇章可能描述的情况是两个人具有冲突的目标(例如,两个人都想拥有某一特定的赛马),根据这个冲突,他们各自的行为都需要解释。当两个目标互相补充或者当人们协作来完成目标时,情况也是这样。

15.9 小结

关于因果关系和日常活动的知识是理解很多篇章所必要的。没有这些知识,我们就不能正确处理很多单词歧义问题和指代歧义问题。这些知识用来生成与输入句子的可能解释相匹配的那些期望。关于怎样控制这个过程以避免生成过多的期望这个问题,已经开展了大量的研究工作,并取得了一些成果。例如,经常采用的方法包括大规模的脚本结构、使用了关于行为和目标的通用知识的更灵活的规划推理系统。但是,要使这些技术真正实用化,我们还有很多工作要做。

15.10 相关工作与深入阅读材料

在篇章理解中,连贯性的概念是非常关键的。篇章理解中的连贯性涉及的是发现句子之间的语义联系而非结构特性,认识到这一点是很重要的。例如,Halliday 和 Hasan(1976)认为篇章连贯性是通过一定的语言手段达到的,主要包括:

指代——使用代词和定指性描述来指代前面句子中的对象(深层和表层形式,one 回指,等等)。

省略——利用一个句子的结构来填充另一个。

连词——用“and”, “further”, “but”等将句子连接起来。

词汇的联系——利用一些相互关联的词汇, 例如反义词、解释和重复, 等等。

这种方法与本章介绍的计算理论有很明显的区别。Halliday 和 Hasan 认为这些现象使句子间具有连贯性。而计算性的方法则认为一个篇章的内容和篇章中的这些话题流使这个篇章呈现连贯性, 且篇章的连贯性又引发了上面这些现象。换言之, 指代性的联系并不能保证文本的连贯性, 而连贯的文本则能够建立指代性的联系。Hobbs(1979)详细讨论了它们之间的不同。

使用世界知识来解释句子的语言理解系统几乎都利用了某一种形式的匹配。虽然 15.2 节中的公式刻画了这些算法的一些基本方面, 但是, 我们并没有深入探讨匹配算法的形式化方面的问题。基于等价性匹配技术的形式化一般都遵循 Charniak(1988)(Charniak 也研究了利用匹配技术来处理定指问题)的描述。Kautz(1990)给出了通用框架内类似的分析算法来解决规划识别的问题。自然语言理解中的通用溯因推理机制的大力推崇者之一是 Hobbs(Hobbs 等, 1993)。

使用通用的世界知识来确定句子之间联系的相关研究中, 较早期的工作之一是 Schank 和 Rieger(1974)。他们给出了 16 种不同的联系, 其中大部分又都用在基于规划的推理系统中。Rieger 的系统主要是从每一个句子的所有推论中进行搜索, 所以效率不高。后续的很多研究工作主要集中于通过使用更专门的关于行为的知识表示方法来提高效率。

Schank 和 Abelson(1977)提出了脚本的概念, Cullingford(1981)建立了第一个基于脚本的系统。15.5 节较详细地描述了这一工作, 不过, 我们是从传统的行为表示的角度进行介绍的。DeJong(1979)曾经使用脚本从新闻故事中抽取信息。这些系统都使用了匹配和推理技术来解决指代问题。

在人工智能的规划问题研究中, 行为表示的研究非常多, 两篇有代表性的文章是 McCarthy 和 Hayes(1969)和 Fikes 和 Nilsson(1971)。Allen 和 Hendler 和 Tate(1990)对规划问题进行了很全面的介绍。Allen 和 Perrault(1980)在规划推理系统中使用这些方法并进行了修改。这里使用的形式化表示与 Litman 和 Allen(1987; 1990)提出的方法类似。要准确定义行为之间的因果关系是不容易的。Goldman(1970)从哲学角度介绍了生成关系, 并把它与使能关系和分解关系进行了比较和区分。Pollack(1990)对这一工作又进行了扩充和改进以便用于计算系统。

在人工智能中, 规划识别的研究[如 Schmidt(1978)]已经有一段时间了, 这与自动诊断和自动教学有一些相似性。Wilensky(1983)以及 Allen 和 Perrault(1980)把规划识别应用于语言理解中。Sacerdoti(1977)提出了层次化规划的概念。规划识别技术可以用于理解语言的内容(如本章), 也可以用于理解说话者的目的(如第 16 章和第 17 章)。Carberry(1991)对语言理解中的规划识别的应用进行了很好的总结。Charniak 和 Goldman(1993)阐述了建立概率型的规划识别系统的方法。15.6 节和 15.7 节中算法的基本思想来自于 Allen 和 Perrault(1980), Kautz(1990), Ferguson 和 Allen(1993)。

也有研究人员使用关于有意图性行为的知识来解释故事, 其中 Wilensky(1983)是这一研究的开拓先锋。15.8 节从“iplan”结构的角度介绍这些技术。Wilensky 也探讨了其他一些问题, 包括目标的冲突和包含。由于篇幅的原因, 我们没有阐述这些问题。Wilensky 对 15.8 节的其他影响还包括他在有关规划的文献中所提出的反应性的规划和智能 agent 框架。Allen, Hendler 和 Tate(1990)探讨了这些问题, 我们将在第 17 章中更详细地进行探讨。

15.11 习题

1. 【易】对于下面的每一个句子,请给出与句中典型行为相关的一些事实,来证明这些句子可以作为句子“Jack walked to school”(Jack 步行去上学)所生成的期望。

- a. He is going to attend class.(他要去上课。)
- b. His car is broken.(他的车坏了。)
- c. He will be tired at school.(到学校时,他将会很疲惫。)

2. 【易】用图的形式表示下面的两个公式。要使这两个公式匹配,需要掌握关于类型、等价性、不等价性的哪些知识? 这一匹配所生成的等价性假设是什么? 说明这一匹配所蕴涵的结果。

*Acquire(E1) & Agent(E1) = Jack1 & Theme(E1) =
Book1 & AtLoc(E1) = Store34*

Buy(E2) & Agent(E2) = Hel & Theme(E2) = It1 & Price(E2) = \$10

3. 【易】假定句子“Jack went to the store”(Jack 去了商店)后面的句子为 a, b, c, d 或 e 之一,对每一种情况请说明这两个句子之间的因果关系(如果存在)和时间关系:

- a. He took his car from the garage.(他从车库中把车开出来。)
- b. He had taken the money from Sue's room.(他从 Sue 的房间拿了钱。)
- c. He bought some peaches.(他买了些桃子。)
- d. Now he has plenty of beer.(现在,他有很多啤酒。)
- e. His car broke down along the way.(他的车在路上坏了。)

4. 【中】考虑下面的篇章:

- a. Jack bought some roses at Honest John's Flower Mart.
(Jack 在 Honest John 的花市买了些玫瑰花。)
- b. He paid thirty dollars for them(他花了 30 美元,)
- c. and Honest John swore that they were fresh.
(Honest John 保证说这些花是新鲜的。)
- d. But they wilted as soon as he left the store.
(但是,他一离开花店,这些花就枯萎了。)

对每一个句子,请给出:

- 这个句子的语义表达式,用基于事件的知识表示语言(KRL)来表示。
- 与前面句子所生成的期望相匹配所需要的等价性假设(第一个句子除外)。
- 为了解释后面的句子,需要从这个句子中生成的期望。
- 一条常识规则,这条规则可以用于从这个句子中生成期望。

5. 【中】编程实现 15.2 节描述的匹配算法。为此,需要实现一个子系统,用于进行第 13 章中描述的关于类型层次结构的推理,以及其他一些子系统,用于跟踪哪些常数是不相

等的。给定类型的层次结构和不相等信息后,程序可以接受两幅图作为输入,如果这两幅图匹配成功,请返回一些等价性假设,并用 15.2 节中的例子验证此程序。另外,用其他例子来验证你的程序的健壮性(能处理其他情况)。

6. 【中】假定有如图 15.4 和图 15.5 所示的行为定义,请给出 GetOn 行为的定义。其中,包括与列车长交流的行为。在这些定义的基础上,给定目标为 TRAVEL-BY-TRAIN 行为,请详细描述使用分解链来识别下列句子的规划推理算法的步骤。并且,按照生成的顺序列出每一个期望,并解释这些期望与输入匹配或不匹配的理由。

Jack gave the conductor a ticket. (Jack 给列车员一张票。)

请画出最后的分解链,并在图上标记每一个角色的值。

7. 【中】根据 15.7 节描述的算法分析下面两个句子,并给出完整的规划推理算法:

Jack needed to have a ticket. (Jack 需要一张票。)

He walked to the ticket clerk. (他走向售票员。)

具体来说,请定义每一个句子的语义并跟踪规划推理算法的每一步。讨论所遇到的任何一个有歧义的问题,并提出能够确定合理解释的方法。给出最后的规划以便将这两个句子联系起来。

8. 【难】使用分解链技术,实现一个规划识别系统。此系统应该使用带分解步骤的行为定义,但不要明确地表示角色的值。换言之,每一个行为都表示为原子,如图 15.11 中的例子。选择一些篇章样例并人工将其中的每一个句子转换成你所需要的表示,以此测试这个系统。

第16章 篇章结构

本章将介绍篇章表示和篇章推理的方法。这里所谓的篇章是指扩展的篇章,而不只是第14章所介绍的用于发现句子之间局部联系的篇章。不能简单地认为扩展的篇章是句子的线性序列。更准确地说,在很多情况下,篇章中的句子往往聚在一起成为一个单元(我们称这个单元为片段),这些片段之间构成一种层次化的结构。16.1节讨论把篇章看做句子的线性序列存在的一些问题;16.2节定义了篇章片段的概念,并解释了说明片段结构的提示语;16.3节说明了片段结构是怎样影响指代表达式(特别是代词)的解释的;16.4节讨论了片段与推理是怎样相互作用以助于理解篇章内容的;16.5节讨论了时态和体态问题,表明了可能发生事件之间的时间和因果联系的解释是怎样需要利用片段结构的;16.6节把先前讨论的不同模块结合在一起以构造一个篇章理解模型;16.7节以一个例子来进一步解释本章所讨论的这些问题。

16.1 篇章结构的必要性

第14章中的指代机制是建立在前面句子的结构和最近约束的基础之上的。而在话题可能转移和改变的对话中,可以看到这些技术是不够的。例如,下面的片段出现在两个人E和A的一段对话的结尾,其中E帮A组装割草机:

- 1a. E: So you have the engine assembly finished.
(这样你就把发动机组装在一起了。)
- 1b. Now attach the rope to the top of the engine.
(现在,把绳子系到发动机的上端。)
- 1c. By the way, did you buy gasoline today?
(顺便问一下,你今天买汽油了吗?)
- 1d. A: Yes. I got some when I bought the new lawn mower wheel.
(是的,当买新的割草机轮子时,我买了一些汽油。)
- 1e. I forgot to take my gas can with me, so I bought a new one.
(我忘记带汽油桶了,所以我买了一个新的油桶。)
- 1f. E: Did it cost much? (它贵吗?)
- 1g. A: No, and I could use another anyway to keep with the tractor.
(没有买那么贵的,因为我还有另外那个可以用。)
- 1h. E: OK.
(好的。)
- 1i. Have you got it attached yet?
(你把绳子系好了吗?)

句子 1i 中的“it”的先行词是指 1b 中的“rope”,因而使句子 1b 后的句子中提到的对象(如 1e 中的“gas can”)能满足所推导出的对“it”的先行词的任何选择性约束。这样,在这段对话中,历史记录列表机制并不能做出正确的预测。实际上,基于篇章实体的线性排序的简单泛化机制并不能提供一个令人满意的解决方案。虽然这样,从直觉上,这个篇章还是比较清晰的。句子 1c 到 1g 是与涉及系绳子的交流中附带的一段对话。在句子 1h 中,说话者 E 通过使用短语“OK”来表示当前话题的结束。那么在对句子 1i 的解释中,与句子 1i 相关的上文是句子 1b。这个结构的解释需要引入篇章片段的概念(篇章片段是一个篇章中连续的一段,其中的每个句子讨论的都是同一个话题)和历史记录列表结构的一种泛化形式(这种形式把篇章片段考虑在内)。

你可能认为,第 15 章中的规划推理模型的泛化形式对于确定这些片段是有用的。利用这一技术,系统可能能够识别出 1c 不是系绳子这一规划的可能延续,因此表示出现了题外话。一旦该题外话结束,规划识别模块就能够分析句子 1i 为句子 1b 中行为的状态。但是,在一个规划识别模块内试图完成所有的工作是很困难的。一旦发生了话题的转移,例如句子 1c 和 1h,规划推理模块将不能发现新旧句子之间的联系,于是系统会由于这个失败而开始一个新的话题。这样做的代价可能是很大的,在一些情况下甚至是不可能的,因为可能存在晦涩的解释,可以将句子 1c 看做是句子 1b 的继续(例如,在系绳子以前,先用汽油擦干净马达)。另外,毫无疑问,说话者 E 通过使用短语“by the way”明确地告诉 A 在句子 1c 中话题改变了。这些短语,即通常所说的提示语,对于提示篇章中话题的变化起了很重要的作用。

另外,基于规划的模型在其他对话场景中可能不适合。例如在争论中,句子之间的联系如“sentence X supports the claim in sentence Y”(句子 X 支持句子 Y 中的论断)或“sentence X contradicts the claim in sentence Y”(句子 X 和句子 Y 中的论断有矛盾)可能是很重要的。在这个场景下也可以使用同样的提示语。根据这些论据,可以得到这样的结论:关于篇章结构的理论不能仅仅根据行为推理来解释。

本章阐述篇章结构的模型,这个模型允许对前面两章介绍的每一种方法进行概括和综合。最重要的思想是一个篇章可以分解为多个篇章片段,其中每一个片段都是紧凑的单元并能够利用前面介绍的方法来分析。

16.2 切分和提示语

虽然大家一致认为篇章切分是必要的,但是对于一个特定篇章的切分是什么和怎样切分并没有一致的共识。缺乏共识的一个原因是除了认为某些句子本质上是聚在一起的直觉之外,对切分并没有准确的定义。尽管存在这些困难,大家还是认为,一个好的切分模型对篇章理解是非常必要的。这个问题可以分解为两个子问题:(1)需要什么技术来分析片段内部的句子;(2)这些片段是怎样联系在一起的。

从实用的目的考虑,一个篇章片段由显示局部连贯性的一系列的子句组成。一个片段内部要具有下面的属性:

- 基于最近上文(例如历史记录记录)的技术应该可以用于指代分析和省略的处理。
- 子句中给出了固定的时间和地点,或者有时间地点的简单变化(如简单的陈述)。
- 有固定的说话者和受话者参与。
- 一系列固定的背景假设也是很重要的。

最后一个属性是篇章的语气要保持不变。例如,在一个片段内部,篇章不能从描述一系列的 actual 事件转移到描述一个假设事件。

一般来说,一个片段需要具有一定的结构。通常,有两种方法来刻画一个片段定义的内容。意图观认为片段中的所有句子都是围绕一个共同的篇章目的;也就是说,相同的交流目的促使说话者说出这个片段中的每一个句子。信息观认为片段中的所有句子彼此之间是通过时间、因果和修饰成分相互联系起来的。例如,一个片段中的描述性句子应该结合在一起描述一个连贯的事件或场景。

这两个定义之间常常有紧密的对应关系。例如,在大部分描述中,作者的篇章目的与信息层次分析有密切的关系。考虑下面这个故事的开始:

- 2a. Jack shopped early in the day. (Jack 很早就去购物了。)
- 2b. He took his car (他开车去的,)
- 2c. and he bought a dozen live lobsters. (并且他买了一打活龙虾。)
- 2d. When he got home, (当他回到家时,)
- 2e. he spent the day preparing the feast. (他花了一整天的时间来准备这个宴会。)

图 16.1 在信息层次和意图层次对这一篇章做了分析。信息层次趋向于描述篇章的“精细结构”,主要是事件在因果和时间上是怎样联系起来的。而意图层次则倾向于根据相关故事的交流目的来描述更全局的结构问题。

描述的事件	信息关系	交流目的
事件 1: Jack goes to store (Jack 去商店)	事件 2 是事件 1 的一部分	事件 1 作为故事的开始
事件 2: Jack drives car (Jack 开车)	事件 2 早于事件 3	详细描述事件 1
事件 3: Jack buys lobsters (Jack 买龙虾)	事件 3 是事件 1 的一部分	详细描述事件 1
事件 4: Jack gets home (Jack 回家)	事件 4 为事件 5 提供了时间背景	详细描述事件 1 后的故事
事件 5: Jack prepares for feast (Jack 准备宴会)	事件 5 在事件 4 的后面, 事件 4 使事件 5 成为可能	详细描述事件 4 后发生的故事

图 16.1 信息关系与交流目的的比较

尽管一个句子似乎总是存在意图层次的分析,但在篇章情境中不一定存在信息层次的分析。例如,在篇章 1 中,句子 1c 没有任何与句子 1a 和 1b 内容相关的信息,而是引入了一个新的话题。在意图层次上,这可以分析为变化到一个新的话题。这样的例子让研究人员认为意图分析是主要的。但事实上,两个层次的分析都是必要的。至于哪一个方法更重要,与要研究的篇章的形式有关。在对话和争论中,基于意图层次的分析提供的信息更多;另一方面,在陈述和描写篇章中,基于信息层次的分析似乎更合适。这两种方法中的每一种都为篇章提供了有用的分析,任何一个都不能代替另一个。

有了这一背景,我们可以更详细地探讨切分的问题。我们定义片段作为子句的序列,一个片段又可以分为几个子片段,从而形成一个层次化的结构。有些观点认为,每一个子句本身构成一个原始片段,这些原始片段可以形成更大的片段。但是,考虑到后面阐述的原因,我们将

不采用这一观点。即使在合适的情况下,一个子句可以自己构成一个片段,它也无须这么做。

片段的最重要方面是其自身有一个层次结构。句子 1i 中的代词“it”不能指代 1e 中的“gas can”的原因就来自于该事实。因为句子 1i 不在 1c ~ 1h 定义的片段中,对句子 1i 来说,“gas can”不能作为 1i 的篇章实体。实际上,句子 1a, 1b 和 1i 定义了一个片段。来自于这个篇章的历史记录列表能够正确预测句子 1i 中“it”的先行词。

这个例子表明了篇章片段的一个重要功能,即为指代表达式的解释定义了局部上下文。然后,层次结构控制不同的局部上下文的有效性,这些局部上下文可以用来处理当前的句子。

片段的第二个功能是以某种形式来组织要表达的信息,这种形式要有助于识别新句子和前面篇章之间的联系。为了支持这一识别过程,对由推理过程构造的每个片段的语义内容,必须有某种表示方法。

每一个片段与局部篇章状态(或简单篇章状态)之间都是具有联系的,局部篇章状态至少包括以下内容:

- 这个片段中的这些句子。
- 局部篇章上下文。这一篇章上下文是利用第 14 章中的方法从这个片段的一些句子中生成的。
- 这个片段中的句子的语义内容以及使该片段连贯起来的句子之间的语义关系。

一个完整的篇章将包括很多片段。图 16.2 通过文本框的形式表示了对话的片段结构。一个片段包括了出现在其内部的片段。同样的信息可以用树的形式表示,如图 16.3 所示。

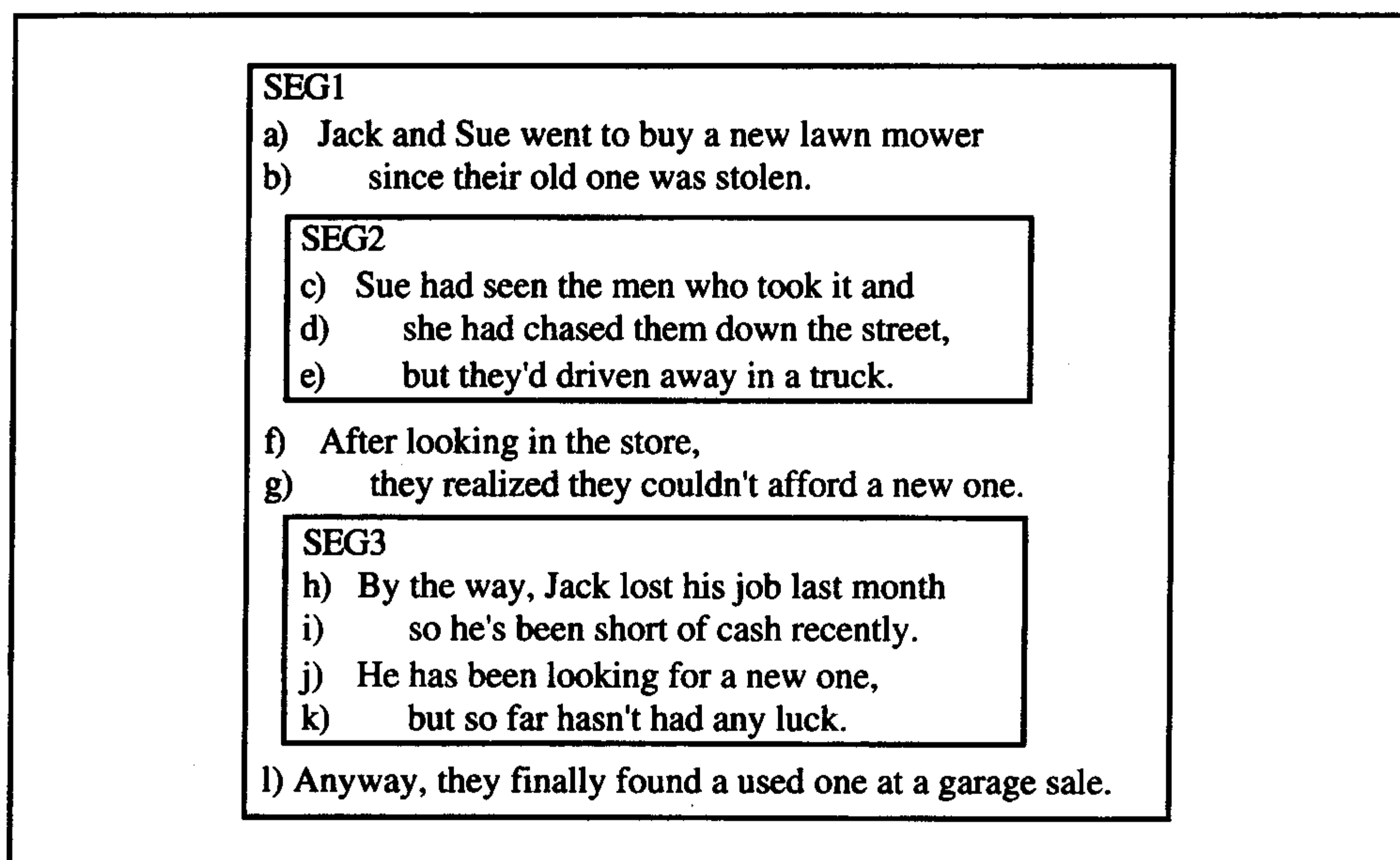


图 16.2 以文本框形式表示的片段层次结构

考虑在线算法时(即,按照顺序一句一句地理解篇章),需要考虑另外一些问题。这一处理模型必须表示为对现有 agent 篇章表示模型的一种带新句子的扩展,以建立这个篇章的更新的

表示。我们称这种表示为关注栈(或篇章栈),因为这种表示方法反映的是,为理解下一个句子,这个 agent 目前关注的内容。

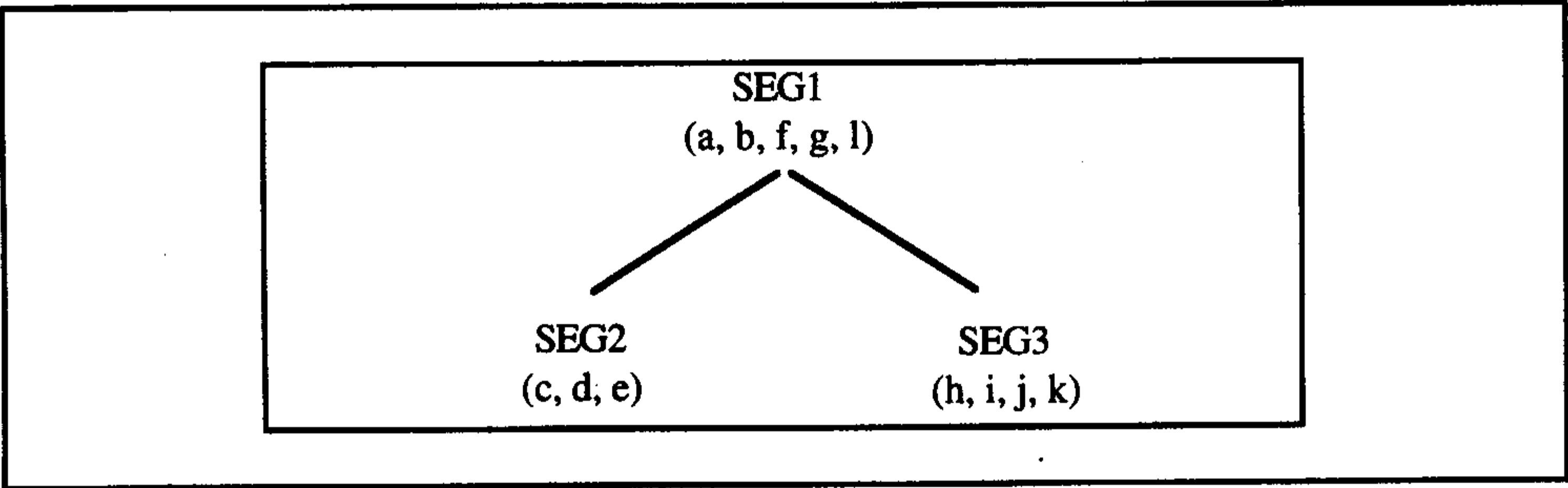


图 16.3 以树的形式来表示的同一个片段层次结构

关注栈由篇章状态构成,其中,篇章状态反映了正在进行篇章的当前结构。篇章栈的所有状态构成了可以被下一个子句所扩展的片段的一个集合。栈顶状态是内嵌最深的可以被扩展的片段。栈的每一个状态都与这样的片段相对应,每一个片段都包含了关注栈中位于这个状态上面每一个状态所对应的片段。为了处理篇章中的一个新的片段,需要往栈中压入新状态。要扩展一个与篇章栈中的较低的状态相对应的片段,需要将这个状态上面的所有状态都从栈中弹出来。这看起来复杂,实际上这个栈模型很简单。具体来说,考虑这个篇章栈的序列。其中,每个栈都位于一个子句的后面,因此这个序列与对这一篇章片段的树进行深度优先遍历的过程类似。图 16.4 简单表示了图 16.2 和图 16.3 所示篇章的篇章栈的序列。为了表示篇章状态与对应片段之间的关系,篇章状态用片段的名称来标记。其中,片段名字后面跟着这个片段中到目前为止的子句列表。这为一个篇章中的每一个篇章状态提供了惟一的名称。例如,篇章状态 SEG1(a,b)与片段 SEG1 相对应,其中子句 a 和子句 b 已经处理完了。片段的名称也应该包括完整的子片段(为了节约空间,图中没有表示出来),以便能够从多个子片段中跟踪某一个片段的状态。这样,子句 f 后面的这个状态的完整名字应该为 SEG1(a,b,seg2,f)。

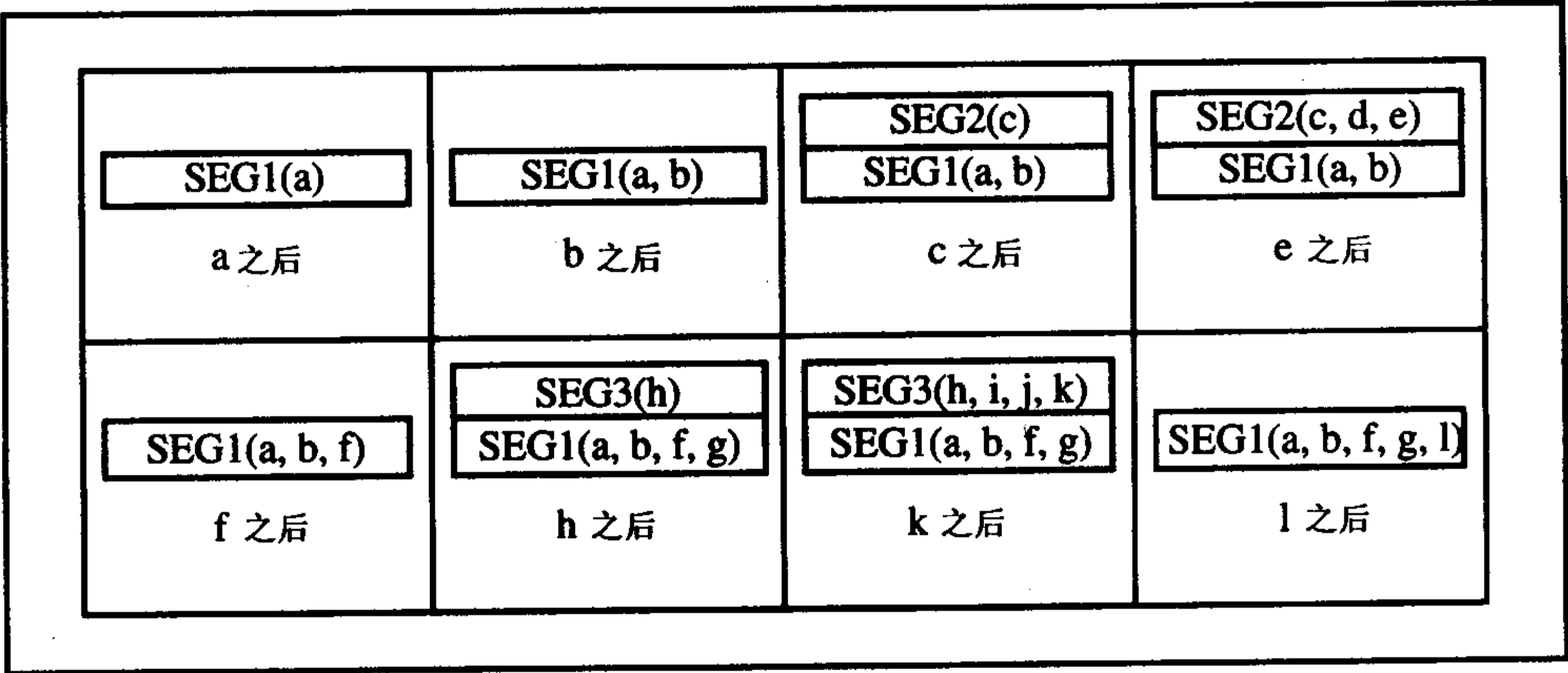


图 16.4 同一篇章的篇章栈序列的一部分

篇章结构最重要的标志之一是利用提示语来表示下一子句与前面篇章之间的关系。基于不同的研究目的,不同的研究人员使用了不同的提示语集合。不过,根据提示语表示的内容不同,它们大体上可以分为两个大类。第一类表示子句或状态之间的语义关系;第二类直接表示篇章结构,而不识别语义关系。

为了介绍第一类提示语,来看下面的两个句子:

Jack went to the store. Sam stayed home.

(Jack 去商店了。Sam 待在家里。)

正如你所看到的,这两个句子之间除了隐含的时间重叠外,没有明显的关系。但是,可以加入一些词,从而明确表示这两个事件之间的关系。

例如,我们可以表示 Jack 去商店的原因是因为 Sam 待在家里:

Jack went to the store because Sam stayed home.

(Jack 去商店了,因为 Sam 待在家里。)

或者表示因为“Jack”去商店了,所以“Sam”必须待在家里:

Jack went to the store. So Sam stayed home.

(Jack 去商店了,所以 Sam 待在家里。)

Jack went to the store. Therefore, Sam stayed home.

(Jack 去商店了,所以 Sam 待在家里。)

或者表示“Sam”待在家里,但是事实上你希望他没有待在家里:

Jack went to the store but Sam stayed home.

(Jack 去商店了,但是 Sam 待在家里。)

Jack went to the store. However, Sam stayed home.

(Jack 去商店了,然而 Sam 待在家里。)

或者表示这两个事件都是某事件的证据:

Jack went to the store. Furthermore, Sam stayed home.

(Jack 去商店了,此外, Sam 待在家里。)

或者表示两个事件之间特定的时间关系:

Jack went to the store. Meanwhile, Sam stayed home.

(Jack 去商店了,同时, Sam 待在家里。)

第二类提示语直接表示了篇章结构,而没有指出语义关系。这些提示语常常表示片段的边界。其中包括可以表示结束当前话题的短语(例如 OK, fine),结束一个篇章的短语(例如 bye, thanks),表示转移到其他主题的短语(例如 by the way, incidentally),或者表示题外话结束的短语(例如 anyway),或者表示一个特定篇章的组织的短语(例如逐条记录, first, second, next, last)。图 16.5 列出了这两类中的一些提示语。

提示结构的一些提示语	典型用途	提示语义关系的一些提示语	典型用途
anyway	题外话的结束	and	继续
by the way	题外话的开始	because	原因
bye	结束对话	but	对照
first	引入子话题 (分项列举)	furthermore	新的子话题
incidentally	题外话的开始	however	对照
last	新的子话题 (分项列举)	meanwhile	新的话题 (同时)
next	新的子话题 (分项列举)	so	结论
now	引入子话题	then	因果/时间性
OK	结束话题	therefore	总结
		though	对照

图 16.5 一些提示短语及其用途

16.3 篇章结构和指代

本章刚开始时所举的例子用指代问题来说明需要一个层次化的篇章结构动机。前面一节介绍的关注栈为解决这一问题提供了一种机制。题外话能够创建一个新的篇章状态,当把这一状态压入栈后,会暂时隐藏原来的篇章状态。当这个题外话结束时,这一状态会从栈中弹出,又可以回到原来的状态。让我们更详细地探讨这个例子。图 16.6 表示了句子 1b(“Now attach the rope to the top of the engine.”)结束时的篇章栈。当前篇章实体及其属性是这个篇章的局部篇章状态的一部分。具体来说,实体 R1(the rope), T1(the top)和 E1(the engine)可以用于随后的指代,R1 是首选的下一个中心。在句子 1c 中,提示语“by the way”表示出现了题外话,所以一个新的篇章状态被压入栈中,这个新的状态被句子 1d ~ 1g 扩展,结果栈如图 16.7 所示。篇章实体描述了新的汽油桶 G3(gas can)和拖拉机 T2(tractor)。句子 1h 中的提示词“OK”指示题外话的结束,篇章状态 SEG2 从栈中弹出,结果如图 16.8 所示的篇章栈。注意,这个状态与图 16.6 中句子 1b 后产生的状态相同,这样,就可以处理句子话语 1i 了。正如我们所料,代词“it”指代 R1。它不可能指代汽油桶 G3(gas can),这是因为那个篇章实体 G3 已经不在篇章状态中了。

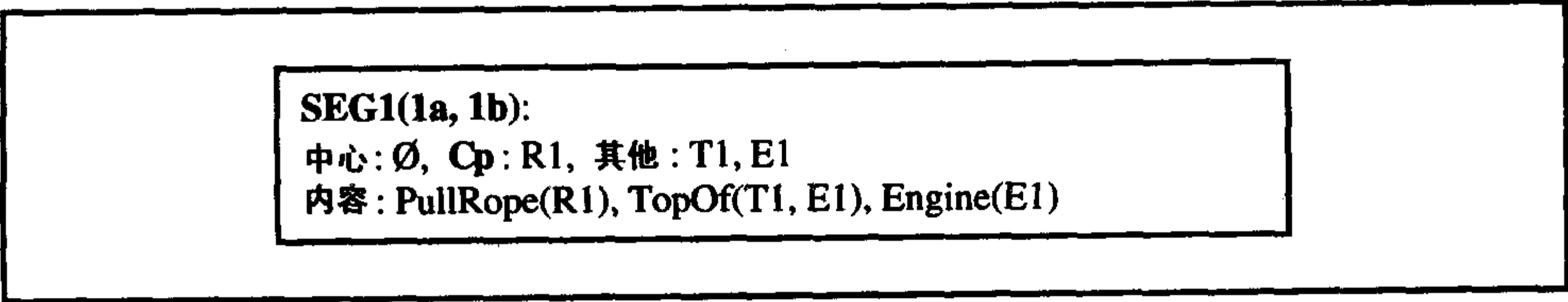


图 16.6 句子 1b 后的篇章栈

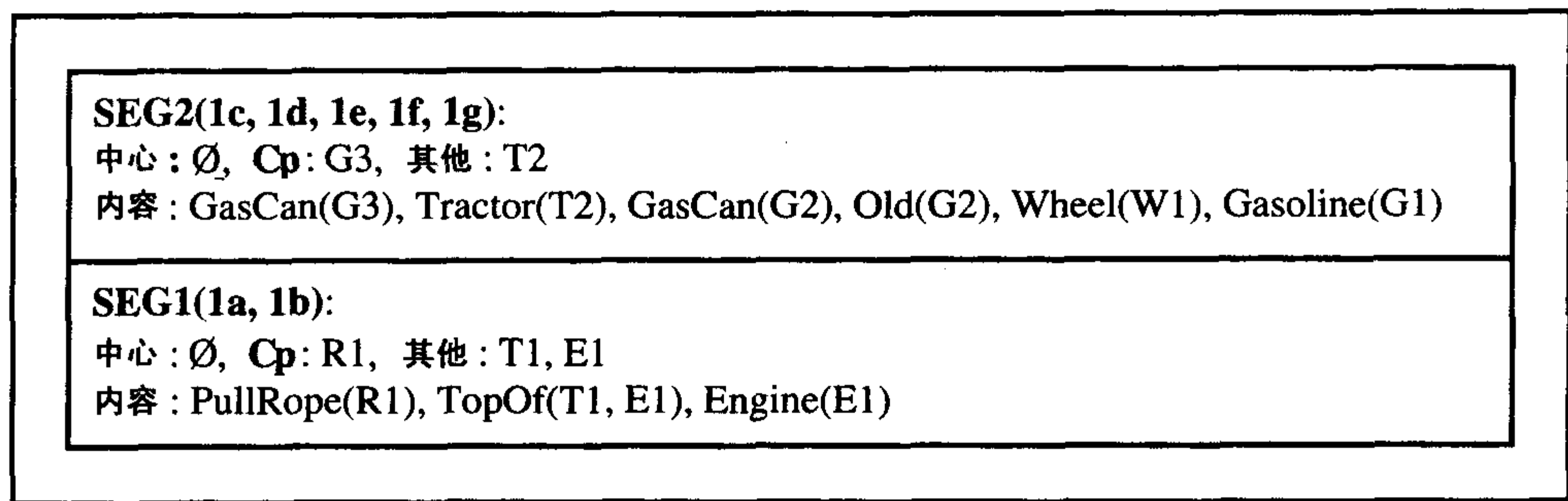


图 16.7 句子 1g 后的篇章栈

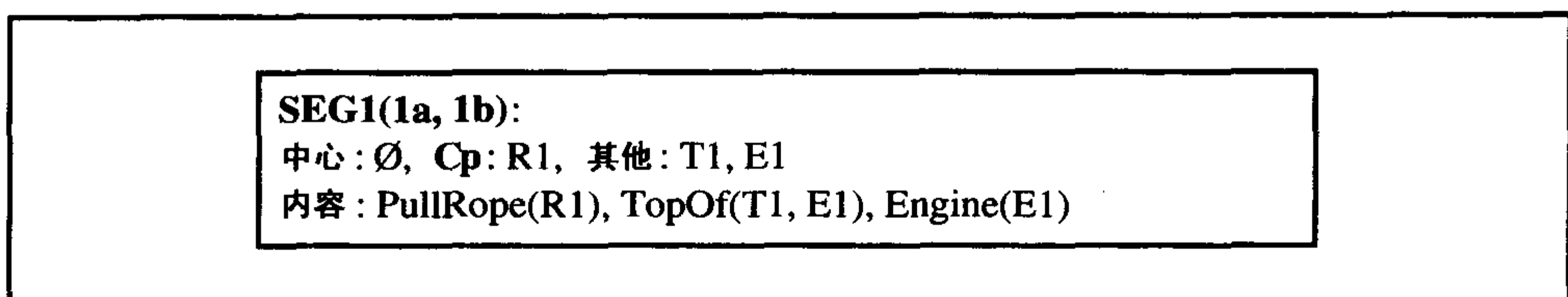


图 16.8 句子 1h 后的篇章栈

在大部分关于篇章结构的论述中,代词具有重要的作用。正如第 14 章所述,这主要是因为对先行词出现的位置具有很强的约束。具体来说,在大多数情况下,先行词在前面的句子中,并受最近上文的约束。这就是为什么这些例子(如句子 1i 中的代词“it”)中具有这样的问题。层次化的篇章模型为这个问题提供了一个完美的解决方案,这个方案与“最近期的信息最重要”这样的直觉相符合。

题外话的分析相当直观,因为典型的提示语表示了片段的边界。其他形式的篇章也具有清楚的片段结构。例如,逐条记录常常用提示语来明确地表明它们的结构,看看下面的篇章:

- 3a. There are many ways to identify a silver maple leaf.
(有很多方法可用来鉴别银槭树的叶子。)
- 3b. First, it has a silvery sheen on the back.
(首先,叶子背面有银色光泽。)
- 3c. If you hold it in your hand and move it,
(如果把叶子放在手中并移动它,)
- 3d. you will see the sun reflect off the back.
(你将看到从叶子背面反射的太阳光。)
- 3e. Second, it has deep, pronounced notches between the points.
(第二种方法,在叶子末端之间有明显的深槽。)
- 3f. The shape is quite similar to a red maple leaf.
(形状很像红槭树的叶子。)
- 3g. And third, if you break its stem, the sap will be milky.
(第三种方法,如果把叶柄弄断,树液是乳白色的。)

3h. Break the stem and wait about 20 seconds and the sap should be visible.

(把叶柄弄断,过 20 秒就可以看见树液。)

这个篇章有三个子片段。每一个片段的开始分别用提示语“first”,“second”和“and third”。每次当一个子片段被弹出,并生成一个新的子片段时,包含句子 3a 的这个片段的篇章状态将用于识别当前子片段中的“it”的合适的先行词。前面子片段中引入的篇章实体不再有效。例如,3g 中的代词“it”不是指句子 3f 中的“red maple leaf”,尽管这个短语在先前的句子中。这个篇章的片段结构如图 16.9 的树所示。

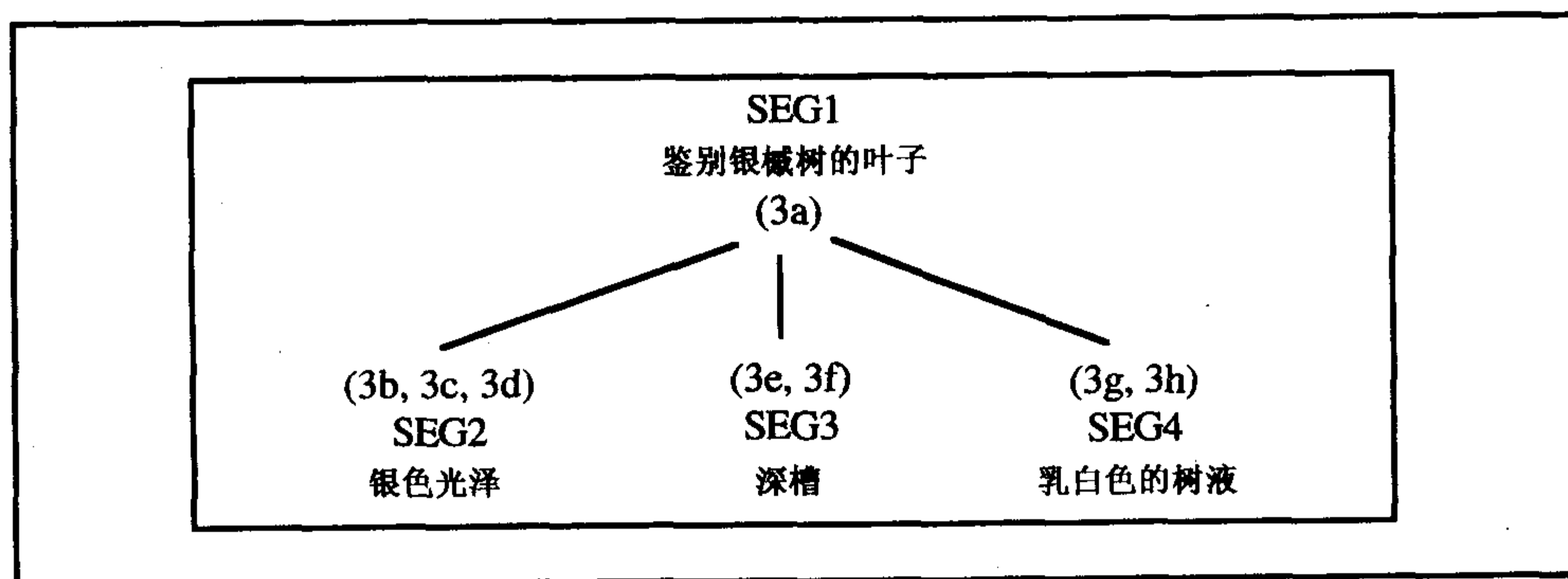


图 16.9 篇章 3 的结构

当一个篇章中没有明显的提示语,且话题的改变也不那么明显时,片段的检测就比较困难。有时候很难区分是否开始了一个新的片段,还是单个话题的自然延续。这样的问题在话题发生连续性进展的陈述性句子和故事中比较常见。这个故事可能描述一个较长时间内的系列事件,然而,从来没有显示明显的层次结构。如果没有出现从栈中弹出一个状态并返回到前面片段的情况,我们可能会认为整个故事只由单个片段所组成。

指代分析本身也能给出片段切分的一些提示。特别是,如果片段中用到的指代表达式需要状态栈低层的篇章状态才能找到其指代对象,将导致篇章栈中位于这个低层状态以上的片段都强制结束。例如,用下面的句子来扩展篇章 3:

3i. These three tests are all you need to know. (这三个测试是我们所需要知道的。)

在处理句子 3i 前,关注栈包括篇章栈 SEG1(3a, seg2, seg3)和 SEG4(3g, 3h)。这样,句子 3i 可以扩展状态 SEG4(3g, 3h),或者 SEG4(3g, 3h)被弹出并且句子 3i 能够扩展 SEG1(3a, seg2, seg3, seg4)。在句子 3i 中没有表示从栈中弹出的提示语,但是为了成功分析指代表达式,SEG4(3g, 3h)必须被弹出。具体来说,就是 SEG1(3a, seg2, seg3, seg4)为“These three tests”提供了篇章实体,“These three tests”指 SEG2, SEG3 和 SEG4 中介绍的三种方法。这 3 种测试方法可以存在的惟一上下文就是 SEG1(3a)。在这种情况下,我们可以说是指代分析导致了一个片段的弹出,因为指代分析在为指代表达式在上下文 SEG4 中产生一个合理的指代对象分析时失败了。

这个例子也引发了关于片段结构的另一方面的争论。让我们讨论是什么短语引发了句子 3i 中的名词短语“these three tests”所指代的篇章实体。事实上,这些篇章不是由某一个短语引发的。例如,第一个测试是由句子 3b, 3c 和 3d 描述的。这三个句子内容的综合体唤起了这个篇章实体。这表明,片段本身也引发能够用于后续指代的篇章实体。

16.4 篇章结构和推理的关联

正如前面所述,一个片段内的句子必须有局部的连贯性。根据不同的篇章形式,连贯性可以来自于因果联系(正如第 15 章详细探讨的)或者来自于其他关系,如争论或辩论中的证据和反面证据。无论用什么机制来建立局部连贯性,很清楚的一点是这个过程与篇章结构是紧密相关的。一方面,篇章的切分应该用来强调并引导联系的发现;另一方面,发现联系(或无法发现联系)能够对片段结构的判断发生影响。这一节探讨了篇章结构和推理之间的关系。

目前,我们已经探讨了关于片段切分的几个例子。当考虑推理过程时,这些例子又分为两类。篇章 1 说明了子片段是题外话的例子,这个例子中片段之间没有推理联系。每一个片段单元都是独立的,并有其自身的连贯结构。篇章 3 则给出了另一个极端,在篇章结构和对使用三种方法来鉴定银槭叶所做的表述之间具有一一对应的关系。

从现有的情况看,在篇章结构和推理结构之间似乎没有什么联系。然而,这两者之间有一个很重要的约束,它能够帮助决定什么样的切分结构是可能的。具体来说,一个句子一旦开始一个新的片段单元时,总会有一个歧义:到底是新的片段结束了前面的片段单元,还是这个片段是前面片段单元的子单元呢? 根据它对片段结构和篇章栈的变化影响,图 16.10 表示了这一选择。在此图中,每一个片段 SEG_i 的篇章状态记为 DS_i 。这个决定将影响什么篇章实体可以用来解决指代,以及这个新片段一旦结束后这个篇章是怎样继续的。

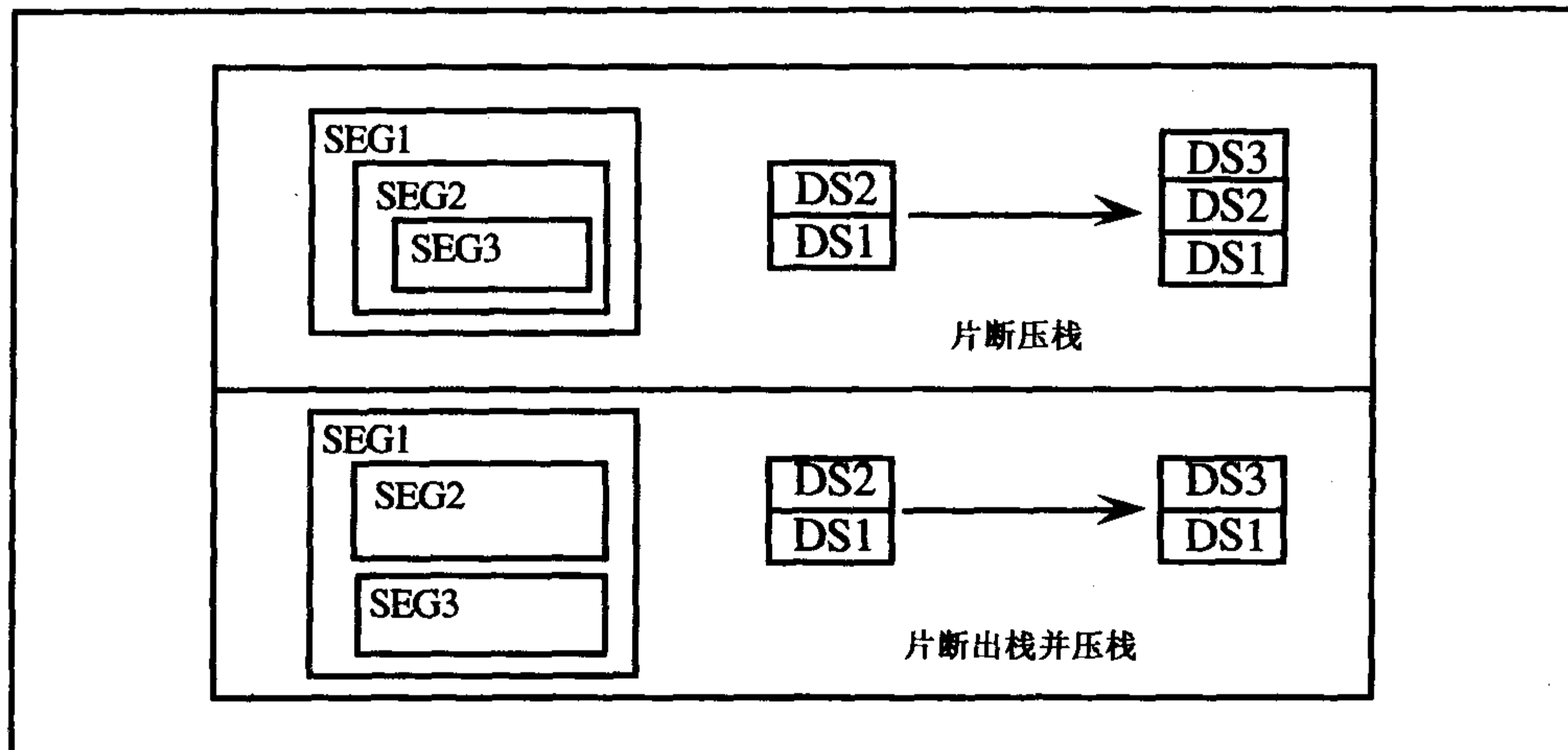


图 16.10 开始一个新片段的两种方法

不置可否的做法总是将新片段压入栈中,因此,允许前面的任何话题都可以再恢复过来。这一方法对于题外话和原来话题的中断是有效的,但是对类似于篇章 3 的篇章结构来说,这种方法与直觉不一致。如果 SEG3 是篇章 3 中的 SEG2 的子片段,则句子 3d 中的“sun”(光线)对后面的代词指代是可行的。在这个篇章中,提示语明显地表示了期望的结构,所以这个解释是不可能的。

遗憾的是,在很多情况下,没有明显的信息来表明哪一个解释是可行的。在这些情况下,这个区别只能建立在推理的基础上。具体来说,如果可以确定一个推理联系,我们也许可以知

道一个片段结束了,因为这个篇章已经转移到逻辑上的下一个话题。例如,在描述一些事件的篇章中,片段的压栈可能对应于事件之间的分解关系,而片段的出栈并压栈可能对应于变换到序列中的下一个事件。

很显然,并不是所有篇章都是描述事件的,层次结构对于很多不同形式的篇章都是有用的。为了把处理不同类型篇章的这些方法统一起来,我们首先简要介绍片段的篇章目的这一概念。虽然准确定义这一概念是很困难的,但是从直觉上来说还是很清楚的。其基本想法是,两个片段之间的约束来自于说话者说出这两个片段中的句子的原因。这种想法认为,当新的片段(与 DS3 对应)完成了当前片段(与 DS2 对应)目标的子目标时,就出现压栈操作。而另一方面,当新的片段不是前面片段目标的子目标,而是包含前面这个片段的更大片段的目标的子目标时,就发生出栈并压栈操作。至于什么样的目标适合作为篇章目的,是由篇章的类型来规定的。

例如,在对话目标是描述(可能较复杂)事件的这样一个篇章中,篇章目的的层次结构可能对应于事件分解的层次结构。特别是,篇章目的的形式都为“描述事件 X”时。如果事件 X 是事件 Y 的一部分,那么篇章目的“描述事件 X”是篇章目的“描述事件 Y”的子目标。

让我们来看另一种形式的篇章——辩论。这种篇章的目的是建立一些论点,子目标关系对应于证据性的支持。例如,如果我们的目标是建立论点 X,那么子目标也许是建立论点 Y。其中,Y 倾向于让其他人相信 X,也就是说,Y 为 X 提供了证据。由此,一个片段的压栈操作可能对应于这样的情形:新片段的篇章目的是为前面片段所阐述的某个论点提供支持。

考虑最后一个例子,假设篇章目的是描述一所住宅中的所有房间。这个篇章目的的层次结构可以反映房子的物理空间的布置。例如,描述客厅是描述住宅的子目标,描述客厅中的壁橱也是描述客厅的子目标。

有了这种从篇章的特定推理过程中总结出来的抽象形式,我们可以用通用的术语来描述篇章结构和推理之间的约束关系。当且仅当篇章目的 DP2(DP, discourse purpose)是 DP1 的子目标时,我们称之为篇章目的 DP1 支配另一个篇章目的 DP2。当且仅当 DP1 支配 DP2,并且不存在这样的篇章目的 DP3(DP3 是 DP1 的子目标,DP2 又是 DP3 的子目标)时,我们称之为篇章目的 DP1 直接支配篇章目的 DP2。

需要记住的重要一点是,不是所有的片段都必须以这些方式彼此联系。事实上,片段可以彼此之间没有联系,正如我们说话时的中断和题外话。所以,一个片段被包含在另一个片段中的事实并不意味着前面的第一个篇章目的必须支配第二个篇章目的。更准确地说,约束在反方向是成立的。当系统的推理模块确定一个支配关系时,对于可能的关注栈就施加了一个约束(同样也对片段结构施加了一个约束)。特别是下面的约束:

支配约束——如果与 DS1 关联的片段的篇章目的直接支配与 DS2 关联的片段的篇章目的,那么,如果 DS2 在关注栈上,则 DS1 在栈中的位置与 DS2 相邻且位于 DS2 的下面。

这一约束促使关注栈的更新与我们的直觉保持一致。例如,考虑下面的篇章(此篇章为篇章 3 中的提示语被删除后的修订版本):

- 4a. There are many ways to identify a silver maple leaf. (有很多方法来确定银槭树的叶子。)
- 4b. It has a silvery sheen on the back. (叶子背面有银色光泽。)
- 4c. If you hold it in your hand and move it, (如果把叶子放在手中并移动它,)

- 4d. you will see the sun reflect off the back.
 (你将看到从叶子背面反射过来的太阳光。)
- 4e. It also has deep, pronounced notches between the points.
 (在叶子末端之间,还有明显的深槽。)
- 4f. It is quite similar to a red maple leaf.
 (它很像红槭树的叶子。)
- 4g. If you break its stem, the sap will be milky.
 (如把叶柄弄断,树液是乳白色的。)
- 4h. Break the stem and wait about 20 seconds and the sap should be visible.
 (把叶柄弄断,过 20 秒就可以看见树液。)

删除了提示语使这一篇章难以理解,但它还是可以理解的。期望的片段结构仍然如图 16.9 所示。即,篇章的最高一层片段是 SEG1,它有三个子片段 SEG2(由 4b,4c 和 4d 组成),SEG3(由 4e 和 4f 组成)和 SEG4(由 4g 和 4h 组成)。在这种场景下,篇章目的就是传达怎样执行识别槭树叶子的任务,以及支配关系是怎样由任务/子任务之间的关系来定义的。假定一个合适领域内的推理系统,即 SEG1 的篇章目的直接支配 SEG2,SEG3 和 SEG4 的篇章目的。那么这一支配约束惟一确定了篇章栈的压栈和出栈的内容。图 16.11 显示了句子 4b 的篇章栈的过程。给定支配约束,没有其他可能性。特别是,篇章状态 SEG1(4a)不能从栈中弹出,因为它的篇章目的支配了以句子 4b 开始的新片段的目的是。

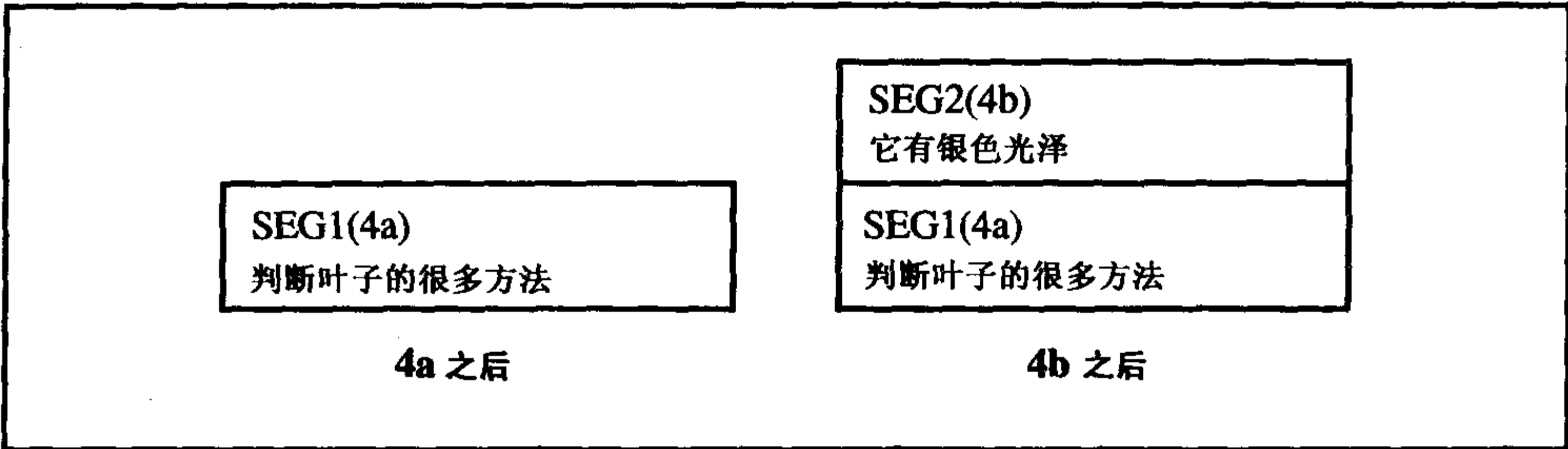


图 16.11 给定句子 4b 的关注栈的更新

图 16.12 显示了句子 4e 导致的关注栈的状态变化。如果特定领域内的推理系统能够判断出 SEG1 直接支配新的片段 SEG3,将只有一个转移能够满足这个支配约束。状态 SEG2(4b, 4c,4e)必须从栈中弹出,以便 SEG3(4e)能够直接位于 SEG1(4a, seg2)之上。

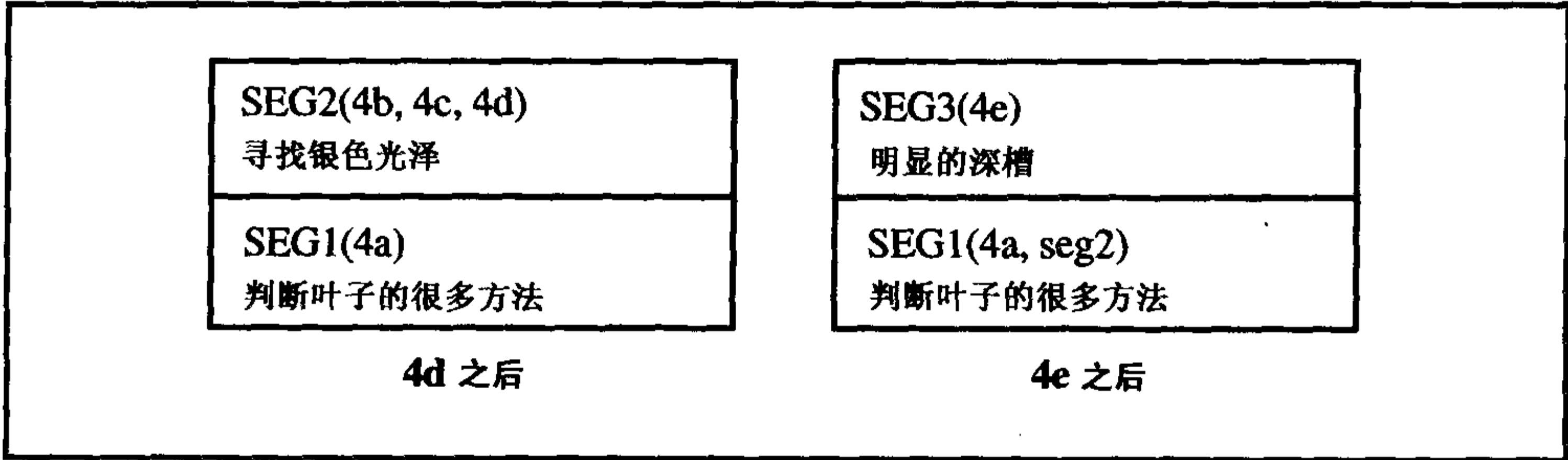


图 16.12 给定句子 4e 的关注栈的更新

这样,要在一个特定的应用中利用这个模型,必须首先确定在这个领域内是什么推理联系导致了篇章片段目的之间的直接支配关系。一旦确定了这一点,支配约束将可以利用推理过程来限制可能的篇章结构和关注栈的状态变化。

篇章目的方法的另一个优点是能够在某种程度上对那些有缺陷的篇章进行分析。例如,可能是槭树叶子没有银色的光泽,是说话者弄错了。不过,篇章分析仍然一样,因为即使它们建立在错误的信念上,篇章目的仍是一样的。当然,为了识别这样的对话结构,我们必须能够识别想要达到的目的,即使是建立在错误的信念上。一般来说,这是一个很困难的问题,但是这个理论为进一步研究留下了空间。需要注意的是,在由提示语来明确提示结构的篇章3中,约束也可以反向使用。在这里,系统可以利用支配约束从篇章结构中推理得到说话者认为槭树叶子背面有银色的光泽,因为这是由篇章结构得到的最合理解释。

16.5 篇章结构、时态和体态

一个篇章片段内的时态和体态提供了与事件相关的丰富的信息,也提供了可以确定片段边界的约束。在这一节中,我们将主要研究时态和体态是怎样同时影响片段切分和(用来推导出建立连贯性所需要的联系的)推理过程的。

首先,考虑单个片段中时态和体态的影响。在单个片段中的简单陈述中,如果句子序列描述一系列事件,那么事件发生的顺序与被描述的顺序是一致的。我们有时候称之为陈述惯例。例如,下面的篇章:

5a. Jack went to the store. (Jack 去了商店。)

5b. He bought some roses. (他买了一些玫瑰花。)

从直觉上理解,Jack 先去商店(“going to the store”),然后买了一些玫瑰花(“bought the roses”)。一些研究人员建议,通过考虑最近被描述的事件后的那些事件的期望,来利用这个约束改进推理算法。但是,这一情况只有在非常简单的情况下才有效。例如,下面的篇章:

6a. Jack went to the store. (Jack 去了商店。)

6b. He walked there along the river. (他是沿着河边走到那儿去的。)

在最自然的解释中,沿河岸走(“walking along the river”)这一事件是去商店(“going to the store”)这个事件的一部分。又如下面的篇章:

7a. Jack showed us his new car. (Jack 给我们展示了他的新车。)

7b. He bought it at Honest John's Auto Mart. (他是在 Honest John 的汽车市场买的。)

在这种情况下,句子 7b 中描述的事件很明显早于句子 7a 描述的事件。

在过去时态的两个句子所描述的事件之间,任何关系好像都为真,而事实上不是这样。这种关系只局限于少数几种特定的因果关系和一种默认的时间关系。我们用一种称为适应关系的新关系来刻画这个事实。在同一个片段中的任意两个连续的事件将有适应关系。定义这个适应关系的第一个合理的尝试方法如下:

如果一个事件 E_1 适应一个事件 E_2 ,那么,

1. 如果 E_2 是 E_1 的一部分,那么 $E_2 \subseteq E_1$ (即 E_2 发生在 E_1 的过程中)。
2. 如果 E_2 是 E_1 能够发生的条件,那么 $E_2 < : E_1$ (即 E_2 在 E_1 之前发生)。
3. 否则, $E_1 < : E_2$ (即 E_1 在 E_2 之前发生)。

条件3给出了默认情况,刻画了陈述惯例。

框 16.1 意图结构的明显标志

有一大类提示语能够明确地提示一个篇章的意图结构,这样,这些提示语就给出了片段的切分。前面,我们已经看到了一个例子,称为“itemization”(逐条说明),其中“first”,“second”等都明确指示了一些子片段序列,其目的是详细描述这个包容性片段的总体目的。其他例子如:

详述结构:创建一个新的子片段以更好地定义被嵌入片段的目的是。下面的提示语表示了一些类型的详细描述:

“*in particular*”——子片段通过更具体的讨论来支持被嵌入片段的目的是。

“*in addition*”——子片段通过补充解释来支持被嵌入片段的目的是。

“*for example*”——子片段通过具体的例子来支持被嵌入片段的目的是。

“*in general*”——子片段通过概括来支持被嵌入片段的目的是。

对比结构:这种结构把表述同一个问题(有时候,从不同的角度来阐述)的一些片段组织在一起。这些提示短语包括:

“*on one hand ... on the other hand*”——子片段给出了某一个论断或观点的两个方面。

“*in contrast*”——子片段描述与前面片段相反的信息。

“*similarly*”——子片段描述与前面的片段相一致的一些信息。

处理篇章的一些方法把诸如“*elaborate*”(详述),“*exemplify*”(举例),“*contrast*”(对比)等一些关系当做篇章结构的基本构成模块,并且只有每一个片段之间都存在这些关系时,这个篇章才被定义为连贯的。两个典型的例子是 Hobbs(1979)的连贯关系方法以及 Mann 和 Thompson(1986)中的修辞结构理论。

但是篇章不仅仅描述事件,必须扩充适应关系以处理其他情况,比如状态。例如,如果一个描述状态的句子后面紧跟一个描述事件的句子,那么在状态保持过程中,事件会发生。这种情况在篇章的直觉理解中非常常见。

8a. Jack was at the store. (Jack 在商店。)

8b. He bought some roses. (他买了一些玫瑰花。)

在这个例子中,Jack 在商店时买了一些玫瑰花。但是,下面的例子说明,当给定适当的因果联系时,还有其他可能。

框 16.2 进行体

进行体引入了另外一些问题,例如下面的篇章:

- a1. Jack was going to the store. (Jack 正在去商店。)
- a2. He bought some roses. (他买了些玫瑰花。)

从这个篇章中,如果我们认为 Jack 先到达商店,然后买了一些玫瑰花,将是不合情理的。应该是他在去商店的路上买了这些玫瑰花。这与对状态的默认解释相同。进行体描述了一个进行中的事件,由此,这个句子的事件时间是所描述事件的时间的一个子区间。并且,句子 a1 的事件时间 E_1 是事件 E_{go} (“Jack 去商店”)的时间的一个子区间。所以,下一个句子的事件时间很倾向于在 E_{go} 中,因为它与 E_1 相关。

通常,特定的因果关系也可以产生其他的解释:

- b1. Jack was walking home. (Jack 在走路回家。)
- b2. His car broke down at work. (他的车坏了。)

这里,b2 中描述的事件是 a2 描述事件的原因。另一个例子,c2 中的事件描述了 c1 中的事件的结束:

- c1. Jack was walking home. (Jack 在走路回家。)
- c2. He arrived home with blister on his feet. (当他到家时,双脚都磨出了水泡。)

在下面的例子中,状态可以使事件发生:

- 9a. Jack had five dollars. (Jack 有五美元。)
- 9b. He bought some roses. (他买了一些玫瑰花。)

在这种情况下,事件紧跟在状态后面。还有一些情况下,事件早于状态,正如:

- 10a. Jack had some roses. (Jack 有一些玫瑰花。)
- 10b. He bought them at the store. (他是在商店买的这些花。)

因为句子 10a 描述了句子 10b 中事件的结果,因此出现了这个解释。对这些事件,当状态和事件之间没有因果联系时,我们可以假定一个默认的解释;但在状态和事件之间有因果联系时,句子的解释由发现的因果关系来确定。更准确地说,我们可以定义状态和事件之间的适应关系,如下:

如果状态 S_1 适应事件 E_2 ,那么,

1. 如果 S_1 使 E_2 发生,那么 $S_1 < : E_2$;
2. 如果事件 E_2 导致状态 S_1 ,那么 $E_2 < : S_1$;
3. 否则, $E_2 \subseteq S_1$ 。

一般来说,对状态/事件句子对中的句子顺序进行改变并不改变篇章的解释。让我们看一下篇章 11,它与篇章 8 的解释相同:

- 11a. Jack bought some roses. (Jack 买了些玫瑰花。)
- 11b. He was at the store. (他在商店。)

当然,也有特例。当两个连续的句子都描述状态时,这两个状态同时发生,如下面的篇章:

- 12a. Jack had some roses. (Jack 有一些玫瑰花。)
- 12b. He was happy. (他很高兴。)

其中,Jack 有玫瑰花和高兴这两个状态是同步的。图 16.13 总结了状态和事件的各种组合之间的适应关系。

X_1 适应 X_2	事件 E_1	状态 S_1
事件 E_2	默认: $E_1 < E_2$ 其他可能: $E_2 \subseteq E_1$ (分解) $E_2 < E_1$ (使能)	默认: $E_2 \subseteq S_1$ 其他可能: $S_1 < E_2$ (使能) $E_2 < S_1$ (因果)
状态 S_2	和 S_1, E_2 一样	默认: $S_1 = S_2$

图 16.13 对适应关系的这些结果的总结

到目前为止,尽管时态的解释提供了有用的信息,可以有助于在搜索期望并发现推理性联系时集中搜索范围,但它好像对片段边界的判断没有多大帮助。然而,当考虑更复杂时态的相互作用时,我们会发现一些约束在很大程度上是由时态而不是因果知识决定的。例如下面的篇章:

- 13a. Jack went to the store. (Jack 去了商店。)
- 13b. He had bought some roses. (他已经买了些玫瑰花。)

我们只考虑买玫瑰花(“buying the roses”)这一事件早于去商店(“going to the store”)事件的解释。虽然篇章 7 中的因果知识提示玫瑰花是在商店买的,但是句子 13b 中的过去完成时的使用却排除了这一解释,从而迫使时间上的回跳。

很多篇章中时态的解释都对时态采用 Reichenbachian 形式的表示,如 13.5 节所述。在这一表示中,每一个时态子句 C_i 都定义了三个时间:说话时间(speech time, S_i)、事件时间(event time, E_i)和参照时间(reference time, T_i)。子句的时态决定了这三个时间之间的关系。在一个包括 n 个子句 C_1, \dots, C_n 的篇章中,我们也可以知道说话时间形成了一个序列,即:

$$S_1 < S_2 < \dots < S_n$$

但是,这个信息自身并没有为序列中所描述的事件之间的时间性的关系提供任何线索。具体来说,在篇章 13 中,第一个子句是简单过去时,所以我们知道 $E_1 = R_1 < S_1$;第二个子句是过去完成时,所以 $E_2 < R_2 < S_2$ 。我们也知道 $S_1 < S_2$ 。把这些约束放在一起,得到:

$$E_1 = R_1 < S_1 \ \& \ E_2 < R_2 < S_2 \ \& \ S_1 < S_2$$

这些约束并没有蕴涵 E_1 和 E_2 之间的任何关系。

为了解决这个问题,研究人员提出了利用来自于篇章中的其他约束的方法。一种建议认为

参照时间是回指性的,必须要回指到前面篇章所引发的某个时间。先行词的参照时间有时候称为时间焦点(temporal focus),以便与篇章焦点或篇章中心相对应。特别是,如果把约束 $R_2 = R_1$ 添加到为篇章 13 引入的约束中,将与我们期望的一样,即蕴涵 $E_2 < E_1$ 。到目前为止,这种方法还是不错的。但是,如果前面的句子也是过去完成时,那就有问题了。如篇章 13 的续篇:

13c. He had wrapped them in fancy paper.(他是用花纹纸包花的。)

由句子 13c 得到的时间性约束是 $E_3 < R_3 < S_3$,篇章约束是 $R_3 = R_2$ 。将这一约束与来自句子 13b 的约束结合在一起,得到 $E_2 < R_2 < S_2$,但这并没有蕴涵说明 E_2 和 E_3 有任何联系(尽管,如我们的期望一样,二者均在 E_1 之前)。因此,不能得到我们根据直觉得出的判断,即 E_2 早于 E_3 。似乎这种直觉来源于默认的描述性解释,正如以简单过去时描述的这两个事件那样。如果是这种情况,因果知识能否像前面一样也超越这个假设呢? 答案是肯定的,从而期望的约束为 E_2 适应 E_3 。如下面的篇章:

14a. Jack took Helen on a date.(Jack 带 Helen 去约会。)

14b. He had impressed her with his new car the day before.
(约会的前一天,Jack 的新车让她印象很深。)

14c. He had bought it at Honest John's Auto Mart.
(他是在 Honest John 的汽车市场买的这辆车。)

这里,因果知识让我们得到的结论是,句子 14c 中的事件要早于(事实上是使能关系)句子 14b 中的事件。

什么结构能够解释过去完成时的这个特性呢? 一种方法是利用一个称为时态树(tense tree)的结构。一个时态树表示的是一个句子中时态运算符的序列。时态树上的每一个节点对应于时态中的一种不同的形式。图 16.14 显示了一个完全扩展了的时态树,其中刻画了常见的时态形式。时态树的根节点为简单现在时,时态树中的子节点分别表示过去时、完成时和将来时运算符。因为这些运算符是可以嵌套的,所以这个时态树可以向下扩充多次。

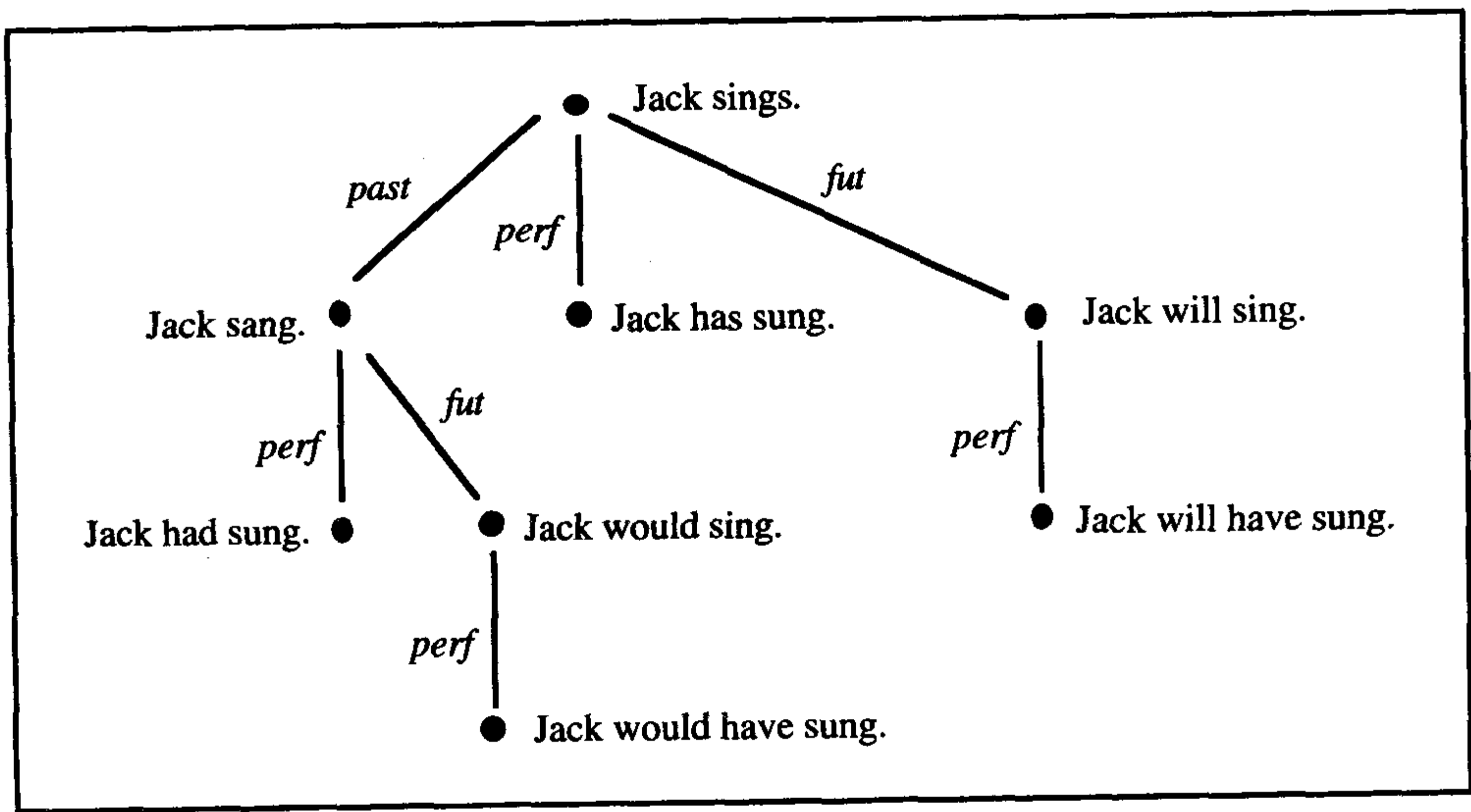


图 16.14 表示为一棵时态树的不同的时态形式

一个篇章中事件之间的这些关系可以根据这些事件的时态将每一个事件放置在时态树中的合适节点上来得到。一旦两个事件放置在同一个节点上,前面的事件就要适应新的事件。不同节点的事件通过一些约束联系起来,这些约束是由连接不同节点的运算符所产生的。例如,图 16.15 所示的篇章 5 和篇章 13 的时态树:

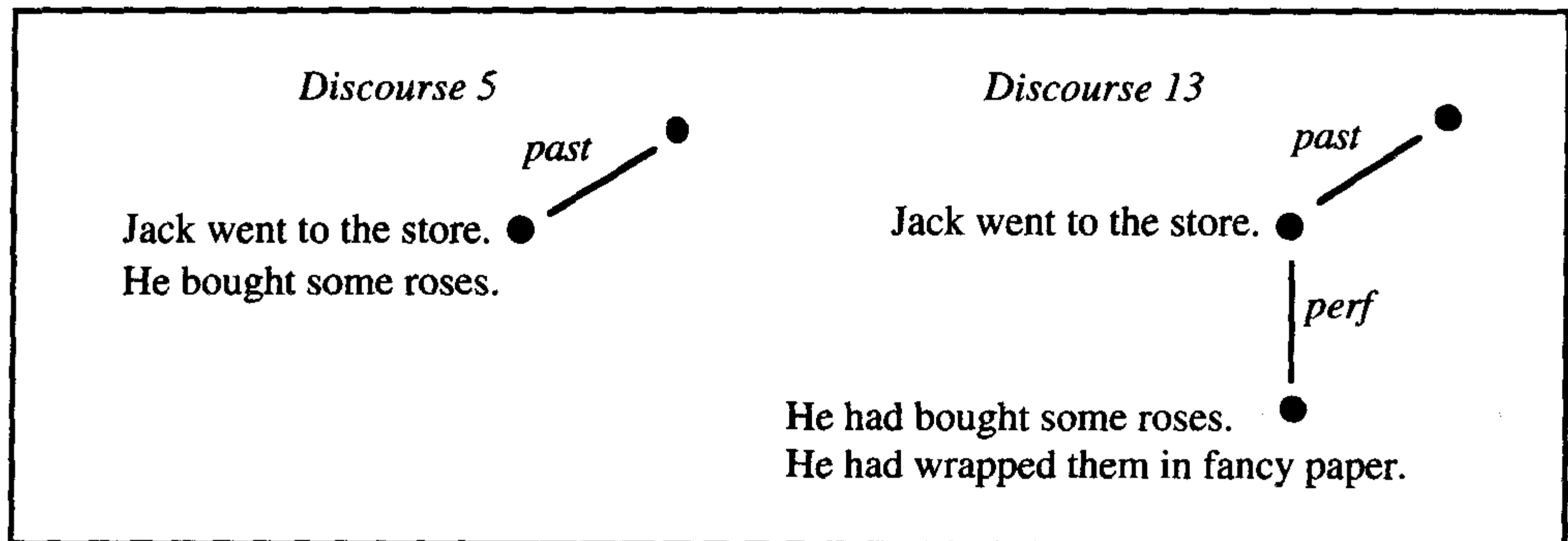


图 16.15 篇章 5 和篇章 13 的时态树

在篇章 5 中,两个事件都映射到同一个节点上,这是因为它们都是简单过去时。这样,第一个事件要适应第二个事件。假定没有因果关系,说明去商店(“going to the store”)这一事件要比买玫瑰花(“buying the roses”)这一事件早。在篇章 13 中,三个事件之间的关系如图所示。由于完成时“perf”的连接关系,我们得到事件买玫瑰花和包装玫瑰花都要早于去商店(“going to the store”)这一事件。另外,因为买玫瑰花事件和包装玫瑰花事件位于同一个节点,那么买玫瑰花这一事件要适应包装玫瑰花这一事件。因此,可以推理得到买玫瑰花这一事件早于包装玫瑰花这一事件。

给定这一框架,我们就可以探讨跨片段边界的时态之间的关系。初始假设就是表示一个片段的篇章状态将包括一个时态树,这个时态树用来在那个片段内的不同状态和事件之间建立起联系。需要注意的是,这一机制允许利用不同的时态将句子联系起来,所以,时态上的变化并不一定意味着片段的变化。但是,这一方法存在一个问题。例如,篇章 15 中有两个地方都使用了过去完成时:

- 15a. Jack went to Helen's house.
(Jack 去了 Helen 的家。)
- 15b. He had bought some roses.
(在去之前,他买了一些玫瑰花。)
- 15c. He dropped them on the carpet when he gave them to her.
(当他把玫瑰花给她时,把玫瑰花掉在了地毯上。)
- 15d. Helen had had the carpet cleaned, so she was upset.
(因为 Helen 刚打扫过地毯,所以她很心烦。)

如果这四个句子在一个片段内,也就是在一个时态树中,句子 15b 和句子 15d 中的事件将被指派到时态树中的同一个节点。这样的结果就是买玫瑰花这一事件(“buying the roses”)将适应事件 Helen 清洁地毯(“Helen cleaning the carpet”)。很显然,这两者之间没有任何联系。

如果每一个片段只与时态树中的一个节点相关,那么前面的分析结果就是可以避免的。

如果一个句子所对应的时态在时态树中的位置是由当前片段的时态转到另一个更低的时态,则对应一个压栈操作。如果一个句子的时态在时态树中的位置比当前片段的时态所处的位置要高,则是一个出栈并继续操作。基于这一思想,篇章 15 的分析在图 16.16 中显示。句子 15b 对应一个压栈操作,因为它的时态转移为过去完成时;另一方面,句子 15c 又回到简单过去时,对应一个片段 SEG1 的出栈并继续操作;句子 15d 也是过去完成时,对应压栈操作,创建了一个与 SEG2 没有联系的新的片段。这就是我们想要的解释。

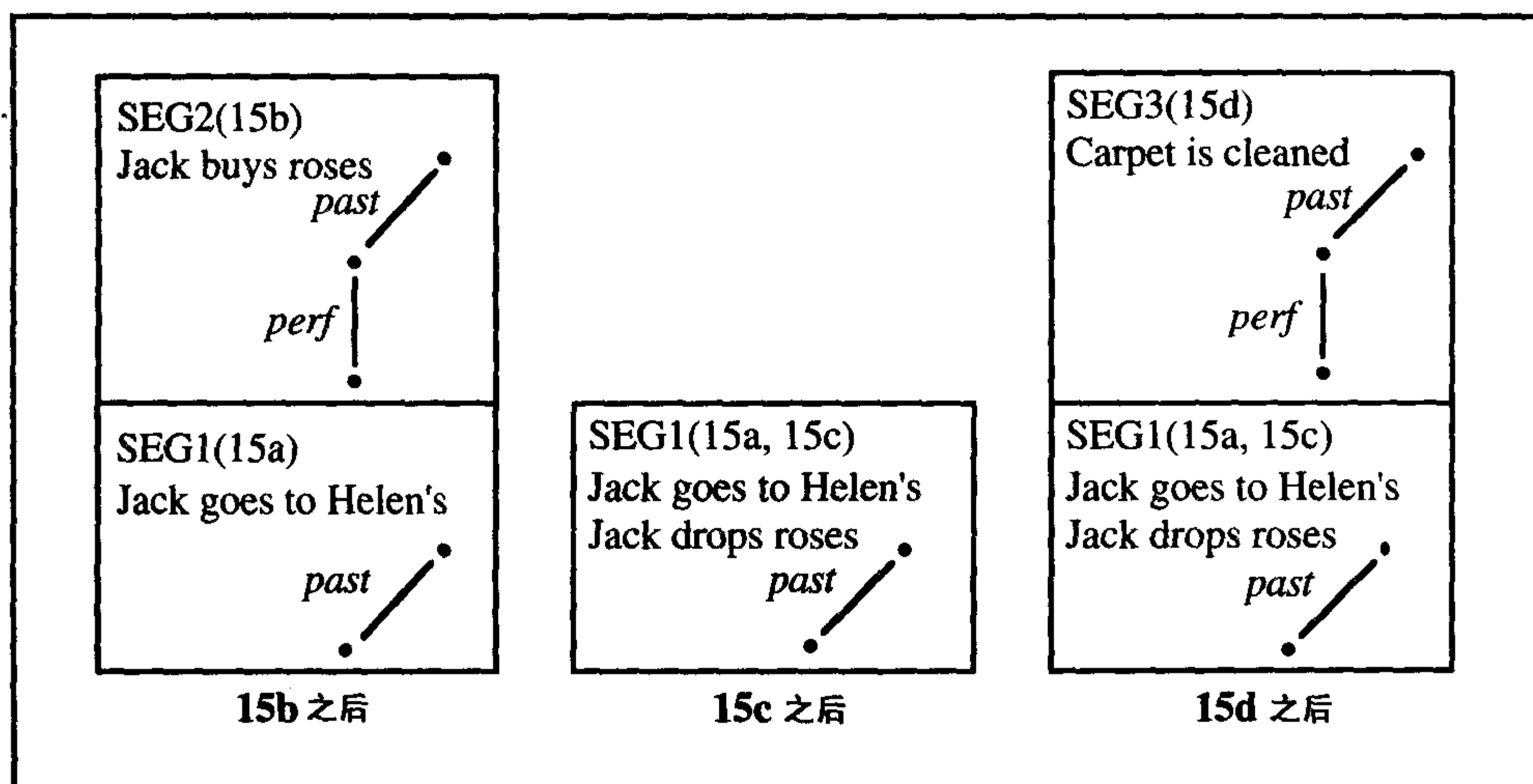


图 16.16 篇章 15 的时态树和切分片段

但是,跟踪时态并不总是那么简单。有时候,说话者利用过去完成时表示过渡到过去的某一个事件,然后再用简单过去时阐述那个事件,例如:

- 16a. Jack was walking over to Helen's house.
(Jack 在步行去 Helen 的家。)
- 16b. He had bought some roses at Honest John's Flower Mart.
(在去以前,他已经在 Honest John 的花市买了些玫瑰花。)
- 16c. He paid thirty dollars for them(他花了 30 美元)
- 16d. and Honest John swore they were fresh.
(Honest John 承诺这些玫瑰花是新鲜的。)
- 16e. But they wilted before he got to Helen's.
(但是,还没到 Helen 的家,这些玫瑰花就枯萎了。)

这里,句子 16b 表示一个新的片段,这个片段中的事件要早于 16a 中的事件。句子 16c 和 16d 又进一步阐述 16b 中的事件。但是这两个句子都是简单过去时,在句子 16b, 16c 和 16d 之间发生了时间视角的过渡。我们可以这样处理这个问题:允许修改和这个片段相关的时态树。句子 16c 后面的关注栈如图 16.17 所示。给定这个上下文,如果下一个句子是简单过去时,那么就无法判断是应该继续这个高层的片段还是重新开始底层的片段。这个问题只能通过推理过程才能确定下来,因为仅仅根据时态信息是无法做出判定的。

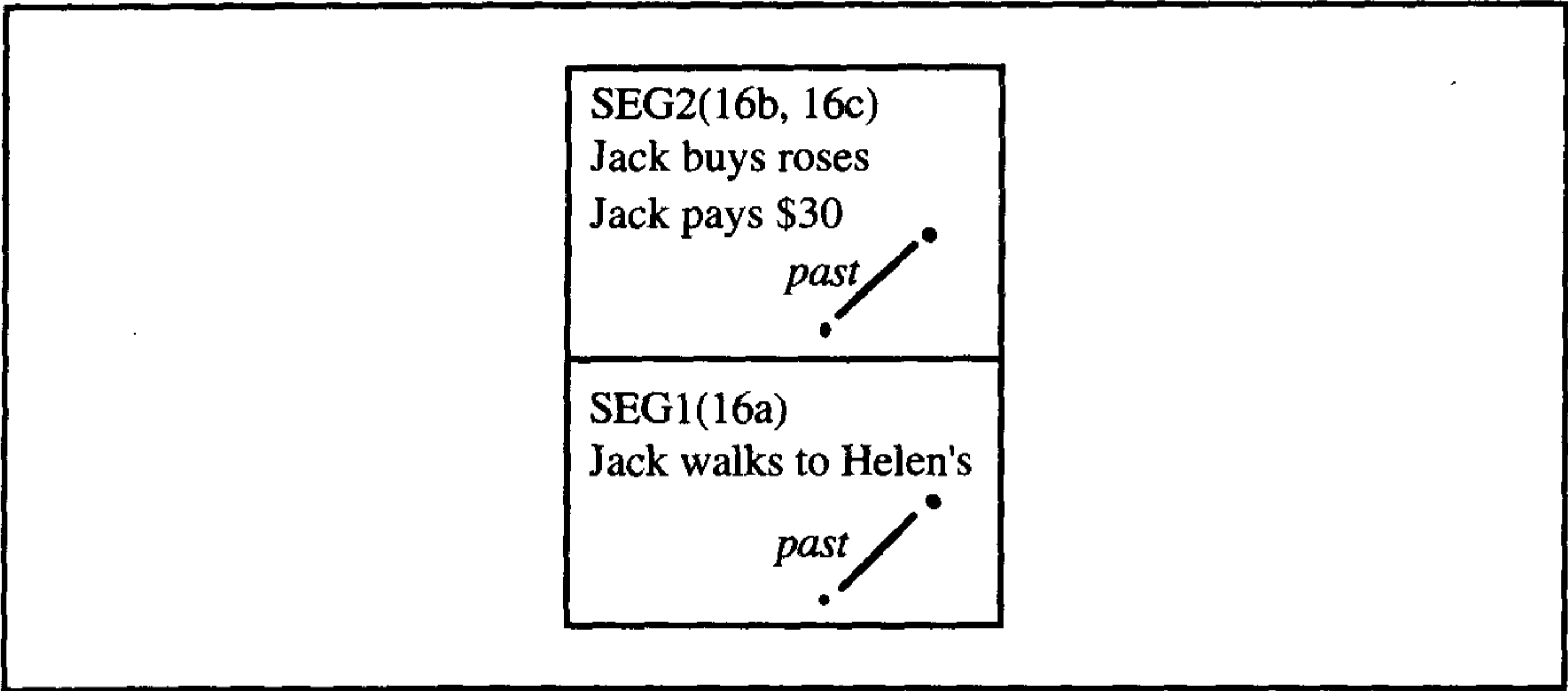


图 16.17 句子 16c 后的关注栈

16.6 管理关注栈

在前面几节中,我们探讨了影响篇章结构的不同问题。在这一节中,我们把这些问题结合在一起,构造一个在线处理篇章的简单模型。为了明确起见,我们将集中于一种特定形式的篇章,即描述现实世界中的事件和情境的篇章。这包括叙述、教学文本和关于某一特定任务的对话。本质上,任何以因果关系和时间推理为基础的篇章都是我们讨论的对象。这些形式的篇章将被称为因果性关联的篇章。

定义了推理的形式,我们就可以详细讨论一个片段的篇章目的。特别是,每一个片段都是由描述一些情境的目标来驱动的。这一情境可以是发生的事件或保持的状态,也可能是任意复杂的情况。例如,它可能描述来自不同 agent 的互相影响的一些行为。只要这些行为是有因果关系或者时间相关的,就都可以聚合在一起成为一个单独的场景。我们假定每一个子句描述一个事件(取决于子句描述内容的不同,可以指事件、过程或状态)。这些事件实际上都是简单的情境,通过寻找与这个情境中已经存在事件的因果关系或时间关系,可以使之结合为一个更复杂的情境。如果一个情境 S' 表示另一个情境 S 的一个子部分,我们称 S 支配 S'。

基于前面的介绍,与一个片段相关的局部篇章状态将包括以下信息:

- 当前的局部篇章上下文(主要从这个片段中的最后一个句子推导出来,如第 14 章所述)
- 来自这个片段中的前面句子的历史记录列表
- 这个片段的时态树(该时态树已经用这个片段中的所有事件进行了标记)
- 这个片段中最近描述的事件
- 刻画了到目前为止这个片段中所有描述过的内容的一个情境

图 16.18 表示了篇章 17 中处理完句子 17b 后的篇章状态:

- 17a. Jack went to the store. (Jack 去了商店。)
- 17b. He bought some roses. (他买了些玫瑰花。)
- 17c. He wanted to surprise Helen. (他想给 Helen 一个惊喜。)

局部篇章上下文和最近的事件反映了这个片段中最近的那个句子。这里,我们可以看到中心是“Jack1”,下一个中心也是“Jack1”,可能的下一个中心是“Roses1”。最近的事件是句子 17b 描

述的 Eb。为这个片段所构建的情境是 Es,它包括 Ea 和 Eb,其中 Ea 使能 Eb(Ea 比 Eb 早)。Ea 和 Eb 在情境 Es 内出现。时态树表示为,这两个事件都是以简单过去时描述的,并且 Ea 适应 Eb。

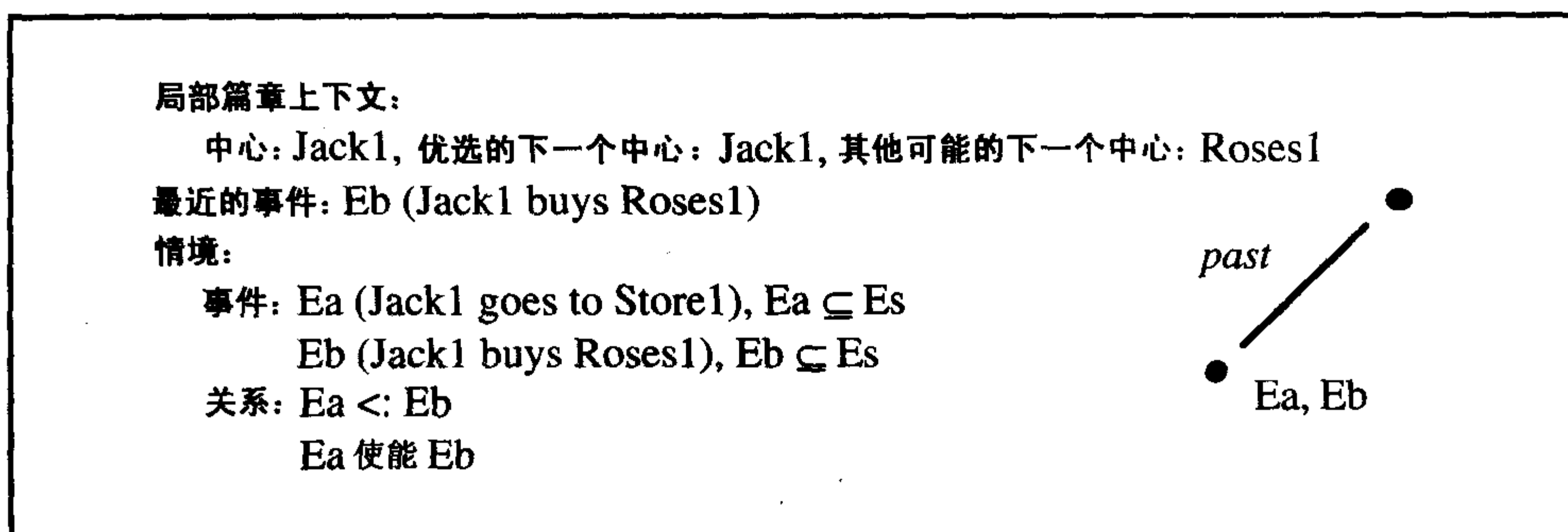


图 16.18 SEG1(17a,17b): 句子 17b 后的篇章状态

16.6.1 处理篇章状态

一个新句子可以用两种方法来影响篇章状态,即可能扩展一个存在的状态(继续一个已经存在的状态)或者创建一个新状态(开始一个新的片段)。当描述一个事件 E 的句子 S 扩展一个状态时,步骤如下:

- 用从 S 推导出的结构更新局部篇章状态。
- 更新历史记录列表。
- 把 E 增加到时态树中,并且推导出合适的适应关系和其他时间性的约束。
- 把状态中最近的事件设置为 E。
- 利用推理模块来推理得出合适的结果,以扩展其目前描述的情境的内容。

考虑一个例子。图 16.19 显示了一个新的篇章状态,这一状态是由图 16.18 的状态用句子 17c 扩展后得到的。

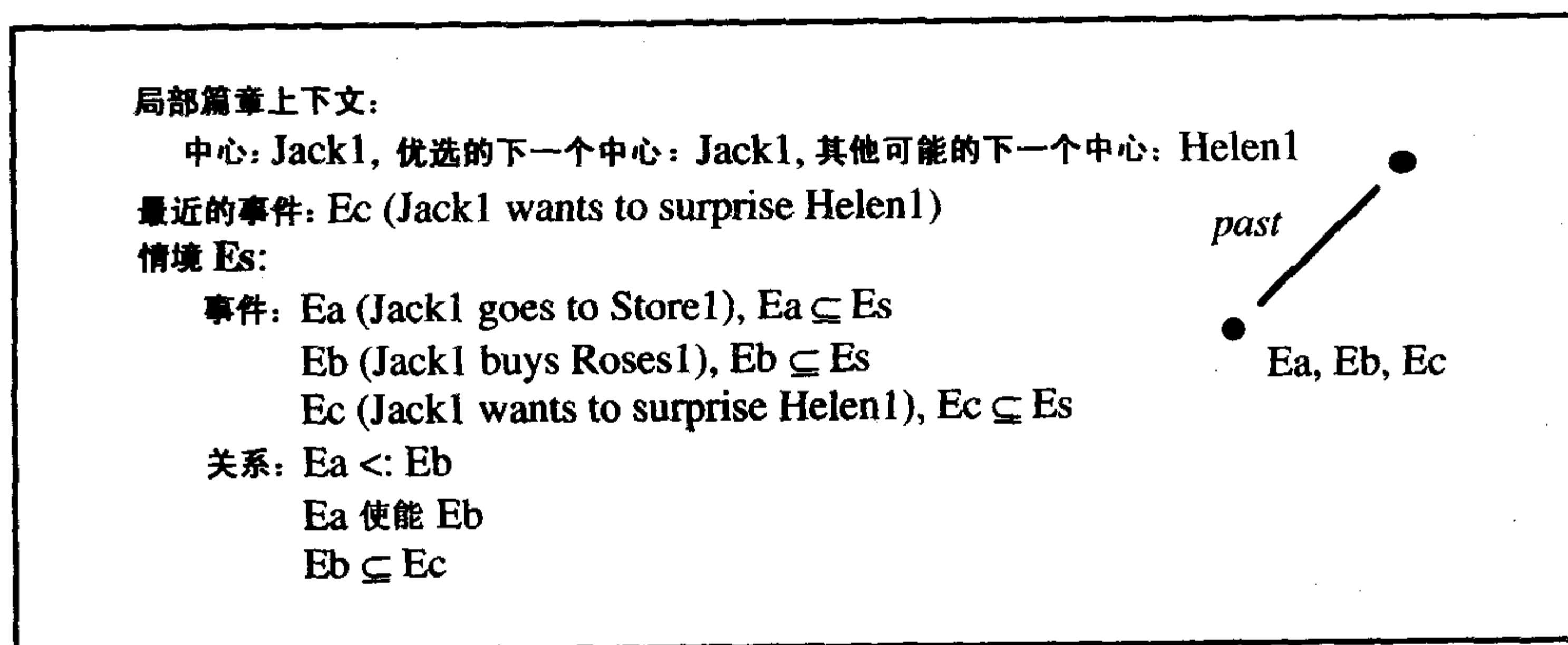


图 16.19 用句子 17c 扩展后的篇章状态

在创建一个新的篇章状态时,根据它与篇章栈顶的篇章状态的不同关系,存在三种情况。具体如下:

- 题外话(完全无关)——新状态与旧状态没有关系
- 时间转移——新状态在时间上与旧状态相关
- 详细阐述——新状态是展开来阐述旧状态中最近事件的

在题外话的情况下,这个新状态是基于这个新句子的,与前面的篇章没有关系。这样,这个新状态的建立完全来自于这个新句子的分析。

在时间转移中,这个新状态包括一个时间性的约束,这个时间性的约束把新情境与前面的情境联系起来。考虑篇章 17 的扩展:

- 17d. He had forgotten her birthday the day before.
(在前一天,他忘了她的生日。)
- 17e. He had been busy at work and hadn't noticed the date.
(他一直忙着工作,没有注意日期。)

句子 17d 引入了一个新的子片段,描述了一个比目前描述的事件还要早的情境。这个新状态被压入到图 16.19 的状态栈的上面,如图 16.20 所示。这里需要注意时间性约束,即新情境 Es2 要早于情境 Es,其中 Es 在栈中位于 Es2 的下面。根据这个上下文,从时间上可以对句子 17e 进行定位,不仅因为它适应于 Ed,而且它比 Es 要早。

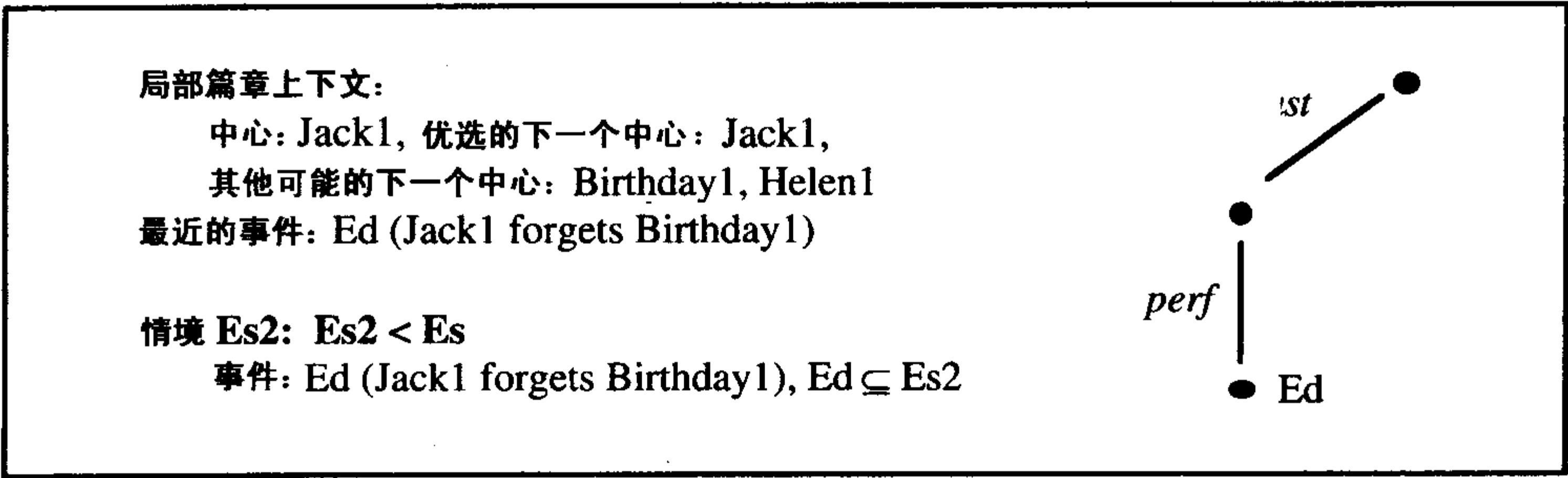


图 16.20 为句子 17d 创建的新的篇章状态

第三种情况是详细阐述,引入了一个新片段,这个新片段将展开来阐述最近那个事件的细节。这种情况可以处理为,在这个新片段中,把最近的事件作为构造新片段的情境。例如,篇章 17:

- 18a. Jack went to the store.(Jack 去了商店。)
- 18b. He bought some roses.(他买了些玫瑰花。)
- 18c. He got ones that were on sale(他买的这些玫瑰花都是待售的,)
- 18d. and paid cash for them.(并支付了现金。)

句子 18c 开始了一个新的片段,这个片段详细描述了句子 18b 中的购买行为。对应的篇章状态如图 16.18 所示。为句子 18c 创建的新状态如图 16.21 所示。需要注意的是,这个片段中描述的情境是 Eb,而不是一个新的情境。其中蕴涵了期望的暗示,即 Ec 在 Eb 的过程中。

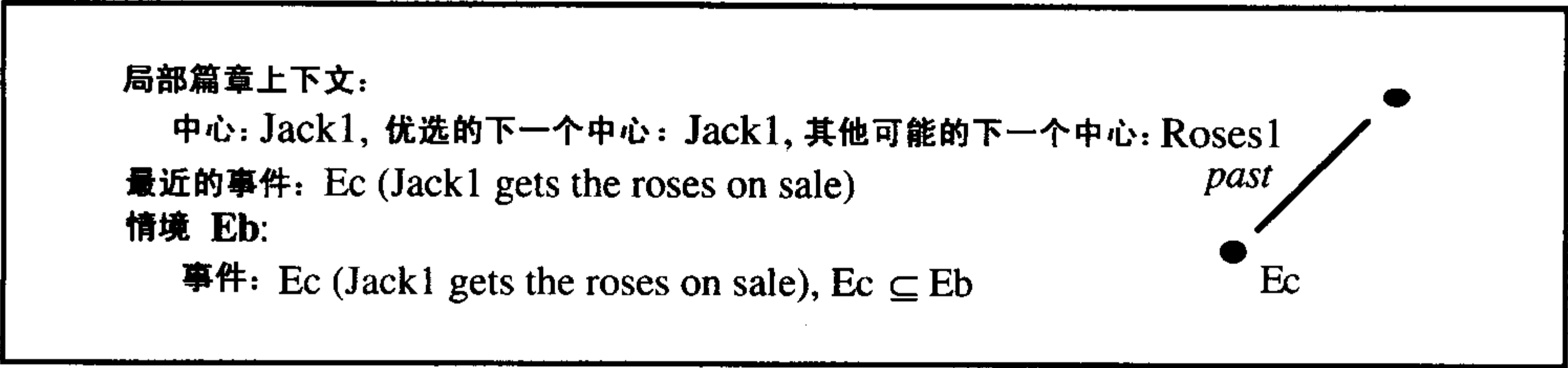


图 16.21 由句子 18c 创建的新的篇章状态

16.6.2 确定栈的更新

到目前为止,我们已经探讨了用来更新关注栈的不同操作,但是没有考虑在特定场景下怎样确定更新的内容。这个问题可以分为两个主要的子问题,这两个子问题可以由下面两个问句来刻画:

- 新句子能够扩展栈顶的状态吗?
- 新句子能够创建新的片段,并把新片段压入到这个栈的当前栈顶状态上面吗?

对这个栈中的每一个篇章状态解答这两个问题,我们就可以得到所有可能的解释。在前面几节中,我们基于篇章片段目的、时态分析和提示语等提出了几种约束,这些都需要用到推理模块。这些约束给出了解答这些问题的一些方法。尽管在某种程度上这些方法还是很简单,但给出了对这些问题处理方式的一种感性认识。

对扩展篇章状态的约束

- 指代约束**——一个句子 S 能够扩展一个状态,仅当这个句子 S 的回指模块可以解释为这个状态中的篇章实体。
- 时态约束**——一个句子 S 能够扩展一个状态,仅当这个句子 S 的时态与为这个状态定义的时态是一致的,或者在不考虑完成体的情况下是一致的(这意味着一个时间视角的转移)。
- 推理约束**——一个句子 S 能够扩展一个状态,只有这个句子 S 描述的事件是这个状态描述的情境的一部分。

根据压栈操作是表示题外话、时间转移或者详细阐述的不同,把新状态压入到一个已存在的状态(称为起始状态)的约束将随之也有所不同。但在所有情况下,起始状态都为解决新句子中的歧义提供了局部篇章上下文。除此之外,每一类压栈都有其自己的约束,例如:

压栈的约束

- 对所有压栈操作的指代约束**——这个句子 S 可以压入一个新状态,仅当这个句子 S 的回指模块可以解释为起始状态中的篇章实体。
- 对时间转移的约束**——或者句子 S 的时态扩展了起始状态中的时态树,或者有一个明显的时间短语用来指明一个新状态。
- 对详细阐述的约束**——一个新句子 S 必须描述这类情境的一部分,该情境是起始状态中最近的事件。
- 对题外话的约束**——题外话必须由提示语或其他明显的方式标记出来。

由于存在歧义,对于一个句子更新关注栈的方式,有很多种不同的解释。对这些情况,我们需要对这些解释给出一些优先顺序,如下所示:

- 已有片段的继续优先于子片段的压入
- 优先考虑使得出栈操作数目最少的那个解释

16.6.3 结构化的提示语

最后一个问题是结构化的提示语对模型的影响。每一个提示语都可以根据三个因素来分类,即这个提示语显示的是一个片段的结束(“POP”)、一个新片段的开始(“PUSH”)、还是已有片段的继续(“CONTINUE”)。一个单独的提示语可能具有一个或多个这样的功能。例如,单词“anyway”同时表示“POP”和“CONTINUE”。当然,虽然这些提示语对于推理过程也许也有影响,但对于我们当前的目的来说,这个信息已经够了。

框 16.3 指代由篇章片段创建的情境

通过在一个篇章检查可以以回指方式引用的事件或场景,可以找到更多的片段结构切分的依据。考虑下面的篇章:

- a. When Jack entered the room, everyone threw balloons at him.
(当 Jack 进门时,每个人都向他扔气球。)
- b. In retaliation, he picked up the ladle and started throwing punch at everyone.
(为了反击,他拿起长柄杓,开始向每个人泼宾治酒。)
- c. Just then, the chairman walked into the room.(这时候,主席走进来。)
- d. Jack hit him with a ladleful, right in the face.
(Jack 泼了他一满杓酒,正好泼在脸上。)
- e. Everyone talked about it for years afterwards.
(多年以后,每个人还都在讨论这件事。)

在句子 e 中,代词“it”可以指代由句子 a 到句子 d 描述的情境。这四个句子形成的一个片段描述了一个复杂的情境,可以由句子 e 中的代词来指代。篇章中描述的哪些事件还可能被位置 e 的句子指代呢?可能只有句子 d 中描述的事件可能被后文指代,如 d 的另一种下文为:

- e'. It was a foolish thing to do.(这么做是很傻的。)

但是,我们不可能在位置 e 构建一个句子,使其中的代词指代“主席走进来”这个事件。Webber(1991)认为,在任何一个给定的时间,可用于回指指代的事件/情境,由最近提到的事件和篇章栈中每一个低层篇章状态中构建的情境组成。

图 16.22 举例说明了一个利用提示语的算法。如果一个相应的提示语在这个句子中,那么这个算法的输入是篇章栈、新句子的分析和三个变量 POP, PUSH 和 CONTINUE。如果句子中出现一个相应的提示语,这个变量就设置为真。第一步检查是否有明显的 POP;如果有,弹出栈顶的状态;然后,这个算法重复检查这个栈中可能的状态扩展;然后,再搜索可能的新状态入栈。

输入参数: STACK——篇章栈
 S——新句子
 POP, PUSH, CONTINUE——提示语的标志

算法:

1. 如果 POP 被设置为真,那么弹出 STACK 一次,然后继续。
2. 如果 PUSH 被设置为真,那么搜索 STACK,寻找可以被 S 扩展的状态。第一个寻找到的状态就是答案。
3. 如果第二步失败了,没有产生答案,并且 CONTINUE 没有设置为真,那么搜索 STACK 得到一个状态,使得 S 可以从中压入一个新状态。

图 16.22 将下一个句子结合到篇章栈中的算法

16.7 例子

考虑上一节描述的模型是怎样用于解释图 16.2 中的篇章的,在这里重写篇章 19:

- 19a. Jack and Sue went to buy a new lawn mower(Jack 和 Sue 要去买一台新的割草机,)
 19b. since their old one was stolen.(因为他们的旧的割草机被偷走了。)
 19c. Sue had seen the men who took it and(Sue 看见了偷走这台割草机的那些人,并且)
 19d. she had chased them down the street,(她还沿着街道追赶这些人,)
 19e. but they'd driven away in a truck.(但是,他们被一辆卡车接走了。)
 19f. After looking in the store,(去商店看了后,)
 19g. they realized they couldn't afford a new one.(他们意识到买不起一台新的割草机。)
 19h. By the way, Jack lost his job last month(顺便说一句,上个月 Jack 丢了工作,)
 19i. so he's been short of cash recently.(所以,他最近手头有点紧。)
 19j. He has been looking for a new one,(他一直在寻找一个新的工作,)
 19k. but so far hasn't had any luck.(但是,目前为止,他都不走运。)
 19l. Anyway, they finally found a used one at a garage sale.

(不管怎样,最后他们买了一台车库现场出售的旧割草机。)

正如我们所期望的,句子 19a 创建了一个新的篇章状态 SEG1(19a),句子 19b 扩展了这个状态。连接词“since”能够帮助推理模块确定句子 19b 中的事件是句子 19a 中的事件的动机。子句 19c 是过去完成时态,表示了时间转移。新的篇章状态 SEG2(19c)被压入到栈 SEG1(19a,19b)上。图 16.23 显示了子句 19c 后的篇章状态。下一个子句 19d 是 SEG2(19c)可接受的扩展,19e 也是可接受的扩展。

子句 19f 没有时态,而且从属于子句 19g,所以子句 19f 的解释由子句 19g 决定。子句 19g 是简单过去时,所以它可能是 SEG1(19b)的重新开始;也可能有时间的转换,即 SEG2(19c,19d,19e)的继续。推理模块必须决定是 Ee(driving in truck)适应 Eg(can't afford lawn mower),还是 Eb(old one stolen)适应 Eg 的可能性更大。因为偷割草机的人不可能考虑买新的割草机,所以 Ee 适应 Eg 是不合理的。所以,给定上下文状态 SEG1(19a,19b),更可能的解释是 Jack 和 Sue 正考虑买割草机,实际上,这也是我们期望的解释。这样,在一个出栈操作后,子句 19g 可以解释为 SEG1(19a,19b)的扩展。需要注意的是,代词“they”的解释受这个决定的影响。如果 19g

已经扩展了 SEG2(19c, 19d, 19e), 那么“they”可能是指“the men”, 而不是 Jack 和 Sue。这个新的篇章栈如图 16.24 所示。

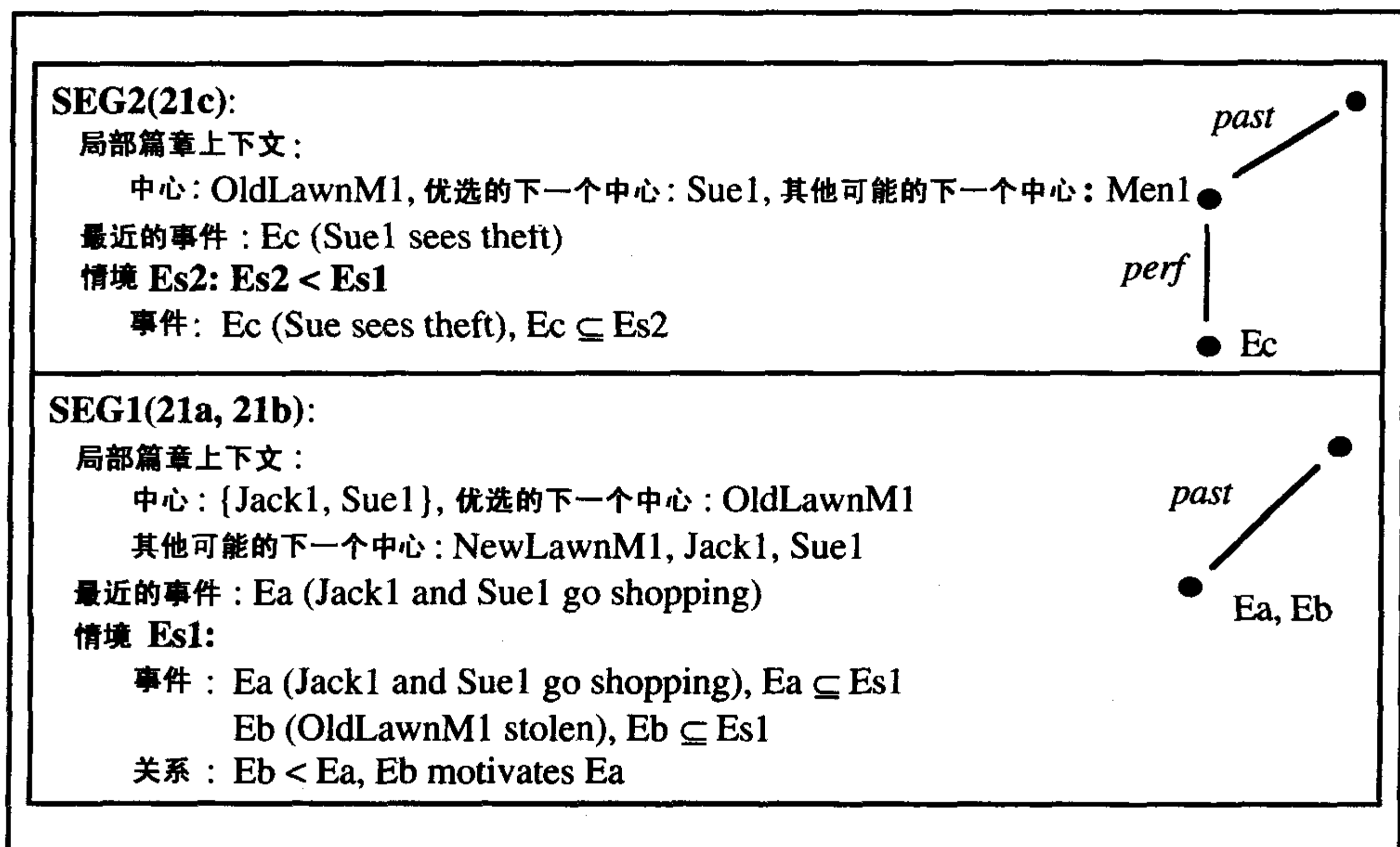


图 16.23 子句 19c 后的篇章状态

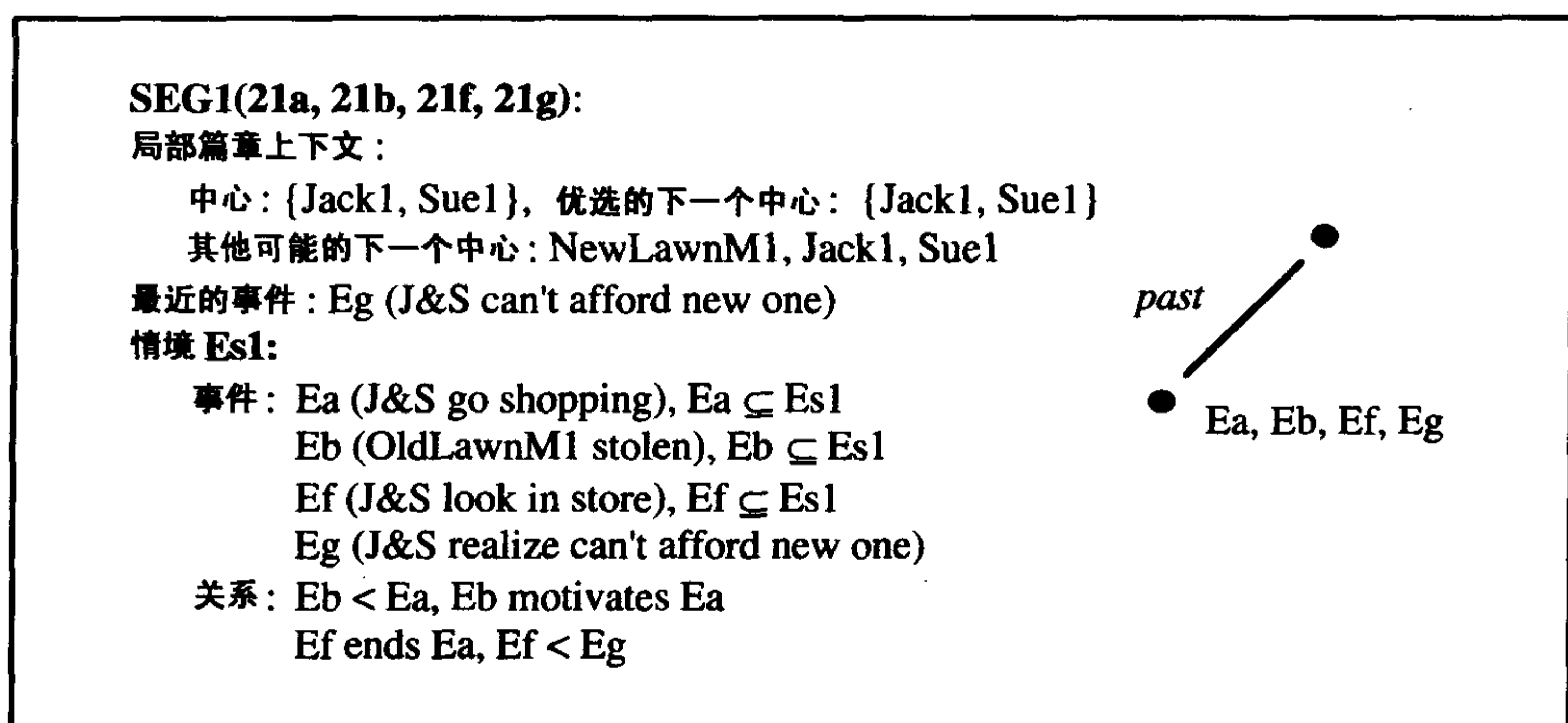


图 16.24 子句 19g 后的篇章栈

现在, SEG2(19c, 19d, 19e) 已从状态中弹出, 它表示的所有信息都发生了什么呢? 很显然, 它不应该完全被忘记, 因为它是篇章理解的组成部分。事实上, 在后面的篇章中, 它包含的事件可以再次被指代。那么, 从栈中弹出的意义何在? 回答就是这一篇章状态不像 SEG1 那样能够在后面的篇章中被继续。换句话说, 你不能像这个上下文仍然处于活动状态那样开始讨论它。但是, 我们可以明显的方式重新激活前面篇章的上下文, 例如, 可以说“Remember when I told you about the men stealing the lawn mower?”这样一来, 我们就可以继续讨论这个场景。这种对前面所讨论话题的一个子部分的指代在文献中有时称为语义回返 (semantic return)。在我们讨论的这个模型中, 这种情况被当做创建一个新的片段来处理 (讨论旧的上下文), 而不是回到前面的篇章状态。

再回到这个例子,子句 19h 包括提示语“by the way”,表示一个话题的分叉。这将新的状态 SEG2(19h)压入篇章栈,这个篇章栈又被子句 19i, 19j 和 19k 所扩展。在子句 19l 中,提示语“Anyway”提示栈的弹出和继续,所以 SEG3(19h, 19i, 19j, 19k)从栈中弹出,19l 被成功地分析为 SEG1(19a, 19b, 19f, 19g)的继续。因此,子句 19l 中的“one”被确定为指代“a lawn mower”。注意,如果提示语“anyway”不存在,很可能会将句子 19l 解释为 SEG3(19h, 19i, 19j, 19k)的继续。如果是这样,代词“one”的解释就是“a new job”。

这个例子说明,通过利用来自时态分析、推理、指代和提示语的信息,我们可以确定句子预期的结构。每一种形式的信息都是很重要的,但任何一种信息都不足以单独完成这个任务。

16.8 小结

为了解释扩展的篇章,很多研究人员把篇章看做片段的层次化的组织。每一个片段都表现了其内部连贯性,这种连贯性说明基于最近上文的歧义消解策略是有效的,而且片段中的这些句子可以看做是描述了同一个问题或同一种情境。这样,第 14 章和第 15 章中提出的算法在一个片段内就是有效的。片段的层次结构表示促使我们把篇章状态组织为一种基于栈的结构,每一个篇章状态刻画了一个单独片段的状态。这种基于栈的机制允许一个片段被暂时搁置,后面的某个片段又继续这个暂时搁置的片段。在这个模型中我们主要讨论了篇章的三个方面的重要问题,并通过例子说明了基于片段的模型怎样提供了一个统一的框架,从而把这些不同的处理技术结合在一起。这些技术包括指代消歧、时态和体态,以及这个领域内的一般性推理。此外,这种结构也为解释篇章中明确表示篇章结构改变的提示语提供了一个框架。

16.9 相关工作与深入阅读材料

篇章分析的不同研究领域大都集中于某一个方面的工作,例如指代消歧、时态分析,等等。很少有研究人员试图把篇章处理的不同方面结合为一个统一的模型,不过也有一些工作提出了全面的框架。如, Grosz 和 Sidner(1986)给出了一个相对统一的模型,这个模型是本章的基础。该模型给出了由片段所显示的篇章结构、关注栈和篇章的意图结构(表示为篇章片段目的)这三者的区别。这里的术语“信息层次”和“意图层次”的概念来自于 Moore 和 Pollack(1992)。

有研究认为,不同句子可以通过分析说话者的目的相互联系起来,这一想法已经有很长的历史了。不过最近,这个方法中影响最大的来源是 Grimes(1975)。Grimes 确定了可以用于篇章中的三类常见的功能:一个句子可以支持或补充前面句子的信息(详细阐述);可以创建一个场景(确定一个事件的时间和地点);也可以确定一个对象(引入一个要讨论的新对象或回到一个旧的对象)。另外,也有相当数量的工作提出了一些面向计算的框架模型,每个模型都引入了一些不同的关系集合。Hirst(1981b)全面综述了这些模型及其在回指分析中的应用。

Linde(1979)和 Grosz(1977)都观察到了这样的事实:在面向任务的对话中,话题流可以按照正在讨论的任务的层次结构来解释。Linde 研究了公寓住宅(apartment)的描述,研究结果表明,房间的选择并不是随机的,篇章话题是按照一次假定的“参观”这个公寓的顺序来推进的。Grosz 研究了一个专家帮助一个学徒安装抽水机时的对话。同样,话题流也遵循安装抽水机的

实际执行步骤来进行。Grosz 为指代定义了全局焦点(global focus)的概念,在其中,全局焦点对应于受篇章栈机制约束的规划推理。Grosz 利用了称为焦点空间(focus spaces)的机制,此机制能够确定规划的哪部分是目前正在讨论的。这些焦点空间用栈结构来维护,这种栈结构为确定定指性描述的指代给出了一些合适的对象。这对应于前面介绍的片段的历史记录列表的组织方式。Grosz 的研究表明,焦点栈被弹出后,被恢复焦点空间中的对象可以立即被代词所指代,而不要求明确的重新引入。

Reichman(1978;1985)提出了一个关于篇章的理论,这一理论对 16.6 节的联机处理模型有很重要的影响。Reichman 的工作指出了将提示语(在她的模型中称为“clue words”)结合到模型中的重要性。该模型使用了基于上下文空间的篇章状态的显式表示方法,她所指的上下文空间与我们所讨论的篇章状态大致对应。在 Reichman 的模型中,任何时候至多有 3 个上下文空间与篇章相关,分别是活动空间、控制空间和生成空间。其中,活动空间对应于我们论述的篇章栈中的栈顶;控制空间对应于栈顶下面的那个元素;生成空间在本章介绍的模型中没有对应者。生成空间是指说话者的一些论点,但还没有进行充分讨论,可能在一段时间后再来继续讨论。但是,当再回来时不允许对焦点元素直接用代词进行指代。Reichman 利用这个空间分析了一些类型的指示代词的指代问题(例如,“this”和“that”的用法)。

很多研究人员都发现,提示语是篇章中的一种很重要的结构[例如,Grosz 和 Sidner(1986),Cohen(1987),Litman 和 Allen(1987;1990)]。Hirschberg 和 Litman(1993)提供了其他研究人员用过的提示语的一个列表,并说明在对话篇章中一些提示语有与众不同的语调特征。

很多文献都对时态和体态都进行了深入的阐述,其中有些文献已经在第 13 章末尾进行了讨论。很多如 Steedman(1982)和 Hinrichs(1986)这样的文章提出了时态的解释与回指关系有关的想法。参照时间作为回指元素的想法也被提出了很多次,Song 和 Cohen(1991)就是很好的例子。阐述时态问题的很多理论仅仅在简单陈述的上下文中有一定的发展,而且这种陈述应该是符合陈述惯例的。Song 和 Cohen(1991)是把时态和体态联系在一起进行规划推理的计算模型的很好的例子,但这个模型仅仅用于简单的陈述。

16.5 节的介绍是基于 Webber(1988),Hwang 和 Schubert(1992)的工作,Webber 探讨了时态解释的回指属性,并引入了时间焦点的概念。她还研究了片段的边界是怎样影响可能解释的范围的。16.6 节介绍的建立情境的模型与她提出的把篇章解释看做建立一个事件结构的模型很相似。Hwang 和 Schubert(1992)提出了时态树的概念,并对适应关系(oriens relation)给出了形式化描述,他们还讨论了与时态树和篇章片段相关的一些问题。时态树能够很好地对嵌入式时态的那些子句进行分析(我们在本章中没有讨论这个问题),而其他的方法则没有很好地解决这个问题。本章使用的适应关系的定义受 Lascarides 等(1992)中介绍的时态解释的方法的影响。这种方法中,在与因果推理没有任何冲突信息的情况下,陈述惯例一般作为默认信息出现。另外,我们在本章中没有阐述,但对于篇章的时间性分析很关键的一个问题就是,对明确指定一个事件时间的时态性连词和时态性副词的分析。Steedman(1982)深入研究了分析时态性连词(如“when”和“while”)的复杂情况。

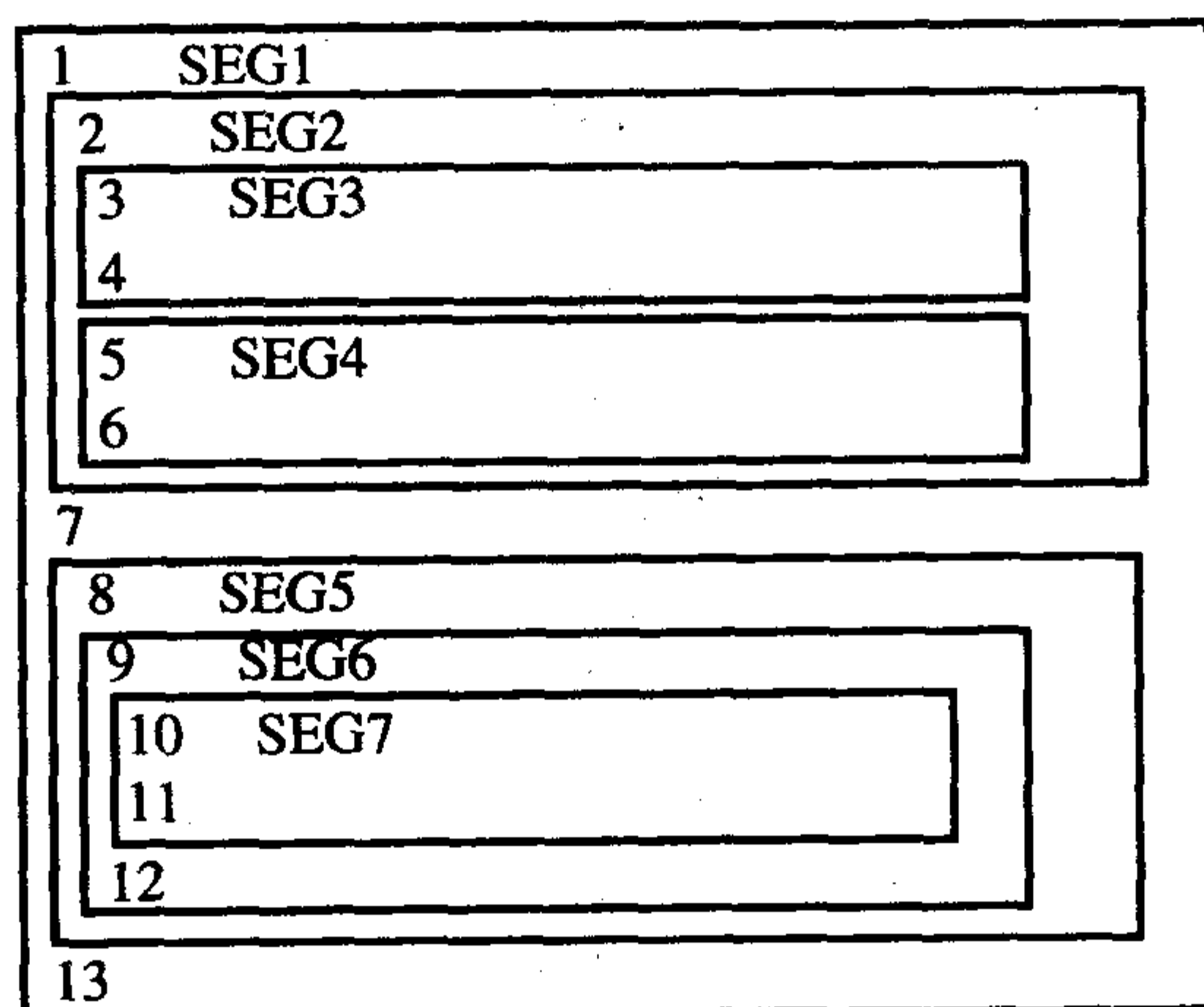
对本章中的很多理念都有影响的一种篇章研究方法是定义连贯关系的研究工作(在框 16.1 中进行了简单的讨论)。例如,Hobbs(1979)定义了一个连贯关系集合,并把篇章解释看做证明输入正确性的一个过程(Hobbs 等,1993),这个过程溯因式地假设了一系列的连贯关系。Lascarides 和 Asher(1993)也采用了类似的方法,他们不是利用溯因推理,而是利用演绎技术。

在这种演绎技术中,采用的是可反驳的推理。Hovy(1993)详细讨论了在自然语言生成系统中利用篇章结构关系的问题。Mann 和 Thompson(1986)定义了修辞结构理论(RST, rhetorical structure theory),该系统给出了一个很大的分类体系,其中包括了存在于句子和片段中的各种不同关系。Fox(1987)对 RST 做了很好的综述。Moore 和 Pollack(1992)认为 RST 混淆了意图和信息这两个不同的分析层次,从而导致了很严重的问题。不过,类似 RST 的模型被证明对自然语言生成有很大帮助。

对篇章结构进行刻画的另一个方法是提出了一种篇章语法。一些篇章语法试图解决句子之间的语义联系,另一些篇章语法则研究话题流、话题中断和话题恢复。Rumelhart(1975)提出了利用一部语法为简单的故事结构建立模型。Scha 和 Polanyi(1988)也给出了一个语法,用于为话题流建模。

16.10 习题

1. 【易】一个篇章对应的片段结构如下图所示(每一个数字表示一个子句)。对这个篇章,用图 16.4 所示的格式绘出其关注栈的序列。



2. 【中】在篇章 16 中的每一个句子后面,绘出篇章栈,给出时态树和每一个状态的情境描述。

16a. Jack was walking over to Helen's house.

(Jack 正在步行去 Helen 的家。)

16b. He had bought some roses at Honest John's Flower Mart.

(他已经在 Honest John 的花市买了些玫瑰花。)

16c. He paid thirty dollars for them(他花了 30 美元,)

16d. and Honest John swore they were fresh.

(而 Honest John 许诺这些花是新鲜的。)

16e. But they wilted before he got to Helen's.

(但是,他还没到 Helen 的家,这些花就枯萎了。)

3. 【中】请分析下面的句子,并跟踪图 16.22 所示的算法。给出每一个子句后的篇章状态,并确定事件和每一个指代表达式的先行词之间推理性的联系。这个篇章是两个人准

备出售修理厂的对话的一部分。我们假定修理厂(the garage)在这个篇章场景中是惟一给定的。

- a. An old lawn mower is next to the TV set in the garage.
(在修理厂中,一台旧的割草机放在电视机边上。)
- b. Last summer, the engine stalled frequently,
(去年夏天,发动机经常停转,)
- c. so no one really used it.
(所以,没有人真正用过它。)
- d. But I still hope to sell it.
(但是,我还是希望卖掉它。)

这个模型是否能够给出第3个句子中的代词“it”的正确解释?如果不能,能否扩充这个模型,以便产生正确的答案?

- 4. 【中】对篇章3中代词“it”的每一次出现,请给出解释这个代词时的篇章栈。在每一种情况下,这个结构能否预测合适的指代对象?如果不能,请说明为什么。
- 5. 【中】随意选择一个文本,取其中半页的篇幅,利用这一章描述的技术来分析并得到这个篇章的切分。请定义这个文本的总的篇章目的,并说明篇章目的的目标/子目标之间的关系对应于什么?请描述确定每一个片段边界的理由。
- 6. 【中】对框16.1中提到的每一个提示语,讨论是否表明了片段的压栈或片段的压栈并出栈操作。举例给出两个篇章片段,它们分别利用了两个提示语,一个提示语表明压栈操作,另一个提示语表明出栈并压栈操作。对每一个篇章,请给出由提示语所显示的片段结构。
- 7. 【中】考虑下面的篇章:
 - a. Jack went to the music store. (Jack 去音像店。)
 - b. He bought a new CD player. (他买了一个新的 CD 播放机。)
 - c. His old one had stopped working. (他的那个旧的播放机已经坏了。)

每一个子句都定义了一个事件时间、参照时间和说话时间。如果我们增加一个篇章约束:这三个子句的参照时间是相同的,那么,关于每一个事件之间的时间关系,可以得到怎样的结论?这是否是我们期望的结论?如果不是,该怎么改进这个方法?利用时态树和适应关系的方法来分析同一个篇章,能否得到我们期望的结论?如果不能,该怎么改进这个方法?

- 8. 【难】在一些情境下,简单过去时和现在完成时看似是可以交换的。一个人去商店买东西,可以表示为“I went shopping”或者“I have gone shopping”。这两种表达的区别很小,但是,它们的效果可能很不相同。我们可以表达为“I went shopping yesterday”,但不能表达为“* I have gone shopping yesterday”。考虑下面的情境,Jack 在一次彩票抽奖中,得到一百万美元,但是他把这些钱全部输在赌博上了。要告诉某人他抽中了大奖,我们可以表达为“Jack won the lottery”。但如果表达为“Jack has won the lottery”就不合适。那么,这两个表达之间的区别何在?请另外举例来支持你的论点。为了在扩展的篇章中发现不同子句之间的时间关系,这种区别能提供哪些隐含的信息?

第 17 章 会话 agent

本章讨论定义会话 agent 所需要的一些概念,这里的会话 agent 是指能够参与自然语言对话的 agent。这样的 agent 系统可以用于任何需要用语言进行人机交互的应用中,包括问答系统、计算机辅助规划、辅助执行、推理系统和检索系统。17.1 节讨论建立对话 agent 的必要条件。17.2 节讨论语言作为多 agent 交流手段的本质。17.3 节探讨 agent 的认知状态表示中的一个重要问题,信念的表示。17.4 节探讨期望、意图和规划的表示问题。17.5 节将语言作为一种行为进行了探讨,提出了言语行为的概念。17.6 节讨论言语行为在规划问题中应用的方法。17.7 节分析识别意图的问题。17.8 节讨论一个 agent 的意图是怎样发生的,17.9 节介绍确定语内表现行为的方法。最后,17.10 节讨论利用基于规划的技术进行篇章结构的推理。

17.1 会话 agent 的必要组成部分

要定义会话 agent,需要回答的一个问题是为什么 agent 要说话?是什么促使 agent 说话或者是什么促使 agent 理解对方所说的内容?有人会说这样的行为只是机械地按照已有程序来运行,并没有什么复杂的。例如,数据库回答 agent 的行为可以分两步来描述:

1. 分析和解释输入的问题,并转换为表示查询的逻辑形式。
2. 在数据库上执行这个查询,得到所需要的答案。

但是,我们不认为这种程序有很高的智能或很强的会话能力。例如,如果数据库中没有输入问题的答案,这一系统就不能返回任何有意义的结果。更进一步来说,没有任何独立的特性能够说明这个系统是具有智能性的。

当然,在某种程度上,即使最复杂的系统都是由上层程序驱动的,并不会比程序所指定的操作更复杂。但是,如果有某种表示方法能够准确反映那些驱动人类活动的因素,那么通过将系统建立在该方法的基础上,我们还是可以在系统中融入部分智能。具体来说,人类的行为来自于他们要达到的目标。另外,人们能够清醒地了解所处的场景,并对这一场景具有正面或负面的感受。人们常常会尝试各种方法来改善自身的处境。例如,如果你发现自己站在高速公路上,你很可能会离开。这一行为看似简单,实际上需要用很复杂的认知结构才能解释。首先基于视觉观察,你确信在高速公路上,并且知道这很危险,所以期望从路上离开。你还能够推理并得到往回退比向前走过公路要快,因此你会选择退回来。这个例子说明了一个复杂的智能 agent 所具备的几个重要模块:

感知(perception)——这个 agent 必须能够感知周围的世界。

信念(belief)——这个 agent 必须能够表示客观世界的当前状态。

期望/需求(desire/want)——这个 agent 必须能够对世界的状态具有积极或消极的反应,能够比较不同状态的期望程度。

规划和推理(planning/reasoning)——这个 agent 必须能够推理,以得到到达其他状态的方法。

落实(commitment)——这个 agent 必须能够做出执行某种动作的决定以到达一个不同的状态。

意图(intention)——这个 agent 必须能够维护所决定的行为的过程。

行为(acting)——这个 agent 必须能够执行某种行为以改变它当前的状态。

一个会话 agent 位于语言和思想都很贫乏的世界中,它只接受话语,其行为仅仅是生成话语。当然,在应用中一个 agent 可能具有其他的感知能力和行为能力。但是,本章只关注于语言方面的能力。

刚才讨论的智能行为的七个方面,包括四个过程(感知、规划、落实和行为)和三个状态(信念、期望和意图),我们把这个模型称为 BDI(belief - 信念, desire - 期望, intention - 意图)。所有这些术语在后面会有更详细的定义。目前,基于我们的直觉理解来分析整个模型。agent 的整个结构如图 17.1 所示,其中框表示认知状态,其他标记表示正在进行的过程。agent 利用感知得到的信息来动态地更新信念,利用信念进行推理和规划,然后基于信念和期望来着手落实某种意图,最后通过行为来实现这些意图。

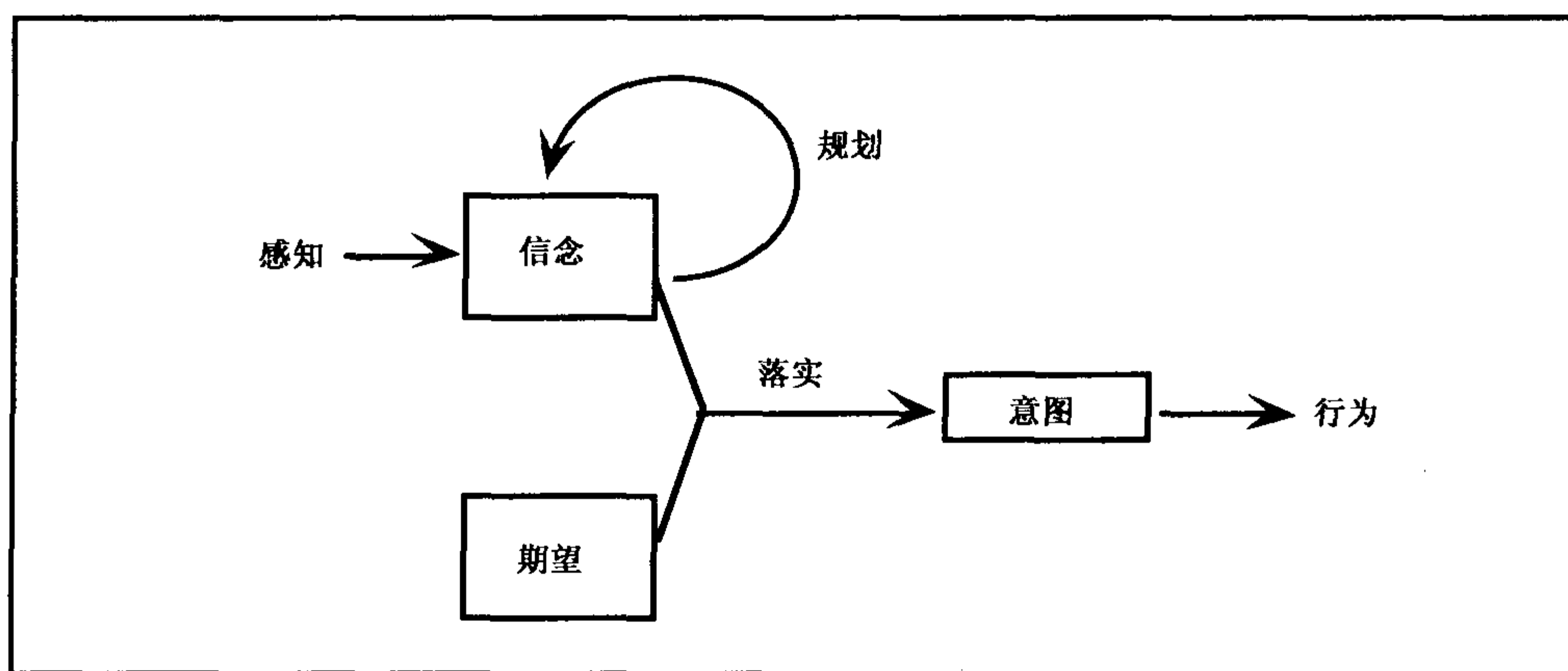


图 17.1 一个智能 agent 的 BDI 模型

要在语言中应用这样的模型,必须明确地做出表达句子的行为。涉及语言的这种行为称为言语行为(speech act),它在会话 agent 的实现中是很重要的。像前面已经讨论过的任何其他行为一样,由对应的英文单词来命名言语行为,例如“ask”(询问),“request”(请求),“inform”(通知),“deny”(否认),“congratulate”(祝贺),“confirm”(确认)和“promise”(承诺)。像其他行为一样,我们应该能够对言语行为进行定义和推理。不过,言语行为比其他行为更复杂,因为这涉及多个 agent,即交流行为。另外,它们是根据 agent 的认知状态来定义的,而不是根据世界的物理属性。

17.2 作为多 agent 活动的语言

语言和其他形式的交流往往涉及多个 agent。在空无一人的房间说话,就达不到交流的目的。其他形式的交流(例如写作)可以令交流行为超越时空。我们现在写的一些东西,10 年后也能够被读取。但是,所有交流的目的都是为了影响他人的认知状态。

另外,一个 agent 如果没有认识到其他人交流的意愿,交流也不会发生。例如,假设在一个场景中,Jack 和 Sue 都是工业间谍,他们制定了一个方案:如果 Sue 造访 Jack 时是安全的,那么 Jack 就让窗户开着。有了这个系统,Jack 就可以借助窗户与 Sue 进行交流。Jack 打开窗户,当 Sue 看见窗户打开时,Sue 就知道可以造访 Jack。然而,如果某一天很热,Jack 打开了窗户,Sue 恰好路过并看见窗户开了,那么她就造访 Jack。这就造成了误解,因为 Jack 打开窗户并不说明 Sue 可以造访,Jack 不是根据这个意图才打开窗户的。相反,假设天很热,Jack 打开窗户来告诉 Sue 她可以来访,但是,当 Sue 路过时,她认为可能是由于天热,Jack 才打开窗户来乘凉的。这种情况下,Jack 进行交流的尝试失败了,因为 Sue 没有通过打开的窗户识别出他的意图。

所以,只有一个 agent 试图交流并且其他的 agent 也认识到了这一意图,交流才能够发生。交流的其他必要条件还包括要有一系列约定的符号及其对应的约定的意义。在前面的例子中,只有两个约定符号,即窗户开着的状态(意思是安全的)和窗户关着的状态(意思是不安全的)。Jack 和 Sue 已经定义了他们交流的专用语言,并且很有效,只是 Jack 在热天时要谨慎一些。

这些必要条件成为语言交流的基础。但是,它们在语言中根深蒂固,我们几乎注意不到。事实上,我们只要看见句子中的这些词就足以判断交流的意图。这是因为,和窗户的例子不同,表达说和写的实际动作很少用于交流以外的其他目的。当某人说某些单词时,我们假定他们试图交流,因为说话这一行为很少用于其他目的。需要注意的是,前面提到的窗户的例子是有歧义的,因此与说话这个行为有些不同。打开窗户可以用于两个目的,与 Sue 交流或者使空气流通。这说明在物理行为和交流行为之间有很重要的不同。Austin(1962)表达了这个观点,他观察到,只要表达某件事情,就存在 3 个行为:

非语内表现行为——说出一系列词语的行为

语内表现行为——说话者说出的这些词所表达的行为

言语表达效果行为——作为说话所导致结果的实际发生的行为

前面介绍的以窗户作为语言的例子说明了这些不同的行为。Jack 打开窗户是非语内表现行为,正如我们看到的那样,它既可以有交流意图,也可能没有交流意图。当 Jack 通过打开窗户来表达一切都很安全这样的意图时,他就执行了一个语内表现行为,这个行为表示一切都是安全的。如果 Sue 看见窗户打开了,并且判断出 Jack 的意图,从而知道一切都很安全。这样就发生了一个言语表达效果行为,该行为是要 Sue 相信一切都是安全的。

一些动词可以描述语内表现行为,而另一些动词描述言语效果行为。语内表现行为用于那些明确说明行为的句子中,称为显性施为句(explicit performatives)。例如,要表达我承诺要割草,可以说“I hereby promise to mow the lawn”。与此相似,在“I hereby inform you that your bank account is overdrawn”中“inform”描述一个语内表现行为。而动词“convince”描述了一个言语效果行为。对于这个言语效果行为,你可能想要它发生在你通知某人某件事时,但它的发生却不在你的控制范围以内。因此,我们不能说“* I hereby convince you that your bank account is overdrawn”。需要注意的是,即使其他 agent 没有认识到交流的意图,我们也能够完成言语效果行为。例如,Sue 可以把银行声明放在桌子上,Sue 不在时,Jack 也可以看到声明,这样 Jack 就知道他的账号被取消了。

自然语言理解中的一个困难问题是确定说话者的语内表现行为。同一个非语内表现行为,例如下面的句子:

Do you know the time?

可能是问一个是/否的问题,也可能是问时间,也有可能是告诉别人时间,这要取决于说话者的意图。言语行为理论的一个关键问题是确定非语内表现行为潜在的意图。当然,语言中有很多方法来明确地表示意图。一种方法是利用显性施为句。但是,这种情况下表达出来的句子是一种很正式、做作的语言。另一种方法是单词“please”的利用。如果把“please”加入到这个句子中,就不能解释为是/否的问题。不过,此类明确的标记在真实的英文中是很少见的。因此,根据上下文来识别意图是自然语言理解的关键问题。

语言作为多 agent 活动所产生的最终结果是,需要采取一些方法来协调交流行为,并判断语言是否被真正理解了。在对话中,agent 采用多种机制来表示彼此理解对方的意思。这包括明确的说明(例如“I understand”)、简单的确认(例如“OK”)和其他方法(例如,重复前面句子的一部分)。一个典型的例子是人们交换电话号码。通常,第一个人说电话号码,第二个人重复一遍,第一个人再进行确认。这些步骤都是两个 agent 对所交流内容达成一致的方法。确认对方的讲话是每一个对话系统的关键部分。

17.3 认知状态:信念的表示

表示 agent 信念的方法有很多,根据你希望该 agent 对它自己和外界感知程度的不同而有所不同。例如,在前面几章中,我们一直利用事实知识库。我们可以认为 agent 相信知识库中的任何信息,这就够了。但是,这样的 agent 其能力可能非常有限。例如,它不能表示其他 agent 信念的信息,所以不能交流。另外,agent 没有自我意识,不能对它所相信的内容和它所观察到的世界的真实情况加以区分。所以,对于怎样使用感知来获取周围世界的更多信息的方法,这里很难给出一个很好的描述。

最简单的扩充办法是把知识库分为几个不同部分,我们称每一部分都为一个信念空间(BS, belief space)。一个空间就是一系列的谓词,也就指定了一个知识库。每一个空间都可以用来表示一些 agent 的信念。例如,一个双方的会话,我们可能需要两个空间,一个空间表示其中一个 agent 的状态,另一个空间表示另一个 agent 的状态。但是,这并不是单个 agent 的直观模型,因为在这个模型中似乎这个 agent 能够直接获取另一个 agent 的信念。更好的做法是,所有的信念与我们要刻画的 agent 相关。为了做到这一点,我们定义了一个谓词 BEL,它的输入是一个 agent 和一个空间。如果这个空间刻画了这个 agent 的信念,这个谓词就为真。由于使用这个谓词的命题可以在其他空间中出现,因而可以定义一个层次化的信念空间。图 17.2 给出了一个简单的例子。这幅图的两边表示了同一个命题集合。首先采用 8.3 节所描述的模态运算符 Bel 来表示信念,然后采用了信念空间的方法。这个知识库首先断定 Jack 的信念可以用空间 BS1 来刻画,其中包括命题“Fido is a dog”(Fido 是一条狗),“dogs bark”(狗会叫)以及“Sue’s beliefs are described in space BS2”(Sue 的信念是在空间 BS2 中描述的)。空间 BS2 描述了 Jack 认为 Sue 是这样想的:“Fido is a dog”(Fido 是一条狗),“dogs bark”(狗会叫),“all dogs that bark are fierce”(所有会叫的狗都很凶猛)。

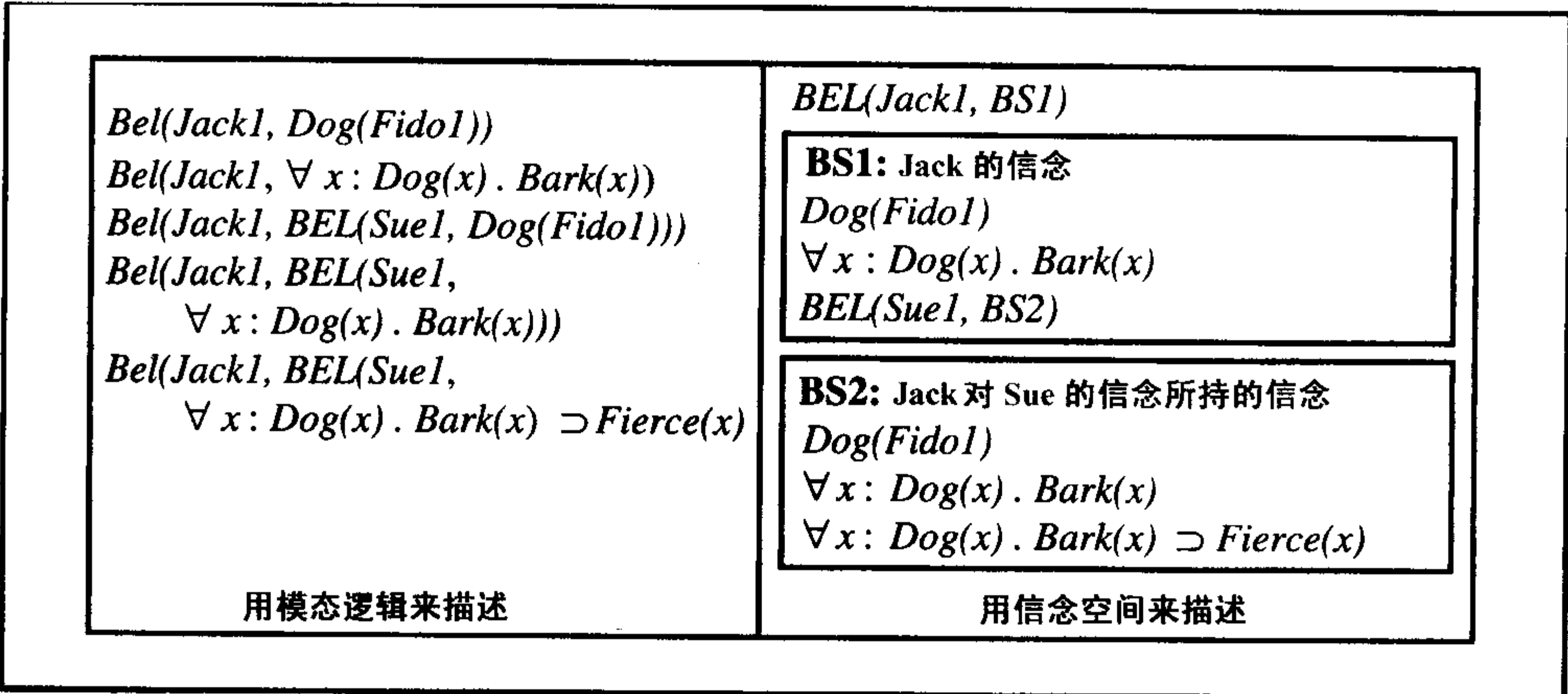


图 17.2 用信念空间表示信念

在这个模型中,推理总是与某个空间相联系。例如,根据空间 BS2,我们能够知道 Fido 很凶猛,最后得出结论,Jack 认为 Sue 的观点是 Fido 是凶猛的。在空间 BS1 中,我们不能得到相似的结论,所以 Jack 并不认为 Fido 是凶猛的。

对于“相信某事”的含义,这产生了一个重要的特征。一个 agent 的显式信念(explicit belief)是适当空间中的命题,隐含信念(implicit belief)是指那些由外在信念经过一些推理过程得到的命题。当我们说“某人相信某件事”时,例如说“Jack believes the world is flat”(Jack 相信世界是平的),我们一般是指与显式信念的概念相似的某些事情,可能再加上所允许的非常有限的一些推理。但是考虑这样的情形,Jack 有一些信念,这些信念逻辑上蕴涵“世界是平的”这样的意义,但是 Jack 从来没有进行这样的推理并得到这样的结论。在这种情况下,我们不能认为 Jack 相信“世界是平的”。

框 17.1 信念的宣告性模型

定义信念运算符的语义有两种方法。第一种方法把 Bel 当做模态运算符,并如 8.8 节所介绍的给出可能的世界语义。运算符 Bel 和 Know 的惟一不同点是 Bel 的访问关系不是自反的。那么,虽然

$$Know(P) \supset P$$

是“know”的公理,但不是 Bel 的公理。因为 agent 相信某件事并不说明这件事就必须为真。

另一种方法与信念空间表示的直觉很相似。Bel 运算符把 agent 与一个命题联系在一起,这就要求我们对逻辑的框架进行扩展,以便能够把命题当做对象。这种扩展是通过“quotation”(引用)运算符来实现的。虽然“Happy(Jack1)”是一个命题,而表达式「Happy(Jack1)」则是一个表示这个命题的项。这样,你就可以把“Jack believes that he is happy”表示为 Bel(Jack1,「Happy(Jack1)」)。经过这个扩展以后,我们需要重新构造所有对 agent 的信念进行推理所需要的公理,因为通常的公理都不能应用于引用表达式。Haas(1986)和 Konolige(1986)就给出了这样的例子。

你相信什么和你认为别人相信什么这两者是不同的,认识到这一点非常关键。当两个说话者具有不同的信念时,这可以很明显地看出来。假定 Jack 相信柜子上安装的是号码锁,但是 Jack 相信 Sue 认为柜子上是用钥匙开的锁。当 Sue 向 Jack 要钥匙时,Jack 应该能够认识到 Sue 是要打开柜子,即使在 Jack 看来用钥匙来打开柜子是一个错误的计划。

另一个例子,信念模型对于确定名词短语的指代是至关重要的。例如,假定 Sue 知道 *The Revenge of Mrs. Smith* 这本书在 Jack 的办公室,但是 Sue 相信 Jack 认为这本书在休息室的桌子上。那么,如果 Jack 问 Sue,“Have you read the book on the table in the lounge?”(你是否读过休息室桌子上的那本书?)Sue 要意识到 Jack 是指 *The Revenge of Mrs. Smith* 这本书,即使 Jack 对这本书的描述是不准确的。

这样,一个 agent 对于另一个 agent 所持信念的信念在自然语言理解中是非常重要的。那么,一个自然的问题就是,信念需要嵌套几层才能够理解所有语言?很容易看出来,三层嵌套是必需的。假定 Jack 相信 Sue 认为她刚刚成功地对他说了一句“外面在下雨”的谎话。在这种情况下 Jack 相信:

Jack 相信外面没有下雨。

Jack 认为 Sue 的想法是:外面没有下雨。

Jack 认为 Sue 的想法是:Jack 以为外面在下雨。

虽然 Jack 和 Sue 都认为没有下雨,但是 Jack 认为 Sue 的想法是,Jack 以为外面在下雨。这是因为 Jack 所持的想法是 Sue 自认为她撒谎是成功的。

还有很多需要更深层嵌套的例子。解决这个问题的方法是引入共享知识库的概念。其中,共享知识库是指两个 agent 都知道的知识以及了解对方知道什么。共享知识来自于 agent 所处的相同背景和处境,以及关于世界的一般知识(例如一些行为的普遍性、一些事物的标准类型层次体系,我们所处的社会的一般性事实,等等)。对于互相了解的或共享一个通用领域的两个 agent 来说,由于以前的相互交流以及在该领域的教育背景,它们将具有大量的共享知识。如果 agent 只利用共享知识制定了规划,则能够合理地确信该规划能够被成功识别。这样,共享知识在成功的交流中起了关键作用。虽然某一个 agent 的信念在一个特定会话中起决定性作用,但是,要解释和理解对方的行为,最需要的知识将是共享知识。

17.3.1 关于其他 agent 所持信念的知识

建立信念模型中最棘手的问题就是我们不知道其他 agent 知道什么(虽然我们知道其他 agent 知道一些信息)。例如,Jack 可能认为虽然自己不知道 Sue 的妈妈的名字,但 Sue 应该知道她妈妈的名字。因为 Jack 不知道 Sue 的妈妈的名字,所以在信念空间中就没有 $\text{NameOf}(\text{Mother}(\text{Sue1}), \text{xxx})$ 这种形式的公式。因此,我们不能通过简单地把这个事实加入到合适的信念空间中的方法来描述这个场景。如果我们利用一个 Skolem 常量(例如 Sk33)并把 $\text{NameOf}(\text{Mother}(\text{Sue1}), \text{Sk33})$ 加到 Sue 的信念空间中,就可以简单地断言 Sue 相信她妈妈有名字,而这并不符合事实。在模态逻辑中,这等价于下面的公式:

1. $\text{Bel}(\text{Sue}, \exists! x . \text{NameOf}(\text{Mother}(\text{Sue1}), x))$

在模态逻辑方法中,这个问题的一般解决方法是允许量词超出信念运算符的辖域。这样,公式 1 指 Sue 相信她妈妈有名字,而

2. $\exists! x. Bel(Sue1, NameOf(Mother(Sue1), x))$

指她实际上知道这个名字是什么。遗憾的是,大部分的知识表示方法都是利用 Skolem 量化方法来处理存在性的量词,而简单的 Skolem 化方法没有保留带模态运算符的量词辖域。不过,通过在创建它的信念空间中为每一个 Skolem 和变量建立索引,我们可以刻画这种辖域特性。这样增加一个对应于公式 1 的事实到知识库中,将创建一个依赖于信念空间 BS2(即,Jack 认为 Sue 的想法是什么)的 Skolem,由此产生的事实:

3. $NameOf(Mother(Sue1), Sk1_{BS2})$

将增加到空间 BS2 中。增加与公式 2 对应的事实将创建依赖于 BS1(即,Jack 相信的事情)的 Skolem,并会将下式

4. $NameOf(Mother(Sue1), Sk2_{BS1})$

加入到空间 BS1 中。

保持存在量词辖域的方法必须有对应的证明方法来验证存在命题,并能够区分一个类似于“Does Sue believe that her mother has a name?”的查询和另一个“Does Sue know her mother's name?”的查询之间的不同。要做到这一点,推理系统必须能够限制变量以便使它只与合适的信念空间中定义的对象相匹配。假如公式 1 被证实,导致公式 3 被加入到空间 BS2。现在假定查询是公式 2,利用标准的方法,公式 2 将转换为以下的查询形式:

5. $Bel(Sue1, NameOf(Mother(Sue1), ?x))$

这将导致在信念空间 BS2 中对形式为 $NameOf(Mother(Sue1), ?x)$ 的公式进行查询,而这将与公式 3(与断言 1 对应)合一。因为变量 $?x$ 应该只与空间 BS1 中定义的对象匹配,而 $Sk1$ 是在空间 BS2 中定义的,所以问题就出现了。如果也根据变量所属的空间对变量进行索引,就可以避免这个问题。具体来说,我们需要对合一进行扩展,使得一个空间 X 中的变量只能匹配一个对 X 可访问的空间中的对象,可访问关系可以用信念空间中的嵌套层次关系来定义。这样,空间 BS1 中定义的常量可以在 BS2 中得到,反之则不可行。图 17.3 表示了信念空间的层次关系,并详细说明了什么变量能够与什么常量匹配。

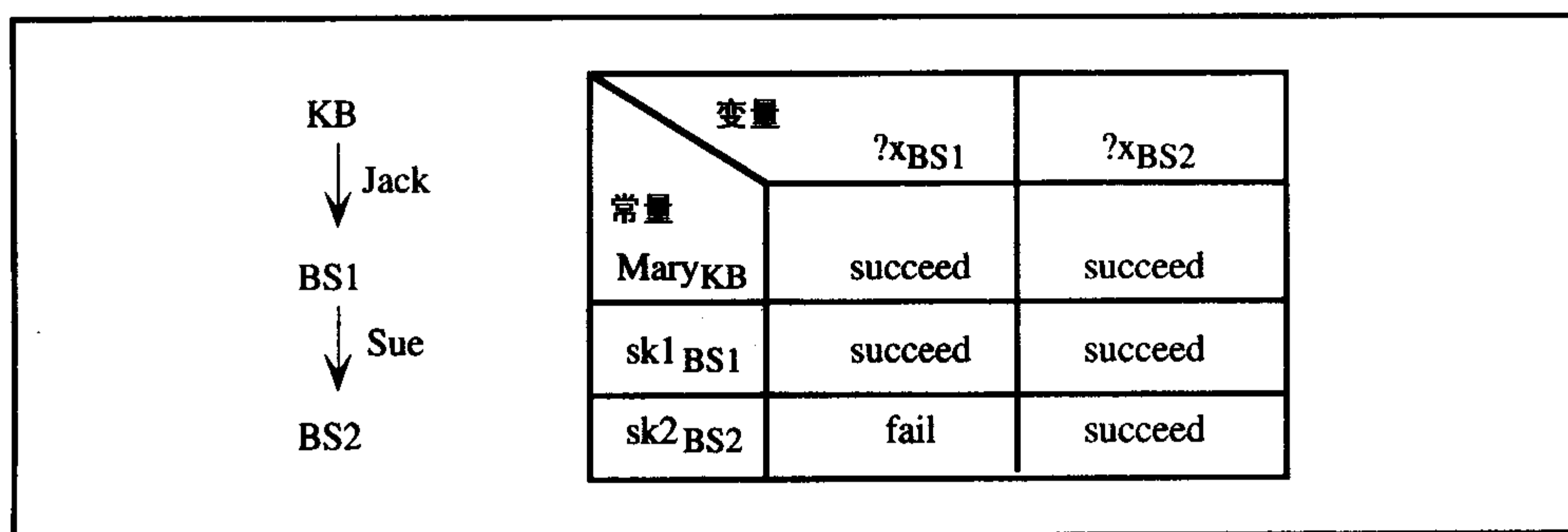


图 17.3 将变量和信念空间上的常量相匹配

为方便起见,我们可以定义一个新的运算符“KnowRef”,这个运算符表示了一个 agent 知道满足某些条件(条件用 λ 表达式来表示)的惟一对象。具体到这里就是:

6. $KnowRef(A, \lambda x P_x) \equiv \exists! y. Bel(A, ((\lambda x P_x) y)) \equiv \exists! y. Bel(A, P_y)$

给定这一公式,公式 2 等价于:

$$\text{KnowRef}(\text{Sue1}, \lambda x \text{NameOf}(\text{Mother}(\text{Sue1}), x))$$

知识的另一个重要的形式包括知道其他人知道某一事实是否为真(但最开始的 agent 不知道这个事件是否为真)。例如,Jack 可能认为 Sue 知道她自己是否拥有 Fido。这表示为模态逻辑中信念的析取,即 Jack 认为, Sue 认为她自己拥有 Fido, 或者 Sue 认为她自己没有 Fido。这样,就有一个与前面讨论过的量词辖域相类似的特性。如果析取是在信念运算符范围之内:

$$7. \quad \text{Bel}(\text{Sue1}, \text{Own}(\text{Sue1}, \text{Fido}) \vee \neg \text{Own}(\text{Sue1}, \text{Fido}))$$

那么 Sue 认为她或者拥有 Fido 或者没有。Sue 或其他任何理性 agent 几乎都认为这样。析取在信念运算符外面的情况为:

$$8. \quad \text{Bel}(\text{Sue1}, \text{Own}(\text{Sue1}, \text{Fido})) \vee \text{Bel}(\text{Sue1}, \neg \text{Own}(\text{Sue1}, \text{Fido}))$$

表示 Sue 很清楚自己是否拥有 Fido。这很难根据信念空间来表示。这里,我们引入一个新的运算符“KnowIf”,它刻画了公式 8 所表述的意义:

$$9. \quad \text{KnowIf}(A, P) \equiv \text{Bel}(A, P) \vee \text{Bel}(A, \neg P)$$

使用信念空间把这个公式化简为一个表达式需要引入另外的机制,这种机制将允许明确地断言一个特定的公式是在一个信念空间中,然后可以根据这种关系的析取来定义“know”。不过,我们在表示中将直接利用“KnowIf”运算符,而不分解为更基本的形式。

我们要讨论的最后一个方面是最难处理的,不过它比前面讨论的“知道”(knowing)的两种形式重要得多。事实上,前面讨论的两种形式可以看做对其他人所了解信息的特殊情形。具体来说,如果你的音响坏了,希望有人能帮忙修,该怎么办呢? 你知道某个人,例如音响维修店的 Joe 可以帮你。Joe 知道修理音响的一些事情,可能告诉你一些需要知道的信息。但是你不知道为了修理这个音响,Joe 应该知道一些什么样的知识。你甚至也不知道该用什么词汇来表示这种知识。不过,你对 Joe 的信念和知识还是有一些了解的,这足以让你能够推导出结论,Joe 就是你应该向他询问有关音响的问题的那个人。

目前来说,只有非常有限的技术可用来解决这个问题。典型的做法是,把这种知识嵌入到为系统用户手工编写的启发式知识之中。例如,对每一类行为,系统明确地表示要执行这个行为,用户应该知道些什么。然后,系统可以引入只有部分描述的行为类型。例如,系统定义了“FixStereo”的前提条件和结果(可能只有其中一部分),但是系统可能没有实现这一行为的具体步骤。后来,我们又可以断定 Joe 知道怎样执行 FixStereo 这种动作。这样,给定目标 Working(S1)(S1 为音响),系统可能发现 FixStereo 与之相关,因为其效果能够与我们的目标相匹配,但是不能将这一行为进行分解。然后,系统会查询它认为的知道怎样执行这一行为的那个 agent,我们称这一关系为 KnowingHow。

17.4 认知状态:期望、意图和规划的表示

文献中使用的很多术语都与期望和意图相关,例如“plan”(规划),“goal”(目标),“want”(需求)和“purpose”(目的)。这些术语描述为会话 agent 建模所需要的一系列概念,这一节我们将试图给出这些术语的定义。当然,并不是所有的研究者都按照这里的定义使用这些术语,不过这里给出的是比较常见的解释。

首先,考虑期望和意图的区别。期望表示的是 agent 对世界的观察中,哪些状态是乐意接受的,哪些状态是不乐意接受的。一个 agent 可能具有很多期望,并且常常互相冲突。例如,你可能想让自己放松一天,去海边玩;而另一方面,你还希望期末考试能够考好,为此需要待在家里学习。这两个期望互相冲突,你的行为将是这两者的折中。与此相反,你的意图通常不冲突,即你不可能同时想去海滩又想待在家里学习。意图是与行为紧密相关的,一次只能够做一件事。

虽然期望和意图都影响你的行为,但它们是很不相同的。期望是潜意识的,你可以评估特定期望对你的吸引程度。而意图表示你要采取的行为是经过深思熟虑的。再如图 17.1 所示的例子,你可以看到期望和信念都可以认为是一些思考过程的输入,而这些思考过程将逐渐落实为你的意图所决定的一系列行为。

意图有两种不同的概念。第一种意图称为行为中的意图,指故意行为这样一个特征。例如有两种场景,第一个场景中你骑自行车不小心撞到别人,在另一个场景中你骑自行车故意撞到别人,这二者具有很重要的区别。如果是后者,你可能被控伤害罪,而前者只是遗憾的事故。第二种意图称为以未来为导向的意图,反映的是你对将来行为所做的决定。例如,经过考虑以后,你决定暂时忘记考试而去海滩玩。于是,你已经采纳了一个以未来为导向的意图,去海边玩。

当然,如果你有以未来为导向的意图,那么将导致有意图的行为。因为你想要去海滩,你可能打算骑自行车去。但是,以未来为导向的意图并不一定能够实现。在你去自行车棚的路上,你可能遇见朋友要去学习,因此你對自己出去玩的决定很后悔。在这种情况下,虽然有去海滩的意图,但是你放弃了,然后有了去学习的意图。

这里以未来为导向的意图是很重要的概念。我们引入运算符 Intend 把 agent 和 agent 想要执行的行为联系起来。作为一个理性 agent,所具有的意图必须满足一些要求,其中部分要求如下:

理性约束 1——一个 agent 不能有意图去做两个自己认为是互相排斥的行为。

理性约束 2——一个 agent 不能有意图去完成自己认为都不可能实现的一个行为。

用形式逻辑来表示这些约束是很复杂的,我们将依赖于规划制定和规划采纳的过程,从程序上保证 agent 只采纳它认为能够完成的意图。

很明显,规划、目标和规划制定都与意图有密切的关系。但是,它们之间的关系比看起来的要复杂得多。首先,如果你说 Jack 有抢银行的规划,可能有两个方面的意思:

- Jack 要去抢银行。
- Jack 已经制定了一个让某人去抢银行的方案。

在第一个意义中,规划可能是指实现 Jack 意图的一种方法,而第二个意义是指 Jack 知道怎样去抢银行。在第二个意义中,一个规划,就像菜谱,是指达到某一结果的一系列行为一样。正如一个菜谱描述了达到某些结果的一系列行为一样。但是,有了菜谱并不意味着你想做饭。

菜谱式规划与第 15 章中描述的人工智能中的规划的概念相对应。当一个规划系统建立一个规划时,并没有任何 agent 来执行这个规划,而只是简单地建立一个达到目标的步骤。需要注意的是,与“有一个规划”的两层意义相对应,“目标”这个词也有两层意义。一层意义中,目标与意图相似;另一层意义中,目标就是菜谱式规划的结果状态。

现在,我们为这些不同的术语给出在本文中的解释。术语“规划”将总是用于表示菜谱式规划的意义。也就是说,一个规划就是要达到某一特定结果的一系列行为,这个结果被称为规划的目标。目标可以是规划中的行为要达到的那个结果的陈述,也可以是规划中的行为将要执行的某个行为。这样,术语“规划识别”指的就是建立一个菜谱式规划的任务,该规划可以体现为一系列可观察到的行为。

规划制定的另一个意义与意图的关系如何? 为了避免混淆,我们不用规划(plan)这个词,而用 agent 的意图结构(intentional structure)。一个 agent 的意图结构是其心理状态的一部分,类似于它的信念和期望。当 agent 决定执行一个规划时,就产生了意图结构。很自然地,规划的结构与所产生的意图结构之间有很强的对应关系。考虑最简单的情况。例如,Jack 知道一个简单的规划:骑自行车去海边。这个规划是一系列的信念,即如果 Jack 按照正确的方向骑自行车,他将能到达海边。然后,Jack 有去海边的愿望。由于了解骑自行车去海边的规划,他可能采纳这个规划。现在,他具有骑自行车去海边的意图。Jack 的关于规划的信念和意图结构如图 17.4 所示。注意,意图必须包括两个行为和这两个行为之间的关系。如果你忽略了骑自行车的行为,那么 Jack 虽然打算去海边,但没有明确的办法。如果忽略了去海边的行为,Jack 将骑自行车去,但不知道到什么地方停下来。如果忽略了生成关系,则 Jack 只能够完成每一个孤立的动作,例如,先骑自行车,然后走路到海边。

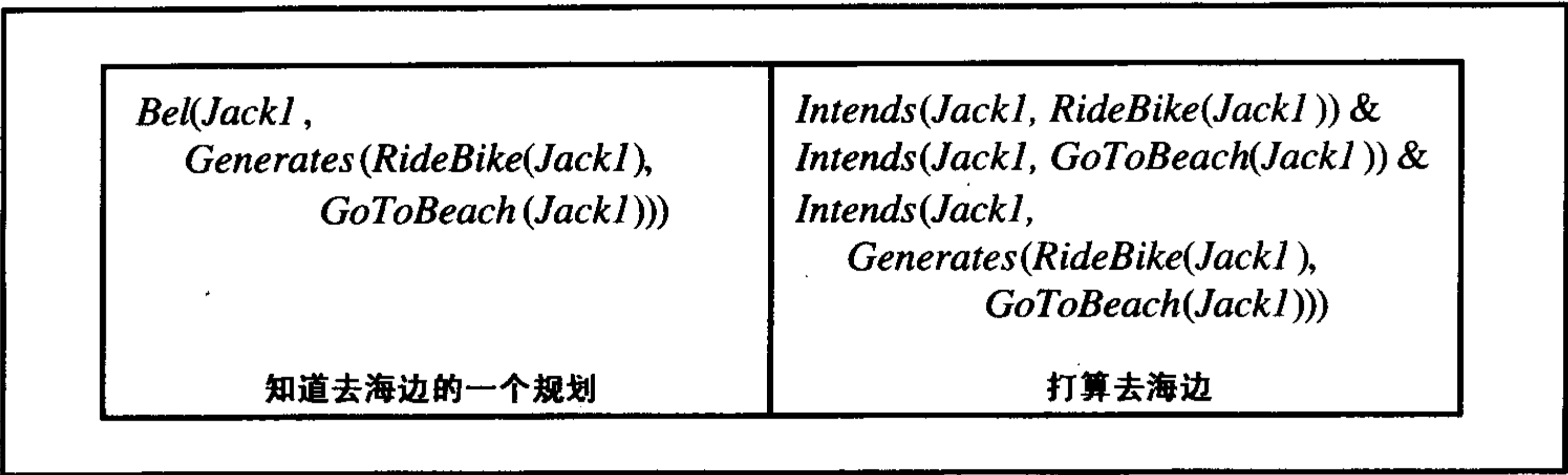


图 17.4 “Jack has a plan to go the beach”的两种解读

从字面上理解,我们常常忽略了解规划和具有意图之间的区别,这是因为在知识表示中,它们的表示方法非常类似。图 17.5 比较了 Jack 了解基于菜谱式规划和 Jack 有一个意图的区别,图中利用第 15 章介绍的基于图的表示方法,把规划图放在 Jack 认为可行的规划空间中。而另一方面,意图结构的一种自然的表示方法同样是把一种新的空间作为“Intend”运算符的参数。这样,除了空间的解释不同,这两种情况的规划图是相同的,因此引起了混淆。

最后,我们总结一下。下面这些是本节我们讨论的 BDI 模型中的一些概念和它们之间的关系:

- 信念——运算符 Bel 和信念空间。
- 以未来为导向的意图——运算符 Intend 和意图状态空间。
- 意图空间中的行为——执行的一个行为,因为它在意图空间中。
- 菜谱式规划——一个规划,常常表示为规划空间中的一个规划图。
- 拥有了意图意义下的规划——与以未来为导向的意图相同。
- agent 的目标——一个以未来为导向的意图。

规划的目标——规划图中的最后结果。
期望——不可分解的,它是在创建目标和评价规划的决策过程中实现的。
希望——期望的另一个表示方法。

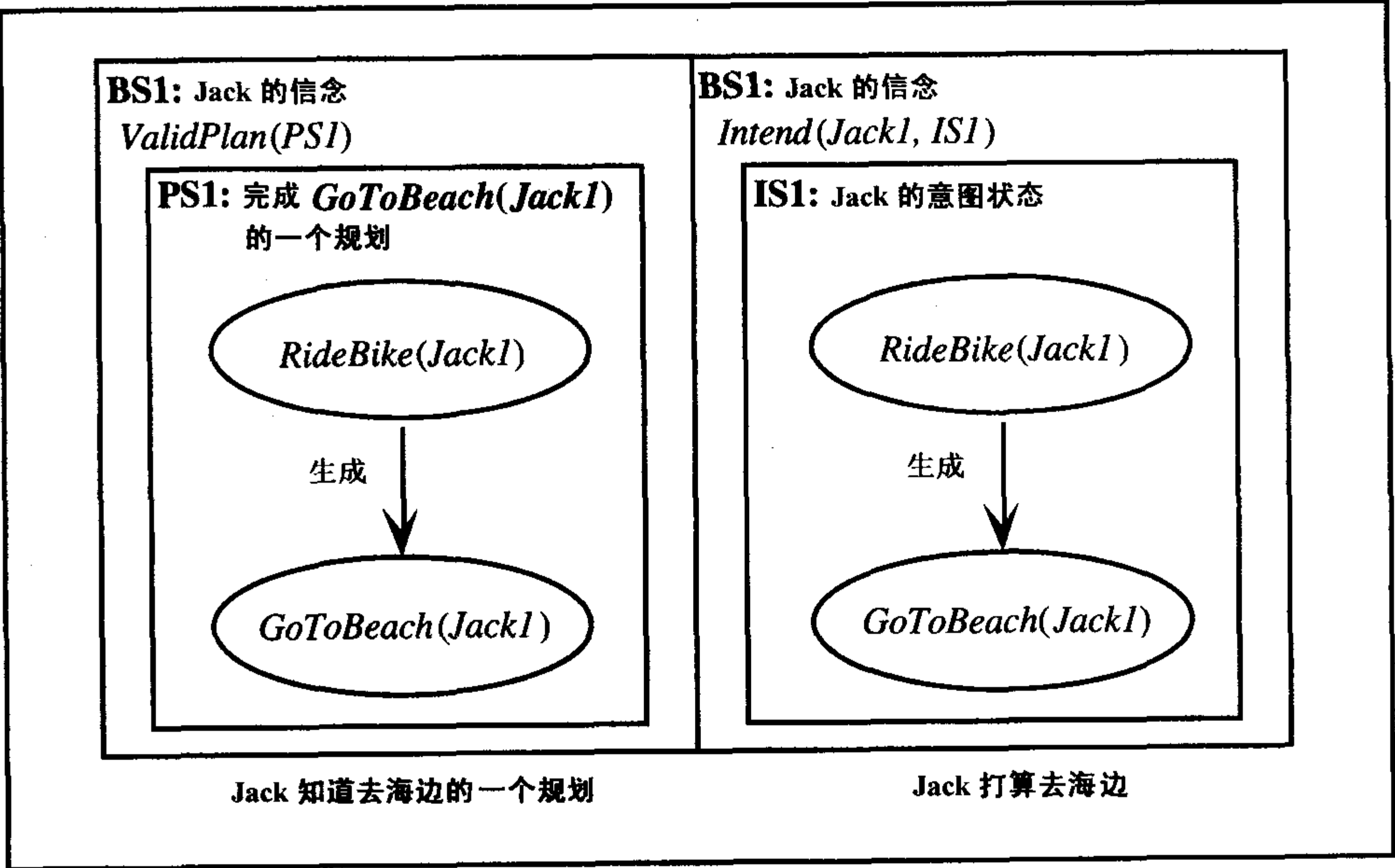


图 17.5 “知道一个规划”与“打算执行一个规划”的比较

17.5 言语行为和交流行为

如前所述,言语行为是指说话的行为,我们希望能够利用第 15 章提出的模型来表示。但是要做到这一点,需要研究好几个复杂的问题,这是因为交流行为的多 agent 的自然属性以及交流行为的结果将受 agent 认知状态变化的影响。

框 17.2 意图和行为的生成

在阐述意图的很多理论中,所谓的意图都是指某种行为。你的疑问可能是,为什么图 17.4 和图 17.5 中的意图表达式的形式是合适的? 特别是,这里的意图表达式似乎隐含着行为之间的生成关系。如果有这样的疑问,我们可以把意图表示为复杂的行为。在这种情况下,复杂行为就是“骑自行车去海边”。为此,我们需要定义合适的行为构造器来创建这些行为。Pollack(1990)引入了运算符 *By*,这样给定两个行为 α 和 β , $By(\alpha, \beta)$ 就是通过行为 β 来实现行为 α 的复杂行为。这样,图 17.4 中的第三种意图就是:

Intend(Jack1 , By(GoToBeach(Jack1) , RideBike(Jack1)))

这样,所有的意图都是根据行为来表述的。

为了达到规划的目的,言语效果行为是最重要的,因为言语效果行为描述了 agent 所期望的结果。例如,一个 agent 试图建立一个规划,其内容是说服其他 agent 相信某些事实或者说服其他 agent 应该执行某个动作。为了完成这个规划,这个 agent 将执行语内表现行为(如断定或请求),并希望其他 agent 做出合适的动作,以便完成言语效果行为。当然,为了完成语内表现行为,agent 将执行非语内表现行为,包括说出合适的句子。

这里的模型包括交流行为,这个交流行为与由语内表现行为生成的言语效果行为相对应。例如,交流行为 `ConvinceByInform` 通过说某些命题是正确的,让其他 agent 相信这些命题。考虑一下成功执行这一交流行为所涉及的步骤是有帮助的。如果 Sam 要成功地通过 `ConvinceByInform` 来说服 Helen 使她相信外面在下雨(“it is raining”),所需要的步骤为:

1. Sam 说 “It’s raining”, 其意图是通知 Helen 这一事实。
2. Helen 听见了句子 “It’s raining”, 并认识到 Sam 的意图是要通知她这一事实。这样, Helen 知道 Sam 是要让她相信确实是在下雨。
3. Helen 考虑了下雨的证据, 包括 Sam 说过在下雨这一事实。
4. 因此, Helen 开始相信确实是在下雨。

这些步骤都是必要的。为了证明这一点,考虑把下面的任意一步去掉。如果去掉步骤 1, 那么 Sam 没有说任何事情, 也就是没有通知 Helen 外面在下雨, 你实际上把非语内表现行为去掉了。如果只去掉步骤 1 中的意图条件, 将得到一个奇怪的场景: Sam 仅仅随机地发出一些声音(也就是说, Sam 处于被催眠状态)。即使 Helen 糊里糊涂地执行了其他步骤, Sam 也可以不承认他通知了 Helen 外面正在下雨。他可以说他不知道所说的内容, 所以 Helen 不能由此做出任何判断。

如果去掉步骤 2, 那么 Helen 没有听见这个句子。如果你只去掉步骤 2 中 Helen 识别出了 Sam 的意图这个条件, Helen 虽然听见 Sam 说话了, 但她不知道 Sam 为什么向她说话, 也就是说, Helen 没有识别出语内表现行为。即使她完成了其他步骤并最终相信外面在下雨, 但她还是可以否认是 Sam 通知了她这一事实。

如果去掉步骤 3, Helen 虽然意识到 Sam 试图通知她某些事情, 但她完全忽略了他告诉她的信息。虽然这样说至少是不太礼貌的, 但我们假设确实是这种情况, 即使她最后相信外面在下雨, 这个信念也与 Sam 所说的内容完全无关。

如果去掉步骤 4, 那么 Sam 告诉 Helen 外面在下雨, 但没有使她相信这一点。在这种情况下, 语内表现行为 `Inform` 成功了, 但是交流行为 `ConvinceByInform` 失败了。

虽然我们可以为言语行为中的三个层次的行为都给出表示方法, 不过, 最终为 agent 的推理提供了联系的主要还是交流行为。图 17.6 给出了在任何交流模型中都必需的两个基本交流行为的定义, 它们对应于简单的通知和请求。这些定义是建立在 agent 是真诚的这一假设上。例如, 对 “`ConvinceByInform`” 的一个约束条件是说话者相信所陈述的命题。这个假设条件会导致我们需要对类似说谎的行为进行处理, 但是这么复杂的情况已经超出了我们要建立的大部分会话系统所处理的范围。目前, 我们先不给出任务的分解(包括语内表现行为), 与解释语内表现行为相关的问题将在 17.9 节中阐述。

行为类 *ConvinceByInform(e)*:
 角色: *Speaker, Hearer, Prop*
 约束: *Agent(Speaker), Agent(Hearer), Proposition(Prop), Bel(Speaker, Prop)*
 前提条件: *At(Speaker, Loc(Hearer))*
 结果: *Bel(Hearer, Prop)*

行为类 *MotivateByRequest(e)*:
 角色: *Speaker, Hearer, Act*
 约束: *Agent(Speaker), Agent(Hearer), Action(Act)*
 前提条件: *At(Speaker, Loc(Hearer))*
 结果: *Intend(Hearer, Act)*

图 17.6 两个交流行为的定义

框 17.3 言语行为的分类

一个有意思的问题是:总共有多少种不同的言语行为? Searle(1979)认为只有 5 种类型的语内表现行为,每一种类型分别刻画不同的方面或目的。在一类语内表现行为中的每一个行为都是通过另外的条件来区分的。这 5 种类型的语内表现行为分别为:

表示型——说话者对他们真正要表达内容的真实性进行确认。例如,由 *inform*(通知), *deny*(否认), *affirm*(确认), *confirm*(确认)等动词描述的行为。

指示型——说话者试图影响另一个 agent 的意图和行为,包括由动词 *request*(要求), *command*(命令), *invite*(邀请), *ask*(询问), *beg*(请求)等表达的行为。

承诺型——说话者承诺完成将来的某个行为,包括由动词 *promise*(许诺), *commit*(答应负责)等描述的行为。

表达型——说话者要表达心理的状态和反应,包括由动词 *apologize*(道歉), *congratulate*(祝贺), *thank*(感谢), *welcome*(欢迎)等描述的行为。

公布型——说话者通过语言行为来完成一些习俗性的或仪式性的行为,包括由动词 *christen*(命名为), *fire*(解雇), *resign*(辞职), *appoint*(任命)等描述的行为。

在下一节,我们将考虑这样的模型,该模型允许一个 agent 像规划其他行为一样,对交流行为进行规划,以完成相应的目标。它为刻画用于理解其他 agent 的交流行为设定了一些步骤。

17.6 规划交流行为

考虑 Jack 试图买火车票这样一个场景。假定他不知道票的准确价格,同时假定他知道买票的一种规划。其中,涉及以下两个子步骤(根据图 15.4 中的“Buy”的定义):

10. *Give*(Jack1, Clerk1, Price(Ticket1))
11. *Give*(Clerk1, Jack1, Ticket1)

Jack 不能简单地直接把这个规划当做一些意图来采纳,这是因为步骤 11 涉及另一个 agent 的行为,而 Jack 无法直接试图完成这个行为。这一步是 Jack 的规划的关键部分,所以不能简单地忽略它。

事实上,有些事情 Jack 必须做,因为“giving”是一个涉及多个 agent 的行为。这个行为需要两个 agent,一个给予,另一个接受。这样,我们可以用一个 Jack 处于给予位置的行为来代替步骤 11。事实上,英语中有一个动词“receive”,但是为了使我们的分析更具有一般性,这里不使用这个动词,而是定义行为运算符“MyPartOf”。该运算符把一个 agent 和一个多 agent 的行为 A 作为输入,输出一个行为,这个行为是该 agent 在行为 A 中所承担的角色。这样,行为“MyPartOf(Jack1, Give(Clerk1, Jack1, Ticket1))”将涉及(1)Jack 伸出手,(2)等车票,(3)握住车票。行为“MyPartOf(Clerk, Give(Clerk1, Jack1, Ticket1))”将涉及(1)把车票递到 Jack 手中,(2)售票员松开手。需要注意的是,这个函数也能够适用于单 agent 的行为。如果 agent 是行为的施动者,那么这个函数为等价函数,例如,“MyPartOf(Jack1, Grasp(Jack1, Ticket1)) = Grasp(Jack1, Ticket1)”。另一方面,如果 agent 不是行为的施动者,这个函数就表示其他 agent 在做的过程中这个 agent 不能干预的一些行为。

这个规划中的意图可能包括以下行为:

12. *MyPartOf(Jack1, Give(Jack1, Clerk1, Price(Ticket1)))*
13. *MyPartOf(Jack1, Give(Clerk1, Jack1, Ticket1))*

这些行为是 Jack 至少试图要做的几件事情,但它们可能不是非常有效。具体来说,如果该售票员没有完成他的那部分行为,对于 Jack 来说,这样做没有任何用处。到目前为止,Jack 的意图还没有涉及到任何让该售票员配合他完成这个规划的问题。为此,Jack 将必须与售票员交流,让售票员具有正确的意图。

在这之前,考虑另一个问题。因为不知道票的价格,Jack 甚至不能执行第一个行为。这引出了到目前为止我们一直忽略的一个关于行为的一般性约束问题。要执行任何行为,一个 agent 必须知道参数的值。我们可以这样理解约束,每一个行为都存在一些隐含的前提条件,即 agent 必须知道参数的值。这有时也被称为知识性前提。在这个例子中,隐含的前提是:

14. *KnowRef(Jack1, $\lambda x(\text{Price}(\text{Ticket1}) = x)$)*

对 Jack 来说,这成了一个新的目标。直觉上,他可以有几种方法达到这个目标。他可以查价格表,或者可以问知道价格的人。后面这一方法涉及语言,所以这个例子中采用这个方法。假定 Jack 有理由相信通过售票员的一个“ConvinceByInform”行为能够达到这个目标。为了让售票员完成这个行为,Jack 可能计划要求售票员来完成“Inform”行为。但是,要求售票员完成的行为是什么呢? 因为 Jack 不知道票的价格,所以无法表示售票员必须采取的行为。其原因与前面类似,就是我们需要表示某人知道某事,但我们并不知道这件事到底是什么。解决方法是引入另外两个“Inform”行为来用于规划,这个规划利用了前面介绍的“KnowRef”运算符和“KnowIf”运算符,如图 17.7 所示。

总结目前的情况,Jack 的规划有两个问题:(1)没有理由期望售票员完成规划中相应的任务,(2)Jack 没有足够的信息来执行规划中相应的任务。如果 Jack 能够影响售票员的意图结构,使售票员能够告诉 Jack 车票的价格,并参与这一购买行为,这两个问题都可以克服。更具体来说,Jack 需要售票员进行下面的操作:

15. *MyPartOf*(*Clerk1*,
 ConvinceByInformRef(*Clerk1*, *Jack1*, $\lambda x(\text{Price}(\text{Ticket1}) = x)$))
16. *MyPartOf*(*Clerk1*, *Give*(*Jack1*, *Clerk1*, *Price*(*Ticket1*)))
17. *MyPartOf*(*Clerk1*, *Give*(*Clerk1*, *Jack1*, *Ticket1*))

Jack 不能仅仅依靠自己的一厢情愿来使这些行为成为现实,但是 Jack 可以试图通过某种方式让售票员获得执行这些行为的意图。更具体来说,他可以为每一个行为执行“*MotivateByRequest*”操作。首先,他可以规划执行下面的行为:

18. *MotivateByRequest*(*Jack1*, *Clerk1*,
 ConvinceByInformRef(*Clerk1*, *Jack1*, $\lambda x(\text{Price}(\text{Ticket1}) = x)$))

如果成功,将达到这个结果:

19. *Intend*(*Clerk1*, *ConvinceByInformRef*(*Clerk1*, *Jack1*, $\lambda x(\text{Price}(\text{Ticket1}) = x)$))

需要注意的是,这些行为都应该有“*MyPartOf*”运算符来表明是谁在做什么。但是,我们将忽略这一点,因为我们很清楚想要的内容是什么。公式 18 中的请求行为可以通过一个非语内表现行为来实现,例如“What is the price of a ticket to Toronto?”

行为类 *ConvinceByInformRef*(*e*):
 角色: *Speaker*, *Hearer*, *Pred_x*
 约束: *Agent*(*Speaker*), *Agent*(*Hearer*), *UnaryPred*(*Pred_x*),
 KnowRef(*Speaker*, *Pred_x*)
 前提条件: *At*(*Speaker*, *Loc*(*Hearer*))
 结果: *KnowRef*(*Hearer*, *Pred_x*)

行为类 *ConvinceByInformIf*(*e*):
 角色: *Speaker*, *Hearer*, *Prop*
 约束: *Agent*(*Speaker*), *Agent*(*Hearer*), *Proposition*(*Prop*),
 KnowIf(*Speaker*, *Prop*)
 前提条件: *At*(*Speaker*, *Loc*(*Hearer*))
 结果: *KnowIf*(*Hearer*, *Prop*)

图 17.7 用于获取信息的 *ConvinceByInform* 的各种形式

类似地,Jack 可能试图发送另外两个请求让售票员执行动作 16 和动作 17。但这样产生的对话不太自然,例如:

What is the price of a ticket to Toronto? (去 Toronto 的车票价格是多少?)

Please take this money I am offering you. (请接着我给你的车票钱。)

Please give me a ticket. (请给我一张车票。)

如果这两个 agent 的每一次交互都需要这些过程,那就太麻烦了。但是,Jack 怎样用更少的话语也能处理这些事情呢? 答案就在于一个信念,售票员也是一个 agent,他能够利用世界的常识来解释行为。而且,从售票员观察到的行为来看,她(或他)也能够看出 Jack 要做什么。我们在下一节将更详细地讨论这个问题。还是让我们分析在这种场景下,Jack 可以依赖于售票员来完成的一些推理过程。

假定他们对通常行为(例如购买)都具有相同的常识,则 Jack 可以信赖售票员,因为售票员能够利用这一信息来解释他的行为。例如,假定知道价格后他说“OK, please give me a ticket”,

这是对行为 17 的请求。Jack 可以相信,售票员会利用买票这个行为的常识,认为动作 16 也会发生,从而形成合适的意图。同样,Jack 也可以只说“OK”,就把钱递过去。在这种情况下,他期望售票员能够意识到他是在买票,他也期望售票员能够完成他自己的工作。另外,Jack 可能只简单地告诉买票这一抽象目标,其他的则依赖于售票员要知道具体的完成细节。例如,Jack 可能只说“OK,I'd like to buy one”,其他的要依赖于售票员完成具体的操作。

总之,在完成所有这些推理以后,Jack 试图采取的一系列行为包括:

20. *MotivateByRequest*(Jack1, Clerk1,
 ConvinceByInformRef(Clerk1, Jack1, $\lambda x(\text{Price}(\text{Ticket1}) = x)$))
21. *MyPartOf*(Jack1, *ConvinceByInformRef*(Clerk1, Jack1, $\lambda x(\text{Price}(\text{Ticket1}) = x)$))
22. *MotivateByRequest*(Jack1, Clerk1, *Buy*(Jack1, Clerk1, Ticket1))
23. *Give*(Jack1, Clerk1, *Price*(Ticket1))
24. *MyPartOf*(Jack1, *Give*(Clerk1, Jack1, Ticket1))

意图 20 包括询问车票的价格,意图 21 是听答案,意图 22 涉及买票的请求,意图 23 和意图 24 是买票的步骤。这些意图和售票员采取的动作一起,可以形成下面的对话:

- 25a. Jack: Hi. How much is a ticket to Toronto?
- 25b. Clerk: Twenty five dollars.
- 25c. Jack: OK. I'd like one. < Jack 把钱递给售票员 >
- 25d. Clerk: OK. Here. < 售票员把票递给 Jack >

可以触发这些意图的规划如图 17.8 所示。其中,为节省空间起见,Jack1 和 Clerk1 分别简写为 J1 和 C1,图中也没有给出“Buy”这一行为的详细分解过程。

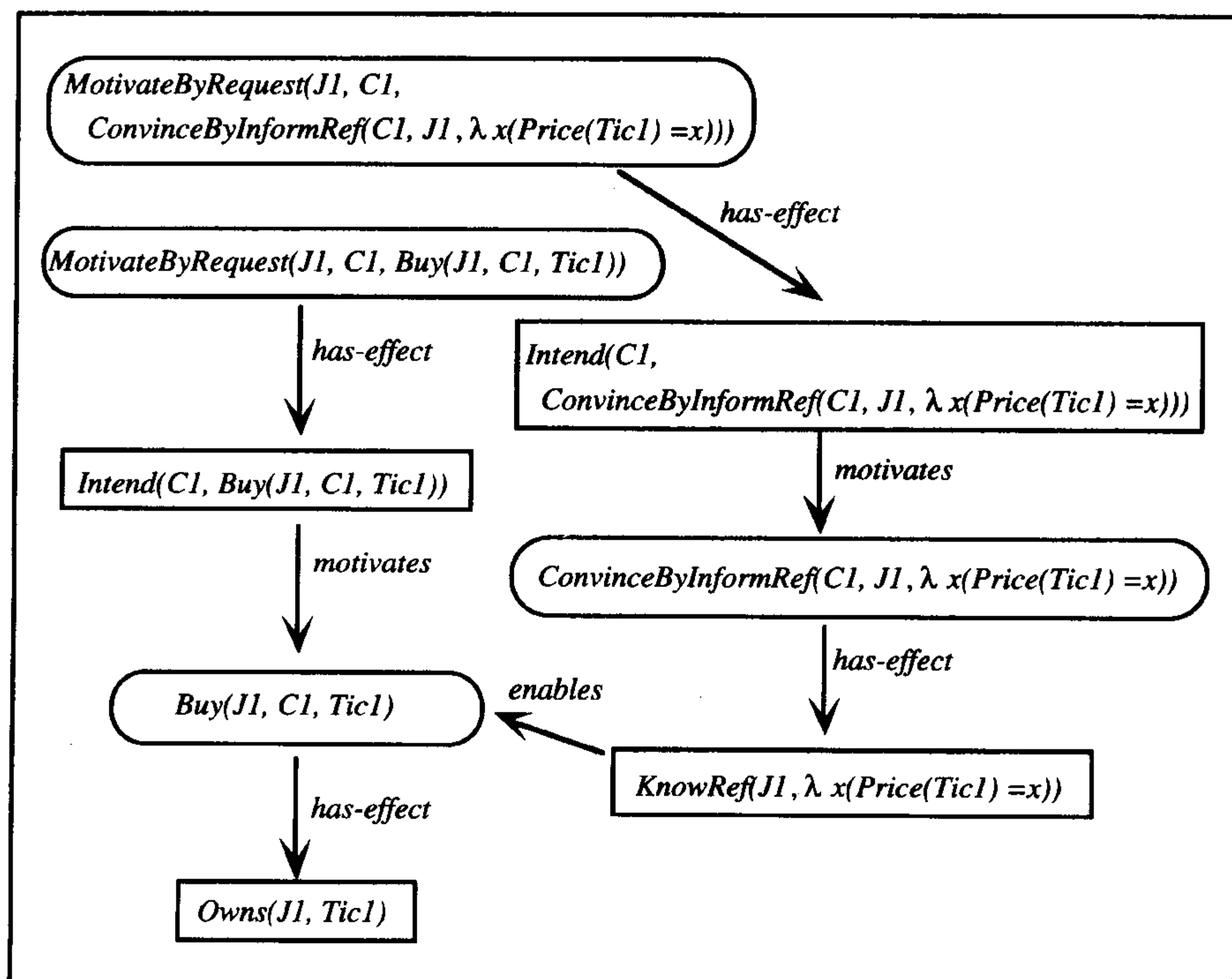


图 17.8 Jack 买票的最终规划

17.7 交流行为和意图判别

前面一节的例子说明,如果一个 agent 不能判别其他 agent 的意图,多 agent 的交流行为即使不是不可能的,也会变得非常困难。尤其是,在对话中理解并做出合适的行为几乎是不可能的。这一节,我们将探讨意图识别的问题以及它是怎样与言语行为解释相关联的。

第 15 章讨论了利用规划识别来发现句子中所描述的事件之间的因果和意图联系。同样的技术在这里看来也可以用于识别说话者的意图。对大部分问题,这可能是对的,但还是有一些复杂的情况。这一节中,我们把执行一个行为的 agent 称为行为 agent,而把观察这个行为的 agent 称为观察 agent。这里,主要探讨的问题是观察 agent 怎样才能确定执行 agent 的意图。

首先,仅仅因为你能识别出一个包括所有观察到的行为的规划,并不意味着这个规划描述了该 agent 的意图。有很多可能的规划能够解释观察到的行为,但是执行 agent 只执行其中的某一个规划。所以,用规划判别算法发现一个规划仅仅给出了执行 agent 的一些可能的意图。事实上,要确定这个规划是否正确从理论上说是不可能的,但基于我们能够判断的最可能规划来确定 agent 的意图,在一定程度上还是可行的。

正如 17.3 节中提到的,给定这个行为 agent 的信念,识别出的规划一定是合理的;而从观察 agent 的角度看,这个规划可能是错误的或不充分的。事实上,这意味着规划判别只能基于共享知识,即使一些关于其他 agent 的信念的特定事实在某些情况下也是可用的。如 17.3 节中的柜子的例子,Jack 知道他的柜子装的是号码锁,而 Sue 认为他的柜子装的是一个用钥匙开的锁。他们共享的知识是,需要开锁来打开柜子、不同交流行为的效果的知识以及有关这个场景的其他知识。例如,他们双方必须都知道 Sue 想打开 Jack 的柜子是合情合理的。这样,Jack 能够从下面的问题中识别出什么呢?

Sue: Can I have the key to your locker? (能给我你的柜子钥匙吗?)

跳过 Jack 是怎样把非语内表现行为确定为索取钥匙的具体细节,他可以得出一个合理的包括下面行为的规划。首先,这句话被确定为下面行为的一部分:

MotivateByRequest(Sue1, Jack1, Give(Jack1, Sue1, KeyTo(Locker1)))

利用关于交流行为的共享知识,Jack 知道 Sue 相信这个行为的一个效果是:

Intend(Jack1, Give(Jack1, Sue1, KeyTo(Locker1)))

它利用意图行为的共享知识,反过来将触发下面的行为:

Give(Jack1, Sue1, KeyTo(Locker1))

给定关于给予行为的共享知识,上面的行为将会有以下效果:

Have(Sue1, KeyTo(Locker1))

关键步骤是 Jack 相信 Sue 认为 Jack 的柜子上装的是一把用钥匙开的锁。这样,他就知道 Sue 认为有钥匙就能够完成这个行为:

Open(Sue1, Locker1)

假定 Jack 发现这个行为在这个环境下是一个合理的意图,他将对为什么 Sue 向他要钥匙建立一个合理的解释。当然,建立的规划是无效的,但是这对于识别这个规划的目的并没有影响。

重要的是 Sue 认为它是有效的。假定这是惟一看似合理的规划,他将假定它给出了 Sue 的意图结构。识别的规划如图 17.9 所示。

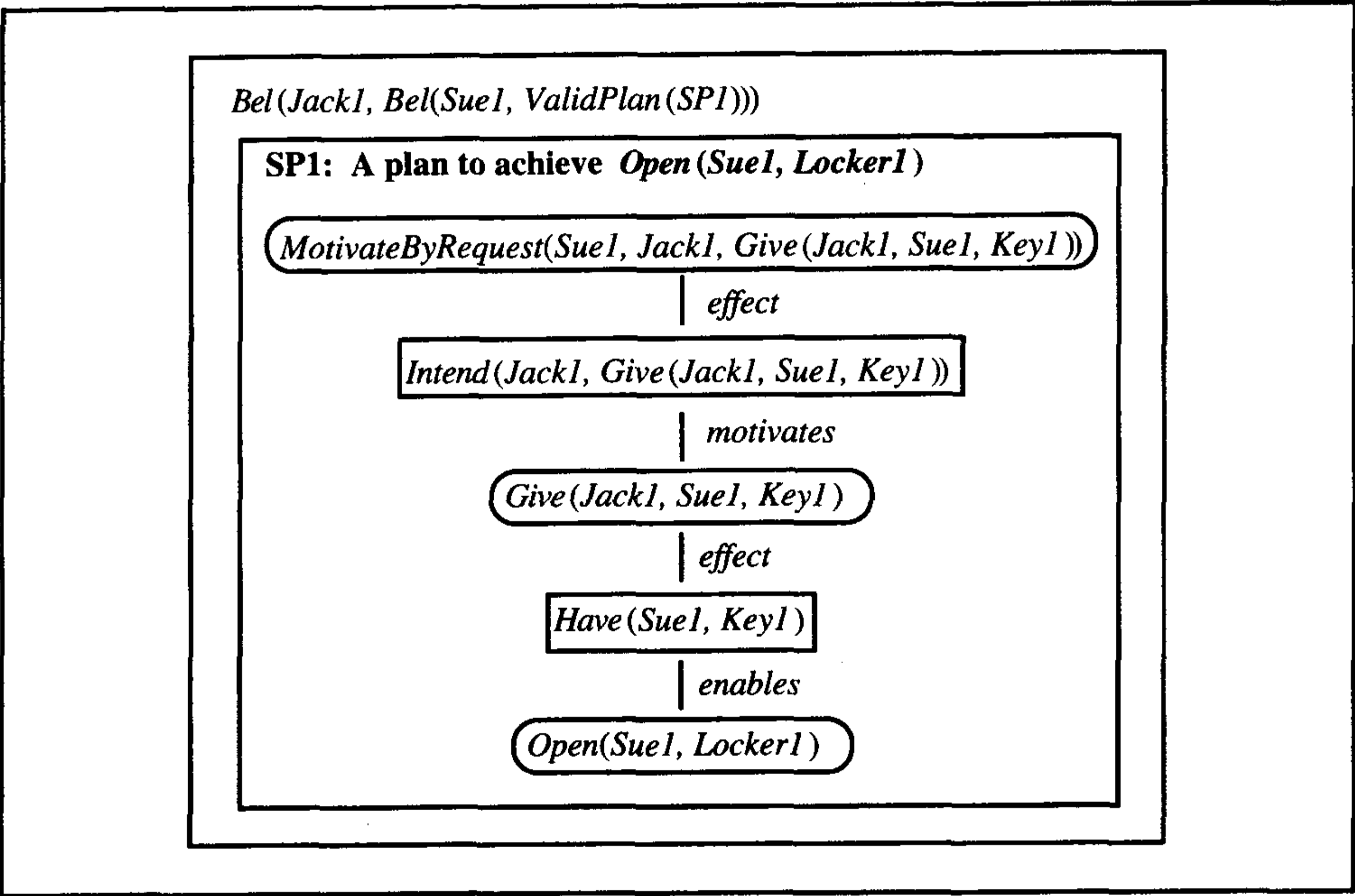


图 17.9 从“Can I have the key to your locker?”中识别出来的一个可能规划

给定这个规划,Jack 猜测 Sue 的主要意图包括:

```
Intend(Sue1, MotivateByRequest(Sue1, Jack1, Give(Jack1, Sue1,
KeyTo(Locker1))))
Intend(Sue1, Achieve(Have(Sue1, KeyTo(Locker1))))
Intend(Sue1, Open(Sue1, Locker1))
```

事实上,一个利用了这一算法的计算机系统也许已经接受了一系列可能的意图,这些意图被看做这个系统的期望。在这个例子中,Intend(Sue1, Open(Sue1, Locker1))就是其中的一个期望。给定一个输入,系统能够利用规划识别来给出一些合理的规划,并希望其中只有一个规划能够与期望的意图相匹配。

当然,我们不需要识别每一个句子隐含的最终意图。只要有能够触发这一句子的合理意图,识别出准确的意图往往是不必要的。例如,我们假定不存在 Sue 希望打开 Jack 的柜子这个期望。这毕竟是一个使这个例子简单化的特定意图。但是,这不是期望的意图,我们期望的意图是更全局的事情,例如去学习或者去海边等。现在,假定共享知识为,Jack 的柜子里有 Jack 和 Sue 正在学习的课程的教材以及一些浴巾。当 Sue 要柜子钥匙时,Jack 不能确定 Sue 的意图——Sue 可能是想去学习或者去海边。不过,图 17.9 给出的规划是这两个规划的共同部分。这两个规划在下一步开始分叉,因为不知道 Sue 拿出书还是浴巾。现在,Jack 也许有兴趣知道 Sue 的意图到底是什么,但是要理解 Sue 的需求,Jack 并不一定要确定 Sue 的意图。Jack 在图 17.9 所描述内容的基础上已经确定了 Sue 提出需求的动机,并且对这几个期望的意图来说,打开柜子都是合理的。因此,这个分析可能是对的,因为它能够成立。

另外,为了要让自己认为已经理解了对方,需要识别的意图的深度取决于你在交互中的目标。在一些会话中,例如当谈判一个和平计划时,准确地了解其他人的意图是非常重要的。在其他会话中,例如某人在大街上拦住你,向你询问时间,除了想知道时间这个明显的意图以外,你可能对此人的其他意图没有任何兴趣。在特定领域的计算机系统中,所要求的详细程度需要根据要完成的任务的范围来确定。

一旦识别了意图,下一个要解释的事情就是,对于与行为 agent 的意图相关的信息,观察 agent 做了什么事情。例如,当 Sue 向 Jack 要钥匙时,Jack 下一步该怎么做? 这个问题又引出了 agent 的规划和推理过程,我们会在下一节讨论这个问题。

17.8 对话中意图的来源

如果我们回头再看图 17.1,将看到这些意图是 agent 的信念和期望综合导致的结果。agent 相信它处于某个状态或将会处于某个状态,并根据 agent 的期望对这个状态进行评价。如果 agent 继续按照目前的情况发展下去,可能会出现某种状态,而改变这种状态的决策,会产生新的意图。我们将期望表示为状态的函数,这个函数衡量不同状态符合我们的期望的程度。基于这样的形式化表示,决策的过程就是不断努力去达到或维持一个更符合我们期望的状态。

一个特定的会话 agent 将具有其自己的期望。其中,这个期望是由其所在领域确定的,并且是这个领域的一个部分。例如,一个简单的数据库问答系统可能只有一个期望,即回答所有输入的问题。这样,它尽可能使自己回答所有问题。并且,当它发现自己有一个问题没有回答时,将通过回答这个问题让自己处于一个更符合期望的状态。但是,即使这么简单的 agent,如果你仔细考虑的话也是非常复杂的。

如果 agent 关心的所有事情不是那些没有回答的问题,那么,就可以简单地不处理任何输入。这样,它就永远不会知道它处于有一个问题没有回答的状态,因为本来问题就不存在。所以,对于 agent 的行为需要有一个约束,我们称之为专心性约束。当问一个问题时,它试图去理解这个问题。这样,这个行为可能不是来自于期望——它可能是作为一个自动的过程嵌入到系统中,一旦输入问题就触发这个过程。这是很自然的方法,对刻画人们的交流过程也是这样。如果有人和你说话,你不可能不解释这个人所说的内容。你可以对说话内容不做出任何反应或忽略说话者,但是很难不去理解你听到的这个句子。所以,回到我们讨论的会话 agent,我们将假定一旦有输入,语言理解过程就自动触发。

假定 agent 不能对输入的问题视而不见,那么有什么方法能够防止该 agent 随便给出一个答案呢? 如果这个 agent 系统是有用的,就一定有原因能够解释为什么该系统应该给出准确答案。利用 BDI 模型,这意味着 agent 必须相信它自己所说的内容,这也称为真诚性条件,在言语行为理论的很多方法中起着重要作用。这样,我们就希望能够确定有一个期望值函数,该函数可以对状态进行评价,使得在其他条件都相同的情况下,系统能够更优先考虑那些来自于真诚反应的状态,而不是那些来自于非真诚反应的状态。我们说具有这样属性的期望值函数有一个真诚性优选。需要注意的是,真诚性已经嵌入到前面给出的“ConvinceByInform”行为的定义中。所以,如果只有满足这些约束该 agent 才执行一个动作,那么这个 agent 将一定是真诚的。

框 17.4 会话准则

一些研究人员认为语言可以通过分析推理行为和多 agent 的协作来解释,这种观点最有影响的支持者之一是 Grice(1975)。他认为,假定所有 agent 都遵循一些会话准则(除非有特别的理由不遵守),那么就可以利用这些准则来分析会话中的潜在含义。这样,这些准则对篇章理解中的一些关键性假设提供了支持。Grice 的 4 个主要准则如下:

数量准则——使会话中的信息恰如其分,而不要有过多的信息。这个准则防止说话者不说出一些关键信息(例如,与预想不同的信息);另一方面,也不要有余余的信息。

质量准则——不要说没有证据的事情。这个准则反映在 17.8 节讨论的真诚性优选中。

关系准则——说话内容要与当前的话题相关。这个准则与第 15 章讨论的连贯性假设一致。

方式准则——要避免含糊和歧义。

这些准则不是固定的规则,而是一些优选和默认的假设。说话者可能因为一些原因而不符合这些准则。例如,说话者可能想误导听者;也可能,说话者可能避而不答对方的问题,而回答另一个不同的问题,以表明你不想回答原来的问题。

由于引入了专心性约束和期望的真诚性,所以有足够的约束来确保一个简单的问答系统的行为。但是,还不能认为这个 agent 是有用的。再考虑 Jack 的柜子的例子,假定问答系统就对应于 Jack 的角色。Sue 问 Jack 是否有柜子的钥匙,Jack 检查他的信念,结果发现他没有柜子的钥匙(因为根本就不存在钥匙),所以可以真诚地给出回答“no”。如果我们知道这个背景,无意中听到这个会话,我们将认为 Jack 的回答是误导的且不合作的。问题在于给出回答“no”,Jack 隐含地表明这个问题在结构上是合理的,并进一步证实了这样的假设:可以用钥匙打开柜子。导致的结果是 Sue 相信的共享知识偏离了 Jack 所认为的情况。虽然 agent 可能想要使自己具有一定的容错能力,但是对话双方都应尽快纠正对话中出现的误解。这样,我们把共享知识优选加入到期望函数中。一个 agent 更倾向于能够使每一个 agent 之间的区别最小化的那个状态。如果其他 agent 表达了一些信息并从这一信息能够推理得到我们的 agent 所认为的一些事情是不正确的,那么我们的 agent 将选择能够解决这一冲突的状态。这样,答案“No, the locker has a combination lock”要比简单地回答“No”好。虽然这两个答案都正确,但是前者服从共享知识优选。

真诚性优选和共享知识优选几乎能够适用于任何会话场景,所以可以看做对话的基本属性。其他的优选就只适用于更有限的情况,例如,对大部分应用特别相关的一种优选是帮助性优选(helpfulness preference)。这个优选能够让其他 agent 更有效地达到他们的意图。考虑回答火车时刻的查询系统。为了能够乘火车,agent 必须在合适的时间到达合适的地点。这样,如果一个人查询到“Windsor”的火车什么时间离开,系统可能给出离开的时间和地点,例如 4 点钟,7 站台。这个答案要比简单地回答 4 点钟要好,因为这避免了“在第几站台发车”的查询。当然,如果会话 agent 认为这个人已经知道了这列火车从哪个站台发车,那么第二个答案就没

有多大意义。为了满足这个优选,agent 应该尽可能地考虑判断其他 agent 的规划,并且能够确定真正需要的信息是什么。

当然,应该限制一个回答所给出内容的多少。假设在一个特定场景中,一个 agent 识别出了其他 agent 的详细规划,能够发现其他 agent 在规划实施过程中需要的信息。很显然,提供多余的信息与提供的信息不够是一样糟糕的。这可以称为简洁性优选,如果所有其他条件都是相同的,我们肯定优选更简洁的答案。(难道你不希望他人利用这个约束?)当然,帮助性优选和简洁性优选也有可能不一致,在这种情况下,需要用其他机制对互相冲突的内容做出选择。

17.9 识别语内表现行为

如前所述,只有相关意图都确定了,语言交流才有可能。但遗憾的是,在单词集合和特定意图之间没有简单的对应关系。例如,句子“Do you know the time?”可以是一个问题、一个请求或者一个提议,这取决于说话者想要表达的内容。这样,实际语言和要交流的内容之间的联系看似还很遥远。但是,如果是这样,一个 agent 是怎样基于其他 agent 的表述来确定其要表达的意义呢?这一节我们要探讨这个问题。

解决这个问题的最常见方法是基于字面意义假设。这个方法假定所有句子都有一个完全基于语言的字面意义。例如,句子“Do you know the time?”的字面意义中这个句子是一个是否问题,问听众是否知道时间。这称为表层言语行为,根据句子的句法模式来确定。给定这个字面意义,意图的言语行为就可以从一个推理过程中推导出来,这里我们利用的是规划识别技术。

分析表层言语行为的方法有两种。第一种方法,表层言语行为是一个语内表现行为,但是在某些情形下可能是错的。例如,诚心问一个问题,那么说话者一定想知道解答。考虑一个场景,Helen 知道 Jack 没有时间意识,Helen 说“Do you know the time?”表层言语行为是一个是否问题,问 Jack 是否知道时间。事实上不是这样的,因为 Helen 并不想知道答案。意识到了表层言语行为的错误促使我们选择其他的意义。这个方法的一个问题是字面行为并不总是错误的,因为这个句子有间接的意义。例如,Helen 可能不想知道 Jack 是否知道时间。这种情况下,Helen 所说的“Do you know the time?”解释为是否问题或提醒都是可行的。

在另一种方法中,表层言语行为不是语内表现行为,而是另一个层次分析的行为,并且必须利用推理来确定意图的言语行为。这种情况下,表层言语行为是句子的语义分析的一部分。例如,“Do you know the time?”可能是查询行为,它的命题内容为“Know(Jack1, the time)”。然后,每一个语内表现行为都将指定用哪一种表层形式来实现。图 17.10 给出了语内表现行为“RequestRef”的定义,这个行为有多个组成分解步骤,每一个步骤都对应一种特殊类型的表层言语行为形式。此图还给出了交流行为“MotivateInformRef”,它是“MotivateByRequest”行为的另一种形式。

这三个分解步骤确定了常常用于表达 RequestRef 行为的三种不同的表层形式。这些形式分别对应于下面的句子:“What is the time?”“Do you know the time?”和“Tell me the time.”如果我们合理地假设这些行为的定义是不同 agent 之间的共享知识,那么一个 agent 是怎样规划合适的表达形式来实现 MotivateInformRef 行为的就很简单了。相似地,一个简单的分解链技术可以用于把特殊的表层形式作为一种交流行为。当然,每一种表层形式都可以出现在很多不同的语内表现行为中,所以仍然具有歧义问题。例如,图 17.11 为是否问题给出了行为的定义。

行为类 *MotivateInformRef(e)*:
 角色: *Speaker, Hearer, Pred_x*
 约束: *Agent(Speaker), Agent(Hearer), UnaryPred(Pred_x),*
 $\neg \text{KnowRef}(\text{Speaker}, \text{Pred}_x)$
 $\text{KnowRef}(\text{Hearer}, \text{Pred}_x)$
 前提条件: *At(Speaker, Loc(Hearer))*
 结果: *Intend(Hearer, InformRef(Hearer, Speaker, Pred_x))*
 分解: *RequestRef(Speaker, Hearer, Pred_x)*
 $\text{DecideTo}(\text{Hearer}, \text{InformRef}(\text{Hearer}, \text{Speaker}, \text{Pred}_x))$

语内表现行为 *RequestRef(e)*:
 分解 1: *WhQuestion(Speaker, Hearer, Pred_x)*
 分解 2: *Interrog(Speaker, Hearer, KnowRef(Hearer, Pred_x))*
 分解 3: *Imper(Speaker, Hearer, InformRef(Hearer, Speaker, Pred_x))*

图 17.10 wh-请求的言语行为定义

行为类 *MotivateInformIf(e)*:
 角色: *Speaker, Hearer, Prop*
 约束: *Agent(Speaker), Agent(Hearer), Proposition(Prop)*
 $\neg \text{KnowIf}(\text{Speaker}, \text{Prop})$
 $\text{KnowIf}(\text{Hearer}, \text{Prop})$
 前提条件: *At(Speaker, Loc(Hearer))*
 结果: *Intend(Hearer, InformIf(Hearer, Speaker, Prop))*
 分解: *RequestIf(Speaker, Hearer, Prop)*
 $\text{DecideTo}(\text{Hearer}, \text{InformIf}(\text{Hearer}, \text{Speaker}, \text{Prop}))$

语内表现行为 *RequestIf(e)*:
 分解 1: *Interrog(Speaker, Hearer, Prop)*
 分解 2: *Interrog(Speaker, Hearer, KnowIf(Hearer, Prop))*
 分解 3: *Imper(Speaker, Hearer, InformIf(Hearer, Speaker, Prop))*

图 17.11 是否问题的言语行为定义

问题“Do you know the time?”与 InformRef 和 InformIf 的分解步骤匹配。具体来说,由语义解释器产生的字面语义为:

Interrog(Helen1, Jack1, KnowRef(Jack1, $\lambda t . \text{NowTime}(t)$))

这将与 RequestRef 的分解步骤 2 和 RequestIf 的分解步骤 1 匹配,产生下面两个可能的解释:

RequestRef(Helen1, Jack1, $\lambda t . \text{NowTime}(t)$)

RequestIf(Helen1, Jack1, KnowRef(Jack1, $\lambda t . \text{NowTime}(t)$))

为了从这两个意义中选择一个,系统可以把每一个选择放到上下文中来检查哪一个更合适。采纳某一个意义,意味着这一意义满足所有的约束。这样,如果一个解释不满足某一个约束,这个意义就行不通。如果我们假定的共享知识为, Helen 已经知道了时间,即 $\text{KnowRef}(\text{Helen1}, \lambda t . \text{NowTime}(t))$,那么就可以删掉 RequestRef 意义,问题“Do you know the time?”由此可以解释为是否问题。而另一个方面,如果共享知识为 Helen 不知道时间,而 Jack 知道时间,那么只有 RequestRef 解释是可接受的。如果没有信息可以作为共享知识,这两个解释都是合理的。

经过基于约束的过滤以后,其他解释可以进一步用规划判别算法来确定哪一个解释与当前场景最搭配。

需要注意的是即使不考虑字面意义,而采取了间接意义,句子的原始形式也会影响意义的选择。这提示我们,在最后的分析中至少要保留字面意义。例如,句子“Do you know the time?”和“Tell me the time.”都是询问时间,分析下面的回答:

26a. Yes, it's three o'clock.

26b. No, I'm afraid not.

26c. Three o'clock.

26d. OK. It's three o'clock.

26e. No. I can't.

其中,26a,26b和26c是问题“Do you know the time?”的合适回答,26d和26e很不自然了。而另一方面,对于直接请求“Tell me the time”,回答26a和26b很不自然,其他3个就很合适。

不是所有的间接言语行为都以常规形式出现。例如,句子“It's cold in here”可以认为是一个请求,希望对方能够关上窗户,但是句子的结构中不能给出这个解释。而这个句子的解释是通过利用下面的两个通用知识得到的:(1)我们感觉有点冷,(2)是窗户打开了才导致房间中太冷的。这样的解释只有对 agent 的信念和意图进行推理才能够得到。这一系统将需要判断出这个句子的意图是为了让对方关上窗户。由于这个意图,这个表达才解释为一个请求。这个问题的细节如框 17.5 所示。

框 17.5 识别意图和定义语内表现行为

不是所有的交流都是意图识别的结果,也不是所有能够对语言交流行为的定义有帮助的意图都被贯彻了。这将使准确定义语内交流行为很难。下面的例子说明了这个问题。

假定 Jack 说话有点轻微的口吃,这样当他向 Sue 介绍自己时,她可能认为 Jack 口吃。这个事实被传递给 Sue 了,但这与 Jack 试图要说的内容没有关系,即与语内交流行为没有关系。因为 Jack 只是想告诉 Sue 他的名字,所以我们不能说 Jack 是要告诉 Sue 他有点口吃。为了完成这个语内交流行为,说话者必须倾向于完成这个行为。但是这还不够,假定 Jack 不是真的口吃,而是在向 Sue 介绍自己时故意装做口吃。在这种情况下,Jack 是想让 Sue 相信他口吃,但是我们仍然不能说 Jack 向 Sue 传递的信息是 Jack 口吃,因为 Sue 没有识别出 Jack 是有意使 Sue 相信他口吃的。但是,即使 Sue 能够判断出 Jack 的意图,这仍然不是 Inform 行为。假定一个朋友告诉 Sue 说 Jack 经常对刚遇见的人搞恶作剧,那么 Sue 就认识到 Jack 是故意装口吃以使她相信的。尽管如此,Jack 仍然没有把他口吃的信息传递给她。这个信息传递失败是因为 Jack 并不想让 Sue 识别出他的意图是装做口吃骗她相信。因此,要成功完成一个语内交流行为,说话者的意图必须要能够识别。你可能会认为这样就解决了问题,但仍然存在反例,即所有的条件都满足,但不能认为是 Inform 行为。要解决这个问题,需要的就是这些 agent 中,说话者想要被对方识别的那些意图的共享知识。Perrault 和 Allen(1980)以及 Allen(1983)介绍了用这样的准则来识别语言行为的计算模型。

17.10 篇章层次的规划

到目前为止,我们讨论的会话 agent 模型还只是试图解释单个句子的意义,没有涉及处理大规模的篇章层次的问题。这一节将在篇章层次探讨基于规划的模型。在这个模型中,句子之间的关系可以用来规划句子以及识别其他 agent 所表达的句子的结构。

很多篇章理论都利用了句子之间的有限几个关系。不同的研究者对这些关系的叫法不一,例如篇章关系、修辞关系、会话策略和篇章中的主题变化,但是,他们都把这些关系看做是在定义合理的篇章结构。因此,确定这些关系对于篇章的理解是非常重要的。关于这些外在的关系是否必要也有一些争论。例如,在篇章理论中,考虑一个外在的问题与答案的关系是否必要。给定本章描述的模型,如果 MotivateInformIf 行为执行成功,那么接收 agent 将具有意图去执行 ConvinceByInformIf 行为。由于每一个 agent 的行为定义中具有倾向性,所以,每一个问题后面都会出现一个答案。于是,我们会注意到篇章通常都组织为问题与答案对的形式。这样,即使不需要明确地识别问题与答案的关系,也可以解释这个篇章。所以,就有了争论,一些人认为存在一些用于产生篇章结构的篇章关系;而另一些人认为这些关系都是附带的,它只是一些更基本的多 agent 行为模型所产生的结果。

不管谁赢得这场争论,对任何应用来说,篇章之间的一些关系将对篇章的解释有很大的帮助。在语言生成系统中更是如此,这样的系统需要使用多个句子表达来传递大量的信息。在这些情况下,把信息组织为一些句子级的单元可以看做一个利用一些篇章关系进行规划的过程。例如,考虑这样的应用,系统提供了大百科数据库的接口,其中一个任务是为不同类的对象给出定义。一般情况下,一个定义可以用几个句子来表达。通过定义一系列篇章层次的行为,每一种行为分别采用不同的方法来定义某个类,由此,这个问题就可以作为一个规划问题来处理。例如,定义某个类的行为如图 17.12 所示。

交流行为 **DefineClass(e):**
角色: *Speaker, Hearer, Class*
分解: *IdentifySuperClass(Speaker, Hearer, Class)*
IdentifyProperties(Speaker, Hearer, Class)
GiveExample(Speaker, Hearer, Class)

交流行为 **IdentifySuperClass(Speaker, Hearer, Class):**
角色: *Speaker, Hearer, Class, SuperClass*
约束: *Agent(Speaker), Agent(Hearer), SubType(Class, SuperClass)*
分解: *ConvinceByInform(Speaker, Hearer, SubType(Class, SuperClass))*

图 17.12 用于定义一类对象的一些篇章层次的行为

DefineClass 的每一步将根据其他行为或言语行为来定义,如图中的 IdentifySuperClass。系统将利用这些定义来帮助规划一系列的定义对象的句子。给定一个合适的数据库和一个查询“*What is a dog*”,系统可能规划下面的言语行为:

```
ConvinceByInform(Speaker, Hearer, SubType(Dog, Animal))
ConvinceByInform(Speaker, Hearer, Property(Dog, CommonPet))
ConvinceByInform(Speaker, Hearer, ExampleOf(Fidol, Dog))
```

产生的结果如下:

Dogs are animals. They are common pets. For example, Fido is a dog.

(狗是动物。它们都是常见的宠物。例如, Fido 是条狗。)

即使在很有限的应用中,也需要很多不同的定义类的策略以及各种约束条件,这些约束条件用于帮助确定对某一特定的任务来说哪个策略最好。例如,对某些类来说,比较好的定义方法是与用户已经知道的另一个类进行对照和对比。

在其他应用中,大量的篇章层次的脚本将用来确定对话的结构。例如,分析一个用对话来订购物品的自动订购系统。在这种情况下,这几个阶段包括获取用户的地址和信用卡、订单和根据用户信息发送商品。这样的系统可以用上层的篇章脚本驱动,如图 17.13 所示。

篇章脚本 *TakeOrder(e)*:

角色: *System, User, Name, Address, CreditCardInfo, Items*

分解: *Greetings(System, User)*

GetAddress(System, User, Address)

GetCreditInfo(System, User, CreditCardInfo)

GetOrder(System, User, Items)

VerifyDeliveryStatus(System, User, Items)

Closings(System, User)

篇章脚本 *GetCreditInfo(e)*:

角色: *System, User, CreditCardInfo*

分解: *MotivateInformRef(System, User, λx CreditCardInfo = x)*

ConvinceByInformRef(User, System, λx CreditCardInfo = x)

VerifyNumber(System, User, CreditCardInfo)

图 17.13 自动订购系统的篇章层次脚本的一部分

当然,需要定义很多不同的行为来处理各种不同的情况。例如, *VerifyNumber* 行为将包括至少两个脚本:(1)如果输入号码有效,将是感谢信息;(2)如果输入号码无效,将进一步用对话框来再次确认这个号码,以得到另一个号码。运用脚本将允许用户快速地定义和修改 agent 的行为。只有当用户行为与脚本一致时,系统才能够继续运行。例如,如果用户先订购物品,再给信用卡,系统可能不理解这个过程(除非有另外一个脚本覆盖了这种情形)。换言之,系统必须控制对话的发展,而不存在任何主动权交替式的交互形式,因为在这种形式中,用户可以控制对话的流程。但是在一些应用中(如根据商品目录订购),这些限制是合理的。

在更一般的对话应用中,篇章层次的规划的重要性还有另外的原因。如果允许用户控制对话的流程,那么系统就必须能够确定用户正在做什么。例如,当用户不愿意继续讨论当前的话题,而要改变话题时,系统必须能够识别。当用户需要前面对话中的一个特定方面的其他细节时,系统也必须能够确定。这样的系统必须能够在篇章层次执行规划识别以便确定这些行为,并把这个层次的分析与在领域层次进行的识别联系起来。

17.11 小结

要对一个对话的参与者进行建模,我们需要定义一个会话 agent。为此,需要定义信念、期

望和意图这三个概念,并把这些概念和规划模型联系起来。交流是一个多 agent 的行为,每一个 agent 都需要认识其他 agent 的意图,特别是与交流过程相关的意图。言语行为可以定义为交流行为的一种。要确定意图的言语行为,系统必须考虑句子的结构和当前的上下文,特别是关于对话双方知道的内容和想要知道的内容的共享知识。可以对言语行为模型进行推广,以定义篇章层次的规划。在很大一类应用中,篇章层次的规划对于控制对话系统是很有用的。

17.12 相关工作与深入阅读材料

agent 的 BDI 模型的研究主要集中于规划和行为系统[如 Bratman, Israel 和 Pollack(1988)]。这一工作主要是阐述意图的本质,这一点在规划系统的以往的研究中被忽略了。Bratman(1987)认为意图是最基本的结构,不能简化为信念和期望,以未来为导向的意图对于解释理性的行为是最重要的。

对于 Believe 和 Intend 之类的运算符的形式化已有很多的研究。计算机领域中的信念的最正式模型是 Hintikka(1969)提出的,它在可能世界语义(Kripke, 1963)中定义了信念。关于知识的计算模型的优秀例子参见 Moore(1977)。关于意图和信念的形式化研究的两个很好的例子是 Cohen 和 Levesque(1990a)以及 Rao 和 Georgeff(1991)。

数据库信念模型使用了由 Moore(1973)提出并由 Cohen(1978)进一步发展起来的技术。Allen 和 Perrault(1980)对这一模型进一步改进并应用在识别言语行为的系统中。Konolige(1986)给出了这些成果的更一般的形式化方法。Levesque(1984)研究了外在信念和隐含信念的不同,他还基于弱化的一些推理规则提出了外在信念(你实际相信的命题)的逻辑。其中,隐含信念(你可以一贯相信的命题)是给定标准一阶语义条件下的外在信念的演绎闭包。

Austin(1962)介绍了言语行为的概念。在大部分情况下,他只考虑了显性施为句(explicit performative)。Searle(1975)利用了 Grice(1957)提出的交流的理论,为日常行为如“inform”(通知)、“request”(请求)和“promise”(承诺)等提出了一个言语行为理论。Grice 认为发生交流的基本需求就是说话者交流的意图和听者也意识到对方的意图。Grice(1975)还介绍了所有交流行为的一些指导原则(见框 17.4)。Schiffer(1972)和 Strawson(1964)从哲学角度详细讨论了共有信念和共享知识的必要性,并被 Perrault 和 Cohen(1981)以及 Clark 和 Marshall(1981)用于指代问题的研究。

Bruce(1975)首先提出了基于规划的言语行为模型,在此基础上,有一系列文献又进一步阐述了这个问题。其中,第一篇是 Cohen 和 Perrault(1979)。这篇文章给出了这一方法的一般性原则,并介绍了要利用标准的规划技术达到目标,应该怎样规划言语行为。但是 Cohen 和 Perrault 没有考虑怎样从言语行为中生成真实文本的细节。Appelt(1985)扩展了这一工作以生成实际的文本。他实现了一种技术,可用于把多个通知行为中的信息综合在一起,删除多余的句子,最后用一个句子来表达。他还把指代短语的生成过程刻画为一个规划过程。

Pollack(1990)强调了规划识别和意图识别的不同。基于精神状态的推论,她定义了一个意图识别的模型,虽然这里的行为模型只限于生成关系。Grosz 和 Sidner(1990)对这一工作进行了扩展,并利用共享规划的概念来解释篇章意图。

Perrault 和 Allen(1980)在 Searle(1975)的研究工作基础上,利用字面意义的规划识别提出了间接言语行为的计算模型。并且,说明了怎样把这个模型用来规划有用的应答。Cohen 和

Levesque(1990b)提出了一个理性行为的模型。其中,没有明确阐述间接言语行为,但是一般意义上的模型都描述不了这些间接言语行为的属性。Perrault(1990)描述了一个使用默认逻辑进行言语行为解释的模型。不过,在所有实际系统中,都利用间接行为的外在表示,这样才能实现有效的识别技术,如 Sidner(1985), Litman 和 Allen(1987;1990), Carberry(1990)以及 Traum 和 Hinkelman(1992)。这些系统把间接行为的一般形式描述为篇章行为的分解步骤的一部分,并使用分解链来识别间接解释。Litman 和 Allen(1987;1990)也介绍了一种规划的多层模型,此模型同时用篇章层次和领域层次的规划来解释句子。

17.13 习题

1. 【易】考虑一个场景, Jack 开车往一个方向行驶, 而 Sue 从相反的方向开过来。Jack 把手伸到窗外表示左转弯, 并且 Sue 理解这一行为。请按照 17.2 节中类似的方法分析这个交流场景, 并说明非语内表现行为、语内表现行为和言语表达效果行为分别是什么? 要使这一交流行为成功, 需要什么约定?
2. 【易】对下面的每一个句子, 请说明这个句子是描述了期望、意图、规划还是无法确定? 请说明理由。对这三种不同情况的解读, 你能否给出语言学上的测试加以区分?
 - a. Sue wants to pass with honors this year, but she also wants to leave town all through March to ski.
 - b. Sue knows how to get good grades on her exams.
 - c. Sue intends to study all night for her exam.
 - d. Jack plans to go to the concert tonight.
 - e. Jack has a plan to get into the concert tonight.
 - f. Jack knows a plan to get into the concert tonight.
3. 【易】很多的多 agent 行为动词允许一种解读, 即描述整个行为的句子可以用来描述其中某一个 agent 的行为, 如:

Jack gave Sue the money, but she wouldn't take it.

可以解释为 Jack 试图给 Sue 钱, 但 Sue 不要^①。相反, 涉及一个 agent 行为的句子就很难这样解释:

* Jack ate the pizza, but it was too hot.

可能的解释是 Jack 吃比萨(可能不小心把嘴烫着了), 但不能解释为 Jack 试图吃比萨, 但是比萨太烫^②。请用这个论据来说明由动词 inform(通知)和 warn(警告)等描述的交流性行为是多 agent 的行为。同样, 这个测试能够说明言语表达效果动词[如 convince(说服)和 scare(吓唬、吓着)]的什么属性?

① 在这种解释中, Jack 想给 Sue 钱, 但实际上并没有给; 这个句子的另一种解释是, Jack 给了 Sue 钱, 但 Sue 不想要。——译者注

② 也就是说, 这个句子只能解释为 Jack 实际上吃了比萨并被烫着了; 而不能解释为 Jack 想吃比萨, 但因为比萨太烫而没有吃。——译者注

4. 【中】利用信念空间画出下面公式的表示形式:

- a. $Bel(Jack1, Happy(Sue1))$
- b. $Bel(Jack1, Owns(Jack1, Fido1))$
- c. $Bel(Jack1, Bel(Sue1, \neg Happy(Sue1)))$
- d. $Bel(Jack1, \exists x. Bel(Sue1, Age(Sue1, x)))$
- e. $Bel(Jack1, Bel(Sue1, \exists x. Owns(Jack1, x)))$

给定这个数据库, 下面每一个查询的结果分别是什么?

- f. $Bel(Jack1, \exists x. Happy(x))$
- g. $Bel(Jack1, \exists x. Bel(Sue1, \neg Happy(x)))$
- h. $\exists x. Bel(Jack1, Bel(Sue1, Age(Sue1, x)))$
- i. $Bel(Jack1, Bel(Sue1, \exists x. Age(Sue1, x)))$
- j. $Bel(Jack1, \exists x. Bel(Sue1, Owns(Jack1, x)))$

5. 【中】考虑言语行为 *promise*, 这个交流行为能够根据信念、期望、意图来定义吗? 如果不能, 还需要什么特征? 请根据 17.5 节定义的方式来定义这个行为。要达到这个目标, 为什么会话 agent 需要对 *promise* 行为进行规划以实现其目标? 请给出一个场景, 在该场景中, 使用 *promise* 的规划有实际的用途。
6. 【中】*ConvinceByInform* 行为的定义是否允许 agent 规划撒谎的情况? 具体来说, 考虑“说话者相信这个命题”这样的约束是否应该从交流行为的定义中删除? 对于保留这个约束的情况, 听者对于说话者的信念会有怎样的信念? 如果这个约束被删除, 听者又会对说话者的信念有怎样的信念? 听者采用哪种信念更有利于对撒谎行为进行分析? 采用你所喜欢的选择, 制定一个规划来处理 Jack 对他的年龄撒谎的情形。除了给出规划外, 还要说明 Jack 的所有相关的信念; 如果撒谎成功, 给出 Sue 具有的信念。
7. 【中】对图 17.13 所示的 *TakeOrder* 行为中需要的其他行为(具体到言语行为层次)进行定义。安排一个通过电话订购物品的对话, 并说明这个对话是怎样根据对话脚本发展的。为了处理更多种类型的对话, 是否需要对定义行为分解的语言进行扩展?
8. 【难】基于 17.3 节介绍的算法, 编写一个程序, 要求能够增加和测试简单的命题信念。并跟踪习题 4 中的事实所对应的那些公式进行的增加操作和测试操作。

附录 A 逻辑和模型论语义学介绍

逻辑有很长的历史,其起源可以追溯到亚里士多德以前。不过,现代逻辑从 19 世纪后期 Frege 的工作以后才开始成形。逻辑的集合论语义学是由 Tarski(1944)发展起来的。记住很重要的一点是,一种逻辑有很多种可能的表示方法,但原理都是一样的。具体来说,在人工智能中使用的很多表示方法都等价于一阶谓词演算(FOPC),或者是其子集。A.1 节介绍了一阶谓词演算,A.2 节和 A.3 节探讨了一阶谓词演算的语义并定义了真值的表示方法。

A.1 逻辑和自然语言

逻辑是一种刻画自然语言重要属性和推理的形式化表示方法。其结果是,逻辑的很多结构特点和自然语言有相似之处。举例来说,下面两种表达方式之间有着重要的区别,一种表达方式用于确定一个实体,另一种表达方式用于断言一个实体的特征以及确定实体之间的关系。在语言中,名词短语承担第一个任务,而句子和从句承担第二个任务。在一阶谓词演算中,同样的区别也存在。项用于表示世界中的个体或实体,而命题用于对世界上的实体做出断言。

项主要有两类,即常量和函数。常量,大多数情况下接近于自然语言中的专有名词,通常用斜体字母,后接一个或多个数字来表示,如 *John1*, *John2*, *C1*, 等等。常数本身没有任何内在的意义。相反,它们的特点只与使用它们的公式相关。自然语言中的名称和一阶谓词演算中的常量之间的一个重要区别在于,自然语言是有歧义的,而一阶谓词演算中的每个符号只有一个意义。*John* 这个名字在自然语言中可以用来指称很多不同的人。在逻辑中这是不会发生的,因为逻辑中必须用不同的常量来代表每一个不同的人。

一阶谓词演算中的函数对应于表示实体的特征或者对应于表示实体间关系的名词短语,如 *John's father* 或 *the king of Prussia*。在一阶谓词演算中,它们记做一个表达式,这种表达式由一个函数名(一种新型的常量),后接圆括号内的变量表(其他的项)组成。举例来说,对应于 *John's father* 的函数是 *father(John1)*。

在一阶谓词演算中,简单命题由一个谓词名(另一种类型的变量)和圆括号内的变量表(项)组成。简单命题对应于自然语言中的简单句,其中谓词名对应于动词。例如,句子 *John likes his father* 可以对应于公式 *Likes(John1, father(John1))*。通过使用逻辑运算符,可以用简单命题构造出更复杂的命题。逻辑运算符在自然语言中也有对应物。最简单的逻辑运算符是否定(negation)运算符,记做 \neg ,对应于自然语言中常见的否定用法。句子 *John doesn't like his father* 对应于以下公式:

$$\neg \text{Likes}(\text{John1}, \text{father}(\text{John1}))$$

其他逻辑运算符对应于语言中的一些连词。例如,语言中的连词 *and*(“与”)对应于“合取”(conjunction)运算符,在一阶谓词演算中写做“&”。语言中的连词 *or*(“或”),对应于两个不同的逻辑运算符。第一个逻辑或运算符称为“析取”(disjunction),记做“ \vee ”,对应于“*John like Sue or John like Mary*”(约翰喜欢苏,或者约翰喜欢玛丽)这一类句子的解释。第二个逻辑或运算符,

称为“异或”(exclusive-or),记做“ \oplus ”,对应的解释是他只能喜欢其中一个人,而不能两个都喜欢。另外一个重要的逻辑运算符是“蕴涵”(implication)运算符,记做“ \supset ”,大致对应于语言中的连词 if...then(“如果……那么”)。例如“If John has no money, then he will not be able to buy a car.”(如果约翰没有钱,他就不可能去买车。)逻辑“等价”(equivalence)运算符,记做“ \equiv ”,对应于短语 if and only if(“当且仅当”)。例如“John will come to the party if and only if Sue comes.”(当且仅当苏来参加舞会,约翰才会来。)换句话说,要么他们都来参加舞会,要么他们都不来参加。

一阶谓词演算中的最后一个构造形式是量化的变量。在逻辑中允许出现一种新的项,称为“逻辑变量”。引入两个新的运算符来指定变量应该如何被解释。存在量词(existential quantifier),记做“ \exists ”,用于“There is a man who likes John.”(有一个人喜欢约翰)这一类句子中的变量,这个句子可以用下面的公式来表示:

$$\exists x. \text{MAN}(x) \& \text{Likes}(x, \text{John})$$

全称量词(universal quantifier),记做“ \forall ”,用于“All men like John.”(所有人都喜欢约翰)这一类句子中的量词,这个句子可以用下面的公式来表示:

$$\forall y. \text{MAN}(y) \supset \text{Likes}(y, \text{John})$$

(就是说,对于所有的对象 y ,如果 y 是一个人,那么 y 喜欢约翰。)

由于逻辑运算符可以互相嵌套,在一个公式中,运算符管辖范围的不同,会导致潜在的歧义。以下面的公式为例:

$$\neg P \& Q$$

可以理解成两个公式的合取($\neg P$ 和 Q),或者一个公式 $P \& Q$ 的否定。为了消除这种可能的歧义,一般约定,否定运算符总是取最小可能的辖域。如果需要指定更大的辖域,可以引入括号。这样,前面的公式取上面的第一种解释,而第二种解释可以表示为:

$$\neg(P \& Q)$$

前面在逻辑量词中还引入了另外一个约定的表示方法。公式中的一个点号(.)用于说明点号前面的运算符的辖域延伸到整个公式的结尾。这是消除深层括号嵌套的一种方便的表示方法。考虑下面的公式,对应于句子“Every boy likes some girl.”(每个男孩都喜欢某个女孩)的一种解释:

$$\forall b. \text{BOY}(b) \supset \exists g. \text{GIRL}(g) \& \text{Likes}(b, g)$$

如果不使用点号约定,这个公式就要写成以下形式:

$$\forall b (\text{BOY}(b) \supset \exists g (\text{GIRL}(g) \& \text{Likes}(b, g)))$$

图 A.1 总结了这些表示方法。

A.1.1 推理:作为思维模型的逻辑

研制一阶谓词演算的一个主要动机是它可以为推理和有效论断提供一种精确的公式化表示形式。例如,你会将下面给出的句子看成是可接受的推理序列:

所有狗都喜欢骨头。Fido 是一条狗。所以 Fido 一定喜欢骨头。

另一方面,你可能会发现下面的论断不正确:

一些狗有跳蚤而 Fido 是一条狗。所以 Fido 一定有跳蚤。

可接受的推理序列表示为一种抽象的形式,这种形式称为推理规则。例如,上面的第一个论断可以在一阶谓词演算中用下面的规则来表示,其中 D 是一个只对狗为真的谓词,而 LB 是一个只对喜欢骨头的对象为真的谓词:

$\forall x. D(x) \supset LB(x)$
 $D(FIDO1)$
 $LB(FIDO1)$

这可以解读为如果公式 $\forall x. D(x) \supset LB(x)$ 和公式 $D(FIDO1)$ 都为真,那么公式 $LB(FIDO1)$ 也必须为真。采取可计算性的术语,我们还可以解读为要证明 $LB(FIDO1)$, 需要先证明 $D(FIDO1)$ 和 $\forall x. D(x) \supset LB(x)$ 。

短语	一阶谓词演算的等价形式
John(约翰)	$John1$
John's father, the father of John(约翰的父亲)	$father(John1)$
John likes his father(约翰喜欢他父亲)	$Likes(John1, father(John1))$
John doesn't like his father(约翰不喜欢他父亲)	$\neg Likes(John1, father(John1))$
Either Sue is happy, or she is rich(or both)	$Happy(Sue1) \vee Rich(Sue1)$
(要么苏是幸福的,要么苏是富裕的,或两者都是)	
Either Sue is rich, or she is poor	$Rich(Sue1) \oplus Poor(Sue1)$
(要么苏是富裕的,要么苏是贫穷的)	
Sue is happy and rich(苏既幸福又富裕)	$Happy(Sue1) \& Rich(Sue1)$
If Sue is rich, then she is happy	$Rich(Sue1) \supset Happy(Sue1)$
(如果苏是富裕的,那么她就是幸福的)	
John is rich if and only if Sue is	$Rich(John1) \equiv Rich(Sue1)$
(当且仅当苏是富裕的,约翰才是富裕的)	
There is a fish in the pond(池塘中有一条鱼)	$\exists f. Fish(f) \& In(f, Pond44)$
All fish are in the pond(所有鱼都在池塘中)	$\forall f. Fish(f) \supset In(f, Pond44)$

图 A.1 一阶谓词演算和某些自然语言短语的对应关系

实际上,推理规则并不针对特定的谓词名称。推理规则使用以下模式表示: p, q, r, s 形式的参数表示任意的命题, a, b, c 形式的参数表示任意的项, x, y, z 形式的参数表示任意的逻辑变量。如果使用下标,如 p_x , 那么这个参数表示任何一个涉及项 x 的命题。前面的论证中使用了一阶谓词演算中的两条推理规则,如图 A.2 所示。

蕴涵消去规则 (或假言推理)	全称消去规则
$p \supset q$ <u>p</u> q	$\forall x. px$ <u> </u> pa

图 A.2 两个推理规则

多条推理规则可以结合起来构造一个证明并推导出相应的结论(注意,不是单数)。一个证明由前提集合和公式序列组成,每一步推导都要使用一些推导规则,由以前的公式和前提派生而来。举例来说,前面关于 Fido 喜欢骨头的论证现在可以通过一个证明进行形式化的验证,如图 A.3 所示。

步骤	公式	理由
1.	$\forall x. D(x) \supset LB(x)$	前提
2.	$D(FIDO1)$	前提
3.	$D(FIDO1) \supset LB(FIDO1)$	根据第 1 步利用全称消去规则 用 FIDO1 取代 x
4.	$LB(FIDO1)$	根据第 2 步、第 3 步利用蕴涵消去规则

图 A.3 “Fido likes bones”的证明过程

一般地,在逻辑中对于每一个逻辑运算符都定义了两条推理规则——消去规则和引入规则。消去规则使用一个包含某个运算符的公式作为前提但是结论不包含这个运算符,而引入规则使用一个或多个不包含某个运算符的公式作为前提但是结论中包含这个运算符。到目前为止,这些表示方法已经可以表示很多的规则,不过,还有一些规则,要求进行一些扩展。典型的情形是,先进行某种假设,然后可以证明某些其他的公式,或者可以证明出矛盾。在某个论据条件下,如果存在一个公式 P 能够使 P 和 $\neg P$ 同时成立,则该论据引起矛盾。例如,否定引入规则,可以直观地表示如下:为了证明 $\neg p$,先假设 p 是真的,然后说明这会导致一个矛盾。假设你被告知杰克不喜欢骨头,而你需要证明杰克不是一条狗,并且假定所有的狗都喜欢骨头。你可以先假定杰克是一只狗,然后说明这会推导出杰克喜欢骨头,而这跟你的初始假设互相矛盾。这个证明如图 A.4 所示。其中,基于假设所做出的子证明以缩进形式显示,以便将其从主证明中区分开来。在第 2.1 步中做出假设,而在第 2.3 步和第 2.4 步中导出了一个矛盾。第 2.4 步简单地复制了第一步的公式,使用了一条称为重复的引用规则,这条规则允许我们在一个证明中将前面已经推导出的公式在后面进行复制引用。虽然严格说起来这并不是必需的,不过这会使得证明更加直观和清晰。在这个例子中这样做使我们可以将第 2.3 步和第 2.4 步放在一起以指出其中的矛盾。

步骤	公式	理由
1.	$\neg LB(JACK1)$	前提
2.	$\forall x. D(x) \supset LB(x)$	前提
2.1.	$D(JACK1)$	子证明的假设
2.2.	$D(JACK1) \supset LB(JACK1)$	根据第 2 步利用全称消去规则
2.3.	$LB(JACK1)$	根据第 2.1 步,第 2.2 步利用假言推理
2.4.	$\neg LB(JACK1)$	重复第 1 步,揭示出一个矛盾
3.	$\neg D(JACK1)$	根据第 2.1 步,第 2.3 步和第 2.4 步利用否定引入规则

图 A.4 通过矛盾导出一个否定结论的证明

在形式化地引入蕴涵运算符时,我们需要用到基于假设的另一条类似的推理规则,非正式地说明如下:假设 p ,如果你可以在子证明中推导出 q ,就可以得出 $p \supset q$ 这个结论。类似的情况对于引入全称量词的规则也是必需的。图 A.5 中给出了其他一些推理规则。

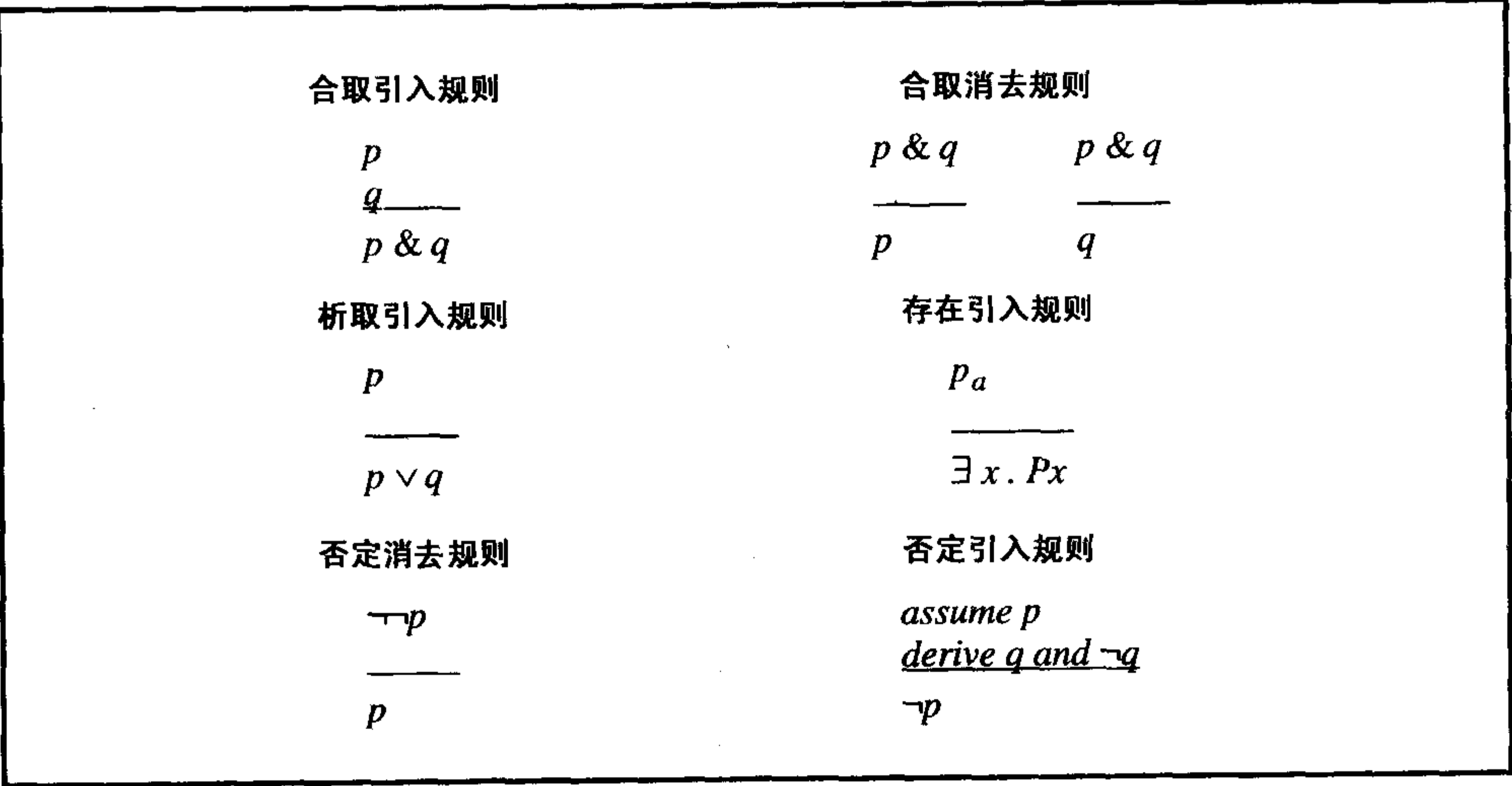


图 A.5 其他一些推理规则

给定一个推理规则的集合,不需要任何前提,就可以证明出一些公式。这些公式被称为重言式,或者更简单地,称为定理,它们反映了这个逻辑内在的特点。给定推理规则系统,一个基本的定理就是排中律。这条定理规定,一个命题要么是真的,要么是假的(不能是“不确定的”)。这条定律说明任何以下形式的公式:

$p \vee \neg p$

一定为真。很重要的一点是,我们要认识到,这条定律并没有说一个给定的推理系统能够证明 p 或者证明 $\neg p$;它所说的全部就是,事实上, p 只能是真的或者是假的。

两个重要的定理称为德摩根律:

- 1. $\neg(p \& q) \equiv \neg p \vee \neg q$
- 2. $\neg(p \vee q) \equiv \neg p \& \neg q$

使用定理 1,你可以看到排中律, $p \vee \neg p$,等价于 $\neg(\neg p \& p)$ 。这就是说,任何一个命题都不能同时既是真的又是假的。

合法公式的句法和推理规则的集合定义了一个丰富的框架,在这个框架下我们可以证明很多东西。不过,就当前的目的而言这并不是逻辑最重要的特点。在这些应用中,逻辑最重要的特点是,你可以独立地给一个概念指定真值并且给任意一个公式赋予“意义”。这就是逻辑语义学所承担的角色,也是本附录下面将要讨论的话题。

A.2 模型论语义学

逻辑的形式语义学的目标是能够回答以下这样一些问题:是否存在一个真值的定义,这个定义不依赖于可以从公理集合中推导出的结论? 如何判断一个系统是否包含了足够的推理规

则? 如何判断一个假设的推理规则集合是合理的——也就是说, 这些规则不可能用于“证明”那些假的事物? 如何判断一个公理的集合是相容的, 并且能够描述真实状况? 要回答这些问题, 我们需要对真实状况的表示方式进行形式化, 并给出逻辑中的公式如何映射到关于这种状况的断言。这种状况称为逻辑的模型, 而这里所描述的语义研究方法被称为模型论。

首先, 考虑一阶谓词演算的一个子集, 它仅仅包含命题和逻辑连接符。也就是说, 其中没有项, 没有变量, 也没有量词。这个子集通常称为命题演算。一个命题演算的模型要将每一个命题公式映射到值 T(表示真) 和 F(表示假)。要构造这样一个模型, 需要指定所有原子命题的映射, 然后使用逻辑运算符的定义推导出复合公式的值。也就是说, 逻辑运算符被映射到从多个真值到单个真值的函数。同样, 这种方法也具体说明了这样一个假定: 逻辑运算符的含义可以仅仅根据其参数的真值来定义, 而与这些参数的实际命题无关。举例来说, 假设命题 P 和 Q 都是真的(都映射到 T)。这样, 对某些命题 R 来说, 如果 $P \& R$ 是真的, 那么 $Q \& R$ 必定也是真的。无论是 P 还是 Q 是其第一个参数, 运算符 $\&$ 都不受影响, 因为它们都有相同的真值。

为了定义一个模型, 需要定义一个函数 V , 它称为值函数, 值函数将公式映射到真值 T 或者 F。给定一个逻辑, 只包含一个命题 P , 因此只能有两种可能的不同的值函数, V_1 和 V_2 , 其中 $V_1(P) = T$, 而 $V_2(P) = F$ 。给定一个逻辑, 这个逻辑只有两个命题 P 和 Q , 所以只有四种可能的不同的值函数。一般地, 给定 n 个命题, 就有 2^n 种可能的值函数。每一个值函数定义了该逻辑的一个可能的模型。

每一个值函数都具有以下特征, 这些特征递归定义了逻辑运算符的含义:

- V.1. $V_m(p \& q) = T$ if $V_m(p) = T$ and $V_m(q) = T$, and is F otherwise.
- V.2. $V_m(\neg p) = T$ if $V_m(p) = F$, and is F otherwise.
- V.3. $V_m(p \vee q) = T$ if $V_m(p) = T$ or if $V_m(q) = T$, and is F otherwise.
- V.4. $V_m(p \supset q) = T$ if $V_m(p) = F$ or if $V_m(q) = T$, and is F otherwise.
- V.5. $V_m(p \equiv q) = T$ if $V_m(p) = V_m(q)$, and is F otherwise.

最初, 我们用自然语言类比的方法给出了这些运算符的定义, 你可能会觉得定义 V.1 到定义 V.3 跟你的初始直觉比较相符。举例来说, 一个合取 $p \& q$ 只有当 p 和 q 都为真时才为真, 这与规则 V.1 完全一致。不过, 对于 V.4 中关于蕴涵的定义, 我们还需要做一些解释。考虑句子 “If John was in the room, then he saw the murder.” (如果约翰在房间里, 那么他看见了凶手。) 如果事实上约翰并不在房间里, 那么这个句子的真值是什么呢? 更一般地说, 公式 $p \supset q$ 的真值在 p 为假的时候是什么? 这里有几个不同的选择, 其中每一种选择都定义了一个不同的运算符。不过按照通常的理解, 以及在为蕴涵构造推理规则时的假定, 都是认为在这种情况下, $p \supset q$ 为真。这也就是 V.4 所表达的含义。

给定一个模型(即一个值函数), 你可以决定这个命题演算的任何一个公式在这个模型下的真值。考虑一个逻辑, 它只有两个命题 P 和 Q 。准确地说, 它有四个可能的模型, 每一个模型给出一种将 T 和 F 赋值给每个命题的可能的方式。这些模型在图 A.6 给出的表中做了一个总结, 其中每一行的真值表示一个模型。表中还给出了简单应用一个逻辑连接符所得到的一些公式的真值。所有这些真值都可以通过规则 V.1 到规则 V.5 导出。分析该表, 你会看到一种说明两个公式等价的新方法。除了构造一个公式 $P \supset Q$ 等价于公式 $\neg P \vee Q$ 的证明以外, 你还可以表明, 这两个公式在所有可能的模型中具有相同的真值。考虑基于图 A.6 所示的模型 2 来确定这两个公式的真值。使用规则 V.4, 你以看到 $P \supset Q$ 的真值是 F。使用公式 V.2, 你可

以看到 $\neg P$ 的值是 F,再通过公式 V.3 可以得到 $\neg P \vee Q$ 的真值是 F。这样,在模型 2 中,这两个公式就具有相同的真值。类似地,你会看到它们在所有其他模型中都具有相同的真值。不管值函数是什么,这两个公式都具有相同的真值。所以这两个公式是逻辑等价的。

这样,你现在有了两种独立的关于逻辑运算符的定义,一种使用推理规则,另一种使用值函数。你是否可以证明对所有的公式来说这两种方法都是一致的? 如果是的话,在通向回答本节开头提出那些问题的路上,你已经前进了一大步。

A.2.1 逻辑和语义之间的关系

我们先来检查推理规则是否可靠——也就是说,它们是否永远都不会用于证明一个公式,除非该公式在所有的模型中都为真。举例来说,考虑假言推理规则(蕴涵消除规则):

$$\frac{p \supset q \quad p}{q}$$

虽然推理规则中的参数可以取遍所有的命题,你还是可以仅考虑只有两个命题 P 和 Q 的简单情形。无论如何,你已经考虑了所有可能的真值组合。为了表明这条推理规则是完备的,需要表明,在任何可能的模型中,只要前提为真,结论也一定成立。在这个例子中,我们分别用 P 和 Q 来代替 p 和 q ,你会看到在图 A.6 中,只有模型 1 中 $P \supset Q$ 和 P 都是真,而此时 Q 也被赋值为 T。惟一的一个前提为真的模型使得结论也同时为真。因此,这条规则是可靠的,至少在只有两个命题的逻辑中是这样。由于其他的命题对这里的参数不产生任何影响,因此这条规则对给定的任何数量的命题都是可靠的。

模型	P	Q	$\neg P$	$P \& Q$	$P \vee Q$	$P \supset Q$	$P \equiv Q$	$\neg P \vee Q$
1.	T	T	F	T	T	T	T	T
2.	T	F	F	F	T	F	F	F
3.	F	T	T	F	T	T	F	T
4.	F	F	T	F	F	T	T	T

图 A.6 P 和 Q 的所有可能的模型

前面我们提出的另外一个问题现在也可以解决了。要说明给定的一个公式集合是相容的,也就是说,可以描述某种实际情况,你就可以找出一个模型,这个模型给这个公式集合中的所有公式都赋值为 T。这样,一个模型可以称为满足了这个公式集合。例如,你可以看到下面这个公式集合:

$$\{\neg P, P \vee Q, Q\}$$

是相容的,因为图 A.6 中的模型 3 给其中每个公式都赋值为 T。相反地,你可以看到下面的公式集合:

$$\{P \supset (Q \& R), P, \neg Q\}$$

是不相容的,因为不存在任何一个模型同时给其中的所有公式都赋值为 T。这个例子的完整的真值表见图 A.7。你可以看到,没有一个模型同时给所有这三个公式都赋值为 T。

模型	P	Q	R	$\neg Q$	$Q \& R$	$P \supset (Q \& R)$
1.	T	T	T	F	T	T
2.	T	T	F	F	F	F
3.	T	F	T	T	F	F
4.	T	F	F	T	F	F
5.	F	T	T	F	T	T
6.	F	T	F	F	F	T
7.	F	F	T	T	F	T
8.	F	F	F	T	F	T

图 A.7 P, Q 和 R 所有可能的模型

我们现在可以给出永真式这个概念了：

一个公式 f 是一个永真式，如果所有模型都给它赋值为 T。

举一个简单的例子，考虑公式 $P \vee \neg P$ 。因为它只涉及到一个命题，所以你只需要考虑两个可能的模型：一个给 P 赋值为 T，另一个给 P 赋值为 F。在这两个模型中，公式 $P \vee \neg P$ 都赋值为 T：如果 P 被赋值为 T，那么通过规则 V.3， $P \vee \neg P$ 被赋值为 T；相反，如果 P 被赋值为 F，那么 $\neg P$ 就被赋值为 T，因而通过规则 V.3， $P \vee \neg P$ 被赋值为 T。所以 $P \vee \neg P$ 是永真式。

在两个公式之间可以定义两种有用的关系。概念蕴涵定义如下：

s 蕴涵 p (记做 $s \models p$)，当且仅当在所有满足 s 的模型中 p 被赋值为 T

概念可证明定义如下：

p 可以从 s 中证明 (记做 $s \vdash p$)，当且仅当存在一个 p 的证明，其中 s 是惟一的前提

要考虑的最后一个问题是，你是否可以证明一个给定的推理规则集合是足够的，即该推理规则集合允许所有为真的公式都能被证明。换句话说，如果通过使用模型论，你发现某个公式 p 在所有模型下都为真，那么你希望能够表明 p 通过这个逻辑中定义的推理规则也是可证明的。使用前面定义的蕴涵概念，这也可以理解为一旦 s 蕴涵 p (即 $s \models p$)，那么 p 能够被 s 证明 (即 $s \vdash p$)。这样一个证明系统被称为是完备的。很多不同的推理规则集合都可以具备完备性。对于可计算的方法来说，最重要的是分解方法给定了一个单一的推理规则，而这个推理规则可以证明是完备的。也就是说，任何一个为真的公式都可以通过重复使用分解规则得到证明。

A.3 一种一阶谓词演算的语义学：集合论模型

一阶谓词演算的语义学要求对模型的概念进行扩充，最方便的是用集合论的术语来表达。问题的复杂性来自于命题以外的表达式意义的定义。一阶谓词演算中的项并不表达真值，而是表达物理的实体、事件、时间、处所，等等。所有这些对象都被认为是称为论域的对象集合的元素。一般地，每一个模型都可以有不同的论域。不过，为了简化表达，这里我们假设所有的模型都具有相同的论域，记做 Σ 。

在命题演算中,值函数将命题符号映射到 T 或者 F。在一阶谓词演算中,命题本身具有结构,而值函数必须对常数、函数和谓词名的解释进行定义。特别地,值函数把每一个项都映射到论域中的一些元素。考虑一个论域 Σ ,它由三个元素 α, σ 和 δ 组成,在一个模型中,值函数 V_1 对三个项 AI 、 BI 和 CI 的定义如下:

$$V_1(AI) = \alpha; V_1(BI) = \sigma; \text{ 且 } V_1(CI) = \delta$$

根据谓词变元数量的不同,值函数还把谓词名映射到不同类型的集合中。我们这里暂时只考虑一元谓词,比如 RED (红),这个谓词只映射到 Σ 的一个子集(也就是被指定为红色的那些物体)。举例来说,我们假设模型将谓词 RED 定义为集合 $\{\alpha, \sigma\}$:

$$V_1(RED) = \{\alpha, \sigma\}$$

建立在一元谓词名和一个变元的基础上,简单命题的真值可以根据以下规则定义如下(其中, P 是一个任意的一元谓词名,而 a 是任意一个项):

$$V_m(P(a)) = T \text{ 如果 } V_m(a) \text{ 是 } V_m(P) \text{ 的一个元素,否则为 } F$$

给定 V_1 的定义和由一元谓词名构造的命题真值的定义,很容易看到:

$$V_1(P(AI)) = T; V_1(P(BI)) = T; \text{ 且 } V_1(P(CI)) = F$$

就像在命题逻辑中一样,你可以为所有可能的逻辑构造真值表。图 A.8 显示了在假定论域 E 上,一个只有三个常数一个谓词名 RED 的逻辑的一些可能的模型。就像你所看到的,对于谓词演算来说,真值表方法是很不实用的,因为有太多可能的模型。在上面的例子中一个完整的表将有 216 行。

模型	AI	BI	CI	RED	$RED(AI)$	$RED(BI)$	$RED(CI)$
1.	α	σ	δ	$\{\alpha\}$	T	F	F
2.	α	α	σ	$\{\alpha\}$	T	T	F
3.	α	σ	δ	$\{\alpha\}$	T	F	F
4.	α	α	α	$\{\alpha, \delta\}$	T	T	T
5.	α	α	σ	$\{\alpha, \delta\}$	T	T	F
6.	α	α	α	$\{\alpha, \delta\}$	T	T	T
7.	α	α	α	$\{\delta, \sigma\}$	F	F	F
8.	α	α	σ	$\{\delta, \sigma\}$	F	F	F
9.	α	σ	δ	$\{\delta, \sigma\}$	F	F	F

图 A.8 假定的论域 $\Sigma = \{\alpha, \delta, \sigma\}$ 上面一些可能的模型(歧义,建议拆成两句)

值函数将一个具有 n 个变元的谓词名 P 映射到一个长度为 n 的元素表的集合。举例来说,二元谓词 $Loves$ (爱)在 V_1 中可以定义如下(其中 α 爱 δ , 而 σ 爱 δ):

$$V_1(Loves) = \{(\alpha \delta), (\sigma \delta)\}$$

由 n 元谓词构造的简单命题的真值定义规则定义如下(其中 P 是任何一个 n 个变元的谓词名,而 a_1, \dots, a_n 是任意项):

$V(P(a_1, \dots, a_n)) = T$ 如果 $(V(a_1) \dots V(a_n))$ 是 $V(P)$ 的成员, 否则为 F

现在, 我们可以决定模型 1 中公式的真值, 举例来说:

$$V_1(\text{Loves}(Al, Bl)) = F; V_1(\text{Loves}(Bl, Al)) = T; V_1(\text{Loves}(Al, Cl)) = T$$

一旦为命题定义了不同的值函数, 定义逻辑运算符的规则就和命题演算中的定义完全一样了。所以我们可以计算 V_1 对于诸如 $\neg(\text{Red}(Al) \vee \text{Loves}(Al, Bl))$ 这样的公式所赋予的值, 如图 A.9 所示。

表达式	$V_1(e)$
<i>Al</i>	α
<i>Bl</i>	σ
<i>Red</i>	$\{\alpha\}$
<i>Loves</i>	$\{(\alpha\delta), (\sigma\alpha)\}$
<i>Red</i> (<i>Al</i>)	T
<i>Loves</i> (<i>Al</i> , <i>Bl</i>)	F
<i>Red</i> (<i>Al</i>) \vee <i>Loves</i> (<i>Al</i> , <i>Bl</i>)	T
$\neg(\text{Red}(Al) \vee \text{Loves}(Al, Bl))$	F

图 A.9 计算模型 1 中 $\neg(\text{Red}(Al) \vee \text{Loves}(Al, Bl))$ 的值

A.3.1 量词的语义

要定义带量化变元的公式的语义, 我们必须能在公式中替换变元。为做到这一点, 需要采用变量替换的方法扩充值函数的概念。我们定义 $V_{1\{x/\alpha\}}$ 为一个扩充的值函数, 除了其中将变量 x 的值定义为论域常量 α 以外, 这个值函数的其他方面跟 V_1 完全相同。例如, 给定前面定义的值函数 V_1 :

$$\begin{aligned} V_{1\{x/\alpha\}}(x) &= \alpha \\ V_{1\{x/\alpha\}}(\text{Red}) &= V_1(\text{Red}) = \{\alpha\} \end{aligned}$$

这样, $V_{1\{x/\alpha\}}(\text{Red}(x)) = T$, 因为 $V_{1\{x/\alpha\}}(x)$ 是 $V_{1\{x/\alpha\}}(\text{Red})$ 的一个成员。

有了变量替换的概念, 含有量词的公式的语义可以用以下值函数规则来定义:

$$\begin{aligned} V_m(\forall x. P) &= T \text{ 如果对于 } \Sigma \text{ 中的每一个元素 } \alpha, V_{m\{x/\alpha\}}(P) = T \\ V_m(\exists x. P) &= T \text{ 如果对于 } \Sigma \text{ 中至少存在一个元素 } \alpha, \text{ 使得 } V_{m\{x/\alpha\}}(P) = T \end{aligned}$$

直观地, 这就是说一个形如 $\forall x. P$ 的公式为真, 当且仅当公式 P 在变量 x 取任何值的时候都为真; 而一个形如 $\exists x. P$ 的公式为真, 当且仅当至少存在 x 的一个值使得 P 为真。

所有前面关于可靠性、相容性、完备性等的定义可以不加修改地全部搬到一阶谓词演算中。举例来说, 要显示一个公式的集合是相容的, 可以通过构造一个模型, 使得其中的每一个公式都为真。例如, 考虑以下公式的集合:

$$\{\forall x. Q(x) \supset R(x), Q(Al), Q(Bl), \neg Q(Cl)\}$$

可以为这些公式构造模型如下: 取 Σ 为集合 $\{\sigma, \delta, \phi\}$, 其中 $V(Al) = \sigma$, $V(Bl) = \delta$, 而 $V(Cl) = \phi$ 。取 $V(Q) = \{\sigma, \delta\}$ 且 $V(R) = \{\sigma, \delta, \phi\}$ 。现在, 你可以看到给定模型前面的每一个公式都取值为 T。

A.4 参考资料

McGawley(1993)是关于逻辑及其与语言学关系的一本好书。Partee 等(1990)提供了关于逻辑以及在语言学中其他形式模型的应用的资料,十分全面。很多人工智能的介绍性书籍也讨论了逻辑。其中一些很好的例子是 Luger 和 Stubblefield(1993),Winston(1992),Rich 和 Knight(1992)以及 Genesereth 和 Nilsson(1987)。有很多关于逻辑的介绍性书籍,但是其中很多都把重点放在证明理论上而没有探讨语义。Barwise 和 Etchemendy(1987)给出了一个极好的介绍,并且完成了一个交互式的程序,可以让你探索逻辑中的问题。

A.5 习题

1. 【易】假设 $\text{FatherOf}(\text{Bill}, \text{Sam})$ 表示 Sam 是 Bill 的父亲,而 $\text{Ancestor}(\text{Bill}, \text{Harry})$ 表示 Harry 是 Bill 的祖先之一。定义其他必须的谓词,并写出一个公式表示以下句子的意义:Bill 的每一个祖先要么是他的父亲或母亲,要么是他们的祖先。
2. 【易】给出两个看上去用谓词演算表示起来很困难的英语句子,并讨论你这样认为的理由。
3. 【易】给出本附录中定义的异或运算符的值函数的定义,然后说明以下推理规则是正确的:

$$\begin{array}{cc} p \oplus q & p \oplus q \\ p \text{ — } & \neg p \text{ — } \\ \neg q & q \end{array}$$

4. 【易】对于以下每个公式集合,通过构造一个真值表来决定其中的公式是否相容。

$$S1 = \{P, \neg P\}$$

$$S2 = \{P \supset Q, \neg P, \neg Q\}$$

$$S3 = \{P \supset Q, Q \supset R, \neg P, R, \neg Q\}$$

$$S4 = \{P \vee Q, \neg P \& \neg Q\}$$

5. 【中】用一阶谓词演算来表述以下事实,并对结论给出一个证明。如果有必要,可以加入一些常识公理,以便使结论遵守给定的假设。
 - a. 假设:明天要么暖和并且天晴,要么阴冷并且下雨。明天将会天晴。
结论:明天天气暖和
 - b. 假设:Tweety 是一只鸟。Tweety 吃所有的好食物。向日葵种子是好食物。
结论:有一种鸟吃向日葵种子。

附录 B 符号计算

本附录介绍了符号计算的一些基本技术。本附录内容假设你对某种编程语言比较熟悉,但你并不需要了解符号数据的处理。如果你对 LISP 或者 PROLOG 这样的语言比较熟悉,就可以跳过 B.1 节而不影响阅读。更进一步,如果你具有人工智能的某些背景知识并且知道合一算法,则可以跳过整个附录。B.2 节探讨匹配与合一。B.3 节描述了搜索技术的基本思想,而 B.4 节在对 PROLOG 的讨论中结合了匹配和搜索的技术。最后,B.5 节详细给出了合一的算法。

B.1 符号数据结构

本文采用很多不同的数据结构来表示一些不同形式的分析结果。所有这些结构都构建在一种简单的数据结构基础上。这种结构称为列表,程序设计语言 LISP 就是基于列表的。一个简单列表由称为原子的表达式序列构成,而原子就是一个简单的字符序列。例如:

(A Happy TOAD)

是由三个原子——A, Happy 和 TOAD 构成的列表。原子的惟一特点是可以根据其名称加以区分。这样,我们就可以知道原子 Happy 和原子 TOAD 是不同的原子,因为它们由不同的字符构成。原子不一定是英语单词,例如,下面是由五个原子组成的有效列表:

(R3 forty XX3Y7 AAA1 ?X3)

通常,一个列表不仅可以包含原子,也可以包含其他列表。下面是一个有效的列表,由四个元素组成:原子 NP、列表(DET the)、列表(ADJ happy)、列表(HEAD toad):

(NP (DET the) (ADJ happy) (HEAD toad))

以后,你会看到很多与之类似的结构。现在这个列表就是英语名词短语“the happy toad”的一种句法表示。

一个列表在另一个列表中嵌套的次数没有任何限制,所以

((A TWO) (B THREE)) ((D)) E)

是一个有效的列表。把这个列表分开,可以看到它是由两个元素组成的列表,包括列表((A TWO)(B THREE))和列表(((D)) E)。其中,第二个元素本身又是由两个元素构成的列表,包括列表((D))和原子 E。元素((D))也是一个列表,它由一个元素组成,也就是由一个原子 D 构成的列表(D)。

包含一个原子的列表,比如(D),跟原子本身有很大的不同,比如 D。它们在一个程序中可以有完全不同的解释。

不包含任何元素的列表记做(),它也是有效的列表,并且可以作为其他列表的元素。这样的列表称为空列表或空表,通常写做 NIL 或者 nil。

有两个基本的操作用来将一个列表分开,它们是 First 和 Rest。图 B.1 给出了这两个函数作用于各种列表的结果。

First(< list >)——取任何一个列表,返回它的第一个元素。这个函数对于空表和原子没有定义。

Rest(< list >)——取任何一个列表,并产生一个包含除了其中第一个元素以外所有元素的列表。它对于原子和空表也没有定义。

First((A B C))	等于	A
First(((A B) C))	等于	(A B)
First((((A B) C)))	等于	(A B C)
Rest((A B C))	等于	(B C)
Rest(((A B) C))	等于	(C)
Rest((((A B) C)))	等于	(), 空表

图 B.1 列表运算符的例子

有两个基本的函数用于构造列表,它们是 **Insert** 和 **Append**。图 B.2 给出了在不同情况下这两个函数的用法。

Insert(< atom > , < list >)——返回一个新列表,其第一个元素是 < atom > ,后面是 < list > 中的元素。

Append(< list1 > , < list2 >)——返回一个新列表,它由 < list2 > 中的元素后面加上 < list1 > 中的元素组成。

Insert(A, (B C))	等于	(A B C)
Insert((A B), (C))	等于	((A B) C)
Insert((A B C), ())	等于	((A B C))
Append((A), (B C))	等于	(B C A)
Append(((A B)), (C))	等于	(C (A B))
Append((A B C), ())	等于	(A B C)

图 B.2 列表构造函数的例子

最后,有一些用于检查列表的测试谓词。根据这些测试谓词的参数结构,返回 TRUE 或者 FALSE。这些函数可用于 if-then-else 类型的语句。

Null(< list >)——只有当 < list > 是空列表的时候返回 TRUE。

Member(< atom > , < list >)——只有当 < atom > 是 < list > 的一个元素的时候返回 TRUE。

一个原子在列表中允许多次出现,这种情况下谓词 **Member** 返回 TRUE。**Member** 仅当这个原子是列表的一个元素的时候返回成功,如果这个原子出现在其嵌套子列表中,返回不成功。图 B.3 给出了这样的例子。

虽然本书在没有任何 LISP 语言知识的情况下也能看懂,不过图 B.4 还是为有兴趣的读者给出了这些函数和谓词的 LISP 对等形式。有关 LISP 语言的更多细节可以参考一些教材,如 Wilensky(1986),Winston 和 Horn(1989),等等。

Member(A, (A B C))	是	TRUE
Member(A, ((A) B C))	是	FALSE
Member((A), (A B C))	是	FALSE
Member((A), ((A) B C))	是	TRUE
Member(B, (A B C))	是	TRUE
Member(B, (A B C B))	是	TRUE
Member(B, ((A B) C))	是	FALSE

图 B.3 谓词 Member 的例子

操作	LISP 对等形式
First((A B C))	(CAR '(A B C))
Rest((A B C))	(CDR '(A B C))
Insert(A, (B C))	(CONS 'A '(B C))
Append((A), (B C))	(APPEND '(B C) '(A))
Null((A))	(NULL '(A))
Member(A, (A B C))	(MEMBER 'A '(A B C))

图 B.4 LISP 对等形式

B.1.1 栈和队列

本文给出的算法中会经常用到称为栈和队列的数据结构。它们都是由列表结构构成的，并且都非常重要，有必要单独加以考虑。栈就是一个列表，其中所有对元素的添加和删除操作都在列表的一端进行，列表的这一端我们称为栈顶。这很类似于餐厅中一堆盘子的处理：收回盘子的时候，总是放到顶部；而取盘子的时候，也是从顶部先取。因此，在任何情况下，刚刚取走的那个盘子总是最后一个加进来的盘子。由于这个原因，栈也通常被称为是一个后进先出 (LIFO) 列表。

在算法中，栈可以用于保存程序需要考虑的一些跟踪数据。这种程序的一般性组织形式如下：

- 1. 用一个或多个项对栈进行初始化；
- 2. 重复以下过程直到找到答案为止：
 - 2.1 移出栈顶部的项；
 - 2.2 分析这个项；
 - 2.3 将任何需要分析的新项加到栈中。

通常，向栈中加入一个项的操作称为向栈顶压入一个项，而移出一个项称为弹出栈。

栈组织的一个更具体的例子是办公室中接收信件的公文托盘。办公室里的人可以按下面这种方式操作：

- 1. 头天晚上的信件都被放入托盘中；
- 2. 重复以下操作直到没有信件需要处理：
 - 2.1 取出顶部的信件；

- 2.2 读这封信件并且回复;
- 2.3 如果同时有新的信件到了,把它们加到托盘中。

当然,如果收到的信件超过这个人能够阅读的数量,一些头天晚上收到的信件就可能永远不会被阅读。为此,有时我们需要另一种数据结构。

队列是一个列表,其中所有的加入操作都在一端进行,而所有的删除操作都在另一端进行。这类似于在银行中排队。人进来时站到队尾,而站到队首的人先接受服务。这种安排通常称为先进先出(FIFO)队列,因为第一个到的人总是第一个接受服务的人。

如果上述那个繁忙的办公室工作人员使用队列而不是栈来组织信件,他(或她)最终将能够处理所有头天晚上到达的信件,即使新的信件以不断增加的速度到达。在后面的一些算法中,当要决定采用栈还是队列结构的时候,你会有一些类似的考虑。

B.2 匹配

贯穿本文的大部分技术都涉及到一些将列表进行匹配的概念。两个列表之间最简单的匹配就是看它们是否相同。为了获得更一般形式的匹配,你需要允许在列表中出现变量,这些变量可以和任何元素匹配。变量可以用一个带“?”前缀的原子表示。

一个变量可以匹配任何原子、子列表或者其他变量。这样一个带变量的列表,例如(A ?x C),可以匹配(A B C),(A (B C) C)或者(A ?z C),但是不匹配(A B D C),因为?x 只能替换单个元素。

知道变量的匹配对象通常是很有用的。我们说一个变量被绑定到它所匹配的值。这样,在前面的例子中,?x 在三次成功的匹配中分别被绑定到 B,(B C)和?z。这些结果见图 B.5。

列表1	列表2	匹配结果	绑定
(A ?x C)	(A B C)	成功	?x ← B
(A ?x C)	(A (B C) C)	成功	?x ← (B C)
(A ?x C)	(A ?z C)	成功	?x ← ?z
(A ?x C)	(A B D C)	失败	-----

图 B.5 带变量匹配的一些简单例子

有了以上背景,我们现在就可以更精确地定义匹配的概念,即:

我们称两个列表合一,如果存在列表变量绑定的某个集合,将变量替换成它们的绑定值,这两个列表会变得完全相同。

给定两个合一的列表,可以存在很多可能的绑定使它们变得相同。举个例子,要将(A ?x C)和(A ?z C)合一,对?x 和?z 有很多可能的绑定,如图 B.6 所示。当这种情况发生时,总是存在一个绑定的集合,这个集合中的绑定可以产生一个新的列表,这个列表可以和所有其他的结果合一。这称为最广合一(most general unifier)。在图 B.6 中,结果 1 是最广合一,因为它能够和结果 2 以及结果 3 合一。结果 2 和结果 3 都不是最广合一,因为它们互相之间不能合一。如果两个列表合一,总是存在一个最广合一,而且可以用一种相当有效的算法找到它,这种算法称为合一算法。

绑定	结果
1. $?x \leftarrow ?z$	$(A ?z C)$
2. $?x \leftarrow B, ?z \leftarrow B$	$(A B C)$
3. $?x \leftarrow C, ?z \leftarrow C$	$(A C C)$

图 B.6 $(A ?x C)$ 和 $(A ?z C)$ 的三种可能的合一结果

考虑合一中一种更复杂的情况。一些变量在同一个列表中可以多次使用。在这种情况下,一些看上去可能成功的匹配实际上会失败,因为不存在该变量的一个单一的值,使得这两个列表相同。例如,列表 $(A ?x ?x)$,就不匹配 $(A B C)$,因为 $?x$ 不能同时被匹配到 B 和 C 。如果 $?x$ 被绑定到 B ,第一个列表就变成了 $(A B B)$,这与 $(A B C)$ 不同。类似地,如果 $?x$ 被绑定到 C ,第一个列表就变成了 $(A C C)$,它也不同于 $(A B C)$ 。

子列表中还会出现更复杂的情况。例如, $(A ?x C)$ 可以匹配 $(A(f ?y) C)$,因为你可以将 $?x$ 绑定到 $(f ?y)$,使得这两个列表相同。更进一步, $(A ?x C)$ 可以匹配 $(A(f ?y) ?y)$,因为如果你让 $?x \leftarrow (f C)$ 并且 $?y \leftarrow C$,这两个列表都变成了 $(A(f C) C)$ 。

用非形式化的语言描述,你可以通过逐个元素地遍历两个列表来对它们进行匹配。每次当你发现一个变量需要绑定时,改写这两个公式,每次出现这个变量时都用它新绑定的值来替换,然后继续匹配。如果你足够小心,没有产生不必要的绑定,就可以得到一个最广合一。图 B.7 和图 B.8 包含了这个过程的两个跟踪结果,其中一次成功,一次失败。在 B.5 节中给出了详细的合一算法。

<p>要将$(A ?x C ?x)$匹配到$(A(f ?y) ?y ?z)$:</p> <ol style="list-style-type: none"> 1. 第一个元素是:A 和 A 2. 第二个元素是:$?x$ 和 $(f ?y)$ 将$?x$绑定到$(f ?y)$,将两个公式重写为: $(A(f ?y) C(f ?y))$和$(A(f ?y) ?y ?z)$ 3. 继续处理第 3 个元素:C 和 $?y$ 将$?y$绑定到 C,重写两个公式为: $(A(f C) C(f C))$和$(A(f C) C ?z)$ 4. 继续处理第 4 个元素:$(f C)$ 和 $?z$ 将$?z$绑定到$(f C)$,重写两个公式为: $(A(f C) C(f C))$和$(A(f C) C(f C))$ <p>两个列表完全相同,所以它们合一。</p>
--

图 B.7 合一过程的非形式化跟踪结果

<p>要将$(A ?x C ?x)$匹配到$(A(f ?y) ?y ?y)$:</p> <ol style="list-style-type: none"> 1. 第一个元素是:A 和 A 2. 第二个元素是:$?x$ 和 $(f ?y)$ 将$?x$绑定到$(f ?y)$,将两个公式重写为: $(A(f ?y) C(f ?y))$和$(A(f ?y) ?y ?y)$ 3. 继续处理第 3 个元素:C 和 $?y$ 将$?y$绑定到 C,重写两个公式为: $(A(f C) C(f C))$和$(A(f C) C C)$ 4. 第 4 个元素是:$(f C)$ 和 C 失败,因为不能使它们相同。
--

图 B.8 一次失败的合一过程的非形式化跟踪结果

B.3 搜索算法

本质上所有的人工智能程序都使用某种形式的搜索来找到想要的结果。事实上,人工智能的一个领域专门研究不同的搜索算法。本节介绍搜索算法中一些基本的思想。

抽象地看,所有的搜索算法都是在探索一个具有不同的可能状态的空间,以找到一些能够满足搜索目标的状态。举例来说,在一个下国际象棋的程序中,一个状态就表示下棋过程中的第一个棋盘局面。而目标就是要找到一个状态,使得下棋程序能够将对手将死。要搜索一个赢的状态,程序要使用一种转换函数,根据输入的一个状态生成另一个状态。在下棋的程序中,这个转换函数可以定义为棋子的移动。给定一个状态,程序根据一个可能的移动计算出一个新的状态。典型地,程序需要根据所有可能的移动来考虑产生的状态集合。这些状态被称为后继状态。

国际象棋是一种非常复杂的游戏,会产生相当复杂的搜索空间。这里,我们先讨论一个非常简单的例子。假设你想找到一个朋友的密码锁的数字组合。你可以随机地尝试各种组合并且希望有好运气,但是如果你想要系统地考虑所有的可能,则应该决定一种专门的搜索策略。我们可以按照下面的方式将这个问题形式化为一个状态空间搜索问题。这里,“状态”表示你已经处理过的数字的一个列表。当你拨另一个数字时,就完成了一次“移动”,并产生了一个新的状态,表现为在原来的状态尾部加入这个新的数字。假设我们并不知道需要拨多少数字。为了使这个例子保持在可控制的范围内,我们还假设密码锁只有两个数字 1 和 2。图 B.9 给出了涉及最多 3 个数字的状态空间。从初始状态()开始,我们拨数字 1 或者 2,这样就有了两个后继状态(1)和(2),如图所示。从这两个状态开始,我们可以拨第二个数字,这样每个状态又有两个后继状态,这样一直继续下去。很多搜索策略的研究都可以在一个一般性的框架下进行,这个框架需要维护一个还需要尝试的状态列表。假设 TODO 就是一个你需要考虑的状态列表。

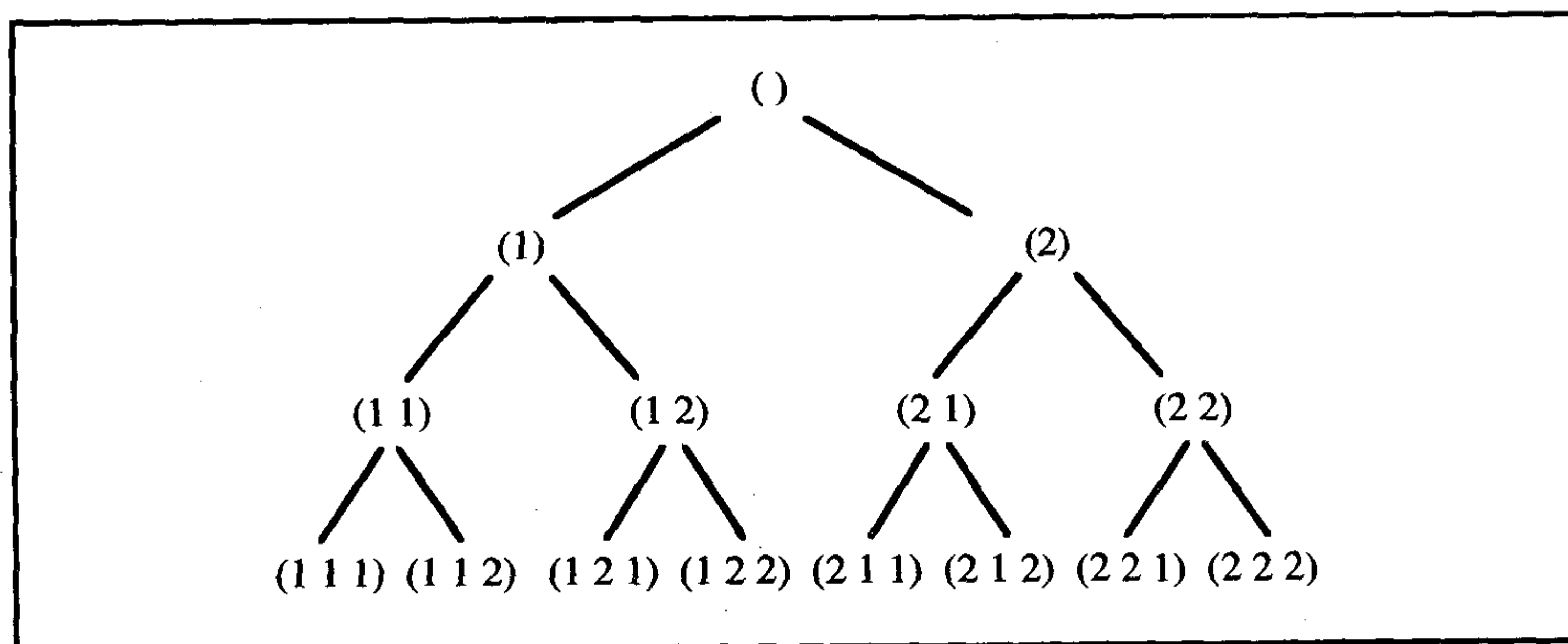


图 B.9 简单密码锁的搜索空间

1. 将 TODO 初始化为仅包含初始状态的(())的列表;
2. 重复下面的操作直至找到目标状态:
 - 2.1 从 TODO 中移出一个状态;
 - 2.2 检查它是否是目标状态(是否能开锁?);
 - 2.3 如果不能,生成所有后继状态,并将其加入到 TODO 列表中。

根据选择下一个要处理的状态的策略的不同,产生了不同的搜索策略。有两种基本的策略。第一种是深度优先策略,使用一个栈作为 TODO 列表。这样,它总是选择最后一个加到栈中的状态做进一步的处理。考虑在密码锁问题的处理中采用这种策略,假设实际的组合是 (2 1 1)。一开始,TODO 中只有初始状态 ()。两个后继状态被加入到列表中,也就是 (1) 和 (2)。加入这两个状态后,TODO 值为 ((1)(2))。下一步选择状态 (1)。发现它不是目标状态 (也就是说,它不能开锁),因此两个后继状态, (1 1) 和 (1 2), 产生并加到了 TODO 列表中,这时 TODO 列表变成了 ((1 1)(1 2)(2))。下面选择状态 (1 1), 发现锁依然无法打开后,两个后继状态 (1 1 1) 和 (1 1 2) 产生出来,并被加入到 TODO 列表中,这时 TODO 列表变成了 ((1 1 1)(1 1 2)(1 2)(2))。知道 (1 1 1) 不是密码后,后继状态 (1 1 1 1) 和 (1 1 1 2) 被加入到 TODO 列表中。现在你也许注意到了,除非给这个算法规定一个需要产生的数字位数的上限,否则这个策略永远不会找到答案。

另外一种基本的策略称为广度优先搜索,将 TODO 当做队列而不是栈。这样,后继状态被添加到 TODO 列表的末尾。还是从状态 () 开始,TODO 列表在算法运行过程中依次取下面这些值:

((1)(2))
 ((2)(1 1)(1 2))
 ((1 1)(1 2)(2 1)(2 2))
 ((1 2)(2 1)(2 2)(1 1 1)(1 1 2))
 ((2 1)(2 2)(1 1 1)(1 1 2)(1 2 1)(1 2 2))
 等等

可以看到这两种策略以很不一样的顺序探索这个空间。如果在图 B.9 所示的表示这个搜索空间的树中依据这两种策略来跟踪搜索到的状态,会看到深度优先策略在这棵树上快速向下移动,不断地在更深的层次上探索状态。另一方面,广度优先策略总是系统地探索树的某一个层次上所有可能的状态,然后才考虑下一个层次的状态。

对于有限的搜索空间[举例来说,我们可以指定密码最多只有 4 位,因此类似 (1 1 1 1) 的状态就没有后继状态],通常深度优先策略会更快地找到答案。不过,对于无限的搜索空间,就像对于不限长度的密码锁问题一样,深度优先策略不能保证会找到答案,因为它可能会沿着一条具有无限后继状态但又不包含答案的链搜索下去。广度优先策略可以保证如果存在一个答案就一定能够找到。

很多其他搜索策略也是可能的。一种称为迭代深化的策略类似于深度优先策略,不过它在到达一个特定的深度后就会停下来。如果在特定的深度上没有找到答案,它就增加深度并重复上述操作。虽然这会导致重复劳动,不过人们发现它总体上要好于纯粹的深度优先或广度优先策略。作为选择,你也可以使用启发式搜索,这种策略采用某种启发式信息来选择下一个状态。举例来说,如果你认为不包含相同数字反复出现的序列更有可能是密码,你也许总是尽可能选择一个不包含重复数字的状态,只有在不存在这种状态时才考虑其他的可能性。你可能将包含重复数字的状态加入到 TODO 列表的末尾,而将其他状态加入到前面。

((1)(2))
 ((1 2)(2)(1 1))
 ((1 2 1)(2)(1 1)(1 2 2))
 等等

启发式搜索的一种一般化的形式描述称为最佳优先搜索。这种策略使用一个函数,为每一个状态返回一个启发式的值。在搜索的每一步,你都选择值最高的状态。

B.4 逻辑程序设计

很多自然语言处理应用程序都是用编程语言 PROLOG 实现的,或者用建立在 PROLOG 上的某种扩充来实现。PROLOG 被证明是一种有用的工具,它建立在合一和搜索的概念基础上。这样,它提供了很多人工智能应用所必需的基本构造单元。

PROLOG 建立在霍恩子句(Horn clause)的概念基础上,霍恩子句可以定义为一个逻辑蕴涵式,蕴涵式的左侧是单一的简单命题。这样,PROLOG 中的表达式:

friendly(fido1) :- dog(fido1) well-fed(fido1)

对应于断言:Fido 是友好的,如果它是一条狗而且它吃饱了。这在一阶谓词演算中表达如下:

$(Dog(Fido1) \& Well-Fed(Fido1)) \supset Friendly(Fido1)$

当然,更有用的规则会使用变量,以便应用于一系列的实体。一个关于“吃饱了的狗是友好的”一般性语句表达如下,其中,变量用一个大写字母开头的符号表示:

1. **friendly(X) :- dog(X) well-fed(X)**

这等价于一阶谓词演算中的断言 $\forall x (DOG(x) \& WELL-FED(x)) \supset FRIENDLY(x)$ 。在 PROLOG 中,它的解释是过程性的:对任何实体 x,要证明 x 是友好的,可以证明 x 是一条狗,而且 x 吃饱了。

没有右侧的霍恩子句称为事实。要声明 Fifi 是一条狗而且它吃饱了,可以加入以下事实:

2. **dog(fifi) :-**

3. **well-fed(fifi) :-**

使用子句 1,子句 2 和子句 3,PROLOG 现在可以如下证明 Fifi 是友好的:

Goal: Friendly(fifi)

使用合一算法,子句 1 被具体化以便用于这个目标(也就是说,变量 X 被绑定到 fifi)

4. **friendly(fifi) :- dog(fifi) well-fed(fifi)**

为了根据这条规则给出目标的证明,需要证明以下子目标:

5. **Goal: dog(fifi)**

6. **Goal: well-fed(fifi)**

由于这些子目标就是子句 2 和子句 3 的断言,其证明是不言而喻的,所以初始的目标得到证明。

一般地,很多不同的规则可以应用于一个目标。可以逐个尝试它们,直到有一个成功。作为另外一个例子,考虑以下公理:

所有的鱼都生活在海里。

7. **live-in-sea(X) :- fish(X)**

所有的鳕都是鱼。

8. `fish(X) :- cod(X)`

所有的鲭都是鱼。

9. `fish(X) :- mackerel(X)`

鲸生活在海里。

10. `live-in-sea(X) :- whale(X)`

Homer 是一条鲭。

11. `mackerel(homer) :-`

Willie 是一头鲸。

12. `whale(willie) :-`

给定这些公理,系统可以证明 Willie 生活在海里,证明中使用的方法称为回溯搜索。它使用深度优先搜索策略来系统地搜索每一种可能的子句应用序列,用以检查目标是否确立。图 B.10 显示了一个典型的 PROLOG 搜索,根据前面给定的子句,目标是:

`live-in-sea(willie)`

感觉 PROLOG 最好的办法就是运行系统,并使用跟踪工具来精确地观察它是如何工作的。PROLOG 系统在大多数类型的工作站和个人计算机上都可以得到。

```

G1: live-in-sea(willie)
  Trying clause 7, which after unification with G1 is
    live-in-sea(willie) :- fish(willie)
  This creates a new subgoal G2.
    G2: fish(willie)
      Rule 8 applies, giving
        fish(willie) :- cod(willie)
      so there is a new subgoal
        G3: cod(willie)
        × No rule applies, try other ways to prove G2
      Rule 9 applies, giving
        fish(willie) :- mackerel(willie)
      so there is a new subgoal
        G4: mackerel(willie)
        × No rule applies, try other ways to prove G2
      × No other rules apply to G2, try other ways to prove G1
    Rule 10 applies giving
      live-in-sea(willie) :- whale(willie)
    So there is a new subgoal
      G5: whale(willie)
      Fact 12 unifies with G5
      √ Thus Goal G5 is Proved.
    √ Goal G1 is Proved.
  
```

图 B.10 `live-in-sea(willie)` 的证明

B.5 合一算法

本节详细给出合一算法。回想一下,两个任意的包含常数和变量的列表合一,如果存在一个变量绑定的集合,使得这两个列表完全相同。

变量值可以保存在一个称为符号表(ST, symbol table)的数据结构之中。假设你有一个如下的 ST:

变量	值
?x	A
?y	?z
?z	B

这表示?x 具有值 A, ?y 具有值 ?z, 而 ?z 具有值 B(这样 ?y 也具有值 B)。需要定义两个函数:

ADD-TO-ST(varname, value)——向 ST 中加入一个新的条目。

GET-VALUE(varname)——返回某一个变量的值。

GET-VALUE 必须反复检查符号表以找到该变量的最具体的值。例如,上面的 ST 中 GET-VALUE(?y)应该返回 B。如果一个变量在 ST 中没有相应的条目,该变量名本身被返回。这样,给定上述符号表,GET-VALUE(?t)将返回?t。

有了这些工具,图 B.11 定义了一个公式 T1 和公式 T2 之间的匹配算法。这次不是在找到每一个变量绑定的时候都重写两个公式,而是把信息保存在符号表中。如果匹配成功,函数返回 SUCCESS,同时符号表中包含了所有的变量值。

要匹配 T1 和 T2

1. If T1 是个变量

then 将 T1 赋值为 GET-VALUE(T1)

2. If T2 是个变量

then 将 T2 赋值为 GET-VALUE(T2)

3. If T1 = T2

then 返回 SUCCESS

else if T1 是个变量

then ADD-TO-ST(T1, T2), 并且返回 SUCCESS

else if T2 是个变量

then ADD-TO-ST(T2, T1), 并且返回 SUCCESS

else if T1 和 T2 都是列表

then if MATCH(First(T1), First(T2))成功

then 返回 MATCH(Rest(T1), Rest(T2))的结果

else 返回 FAIL

else 返回 FAIL

图 B.11 一个初步的匹配算法

图 B.11 中第 1 步和第 2 步查找变量值,如果它们的值是已知的。第 3 步进行实际的匹配。如果其中一个公式还是一个变量,那么这个变量就被绑定到另一个公式的值。最后,如果 T1 和 T2 都是列表,就检查其每个元素是否匹配,以决定这两个列表是否匹配。看一个例子。考虑在图 B.12 中对这个算法的跟踪,将公式(A ?y ?z)和公式(?x(B ?x) ?x)匹配,初始 ST 为空。

```

匹配(A ?y ?z)和(?x(B ?x)?x)
二者都是列表,因此
  匹配 A 和?x
  ← 返回 SUCCESS,并且 ST 中变量?x 的值为 A
  匹配(?y ?z)和((B ?x)?x)
  二者都是列表,因此
    匹配?y 和(B ?x)
    ←返回 SUCCESS,并且 ST 中变量?y 的值为(B ?x)
    匹配(?z)和(?x)
    二者都是表,因此
      匹配?z 和?x
      第 2 步:?x 具有值 A
      ←返回 SUCCESS,并且 ST 中?z 具有值 A
    ←返回 SUCCESS
  ← 返回 SUCCESS

```

图 B.12 匹配算法跟踪

最后结果是 SUCCESS,并且 ST 的内容如下:

变量	值
?x	A
?y	(B ?x)
?z	A

如果将其中一个公式用 ST 中的变量进行实例化,会得到结果(A(B A) A)。注意?y 被绑定到 (B ?x),而?x 被绑定到 A,这样?y 实际上被绑定到值(B A)。

还有一个小问题需要解决。如果你试图用现在的算法匹配公式(P ?x ?x)和(P(f ?y)?y),算法会成功返回并得到 ST 为:

变量	值
?x	(f ?y)
?y	(f ?y)

问题在于现在没有一种变量的实例化方法能够使得这两个公式变得相同。这时因为?y 具有值(f ?y),这实际上是(f(f ?y)),这又应该替换为(f(f(f ?y))),这样一直重复下去。每次当你将?y 替换为变量值时,另一个?y 又被引入进来。通过阻止一个变量匹配到一个包含这个变量本身的价值,这种情况可以消除掉。最终的算法见图 B.13。不过,实际上,大多数 PROLOG 系统出于效率的原因都使用这个更简单的版本。

B.6 参考资料

人工智能程序设计技术的教材是这个领域的最佳补充阅读材料,其中包括模式匹配的章节,例如 Winston 和 Horn(1989)以及 Wilensky(1986)。Gazdar 和 Mellish(1989b)显示了如何将 PROLOG 用于广泛的自然语言处理应用中。人工智能程序设计技巧的一本极好的书是 Norvig (1992)。

要匹配 T1 和 T2

1. **If** T1 是个变量
 then 将 T1 替换为 GET-VALUE(T1)的值
2. **If** T2 是个变量
 then 将 T2 替换为 GET-VALUE(T2)的值
3. **If** T1 = T2
 then 返回 SUCCESS
 else if T1 是个变量并且 T2 中不包含 T1
 then ADD-TO-ST(T1, T2), 并且返回 SUCCESS
 else if T2 是个变量并且 T1 中不包含 T2
 then ADD-TO-ST(T2, T1), 并且返回 SUCCESS
 else if T1 和 T2 都是列表
 then if MATCH(First(T1), First(T2))成功
 then 返回 MATCH(Rest(T1), Rest(T2))的结果
 else 返回 FAIL
 else 返回 FAIL

图 B.13 合一算法

B.7 习题

1. 【易】使用列表函数 First, Rest, Insert 和 Append, 写一个函数进行以下转换:
 - a. 将(B A)转换成(A B)
 - b. (A B)转换成(A B C)
 - c. 将(D B C)转换成(A B C)和 D
2. 【易】如果可能, 将下面的列表进行合一。如果不能, 指出它们为什么不能合一。给出最后的符号表和最广合一。
 - a. ((A ?X) ?X), (?Y(B c))
 - b. (?X ?Y), ((A ?Y)(B ?X))
 - c. ((A ?X) ?X), (?Y ?Z)
3. 【易】找到以下子句的最广合一, 如果找不到, 请解释它们为什么不能合一(其中 x, y, z 是变量):
 - a. P(f(x), y), P(z, g(z))
 - b. P(f(x, x), A), P(f(y, f(y, A)), A)
 - c. P(f(A), x), P(x, A)
4. 【易】假设图 B.9 是密码锁问题的完全搜索空间(也就是说, 你已经知道密码不超过 3 位)。考虑按照深度优先搜索, 重新画出整棵树, 给出所有经过的状态, 直到找到答案(1 2 2)。为每个状态按照考虑的顺序标记一个序号。现在, 按照广度优先算法做同样的事情。哪种策略找到答案更快? 如果答案是(2 2), 你发现哪种方法先找到答案?
5. 【易】使用图 B.10 的形式, 跟踪以下的证明:

live-in-sea(homer)

附录 C 语音识别与口语

虽然本书的大部分内容都关注书面语言,不过这些技术对于口语理解也都是适用的。由于近期语音识别技术的进展,你会在未来的几年内看到很多语音驱动的应用系统。本附录介绍了语音识别技术及其在口语理解系统中的应用。C.1 节讨论了语音识别系统的各种一般性特点。C.2 节概述了语言的基本声音结构,而 C.3 节讨论了信号处理算法。之后,C.4 节描述了一个语音识别系统是如何工作的,而 C.5 节讨论了如何在口语理解中使用这些系统。最后,C.6 节讨论了韵律和语调以及它们在语音中的角色。

C.1 语音识别中的问题

语音在人类交流模式中占支配地位。当然,书面语言是很重要的,很多知识都是以书面语言的形式一代一代地传下来的,但在日常的交流中,语音还是最主要的模式。很自然地可以假设语音也是人机交互的首选模式。语音非常有效和方便,可以让你的双手自由地做其他工作。不过,直到最近,语音识别系统才做到足够准确和高速,以支持有效的应用。随着新的识别技术的产生和更快速的计算机的出现,这种转变非常之快。

语音识别系统分为两类,分别是孤立词识别系统和连续语音识别系统。孤立词识别系统一次只识别一个单词。要使用这样一个系统,必须在单词之间有一个停顿。连续语音识别系统可以识别我们平常说话那样的语音,其中的单词在一个连续流中一起说出。目前,市场上大部分系统都使用孤立词识别技术。连续语音识别系统处于积极的开发之中,不过,已经很接近实际的应用^①。其他区分各种不同系统的主要因素是词汇量和能够处理的说话者的范围。一些低端系统可以识别单个用户的 30 个左右的单词,而高端系统能够识别多个说话者的 20 000 个单词。当比较不同系统的识别率的时候,很重要的一点是要记住,要在大词汇量、多说话者的连续语音识别中达到很高的准确率是非常困难的。

虽然在口语和书面语言的处理中都可以使用一些相同的基本技术,如句法分析、语义解释和上下文解释,但还是存在一些显著的区别,会对这两种系统的设计发生影响。举例来说,对于口语输入,系统必须处理不确定性。在书面语言中,系统准确地知道要被处理的单词。而在口语中,用户说的是什么只是一个猜测。而且,口语在结构上和书面语很不相同。实际上,有时一份完全能够理解的语音的记录在阅读时却令人无法理解。口语的表现更增量化,每次一个短语,包含了书面语中所没有的丰富的语调信息。口语中还包含很多更正,说话者用来纠正或者修改他刚才说的话。更进一步,口语对话有丰富的表示认可或者确认的交互,以维持一次对话,而这些在书面形式中都不会出现。

口语理解系统的基本结构如图 C.1 所示,其中整个的自然语言系统都压缩到了一个框中。说话者发出的声音首先通过一个模拟/数字转换器转换成数字形式。这种信号通过处理抽取

^① 这是原书出版时的市场情况,现在连续语音识别系统应用已经很普遍。——译者注

出各种特征,如在不同频率上的声音强度和随着时间流逝声音强度发生的变化。这些特征用做语音识别系统的输入,语音识别系统一般都采用隐马尔可夫模型(HMM, Hidden Markov Model)技术(见第 7 章)来确定最有可能产生这种语音的词语序列。然后,语音识别器输出这种最可能的词语序列作为自然语言理解系统的输入。当自然语言系统需要产生一个语音的表示时,它将句子传递给一个将单词翻译成音素序列并决定语调轮廓的模块,然后将这些信息传递给一个语音合成系统,语音合成系统产生口语输出。

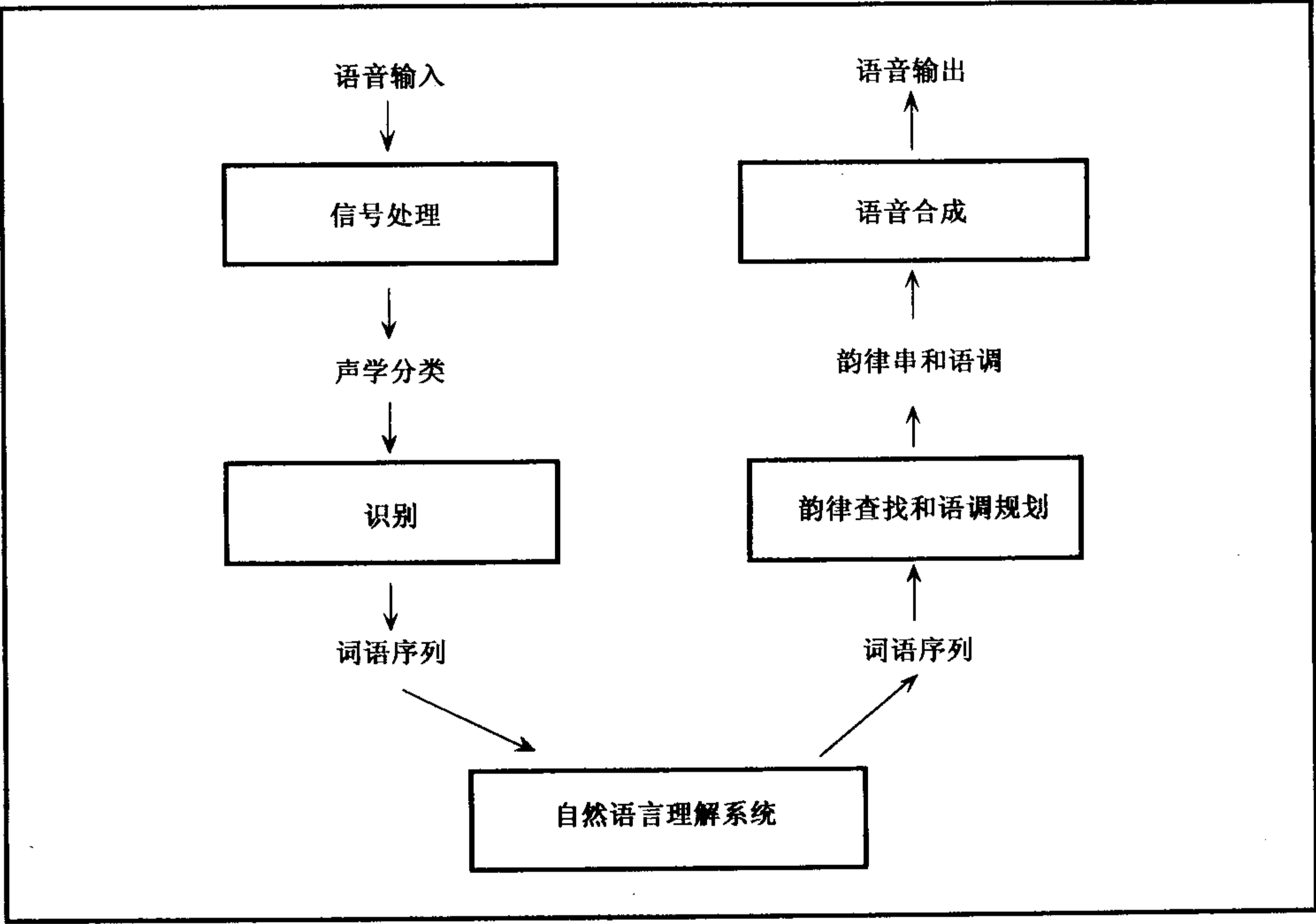


图 C.1 语音理解系统的结构

C.2 语言的声音结构

口语的声音结构可以解释为两个不同的层次,分别是根据声音在语言中的作用进行分类的音素层,以及根据声音的物理特征进行分类的声学层。

在音素层,语音被划分为意义的基本构成单位,类似于在词语形态学中将词语划分成意义的基本构成单位。为了做到这一点,你必须考虑在一种语言中使用的所有不同的声音,并把它们进行归类,使得那些从来不会用于区分单词的声音被归入同一个类中。举例来说,在英语中,与 th 相关联的有两个不同的声音。一个发音时带有较强的气流,可以在单词 thin 中找到这个音,另一个声音则较为有力,可以在单词 then 中找到。它们可能是不同的音素,但是在英语中可以发现,不存在这么一对单词,除了 th 的发音以外完全相同。这样,我们就可以将这两个声音归入一类,作为同一个音素的不同变化形式(音位变体)。如果在英语中进行这种分类,最终会得到 40 个不同的类别,如图 C.2 所示。其他语言可以有不同的集合,有些区分可能和这里一样,而有些分类可能比英语更细致或者更粗略。举例来说,在日语中就不区分/r/和/l/这两个音。

音素	例子	音素	例子	音素	例子
/ee/	beat	/p/	pin	/s/	sin
/i/	bit	/t/	tin	/z/	zoo
/e/	bait	/k/	kin	/zh/	plea sure
/ε/	bet	/b/	bin	/jh/	jail
/ae/	bat	/d/	din	/m/	mail
/a/	bought	/g/	go	/n/	nail
/o/	boat	/f/	fin	/ng/	sing
/u/	book	/th/	thin	/l/	line
/oo/	boot	/sh/	shin	/r/	rib
/uh/	but	/ch/	chin	/w/	will
/er/	bird	/h/	honk	/y/	yes
/ai/	bite	/v/	verb		
/au/	bough				
/oi/	boil				

图 C.2 美国英语的音素

根据某些音素所共有的区别性特征,音素可以划分成一些更一般的组。举例来说,所有的元音和一些辅音都要使用声带发出声音,而有些辅音则不需要,如/s/和/sh/就不使用声带发声。前一组称为浊音,而后一组称为清音。有些辅音,例如/p/和/k/,需要短暂地阻止气流然后快速地释放。这些音称为爆破音。还有一些辅音,例如/n/和/m/,需要用鼻腔来改变声音,因而被称为鼻音。根据发音,很多不同的方法将音素进行分类,一个音素集合可以表现为一个用于独立区分每一个音素的特征的集合。

对于语音识别来说,另一个很重要分析层次是声学层。在这个层次上,声音根据语音信号的物理特征进行分类。本质上,一个语音识别系统必须找到一种办法,将一系列的声音模板跟新输入的语音进行匹配,以确定说出的是哪些单词。遗憾的是,语音信号本身都太过复杂,无法直接表示出来,并且两次说同一个单词产生的信号会有戏剧性的不同,即使是同一个说话者也会这样。声音信号受到发音速度、背景噪声和说话者的情绪状态等因素的极大影响,所有这些因素都与所说的单词没有直接的关系。关键的技巧在于要确定一组声学特征,这些特征能够在同一个声音的很多发音实例中表现一致。这个问题将在下一节中探讨。在这之前,我们考虑一些语音信号的更一般的特点。

一种识别信号特征的基本技术是使用频谱分析,也就是对信号的频率进行分析。这种信息可以通过对输入信号使用模拟滤波器得到,也可以使用数字方法如快速傅里叶变换。这种信息对于区分不同的声音模式是非常有效的。例如,任何一个浊音在低频带往往强度较大,而清辅音的强度在各个频段上的分布较为均匀。为了说明这一点,请考虑图 C.3,该图显示了一个频谱图,这幅频谱图指出了单词 sad 在不同的频率段上的信号强度。频率沿纵坐标变化,从 0 Hz 到 4000 Hz,横坐标是时间。在这个例子中,这个单词出现在这段话开始以后的 3.9 秒到 4.6 秒。黑色说明在这个频率上没有声音,而白色说明在这个频率上声音强度很大。单词 sad 由三个音素组成,分别是/s/,/ae/和/d/。你可以看到/s/在 3.9 秒到 4.1 秒之间,声音强度主要分布在高频段,特别是在 2000 Hz 到 4000 Hz 这一段。元音从 4.1 秒开始,标志是低频段的声音强度增加了。注意有三个强峰,大约在 660 Hz,1700 Hz 和 2400 Hz 的位置。这是浊音的特点,

称为共振峰。这个共振峰频率的组合是元音/ae/的典型特点。第三个共振峰随着元音的持续强度逐渐减弱,而第一个共振峰有轻微的上升和下降。在 4.4 秒声音有一个突然的中断,实际上是产生了一个停顿。这是爆破音的典型特点,包括/d/。短暂停顿持续到 4.5 秒之前,声音又恢复了。注意在 4.5 秒前后有一些弱的共振峰,这是因为/d/是一个浊辅音。如果这个单词是 sat 而不是 sad,那么这个信号的最后一段看上去就会像是开头的/s/音,因为/t/是清音。

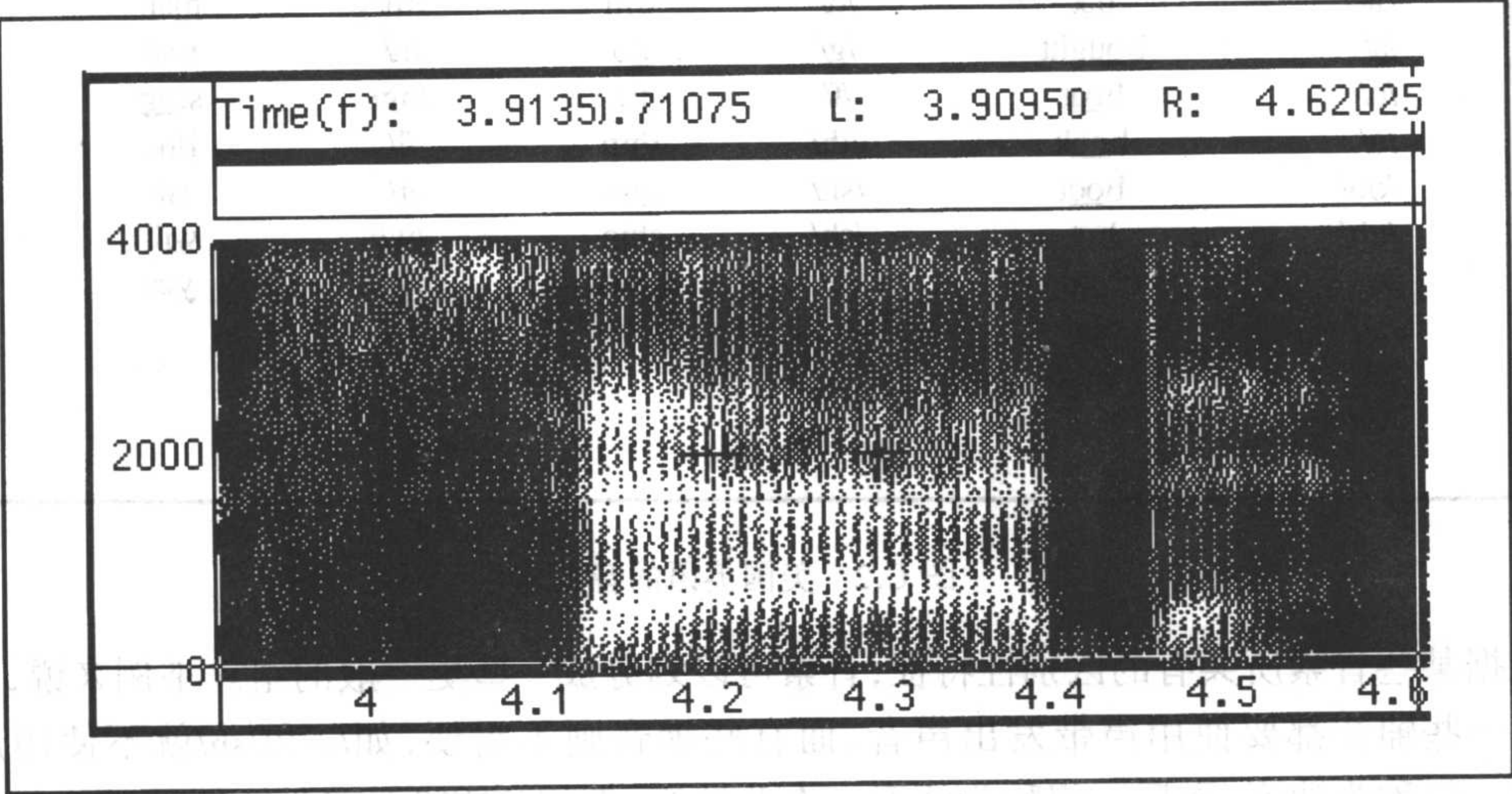


图 C.3 单词 sad 的一幅频谱图

对元音而言前三个共振峰是相当可靠的指示器。图 C.4 显示了一些英语元音的前三个共振峰的平均频率数据,这些数据摘自 Denes(1993)。这些共振峰反映了声道形状的共振,并根据声带发出的频率而有所不同,这个频率称为基频。在大多数语音中,基频作为表示疑问、做出断言、表达惊叹等自然语调的一部分而有很大的变化。但即使它们变了,元音的这些共振峰还是保持相对稳定,因为它们反映了声道的形状。

元音	ee	I	e	ε	ae	a
第一个共振峰	270	390	500	530	660	730
第二个共振峰	2290	1990	1880	1840	1720	1100
第三个共振峰	3010	2550	2520	2410	2410	2440

图 C.4 一些元音的平均共振峰频率

遗憾的是,在音素和声学特征之间不存在一一对应关系。举例来说,音素/t/是拼读单词 time, string, stripe 和 fit 的音素的一部分。但是请你在读这些单词的时候,仔细听其中的每一个音。在每一种情况下,/t/都发出一个相当不同的声音,因为它和周围的音素组合起来了。如果你去看每一个词的频谱,这种差别会更加明显。即使在同一个单词中,根据这个词在句子中是重读还是轻读的不同,与音素相应的声音也会有显著的不同。虽然重读的元音通常可以很可靠地识别出来,而对于非重读的单词或者语速很快的情况下,元音通常会严重衰减,以至于互相之间完全无法区分。这些特点使得语音识别比你所期望的要困难得多。

C.3 信号处理

麦克风根据复杂的声波将声音转换成变化的电流。立体声系统的存在,显示这种电信号包含了声音中包含的相关信息。这是因为,使用一个音箱,电流能够还原成声音,而且还原的声音和原始的声音事实上是无法区分的。使用这样的信号作为计算机的输入,必须首先把它数字化。这由一个模拟/数字(A/D)转换器来实现,而且事实上该转换器是所有现代个人计算机中的一个标准部件。在对一个信号进行数字化的时候有两个重要的因素。第一个是采样频率,表示以多高的频率对电流进行测量;而第二个是量化因子,表示可以被区分的不同声音强度等级的数量。采样频率决定了信号中多高频率的信号能够被记录下来。信息论中的一个定理说明,采样频率必须是分析所需要的最高频率的两倍。作为一个参考值,知觉研究通常指出,语音中出现的频率上限大约为 10 kHz,但如果限制在相当窄的一个频率范围内,语音仍然是可理解的。举例来说,电话语音,传统上只传输最高 3 kHz 的频率信号(虽然现代的电话技术使用了更宽的频率范围)。一般在语音识别应用中,通常使用 8 ~ 20 kHz 的采样频率。

第二个重要的因素是量化因子,它决定了用多大的刻度范围来表现信号的强度。举例来说,1 比特量化只能说明一个信号比某个中间水平的强度高还是低。一个 2 比特度量可以区分四个不同的级别,等等。一般而言,似乎 11 比特的数字可以捕获足够的信息,虽然通过使用对数刻度,8 比特的数字也勉强够用。

这样一个语音信号的典型表现形式是一个 8 比特的数字流,速率是每秒 10 000 个数字,很明显是个很大的数据量。语音识别面临的难题是要将这个数据量缩减为一个易于处理的表示方法。实际上,大多数当前的语音识别系统最终都是将每个信号片段划分到 256 个不同的类别中。很明显,我们必须非常仔细地选取这些类别以保证它们能够捕获信号之间重要的区别而忽略那些无关的变化。本节下面的部分描述如何完成这项任务。

第一种技术是将信号表示为一个片段的序列。在一个信号片段中要捕获多少信号这个问题上有一个折中。片段越大,就有越多的数据可用于分类,但是对于表现那些识别爆破音和其他过渡性辅音所必须的快速转换来说,分类的敏感度会降低。一个典型的片段大小大概是 20 毫秒(在使用 10 kHz 的采样频率时包含 200 个样本)。大部分系统一般都不是将信号划分成孤立的片段,而是采用重叠的片段,以保证片段的边界不会碰巧覆盖了重要的快速转换。一个典型的增量是 10 毫秒,产生的一个片段序列如图 C.5 所示。

现在的任务是提取每一个片段内的信号特征,要尽量捕获那些能够在一个很宽的条件范围内最可靠地标识特定语音信号的信息。对一个片段的两个简单度量标准是:

总强度——强度可以用片段中的数字的平方和来度量,这是判断这个片段是浊音、清辅音还是静音一部分的一个良好指标。

峰值度量——信号中明显的强度峰值的间隔时间很可能反映了元音的基频。

片段的其他度量可以利用频谱分析来获得,可以使用诸如快速傅里叶变换之类的技术,来分析这个信号在不同频率上的强度。频谱分析可用于很多不同的目的。你可以确定频谱中的峰值,这可以反映元音中的共振峰,而不同的频谱模式可以反映不同种类的辅音。通常,在进行更加深入处理以提取关键的频谱特征之前,频谱分析都被用做一个中间阶段。举例来说,你

可能用一个多项式曲线来拟合这幅频谱图,并使用这个多项式的解来作为这个频谱的一个抽象表达。相关文献中有大量的技术将这些信息压缩为少量的特征。

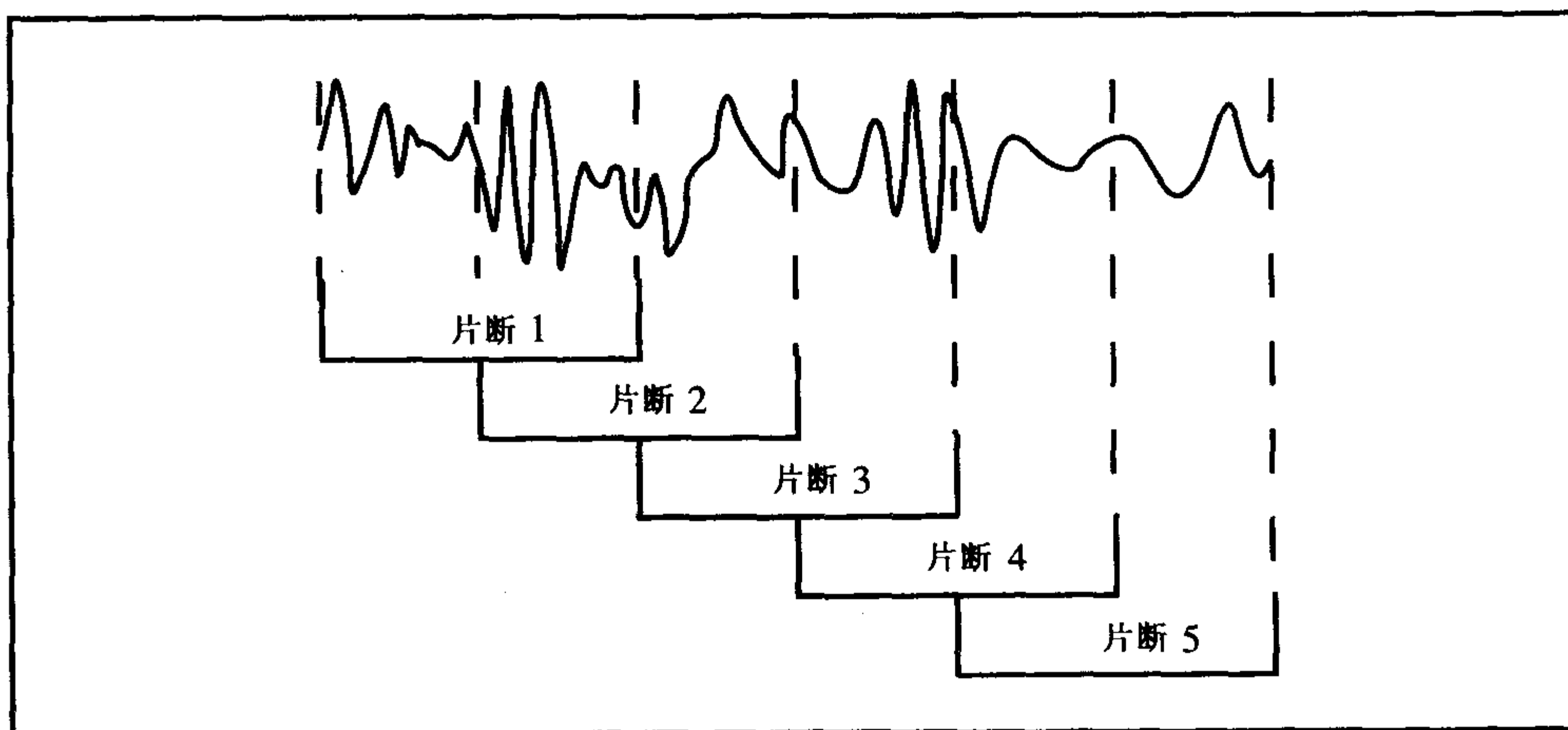


图 C.5 以 10 毫秒间隔的 20 毫秒片段

另外一大类片段分类技术刻画一个片段如何与前面一个片段相关联。特别地,由于片段的重叠和片段持续时间的相对短暂,很多相邻的片段是非常相似的。举个例子,一个元音持续 100 毫秒,在这个区间上会产生 10 个片段,片段之间的变化也会很平缓。对于类似爆破音这样的辅音来说,在一个短暂的区间内会有一个急剧的变化。一个片段可能会被识别为静音而下一个片段可能被识别为浊辅音。

很多系统使用的一种技术是线性预测编码(LPC, linear predictive coding)。这种技术直接在实际的样本值上进行操作,我们把这些样本值记为 $x(1), x(2)$, 等等。在一个 10 kHz 的采样频率上,在一秒钟内有 10 000 个这样的值。这种技术需要寻找一个参数集合 α_k , 使得当前的样本值 $x(k)$, 可以跟据前面的 n 个样本, 采用下面的公式进行预测:

$$x(k) = \sum_{i=1, n} \alpha_i x(k-i)$$

基本的思想是,语音信号的模式可以用找到的系数来反映。要使用这种技术,必须用同一个系数的集合来估计一个单一片段中的所有样本值。当然,这就意味着在预测某些样本值的时候会有一些错误。在一个片段的所有样本上使得这些错误最小的系数用于刻画这个片段。当信号周期性较好的时候,如对于浊音片段,误差会比较小。而对于非周期性的信号,如清音片段,误差会较大。对于浊音片段,这些系数可以用于产生相当可靠的信号共振峰。

总而言之,片段可以通过基于如下信息的特征来进行分类:

- 信号的总强度
- 在各种不同频段上的总强度
- 共振峰
- 上述测量值在片段之间的变化速率

最简单的信号处理系统可能只在 5 到 10 个不同的频率段上度量强度值,而最复杂的系统可能使用非常精巧的算法来标识频谱的形状以及这种形状如何随着时间发生变化。不管怎样,最后的结果都是一个小的数值集合,这个集合构成了语音识别系统的输入。

实际的识别过程是重复进行以下操作:将从训练数据中构造的片段模板与从新输入中构造的片段进行比较。要进行这种匹配,必须基于从每个片段中抽取的特征定义一些相似度的度量准则。为了解释这一点,考虑一个非常简单的系统,这个系统试图将每一个片段分类为一个浊音音素、一个清音音素或者一个静音。假定每一个片段通过一个含有三个数值(从 0 到 256)的向量来表示,这些数值分布代表信号的总强度以及信号在两个频率段上的强度,这两个频率段是 100 ~ 600 Hz 和 600 ~ 1500 Hz。例如,一个由向量(210 140 60)表示的片段的总强度为 210,在 100 ~ 600 Hz 之间强度为 140,而在 600 ~ 1500 Hz 之间强度为 60。在通过一个标注了的数据样本进行训练以后,假设基于每一类型片段的平均值产生了如下的样本:

浊音片段:(180 120 50)

清音片段:(50 10 10)

静音片段:(30 15 10)

我们将通过计算一个输入片段和一个模板之间的最小平方差来度量它们之间的相似度。这个数值越小,这两个模式之间就越相似。输入向量(210 140 60),将产生以下度量值:

与浊音的差距: $(210 - 180)^2 + (140 - 120)^2 + (60 - 50)^2 = 1400$

与清音的差距: $(210 - 50)^2 + (140 - 10)^2 + (60 - 10)^2 = 45\ 000$

与静音的差距: $(210 - 30)^2 + (140 - 15)^2 + (60 - 10)^2 = 50\ 525$

这样,这个片段会被归为一个浊音音素。而另一方面,一个表示为(40 10 12)的片段将会归为一个清音音素(分数为 104,相应的静音分数为 129,浊音分数为 33 144)。

当然,一个语音识别系统会需要比这复杂得多的片段表示方法。一个典型的向量大小含有 24 个数值,捕获各种各样的度量值,诸如强度、共振峰以及与前一个片段相比发生的变化,正如前面讨论的那样。另外,相似度的度量也可能相当复杂,可以反映出某些度量值比另外一些度量值更为重要,或者反映出这些度量值之间的相互依赖关系。一个典型的系统使用 256 个模板进行片段分类,这些模板定义了一个符号的集合,称为码本。有一种自动的技术,可以根据一个给定的训练数据集来设计一个码本以达到最优化的识别性能。

C.4 语音识别

一旦信号处理过程根据码本将信号简化为一个符号序列,语音识别任务看上去会更像一个传统的句法分析问题。具体说,就是给定一个符号序列,必须确定一个单词序列,这个单词序列最有可能产生这个输入的符号序列。这类似于决定句子中每个单词的最佳词性的问题,似乎隐马尔可夫模型应该会很有效。事实确实是这样,今天最成功的语音识别系统都是基于隐马尔可夫模型技术的。

一个重要的问题是考虑在哪一个层次上应用隐马尔可夫模型。显然,一个可能的层次是词语层。对于词典中的每个单词,我们可以定义一个隐马尔可夫模型网络,这个网络定义了可以实现这个词语的最可能的码本符号序列。对于有限词汇表的应用来说,这是一种可行的技术。我们可以获得足够的训练样例来为每一个单词定义这样一个网络。结果将是一个健壮的识别系统。不过,如果你要考虑大词汇量的系统,则找到足够的训练数据是很困难的。举例来说,现在的某些语音识别系统的词汇量已达到 20 000 词或者更多。要为这么大的词汇量找到足够的训练数据几乎是不可能的。

连续语音应用中会出现另外一个问题。一个单词根据其周围词的不同会表现出不同的发音。这种协同发音效应对一个单词的实际发音会有显著的影响。这种现象对于功能词尤为突出,如 the,就受到其紧邻的下一个单词的严重影响。为了处理这种现象,你不仅需要单个词来训练这个模型,而且需要用两个单词的组合来进行训练,这使得训练问题变得更加困难。

由于这些困难,典型的大词汇量语音识别系统都是基于亚词单元的。音素看上去是个自然的单元。在一个基于音素的系统中,用一个隐马尔可夫模型网络来定义每一个音素最可能表现出的码本符号序列。即使对一个包含所有常见音位变体的自由音素集合来说,也只有不超过 100 个单元需要进行训练,这样只训练 5 分钟就可以了。回到前面提到的问题,即音素的实际表现跟周围的上下文高度相关。我们记得音素 /t/ 根据其所处的上下文发音会很不相同。这个问题的一个成功的解决办法是采用三音子(triphone),或称为上下文中的音素(PIC, phonemes in context)。在这个模型中,隐马尔可夫模型并不针对独立的音素 /d/,可能针对的是一个前面是静音后面是原音 /ee/ 的音素 /d/,记为 sil/d/ee,或者另一种组合 sil/d/I, I/d/sil, 等等。当然,完整的三音子集合可能包含至少 40^3 (64 000) 个三音子,这样训练问题又变成难以控制的了。幸运的是,在一种给定的语言中,很多组合永远不会出现,因此实际的数字远没有这么大。而且,可以将前面和后面对中心音素发生相同影响的音素合并成更一般的类。使用这些技术,需要建模的三音子数量就可以保持在一个可控制的规模(在 1000 这个数量级)内。还有,在这个模型中有可能使用平滑技术。举例来说,如果一个特定的三音子从来没有被观察到,也许可以基于两个音素的组合模型,或者更一般地,基于上下文无关的音素模型来估计它的特点。

每一个三音子表示为一个隐马尔可夫模型,这个模型记录了实现这个信号的最可能的码本符号序列。图 C.6 显示了一个音素的简化隐马尔可夫模型。标记为 S 的状态为初始状态。这些状态刻画了符号序列的一般性处理过程,一个典型的这种过程包括三音子的开始部分(状态 B)、然后是中间部分(状态 M)、最后是结束部分(状态 E),当音素识别完成的时候就到达了终止状态 F。每一个节点都与码本符号(C_1, \dots, C_{256})上的一个概率分布相关联。

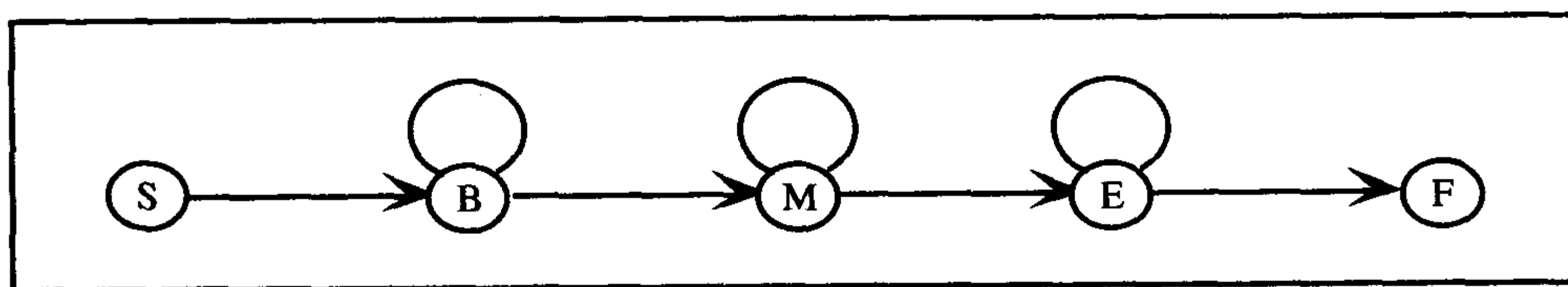


图 C.6 上下文中音素的一个简化的隐马尔可夫模型模板

考虑一个隐马尔可夫模型训练的例子。对于每一个三音子,从数据中选择一个训练用例的集合。每一个训练用例都已表现为一个密码表符号序列,这个序列是由信号处理部件产生的。举例来说,对于三音子 sil/p/I,我们可以从训练数据中找到下面这样一些实例:

C1 C1 C2 C1 C3 C3 C9 C3 C4 C7 C8
 C1 C2 C2 C4 C4 C3 C5 C9
 C2 C2 C3 C5 C5 C6 C3 C8 C7 C7
 C3 C4 C3 C3 C5 C6 C6 C6 C8 C9
 C1 C2 C1 C4 C4 C4 C5 C8 C8

使用标准的隐马尔可夫训练技术,系统可以产生如图 C.7 所示的一个模型,其中转移概率和输出概率的选择使得该网络产生这五个观察序列的概率达到局部最优。注意,对于一个单词通常以相当不同的速率说出所导致的难题,隐马尔可夫模型提供了一种解决办法。这样,一个音素可能有时用 10 个码本符号(100 毫秒)实现,有时又仅以 3 到 4 个符号(30 到 40 毫秒)实现。有些边回到它们的出发状态,允许模型对多个输入符号停留在同一个状态的情况赋予一个概率。不过,还是有很多系统发现这个模型过于粗糙并用一些其他技术对识别系统加以扩充,以便明确地模拟不同音节的延续时间。

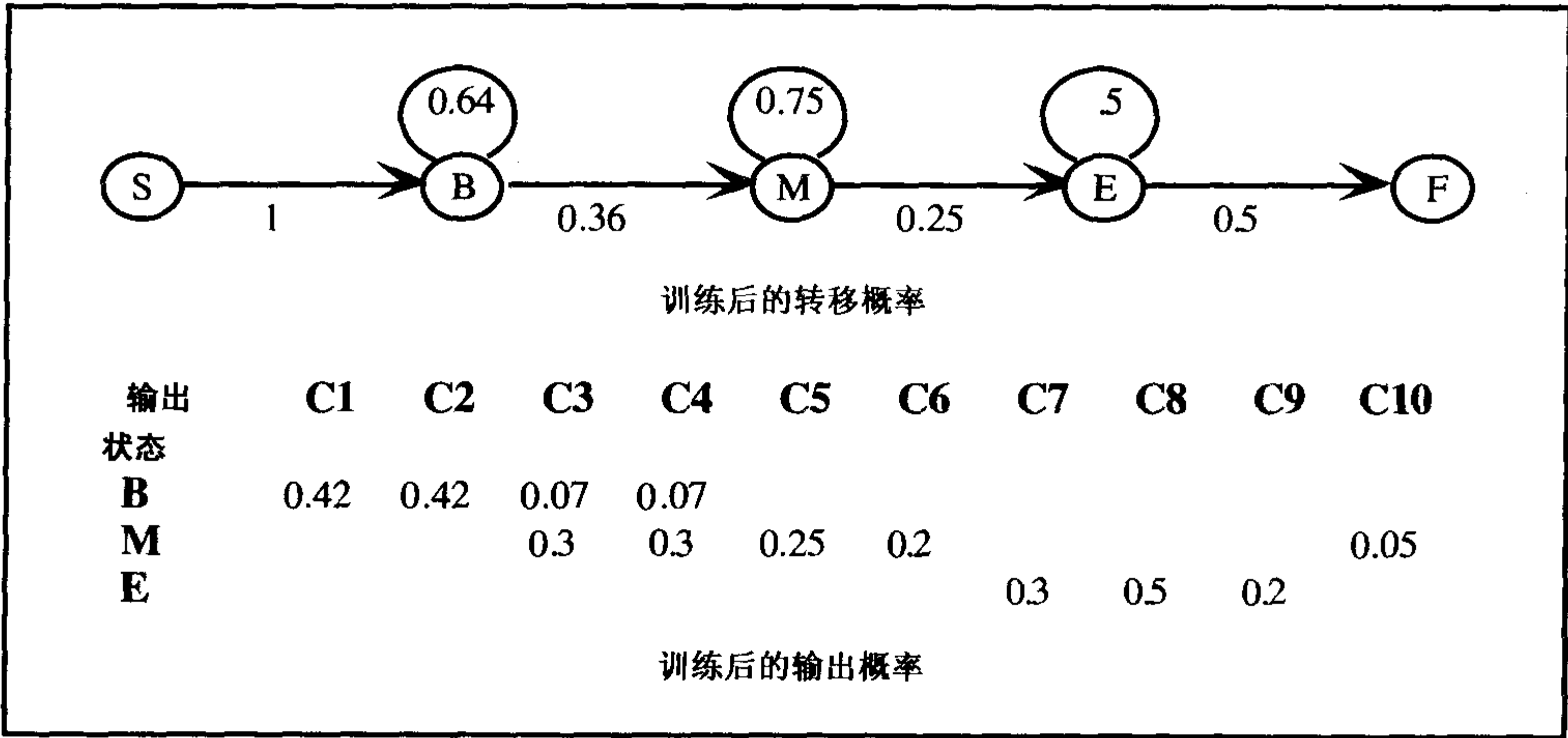


图 C.7 训练后 sil/p/l 的隐马尔可夫模型

一般情况下,音素的隐马尔可夫模型应该比这更复杂,主要是处理一些由于音素处于强调或是简化的音节,或处于诸如非常快速的语音中所导致的变化情况。例如,SPHINX 系统(Lee, 1990)中所使用的基本隐马尔可夫模型如图 C.8 所示。

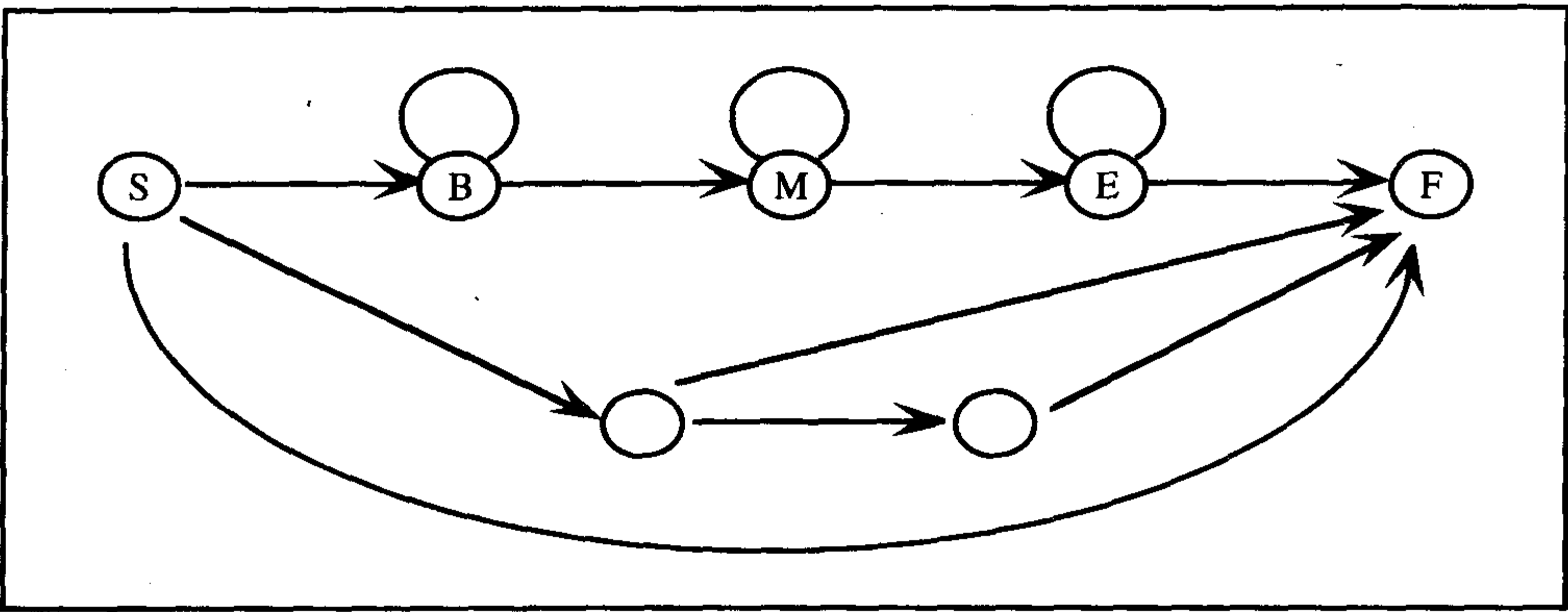


图 C.8 SPHINX 系统中使用的音素隐马尔可夫模型

给定一个观察序列(表示为码本中的符号), O_1, \dots, O_n , 以及为每一个三音子训练好的隐马尔可夫模型,第 7 章中描述的 Viterbi 算法就可以用于计算经过每个表示三音子的隐马尔可夫模型的最佳路径。据此,我们可以在给定每个 HMM 的情况下计算这个序列的概率,从而确定以最高概率产生该序列的三音子。

但是,这种简单的方案只有在输入可以被切分成音素单元并每次一个地进行分析时才有效。实际上这是不可能的,因此识别系统必须一次性解决两个问题:音素的切分和音素的识别。为了对这个任务的范围加以约束,你可以利用定义好的单词来预测哪些音素的序列更可能出现。例如,单词 `please` 按音素拼写为 `/p//l//ee//z/`。为了在输入中识别这个单词,你需要识别以下的三音子序列:

`sil/p/l p/l/ee l/ee/z ee/z/sil`

这些三音子中的每一个都定义为一个隐马尔可夫模型,这样我们就有了一个隐马尔可夫模型序列来对输入集合进行解释。通过把三音子的隐马尔可夫模型连接起来,可以为单词 `please` 构造一个隐马尔可夫模型,这样就创造了一个如图 C.9 所示的新的网络。使用这个网络的 Viterbi 算法可以将观察值按照最佳的三音子边界进行切分,条件是搜索到可能性最大的状态序列。

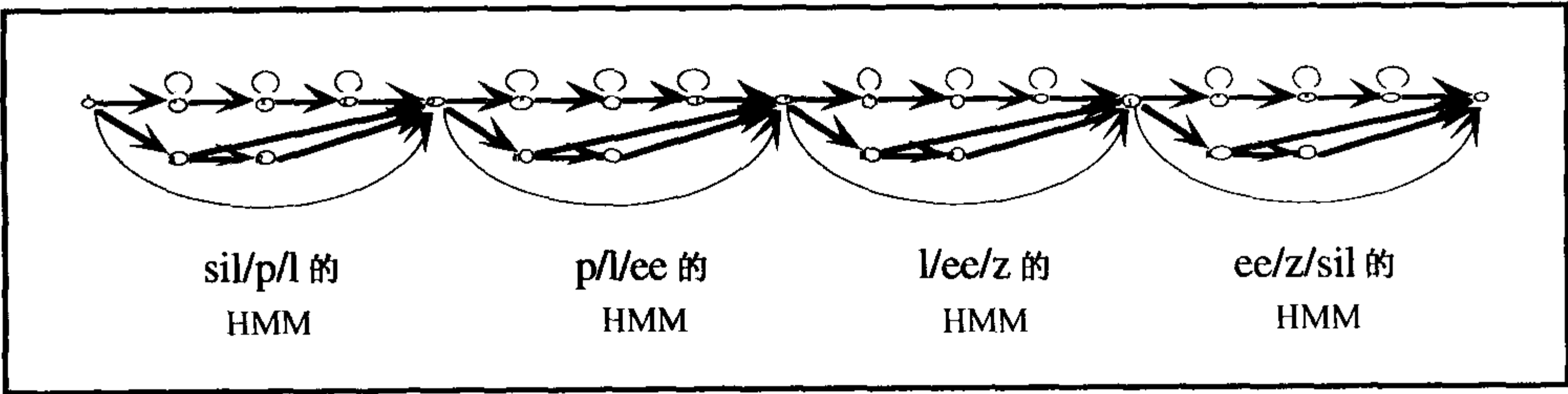


图 C.9 连接三音子的隐马尔可夫模型以形成单词 word 的隐马尔可夫模型

当然,单词 `please` 可以有多个发音,因此多个可选的音素拼写形式也应该是允许的。如果你希望对不同拼写形式的不同使用频率加以考虑,则可以为每个单词定义一个指定该单词实现时的不同音素序列概率的隐马尔可夫模型网络。

这个模型对孤立词来说是有效的,不过对连续语音来说还不够。特别地,两个邻接的单词之间有可能会没有停顿。要处理这个问题,可以构造一个运行两种不同的三音子序列的网络,一种序列有停顿而另一种没有。模拟单词序列 `please send` 的两个可能的音素序列可以是:

`sil/p/l p/l/ee l/ee/z ee/z/sil sil/s/ε s/ε/n ε/n/d n/d/sil`
`sil/p/l p/l/ee l/ee/z ee/z/s z/s/ε s/ε/n ε/n/d n/d/sil`

为了模拟可能的词语序列,语音识别系统的典型做法是使用一个二元语法模型,就如第 7 章所讨论的。这个模型定义了另外一个层次的隐马尔可夫模型,该模型可以模拟类别的转移,并产生词语序列的概率。二元语法模型的应用可以大大提高语音识别系统的精度,因为它相对于其他方法消除了很多可能性。例如,在一次实验中,加入二元词语法以后 SPHINX 系统的准确率从 70% 提高到了 95%(在一个为 ARPA 支持的系统进行评测的资源管理任务中)。同一个系统只使用单音模型而不是三音模型且不使用语法时准确率只达到 50%,但使用二元语法时准确率达到 90%。图 C.10 给出了这个结果。很明显,对于目前语音识别系统的成功,词语法做出了非常重要的贡献。

	不使用语法	使用二元词语法
简单音素模型	50%	90%
三音子模型	70%	95%

图 C.10 使用三音子和二元词语法在性能上的效果

C.5 语音识别和自然语言理解

正如你所看到的,用二元词语法对语音识别系统加以改进可以显著地提高系统的性能。这启发我们,如果采用一个更加全面而详尽的模型,会产生更好的结果。不过,实际上,这是很难做到的。可以使用三元词语法,但是这需要多得多的数据。直接集成一个概率上下文无关语法也会带来困难。首先,语音系统目前很好地集成了二元语法模型、词模型和音素模型,因为它们都可以用同一个框架来表示,即隐马尔可夫模型。引入上下文无关语法机制作为句法部件难于进行有效的集成,并且会对识别的准确率或者有效性造成负面的影响。其结果是,所有现有的口语理解系统都在语音识别和自然语言理解系统之间维持一个严格的分界,如图 C.1 所示。

按照这种划分,在设计接口的时候还是有很多种做法的。最简单的接口,也是最常用的,就是语音识别输出所找到的单一的最佳单词序列。然后,语言处理系统在这个基础上进行处理,并期望没有严重的识别错误。将这种做法更一般化的一种方法称为 N-best 方法,在 N-best 方法中,语音识别输出它所找到的 N 个最佳序列。这允许句法分析器在第一个结果失败时尝试其他解释。虽然很多系统都具有使用 N-best 方法的能力,不过实践证明这只是使得准确率略微有所提高,但比起由此导致的额外的处理开销来说,这样做并不值得。其中的一个原因是,N 个最佳的序列本质上是相同的,只在一到两个单词上有区别。这样,如果语音识别系统识别错了某一个特定的单词,将很有可能在输出的 N 个最佳选择中都有同样的错误。

取代 N-best 的一种有趣的方法是输出词格。在这种方法中,语音识别系统输出一个格子,这个格子给出了输入中最可能的单词。一个词格给出了提供一种大量可能句子的压缩表示形式,并为基于句法分析器和语义解释的错误恢复提供了一个表达能力足够丰富的环境。注意,你可以将词格看做 chart 句法分析器的初始线图。在某个位置出现某个单词有多种选择这一事实对基本的句法分析算法没有影响。

这种通用技术目前还没有被充分研究,因为目前的系统大多都使用高度领域化的技术来优化其短期性能。举例来说,在航空旅行信息系统(ATIS, Air Travel Information System)领域中开发了多个不同的口语理解系统,这些系统基于航空时刻表来回答问题。因为这个应用领域非常集中而且狭窄,所以第 11 章描述的特定领域的解释技术可以用于正确地解释一个查询,即使其中的一部分被错误地识别也没有太大的影响。研究者发现,在短期内改进特定领域解释的启发式方法比探索语音识别和自然语言处理系统之间更通用、更健壮的接口来得更有效。当应用变得更为复杂时,这种现状应该会有所改变。

即使在给定受限领域的情况下,ATIS 领域的系统性能也给人以深刻的影响。一般而言,一个典型的 ATIS 系统包括大约 2000 个单词,独立于说话者,使用标准的不带特殊信号处理硬

件的标准工作站完成语音识别任务时几乎没有可察觉的延迟。系统使用一个包含数百个事先没有见过的查询语句的集合进行测试。1993年,最好的语音识别系统达到了95%的单词准确率。总体最好的系统对于88%的查询语句可以产生合适的回答。

C.6 韵律和语调

语音中还有一个对语言理解特别重要的深层因素,即韵律。韵律是与句子被说出来的方式而不是与内容相关的一些现象的总和。韵律包含多种多样的感知线索,包括:

语调轮廓——随着句子被说出,基频上升和下降所构成的轮廓线。

语速——说话者不断变化语速,拉长某些音节,缩短其他音节,在单词间加入停顿,等等。

强度——说话者在说话的过程中改变强度,强调某个特定的单词,弱化其他单词。

对于口语理解来说韵律是极为重要的,因为韵律传递了说话者与他所说的内容之间的关系。在某些情况下,起到了标点符号在书面语表达中的作用。实际上,应该从反方向来看:书面语中的标点符号扮演了口语中韵律所扮演的角色。举例来说,在一个普通的英语疑问句韵律模式中,在句子的末尾基频往上升。而在一个断言句中,基频往下降。在决定什么句子是疑问句时,韵律模式所起的作用往往超过其他因素,例如句法因素。举例来说,句子“You are coming to the party”通过一个上升语调可以用做一个疑问句,对应的书写形式为:

You are coming to the party?

不仅如此,语调的内容比标点符号要丰富得多。基于不同的语调模式,这个句子可以当做一个邀请、一个真的疑问或者表示惊喜。而在书面语中所有这些解释都用问号来表示。

口语中的另一个重要因素是重音。所有的句子都有一个通常使用的典型重音模式,但说话者可以改变重音以便指出什么是重点,或指出对照和更正的内容。语音中的重音通过声音强度的增大、单词的拉长以及使用特定的韵律模式,如在重音单词上基音短促升高来表示。在书面语中,有时重音表示为把单词中的所有字母都大写,例如:

I only took TWO candies. (that is, not three)

我只拿了两颗糖果。(就是说,不是三颗)

I only took two CANDIES. (that is, no cookies)

我只拿了两颗糖果(就是说,没拿饼干)

考虑对问句“Do you own a red car?”(你是否有一辆红色小汽车?)的回答:

No, but I own a red BIKE.

不,但我有一辆红色自行车。

No, but I own a BLUE car.

不,但我有一辆蓝色小汽车。

如果用下面的句子回答就会让人觉得很奇怪:

* No, but I own a RED bike.

* 不,但我有一辆红色自行车。

* No, but I own a blue CAR.

* 不,但我有一辆蓝色小汽车。

原因在于,重音要强调新的和对比的信息。强调在给定的上下文中已有的单词没有任何意义。

语调轮廓和语速在措辞中也非常重要,可以指明何处是短语或句子的开始或结束。这里是某个人在一次对话中说出的一个单词序列。如果没有韵律信息,你无法知道何处是句子和短语的边界:

Not at the same time OK we're gonna hook up engine E2 to the boxcar at Elmira and send that off to Corning now while we're loading that boxcar with oranges at Corning we're gonna take engine E3 and send it over to Corning hook it up to the tanker car and send it back to Elmira.

没有韵律信息,如果考虑到句子边界位置的变化,则这个词语序列具有严重的歧义。考虑下面这一行中的单词 now。它可能是一个时态性副词,是句子的一部分:

Send that off to Corning now while we're loading the boxcar

另外,它也可以是一个对话的提示词,从而得到以下的句子序列:

Send that off to Corning

Now

While we're loading the boxcar,...

不同的语句分割导致完全不同的解释,而你无法从单词中判断哪一种是话的原意。不过,如果你能够听到原始口语形式,会发现很明显其结构符合后一种解释。

虽然韵律明显地在口语对话中扮演了关键性的角色,但其计算方法并没有被深入地研究。原因之一是目前的语音识别系统都设计为一次只处理单一的一句话(也就是说,句子边界已经在外部被确定了),并且只有很有限的对话容量。其结果是,由韵律传达的大部分信息都与当前的应用无关。一些研究工作显示,韵律分析可以改进语音识别的准确率,不过还没有哪个系统以任何明显有效的方式集成了韵律处理。

韵律起到关键性重要作用的一个地方是语音合成。一个不使用韵律处理的语音合成系统听起来人工的痕迹非常明显而且难于理解。所以,所有最好的语音合成系统都使用启发式方法来决定一个韵律轮廓,典型的做法是基于统计方法来判断书面文本的短语边界。确保这种判断的有效性是非常重要的,因为一个使用不合适韵律的语音合成系统比完全不使用韵律的系统更加糟糕。

C.7 参考资料

Denes(1993)是关于人类语音产生与感知的一个极好的介绍,它很好地描述了语音如何产生及其一般特点,并对语音感知心理学中活跃的研究领域给出了一个成功的综述。一篇有关语音识别的很好的文章是 Rabiner 和 Juang(1993)。

有关语音识别的论文的最好的来源是 Waibel 和 Lee(1990)的论文集。其中包含了很多入门方面的文章。比如,Rabiner(1989)是对隐马尔可夫模型及其在语音识别中的应用的极佳介绍。Schafer 和 Rabiner(1975)是对表示语音信号的不同信号处理技术的优秀综述。Lee(1990)

包含对语音识别中三音子应用的一个极好的讨论。这本论文集还包含了很多阐明重要技术和描述实例系统的文章。

这个领域的当前研究进展的主要信息来源是 *IEEE Transactions on Acoustics, Speech and Signal Processing* 和 *Computer Speech and Language*, 以及由 *Morgan Kaufmann* 发布的 *DARPA Workshops on Speech and Natural Language Processing* 的论文集。

有大量关于韵律的一般性文献, 不过其中很少是可计算的。Pierrehumbert 和 Hirschberg (1990) 的论文描述了一种特定的韵律结构理论, 并讨论了这种理论如何影响对话处理的各个方面。Wang 和 Hirshberg (1992) 是一个语调短语边界自动识别工作的优秀例子。

C.8 习题

1. 【易】给你一个未知元音的频谱, 假定你估计出其前三个共振峰位于 600 Hz, 1800 Hz 和 2400 Hz。给定图 C.4 中指定的共振峰数据, 设计一种对这个未知元音进行分类的数值检验方法。最可能的两个元音是什么? 你的衡量标准更偏向于这二者中的哪一个?
2. 【易】根据以下特征对图 C.2 中给出的英语音素进行分类:
 - +/- 浊音——是否使用声带(如在 /e/, /n/ 和 /d/ 中)
 - +/- 爆破音——音素中是否有一个短促的停顿后接一个快速的释放(如在音 /d/ 和 /t/ 中)
 - +/- 摩擦音——音素中是否包含一个高频的“嘶嘶”声(如在 /s/ 中)
3. 【中】在图 C.7 定义的隐马尔可夫模型上使用第 7 章描述的 Viterbi 算法, 决定下面哪一个输入最可能是三音子 sil/p/l:

C1 C1 C2 C4 C5 C6 C5 C8 C8

C1 C2 C3 C4 C5 C6 C7 C8

参 考 文 献

很多会议论文集和刊物名称都使用了缩写。具体来说,AAAI 是 American Association for Artificial Intelligence(美国人工智能协会)的缩写,其会议论文集由 MIT 出版社发行。IJCAI 是 International Joint Conference on Artificial Intelligence(国际人工智能联合学术会议)的缩写,其论文集由 Morgan Kaufmann Publishers 发行。

ACL 是 Association of Computational Linguistics(计算语言学学会)的缩写,COLING 是 International Conference on Computational Linguistics(国际计算语言学学术会议)的缩写,而 AJCL 是 *American Journal of Computational Linguistics*(《美国计算语言学》)杂志的缩写,它是 *Computational Linguistics*(《计算语言学》)杂志的前身。这些论文集和刊物都可以从以下地址得到:the Association for Computational Linguistics; Walker; C.N.925; Bernardsville, New Jersey, USA, 07924-0925。

Readings in Natural Language Processing, 由 B. Grosz, K. Jones 和 B. Webber 编辑,并由 Morgan Kaufmann Publishers, Inc., 于 1986 年出版,是一本很好的论文集。这本论文集集中的论文在下面都用缩写词 RNLP 来标记。

Aho, A. V., and J. D. Ullman. 1972. *The Theory of Parsing, Translation and Compiling*. Englewood Cliffs, NJ: Prentice-Hall.

Aho, A. V., R. Sethi, and J. D. Ullman. 1986. *Compilers: Principles, Techniques, and Tools*. Reading, MA: Addison-Wesley.

Allen, J. F. 1983. "Recognizing intentions from natural language utterances." In M. Brady and R. C. Berwick (eds.), *Computational Models of Discourse*, 107–166. Cambridge, MA: MIT Press.

Allen, J. F. 1984. "Towards a general theory of action and time," *Artificial Intelligence* 23, 2:123–154.

Allen, J. F., and C. R. Perrault. 1980. "Analyzing intention in utterances," *Artificial Intelligence* 15, 3:143–178. Reprinted in RNLP.

Allen, J. F., R. Fikes, and E. Sandewall (eds.). 1991. *Proc. 2nd Conf. on Principles of Knowledge Representation and Reasoning* (Cambridge, MA, April 1991). San Mateo, CA: Morgan Kaufmann.

Allen, J. F., J. Hendler, and A. Tate. 1990. *Readings in Planning*. San Mateo, CA: Morgan Kaufmann.

Alshawhi, H. (ed.). 1992. *The Core Language Engine*. Cambridge, MA: MIT Press.

Appelt, D. E. 1985. *Studies in Natural Language Processing: Planning English Sentences*. Cambridge, U.K.: Cambridge U. Press.

Appelt, D. E., J. R. Hobbs, J. Bear, D. Israel, and M. Tyson. 1993. "FASTUS: A finite-state processor for information extraction from real-world text," *Proc. IJCAI*.

Austin, J. L. 1962. *How to Do Things with Words*. New York: Oxford U. Press.

Ayuso, D. 1989. "Discourse entities in Janus," *Proc. ACL*.

Bach, E. 1986. "The algebra of events," *Linguistics and Philosophy* 9, 1:5–16.

Baker, C. L. 1989. *English Syntax*. Cambridge, MA: MIT Press.

- Ballard, B. 1988. "A general treatment of comparatives for natural language question answering," *Proc. ACL*, 41–48.
- Bar-Hillel, Y. 1960. "The present status of automatic translation of languages." In F. L. Alt (ed.), *Advances in Computers*. Vol. 1. New York: Academic Press, 91–163.
- Barwise, J., and R. Cooper. 1981. "Generalized quantifiers and natural language," *Linguistics and Philosophy* 4:159–219.
- Barwise, J., and J. Etchemendy. 1987. *Tarski's World*. Chicago, IL: Chicago U. Press.
- Barwise, J., and J. Perry. 1983. *Situations and Attitudes*. Cambridge, MA: Bradford Books, MIT Press.
- Bates, M. 1978. "The theory and practice of augmented transition networks." In L. Bloc (ed.), *Natural Language Communication with Computers*. New York: Springer-Verlag.
- Bates, M., M. G. Moser, and D. Stallard. 1986. "The IRUS transportable natural language database interface." In L. Kerschberg (ed.), *Expert Database Systems*. Redwood City, CA: Benjamin/Cummings.
- Baum, L. E. 1972. "An inequality and associated maximization technique in statistical estimation for probabilistic functions of a Markov process," *Inequalities* 3:1–8.
- Berwick, R. C. 1985. *The Acquisition of Syntactic Knowledge*. Cambridge, MA: MIT Press.
- Berwick, R. C., and A. Weinberg. 1984. *The Grammatical Basis of Linguistic Performance: Language Use and Acquisition*. Cambridge, MA: MIT Press.
- Birnbaum, L., and M. Selfridge. 1981. "Conceptual analysis of natural language." In R. Schank and C. Riesbeck (eds.), *Inside Computer Understanding*. Hillsdale, NJ: Lawrence Erlbaum.
- Blank, G. D. 1989. "A finite and real-time processor for natural language," *Commun. of the ACM* 32, 10, 1174–1189.
- Bobrow, R. J., and B. L. Webber. 1980. "Knowledge representation for syntactic/semantic processing," *Proc. AAAI*, 316–323.
- Bobrow, D. G., and T. Winograd. 1977. "An overview of KRL, a knowledge representation language," *Cognitive Science* 1, 3:3–46.
- Brachman, R. J., and H. Levesque (eds.). 1985. *Readings in Knowledge Representation*. San Mateo, CA: Morgan Kaufmann.
- Brachman, R. J., H. Levesque, and R. Reiter (eds.). 1989. *Proc. 1st Conf. on Principles of Knowledge Representation and Reasoning*. San Mateo, CA: Morgan Kaufmann.
- Bratman, M. E. 1987. *Intentions, Plans and Practical Reasoning*. Cambridge, MA: Harvard U. Press.
- Bratman, M. E., D. Israel, and M. Pollack. 1988. "Plans and resource-bounded practical reasoning," *Computational Intelligence* 4:349–355.
- Brennan, S., L. Friedman, and C. Pollard. 1987. "A centering approach to pronouns," *Proc. ACL*.
- Brown, J. S., and R. R. Burton. 1975. "Multiple representations of knowledge for tutorial reasoning." In D. G. Bobrow and A. Collins (eds.), *Representation and Understanding*. New York: Academic Press.
- Bruce, B. C. 1975. "Generation as a social action," *Proc. Theoretical Issues in Natural Language Processing (ACL)*, 64–67. Reprinted in RNLP.
- de Bruin, J., and R. Scha. 1988. "The interpretation of relational nouns," *Proc. ACL*, 25–32.
- Carberry, S. 1991. *Plan Recognition in Natural Language Dialogue*. Cambridge, MA: MIT Press.
- Carlson, G. 1979. "Generics and atemporal when," *Linguistics and Philosophy* 3:49–98.

- Carlson, G. 1982. "Generic terms and generic sentences," *J. Philosophical Logic* 11:145–181.
- Chafe, W. L. 1975. "Givenness, contrastiveness, definiteness, subjects, topics, and points of view," *Linguistic Inquiry*, 25–55.
- Charniak, E. 1981. "The case-slot identity theory," *Cognitive Science* 5, 3:285–292.
- Charniak, E. 1983. "A parser with something for everyone." In M. King (ed.), *Parsing Natural Language*. New York: Academic Press.
- Charniak, E. 1988. "Motivation analysis, abductive unification, and nonmonotonic equality," *Artificial Intelligence* 34, 3:275–296.
- Charniak, E. 1993. *Statistical Language Learning*. Cambridge, MA: MIT Press.
- Charniak, E., and R. Goldman. 1993. "A Bayesian model of plan recognition," *Artificial Intelligence* 64, 1:53–80.
- Charniak, E., and D. McDermott. 1985. *An Introduction to Artificial Intelligence*. Reading, MA: Addison-Wesley.
- Chierchia, G., and S. McConnell-Ginet. 1990. *Meaning and Grammar*. Cambridge, MA: MIT Press.
- Chincor, N., L. Hirschman, and D. Lewis. 1993. "Evaluating message understanding systems," *Computational Linguistics* 19, 3:409–450.
- Chitrao, M., and R. Grishman. 1990. "Statistical parsing of messages," *Proc. DARPA SNLP Workshop*, San Mateo, CA: Morgan Kaufmann.
- Chomsky, N. 1956. "Three models for the description of language," *IRE Transactions PGIT*, 2, 113–124.
- Chomsky, N. 1965. *Aspects of the Theory of Syntax*. Cambridge, MA: MIT Press.
- Chomsky, N. 1981. *Lectures on Government and Binding*. Cinnaminson, NJ: Foris Publications.
- Church, K. 1988. "A stochastic parts program and noun phrase parser for unrestricted text," *Proc. 2nd Conf. Applied Natural Language Processing*, 136–143.
- Church, K., and R. Patil. 1982. "Coping with syntactic ambiguity, or how to put the block on the box on the table," *AJCL* 8, 3–4:139–149.
- Clark, H. H., and S. Haviland. 1977. "Comprehension and the given-new contract." In R. O. Freedle (ed.), *Discourse Production and Comprehension*. Norwood, NJ: Ablex.
- Clark, H. H., and C. R. Marshall. 1981. "Definite reference and mutual knowledge." In A. Joshi, B. Webber, and I. Sag (eds.), *Elements of Discourse Understanding*. New York: Cambridge U. Press.
- Clark, K. 1978. "Negation as failure." In H. Gallaire and J. Minker (eds.), *Logic and Data Bases*. New York: Plenum Press, 293–322.
- Clifton, C., L. Frazier, and K. Rayner. In press. *Perspectives on Sentence Processing*. Hillsdale, NJ: Lawrence Erlbaum.
- Clocksin, W., and C. Mellish. 1981. *Programming in PROLOG*. New York: Springer-Verlag.
- Cohen, P. R. 1978. "On knowing what to say: Planning speech acts," Ph.D. thesis and TR 118, Computer Science Dept., U. Toronto.
- Cohen, P. R., and H. J. Levesque. 1990a. "Intention is choice with commitment," *Artificial Intelligence* 42, 3.
- Cohen, P. R., and H. J. Levesque. 1990b. "Rational interaction as the basis for communication." In Cohen et al. (1990), 221–256.
- Cohen, P. R., and C. R. Perrault. 1979. "Elements of a plan-based theory of speech acts,"

- Cognitive Science* 3:177–212. Reprinted in RNLP.
- Cohen, P. R., J. Morgan, and M. Pollack. 1990. *Intentions in Communication*. Cambridge, MA: MIT Press.
- Cohen, R. 1987. "Analyzing the structure of argumentative discourse," *Computational Linguistics* 13, 1–2:11–24.
- Colmerauer, A. 1978. "Metamorphosis grammars." In L. Bloc (ed.), *Natural Language Communication with Computers*. Berlin: Springer-Verlag.
- Computational Linguistics* 9, 3–4 (special issue on ill-formed input), 1983.
- Computational Linguistics* 14, 2 (special issue on tense and aspect), 1988.
- Computational Linguistics* 19, 1–2 (special issue on large corpora), 1993.
- Cooper, R. 1983. *Quantification and Syntactic Theory*. Dordrecht: D. Reidel.
- Cottrell, G. W., and S. L. Small. 1983. "A connectionist scheme for modelling word sense disambiguation," *Cognition and Brain Theory* 6:89–120.
- Crain, S., and M. Steedman. 1985. "On not being led up the garden path." In D. R. Dowty, L. Karttunen, and A. Zwicky (eds.), *Natural Language Parsing*. New York: Cambridge U. Press.
- Cullingford, R. 1981. "SAM." In R. Schank and C. Riesbeck (eds.), *Inside Computer Understanding*. Hillsdale, NJ: Lawrence Erlbaum, 75–119.
- Dahlgren, K. 1988. *Naive Semantics for Natural Language Understanding*. Boston, MA: Kluwer Academic Publishers.
- Dale, R. 1992. *Generating Referring Expressions*. Cambridge, MA: MIT Press.
- Dale, R., C. Mellish, and M. Zock (eds.). 1990. *Current Research in Natural Language Generation*. New York: Academic Press.
- Dalrymple, M., S. Shieber, and F. Pereira. 1991. "Ellipsis and higher-order unification," *Linguistics and Philosophy* 14, 4.
- Davidson, D. 1967. "The logical form of action sentences." In N. Rescher (ed.), *The Logic of Decision and Action*. Pittsburgh, PA: U. Pittsburgh Press.
- Davis, E. 1990. *Representations of Commonsense Reasoning*. San Mateo, CA: Morgan Kaufmann.
- DeJong, G. 1979. "Prediction and substantiation: A new approach to natural language processing," *Cognitive Science* 3:251–273.
- DeJong, G. 1982. "An overview of the FRUMP system." In W. Lehnert and M. Ringle (eds.), *Strategies for Natural Language Processing*. Hillsdale, NJ: Lawrence Erlbaum, 149–176.
- Denes, P. 1993. *The Speech Chain (2nd ed.)*. San Francisco, CA: Freeman.
- DeRose, S. 1988. "Grammatical category disambiguation by statistical optimization," *Computational Linguistics* 14, 1:31–39.
- Devlin, K. 1991. *Logic and Information*. Cambridge, U.K.: Cambridge U. Press.
- Donnellan, K. 1966. "Reference and definite descriptions," *Philosophical Review* 75:281–304. Reprinted in S. Schwartz (ed.). 1977. *Naming, Necessity, and Natural Kinds*. Ithaca, NY: Cornell U. Press.
- Dowty, D. R. 1979. *Word Meaning and Montague Grammar*. Dordrecht: D. Reidel.
- Dowty, D. R. (ed.). 1986. "Tense and aspect in discourse," *Linguistics and Philosophy* 9, 1 (special issue).
- Dowty, D. R. 1989. "On the semantic content of the notion 'thematic role'." In G. Chierchia, B. Partee, and R. Turner (eds.), *Properties, Types and Meaning*. Vol. 2. Dordrecht: Kluwer Academic Publishers, 69–130.

- Dowty, D. R., L. Karttunen, and A. Zwicky (eds.). 1985. *Natural Language Parsing*. New York: Cambridge U. Press.
- Dowty, D. R., R. E. Wall, and S. Peters. 1981. *Introduction to Montague Semantics*. Dordrecht: D. Reidel.
- Dyer, M. 1983. *In-Depth Understanding: A Computer Model of Integrated Processing for Narrative Comprehension*. Cambridge, MA: MIT Press.
- Earley, J. 1970. "An efficient context-free parsing algorithm," *Commun. of the ACM* 13, 2:94–102. Reprinted in RNLP.
- Etherington, D., and R. Reiter. 1983. "On inheritance hierarchies with exceptions," *Proc. AAAI*, 104–108.
- Fano, R. 1961. *Transmission of Information*. Cambridge, MA: MIT Press.
- Ferguson, G. M., and J. F. Allen. 1993. "Generic plan recognition for dialogue," *Proc. ARPA Human Language Technology Workshop*, San Mateo, CA: Morgan Kaufmann.
- Fikes, R. E., and N. J. Nilsson. 1971. "STRIPS: A new approach to the application of theorem proving to problem solving," *Artificial Intelligence* 2, 3/4:189–208.
- Fillmore, C. J. 1968. "The case for case." In E. Bach and R. Harms (eds.), *Universals in Linguistic Theory*. New York: Holt, Rinehart, and Winston, 1–90.
- Fillmore, C. J. 1977. "The case for case reopened." In P. Cole and J. Sadock (eds.), *Syntax and Semantics*. Vol. 8: *Grammatical Relations*. New York: Academic Press, 59–81.
- Findler, N. 1979. *Associative Networks*. New York: Academic Press.
- Finin, T. 1980. "The semantic interpretation of nominal compounds," *Proc. AAAI*, 310–312.
- Ford, M., J. W. Bresnan, and R. M. Kaplan. 1982. "A competence based theory of syntactic closure." In J. W. Bresnan (ed.), *The Mental Representation of Grammatical Relations*. Cambridge, MA: MIT Press.
- Fox, B. 1987. *Discourse Structure and Anaphora: Written and Conversational English*. New York: Cambridge U. Press.
- Francis, W., and H. Kucera. 1982. *Frequency Analysis of English Usage: Lexicon and Grammar*. Boston: Houghton Mifflin.
- Freuder, E. C. 1982. "A sufficient condition for backtrack-free search," *JACM* 29, 1.
- Gazdar, G. 1982. "Phrase structure grammar." In P. Jacobson and G. K. Pullum (eds.), *The Nature of Syntactic Representation*. Dordrecht: D. Reidel, 131–186.
- Gazdar, G., and C. Mellish. 1989a. *Natural Language Processing in LISP*. Reading, MA: Addison-Wesley.
- Gazdar, G., and C. Mellish. 1989b. *Natural Language Processing in PROLOG*. Reading, MA: Addison-Wesley.
- Gazdar, G., E. Klein, G. K. Pullum, and I. Sag. 1985. *Generalized Phrase Structure Grammar*. Oxford: Basil Blackwell.
- Genesereth, M., and N. Nilsson. 1987. *Logical Foundations of Artificial Intelligence*. San Mateo, CA: Morgan Kaufmann.
- Goldman, A. 1970. *A Theory of Human Action*. Princeton, NJ: Princeton U. Press.
- Goodman, B. 1986. "Reference identification and reference identification failures," *Computational Linguistics* 12, 4.
- Grice, H. P. 1957. "Meaning," *Philosophical Review* 66, 377–388. Reprinted in D. Steinburg and L. Jakobovits (eds.). 1971. *Semantics*. New York: Cambridge U. Press.
- Grice, H. P. 1975. "Logic and conversation." In P. Cole and J. Morgan (eds.), *Syntax and*

- Semantics*. Vol. 3: *Speech Acts*. New York: Academic Press, 41–58.
- Grimes, J. E. 1975. *The Thread of Discourse*. The Hague: Moulton Press.
- Grishman, R., and J. Sterling. 1992. "Acquisition of selectional patterns," *Proc. 14th COLING*, 658–664.
- Grosz, B. J. 1974. "The structure of task oriented dialog," *IEEE Symp. on Speech Recognition*. Reprinted in L. Polanyi (ed.). 1986. *The Structure of Discourse*. Norwood, NJ: Ablex.
- Grosz, B. J. 1977. "The representation and use of focus in a system for understanding dialogs," *Proc. IJCAI*, 67–76. Reprinted in RNLP.
- Grosz, B. J., and C. Sidner. 1986. "Attention, intention, and the structure of discourse," *Computational Linguistics* 12, 3.
- Grosz, B. J., and C. Sidner. 1990. "Plans for discourse." In Cohen et al. (1990), 417–444.
- Grosz, B. J., A. K. Joshi, and S. Weinstein. 1983. "Providing a unified account of definite noun phrases in discourse," *Proc. ACL*, 44–50.
- Grosz, B. J., D. Appelt, P. Martin, and F. Pereira. 1987. "TEAM: An experiment in the design of transportable natural-language interfaces," *Artificial Intelligence* 32, 2:173–244.
- Haas, A. R. 1986. "A syntactic theory of belief and action," *Artificial Intelligence* 28, 3:245–292.
- Halliday, M. A. K. 1967. "Notes on transitivity and theme in English," *Journal of Linguistics* 3:199–244.
- Halliday, M. A. K. 1985. *A Short Introduction to Functional Grammar*. London: Arnold.
- Halliday, M. A. K., and R. Hasan. 1976. *Cohesion in English*. London: Longman.
- Hankamer, J., and I. Sag. 1976. "Deep and surface anaphora," *Linguistic Inquiry* 7, 3:391–426.
- Harper, M. 1992. "Ambiguous noun phrases in logical form," *Computational Linguistics* 18, 4, 419–466.
- Hayes, P. J. 1977. "On semantic nets, frames, and associations," *Proc. IJCAI*, 99–107.
- Hayes, P. J. 1979. "The logic of frames." In D. Metzger (ed.), *Frame Conceptions and Text Understanding*. New York: de Gruyter.
- Heim, I. 1982. "The semantics of definite and indefinite noun phrases," Ph.D. dissertation, U. Massachusetts.
- Hendrix, G. G., E. Sacerdoti, D. Sagalowicz, and J. Slocum. 1978. "Developing a natural language interface to complex data," *ACM Trans. on Database Systems* 3, 2:105–147.
- Herskovits, A. 1986. *Language and Spatial Cognition*. New York: Cambridge U. Press.
- Hewitt, C. 1971. "PLANNER: A language for proving theorems in robots," *Proc. IJCAI*.
- Hindle, D. 1989. "Acquiring disambiguation rules from text," *Proc. ACL*, 118–125.
- Hindle, D., and M. Rooth. 1993. "Structural ambiguity and lexical relations," *Computational Linguistics* 19, 1.
- Hinrichs, E. 1986. "Temporal anaphora in discourse of English," *Linguistics and Philosophy* 9, 1:63–82.
- Hirschberg, J., and D. J. Litman. 1993. "Empirical studies on the disambiguation of cue phrases," *Computational Linguistics* 19, 3, 501–530.
- Hintikka, J. 1969. "Semantics for propositional attitudes." In J. W. Davis, D. J. Hockney, and K. W. Wilson (eds.), *Philosophical Logic*. Dordrecht: D. Reidel. Also appeared in L. Linsky (ed.). 1971. *Reference and Modality*. New York: Oxford U. Press.
- Hirst, G. 1981a. *Anaphora in Natural Language Understanding*. Berlin: Springer-Verlag.

- Hirst, G. 1981b. "Discourse oriented anaphora resolution in natural language understanding: A review," *AJCL* 7, 2:85-98.
- Hirst, G. 1987. *Semantic Interpretation Against Ambiguity*. New York: Cambridge U. Press.
- Hobbs, J. R. 1978. "Resolving pronoun references," *Lingua* 44, B11-338. Reprinted in RNLP.
- Hobbs, J. R. 1979. "Coherence and co-reference," *Cognitive Science* 3, 1:67-82.
- Hobbs, J. R., and S. M. Shieber. 1987. "An algorithm for generating quantifier scopings," *Computational Linguistics* 13:1-2.
- Hobbs, J. R., M. Stickel, D. Appelt, and P. Martin. 1993. "Interpretation as abduction," *Artificial Intelligence* 63, 1-2:69-142.
- Hobbs, J. R., W. Croft, T. Davies, D. Edwards, and K. Laws. 1987. "Commonsense metaphysics and lexical semantics," *Computational Linguistics* 13, 3-4:241-250.
- Hovy, E. 1993. "Automated discourse generation using discourse structure relations," *Artificial Intelligence* 63, 1-2:341-386.
- Huddleston, R. 1988. *English Grammar: An Outline*. New York: Cambridge U. Press.
- Hwang, C. H., and L. K. Schubert. 1992. "Tense trees as the 'fine structure' of discourse," *Proc. 30th Annual Meeting, Assoc. for Computational Linguistics*, 232-240.
- Hwang, C. H., and L. K. Schubert. 1993a. "Episodic logic: A situational logic for natural language processing." In P. Aczel, D. Israel, Y. Katagiri, and S. Peters (eds.), *Situation Theory and its Applications*, Vol. 3, CSLI Series (Center for the Study of Language and Information, Stanford U.). Chicago: Chicago U. Press.
- Hwang, C. H., and L. K. Schubert. 1993b. "Meeting the interlocking needs of LF-computation, deindexing and inference: An organic approach to natural language understanding," *Proc. IJCAI*, 1297-1302.
- Jackendoff, R. S. 1972. *Semantic Interpretation in Generative Grammar*. Cambridge, MA: MIT Press.
- Jackendoff, R. S. 1990. *Semantic Structures*. Cambridge, MA: MIT Press.
- Jacobs, P., and L. Rau. 1990. "SCISOR: A system for extracting information from on-line news," *Commun. of the ACM* 33, 11, 88-97.
- Jelinek, F. 1990. "Self-organized language modeling for speech recognition." In A. Waibel and K. F. Lee (eds.), *Readings in Speech Recognition*. San Mateo, CA: Morgan Kaufmann, 450-506.
- Jensen, K., and J. Binot. 1987. "Disambiguating prepositional phrase attachments by using on-line dictionary definitions," *Computational Linguistics* 13:3-4.
- Johnson, M. 1991. "Features and formulae," *Computational Linguistics* 17, 2:131-153.
- Joshi, A. 1985. "Tree-adjoining grammars: How much context sensitivity is required to provide reasonable structural descriptions." In D. R. Dowty, L. Karttunen, and A. Zwicky (eds.), *Natural Language Parsing*. New York: Cambridge U. Press.
- Kamp, H. 1975. "Two theories about adjectives." In E. Keenan (ed.), *Formal Semantics of Natural Language*. Cambridge, U.K.: Cambridge U. Press.
- Kamp, H. 1981. "A theory of truth and semantic representation." In J. Groenendijk, T. Janssen, and M. Stokhof (eds.), *Formal Methods in the Study of Language*. Amsterdam: Amsterdam Press.
- Kaplan, R. M. 1973. "A general syntactic processor." In R. Rustin (ed.), *Natural Language Processing*. New York: Algorithmics Press.
- Kaplan, R. M., and J. Bresnan. 1982. "Lexical-functional grammar: A formal system for grammatical representation." In J. Bresnan (ed.), *The Mental Representation of Grammatical Relations*. Cambridge, MA: MIT Press.

- Karttunen, L. 1976. "Discourse referents." In J. McCawley (ed.), *Syntax and Semantics*. Vol. 7. New York: Academic Press, 363–386.
- Katz, J. J., and J. A. Fodor. 1963. "The structure of semantic theory," *Language* 39, 170–210. Reprinted in J. A. Fodor et al. (eds.). 1984. *The Structure of Language: Readings in the Philosophy of Language*. Englewood Cliffs, NJ: Prentice-Hall.
- Kautz, H. 1990. "A circumscriptive theory of plan recognition." In Cohen et al. (1990), 105–134.
- Kay, M. 1973. "The MIND system." In R. Rustin (ed.), *Natural Language Processing*. New York: Algorithmics Press, 155–188.
- Kay, M. 1980. "Algorithm schemata and data structures in syntactic processing," CSL-80-12, Xerox Corporation. Reprinted in RNLP.
- Kay, M. 1982. "Parsing in functional unification grammar." In D. R. Dowty, L. Karttunen, and A. Zwicky (eds.), *Natural Language Parsing*. New York: Cambridge U. Press, 251–278. Reprinted in RNLP.
- Kimball, J. 1973. "Seven principles of surface structure parsing in natural language," *Cognition* 2, 1:15–47.
- Klein, E. 1980. "A semantics for positive and comparative adjectives," *Linguistics and Philosophy* 4, 1–45.
- Knuth, D. E. 1968. "Semantics for context-free languages," *Mathematical Systems Theory* 2, 127–145.
- Konolige, K. 1985. "A computational theory of belief introspection," *Proc. IJCAI*, 502–508.
- Konolige, K. 1986. *A Deduction Model of Belief*. San Mateo, CA: Morgan Kaufmann.
- Koskenniemi, K. 1983. "Two-level model for morphological analysis," *Proc. IJCAI*, 683–685.
- Kripke, S. 1963. "Semantical consideration on modal logic," *Acta Philosophica Fennica* 16:83–94.
- Lascarides, A., and N. Asher. 1993. "Temporal interpretation, discourse relations, and common-sense entailment," *Linguistics and Philosophy* 16, 5.
- Lascarides, A., N. Asher, and J. Oberlander. 1992. "Inferring discourse relations in context," *Proc. ACL*, 1–8.
- Lee, K. 1990. "Context dependent phonetic hidden Markov models for continuous speech recognition," *IEEE Trans. on Acoustics, Speech and Signal Processing*. Reprinted in Waibel and Lee (1990).
- Leech, G. 1987. *Meaning and the English Verb (2nd ed.)*. London: Longman.
- Leech, G., and J. Svartvik. 1975. *A Communicative Grammar of English*. Singapore: Longman Singapore Publishers Ltd.
- Levesque, H. J. 1984. "A logic of implicit and explicit belief," *Proc. AAAI*.
- Levin, J., and J. Moore. 1977. "Dialogue-games: Metacommunication structures for natural language interaction," *Cognitive Science* 1, 4:395–421.
- Lewis, D. K. 1973. *Counterfactuals*. Oxford: Basil Blackwell.
- Lewis, D. K. 1979. "Scorekeeping in a language game." In R. Bäuerle, U. Egli, and A. von Stechow (eds.), *Semantics from Different Points of View*. Berlin: Springer-Verlag.
- Linde, C. 1979. "Focus of attention and the choice of pronouns in discourse." In T. Given (ed.), *Syntax and Semantics*. Vol. 12. New York: Academic Press.
- Litman, D. J., and J. F. Allen. 1987. "A plan recognition model for subdialogues in conversations," *Cognitive Science* 11, 2:163–200.

- Litman, D. J., and J. F. Allen. 1990. "Discourse processing and commonsense plans." In Cohen et al. (1990), 365–388.
- Luger, G., and W. Stubblefield. 1993. *Artificial Intelligence (2nd ed.)*. Redwood City, CA: Benjamin/Cummings.
- Lytinen, S. L. 1986. "Dynamically combining syntax and semantics in natural language processing," *Proc. AAAI*, 574–578.
- MacWorth, A. K. 1977. "Consistency in networks of relations," *Artificial Intelligence* 8, 1.
- Magerman, D., and C. Weir. 1992. "Efficiency, robustness and accuracy in picky chart parsing," *Proc. ACL*.
- Mann, W. C., and C. Mathiesson. 1985. "Nigel: A systemic grammar for text generation." In R. O. Freedle (ed.), *Systemic Perspectives on Discourse*. Norwood, NJ: Ablex.
- Mann, W. C., and S. Thompson. 1986. "Rhetorical structure theory: Description and construction of text structures." In G. Kempen (ed.), *Natural Language Generation*. Boston, MA: Kluwer Academic Publishers, 279–300.
- de Marcken, C. 1990. "Parsing the LOB corpus," *Proc. ACL*, 243–251.
- Marcus, M. 1980. *A Theory of Syntactic Recognition for Natural Language*. Cambridge, MA: MIT Press.
- Marcus, M., D. Hindle, and M. Fleck. 1983. "D-Theory: Talking about talking about trees," *Proc. ACL*.
- Marcus, M., B. Santorini, and M. Marcinkiewicz. 1993. "Building a large annotated corpus of English: The Penn Treebank," *Computational Linguistics* 19, 2.
- May, R. 1985. *Logical Form*. Cambridge, MA: MIT Press.
- McCarthy, J. 1980. "Circumscription: A form of non-monotonic reasoning," *Artificial Intelligence* 13:27–39.
- McCarthy, J. and P. J. Hayes. 1969. "Some philosophical problems from the standpoint of artificial intelligence." In B. Meltzer and D. Michie (eds.), *Machine Intelligence 4*. Edinburgh: Edinburgh U. Press.
- McCawley, J. D. 1993. *Everything That Linguists Have Always Wanted to Know about Logic (2nd ed.)*. Chicago: Chicago U. Press.
- McCord, M. C. 1980. "Slot grammars," *AJCL* 6, 1:31–43.
- McCord, M. C. 1986. "Focalizers, the scoping problem, and semantic interpretation rules in logic grammars." In D. H. Warren and M. van Canegham (eds.), *Logic Programming and its Applications*. Norwood, NJ: Ablex.
- McKeown, K. R. 1985. *Text Generation*. New York: Cambridge U. Press.
- Mellish, C. 1985. *Computer Interpretation of Natural Language Descriptions*. New York: Wiley.
- Miller, G. 1990. "WordNet: An online lexical database," *International Journal of Lexicography* 3, 4.
- Minsky, M. 1975. "A framework for representing knowledge." In P. H. Winston (ed.), *The Psychology of Computer Vision*. New York: McGraw-Hill, 211–277.
- Montague, R. 1974. *Formal Philosophy*. New Haven: Yale U. Press.
- Moore, J., and M. Pollack. 1992. "A problem for RST: The need for multi-level discourse analysis," *Computational Linguistics* 18, 4:537–544.
- Moore, R. C. 1973. "D-SCRIPT: A computational theory of description," *Proc. IJCAI*, 223–229.
- Moore, R. C. 1977. "Reasoning about knowledge and action," *Proc. IJCAI*, 223–227. Extended

- version in J. R. Hobbs and J. Moore (eds.). 1985. *Formal Theories of the Common Sense World*. Vol. 1. Norwood, NJ: Ablex.
- Moore, R. C. 1981. "Problems in logical form," *Proc. 19th ACL*, 117–124. Reprinted in RNLP.
- Moore, R. C. 1989. "Unification-based semantic interpretation," *Proc. ACL*.
- Nebel, B., C. Rich, and W. Swartout (eds.) 1992. *Proc. 3rd Int'l. Conf. on Principles of Knowledge Representation and Reasoning*. San Mateo, CA: Morgan Kaufmann.
- Norvig, P. 1992. *Paradigms of Artificial Intelligence Programming*. San Mateo, CA: Morgan Kaufmann.
- Palmer, M., R. Passonneau, C. Wier, and T. Finin. 1993. "The KERNEL text understanding system," *Artificial Intelligence* 63:17–68.
- Parsons, T. 1990. *Events in the Semantics of English*. Cambridge, MA: MIT Press.
- Partee, B. 1984. "Nominal and temporal anaphora," *Linguistics and Philosophy* 7:243–286.
- Partee, B., A. ter Meulen, and R. Wall. 1993. *Mathematical Methods in Linguistics (corrected 1st ed.)*. Boston: Kluwer Academic Publishers.
- Patten, T. 1988. *Systemic Text Generation as Problem Solving*. Cambridge U. Press.
- Paxton, W., and A. Robinson. 1973. "A parser for a speech understanding system," *Proc. IJCAI*.
- Pelletier, J. 1979. *Mass Terms: Some Philosophical Problems*. Dordrecht: D. Reidel.
- Pereira, F. C. N. 1981. "Extraposition grammars," *AJCL* 7, 4:243–256.
- Pereira, F. C. N. 1983. "Logic for natural language analysis," SRI Technical Note 275, SRI International, Menlo Park, California.
- Pereira, F. C. N. 1985. "Characterization of attachment preferences." In D. R. Dowty, L. Karttunen, and A. Zwicky (eds.), *Natural Language Parsing*. New York: Cambridge U. Press, 307–319.
- Pereira, F. C. N., and M. Pollack. 1991. "Incremental interpretation," *Artificial Intelligence* 50:37–82.
- Pereira, F. C. N., and Y. Schabes. 1992. "Inside-outside re-estimation from partially bracketed corpora," *Proc. ACL*, 128–135.
- Pereira, F. C. N., and S. M. Shieber. 1987. *Prolog and Natural Language Analysis*. Chicago: Chicago U. Press.
- Pereira, F. C. N., and D. H. D. Warren. 1980. "Definite clause grammars for language analysis—A survey of the formalism and a comparison with augmented transition networks," *Artificial Intelligence* 13, 3:231–278. Reprinted in RNLP.
- Perrault, C. R. 1984. "On the mathematical properties of linguistic theories," *Computational Linguistics* 10, 3–4:165–176. Reprinted in RNLP.
- Perrault, C. R. 1990. "An application of default logic to speech act theory." In Cohen et al. (1990), 161–186.
- Perrault, C. R., and J. F. Allen. 1980. "A plan-based analysis of indirect speech acts," *AJCL* 6, 3–4:167–182.
- Perrault, C. R., and P. R. Cohen. 1981. "It's for your own good: A note on inaccurate reference." In A. Joshi, B. Webber, and I. Sag (eds.), *Elements of Discourse Understanding*. New York: Cambridge U. Press.
- Perrault, C. R., and B. J. Grosz. 1986. "Natural language interfaces," *Annual Review of Computer Science* 1:47–82.
- Pierrehumbert, J., and J. Hirschberg. 1990. "The meaning of intonational contours in the interpretation of discourse." In Cohen et al. (1990), 271–312.

- Pollack, J., and D. Waltz. 1985. "Massively parallel parsing: A strongly interactive model of natural language interpretation," *Cognitive Science* 9:51-74.
- Pollack, M. 1990. "Plans as complex mental attitudes." In Cohen et al. (1990), 77-104.
- Pollard, C., and I. Sag. 1987. *Information-Based Syntax and Semantics*. Vol. I: *Fundamentals*. CSLI Lecture Notes 13. Chicago: Chicago U. Press.
- Pollard, C., and I. Sag. 1993. *Head-Driven Phrase Structure Grammar*. Chicago, IL: Chicago U. Press.
- Prince, E. 1981. "Towards a taxonomy of given-new information." In P. Cole (ed.), *Radical Pragmatics*. New York: Academic Press, 236-256.
- Prior, A. N. 1967. *Past, Present, and Future*. Oxford: Oxford U. Press.
- Pustejovsky, J. 1991. "The syntax of event structures," *Cognition* 41:47-81.
- Quillian, M. R. 1968. "Semantic memory." In M. Minsky (ed.), *Semantic Information Processing*. Cambridge, MA: MIT Press.
- Quirk, R., S. Greenbaum, G. Leech, and J. Svartik. 1972. *A Grammar of Contemporary English*. New York: Seminar Press.
- Rabiner, L. R. 1989. "A tutorial on hidden Markov models and selected applications in speech recognition," *Proc. IEEE*. Reprinted in Waibel and Lee (1990).
- Rabiner, L. R., and B.-H. Juang. 1993. *Fundamentals of Speech Recognition*. Englewoods Cliffs, NJ: Prentice-Hall, Inc.
- Radford, A. 1981. *Transformational Syntax*. New York: Cambridge U. Press.
- Rao, A., and M. Georgeff. 1991. "Modeling rational agents within a BDI architecture." In J. F. Allen, R. Fikes, and E. Sandewall (eds.), *Proc. 2nd Conf. on Principles of Knowledge Representation and Reasoning*. San Mateo, CA: Morgan Kaufmann, 473-484.
- Reichenbach, H. 1947. *Elements of Symbolic Logic*. New York: Macmillan.
- Reichman, R. 1978. "Conversational coherency," *Cognitive Science* 2, 4:283-328.
- Reichman, R. 1985. *Getting Computers to Talk Like You and Me*. Cambridge, MA: MIT Press.
- Reinhart, T. 1983. "Co-reference and bound anaphora: A restatement of the anaphora question," *Linguistics and Philosophy* 6:47-88.
- Reiter, R. 1980. "A logic for default reasoning," *Artificial Intelligence* 13:81-132.
- Resnik, P. 1993. "Semantic classes and syntactic ambiguity," *Proc. ARPA Human Language Technology Workshop*, San Mateo, CA: Morgan Kaufmann.
- Rich, E., and K. Knight. 1992. *Artificial Intelligence (2nd ed.)*. McGraw-Hill.
- Riesbeck, C., and R. C. Schank. 1978. "Comprehension by computer." In W. Levelt and G. B. Flores d'Arcais (eds.), *Studies in the Perception of Language*. Chichester, U.K.: Wiley.
- Ritchie, G. D. 1980. *Computational Grammar*. New York: Barnes and Noble.
- Ritchie, G. D., G. Russell, A. Black, and S. Pulman. 1992. *Computational Morphology*. Cambridge, MA: MIT Press.
- Robinson, J. A. 1965. "A machine-oriented logic based on the resolution principle," *Journal of the ACM* 12, 1.
- Robinson, J. J. 1982. "DIAGRAM: A grammar for dialogues," *Commun. of the ACM* 25, 1:27-47. Reprinted in RNLP.
- Rosenschein, S. J., and S. M. Shieber. 1982. "Translating English into logical form," *Proc. ACL*, 1-8.
- Ross, S. 1988. *A First Course in Probability (3rd ed.)*. New York: Macmillan.

- Rounds, W. C. 1988. "LFP: A logic for linguistic description and an analysis of its complexity," *Computational Linguistics* 14, 4:1-10.
- Rumelhart, D. E. 1975. "Notes on schema for stories." In D. Bobrow and A. Collins (eds.), *Representation and Understanding*. New York: Academic Press.
- Rumelhart, D. E., and J. McClelland. 1986. *Parallel Distributed Processing*. Cambridge, MA: MIT Press.
- Russell, B., and A. N. Whitehead. 1925. *Principia Mathematica* (2nd ed.). Vol. 1. New York: Cambridge U. Press.
- Sacerdoti, E. 1977. *A Structure for Plans and Behavior*. New York: Elsevier North-Holland.
- Sager, N. 1981. *Natural Language Information Processing: A Computer Grammar of English and its Applications*. Reading, MA: Addison-Wesley.
- Saint-Dizier, P. 1985. "Handling quantifier scoping ambiguities in a semantic representation of natural language sentences." In V. Dahl and P. Saint-Dizier (eds.), *Natural Language Understanding and Logic Programming*. Amsterdam: North-Holland.
- Saint-Dizier, P. 1986. "An approach to natural language semantics in logic programming," *Journal of Logic Programming* 3, 4:329-356.
- Scha, R., and L. Polanyi. 1988. "An augmented context free grammar for discourse," *Proc. COLING*.
- Schafer, R. W., and L. R. Rabiner. 1975. "Digital representations of speech," *Proc. IEEE* 63, 4:662-667. Reprinted in Waibel and Lee (1990).
- Schank, R. C. (ed.). 1975. *Conceptual Information Processing*. Amsterdam: North-Holland.
- Schank, R. C., and R. Abelson. 1977. *Scripts, Plans, Goals and Understanding*. Hillsdale, NJ: Lawrence Erlbaum.
- Schank, R. C., and K. M. Colby. 1973. *Computer Models of Thought and Language*. San Francisco, CA: Freeman.
- Schank, R. C., and C. J. Rieger. 1974. "Inference and the computer understanding of natural language," *Artificial Intelligence* 5:373-412.
- Schank, R. C., and C. K. Riesbeck. 1981. *Inside Computer Understanding*. Hillsdale, NJ: Lawrence Erlbaum.
- Schiffer, S. R. 1972. *Meaning*. London: Oxford U. Press.
- Schmidt, C., N. Sridharan, and J. Goodson. 1978. "The plan recognition problem: An intersection of psychology and artificial intelligence," *Artificial Intelligence* 11:45-83.
- Schubert, L. K. 1986. "Are there preference tradeoffs in attachment decision?," *Proc. AAAI*, 601-605.
- Schubert, L. K., and F. J. Pelletier. 1982. "From English to logic: Context-free computation of conventional logical translation," *AJCL* 8, 1:165-176. Reprinted in RNLP.
- Schubert, L. K., and F. J. Pelletier. 1987. "Problems in the representation of the logical form of generics, plurals, and mass nouns." In E. LePore (ed.), *New Directions in Semantics*. New York: Academic Press, 385-451.
- Searle, J. R. 1975. "Indirect speech acts." In P. Cole and J. Morgan (eds.), *Syntax and Semantics*. Vol. 3: *Speech Acts*. New York: Academic Press, 59-82.
- Searle, J. R. 1979. "A taxonomy of speech acts." In J. Searle (ed.), *Expression and Meaning*. Cambridge, U.K.: Cambridge U. Press, 1-29.
- Sells, P. 1985. *Lectures on Contemporary Syntactic Theories*. Chicago: Chicago U. Press.
- Shapiro, S. C. 1982. "Generalized augmented transition network grammars for generation from semantic networks," *AJCL* 8, 1:12-25.

- Shapiro, S. C. (ed.). 1992. *Encyclopedia of Artificial Intelligence* (2nd ed.). New York: Wiley.
- Shieber, S. M. 1984. "The design of a computer language for linguistic information," *Proc. COLING*, 362-366.
- Shieber, S. M. 1986. *An Introduction to Unification-Based Approaches to Grammar*. CSLI Lecture Notes 4. Chicago: Chicago U. Press.
- Shieber, S. M., G. van Noord, F. C. N. Pereira, and R. C. Moore. 1990. "Semantic-head-driven generation," *Computational Linguistics* 16, 1:30-42.
- Sidner, C. 1983. "Focusing in the comprehension of definite anaphora." In M. Brady and R. C. Berwick (eds.), *Computational Models of Discourse*. Cambridge, MA: MIT Press, 267-330. Reprinted in RNLP.
- Sidner, C. 1985. "Plan parsing for intended response recognition in discourse," *Computational Intelligence* 1, 1:1-10.
- Slocum, J. 1985. "A survey of machine translation," *Comput'l. Linguistics* 11, 1:1-17.
- Small, S. L., and C. Rieger. 1982. "Parsing and comprehending with word experts." In W. Lehnert and M. Ringle (eds.), *Strategies for Natural Language Processing*. Hillsdale, NJ: Lawrence Erlbaum.
- Song, F., and R. Cohen. 1991. "Tense interpretation in the context of narrative," *Proc. AAAI*, 131-136.
- Sowa, J. 1984. *Conceptual Structures*. Reading, MA: Addison-Wesley.
- Sowa, J. (ed.). 1991. *Principles of Semantic Networks*. San Mateo, CA: Morgan Kaufmann.
- Sproat, R. 1992. *Morphology and Computation*. Cambridge, MA: MIT Press.
- Stabler, E. 1992. *The Logical Approach to Syntax*. Cambridge, MA: MIT Press.
- Stalnaker, R. 1974. "Pragmatic presuppositions." In M. Munitz and P. Unger (eds.), *Semantics and Philosophy*. New York: New York University Press.
- Steedman, M. 1982. "Reference to past time." In R. J. Jarvella and W. Klein (eds.), *Speech, Place and Action*. New York: Wiley.
- Steedman, M. 1987. "Combinatory grammars and parasitic gaps," *Natural Language and Linguistic Theory* 5, 403-439.
- Strawson, P. F. 1964. "Intention and convention in speech acts," *Philosophical Review* 73, 4:439-460. Reprinted in J. R. Searle (ed.). 1971. *The Philosophy of Language*. New York: Oxford U. Press.
- Tarski, A. 1944. "The semantic conception of truth and the foundations of semantics," *Philosophy and Phenomenological Research* 4:341-375.
- Tedeschi, P and A. Zaenen (eds.). 1981. *Syntax and Semantics*. Vol. 14: *Tense and Aspect*. New York: Academic Press.
- Thomason, R. 1970. *Symbolic Logic*. New York: Macmillan.
- Tomita, M. 1986. *Efficient Parsing for Natural Language*. Boston: Kluwer Academic Publishers.
- Traum, D. R., and E. A. Hinkelman. 1992. "Conversation acts in task-oriented spoken dialogue," *Computational Intelligence* 8, 3 (special issue on computational approaches to non-literal language).
- VanLehn, K. 1978. "Determining the scope of English quantifiers," TR AI-TR-483, AI Lab, Massachusetts Inst. of Technology.
- Vendler, Z. 1967. *Linguistics in Philosophy*. Ithaca, NY: Cornell U. Press.
- Vijay-Shankar, K. 1992. "Using descriptions of trees in tree adjoining grammar," *Computational Linguistics* 18, 4.

- Viterbi, A. J. 1967. "Error bounds for convolution codes and an asymptotically optimal decoding algorithm," *IEEE Trans. on Information Theory* 13:260-269.
- Waibel, A., and K. F. Lee (eds.). 1990. *Readings in Speech Recognition*. San Mateo, CA: Morgan Kaufmann.
- Walker, M. 1989. "Evaluating discourse processing algorithms," *Proc. ACL*, 251-261.
- Waltz, D. L. 1975. "Understanding line drawings of scenes with shadows." In P. H. Winston (ed.), *Psychology of Computer Vision*. Cambridge, MA: MIT Press.
- Wang, M., and J. Hirschberg. 1992. "Automatic classification of intonational phrase boundaries," *Speech and Language* 6:175-196.
- Warren, D. H. D., and F. C. N. Pereira. 1982. "An efficient easily adaptable system for interpreting natural language queries," *Computational Linguistics* 8, 3-4:110-122.
- Webber, B. L. 1983. "So what can we talk about now." In M. Brady and B. Berwick (eds.), *Computational Models of Discourse*. Cambridge, MA: MIT Press, 331-370. Reprinted in RNLP.
- Webber, B. L. 1988. "Tense as discourse anaphora," *Comput'l. Linguistics* 14, 2:61-73.
- Webber, B. L. 1991. "Structure and ostension in the interpretation of discourse deixis," *Language and Cognitive Processes* 6:107-135.
- Webber, B. L. 1992. "Question answering." In S. C. Shapiro (ed.), *Encyclopedia of Artificial Intelligence*. New York: Wiley, 814-822.
- Weischedel, R. M. 1979. "A new semantic computation while parsing: Presupposition and entailment." In C. Oh and D. Dineen (eds.), *Syntax and Semantics*. Vol. II: *Presupposition*. New York: Academic Press, 155-182. Reprinted in RNLP.
- Weischedel, R. M., M. Meteer, R. Schwartz, L. Ramshaw, and J. Palmucci. 1993. "Coping with ambiguity and unknown words through probabilistic models," *Computational Linguistics* 19, 2 (special issue on using corpora).
- Weizenbaum, J. 1966. "ELIZA," *Commun. of the ACM* 9:36-45.
- Whittemore, G., K. Ferrara, and H. Bruner. 1990. "Empirical study of predictive powers of simple prepositional attachment schemes for post-modifier prepositional phrases," *Proc. ACL*, 23-30.
- Wilensky, R. 1983. *Planning and Understanding*. Reading, MA: Addison-Wesley.
- Wilensky, R. 1986. *Common LISPcraft*. New York: W. W. Norton.
- Wilks, Y. 1975. "An intelligent analyzer and understander of English," *Commun. of the ACM* 18, 5:264-274. Reprinted in RNLP.
- Winograd, T. 1972. *Understanding Natural Language*. New York: Academic Press.
- Winograd, T. 1973. "A procedural model of language understanding." In R. C. Schank and K. M. Colby (eds.), *Computer Models of Thought and Language*. San Francisco, CA: Freeman, 152-186.
- Winograd, T. 1983. *Language as a Cognitive Process*. Vol. 1: *Syntax*. Reading, MA: Addison-Wesley.
- Winston, P. H. 1992. *Artificial Intelligence (3rd ed.)*. Reading, MA: Addison-Wesley.
- Winston, P. H., and B. K. P. Horn. 1989. *LISP (3rd ed.)*. Reading, MA: Addison-Wesley.
- Woods, W. A. 1970. "Transition network grammars for natural language analysis," *Commun. of the ACM* 13:591-606. Reprinted in RNLP.
- Woods, W. A. 1973. "An experimental parsing system for transition network grammars." In R. Rustin (ed.), *Natural Language Processing*. New York: Algorithmics Press.

-
- Woods, W. A. 1975. "What's in a link: Foundations for semantic networks." In D. G. Bobrow and A. Collins (eds.), *Representation and Understanding: Studies in Cognitive Science*. New York: Academic Press.
- Woods, W. A. 1977. "Lunar rocks in natural English: Explorations in natural language question answering." In A. Zampoli (ed.), *Linguistic Structures Processing*. New York: Elsevier North-Holland.
- Woods, W. A. 1978. "Semantics and quantification in natural language question answering." In M. Yovitz (ed.), *Advances in Computers*. Vol. 17. New York: Academic Press. Reprinted in RNLP.
- Woods, W. A. 1980. "Cascaded ATN grammars," *AJCL* 6, 1:1-12.
- Yarowsky, D. 1992. "Word-sense disambiguation using statistical models of Roget's categories trained on large corpora," *Proc. COLING*, 454-460.

索引

- * , in augmented transition networks * , 扩充转移网络中的 4.6
- A over A Constraint A 上 A 约束 5.3.1
- abbreviation conventions, for CFGs 缩写约定, CFG 的 4.4
- abduction 溯因 15.2.4
- abductive inference 溯因推理 13.1.1
- Abelson, R. 15.10
- accommodation 适应性 14.4.1
- accomplishment propositions 成就性命题 13.5
- achievement propositions 达成性命题 13.5
- acting 行为 17.1
- action generation 行为生成 17.5
- action library, for plan recognition 行为库, 规划识别的 15.6
- actions, in augmented transition netw 操作, 扩充转移网络中的 4.6
- active arcs 活动边 3.4
- activity propositions 活动性命题 13.5
- adjective phrase 形容词短语 2.5
- adjectives 形容词 2.1, 2.2
 - semantic interpretatio ……的语义解释 12.5
- adjoining, in TAG 邻接, 树邻接语法中的 5.8
- adverb 副词 2.5
 - preposing 前置 5.2
- agenda, in chart parsing 待处理表, chart 句法分析器中的 3.4
- AGENT role AGENT 角色 8.6
- AGR feature AGR 特征 4.2.1
- Aho, A. 3.10, 6.7
- Allen, J. 13.10, 13.10, 15.10, 16.9, 17.12
- allophones 音位变体 C.2
- Alshawi, H. 4.10, 5.8, 6.7, 8.10, 9.9, 12.9
- ambiguity 歧义 8.2
 - and vagueness ……与模糊性 8.2
 - in the logical form 逻辑形式中的 8.4
 - semantic 语义的 8.2
- analog-to-digital conversion 模数转换 C.3
- anaphor 回指对象 12.4
- ancestor, of a node 祖先, 节点的 3.1
- antecedent, of a pronoun 先行词, 代词的 12.4
- Appelt, D. 11.6, 17.12
- arc extension algorithm 边扩展算法 3.4
 - with features 带特征的 4.5
- architecture of natural language syst 自然语言系统的结构 1.6
- Aristotle 亚里士多德 8.2, A.1
- article 冠词 2.2
- Asher, N. 16.9
- ASSERT act ASSERT 行为 8.7
- AT-LOC role AT-LOC 角色 8.6
- ATIS C.5
- ATN 4.6, 13.7
- atoms, in LISP 原子, LISP 语言中的 B.1
- attachment ambiguity 附着歧义 6.4
 - and probabilistic parsing ……与概率句法分 7.7
- attentional stack 关注栈 16.2
- attentiveness constraint 专心性约束 17.8
- augmented grammars 扩充语法 4.1
- augmented transition network 扩充转移网络 4.6
- Austin, J. 17.12
- automated deduction 自动演绎 13.6
- auxiliary verb 助动词 2.3
 - grammar for ……的语法 5.1
- Ayuso, D. 14.9
- Bach, E. 13.10
- backtracking 回溯 3.3
- backtracking search 回溯搜索 B.5
- backup states, in a top-down parser 后备状态, 自顶向下句法分析器中的 3.3
- backward probability 后向概率 7.4
- backward-looking center 回顾式中心 14.3
- Baker, C. 2.8, 5.2
- Ballard, B. 12.9
- Bar-Hillel, Y. 10.8
- Barwise, J. 8.10, A.4
- base, value of VFORM feature 原型, VFORM 特征的值 4.2.2
- Bates, L. 4.10, 5.8, 14.8

- Baum, L. 7.9
- Bayes' rule 贝叶斯法则 7.1
- BDI models BDI 模型 17.1
- belief 信念 17.1
- explicit vs. implicit 明确……与隐含……的对比 17.3
- representing in a knowledge base 知识库中的……表示 17.3
- sentential models of ……的宣告性模型 17.3
- BENEFICIARY role BENEFICIARY 角色 8.6
- Berwick, R. 12.9
- best-first parsing 最佳优先句法分析 7.6
- best-first search 最佳优先搜索 B.4
- bidirectional grammar 双向语法 1.6
- bigram 二元语法 7.3
- binary features 二元特征 4.3
- Binot, J. 10.8
- Birnbaum, L. 11.6
- black box evaluation 黑箱评价 1.3
- Blank, G. 3.10
- Bobrow, D. 13.10
- Bobrow, R. 11.6
- BORIS system BORIS 系统 10.8
- bound variable interpretation 绑定变量解释 12.4.1
- Brachman, R. 13.10
- Bratman, M. 17.12
- breadth-first search 广度优先搜索 3.3.2, B.3
- Brennan, S. 14.9
- Bresnan, J. 5.8, 6.7, 10.8, 11.6
- Brown, J. 11.6
- Bruce, B. 17.12
- Burton, R. 11.6
- C-command C-统制 12.4
- canonical interpretation, in parsing 经典解释, 句法分析中 6.5
- Carberry, S. 15.10, 17.12
- cardinal 基数词 2.2
- Carlson, G. 12.9
- cascaded ATN 层叠 ATN 11.6
- case 格 2.2
- case grammar 格语法 8.5
- causality 因果关系 15.4, 15.4
- center 中心 14.3
- CFG 3.1
- Chafe, W. 14.9
- Charniak, E. 3.10, 6.7, 7.9, 8.10
- chart 3.4
- chart parsing, active arcs chart 句法分析, 活动边 3.4
- agenda 待处理表 3.4
- bottom-up 自底向上的 3.4, 3.4
- dotted rules 点号规则 3.4.1
- efficiency considerations 效率方面的考虑 3.4
- top-down 自顶向下的 3.6
- charts, packed chart, 压缩的 6.4
- CHAT-80 13.10
- Chierchia, G. 8.10, 9.9, 12.9, 14.9
- child node 子节点 3.1
- Chitrao, M. 7.9
- Chomsky Hierarchy 乔姆斯基层次体系 3.2
- Chomsky, N. 3.2, 3.10, 5.8, 12.9
- Church, K. 6.7, 7.9
- circumscription 界限 13.1.1
- Clark, H. 14.9, 17.12
- Clark, K. 13.10
- clause, in logic 子句, 逻辑中的 13.6
- Clifton, C. 6.7
- closed class word 封闭类词语 2.1
- closed world assumption 封闭世界假设 13.1.1
- clue words 提示语 16.9
- CO-AGENT role CO-AGENT 角色 8.6
- co-articulation effects 协同发音 C.4
- co-reference 互指 12.4
- co-referring expressions 互指表达式 14.1
- CO-THEME role CO-THEME 角色 8.6
- codebook, for speech 密码本, 语音的 C.4
- Cohen, P. 17.12
- Cohen, R. 16.9
- coherence relations 一致关系 16.5
- coherence, and discourse 连贯性, ……和篇章 15.1
- collective interpretation of noun phr 名词短语的整体解释 14.1
- collective reading 整体性解读 8.3, 12.1.1
- of noun phras 名词短语的 13.4
- collocations 搭配 10.4
- Colmerauer, A. 4.10
- COMMAND act COMMAND 行为 8.7
- commitment 落实 17.1
- communicative acts 交流行为 17.5
- comparative adjective 比较级形容词 12.5.1
- complement 补足语 2.2
- completeness, in logic 完备性 A.2.1
- Complex-NP Constraint 复杂 NP 约束 5.3.1

- compositionality, in semantic interpr 组合理论, 语义解释中的 9.1
- computational linguistics 计算语言学 1.1
- conceptual dependency 概念依存 11.4, 13.3
- conditional probability 条件概率 7.1
- conjunctive normal form 合取范式 13.6
- consistency, of formulas 相容性, 公式的 A.2.1
- constants, in logic 常量, 逻辑中的 A.1
- in the logical form 逻辑形式中的 8.3
- constituent 语法成分 3.2
- constrained variables 约束变量 4.1
- constraint satisfaction 约束满足 10.1
- algorithm算法 10.1
- context spaces 上下文空间 16.9
- context-dependent probabilistic pars 上下文相关概率文法分析 7.7
- context-free grammar 上下文无关文法 3.1
- context-sensitive grammar 上下文相关文法 3.2
- contextual interpretation 上下文解释 8.1
- continuous speech recognition 连续语音识别 C.1
- contradiction 矛盾 8.8.1
- in logic 逻辑中的 A.1.1
- conversational agent 会话 agent 17.1
- conversational maxims 会话准则 17.8
- Cooper, R. 8.10
- Coordinate Structure Constraint 并列结构约束 5.3.1
- co-reference 互指 12.4
- co-referring expressions 互指表达式 14.1
- corpora 语料库 7.1
- CO-THEME role CO-THEME 角色 8.6
- Cottrell, G. 10.8
- count noun 可数名词 2.2
- counterfactual 反事实 15.4
- Crain, S. 10.8
- cross-validation 交叉验证 7.2.1
- cue phrases 提示语 16.1, 16.5
- and discourse struct和篇章结构 16.6.3
- culmination point 顶点 13.5
- current state, in a top-down parser 当前状态, 自顶向下句法分析器中的 3.3.1
- Dahlgren, K. 10.8
- Dale, R. 9.9
- Dalrymple, M. Dalrymple, M. 14.9
- database query 数据库查询 12.3.1
- systems系统 13.7
- Davidson, D. 8.5, 8.10
- Davis, E. 13.10
- de Bruin, J. 12.9
- de Marcken, C. 7.6
- declarative vs. procedural inference 描述性与程序性推理的比较 13.1.2
- decomposition chaining 分解链 15.6
- decomposition, of actions 分解, 行为的 13.3.1, 15.4
- deductive inference 演绎推理 13.1.1
- deep anaphora 深层回指 14.7
- deep structure 深层结构 5.2
- default rule 默认规则 13.1
- default value, of features 默认值, 特征的 4.2.4
- defeasible inference 可推翻的推理 13.1.1
- definite descriptions 确定性描述 14.4
- and generation和生成 14.5
- and scoping和辖域指定 12.2
- definite reference 定指 12.1
- to sets 集合的 14.5
- DeJong, G. 11.6, 17.12
- demonstrative 指示词 2.2
- Denes, P. C.7
- depth-first search 深度优先搜索 3.3.2, B.3
- derivational form 派生式 2.1
- DeRose, S. 7.9
- desire 期望 17.1, 17.4
- determiner 限定词 2.2
- deterministic parsing 确定型句法分析 6.3
- Devlin, K. 8.10
- dialogue-based applications 基于对话的应用系统 1.2
- digression, in discourse 题外话, 篇章中的 16.6.1
- direct object 直接宾语 2.3.1
- disambiguation, statistical 消歧, 统计 10.4
- discourse 篇章 1.4
- discourse anaphora 篇章回指 12.4
- discourse entities, and universal qua 篇章实体,和全称量词 14.1.1
- discourse entity list 篇章实体列表 14.1
- discourse focus 篇章焦点 14.3
- discourse purpose 篇章目的 16.4
- discourse representation theory 篇章表示理论 14.9
- discourse segments 篇章片段 16.1
- different views of的不同解释 16.2
- discourse stack 篇章栈 16.2
- discourse structure, and inference 篇章结构,和推理 16.4
- and reference和指代 16.3

- and tense and aspect和时态与体态 16.5
- discourse-level planning 篇章层次规划 17.10
- distributive interpretation of NPs 名词短语的个体解释 14.1.1
- distributive reading 个体性解读 8.3, 12.1.1
- of noun phrases 名词短语的 13.4
- domain of interpretation 解释论域 8.8
- dominating constituent 支配成分 12.1.2
- domination relation, between segments 支配关系, 片段之间 16.4
- domination, between nodes 管辖, 节点间的 3.1
- Donellan, K. 14.9
- dotted rules, in a chart parser 点号规则, chart 句法分析器中的..... 3.4
- Dowty, D. 8.10, 9.9, 13.10
- D-theory D 理论 6.4
- Dyer, M. 10.8
- Earley algorithm Earley 算法 3.10
- Earley, J. 3.10
- effect causality 结果因果关系 15.4
- effects, of actions 结果, 行为的 13.3.1
- ELE, 参见 expected likelihood estimator 7.2
- ELIZA 1.3
- ellipsis 省略 14.1, 14.6
- empty list 空表 B.1
- enablement, of actions 使能, 行为的 15.4
- EN-PASTPRT feature EN-PASTPRT 特征 4.3
- entailment 蕴涵 8.8.1, 13.1
- in logic 逻辑中的 A.2
- Etchemendy, J. 8.10, A.4
- Etherington, D. 13.10
- evaluation of natural language systems 自然语言系统的评价 1.3
- events, in the logical form 事件, 逻辑形式中的 8.4
- evoked objects, in a sentence 引发的对象, 一个句子中的 14.1
- existential node, in a semantic network 存在节点, 语义网络中的 10.3
- existential quantifiers 存在量词 12.1.1
- in logic 逻辑中的 A.1
- existential readings of definite descriptions 确定性描述的存在性解读 14.4.1
- expansion, in discourse 详细阐述, 篇章中的 16.6.1
- expectation 期望 15.2
- expectation matching 期望匹配 15.2
- and abduction和溯因推理 15.2.4
- with equality assumptions等价性假设 15.2.3
- expectation nodes, in plan inference 期望节点, 规划推理中的 15.7.1
- expectations, from causal reasoning 期望, 来自于因果推理的 15.4
- expected likelihood estimator 期望似然估计 7.2
- EXPERIENCER role EXPERIENCER 角色 8.6
- explicit beliefs 明确信念 17.3
- explicit performatives 显性施为句 17.2
- extensional meaning 外延意义 8.8.2
- extraposition 外置转换 5.2
- extraposition grammar 外置转换语法 5.6
- failure of substitutivity in modal contexts 情态上下文中的替换失效 8.3
- Fano, R. 10.8
- Fast Fourier Transform 快速傅里叶变换 C.2
- FASTUS system FASTUS 系统 11.6
- feature propagation in GPSG GPSG 中的特征传播 5.4
- feature structure 特征结构 4.1
- formalizing的形式化 4.1
- feature-based semantic interpretation 基于特征的语义解释 9.6
- features 特征 4.1
- AGR 4.2.1
- binary 二元 4.2.3
- default values的默认值 4.2.4
- EN-PASTPRT 4.3
- GAP 5.3
- heak 中心 4.4
- in augmented transition networks 扩充转移网络中的 4.6
- INV 4.2.3
- IRREG-PAST 4.3
- IRREG-PRES 4.3
- number 数 4.2.1
- person 人称 4.2.1
- PFORM 4.2.2
- SUBCAT 4.2.2
- VFORM 4.2.2
- WH 5.3
- Ferguson, G. 15.10
- FIFO 3.3.2
- Fikes, R. 13.10, 15.10
- Fillmore, C. 8.10
- fin, value of WORM feature 限定形式, VFORM 特征的值 4.2.2

- Findler, N. 10.10
- finite state machine 有限状态自动机 3.5
- finite state transducer 有限状态转录机 3.7
- first-in first-out 先进先出 3.3.2
- first-order predicate calculus 一阶谓词演算 A.1
as a representation language 作为表示语言 13.2
- focus spaces 焦点空间 16.12
- Fodor, J. 10.8
- FOPC, 参见 first-order predicate calculus A.0
- Ford, M. 6.7, 10.8
- formants 共振峰 C.2
- forward probability 前向概率 7.4
- forward-looking center 前瞻性中心 14.3
- Fox, B. 16.9
- frames 框架 13.3
- Francis, W. 7.9
- Frazier, L. 6.7
- Freuder, E. 10.8
- Friedman, L. 14.9
- FROM-LOC role FROM-LOC 角色 8.6
- FRUMP system FRUMP 系统 11.6
- FSM, 参见 finite state machine 3.5
- FST, 参见 finite state transducer 3.7
- functions, in logic 函数, 逻辑中的 A.1
- fundamental frequency 基频 C.2
- future-directed intention 以未来为导向的意图 17.4
- GAP feature GAP 特征 5.3
- gap features, adding to a grammar 缺位特征, 加入到语法中的 5.3
- gap threading, in logic grammars 缺位索引, 逻辑语法中的 5.6
- gaps and fillers 缺位与填充成分 5.2
- garden-path sentences 花园幽径型句子 6.1
- Gazdar, G. 3.10, 4.10, 5.8
- gender 性别 2.2
- generalized phrase structure grammar 广义短语结构语法 5.4
feature propagation 特征传播 5.4
- generalized quantifiers 广义量词 8.3, 8.4
- generation of sentences 句子生成 9.7
- generation relationship, between actions 生成关系, 行为之间的 15.4
- generative capacity, of grammars 生成能力, 语法的 3.2
of transition networks 转移网络的 3.7
- generic interpretations 泛指解释 12.7.2
- Genesereth, M. 13.10, A.4
- Georgeff, M. 17.12
- gerundive 动名词形式的形容词 2.4
- glass box evaluation 白箱评价 1.3
- global focus 全局焦点 16.7
- goal 目标 15.6, 17.4
- Goldman, A. 15.10
- Goldman, R. 15.10
- Government and Binding 管辖与约束 12.9
- government-binding (GB) theory 管辖约束(GB)理论 5.2
- grammar 语法 3.1
augmented 扩充 4.1
context-sensitive 上下文相关 3.2
evaluating评价 3.2
regular 正则 3.2
type 0 0型 3.2
- grammatical dependencies 语法依存 11.1
- grammatical relations 语法关系 11.1
- Grice, P. 17.8, 17.12
- Grimes, J. 16.9
- Grishman, R. 7.9, 10.8
- Grosz, B. 1.8, 12.9, 13.10, 14.9, 16.9, 17.12
- Haas, A. 17.3
- Halliday, M. 4.4, 14.9, 15.10
- Hankamer, J. 14.9
- Harper, M. 14.9
- Hasan, R. 15.10
- Haviland, S. 14.9
- Hayes, P. 10.8, 13.10, 15.10
- head 中心语 2.1
- head features 中心特征 4.4
- head-driven phrase structure grammar 中心驱动短语结构语法 5.8
- head-driven realization algorithm 中心成分驱动的实现算法 9.7
- Heim, L. 14.9
- Hendler, J. 15.10
- Hendrix, G. 11.6, 17.12
- Herskovits, A. 12.9
- Hewitt, C. 13.10
- Hidden Markov Model 隐马尔可夫模型 7.5, C.1
for speech 语音的 C.4
training的训练 C.4
- Hierarchical Lexicons 层次词典 9.4.1
- hierarchical plans 层次化的规划 15.6
- Hindle, D. 6.7, 7.9, 10.8
- Hinkelman, E. 17.12

- Hinrichs, E. 16.9
- Hintikka, J. 17.12
- Hirschberg, J. 16.9, C.7
- Hirst, G. 9.9, 10.8, 11.6, 16.9
- history list 历史列表 14.2
- HMM, 参见 Hidden Markov Model 7.3
- Hobbs, J. 8.10, 12.9, 14.9, 15.10, 16.5, 16.9
- hold action, in ATNS 保留操作, ATNS 中的 5.5
- hold list 保留表 5.2
- in ATNS ATNS 中的 5.5
- Horn clause 霍恩子句 13.6, B.4
- homogeneous propositions 同质命题 13.5
- horizontally related quantifiers 横向相关量词 12.1
- Horn, B. B.5
- Hovy, E. 16.9
- Huddleston, R. 2.6
- human parsing preferences 人的句法分析优选策略 6.1
- human-machine communication 人机通信 1.1
- Hwang, C. 8.10, 9.9, 16.9
- hybrid knowledge representations 综合的知识表示 13.8
- illocutionary act 语内表现行为 17.2
- recognition of ……的识别 17.9
- immediate domination, in D-theory 直接控制, D 理论中的 6.5
- implication 隐含 13.1
- between sentences 句子之间的 8.8.1
- implicit beliefs 隐含信念 17.3
- inconsistency, in logic 不相容性, 逻辑中的 A.2.1
- indefinite noun phrases, and discours 不确定性名词短语, ……和篇章实体 14.1.1
- indefinite quantifiers 不确定性量词 12.1.1
- independent events 独立事件 7.1
- indexical 指示的 8.1
- indirect object 间接宾语 2.3.2, 11.1
- indirect reference 间接指代 14.4.1
- individuals, in KRs 个体, 知识表示中的 13.1
- inductive inference 归纳推理 13.1.1
- inf, value of VFORM feature 不定式, VFORM 特征的值 4.2.2
- inference rule, in logic 推理规则, 逻辑中的 A.1.1
- inference, abductive 推理, 溯因 13.1.1
- and discourse structure ……和篇章结构 16.4
- declarative 描述性 13.1.2
- deductive 演绎 13.1.1
- defeasible 可推翻的 13.1.1
- in knowledge representations 知识表示中的 13.1
- procedural 程序性 13.1.2
- inflectional form 屈折式 2.1
- information retrieval 信息检索 1.2
- informational vs. intentional models 信息层次模型与意图模型
 的比较 16.2
- ing, value of VFORM feature 现在分词, VFORM 特征的值 4.2.2
- inheritance, in lexicons 继承, 词典中的 9.4.1
- in semantic networks 语义网络中的 10.3
- inside probability 内部概率 7.5
- INSTR role INSTR 角色 8.6
- intended speech act 意图的言语行为 17.9
- intensional meaning 内涵意义 8.8.2
- intensional operators 意愿运算符 12.7.3
- intention in action 行为中的意图 17.4
- intention 意图 17.1, 17.4
- recognition of ……的识别 17.7
- interpretation function 解释函数 8.8
- intersective adjective 交集型形容词 12.5
- intersentential anaphora 句内回指 12.4
- intonation 语调 C.6
- intonation contour 语调轮廓 C.6
- intransitive verb 不及物动词 2.3.1
- intrasentential anaphora 句间回指 12.4
- INV feature INV 特征 4.3
- iplan 15.8
- IRREG-PAST feature IRREG-PAST 特征 4.3
- IRREG-PRES feature IRREG-PRES 特征 4.3
- island constraints 孤岛约束 5.3.1
- isolated word recognition 孤立词识别 C.1
- Israel, D. 17.12
- iterative deepening 迭代深化 B.3
- Jackendoff, R. 8.10
- Jelinek, F. 7.9
- Jensen, K. 10.8
- Johnson, M. 4.10
- Joshi, A. 14.9
- jump arcs, in an RTN 跳跃边, RTN 中的 3.5
- Kamp, H. 14.9
- Kaplan, R. 4.10, 5.8, 6.7, 10.8
- Karttunen, L. 14.9
- Katz, J. 10.8
- Kautz, H. 15.10, 15.10
- Kay, M. 3.10, 4.10
- KB, 参见 knowledge base 13.1
- KERNEL system KERNEL 系统 11.6

- key, in a chart parser 键, chart 句法分析器中的…… 3.4
- Kimball, J. 6.7
- KIMMO 3.10
- kinds 类 12.7.2
- Klein, E. 4.10
- Knight, K. 3.10, A.4
- KnowIf 17.3.1
- KnowingHow 17.3.1
- knowledge base 知识库 13.1
- knowledge preconditions 知识前提 17.6
- knowledge representation language 知识表示语言 13.1
- knowledge representation 知识表示 1.6, 12.9
and logical form ……和逻辑形式 12.9
- KnowRef 17.3.1
- Knuth, D. 4.10
- Konolige, K. 17.3, 17.12
- Koskenniemi, K. 3.10
- KR, 参见 knowledge representation 13.0
- Kripke, S. 17.12
- KRL, 参见 knowledge representation language 13.1
- Kucera, H. 7.9
- labeled arcs, in transition network grammars 标记边, 转移网络语法中的…… 3.5
- lambda abstraction λ 提取 9.1, 9.7
- lambda calculus λ 演算 9.1, 9.1
- lambda reduction λ 归约 9.1
- Lascarides, A. 16.9
- last-in first-out 后进先出 3.3.2
- late closure 后闭合 6.1.2
- leaves, of a tree 叶子节点, 树的 3.1
- Lee, K. 7.9, C.4, C.7
- Leech, G. 2.8, 16.9
- Levesque, H. 13.10, 17.12
- Lewis, D. 14.9
- lexical preferences 词汇优先原则 6.1.3
- lexical rules, for morphological analysis 词法规则, 词语形态分析的 4.3
in a grammar 语法中 3.3
- lexical symbols, in a grammar 词法符号, 语法中的 3.1
- lexical-generation probabilities 词语的生成概率 7.3
- lexicalized semantic interpretation 词汇化语义解释 9.4
- lexicon 词典 3.3
for semantic interpretation ……的语义解释 9.2
- LIFER system LIFER 系统 11.6
- LIFO 3.3.2
- Linde, C. 16.9
- linear predictive coding 线性预测编码 C.3
- Linguistic Data Consortium 语言数据联盟 7.9
- link, in a tree 连接, 树中的 3.1
- LISP LISP 语言 B.0
- lists, in LISP 表, LISP 语言中的 B.1
- literal meaning hypothesis 字面意义假设 17.9
- literals, in logic 文字问题, 逻辑中的 13.6
- Litman, D. 15.10, 16.9, 17.12
- local discourse context 局部篇章上下文 14.1
- local discourse state 局部篇章状态 16.2
- locutionary act 非语内表现行为 17.2
- logic programming 逻辑程序设计 3.8, B.4
- logical form 逻辑形式 1.5.2, 8.1
and knowledge representation ……和知识表示 13.1
- logical form language, semantics for 逻辑形式语言, ……语义 8.7
summary of ……总结 8.8
- logical object 逻辑宾语 11.1
- logical operators, in the logical form 逻辑运算符, 逻辑形式中的 8.3
- logical subject 逻辑主语 11.1
- logical variable 逻辑变量 A.1
- LR(1) parsing LR(1)句法分析 6.7
- Luger, G. A.4
- LUNAR 1.8, 13.8
- LUNAR system LUNAR 系统 8.10, 12.9
- Lyttinen, S. 11.6
- MacWorth, A. 10.8
- Magerman, D. 7.9
- Mann, W. 4.4, 16.5, 16.9
- Marcus, M. 6.7, 7.9
- margin of error 误差区间 7.2
- Markov assumption 马尔可夫假设 7.3
- Markov chains 马尔可夫链 7.3
- Marshall, C. 17.12
- mass noun 物质名词 2.2, 12.7.1
- matching lists 匹配表 B.2
- Mathiesson, C. 4.4
- maximum likelihood estimator 最大似然估计 7.2
- May, R. 12.9
- McCarthy, J. 15.10
- McCawley, J. 8.10, 12.9, 13.10, 14.9, A.4
- McClelland, J. 10.8
- McConnell-Ginet, S. 8.10, 12.9, 14.9
- McCord, M. 4.10, 8.10, 9.9, 12.9
- McDermott, D. 3.10

- McDonald, D. 9.9
- meaning 意义 8.1
- meaning postulates 意义公设 8.5
- meets relation, in temporal logical 接续关系, 时态逻辑中的 13.5
- Mellish, C. 3.10, 10.8, B.6
- message understanding 消息理解 11.3
- meta-rules, in GPSG 元规则, GPSG 中的 5.4
- Miller, G. 10.8
- minimal attachment principle 最小附着原则 6.1.1
- Minsky, M. 13.10
- MLE, 参见 maximum likelihood estimator 7.2
- modal auxiliaries 情态助动词 5.1
- modal operators, in the logical form 情态运算符, 逻辑形式中的 8.3
- semantics for ……语义 8.8.2
- modal verb 情态动词 2.3
- model 模型 8.8
- model theory 模型论 A.2
- for logic 逻辑的 A.3
- monotonic inference 单调推理 13.1.1
- Montague, R. 8.8.2, 8.10, 9.9
- mood 语气 2.3
- Moore, J. 16.9
- Moore, R. 8.10, 9.9, 17.12
- morphological analysis 词语形态分析 3.7, 4.3
- morphology 词语形态学 1.4, 2.1
- most general unifier 最广合一 B.2
- mother, of a rule 母亲符号, 规则的…… 3.1
- movement, local versus unbounded 移位, 局部……与无界……的对比 5.2
- MUC 11.3
- mutual belief, see shared knowledge 相互的信念, 参见 shared knowledge 17.3.1
- mutual information 互信息 10.4.1
- n-best output n-best 输出, n 最优输出 C.5
- n-gram n 元语法 7.3
- name 名称 2.2
- narrative convention 陈述惯例 16.5
- nasal phonemes 鼻音 C.2
- natural language understanding 自然语言理解 1.2
- negation as failure 失败否定 13.6
- Nebel, B. 13.10
- Nilsson, N. 13.10, 15.10, A.4
- node, in a tree 节点, 树中的 3.1
- nodes, in transition network grammars 节点, 转移网络语法中的 3.5
- nominalizations 名词化 12.6
- nonintersective adjective 非交集型形容词 12.5
- nonmonotonic inference 非单调推理 13.1.1
- nonmonotonic logic, semantics of 非单调逻辑, ……的语义 13.1.1
- nonterminal symbols 非终结符 3.1
- normal form, for FOPC 范式, 一阶谓词演算的…… 13.6
- Norvig, P. 3.10, B.7
- noun 名词 2.1
- noun group 名词词组 6.5, 11.3
- noun-noun modification 名词修饰名词 2.2
- noun-noun modifiers 名词-名词修饰语 12.7.4
- now point, in scripts 当前点, 脚本中的 15.5
- number agreement 数的一致性 4.1
- number feature 数特征 4.2.1
- numerical quantifiers 数值量词 12.6
- ontology 知识本体 8.2
- open class word 开放类词语 2.1
- oracle, in shift-reduce parsing 预测表, 移进归约句法分析中的 6.2.2
- ordinal 序数词 2.2
- orients relation 适应关系 16.5
- output probability 输出概率 7.3
- packed charts 压缩 chart 6.4
- Palmer, M. 11.6
- parent node 父节点 3.1
- PARRY 1.8
- parsing 句法分析 3.1
- bottom-up 自底向上 3.1
- bottom-up chart 自底向上的 chart 3.4
- partial 浅层 6.5
- semantically driven 语义驱动的 11.4
- top-down 自顶向下 3.1, 3.3
- top-down chart 自顶向下的 chart 3.6
- Parsons, T. 8.10, 13.10
- part-of hierarchy 整体-部分层次 10.3
- part-of-speech tagging 词性标注 7.3
- Partee, B. 8.10, 9.9, 16.9, A.9
- partial parsing 浅层句法分析 6.5
- and template matching ……与模板匹配 11.3
- particle 语助词 2.3.2, 3.2
- passive 被动的 2.3.1, 5.1
- past, value of VFORM feature 一般过去时态, VFORM 特征的值 4.2.2
- pastprt, value of VFORM feature 过去分词, VFORM 特征的

- 值 4.2.2
- PATH-LOC role PATH-LOC 角色 8.6
- Patten, T. 4.4
- Paxton, W. 7.9
- Pelletier, J. 8.10, 9.9, 13.10
- Penn Treebank 宾州树库 7.3, 7.9
- Pereira, F. 3.10, 4.10, 5.8, 6.7, 7.9, 9.9, 12.9, 13.10
- perlocutionary act 言语表达效果行为 17.2
- Perrault, R. 1.8, 15.10, 17.10, 17.12
- Perry, J. 8.10
- person 人称 2.2
- person feature 人称特征 4.2.1
- PFORM feature PFORM 特征 4.2.2
- phonemes 音素 C.2
- in context 上下文中的 C.4
- phonology 音韵学 1.4
- Pierrehumbert, J. C.7
- pitch, in speech 基频, 语音的 C.2
- plan 规划 15.6
- plan inference 规划推理 15.6, 15.7.1
- formalizing ……的形式化 15.8
- plan recognition 规划识别 15.6, 17.4
- PLANNER 13.10
- planning 规划的制定 17.1
- plans 规划 17.4
- plosives 爆破音 C.2
- plural NPs, and discourse entities 复数名词短语, ……和篇章实体 14.1.1
- Polanyi, L. 16.9
- Pollack, J. 10.8
- Pollack, M. 12.9, 15.10, 16.9, 17.12
- Pollard, C. 5.8, 14.9
- pop arcs, in an RTN 弹出边, RTN 中的 3.5
- pop operation, on stacks 弹出操作, 栈的 B.1.1
- possessive noun phrase 所有格名词短语 2.2
- possibilities list 可能状态列表 3.3.1
- possible worlds semantics 可能的世界语义 8.8.2
- potential next centers 潜在的下一个中心 14.3
- PP attachment 介词短语附着 7.7
- ambiguity of ……歧义 6.4
- pragmatics 语用 1.4
- precondition causality 前提因果关系 15.4
- preconditions, of actions 前提, 行为的 13.3.1
- predicate completion 谓词完备性 13.1.1
- predicate modifiers 谓词修饰成分 12.5
- predicate operators, in the logicalf 谓词运算符, 逻辑形式中的 8.3
- predicates, in the logical form 谓词, 逻辑形式中的 8.3
- preferred next center 优选的下一个中心 14.3
- prepositional phrases, semantic interpretation of 介词短语, ……的语义解释 9.3
- pres, value of VFORM feature 现在时, VFORM 特征的值 4.2.2
- primitives, in knowledge representati 原语, 知识表示中的 13.2
- Prince, E. 14.9
- Prior, A. 13.10
- priority queue 优先队列 7.6
- probabilistic context-free grammars 概率上下文无关文法 7.5
- probabilistic parsing, context dependent 概率句法分析, 上下文相关的 7.6
- context-free 上下文无关的 7.5
- probability 概率 7.1
- probability estimates 概率估计 7.2
- procedural attachment 附属程序 13.8
- procedural semantics, and question answering 程序语义学, ……和问答系统 13.7
- progressive 进行时的 2.3, 16.5
- PROLOG 3.8, B.0, B.4
- pronoun, agreement 代词, 一致性 2.2
- pronouns and centers 代词和中心 14.3
- proof by failure 失败证明 13.6
- proof, in logic 证明, 逻辑中的 A.1.1
- proper noun 专有名词 2.2
- proper noun phrases 专有名词短语 6.5
- propositional calculus 命题演算 A.2
- propositions, in logic 命题, 逻辑中的 A.1
- in the logical form 逻辑形式中的 8.3
- prosody 韵律 C.6
- provability, in logic 可证明性, 逻辑中的 A.2.1
- Pullum, G. 4.10
- push arcs, in an RTN 压入边, RTN 中的 3.5
- push operation, on stacks 压入操作, 栈的 B.1
- Pustejovsky, J. 13.10
- qualifier 修饰词 2.2
- quantification, handling in a KR 量化, 知识表示中的处理 13.4
- quantifiers, classification of 量词, ……的分类 12.1.1
- in logic 逻辑中的 A.3.1
- quantifying determiner 数量修饰语 2.2
- quantization factor, in signal processing 量化因子, 信号处理

- 中的 C.3
- quasi-logical form 准逻辑形式 8.4
- question answering 问答 13.7
- question-answering systems 问答系统 1.2
- questions, semantic interpretation of 疑问句,……的语义解释 9.5
- queues 队列 B.1.1
- Quilhan, M. 10.8
- Quirk, R. 2.8
- quotation operator 引用运算符 17.3
- Rabiner, L. 7.9, C.7
- Radford, A. 5.8
- random variable 随机变量 7.1
- Rao, A. 17.12
- rational behavior, and plan inference 理性行为,……和规划推理 15.8
- rational updates 理性更新 15.8
- rationality, properties of 理性,……的属性 17.4
- Rayner, K. 6.7
- realized nodes, in plan inference 实现节点,规划推理中的 15.7.1
- recency constraint for anaphora 回指的最近约束 14.2
- recursive transition network 递归转移网络 3.5
- parsing algorithm ……的句法分析算法 3.5.1
- reduce action, in shift-reduce parsing 归约操作,移进归约句法分析中的 6.2.2
- reduced relative clause 简化的关系从句 2.4, 5.4
- reference time 参照时间 13.5.1
- reference, and discourse structure 指代,……和篇章结构 16.3
- and expectation matching ……和期望匹配 15.3
- referentially opaque 指代不透明的 12.7.3
- referentially transparent 指代透明的 12.7.3
- referring expressions, and generation 指代表达式,……和生成 14.4
- reflexive pronoun 反身代词 12.4
- refutation proof 反证 13.6
- registers, in ATNs 寄存器, ATN 中的 4.6
- regular grammar 正则文法 3.2
- Reichenbach, H. 13.5.1, 13.10
- Reichman, R. 16.9
- Reinhart, T. 12.9
- Reiter, R. 13.10, 13.10
- relational nouns 关系名词 12.6
- relative clause 关系从句 2.4, 5.4
- semantic interpretation of ……的语义解释 9.6
- relative pronoun 关系代词 2.4
- repairs; in spoken language 更正, 口语中的 C.1
- Resnik, P. 10.8
- resolution rule 消解规则 13.6
- rewrite rules 产生式规则 3.1
- Rhetorical Structure Theory 修辞结构理论 16.5, 16.9
- Rich, C. 13.10
- Rich, E. 3.10, A.4
- Rieger, C. 15.10
- Riesbeck, C. 11.6, 13.3
- right association 右关联 6.1.2
- Ritchie, G. 3.10, 11.6
- Robinson, A. 7.9
- Robinson, J. 4.10, 5.8
- role instantiation, in scripts 角色实例化, 脚本中的 15.5.1
- roles, in frame-based representations 角色, 基于框架表示中的 13.3
- root 词根 2.1
- Rooth, M. 7.9, 10.8
- Ross, S. 7.9
- Rounds, W. 4.9
- RTN, 参见 recursive transition network 3.5
- rule-by-rule interpretation 逐条规则的解释 9.1
- Rumelhart, D. 10.8, 16.9
- RUS system RUS 系统 11.6
- Russell, B. 14.9
- Sacerdoti, E. 15.10
- Sag, L. 4.10, 5.8, 14.9
- Sager, N. 4.10
- Saint-Dizier, P. 9.9, 12.9
- sampling rate, in signal processing 采样频率, 信号处理中的 C.3
- Sandewall, E. 13.10
- Scha, R. 12.9, 16.9
- Schabes, Y. 7.9
- Schafer, R. C.7
- Schank, R. 1.8, 11.6, 13.3, 13.10
- Schiffer, S. 17.12
- Schmidt, C. 15.10
- Schubert, L. 6.7, 8.10, 9.9, 12.9
- scope ambiguity 辖域歧义 12.1
- scope determination while parsing 句法分析中的辖域判定 12.3
- scoping, in a database query application 辖域指定, 数据库查询程序中的 12.3
- in the logical form ……在逻辑形式中的 8.4

- scripts 脚本 15.5
- search algorithms 搜索算法 B.3
- search strategy, breadth-first 搜索策略, 广度优先 3.3.2
- depth-first 深度优先 3.3.2
- Searle, J. 17.5, 17.12
- selectional restrictions 选择约束 10.1
- Selfridge, M. 11.6
- Sells, P. 4.10
- SEM feature SEM 特征 9.2
- semantic ambiguity 语义歧义 8.2
- semantic filtering 语义过滤 10.2
- semantic grammar 语义句法 11.2
- semantic interpretation 语义解释 8.1, 9.1
 - compositional 组合 9.1
 - of adjectives 形容词的 12.5
 - rule-by-rule 逐条规则的 9.1, 11.1
 - using features 用特征的 9.6
- semantic network 语义网络 10.3
- semantic preferences 语义优选 10.1
 - statistical 统计 10.5
- semantic realization 语义实现 9.1, 9.7
- semantic return 语义回返 16.7
- semantically driven parsing 语义驱动的句法分析技术 11.4
- semantics 语义学 1.4
 - of non-monotonic logic 非单调逻辑的 13.1.7
 - of the logical form language 逻辑形式语言的 8.8
- sense, of a word 意义, 词的 8.2
- Sentential Subject Constraint 句子主语约束 5.3.1
- Sethi, R. 3.10
- Shapiro, S. 1.8, 9.9
- shared knowledge 共享知识 17.3
- Shieber, S. 4.10, 6.7, 8.10, 9.9, 12.9, 14.9
- shift action, in shift-reduce parsing 移进操作, 移进归约句法分析中的 6.2.2
- shift-reduce parsing 移进归约句法分析 6.2
- Sidner, C. 14.9, 16.9, 17.12
- signal processing 信号处理 C.3
- sincerity condition 真诚性条件 17.8
- situation 情境 8.1
- Skolem constants, as discourse entities Skolem 常数 14.1
- Skolem functions Skolem 函数 13.2
- skolemization skolem 化 13.2
- slash categories 斜杠类 5.2
- Slocum, J. 1.8
- slots, in frame-based representations 槽, 基于框架的表示中的 13.3
- Small, S. 10.8
- smoothing probability estimates 平滑概率估计 7.4
- Song, F. 16.9
- SOPHIE system SOPHIE 系统 11.6
- soundness, of inference 可靠性, 推理的 13.1, A.2.1
- Sowa, J. 10.8, 13.10
- sparse data problem 数据稀疏问题 7.2
- specifier 指定词 2.2
- spectral analysis 频谱分析 C.2
- speech acts 言语行为 17.1, 17.5
 - taxonomy of ……的分类体系 17.5
- speech rate 语速 C.6
- speech recognition 语音识别 1.2, C.1
- SPHINX system SPHINX 系统 C.4
- Sproat, R. 3.10
- Stabler, E. 12.9
- stacks 栈 B.1.1
- Stalnaker, R. 14.9
- states, in the logical form 状态, 逻辑形式中的 8.5
- statistical word sense disambiguation 统计词义消歧 10.4
- stative propositions 状态性命题 13.5
- Steedman, M. 10.8, 16.9
- Sterling, J. 10.8
- stops 爆破音 C.2
- Strawson, P. 17.12
- stress, in speech 重音 C.6
- Stubblefield, W. A.4
- SUBCAT feature SUBCAT 特征 4.2.2
- SUBJ feature SUBJ 特征 9.6
- subject-aux. inversion 主助倒置 5.2
- subsecutive adjective 子集型形容词 12.5
- suffix 后缀 2.1
- summarization 总结 1.2
- supports relation, in model theory 支持关系, 模型论中的 8.8
- surface anaphora 表层回指 14.7
- surface speech act 表层言语行为 8.7, 17.9
- surface structure 表层结构 5.2
- Svartvik, J. 2.6
- Swartout, R. 13.10
- symbol list, in a top-down parser 符号表, 自顶向下句法分析器中的 3.3
- symbol table 符号表 B.5
- syntactic structure 句法结构 1.5.1
- syntax 句法 1.4

- systemic grammar 系统语法 4.4
 TAG, 参见 tree-adjoining grammar 5.8
 Tarski, A. A.1
 Tate, A. 15.10
 tautology, in logic 重言式, 逻辑中的 A.1.1
 TEAM 13.10
 telic propositions 完成性命题 13.5
 template matching 模板匹配 11.3
 temporal duration 持续时间 13.5
 temporal focus, in discourse 时间焦点, 篇章中的 16.5
 temporal logic 时态逻辑 13.5
 temporal shift, in discourse 时间转移, 篇章中的 16.6.1
 tense 时态 2.3
 and discourse structure ……和篇章结构 16.5
 representation of ……的表示 13.5.1
 tense trees 时态树 16.5
 term subsumption languages 项分类语言 13.10
 terminal symbols 终结符 3.1
 terms, in logic 项, 逻辑中的 A.1
 in the logical form 逻辑形式 8.3
 test set 测试集 7.2.1
 tests, in augmented transition networks 测试, 扩充转移网络中 4.6
 thematic roles 论旨角色 8.5, 8.6
 and semantic interpretation ……与语义解释 9.4
 listing ……列表 8.6
 THEME role THEME 角色 8.6
 theorem, in logic 定理, 逻辑中的 A.1
 Thomason, R. 8.10
 Thompson, S. 16.5, 16.9
 time and aspect 时间和体态 13.5
 time intervals 时间段 13.5
 time point 时间点 13.5
 TO-LOC role TO-LOC 角色 8.6
 Tomita, M. 6.7
 top-down parser, algorithm 自顶向下句法分析器, 算法 3.3.1
 topicalization 主题化 5.2, 5.8
 training set 训练集 7.2.1
 transformational grammar 转换语法理论 5.2
 transformations 转换 5.2
 transition networks 转移网络 3.5
 generative capacity ……的生成能力 3.6
 transitive verb 及物动词 2.3.1
 translation 翻译 1.2
 Traum, D. 17.12
 tree 树 3.1
 tree-adjoining grammar 树邻接语法 5.8
 trigram 三元语法 7.3
 triphones 三音子 C.4
 truth, of a proposition 真值, 命题的 8.8
 type 0 grammar 0型文法 3.2
 type hierarchies, and disambiguation 类型层次体系, ……与消歧 10.1
 types, in knowledge representations 类型, 知识表示中 13.1
 Ullman, J. 3.10
 unbounded movement 无界移位 5.2
 understanding natural language 理解自然语言 1.2
 unification algorithm 合一算法 B.2, B.5
 unification, in automated deduction 合一, 自动演绎中的 13.6
 universal quantifiers 全称量词 12.1.1
 in logic 逻辑中的 …… A.1
 unknown words 未登录词 7.6
 vagueness 模糊性 8.2
 valuation function, in model-theoretics 值函数, 模型论语义学中的 A.2
 value, of a feature 值, 特征 4.1
 VanLehn, K. 12.9
 verb group 动词词组 6.5, 11.3
 vertically related quantifiers 纵向相关量词 12.1.2
 VFORM feature VFORM 特征 4.2.2
 Vijay-Shankar, K. 6.7
 Viterbi algorithm Viterbi 算法 7.3, C.4
 Viterbi, A. 7.9
 voiced phonemes 浊音 C.2
 Waibel, A. 7.9, C.7
 Waltz, D. 10.8, 10.8
 Wang, M. C.7
 want 需求 17.1
 Warren, D. 3.10, 4.10, 13.10
 Webber, B. 11.6, 13.10, 14.9, 16.9
 Weinberg, A. 12.9
 Weinstein, S. 14.9
 Weir, C. 7.9
 Weischedel, R. 7.6, 7.9
 Weizenbaum, J. 1.8
 well-formed substring table 良构子串表 3.5.1
 WFST, 参见 well-formed substring table 3.5.1
 WH feature WH 特征 5.3
 wh-determiner 特殊疑问词 2.2
 Wh-Island Constraint Wh-孤岛约束 5.3.1

- wh-movement wh-移位 5.2
- WH-QUERY act WH-QUERY 行为 8.7
- wh-questions 特殊疑问句 5.2, 5.3
- Whitehead, A. 14.9
- Whittemore, G. 7.9
- Wilensky, R. 15.10, B.6
- Wilks, Y. 10.8, 11.6, 13.10
- Winograd, T. 1.8, 3.10, 4.4, 5.8, 13.10
- Winston, P. 3.10, A.4
- Woods, W. 1.8, 3.10, 4.10, 5.8, 8.10, 10.8, 11.6, 12.9,
13.10, 14.9
- word sense disambiguation, statistical 词义消歧, 统计 10.4
- word sense hierarchies 词义层次体系 10.1
- word senses 词义 8.2
- WordNet 10.8
- world knowledge 世界知识 1.4
- Y/N-QUERY act Y/N-QUERY 行为 8.7
- Yarowsky, D. 10.8
- yes/no questions 一般疑问句 5.2

正文中使用的符号

句法范畴		
ADJ	adjective	形容词
ADJP	adjective phrase	形容词短语
ADV	adverb	副词
ADVP	adverbial phrase	副词短语
ART	article	冠词
AUX	auxiliary verb	助动词
CNP	common noun phrase	普通名词短语
DET	determiner	限定词
DETP	determiner phrase	限定词短语
N	noun	名词
NP	noun phrase	名词短语
P	prepositional	介词
PP	prepositional phrase	介词短语
PRO	pronoun	代词
REL	relative clause	关系从句
REL-PRO	relative pronoun	关系代词
QDET	quantifying determiner	数量限定词
PP-WRD	word acting like a PP	具有介词短语特征的词语
S	sentence	句子
V	verb	动词
VP	verb phrase	动词短语
数值关系		
<	less than	小于
≤	less than or equal	小于或等于
>	greater than	大于
≥	greater than or equal	大于或等于
=	equals	等于
≠	not equals	不等于
时间关系		
<	before	前于
:	meets	相接于
<:	before or meets	前于或相接于
⊆	is a subinterval (or equal)	包含于或等于
逻辑符号		
∀	universal quantifier	全称量词

\exists	existential quantifier	存在量词
\neg	negation("not")	否定("非")
$\&$	conjunction("and")	合取("与")
\vee	disjunction("or")	析取("或")
\supset	implication("if...then...")	蕴涵("如果……那么……")
\equiv	equivalence("if and only if")	等价("当且仅当")
$=$	equals	等于
\neq	not equals	不等于
\Rightarrow	default implication	默认蕴涵
集合		
\in	is a member of	属于
\notin	is not a member of	不属于
\subseteq	is a subset (or equal)	包含于或等于
\cup	intersection	交集
\cap	union	并集
$ S $	size of S	S 中的元素个数
$\{A, B\}$	set consisting of A and B	由 A 和 B 组成的集合
$\{X P_x\}$	set of objects such that P is true	使得 P 为真的所有对象集合
首字母缩写词		
ATN	augmented transition network	扩充转移网络
CFG	context free grammar	上下文无关文法
DCG	definite clause grammar	定子句文法
FOPC	first-order predicate calculus	一阶谓词演算
FSM	finite state machine	有限状态自动机
FST	finite state transducer	有限状态转录机
GPSG	generalized phrase structure grammar	扩充短语结构语法
HMM	hidden Markov model	隐马尔可夫模型
KB	knowledge base	知识库
KR	knowledge representation	知识表示
LF	logic form	逻辑形式
RTN	recursive transition network	递归转移网络