第8章 一阶逻辑

我们注意到世界幸福地拥有着许许多多的对象，对象之间可能存在联系，我们尽力对它们进行推理。

第 7 章讨论了用基于知识的 agent 表示它所处的世界并推理将要采取的行动。我们把命题逻辑作为表示语言，因为它足以阐述逻辑和基于知识的 agent 的基本概念。不幸的是，命题逻辑是一种表达能力很弱的语言，无法以简洁的方式表示复杂环境的知识。本章考察**一阶逻辑**[1]，它具有丰富的表达能力，可以表示大量常识知识。它还包含或形成了很多其他表示语言的基础，已经被深入研究了几十年。8.1 节讨论一般的表示语言；8.2 节涵盖了一阶逻辑的语法和语义；8.3 节和 8.4 节说明了一阶逻辑在简单表示中的运用。

8.1 重温表示

本节讨论表示语言的本质。我们的讨论将引出一阶逻辑的发展，它是一个比第 7 章所介绍的命题逻辑表达能力更强的语言。我们将着眼于命题逻辑和其他类型的语言以便了解这些语言能做什么不能做什么。我们的讨论有些粗略，把几个世纪的思想、试验和错误浓缩为几段文字。

程序设计语言（如 C++、Java 或 Lisp）是到目前为止常用的形式语言中最大的一类。程序本身在直接的意义下表示的是计算过程。程序中的数据结构可以表示事实；例如，程序可以用 4 × 4 数组表示 Wumpus 世界。那么，程序设计语言的语句 *World*[2,2]←*Pit* 是加入断言[2, 2]有陷阱的很自然的方式（这样的表示可能被认为是专门的；数据库系统的精确开发提供了一种更通用的、独立于领域的方法来存储和检索事实）。程序设计语言缺乏的是从其他事实推导出其他事实的通用机制；数据结构的每次更新都是通过领域特定的过程来完成，该过程的细节是由程序员根据他或她自己拥有的关于该领域的知识得出的。这种**过程性**方法可以和命题逻辑的**描述性**本质相对，知识和推理是分开的，而且推理完全独立于领域。

程序（以及数据库）中的数据结构的第二个缺点是缺乏任何简便的表述方式，例如，“[2, 2]或[3, 1]中有一个陷阱”或者“如果 Wumpus 在[1, 1]，那么它不在[2, 2]中”。程序可以为每个变量保存一个单独的值，而且某些系统中允许该值是“未知的”，但是它们缺乏处理不完全信息所需的表达能力。

命题逻辑是一种描述性语言，因为它的语义是基于语句和可能世界之间的真值关系。它有充分的表达能力，可以采用析取式和否定式来处理不完全信息。命题逻辑所拥有的第

1 也称为**一阶谓词演算**，有时缩写为 **FOL** 或 **FOPC**。