PROLOG程序设计

〔英〕W.F.克洛克辛　C.S.梅利什　著

李德毅　赵立平　译

内 容 简 介

计算机程序设计语言 PROLOG 是以处理逻辑推理问题为背景而设计的一种交互式语言。本书通俗而全面地讨论了 PROLOG 语言的内容和特点，所有这些，都可以作为 PROLOG 程序设计的基础。书中还通过大量的例子，介绍了 PROLOG 在数理逻辑、抽象问题求解、自然语言理解、数据库系统等各个领域中的应用。全书共分十一章，其中，不少章节附有习题，书末有部分习题解答。

本书可作为大专院校计算机专业师生学习 PROLOG 程序设计的教科书；对于力图把 PROLOG 用于专门领域的广大计算机工程人员，也是一本很好的自学用书；当然，了解 PROLOG 语言，对计算机科学工作者，特别是对于从事人工智能和第五代计算机研究工作的人则更为重要。

PROGRAMMING IN PROLOG
W. F. Clocksin C. S. Mellish
Springer-Verlage Berlin Heidelberg 1981

*

PROLOG程序设计

〔英〕 W. F. 克洛克辛 C. S. 梅利什 著
李德毅 赵立平 译

*

国防工业出版社出版
新华书店北京发行所发行 各地新华书店经售
国防工业出版社印刷厂印装

*

850×1168 1/32 印张 千字
1988年4月第一版 1988年4月第一
数：0,001—3,700册

ISBN7-118-00363-7/T 定价：2.90元

译 者 序

计算机程序设计语言 PROLOG (Programming in Logic) 是以符号逻辑为理论基础的一种交互式语言，特别适用于解决人工智能方面的问题。PROLOG 以其简洁的文法、丰富的表现力、高度的模块化以及独特的非过程性而越来越受到人们的重视。

PROLOG理论的发明人是英国帝国理工学院的R. Kowalski教授。第一个实用的解释文本由法国马赛大学 A. Colmerauer 和 P. Roussel 在 1972 年设计而成；近几年来，英国爱丁堡大学在 PROLOG 开发和推广方面做了大量的工作；匈牙利人也很重视 PROLOG 的应用。目前，在世界上广泛流传的各种 PROLOG 文本主要来源于这三个地方。近来，不少美国计算机学家也正在把对 LISP 的兴趣转向 PROLOG。就其应用的广泛性来看，英国爱丁堡的 PROLOG 在世界上处于比较领先的地位，日本则有后来居上的趋势。

PROLOG 被广泛地应用于关系数据库系统、数理逻辑、抽象问题求解、自然语言理解、符号方程推导、系统结构设计、专家系统以及计算机辅助设计等各个不同的领域，在各种微型机上也纷纷推出 PROLOG，如 Micro-PROLOG。特别值得注意的是，1981 年 10 月日本宣布了研制第五代计算机——知识信息处理系统的十年规划，确定把 PROLOG 作为该系统的核心语言来研究和开发。当前，世界计算机界正在形成一股 PROLOG 热。

本书既全面而又深入浅出地介绍了 PROLOG 语言的特点和内容，详细论述了模型匹配和回溯等基本概念，逐一讨论了核心 PROLOG 所拥有的内部谓词，给出了 PROLOG 程序的调试方法，列举了大量的应用实例，同时还提供了有关 PROLOG 的比

较完整的附录。它是世界上第一本全面介绍PROLOG的教科书。但原书语言不够简炼，印刷错误较多，译者已作了必要的修正。

在全书翻译过程中，得到了总参第61研究所、北京航空学院计算机科学和工程系以及科学院数学所计算机科学研究室等很多单位和同志的帮助，在此表示感谢。

赵立平同志翻译了本书的第二章至第六章，李德毅同志翻译了其余部分，娄子勤同志对全书进行了校对。

由于我们对PROLOG的研究和使用不够，书中错误在所难免，请读者批评指正。

原 书 序

目前，计算机程序设计语言 PROLOG 正迅速在全世界普及。自从它 1970 年左右问世以来，越来越多的人把 PROLOG 用于符号计算的各个领域。这些领域包括：

- 关系型数据库系统；
- 数理逻辑；
- 抽象问题求解；
- 自然语言理解；
- 系统结构设计；
- 符号方程求解；
- 生物化学结构分析；
- 人工智能研究。

到目前为止，还没有一本专门把 PROLOG 作为实用程序设计语言的教科书。为了学习 PROLOG，许多人只好查阅简明参考手册和一些已经发表的论文，或者阅读各种手抄本。然而，在大量大学生和研究生都开始学习 PROLOG 的新形势下，就迫切需要一本关于 PROLOG 的教材，但愿本书的出版能在一定程度上满足这一需求。

许多初学者发现，编写 PROLOG 程序与用常规的程序设计语言描述算法不同，PROLOG 程序员要更多地弄清楚在他所要解决的问题中所出现的各种客体及客体之间的关系，还要弄清楚在他所期望的结果中，什么样的关系“为真”。因此，我们可以把 PROLOG 看作为一种**描述性**语言或**叙事式**语言。PROLOG 的方法是在一个问题中描述有关的已知事实和关系，而不是规定计算机在求解这一问题时必须采取的一系列步骤。对于 PROLOG

程序而言，计算机执行计算的实际过程部分取决于 PROLOG 的逻辑说明语义，部分取决于 PROLOG 能从给定的事实推演出什么新的事实来，只有一部分由程序员提供的明显的控制信息决定。

PROLOG 是执行许多“智能”程序的实用而有效的手段，例如非确定性问题、并发机理和面向模式的过程调用。该语言提供一种归一化的数据结构，称之为项，所有数据乃至 PROLOG 程序都由项构成。一个 PROLOG 程序由一组子句组成，每个子句要么是一件说明已知信息的事实，要么是一条规则，用以表明所要求的解答如何与给定的事实相关联，或如何从已知事实中“推导”出来。因此，PROLOG 可看作是朝向逻辑程序设计的最终目标迈出的第一步。本书不打算详细阐述逻辑程序设计的更广泛的含义，也不去论述为什么 PROLOG 不是最终的逻辑程序设计语言，而着重讨论如何使用现有的 PROLOG 系统编出实用的程序来。

无论是初学者还是有经验的程序员，都可以从不同的角度使用这本书。然而，该书的目的不是讲授程序设计技巧，因为我们认为，单凭看书或听讲是学不会程序设计的，只有通过亲自实践才能掌握。对于没有数学基础的初学者，本书可作为程序设计课程的教科书，他们最好能得到懂得 PROLOG 的程序员的指导。我们假定初学者可以通过在终端使用配有 PROLOG 系统的计算机，并具备有使用计算机终端的基本知识。对于有经验的程序员，本书完全可以作为自学用，而且不必受数学上的种种限制。这本书的前身曾用来教过只有中学数学基础的哲学和心理学专业的大学毕业生。

根据我们的经验，刚入门的程序员会觉得 PROLOG 似乎比用常规语言写出的等价程序还更好理解一些。他们不欣赏常规语言对使用计算资源所加的种种限制。相反，对常规语言已经很熟悉的程序员，似乎习惯于接受诸如变量和控制流等抽象概念，尽管他们有先前的经验，但仍可能感觉难以适应 PROLOG，也

许要等到取得大量的令人信服的事实之后，他们才把 PROLOG 当作有用的程序设计工具。当然，我们也知道，许多有丰富经验的程序员始终满腔热情地使用 PROLOG。不管怎么样，本书的目的不在于转变人们的心理，而是进行讲授。

如同其它大多数程序设计语言一样，PROLOG 也有许多不同的实现系统，它们具有各自的语义语法特色。在这本书中，我们采用一种“核心 PROLOG”，书中所有例题都遵从这一标准文本。该文本基于四个不同计算机系统，它们基本上都是在爱丁堡研制的。这四个系统是：DEC-10 机器上运行的 TOPS-10 操作系统；DEC-PDP-11 机器上运行的 UNIX 操作系统；DEC LSI-11 机器上运行的 RT-11 操作系统和 ICL2980 机器上运行的 EMAS 操作系统。DEC 计算机上的 PROLOG 大概流传得最为广泛。本书中的所有例题都可在这四个系统上运行。附录中列出了主要的 PROLOG 实现系统，并指出了它们与标准文本的差异，读者将会看到，大多数差异仅是表面上的不同而已。

本书的安排适合顺序阅读，不过，当读者着手写一个包含十个以上子句的 PROLOG 程序时，则有必要先看一下第八章。此外，书末的几个附录很值得一读，附录给出了 PROLOG 的特定实现，还介绍了如何输入程序，采用什么样的调试手段，以及其它的实用内容。浏览全书当然未尝不可，不过最好不要跳过前面的几章。

每章分为数节，节末常附有习题，建议读者多做练习，书末附有部分习题的答案。第一章指导性地介绍了 PROLOG 的基本概念，使读者对 PROLOG 程序设计有一个感性认识，建议认真阅读；有关 PROLOG 的基本要点在第二章作了较完整的论述；第三章讨论数据结构，引出几个简单的程序示例；第四章对“回溯”这个重要概念做了详细讨论，并引入了控制回溯的“切断”符号；第五章介绍输入输出方法；第六章逐一解释了核心 PROLOG 提供的内部谓词；第七章广泛收集了一些很有用的例题程序，并作了必要的解释；第八章讨论 PROLOG 程序的调试，提出了一

个新的控制流模型；第九章介绍了关于英语文法规则的语法，探讨在用文法规则分析自然语言的某些方面的设计决策；第十章阐述了 PROLOG 与数学定理证明以及逻辑程序设计的渊源关系，第十一章列举了许多课题项目，有兴趣的读者可借此施展自己的程序设计才华。

在此，我们感谢我们的老师 Rod Burstall，Peter Scott Langston 和 Robin Popplestone，他们对我们程序设计的思维方式给予了很大影响；还感谢在开发 PROLOG 使之成为实际可用的程序设计工具的通力合作中以及在本书的准备过程中曾经鼓励过我们的朋友们：Alan Bundy，Lawrence Byrd，Robert Kowalski，Fernando Pereira和David Warran。特别是Lawrence Byrd 从一开始就支持我们写这本书，提出了很多好的建议，供给了许多程序、例题以及列在第十一章中的应用项目；我们还要感谢帮助过我们的朋友们，他们对本书的草稿提出了宝贵的意见。他们是：Jon Cunningham，Richard O'keefe，Helen Pain，Fernando Pereira，Gordon Plotkin，Robert Rae，Peter Ross，Maxwell Shorter，Aaron Sloman 和 David Warren。此外，W. F. C 特别感谢 Epistemic 学校和爱丁堡大学人工智能系的研究生，他们是程序设计课程的教学对象。我们已经不再靠 PROLOG 的手抄本来编写程序了。对 PROLOG 的开发工作做出贡献的有些单位，本书可能未曾提及，在此我们表示歉意。

该书是作者在爱丁堡大学人工智能系工作时着手编写的，我们感谢系主任 Jim Howe 提供的所有方便。

W. F. C.

C. S. M.

于苏格兰 爱丁堡

1981.6.

目　录

给出许多用于各个领域的实用程序，例如表的处理、集的运算、符号微分以及公式化简等。

在前几章的基础上，读者已有能力写出很好的程序。本章讨论程序设计问题，包括控制流模型，程序中的常见错误和调试技术。

第一章　PROLOG简介

PROLOG 是一种计算机程序设计语言，用来解决各种客体(object) 和客体之间相互关系的问题。本章介绍 PROLOG 的核心思想，但不作详细地描述，不追求完整性和正规性。其目的是使读者很快地写出有用的 PROLOG 程序。为此，着重强调几个基本概念，即事实、问题、变量、合取(Conjunction)和规则。PROLOG 的其它特点，诸如表和递归等概念，将在以后的几章中介绍。

许多实际问题常常可以用若干客体和客体之间的相互关系的形式来表达，当希望计算机解决这类问题的时候，我们采用 PROLOG 语言。例如，我们说“约翰有这本书”．是在说明一个客体“约翰”和另一个客体“书”之间的所属关系，而且这种关系有一定的顺序：约翰有这本书，这本书并不拥有约翰！当提出问题：“约翰有这本书吗?”，那就是我们正在试图找出一种关系。

有些关系并不总是包含所有可能涉及到的客体。例如当我们说“宝石是贵重的”，意在说明一种和宝石有关的关系，称为“是贵重的”，我们并没有指明谁发现宝石贵重，也没有说为什么宝石贵重。所有这一切都取决于要说明的问题。当用 PROLOG 语言在计算机上对这样一些关系进行程序设计时，所提供的信息详细到什么程度，完全取决于用户想让计算机做些什么。

在编写程序之前，还有一点常识要说明。大家都熟悉用规则来描述客体之间的关系，例如：“如果两个人都是女的，又同父母，那么她们是姐妹俩”，这是一条规则，告诉我们“姐妹”的含义，还告诉我们怎样去辨别两个人是不是姐妹：只要检验他们是否都是女的并有共同的双亲。值得注意的是，人们常常把规则过分简化，但大家仍接受这类简化了的定义。当然，不能期待着一

条定义会包罗某件事物的所有方面。例如，大多数人都会同意在现实生活中作为姐妹关系其含义要远远超过上面的规则，然而当我们解决一个特定问题的时候，应该把精力集中在有助于解决这一问题的有关规则上，只要能给出足以解决我们问题的假想的和简化了的定义就行了。

用 PROLOG 语言进行计算机程序设计，包含三个方面内容：

- 说明有关客体和客体之间关系的若干事实；
- 定义有关客体和客体之间关系的若干规则；
- 就客体和它们之间的关系提出问题。

例如，假定已经告诉了 PROLOG 关于姐妹的规则，我们就不妨提出这样的问题："玛丽和珍妮是姐妹俩吗？"，PROLOG 将搜寻到目前为止已经告诉它的有关玛丽和珍妮的信息，根据这些信息回答"是"或者"不是"。因此，可以认为 PROLOG 是一个容纳事实和规则的知识库，它根据这些事实和规则来回答问题。PROLOG 的程序设计就是提供所有的事实和规则，而 PROLOG 系统使计算机能被用来作为存储事实和规则的知识库，并且提供能从一个事实推导出另外一个事实的若干推理方法。

PROLOG 是一种会话式语言，这就是说，借助它，用户同计算机可以进行某种形式的对话。假定用户通过计算机终端提出了使用 PROLOG 的请求（所用的这台终端由键盘和显示器组成），用户通过键盘给计算机打入字符，计算机通过显示器（屏幕或纸）把结果返回给用户。待打入所要解决问题的有关事实和规则后，用户可以提出适当的问题，PROLOG 将找出答案，并将答案呈现在显示器上。

现在逐一介绍 PROLOG 的基本要点。读者不必担心眼下对 PROLOG 的每一特点都没有完整概念，我们在以后的章节里将会给出全面总结和较多的例题。

1.1 事 实

首先讨论有关客体的事实。假定想告诉 PROLOG 这样一件

事："约翰喜欢玛丽"，这一事实由两个客体（称为"Mary"和"John"）和一个关系（叫做"likes"）组成。PROLOG要求用一个标准的形式来书写事实，这就是

likes (john, mary).

请注意以下几点：

·所有客体和关系的名称必须以小写字母开头，例如likes，john，mary。

·先写关系，后写客体，客体之间用逗号"，"分开，并用一对圆括号把所有客体括起来。

·用句号"."来结束一个事实。

当用事实来定义客体之间的关系时，必须注意圆括号内客体的书写顺序。这顺序本来是任意的，但一旦定下来，就要保持始终如一。例如在上面这个事实中，我们把喜欢者安排在圆括号里的第一项，被喜欢者放在第二项。于是这一事实likes(john，mary)就不同于另一事实likes (mary，john)。照目前的排列规定，前者说明约翰喜欢玛丽，后者表明玛丽喜欢约翰，如果真想表明玛丽喜欢约翰，必须明确给出：

likes (mary，john).

下面几个例子给出一组事实和它们可能表示的含义：

valuable(gold). "金子是贵重的"

female(jane). "珍妮是女人"

owns(john，gold). "约翰拥有金子"

father(john，mary). "约翰是玛丽的父亲"

每当用了一个名称，它就对应一特定的客体。人们一眼就可以看出john和jane是指两个特定的人。但是在其它一些事实中，用了名称gold，其含义不要求一目了然，不过当我们使用名称时，要确定如何来解释该名称。例如名称gold可能代表一个客体，此时我们把名称gold认作某一特定的金块，当用PROLOG叙述valuable(gold)时，我们的意思是说具有名称gold的这一特定金块是贵重的；另外，还可以用名称gold表示"金矿砂"，当我

们说 valuable(gold) 时，是指该金矿砂是贵重的。所以，对一个名称可能有多种理解，这完全取决于程序员自己如何去解释它，只要保持这种定义始终如一就行了。在上面这个事实中，我们认为 gold 是代表金矿砂，而其他人可用 gold 表示另外的含义，用 PROLOG 表达出来的事实则毫无差别。因此，应该尽早地说明不同含义之间的差别，搞清一个名称的真正含义。

现在我们讨论几个术语。在每个事实中，出现在括号里的客体称之为**元**（argument），关系的名称总是放在圆括号的前面，称之为**谓词**，例如 valuable 是一个谓词，该谓词有一个元。likes 也是谓词，它有两个元。

客体和关系的名称完全是任意的，例如用“a”表示“likes”，用“b”表示“john”，用“c”表示“mary”，就可以用 a（b，c）来代替规则 likes（john，mary）。当然，人们通常选择有助于记忆其含义的名称。总之，必须预先决定名称的含义和诸元的顺序，并从此保持其名称的一致性。

PROLOG 并不限制一个关系中包含多少个元。如果定义这样一个谓词“play”，用来表征两个选手和他们之间进行的一项运动，我们需要三个元。例如

```
play (john, mary, football).
play (jane, jim, badminton).
```

人们还可以定义一个不真实的事实，可以假定约翰是法国的现任国王：king（john，france）. 这里 king 代表“是国王”这样一个关系，这当然不是真实的，法国在 1792 年左右就废除了君主制。但是 PROLOG 不关心这些，PROLOG 中的事实允许表达客体之间的任意关系。

在 PROLOG 系统中，一组事实称为**数据库**或**知识库**[1]。无论什么时候，当我们搜集了一组事实（包括以后要提及的规则）用于解决一个特定的问题时，我们就使用数据库或知识库这个术语。

[1] 这里所指的数据库，除了含有数据外，还包含有资料或信息。人们一般把事实的集合称为数据库，把事实和规则的集合称为知识库。——译者注