应用于人工智能的 PROLOG 程序设计

〔南斯拉夫〕I. 布拉特科 著

科学出版社

应用于人工智能的PROLOG程序设计

〔南斯拉夫〕I. 布拉特科 著

夏 莹 陈群秀 译

黄昌宁 校

科学出版社

1991

内 容 简 介

本书是关于 Prolog 程序设计的一本成熟的、有趣的书籍，是一本专门介绍 Prolog 语言在人工智能中应用的书. 由于作者对 Prolog 语言有透彻的理解，所以书中提出了关于 Prolog 程序设计的许多深刻见解，并提供了大量生动的实例.

全书分两部分，共十六章. 第一部分(前八章)介绍 Prolog 语言；第二部分(后八章)介绍 Prolog 在人工智能中的应用，其中包括人工智能中诸如问题求解和启发式搜索、专家系统、博弈以及模式导引系统等许多基本技术和应用. 本书各章节后还附有一定数量的有关重要概念的习题，并在书末附有相应的解答.

本书适宜作为高等院校计算机系程序设计和人工智能课程的教学参考书，也可供有关专业的科技人员阅读.

Ivan Bratko
PROLOG PROGRAMMING FOR ARTIFICIAL INTELLIGENCE
Addison-Wesley 1986

应用于人工智能的 PROLOG 程序设计
〔南斯拉夫〕I. 布拉特科 著
夏 莹 陈群秀 译
黄昌宁 校
责任编辑 那莉莉
科学出版社出版
北京东黄城根北街 16 号
邮政编码：100707
中国科学院印刷厂印刷
新华书店北京发行所发行 各地新华书店经售
*
1991年 8 月第 一 版 开本：850×1168 1/32
1991年 8 月第一次印刷 印张：14 1/2
印数：0001—3 000 字数：375 000
ISBN 7-03-002298-X/TP · 168
定价：12.50 元

译 者 的 话

《应用于人工智能的 Prolog 程序设计》一书的作者是南斯拉夫 Kardelj 大学的 Ivan Bratko 教授，他也是图灵研究所的访问学者，曾在世界各地讲授过 Prolog 程序设计．我们认为这本书有很多特点，值得翻译给国内读者．

1. 由于作者对 Prolog 语言有透彻的理解，所以他对 Prolog 程序设计提出了许多深刻的见解，并提供了大量的实例，使初学者和已经懂得 Prolog 语言的读者受益匪浅．譬如，作者在第一章 1.5节中提出的 Prolog 程序的陈述含义和过程含义这两个概念就是极其精辟的，在其它同类书中没有见到．

2. 本书是世界上第一本专门介绍 Prolog 语言在人工智能中应用的书籍，其第一部分（前八章）介绍 Prolog 语言，第二部分（后八章）介绍 Prolog 在人工智能中的应用，其中包括了人工智能中诸如问题求解和启发式搜索、专家系统、博弈以及模式导引系统等许多技术和应用．因此许多高等院校人工智能的课程可以选用它作为参考书，以便向学生介绍许多人工智能的程序实例，这显然将提高学生对这个领域的兴趣并加深他们的具体认识．

3. 本书是 Prolog 程序设计的一本成熟的、有趣的教科书．它不仅深入浅出地阐明了 How 型思想（How 型语言）和 What 型思想（What 型语言）之间的差别，而且用一整章（第八章）篇幅讨论了程序设计的风格和技巧．更为突出的是从第一章到第十六章源源不断地提供了丰富的实例，各章节后还附有一定数量的有关重要概念的习题，并在书后附有相应的解答．

本书由清华大学计算机科学与技术系夏莹和陈群秀翻译，由黄昌宁教授校阅．陈群秀翻译序、前言、第一部分和索引，夏莹翻

译第二部分和练习选答。

由于译者水平所限，译文中难免有不妥之处，欢迎读者批评指正。

1990 年 7 月 4 日

序

在中世纪，拉丁文和希腊文的知识对所有学者来说，都是必不可少的．只懂一种语言的学者必然是一个残缺不全的学者，他缺乏从两个方面来观察世界所获得的那种理解力．同样，现代从事人工智能的专业人员如果不能同时大致通晓 LISP 语言和 Prolog 语言，也犹如一个残疾人，因为就广义来说，这两种主要的人工智能语言的知识都是必不可少的．

我一直热衷于 LISP，LISP 是在美国麻省理工学院被创造并且在那儿成长起来的．尽管如此，我还是从未忘记，当眼看着我的第一个 Prolog 风格的程序运转时的那种激动心情．这个程序是 Terry Winograd 的著名 Shrdlu 系统的一部分，其积木世界问题求解器安排了一只模拟的机械手在屏幕上移动积木，根据人指定的目标解决复杂的问题．

Winograd 的积木世界问题求解器是用 Microplanner 写的，现在看来这种语言与 Prolog 同属一类．虽然 Microplanner 有缺陷，但人们还是根据目标明确地编制了积木世界问题求解器．因为具有 Prolog 风格的语言鼓励程序编制者根据目标去思考．用面向目标的过程来表示抓住、清除、摆脱、移动和放开等动作，这就有可能使得一个清晰、透彻、简洁的程序显得智能超群．

Winograd 的积木世界问题求解器彻底改变了我考虑程序的方式．我甚至为我的 LISP 教科书用 LISP 重写了积木世界问题求解器，因为那个程序使我对面向目标的程序设计思想的能力和编写这种程序的乐趣有深刻的印象．

但是通过 LISP 程序来学习有关面向目标的程序设计，有点像用英语以外的语言来阅读莎士比亚的著作——虽有某些精彩的片段，但不像原著那样带劲．同样地，学习面向目标的程序设计的

最好方法是用 Prolog 来读写面向目标的程序，因为面向目标的程序设计正是 Prolog 的宗旨.

概括地说，计算机语言的发展正是一个由 How 型低级语言向 What 型高级语言进化的过程. 在 How 型语言中，程序编制者必须详细地说明运算是怎样（How）一步一步地进行的；而在 What 型语言中，程序编制者只需简单地说明要做的事情是什么（What）. 以编写 FORTRAN 程序为例，程序编制者不再被迫地用地址和寄存器一类的低级语言来同计算机对话. 取而代之的是，FORTRAN 程序编制者能够用他们自己的语言来说话，或几乎差不多如此，即通过采用一种一维 80 列空间的标记来作出仅有的让步.

但是，FORTRAN 和几乎所有其它语言都仍然是 How 型语言. 我认为，现代的 LISP 是这些语言中的佼佼者，因为采用 Common LISP 格式的 LISP 具有非凡的表现力，但是如何做某件事情仍然有待于 LISP 程序编制者来表达. 相反，Prolog 是一种明显冲破了 How 型语言陈规的语言，它鼓励程序编制者去描述情况和问题，而不是那些用来解决问题的详细步骤.

因此，对于学习计算机科学的学生而言，了解一下 Prolog 是很重要的，因为舍此不再有别的更好的途径来认识 What 型程序设计的概念.

特别是本书清楚地阐明了 How 型思想和 What 型思想之间的差别. 譬如，在第一章中，作者通过涉及家庭关系的问题来说明这种差别. Prolog 程序编制者用清晰、自然的术语直截了当地描述了祖父（grandfather）的概念：祖父是父母亲（parent）的父亲（father）. 下面是这一概念的 Prolog 表记：

```
grandfather(X,Z) :- father(X,Y), parent(Y,Z).
```

一旦 Prolog 知道了祖父是什么，就很容易向它提出问题：例如，谁是 Patrick 的祖父？ 下面是这一问题的 Prolog 的表记及其特有的回答：

```
?- grandfather(X,Patrick).
```

X = james;

X = carl

Prolog 的任务正是通过对含有 father 和 parent 关系的一个知识库的搜索来解决这个问题的. 程序编制者只需要说明已知的是什么和要解决的问题是什么，从而使程序编制者可以较多地去考虑知识，而较少地去考虑运用这些知识的算法.

我们已知学习 Prolog 是重要的，下一个问题就是如何学的问题. 我相信，在许多方面，学习程序设计语言类似于学习自然语言. 例如，在学习程序设计语言中参考手册是有用的，正如在学习自然语言中一本词典是有用的一样. 但是，没有一个人可以仅借助一本词典而学好自然语言，因为词汇仅仅是必须学习的一部分. 一个学习自然语言的学生必须学习如何把这些词汇合法地放在一起的那些惯例，随后，学生还应当学习以某种风格把这些词组合在一起的技巧.

同样地，没有一个人可以仅仅根据一本参考手册来学习一种程序设计语言. 因为一本参考手册很少或根本不会谈及精通这种语言的人怎样使用该语言原始材料的那些方法. 正因为这样，才需要教科书，好的教科书将提供丰富的实例，因为好的实例是经过精炼的经验，而我们主要是通过经验来学习的.

在本书中，第一个实例出现在第一页，其余的部分源源不断地给出一个热情的 Prolog 程序编制者所编写的 Prolog 程序，而这个 Prolog 程序编制者忠实于 Prolog 的观点. 通过仔细地学习这些实例，读者不仅可以掌握这种语言的机理，而且可以获得一批个人收藏的先例，这些实例可以随时被拆开、改写和重新装配在一起以形成新程序. 在获得这些前人知识的同时，便开始了从初学者到熟练程序编制者之间的过渡.

当然，一个好的程序设计实例除了程序设计本身外，还展示了一点儿有趣的科学. 在本书中，实例后面的科学是人工智能. 读者可以学到有关问题求解的思想，诸如问题归约、正向和逆向推理、“how” 和 “why” 的询问以及各种各样的搜索技术.

事实上，Prolog 的最大特色之一是它简单得足以使学习人工智能导论课的学生能直接地学习使用．我期望许多教师采用本书作为人工智能课程的部分讲义，以便使他们的学生能够看到抽象的思想怎样被直接地简化为具体、生动的形式．

在 Prolog 的教科书中，我预见这本书将会特别普及，这不仅是因为它的实例，而且也因为许多其它的特色，例如：

· 细致的提要贯穿全书．
· 众多的练习巩固了所有的概念．
· 结构选择器介绍了数据抽象化的概念．
· 占有一整章关于程序设计风格和技巧的明晰讨论．
· 除了 Prolog 程序设计中的乐趣外，对 Prolog 程序设计中所遇到的其它问题也给予了客观的描述．

上述特色使本书成为一本成熟、有趣和有教益的书．

P. H. 温斯顿

马萨诸塞州，坎布里奇

1986 年 1 月

前　言

Prolog 是一种以一组为数不多的基本机制为中心的程序设计语言，这些基本机制包括模式匹配、基于树的数据结构和自动回溯．它们形成了一个极为有效和灵活的程序设计骨架．Prolog 特别适合于含有对象（特别是结构化对象）和对象之间关系的问题．例如，Prolog 的一个较容易的习题是表达本书封面上图示的空间关系——譬如，顶上的球是在左边那个球的后面．阐述一条比较一般的规则也不难：如果 X 比 Y 更接近于观察者，并且 Y 比 Z 更接近于观察者，那么 X 必定比 Z 更接近于观察者．现在 Prolog 就能够推论与这条一般规则有关的空间关系和它们的前后位置．这样的特性使 Prolog 成为一种应用于人工智能和一般的非数值程序设计的有效语言．

Prolog 是 Programming in Logic 的缩写，它表示用逻辑作为程序设计语言．这一思想早在 1970 年就出现了，其最早的开发者包括爱丁堡的 Robert Kowalski（在理论方面）、爱丁堡的 Maarten van Emden（实验性论证）、马赛的 Alain Colmerauer（实现）．现在流行的 Prolog 基本上应该归功于 70 年代中期爱丁堡 David Warren 的高效率实现．

因为 Prolog 在数理逻辑中有它的根，所以经常通过逻辑介绍它．但是，如果把 Prolog 作为一种实用的程序设计工具来讲授，那么如此数学化的介绍则不是很好．因此，本书不打算涉及数学方面，而是集中在如何通过应用 Prolog 的少数基本机制来解决有趣问题的那种技巧上．鉴于传统的语言是面向过程的，Prolog 引入了描述性（或陈述性）的观点．这一点大大地改变了思考问题

的方式，并且使研究 Prolog 程序成为一种激动人心的智力挑战.

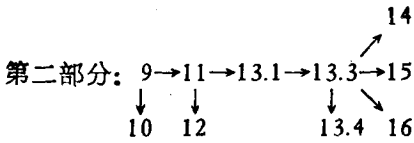
本书的第一部分介绍 Prolog 语言并说明如何编写 Prolog 程序；第二部分用实例说明应用于人工智能某些核心领域的 Prolog 的功能，这些核心领域包括问题求解和启发式搜索、专家系统、博弈以及模式导引系统.本书首先介绍了人工智能的基本技术，然后朝着其 Prolog 实现的深度发展，最终形成一些完整的程序.这些程序可以在很复杂的应用中作为建筑构件来使用. 这部分还收入了处理像树和图这类重要数据结构的技术，虽然严格说来它们并不属于人工智能. 这些技术经常用在人工智能程序中，而且它们的实现有助于学习 Prolog 程序设计的一般技巧. 本书的侧重点将始终放在程序的清晰度上；对依赖于 Prolog 实现特性的某些效率方面的技巧则尽量加以回避.

本书是供学习 Prolog 和人工智能的学生用的. 它可以作为 Prolog 或人工智能课程的讲义.在这些课程中，人工智能的原理通过 Prolog 显得更具活力.本书假定读者具有计算机的基本知识，但不需要人工智能知识，也不需要专门的程序设计经验；事实上，对传统的过程性程序设计（例如 Pascal）的丰富经验和执著，甚至可能是对以崭新方式去思考 Prolog 的要求的一种障碍.

在几种 Prolog 版本中，爱丁堡句法（又称为 DEC-10 句法）是最普及的，因此本书中选用的也是它. 为了同 Prolog 的各种实现兼容，本书仅仅采用许多 Prolog 共享的一个相对来说比较小的内部谓词子集.

怎样读这本书呢?在第一部分，自然的阅读顺序与本书的顺序一致. 但是，可以跳过用比较形式化的方法来描述 Prolog 的过程性意义的 2.4 节. 第四章介绍的程序设计实例可有选择地阅读（或跳过去）. 第二部分可以比较灵活地阅读，因为这些章节故意写成互相独立的. 但是，某些专题将仍然自然地写在其它章节之前，例如，数据结构的基础（第九章）和基本的搜索策略（第十一章和第十三章）. 下面的图表概括了对自然阅读顺序的约束：

第一部分： 1 → 2 → 3 →4(可选择)→ 5 → 6 → 7 → 8



历史上一直对 Prolog 存在着某些争议. 作为一种实际的程序设计工具，Prolog 首先在欧洲赢得众望. 在日本，Prolog 被看成是第五代计算机发展的核心. 而由于某些历史上的原因，在美国对它的承认被推迟了.原因之一是美国人早先使用 Microplanner 语言,而 Microplanner 也具有类似于逻辑程序设计的思想，但执行效率低.Microplanner 的这种缺点被不公正地归结到 Prolog 上,但是后来经 David Warren 对Prolog 的有效实现而被纠正了过来. 对 Prolog 的保留还由于对逻辑程序设计"正统学派"的反作用."正统学派"坚持只使用纯逻辑,以免由于添加与逻辑无关的实用工具而使纯逻辑遭到破坏. 这一不妥协的立场被 Prolog 专业人员改变了,他们采用更加实用的观点,从陈述性方法与传统的过程性方法的结合中获益. 延迟接受 Prolog 的第三个原因是，长期以来 LISP 在用于人工智能的语言中没有真正的对手.所以，在具有强大的 LISP 传统的研究中心，存在着一种对 Prolog 的自然抵制. Prolog 相对于 LISP 的这种困境这些年来已经得到缓解,现在许多人相信,这两种语言各有所长，可以很好地相互补充.

致　谢

Donald Michie 是第一个引起我对 Prolog 发生兴趣的人. 为了 Lawrence Byrd, Fernando Pereira 和 David H. D. Warren 对程序设计的建议和多次讨论,我得感谢他们，他们曾经是爱丁堡(大学) Prolog 研究小组的成员. 本书还大大得益于 Andrew McGettrick 和 Patrick H. Winston 的评述和建议. 阅读过部分原稿并提出重要意见的有：Igor Kononenko, Tanja Majaron, Igor Mozetic, Timothy B. Niblett 和 Franc Zerdin. 我还想感谢 Addison-Wesley 的 Debra Myson-Etherington 和 Simon

Plumtree、他们在编著本书的过程中做了大量的工作。最后，还要说明一下，如果没有国际逻辑程序设计界开创性的推动，是不可能写出这本书的。

I. 布拉特科
格拉斯哥，图灵研究所

1986 年 1 月

目　　录

第一部分　PROLOG　语言

第二部分 PROLOG 在人工智能中的应用