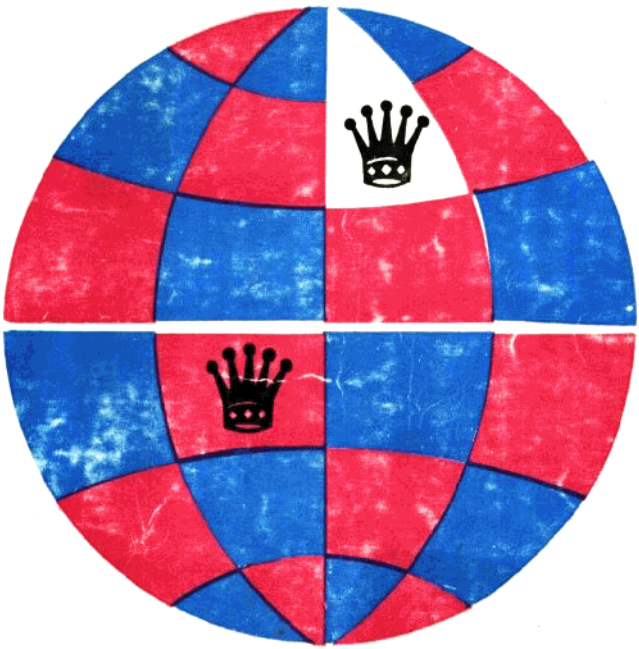
PROLOG
程序设计基础与技巧

陈怀谟　刘毅　孙焕东　编著



PROLOG

前　言

逻辑程序设计语言 PROLOG 是一种新型的人工智能程序设计语言．人工智能、专家系统以及第五代计算机都与 PROLOG 语言有着密切的关系．由此可见这种语言的重要性．

在人工智能发展的早期阶段，人们追求的目标是发现人类智能行为的普遍规律，研究工作集中在用计算机下棋以及证明数学定理等问题上．他们认为，一旦将智能行为的普遍规律从这些典型的智能问题中分离出来，就可以将它们应用于其它需要运用智能的工作中去．就智能行为来说，普遍规律是必要的，但是还远远不够，就象对一个智慧人来说，"聪明"是否已包含了他有知识?除开聪明外，更重要的是他有知识，而且是大量的专业知识．于是，人工智能的研究发生了根本性的转变，人们不再去追求智能行为的普遍规律，而是转向以知识为中心的系统．人工智能转向以知识为中心便诞生了专家系统．专家系统是一个高性能的计算机程序，它拥有某专业领域的专家级的知识以及使用这些知识求解问题的方法．它的工作方式从外观来看很类似人类专家．构成专家系统的基本成分是知识库和推理机制．

人类的智能行为以知识为中心的思想不只是产生了专家系统，而且使机器翻译、自然语言理解、机器人学以及机器学习等领域的研究发生了根本转变．的确，人的所有行为如理解、求解问题、乃至学习能力都是以知识为基础的．人必须先有知识才能去理解，才能知道更多的东西．

在对 90 年代社会上需要什么样的计算机进行了大量的调查研究之后，1981 年日本宣布了他的第五代计算机计划，震动了世界．该计划的目标是用 10 年建立第五代计算机系统的样机——知识信息处理系统．第五代计算机的核心语言是 PROLOG．在众多

的计算机程序设计语言中，日本人偏偏选择了 PROLOG 作为第五代计算机的核心语言，这是因为，PROLOG 语言适合于设计知识库、专家系统，适合于表达知识、管理知识以及在知识上进行推理；它提供改进的程序设计方法，只要说明做什么，不必象传统的做法那样说明如何做。

逻辑程序设计语言 PROLOG 1972 年诞生于法国马赛大学，后来在英国得到完善和发展。真正懂得它的原理的人都认为这是计算机科学的重大发展。学习用 PROLOG 语言写程序时会感到象第一次学写程序那样兴奋。PROLOG 程序设计显示了一种新的编程思维方式和一种描述问题的新方法，这种方法极其有效和优雅。PROLOG 程序与传统的计算机语言程序差别甚大，它由事实和规则构成，执行它的是"定理证明器"。PROLOG 系统的搜索、匹配以及回溯等机制用到了人类思维的某些机理。用 PROLOG 语言实现专家系统以及其它的人工智能系统要比用其它传统计算机语言快得多。

本书是作者多年从事 PROLOG 教学、研制、开发及应用的经验总结。全书共分十章。第一章概述 PROLOG 语言的发展、特点以及一些基本概念。第二章介绍 PROLOG 语言的语法。第三章详细叙述一致化匹配——模式匹配，从不同的角度，全面、系统地描述 PROLOG 程序的执行原理。第四章详细陈述 PROLOG 控制机制的作用、作用原理、一般用法以及副作用。第五章叙述 PROLOG 内部谓词的特点，以及几类常用的内部谓词。第六章通过大量的程序设计实例来阐述用 PROLOG 进行程序设计的常用方法。第七章指出如何利用模式匹配、回溯等 PROLOG 自身的机制。第八章阐述尾递归程序设计、数据结构的选择、全局变量以及多维结构的表示。第九章用由浅入深、循序渐进的方式叙述各种元级解释器的实现，其中包括专家系统工具、智能程序调试环境以及不精确推理等元级解释器的实现。第十章介绍提高 PROLOG 程序效率的一些原则和方法。

本书第一、二、三、六、七、八章以及附录由陈怀谟编写，第

四、五章由陈怀谟和刘毅合编，第九、十章由孙焕东和陈怀谟合编。全书的总体构思、各章内容的统编以及程序的调试由陈怀谟完成。俞咸宜教授全面、仔细地审阅了本书，并提出了许多宝贵意见，齐治昌教授、王广芳教授、刘凤岐教授就全书的内容与结构曾提出过指导性建议，在此一并表示感谢！

作　者

1990 年 12 月

目　录

第一部分　PROLOG 程序设计基础

第二部分 程序设计方法与技巧

第一部分　PROLOG 程序设计基础

要学会用 PROLOG 语言进行程序设计，首先需要掌握如下三方面的知识：

1. PROLOG 语言的语法。

2. PROLOG 程序的执行过程以及对执行过程的控制。

3. PROLOG 系统提供的内部谓词，相当于 BASIC、FORTRAN 和 PASCAL 等语言中的内部函数（如：SIN(X)，SQRT(X)，LGE(X，Y)，…）。

第一章　概　　述

在计算机发展的 40 多年间，传统的程序设计语言也相应地经历了由低级到高级，即由机器语言、汇编语言到 FORTRAN 和 PASCAL 等高级语言的发展过程。尽管这些高级语言的使用大大地改善了人机之间的联系，但人类自然语言与计算机语言仍然相差很大。传统的程序设计语言要求程序员告诉计算在解决某一问题时"怎样去做"，即把解决问题的全部过程细节告诉计算机。这使得软件的设计与实现面临许多困难，如：需要耗费大量的人力和资金，设计软件的工作极其烦琐，软件庞大、容易出错并且难以调试等。这样研制的软件不但昂贵而且可靠性低。如何提高软件的可靠性？如何把软件设计人员从繁重的软件开发中解脱出来？这已经成为迫切需要解决的问题。

为了缩小人类的要求与机器求解问题之间的差距，为了缩小人类自然语言与计算机程序设计语言之间的差距，人们开发了一些新型的程序设计语言，比较典型的有函数程序设计语言、逻辑程序设

计语言以及面向对象的程序设计语言．逻辑程序设计语言描述问题的方法已经向人类的自然语言靠近了一大步．PROLOG 语言是一种典型的逻辑程序设计语言．利用 PROLOG 语言进行程序设计只需要程序员写出与问题有关的事实、规则以及要求解的具体问题，而不再象传统的程序设计语言那样，程序员必须一步不漏地把求解问题的全部过程细节都编写出来，即告诉计算机如何求解问题．由此可看出，这种新型程序设计语言的特点和针对上述迫切需要解决的问题所产生的效益。

§1.1 PROLOG 语言的发展

70 年代初，英籍计算机科学家 R.Kowalski 提出了一整套逻辑程序设计的思想．这种思想的核心是在一阶谓词逻辑的基础上，用句型（主要是 Horn 短句）对问题进行描述．

1972 年，A.Colmerauer 和 P.Roussel 在马赛大学首次按照 Kowalski 的思想设计了 PROLOG 语言，并且在 IBM-360 计算机上用 ALGOL-W 语言实现了世界上第一个 PROLOG 解释程序．PROLOG（PROgramming in LOGic）是逻辑程序设计的缩写，表示用逻辑进行程序设计．

在 PROLOG 诞生后的相当长一段时间里，尽管人们又开发了各种版本的 PROLOG，如：Warren 等人的爱丁堡 DEC-10 PROLOG，Robents 的滑铁泸 PROLOG，Clark 等人的 IC-PROLOG 等等，但是，它并没有受到人们的足够重视．

1981 年 10 月，日本宣布了它的第五代计算机研制计划，并将 PROLOG 语言作为其核心语言——第五代计算机的机器语言．从此，PROLOG 就举世瞩目，各国无不在竞相研究与开发．

1984 年，英国逻辑程序有限公司率先推出了商品化的 PROLOG 系统——Micro-PROLOG 3.1．它使得世界上成千上万台微机都装上了 PROLOG 语言．人们用它实现了一些有价值的系统，如：立法专家系统，建筑设计专家系统等．在英国，它还作

为儿童学习计算机的入门语言。

1986 年，美国 Borland 公司推出了速度很快的 Turbo-PROLOG 1.0 系统。同年，我国研制出的 PROLOG 系统有 YHW-PROLOG 1.0 中西文兼容系统，GKD-PROLOG 和 H-PROLOG 等。YHW-PROLOG 1.0 可运行在 IBM-PC 及其兼容机上。它兼容 Micro-PROLOG 3.1，提供给用户的工作空间是机器的全部内存（除操作系统所占部分），并且速度是它的两倍。我们已经用 YHW-PROLOG 1.0 实现了很多有价值的实用系统，如：通用专家系统建造工具，中医诊疗专家系统等等。

今天，从个人计算机到工作站，从超级小型计算机到大型计算机，直至巨型计算机，所有流行的计算机上都配有相当好的 PROLOG 解释系统或编译系统。随着人们对 PROLOG 的进一步深入研究，它的功能在不断加强，实现技术也得到进一步的改进。如模块化、以及通过对 PROLOG 源程序进行数据流分析、对确定性问题进行优化，使 PROLOG 程序的运行效率得到进一步提高。

尽管如此，在现行的计算机上提高 PROLOG 的效率是有限度的，原因是现行的计算机体系结构不太适应 PROLOG 的高效实现。因此，必须研制直接面向 PROLOG 的计算机。1984 年，日本推出了第一台顺序 PROLOG 计算机 PSI。目前，PSI 已经商品化。前不久，PROLOG 芯片在美国问世。当今，高效并行 PROLOG 计算机的研究正在世界各国蓬勃开展。

§1.2 PROLOG 语言的特点

PROLOG 是一种描述性的计算机程序设计语言。由于 PROLOG 的描述性风格，所以在传统的程序设计中应用的流图及编程技术大部分已不再适用。虽然 PROLOG 的事实和规则是在现有的计算机上执行的，这使得 PROLOG 程序的执行具有一定的过程性，但它已不再是传统程序中的线性流，而是回溯搜索执行。在传统的过程性语言中具有下述语句：

- 赋值语句
- goto 语句
- if-then-else 语句
- do 循环、for 循环和 while 循环语句

在 PROLOG 语言中并没有上述语句。最令人惊奇的是，在 PROLOG 语言中没有赋值语句，在 PROLOG 程序中不可能彻底改变变量的值。如果出现

$$X = X+1$$

这样的语句，结果将总是假的，因为没有 X 会等于 X+1。所以，程序员在用 PROLOG 语言进行程序设计时，必须抛弃传统的过程习惯，掌握描述性的思想方法。与传统的计算机语言相比，PROLOG 语言具有下列特点：

1. PROLOG 语言是一种智能型的描述语言。用传统的高级语言求解问题，程序员必须写出求解问题的全部过程细节，即告诉计算机如何求解问题。用 PROLOG 语言求解问题，只需程序员告诉计算机要求解的问题是什么。

2. PROLOG 语言接近于自然语言，语句种类少，只有事实、规则和目标三种语句，易于学习和掌握。

3. 用 PROLOG 语言中的事实和规则可方便地表达知识，元逻辑操作使得存取、管理以及处理知识更加方便、灵活。

4. 程序与数据统一。

5. 逻辑性强。PROLOG 语句采用逻辑表达式形式，所以，易于保证 PROLOG 程序的正确性。

6. PROLOG 程序的执行以归结原理为基础自动地实现对问题的求解，从而使它具有超越传统高级语言的功能。因此，人们也称 PROLOG 为超高级语言。

PROLOG 语言的这些特点使得它更适合于如下领域：

- 专家系统
- 自然语言理解
- 演绎数据库

· 规划生成

· 军事决策、指挥与战争模拟

· 办公自动化以及各种智能管理系统等

在这些领域，程序员用 PROLOG 编写一个实用软件所花的时间要比用传统高级语言少得多.

§1.3 PROLOG 程序

下面通过给出两个 PROLOG 程序来阐述 PROLOG 语言中的几个概念以及 PROLOG 程序的结构.

1. 阶乘

非负整数的阶乘用数学公式定义如下:

$$\begin{cases} 0! = 1 \\ N! = N \cdot (N-1)! & N > 0 \end{cases}$$

"0!=1"可以用 PROLOG 语言表示为:

```
阶乘(0, 1).
```

这就是一条 PROLOG 语言的语句. 它可以读作: "0 的阶乘是 1."象这种无条件成立的结论称之为事实或事实短句.

当 N 为大于 0 的整数时，N!=N·(N−1)!. 这可以用 PROLOG 语言表示为:

```
阶乘(N, X)    如果
        N>0   并且  int(N)  并且
        M=N − 1     并且
        阶乘(M, Y)     并且
        X=N * Y.
```

这也是一条 PROLOG 语言的语句. 其中: >、int 以及 = 是内部谓词. 对于这些内部谓词，在此只需了解: 当 N 大于 0 时，条件"N>0"成立; 当 N 为整数时，条件"int(N)"成立; 当 M 等于 (N−1) 时，条件"M=N−1"成立; 当 X 等于 (N·Y) 时，条件

"X = N　Y"成立.

上述语句可以读作:

如果　N 大于 0　并且
　　N 是整数　并且
　　M 等于 N 减 1　并且
　　M 的阶乘是 Y（即 Y = (N−1)!)　并且
　　X 等于 N 乘 Y（即 X = N · (N−1)!)
那么　N 的阶乘是 X

象这种可以读作"如果…那么…"的 PROLOG 语言语句称为规则或规则短句. 也有人将短句称为子句.

在上面的规则中, N、M、Y 和 X 并不特指某个数, 它们都是 PROLOG 语言中的变量."如果"以及"并且"是联结词, 它们相当于传统高级语言中的保留字."阶乘"是谓词符号, 它相当于传统高级语言程序中的过程名.

上面的两条短句就构成一个 PROLOG 程序——程序 1-1.

程序 1-1

```
阶乘(0, 1).
阶乘(N, X)  如果
        N>0  并且  int(N)  并且
        M=N - 1  并且
        阶乘(M, Y)  并且
        X=N*Y.
```

这个程序是由一条事实和一条规则组成的. 它描述了非负整数的阶乘是什么. 一般地, 一个 PROLOG 程序由多条事实和规则组成.

实际上, 程序 1-1 中的两条短句定义了谓词原子

阶乘(<参数 1>, <参数 2>)

的含义——第二个参数的值是第一个参数的值的阶乘. 即当第一个参数的值是 0 时, 第二个参数的值是 1 (=0!); 当第一个参数的值是大于 0 的整数时, 第二个参数的值是第一个参数的值乘以它的值减 1 的阶乘. 由于谓词原子"阶乘(<参数 1>, <参数 2>)"的

含义集中体现在谓词符号"阶乘"上，所以，也称程序 1-1 中的两条短句定义了谓词"阶乘".

有了程序 1-1，若要求某个数的阶乘，则可以使用 PROLOG 语言中的目标短句．目标短句简称为目标，也有人将它称为询问.

? 阶乘(4, X).

就是一个目标．该目标在程序 1-1 上的执行将求得 X = 24.

2. 简单的家族关系

图 1-1 给出了荣国府贾家近几代的家谱.

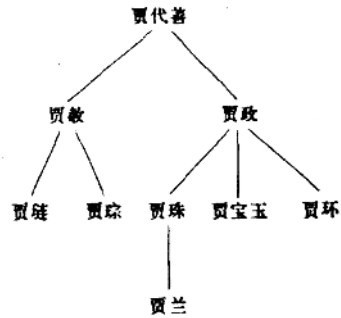


图 1-1 贾家近几代家谱

图中的每一条连线都表示连线上方的人是连线下方的人的父亲，即连线下方的人是连线上方的人的儿子．例如，贾代善与贾赦之间的连线就表示贾代善是贾赦的父亲，即贾赦是贾代善的儿子．所有的"父亲——儿子"关系对由 8 条连线表示出来．这些父子关系对可以用 PROLOG 语言表示如下:

```
fuzi (jiadaishan,  jiashe).
fuzi (jiadaishan,  jiazheng).
fuzi (jiashe,      jialian).
fuzi (jiashe,      jiacong).
fuzi (jiazheng,    jiazhu).
fuzi (jiazheng,    jiabaoyu).
```

fuzi (jiazheng, jiahuan).
fuzi (jiazhu, jialan).

这就是一个 PROLOG 程序．它由 8 条事实组成．这些事实描述了贾家成员之间的父子关系．一条事实描述一个"父——子"关系对．例如，事实

fuzi (jiadaishan, jiashe).

描述了贾代善与贾赦之间的父子关系，其中：汉语拼音 jiadaishan 表示贾代善这个人，jiashe 表示贾赦这个人．jiadaishan 和 jiashe 可以理解为在 PROLOG 程序中我们给贾代善和贾赦这两个人取的名字，称它们为对象名．fuzi 是我们给父子关系取的名字，称它为谓词符号、谓词名或者关系名，有时也简称它为谓词或关系．

谓词符号和对象名是可以任意选取的．在上面的程序中，选取了"父子"的汉语拼音 fuzi 作为谓词符号，即作为父子关系的名字，当然也可以选取其它的（如选取 a 也行）．但是，在整个程序中，一个关系只能取一个名字．一旦选定 fuzi 作为父子关系的名字，在整个程序中就不能再选其它的作为父子关系的名字．对象名的选取同谓词符号的选取一样．

虽然谓词符号和对象名可以任意地选取，但是，在选取时最好遵循"自然、有助于理解并且便于记忆"的原则．在上面的程序中，选取了 fuzi 作为谓词符号，人名的汉语拼音作为对象名．对于母语为汉语的中国人来说，这就不太符合上述原则．如果选取"父子"作为谓词符号，人名本身作为对象名，那么，用事实

父子(贾代善，贾赦).

描述贾代善与贾赦之间的父子关系就要自然、明了得多．它可以读作："贾代善是贾赦的父亲．"或者"贾赦是贾代善的儿子．"用这种形式的事实描述荣国府近几代成员之间的父子关系的程序如下：

程 序 1-2

```
父子(贾代善，贾赦).
父子(贾代善，贾政).
父子(贾赦，  贾琏).
父子(贾赦，  贾琮).
```

```
父子(贾政,  贾珠).
父子(贾政,  贾宝玉).
父子(贾政,  贾环).
父子(贾珠,  贾兰).
```

这些短句描述了"父子"关系，也称它们定义了谓词"父子"。

从描述父子关系的事实以及程序 1-1 中描述"0 的阶乘是 1."的事实可以看出，每一条事实都描述一个无条件成立的结论——客观存在。事实一般具有如下形式：

＜谓词符号＞(＜参数 1＞,＜参数 2＞,…,＜参数 n＞).

其中：参数 i (i=1，2，…，n) 可以是数或对象名等，逗号","是参数之间的分隔符，句点"."是短句结束符。

对于任何一条事实，其中参数之间的次序是任意的。但是，一旦我们确定了某种次序，在整个程序中就必须遵守这一次序。例如，在程序 1-2 中，每一条描述父子关系的事实都是将父名放在前，子名放在后。

有了程序 1-2，就可以用目标来求解一些有关的问题。例如：若要问贾政是不是贾琏的父亲，则可以使用目标

? 父子(贾政，贾琏).

这一目标的执行将给出否定的回答。若要问贾政有哪些儿子，则可以使用目标

? 父子(贾政，X).

该目标的执行将首先求得贾政的一个儿子

X＝贾珠

若在其后键入";"，又将求得贾政的另一个儿子

X＝贾宝玉

在这之后同样还可以通过键入";"让目标继续执行从而求出贾政的其他儿子。如果不想求贾政的其他儿子，则可以键入"."。若要问谁是贾琏的父亲，则可以使用目标

? 父子(X，贾琏).

若要问贾代善有哪些孙子——贾代善的儿子的儿子有哪些，则可以