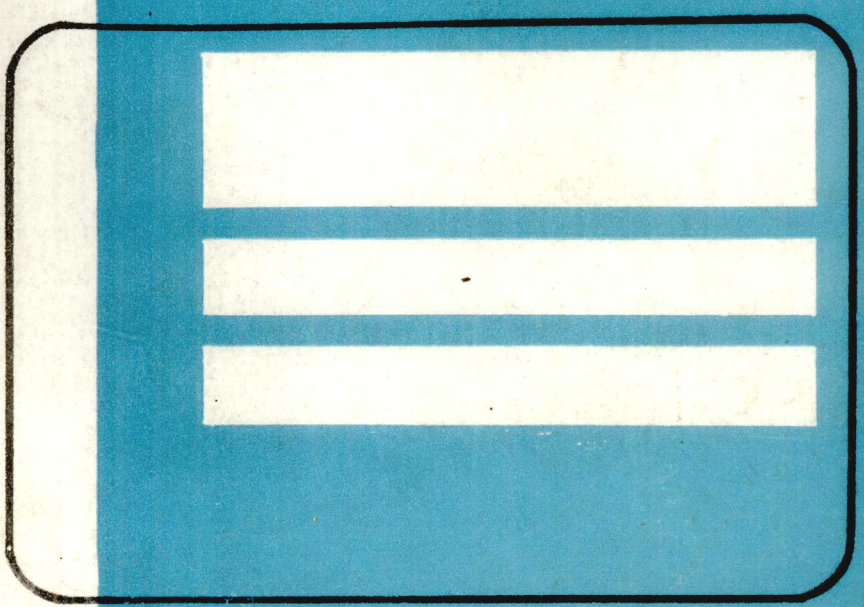
纪有奎　杨正林　编译

微型机 micro-PROLOG 语言及应用

（附参考手册和上机操作）



海　洋　出　版　社

前　言

在计算机程序设计语言领域里，出现了一颗明星般的语言——PROLOG（PROLOG表示PROgramming in LOGic；本书micro表示微型机），它以奇特、优美的程序风格，博得人们的青睐和赞叹！

它具有递归、回溯、模式匹配、数据库等很强的功能，逻辑推理是它的特长。应用范围广泛：适于数学论证，非数值处理，自然语言的理解，专家系统等人工智能领域；适于编写管理程序；用它编写智力游戏题的程序简洁而生动。Prolog语言易学、易用，程序简明易读（有人说在易学、易用，人机对话等方面，胜过BASIC语言）。学习者称赞该语言新颖、别具一格，容易理解（因为它面向自然、语言）。它不仅适于写小程序，更适于写大程序。有人统计过一个要用600多行 PASCAL 语言写的程序，用 LISP 语言写时要用100行，而用Prolog编程则行数更少。

七十年代已开始在欧洲传播；美国近年十分重视它；1982年日本研制第五代智能计算机把它选为核心语言。一些国家在大、中、小学开设此课，目前它正在世界范围迅速传播。

我国有关方面也十分重视此语言。《微型机应用》杂志发表文章题名“惹人注目的Prolog”。许多地区举办学习班，有的学校讲授此课，在研究、应用中传出喜讯。为适应各级学校、各阶层的需要而编译本书。

本书是以1984年英国专家K.L.Clark等人所著《“micro-PROLOG:Programming in logic”》一书为蓝本（这权威性原著，流传到许多国家，受到欢迎和好评）进行编译的，是教材型。编译时，考虑到我国推广学习本语言的需要补充了新的内容，并在书末另附上机参考资料。本书内容十分丰富，理论与大量实例相结合，书中习题在书末附答案，通俗易懂，是广大自学者指导性教材。全书包括六大部分，共十六章，由入门、概念、语法规则、上机操作、编程方法、趣味游戏程序、实用程序，逐步深入到专家系统，问题探讨等等，难易结合。

本书可用于CP/M80，MSDOS，UNIX等操作系统。机型为Z80，IBM PC，Sinclair 等以及它们的兼容机。海洋出版社发行的micro-PROLOG 系统软盘（1984年3.1版本），可在IBM PC机系列及其兼容机上运行本书的程序。

本书适用面广，特别适合不同文化程度的广大自学者（大、中、小学师生均可选用所需章节），可供科技人员，管理人员，计算机工作者，计算机房等人员阅读。尤其适于各级学校选为教材。

在编译本书时，得到梁永一、全知远、陈燕保等同志的协助；冀翻、唐锡南同志编写第十四、十五章，并审阅部分章节。在此一并表示感谢。由于我们水平有限，时间仓促，难免有误，敬请广大读者指正。

编译者

1985年8月

第零章 绪 论

0.1 入门知识介绍

虽然PROLOG①系统不同，但都由事实，规则，询问这三大要素组成，仅是内部谓词、询问格式以及语法规定等稍有差异。

以下分别介绍PROLOG的三大要素：

1. 事实

事实是由关系和个体构成的。关系和个体可由编程者自己定义，因此，用户可给关系和个体起不同的名字。

一个事实是一个条件，也是一个简单句；也就是一个语句，它必须有一个关系名，说明个体或个体之间的关系。如

Mary female 玛丽 女性 (A)

是一个事实，其中female（女性）是一个关系名，Mary（玛丽）是一个个体名，意思为玛丽是女性。由于此事实中只有一个个体名，即一个元素，所以又叫一元关系。在书写时，关系名和个体名一定要用空格间隔开。个体又叫自变量。

Bill likes Mary 比尔 喜欢 玛丽 (B)

这是一个关系名likes，两个个体名Bill（比尔）和Mary（玛丽）。此事实中有两个个体名，即两个元素，所以又叫二元关系。同样，在书写这一事实时关系名与元素间也要用空格隔开。

gives（Bill Mary book） 给（贝尔 玛丽 书） (C)

意思是贝尔给玛丽书。这是一个关系名gives里，拥有Bill，Mary和book（书）三个个体，又叫三元关系。

上述（A）—（C）式是三个独立的条件，是三个简单句，也就是三个事实。可见在每个简单句里有一个关系名，它说明：个体间的关系，如（C）式的gives（给），把三个孤立毫不相干的个体之间产生了联系（关系）；或者关系名说明个体的属性，如（A）式的关系名female说明Mary是女性。

关系名又叫谓词，（B）和（C）式的谓词是英语的动词。但该谓词不一定由英语的动词组成，它可以是形容词、名词以及复合词（短语）等等。

因谓词所在句中的位置不同，而分为前缀、中缀和后缀。（C）式的谓词gives在句中首位，称为前缀；（B）式的谓词likes在句子中间，称为中缀；（A）式的谓词female在句末，故称后缀。

①PROLOG这几个英文字母也可用小写。海洋出版社出版的本书姊妹篇《Prolog语言程序设计及其应用》中，除第一个字母P用大写外，其后的字母全用小写。本书从0.2节开始，一律写成PROLOG或micro-PROLOG，以区别该语言的不同版本。

2. 规则

一个规则是由几个简单句组成，这些简单句之间有依赖性，如果符合条件才能成立，所以这些简单句组成一个条件句，也是一个语句。如

Bill likes Mary if Mary female （D）

比尔 喜欢 玛丽 假如 玛丽 女性

这由（A），（B）式两个简单句组成一个条件句（D）式。意思为假若Mary（玛丽）是female（女性），Bill（比尔）就喜欢她。这两个简单句都得到满足，该条件句才能成立。

（D）式又可将if（假若）用and（与）或者用&代替，如

Bill likes Mary and Mary female

或

Bill likes Mary & Mary female

这两个条件句都与（4）式是等价的。

将上述事实和规则，用终端键盘打入，建立PROLOG数据库。数据库是由一系列事实以及规则组成，二者兼有或只有其中之一，这是PROLOG程序。也就是上述（1）—（4）式编程者自己定义，输入计算机后，便作为数据库，以便在询问时查询。

3. 询问

询问的目的是为了求解问题，找出问题的答案。

常用的基本询问方式有三种：which，is，one。其用法简介如下：

（1）which询问

格式为 &.which（x:x likes Mary）

机器启动后出现提示符&.，可由键盘打入提示符右端的内容，x表示变量。上述为

&. 哪个（x:x 喜欢 玛丽）

意思是询问谁（x）喜欢玛丽，即求解这个问题的答案，找出变量x究竟代表哪个个体（自变量）。

执行时，便在由事实、规则建立的数据库中扫描，按它们在数据库中排列顺序由上至下（即自顶向下）依次寻找，即先由（A）式开始寻找。询问中关系名为likes，而（A）式中关系名为female，二者关系名不同，无法找答案，于是再往下查询（B）式，（B）式的关系名与询问中的关系名一致（都是likes），这时再查对个体名，而likes右边都是Mary（一致），询问中likes左端为变量x，与（B）式中likes左端的Bill相对应，即求解的x便为Bill。注意，这叫模式匹配，简称匹配，即x与Bill相匹配，于是在终端显示屏上立即显示出

Bill

又继续由（B）式往下继续寻找还有别的答案吗？结果再没有找到，这时显示屏上出现

No（more）answers 没有（没有更多）答案。又如

&. which（x:x female if Bill likes x） 询问谁（x）是女性并且比尔喜欢她（x）。

在if左边由（A）式匹配得出x=Mary，随后验证if右边所问，由（B）式得到证实（x=Mary）。这在运行中推理显示出：Mary

（2）is询问

格式为&.i（Bill like Mary）是(比尔 喜欢 玛丽），意思为比尔喜欢玛丽吗？

于是又在数据库清单中由上至下依次寻找，在(B)式得到解答，因为关系名相同（都是likes），其左边和右边的个体名（自变量）又都吻合，因而立即在显示终端上显示出

YES 是

假若如下询问：

&.is（Mary likes bill） 是（玛丽 喜欢 比尔）

意思为玛丽喜欢贝尔吗？

在已建立的数据库清单中依次查寻，而（A）—（D）式中没有这条件，没有这句子，因为没有建立这个事实，所以终端显示出

NO 没有

（3）one 询问

&.one（x:Bill likes x） 一个（x:比尔 喜欢x）

意思为找出比尔所喜欢的x(人或物)。

在数据库的清单中依次扫描寻找，因询问中的关系名和（B）式中的关系名同名，关系名左边的个体名又相同（都是Bill），所以关系名右边的自变量Mary和变量x得到匹配，即x为Mary。便在显示屏上可看到

Mary

找到一个答案便自动停止，但立即又显示

more（y/n）

意思为还要再找一个吗？让你选择y（yes）或n（no）键。若不需要，便按n键，则停。若按y键，便在数据库中由（B）式的下一个式子继续找其他别的答案（如果数据库中还有其他匹配的话）。

当询问到规则右端失败时需其左端再变换，或即使得到某个解，还再试图找其他解答，这时要再返回数据库依次再往下查找可能的匹配答案，这过程叫回溯。为防止没完没了的回溯，采用符号“/”来截断回溯。

以上多次提到的x是变量名之一，micro-PROLOG变量名用x，y，z，X，Y，Z和由这些字母打头后面跟着若干个数字的符号表示，如y2，Y5等。作为自变量的人名第一个字母可用大写，自定义的关系名用小写。若系统定义的关系名其所有的英文字母全用大写，这叫原语，不容许用户改变。

以上所述是初步介绍PROLOG概括情况，以便使读者先初步地建立总体概念。在以后的章节里还要重复上述概念，并进一步深入细致地描述和扩展。

由以上介绍可以看出PROLOG语言的主要特点是具有逻辑推理功能。例如：若自定义事实甲是乙的父亲，又是丙的父亲，若乙、丙是女性，则可推理出乙、丙是亲姊妹。

如图书管理时，将存书的事实置入数据库，借书时可询问x书有否。若再加些规则定

义（描述）有关书之间的关系，不仅可寻找出某本书，而且还可被告知（推理）该书与有关书的内存联系。人事档案之类的管理也如此。

又如根据有经验的医生把一类疾病的症状及治疗原则写成事实和一些规则，建立数据库。当提出询问时，可推理得出该患者的疾病诊断及治疗方案等。

再如数学定理证明时，把公理和定义写成事实和规则，建立数据库，提出有关定理证明询问时，便在数据库中寻找匹配答案，由推理得出结论。

如果提供些英语单词，可组成英语句子；或给出英语句子，程序分析出名词短语、动词短语等语法结构。这些也是由事实、规则中分析推论出来的。这是对自然语言的理解。

这类问题在自然科学和日常生活中有许多的例子。读者可大胆联想和尝试。贯穿全书的例子，仅是一小部分。

可见PROLOG语言与惯用的常规语言(如BASIC, FORTRAN等)不同。前者最关心的是输入一些事实来告诉计算机什么是真（自定义为真），而后输出（推理得出）新的事实。关心输入和输出，而在什么时候、什么步骤做什么，似乎不太关心；而后者（指常规语言）还关心一条条语句（即一个个命令）的执行过程，是命令似的，有人称其为命令式语言。

PROLOG是将一些事物进行描述，描述事实和规则，如本节开始时的（A）—（D）式，因而又称描述性语言。提出询问时启动程序运行，经过谓词演算、匹配等推理出新的事实，可见它是在一阶谓词逻辑基础上开发的语言。推理的执行过程并非真的不关心，书中还介绍了利用跟踪管理程序来跟踪执行过程中的每一步。而常规语言也有描述部分……。为了突出语言的特点，强调了有关方面。

PROLOG作为一种超高级语言，具有很强的描述和解题功能。但令人吃惊的是——PROLOG的基本语法却比任何一种常规高级语言都要简单，这应归功于理论基础的一阶谓词表示法的简洁和严谨。PROLOG的推广和传播迅速，原因之一就在于由语法简单而导致的实现简单。

看完本节的入门知识介绍，再返回去理解本书的前言论述，或继续往下阅读全书，是有益的。

0.2 各章说明

这本书（除第零章外）包括六部分，共十五章。下面我们简要介绍各章的内容。

第一部分——基本概念，其中：

第一章介绍使用micro-PROLOG去建立和询问事实数据库。用micro-PROLOG能构成和询问数据库的简易性是这种语言最突出的特点之一。本章也介绍了micro-PROLOG内部的算术特性。这与常规的程序设计语言完全不同。我们通过询问加法关系的（隐含）数据库进行加和减。同样，我们通过乘法关系数据库进行乘和除。

第二章描述怎样用规则扩充数据库，用规则简化询问，也能用这些规则给出关系的递归定义。

第三章介绍表及其描述，怎样用这些表构成指令。经常把许多语句压缩成指令。使用

特殊的表模式访问表的元素。这个表处理模式是micro-PROLOG的主要特点。这章末尾介绍"isall"条件，可用它建立询问表的答案的集合，它是micro-PROLOG作为数据库语言和表处理语言的接口。

第二部分——使用micro-PROLOG进行逻辑程序设计，其中：

第四章描述了能用于询问和规则的条件的新格式，包括"not"，"forall…then…"和"either…or…"。这些条件的应用有效地加强了数据库应用和"可执行"说明的开发的micro-PROLOG的能力。

这章还描述了关系is-told，用它当回答我们的询问时，使micro-PROLOG询问我们。应用这个关系可以简化自上而下的程序开发和写单询问用户专家系统。

第五章描述了更复杂的表处理任务的若干append程序。特别是考查了表连接的"append"程序。用它还可连接两个表，也可以分裂表，甚至确定一个表的成员关系。这章末尾有三个表排序程序，其中之一是排序关系的说明。

第六章是micro-PROLOG用于语言分析的一个引言。它包括词表变换成反映语法结构的表。语法分析和自然语言理解是逻辑程序设计的主要应用之一，而且这些应用对于micro-PROLOG是极其地适合。

第七章涉及了与micro-PROLOG程序设计实用有关的一些问题，描述了有关能缩减所用空间，或询问求解时所花时间的语言的各种特点。

第八章介绍了micro-PROLOG的命令，这些命令存在内部关系，当求解这些关系时有一种副作用，这种关系的一个例子是从终端读数据。它的逻辑解释为：可以在终端读数据；它的控制解释为：读打入的下一个数据。micro-PROLOG的这些命令有损于它的描述性本质，因为用这些命令的程序不是纯描述性程序。但是，以后我们会明白，对于一个或二个辅助关系的定义，可能常常限制使用这些命令，程序的其余部分完全是描述性程序。

更确切地说，上述命令如语言原语的可用性，使编程人员能够通过开发自己的程序来开发系统，改编他自己需要的系统。这可以通过第四章里介绍的is-told关系的一种简化形式来说明。

第三部分——核心micro-PROLOG，其中：

在第九章，我们描述了micro-PROLOG程序的标准语法。这是micro-PROLOG解释程序存取和求解的那些事实和规则的形式。这种形式也是在后备存储器里保存程序的形式。第一至第八章里所用的语法是用户通过第一和第二部分里所开发的SIMPLE程序的开发系统翻译成标准语法。SIMPLE本身就是以标准语法写的一种micro-PROLOG程序。

全部的micro-PROLOG程序恰是一种特殊形式的表。因此，编写micro-PROLOG表处理程序要比编写micro-PROLOG其他的程序容易。在第九章里我们已指出如何通过SIMPLE程序编写表程序，并且介绍了micro-PROLOG的一、二个特点，以便以标准语法写程序时应用。我们也给出了询问的各种形式的micro-PROLOG定义。这些定义曾在第一和第二部分里运用过。

第四部分——micro-PROLOG的应用，分四章，在此举例说明micro-PROLOG的一些应用，其中：

第十章里介绍的Frank Kriwaczek在工程管理中使用的关键路径分析法是搜索图的日常应用——复杂工程的活动形成一个无圈图，并用关键路径分析程序搜索各种路径图。

第十一章里Peter Hommond介绍了当前专家系统所必须的一些方法。一个专家系统是一个专家在他的专门领域里用以解决问题的“经验法则”的集合。由一定形式的专家经验法则构成一个专家系统，并用micro-PROLOG去解决问题。

第十二章Maarten Van Emdem和Keith Clack提出用于博弈的游戏程序的一些典型算法。NIM游戏是用计算机下棋的许多程序方法中的一个简单的例子。

第十三章是用micro-PROLOG对一些著名的趣味智力题编程。例如，农夫、山羊、狼与卷心菜渡河问题；几个水容器之间互相灌水、倒水问题；八皇后布局方法等。从而介绍了深度优先策略、广度优先策略以及其他编程技巧和常用方法。

第五部分——应用再举例和问题探讨，其中：

第十四章是micro-PROLOG在逻辑电路自动诊断中的应用。利用本书介绍的逻辑语言编写程序——诊断电路故障的模拟、检测程序，既直观、简练，又易读，并且行之有效。虽以一个较简单的全加器为例进行模拟、检测，但思路新颖、别致，并且还可以把它扩大到对复杂电路的描述。

第十五章，总结性地探讨几个问题。如micro-PROLOG整体结构，时空观，与常规语言的等价功能，还附有最短路径程序。

第六部分——两个附录。附录一给出了全书丰富的练习题的答案。附录二中扼要地列出了SIMPLE，核心micro-PROLOG，DEC-10 PROLOG系统等上机操作方法。

第一部分 基本概念

第一章 事实和询问

1.1 建立事实数据库

本章通过micro-PROLOG数据库建立和询问的例子，介绍了逻辑程序设计的一些基本概念。

系统注释——在计算机上使用micro-PROLOG 如果你能使用备有micro-PROLOG的计算机，建议你用这台计算机做完这些例子和练习。为此必须把 SIMPLE 前端系统与micro-PROLOG一起存入计算机。SIMPLE是分布在磁带或磁盘上，用micro-PROLOG提供的一种程序。关于启动 micro-PROLOG和LOAD SIMPLE的具体方法可查阅本书末 micro-PROLOG上机参考资料，并实际上机操作。

1. 建立事实

假定，我们想要建立一个描述某些人家庭关系的数据库，我们先构成这些关系的描述，再建立数据库，一次把它们中的一种关系的描述送入数据库里。

用这些描述表示的逻辑句子有简单句和条件句两种。我们先从表示事实的简单句着手。

在任何家庭里都有关于个人之间关系的大量事实。假定有两个这样的事实：

Henry Snr is the father of Henry （1）

亨利森是亨利的父亲

Henry is the father of Mary （2）

亨利是玛丽的父亲

每个描述家庭成员关系的例子都有许多这样的事实。现在这些英语句子差不多是micro-PROLOG句子。

句子（1）和（2）里的关系名是 "is the father of"，而在micro-PROLOG里，我们必须使它变成用连字符连接的一种复合词。所以，我们必须用 "is-the-father-of" 或简化为 "father-of"。类似地，我们必须用一单词命名个体，再用连字符把它们连起来，写成 "Henry-Snr" 代替 "Henry Snr"。以这种方法重写句子（1）和（2），使其变成为 micro-PROLOG句子

Henry-Snr father-of Henry

亨利-森是亨利的父亲

Henry father-of Mary

亨利是玛丽的父亲

在一个micro-PROLOG数据库里，这两个句子是两个事实（1）和（2）的直接表示法。我们用一个专用的 add 命令把它们写入数据库。

```
&.add (Henry-Snr father-of Henry)
&.为屏幕显示提示符，表示由键盘可打入（亨利森是亨利的父亲）
&.add (Henry father-of Mary)
```

注意，被送入的句子要用括号括起来，这个括号是必不可少的，它告诉micro-PROLOG被加在句子开始和末尾的词的顺序。对于micro-PROLOG的一个句子是一定形式的被括号括起来的词表。

系统注释——错误和提示 add命令前面的“&”不是被用户打入的，而是由micro-PROLOG打印输出的提示符，其作用是告诉我们它准备接受一个命令，而且必须通过按键盘上的RETURN或ENTER键来终止每个添加命令add。在你按这个键之前，可用RUBOUT，BACKSPACE或删除用的DELETE键来纠正打印错误，并返回到这个错误之前。至于你选用哪个键，则随计算机而定。

换句话说，对你正在用的这个版本的micro-PROLOG可以用行编辑程序：用它修正错误不需要重新打印在错误之后的每个事件。在micro-PROLOG资料里所说的行编辑程序是你的计算机专用的（对不同的机器行编辑程序是不一样的）。

当你确信打入的命令无需再改动时，按RETURN或ENTER键，那么micro-PROLOG将服从这个命令。如果在句子的语法里有一个错误，例如，如果你在“father-of”里忘了放连字符（注：即“-”），将给出一错误信息，告诉你这个句子是一个不正确的简单句形式，或这个简单句形式是无效的。如果你把“add”错写为“ADD”，你将得到这样的错误信息：

```
NO definition for relation
trying ADD (.....)
```

这是因为这个关系/命令名不是micro-PROLOG已定义的命令，也不是我们正在使用的SIMPLE前端系统已定义的命令。如果你得到另一个错误信息，这个句子没有被接受，请再改用add命令（如果你正确地拼写add，并且得到这个形式的一个错误信息：

```
Error:2
```

这意味着你忘记了LOAD SIMPLE）。

你无需在一行上打入括起来的句子的全部内容。的确，有的句子也许比你使用的计算机的显示行要长，正当使用到显示行的末端时，需要检查你在那行上打入的是否正确，并且如果需要则可以对它进行编辑。

当你确信打入的信息没有错误时，按RETURN或ENTER键，你会得到提示“1.”，它代替通常的命令提示“&”。这表示micro-PROLOG知道现行命令是不完整的。实际上，这个“1”表示micro-PROLOG一直在等待被加在句子末尾的单右括号。“.”是当从键盘准备就绪要读时，micro-PROLOG一直显示的读提示符（1.相当换行符号）。

如果句子里用了括号，就应当注意句子里括号表的使用，这个提示也许是“2.”或“3.”或一些较高的数。这个数总是正确地结束句子所必须的右括号的数。当我们开始使

用表时，你会发现右括号提示符非常有用。

（1） 不同种类的关系

一种关系如“father-of”（父亲-属于）适用一对个体之间，在这种情况下是一个“father”和一个“child”之间的关系。这是一种二元关系。并不是所有的关系都是二元的，有一些涉及三个或更多个个体，有些适用于单个个体的性质，性别“male”（男性）和“female”（女性）就是这样的性质（它们是一元关系）。对某人把东西送给别人间的这种关系是三元关系。给出这些非二元关系的事实的句子的语法稍有不同。

有关性质描述的句子用后缀形式写出，如：

Henry male　　亨利　男性

在这种句子里属性名紧跟在个体名后面。如果用前缀形式写出，例如

gives（Henry Mary book）　　亨利给玛丽书

SUM（2 3 5）　　2加3的和是5

在这种句子里，关系名先于通过关系相关联的个体括号表。

对于二元关系和属性的句子可以使用前缀形式表示，如

father-of（Henry-Snr Henry）

是被承认的，它等价于

Henry-Snr father-of Henry

但是，看起来中缀和后缀形式更清楚，即使你以前缀形式的micro-PROLOG写二元关系或一元属性的句子，当你列表或编辑程序时，也会以二元中缀和一元后缀形式显示它们。

（2） 一个技术术语——关系的自变量

事实告诉我们，通过某些关系相关联的一些个体，在数学和逻辑里称这些个体为关系的自变量。也称为关系的第一自变量，第二自变量等。这是按照自变量在前缀形式句子的自变量表的位置而得名。在句子

gives（Henry Mary book）

中，“Henry”是第一自变量，“Mary”是第二自变量。而“book”是第三自变量。

（3） 系统注释——空格的用途

用空格把个体名和关系名分开是必要的。以micro-PROLOG空格和由于按RETURN或ENTER键所产生的换行是字分隔符。只有micro-PROLOG知道由按RETORN或ENTER键引起换行。由于micro-PROLOG忽视了超过上行的结束，你的打入将引起自动换行，这不作为分隔符。

虽然使用分隔符的数量无关紧要，但漏用一个分隔符，micro-PROLOG会把需要构成的两个名字错误地构成为一个名字。

你不能总需要用一个分隔符。micro-PROLOG有时通过字符类型的改变检测一个字的结束和下一个字的开始。例如，“(”或“)”总是在它前面的字结束时发的信号，所以不需要在括号前或后边用一个空格。

关于micro-PROLOG了解或不了解一个字的边界的更详细的资料，读者可参阅参考手册。

需要用空格作为分隔符的转换，如对上面的 "father of" 短语按顺序用连接符连接起来，使其成为一个关系名，而不是两个关系名。即father-of。

2. 若干事实的键入

我们键入家庭关系若干事实：

```
&.add(Katherine mother-of Mary)          &.add(凯瑟琳是玛丽的母亲).
&.add(Henry father-of Elizabeth 2)       &.add(亨利是伊丽莎白2的父亲).
&.add(Ann mother-of Elizabeth 2)         &.add(安是伊丽莎白2的母亲).
&.add(Henry father-of Edward)            &.add(亨利是爱德华的父亲).
&.add(Jane mother-of Edward)             &.add(珍妮是爱德华的母亲).
&.add(Henry-Snr male)                    &.add(亨利-森是男性).
&.add(Henry male)                        &.add(亨利是男性).
&.add(Elizabeth 1 female)                &.add(伊丽莎白1是女性).
&.add(Katherine female)                  &.add(凯瑟琳是女性).
&.add(Mary female)                       &.add(玛丽是女性).
&.add(Elizabeth2 female)                 &.add(伊丽莎白2是女性).
&.add(Ann female)                        &.add(安是女性).
&.add(Female(Jane))                      &.add(女性珍妮).
&.add(Male(Edward))                      &.add(男性爱德华).
```

注意，我们列出了一些 "mather-of" 和一些性别是男和女的事实。我们用一条键入命令在任何时间可以键入任何关系的句子。把关系名和个体名分别收集在一起。一个程序的词汇表由关系名和个体名组成；词汇表定义了有关询问可涉及的内容。至此，我们的词汇表为

Henry-Snr	亨利-森	个体名
Henry	亨利	
Mary	玛丽	
Elizabeth 1	伊丽莎白1	
Katherine	凯瑟琳	
Elizabeth 2	伊丽莎白2	
Ann	安	
Edward	爱德华	
Jane	珍妮	
father-of	父亲-属于	关系名
mother-of	母亲-属于	
Male	男(性)	
Female	女(性)	

注意，我们已用数字表示名字 "Elizabeth 1"和 "Elizabeth 2"来区分这两个 Elizabeths。键盘的数字和 "-" 及全部字母符号都可用于表示名字。但人名不能以数字开始，因此

4jane

是不允许的，而

jane4

是允许的。实际上，micro-PROLOG会把4jane解释为4 jane，即象数“4”后面是分离名“jane”。这是micro-PROLOG解释符号类型变化状态的一个例子，它同一个分隔符空格的插入是等价的。

由字母、数字和“-”组成的名字只是名字的一种类型，把它们叫做字母数字常量。其他种常量如符号常量和引号常量也可用于名字，请读者翻阅micro-PROLOG参考手册。我们基本上用字母数字常量作为名字。

3. 接受（accept）命令

在上述家庭关系数据库中，我们所加的关于男性和女性的最后两个事实是以前缀形式表示的。有一个专用命令可加速以前缀形式表示的一组事实的输入。这个命令是接受命令accept。输入

accept female

会得到提示

female.

现在输入你想要输入的female事实参量表。虽然每次只打入一个参量表，并一次次得到这个关系名的提示。你可以继续以这种方式一次次打入参量表，直到输入完毕。当你再接到关系名提示时，则应输入end，表示该关系名的参量表全部输入完。

使用accept，下列关系能用来输入迄今我们增加的全部male和female事实。

&.accept male	&.accept 男性
male.(Henry-Snr)	男性.(亨利-森)
male (Henry)	男性.(亨利)
male.(Edward)	男性.(爱德华)
male.end	男性.结束
&.accept female	&.accept 女性
female.(Elizabeth1)	女性.(伊丽莎白1)
female.(Katherine)	女性.(凯瑟琳)
female.(Mary)	女性.(玛丽)
female.(Elizabeth2)	女性.(伊丽沙白2)
female.(Ann)	女性.(安)
female (Jane)	女性.(珍妮)
female.end	女性.结束

这里强调的是我们所输入的事实，提示由micro-PROLOG供给。

4. 列表和保存入一个程序

用命令List可以显示我们的数据库程序，也可以用这个命令在屏幕上显示输入的全部句子，或仅显示特指定的关系名的句子。

要列出整个程序，需打入

&.list all	&.list all
Henry-Snr father-of Henry	亨利-森是亨利的父亲
Henry father-of Mary	亨利是玛丽的父亲
Henry father-of Elizabeth2	亨利是伊丽莎白2的父亲
Henry father-of Edward	亨利是爱德华的父亲
Elizabeth1 mother-of Henry	伊丽莎白1是亨利的母亲
Katherine mother-of Mary	凯瑟琳是玛丽的母亲
Ann mother-of Elizabeth2	安是伊丽莎白2的母亲
Jane mother-of Edward	珍妮是爱德华的母亲
Henry-Snr male	亨利-森 男性
Henry male	亨利 男性
Edward male	爱德华 男性
Elizabeth1 female	伊丽莎白1 女性
Katherine female	凯瑟琳 女性
Mary female	玛丽 女性
Elizabeth2 female	伊丽莎白2 女性
Ann female	安 女性
Jane female	珍妮 女性
&.	&.

这些句子的组合不是按它们的输入顺序，而是与它们的关系名相一致。但是，对于每个关系的句子，列表与他们输入的顺序对应。

我们可以选择一个特指定的关系名进行列表，例如指定mother-of

```
&.list mother-of
Elizabeth1 mother-of Henry
Katherine mother-of Mary
Ann mother-of Elizabeth2
Jane mother-of Edward
&
```

我们在磁盘（或磁带上，根据你用的计算机的情况）保存数据库的现行状态，给出一个我们选择的文件名。举例如下：

```
&.save FAMILY
```

现行程序里的全部句子的副本变成文件存于后备存储器（外存）。现行的句子仍然保存在数据库里。但是，我们也可以通过打入

```
&.load FAMILY
```

取出这些句子，并把它们增加到数据库里。

（1）通过增加和删除句子进行编辑

通过删除一个个句子并以增加一个新的句子代替它的办法可以完成对micro-PROLOG程序的编辑。假定Elizabeth2的母亲的名字拼错，而且它应该是“Anne”，消去句

子“Ann mother-of Elizabeth2”最简单的方法是用

```
& .delete ( Ann mother-of Elizabeth2 )
```

delete的应用与add相反。在程序中，如果括号中的句子作为自变量放在命令后，则该delete命令可以消去它。如果在程序里没有该句子，你将得到一个信息，告诉你没有这样的句子。要被删除的句子和现行数据库的一些句子间如果没有完全匹配，也会通知你。

有另一种删除一个句子的方法，即通过待删除的句子在关系句子列表里的位置来删除。在已知的关系“mother-of”的列表里，上述的句子“Ann mother-of Elizabeth2”是第三个句子。于是，我们可使用一种删除命令的择一形式代替删除的已知句子，并按照它的关系名和位置识别这个句子。

```
& .delete mother-of 3
```

使用delete命令的另一种形式删除这个句子，可以增加新的型式：

```
& .add ( Ann mother-of Elizabeth2 )
```

现在如果我们显示“mother-of”关系，会得到

```
& .list mother-of
Elizabeth1 mother-of Henry
Katherine mother-of Mary
Jane mother-of Edward
Anne mother-of Elizabeth2
& .
```

这个新句子

```
Anne mother-of Elizabeth2
```

被列在末端，因为它是最后输入的。

现在我们校正在“female”关系里“Ann”的拼法。此时，将用“Anne female”代替句子“Ann female”。我们是通过删除旧句子和增加新句子来完成的，因此，新句子占有“female”句子列表里被删除的旧句子的同一位置。下面是必要的命令以及micro-PROLOG应答。

```
& .list female
Elizabeth1 female
Katherine female
Mary female
Elizabeth2 female
Ann female
Jane female
& .delete female 5
& .add 5(Anne female)
& .list female
Elizabeth1 female
```

Katherine female
Mary female
Elizabeth2 female
Anne female
Jane female
&

这里我们用了add命令的另一种形式——添加的句子位置是给定的。

add 5 (Anne female)

使增加的句子为关系新表里的第五句。在现行第四和第五句之间插入上述句子，就是在那位置删除了该句子。

(2) 使用行编辑程序进行编辑

改变一种关系中的一句，比较快的方法，尤其是当文本改动不大时，免不了对你正用的计算机使用行编辑程序。你调用行编辑程序使用edit命令，类似删除的第二种形式，通过它们的关系名以及它们在关系句子列表里的位置识别这个句子。

edit female 5 (编辑女性列表里第五句)

将导致显示出

5 (Anne female)

准备好使用行编辑程序进行编辑。

(3) **系统注释**——在你的计算机上的行编辑程序

对于使用编辑程序的具体方法和结束时怎样退出，可参阅你正在使用的计算机的micro-PROLOG专用资料。

注意，应当把句子的位置与句子本身一起给出。如果要更改位置，比如使它从5变到4，则可通过编辑位置重新安排句子。正如增加一个句子时一样，当退出行编辑时，micro-PROLOG需要括号划定出句子内容。

5. 程序编辑命令归纳如下

下面所有命令对通过micro-PROLOG保存在用户工作区的现行程序有效。在给出的每个命令的一般形式里，我们应使用单尖括号〈 〉，以表示某种语法形式。例如，用Sentence表示可使用的任何句子。

(1) add

1) add (〈Sentence〉)

将增加它的括号句子自变量到对于相应关系的句子现行列表的末尾。

2) add n (〈sentence〉)

在它的关系句子列表里将增加括号句子作为一个新的第n个句子。现在如果比n个句子少，则它便成为新的最后的句子。换句话说，它被插入在现行第n－1个句子和现行第n个句子之间。

(2) delete

1) delete (〈Sentnce〉)

将从现行程序中删去〈Sentence〉。

2) delete 〈relation name〉 n

在对于命名关系的句子的现行表里删去第n个句子。

(3) list

list 〈relation name〉

在现行程序里列命令关系的所有句子。

2) list all

列工作区程序里的所有句子。

(4) save

save 〈file name〉

在外存储器里，将保存你的程序的现行状态的所有句子。

系统注释——micro-PROLOG 文件 给定文件名必须同程序的任何关系名以及任何命令名相区别。否则，将得到“File error”信息，而且save操作将不起作用，请再使用一个不同的名字。我们建议在文件名里全部用大写字母，以避免同关系名冲突。

在外存储器保存的程序文件名与你在命令中给出的形式稍有不同。然而，在micro-PROLOG 中你仍可通过选定的文件名来访问该文件。例如，对于在CP/M 下的micro-PROLOG，目录中的文件名以“LOG”结尾。此外，如果你检查或列表保存的程序文件不在micro-PROLOG 系统中，则会发现程序的句子不是以输入它们那种形式保存的，而是以micro-PROLOG标准语法的一种特殊的编译形式保存的。

(5) kill

1) kill 〈relation name〉

删除指定关系的全部句子。

2) kill all

删除内存工作区程序的全部句子，在你保存程序以后，为了一个新的程序才可用这个命令清除内存工作区。

(6) edit

edit 〈relation name〉 n

使用micro-PROLOG 的行编辑程序能够编辑已命名关系的第 n 个句子 显示括号里的句子及它的位置，并准备编辑。通过改变位置号，可以使它的关系的句子在列表时发生变化。如果想改变句子所在的位置，而不改变句子的内容，可以只修改句子的位置号，也可以改变被编辑的句子的关系名。位置号是为更改关系名加到关系列表里的被编辑句子的位置。

(7) QT

QT. 这个命令是从micro-PROLOG 中退回到主机操作系统的。正如“kill all”一样，在使用它之前，你应该保存内存工作区程序。在QT之后的“·”是重要的。由于所有的micro-PROLOG命令至少有一个变量，所以这个“·”是必须的。在这种情