
Prolog 调研报告

人员
吴金泽

摘 要

Prolog 语言是一种说明性语言，它是英文 Programming in Logic 的缩写，其理论基础是一阶谓词逻辑。它是人们把逻辑作为程序设计的一种语言的努力结果。她被广泛的应用在人工智能的研究中，它可以用来建造专家系统、自然语言理解、智能知识库等。同时它对一些通常的应用程序的编写也很有帮助。使用它能够比其他的语言更快速的开发程序，因为它的编写方法更像是使用逻辑的语言来描述程序，在运用 Prolog 进行程序设计时，重点在于对那些与问题有关的对象间的逻辑描述。在这种逻辑描述的基础上，Prolog 运用自身具有的问题求解机制寻求答案。因此能够使 2 自动化求解问题和定理证明具备可行性。

本报告中主要对 Prolog 的产生发展，常用版本，理论基础，贡献，语言特点，基本数据语法结构，应用，以及相关拓展如 Datalog 等做了简单的梳理和介绍。

目 录

摘要	
第 1 章 简介	2
1.1 关于 Prolog	2
1.2 发展历史	2
1.3 常用版本介绍	3
1.3.1 Turbo Prolog	3
1.3.2 PDC Prolog	3
1.3.3 Visual Prolog	4
1.4 理论基础	5
1.5 贡献	5
第 2 章 Prolog 语言	6
2.1 特点	6
2.2 基本结构	7
2.2.1 事实	7
2.2.2 规则	7
2.2.3 目标	7
2.3 数据结构	8
2.3.1 项	8
2.3.2 结构和树	8
2.3.3 表	8
2.4 基本原理	9
2.4.1 SLD 归结	9
2.4.2 回溯	9
2.4.3 控制	10
第 3 章 相关应用	11
3.1 数学应用	11
3.2 专家系统	12
3.3 自然语言理解	15

3.4 智能知识库	15
第 4 章 Datalog.....	19
4.1 简介	19
4.2 基本结构	20
4.3 规则	20
4.4 基本关系运算	20
4.4.1 并	21
4.4.2 差	21
4.4.3 积	21
4.4.4 选择	21
4.4.5 投影	21
4.4.6 其他	22
4.5 多重查询	22
4.6 递归查询	22
参考文献	23

第 1 章 简介

1.1 关于 Prolog

Prolog 是面向逻辑、面向问题，描述逻辑关系和抽象概念，处理对象是知识（确切地说是符号）的一种逻辑型人工智能设计语言，Prolog 是陈述性语言而不是过程性语言，在 Prolog 程序中不需要告诉计算机“怎么做”，只需要告诉计算机“做什么”，即只要给出所需的事实和规则，Prolog 使用演绎推理的方法就可以自动的对问题进行求解。Prolog 是一个典型的符号逻辑形式系统，并且是以一阶谓词逻辑为基础设计的，其目标就是处理逻辑推理。它是以一阶谓词逻辑的 Horn 子句集为语言，以归结原理为工具，加上深度优先搜索的控制策略而形成的逻辑程序。由于其程序子句的形式和一阶谓词逻辑演算表示事物的方式十分相似。因而很适合表示人的思维和推理规则。此外，Prolog 本身就是一个演绎推理机，具有匹配（即合一）、回溯、递归等功能，并且具有正向推理和反向推理策略，从而可以在计算机上完成自动推理。

1.2 发展历史

Prolog 语言最早由 Aix-Marseille 大学的 Alain Colmerauer 与 Phillipe Roussel 等人于 60 年代末研究开发。1972 年被公认为是 Prolog 语言正式诞生的年份，当年在法国马赛，AColmerauer 和 P.PRoussed 按照 Kowalaki 的工作基础上（霍恩子句的过程化表示）设计了 Prolog 语言。

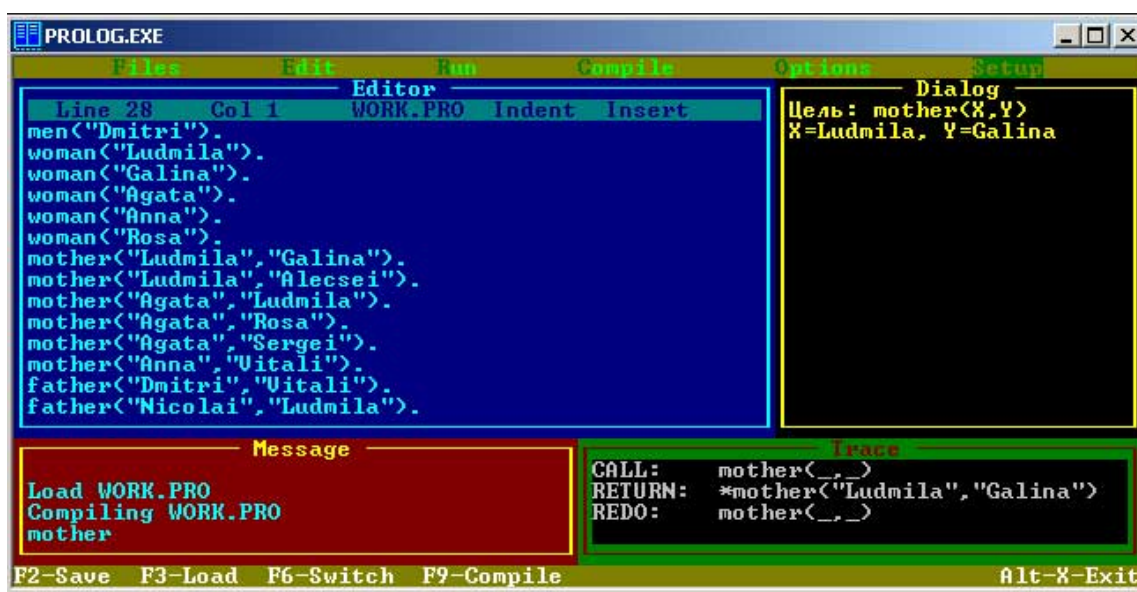
当时的欧洲，Prolog 毫无疑问是最流行的语言，而在北美大陆，则有另一种函数式语言几乎垄断了整个人工智能领域的编程—Lisp，而这个语言的创立人则是鼎鼎大名人工智能之父约翰·麦卡锡。自 1972 年 Prolog 发布以后，分支出多种 Prolog 的方言。最主要的两种方言为 Edinburgh 和 Aix-Marseille。最早的 Prolog 解释器由 Roussel 建造，而第一个 Prolog 编译器则是 David Warren 于 1977 编写的，即 DEC-10 PROLOG 系统，使之达到了实用化的程度。而在此之后，Prolog 被全世界大范围的使用，日本政府曾经为了建造智能计算机而用 Prolog 来开发 ICOT 第五代计算机系统。在早期的机器智能研究领域，Prolog 曾经是主要的开发工具。

80 年代 Borland 开发的 Turbo Prolog1.0 系统，进一步普及了 Prolog 的使用。后来又经历了 PDC PROLOG, Visual Prolog 不同版本的发展。并且的逻辑语言也于 80 年代初开始研制，其中比较有名的有 PARLOG, Concurrent PROLOG 等。同时我国也开发了几种版本的 prolog 系统。1995 年确定了 ISO Prolog 标准。

目前比较流行的实现工具包括 SWI-Prolog, Yap 等

1.3 常用版本介绍

1.3.1 Turbo Prolog



由美国 Prolog 开发中心（Prolog Development Center, PDC）于 1986 年开发成功、Borland 公司对外发行，其 1.0, 1.0, 2.1 版本取名为 Turbo Prolog，主要在 IBM PC 系列计算机，MS-DOS 环境下运行。

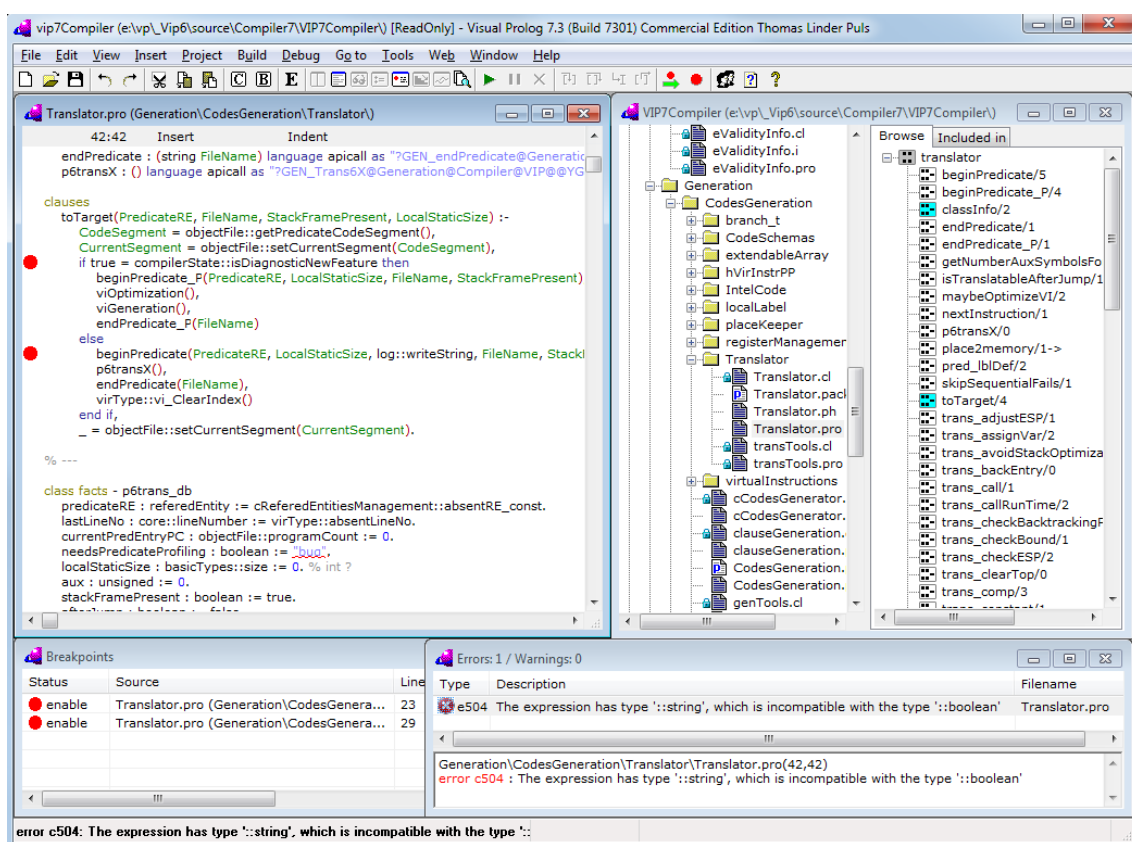
1.3.2 PDC Prolog

1990 年后，PDC 推出新的版本，更名为 PDC Prolog3.0, .2，它把运行环境退找到 OS/2 操作系统，并且向全世界发行。它的主要特点是：

- 速度快。编译及运行速度都很快，产生的代码非常紧凑。
- 用户界面友好。提供了图形化的集成开发环境。
- 提供了强有力的外部数据库系统。

- 提供了一个用 PDC Prolog 编写的 Prolog 解释起源代码。用户可以用它研究 Prolog 的内部机制，并且创建自己的专用编程语言、推理机、专家系统外壳或者程序接口。
- 提供了与其他语言（如 C、Pascal、Fortran 等）的接口。Prolog 和其他语言可以相互调用对方的子程序。
- 具有强大的图形功能。支持 Turbo C、Turbo Pascal 同样的功能。

1.3.3 Visual Prolog



Visual Prolog 是基于 Prolog 语言的可视化集成开发环境，是 PDC 推出的基于 Windows 环境的智能化编程工具。目前 Visual Prolog 在美国、西欧、日本、加拿大、澳大利亚等国家和地区十分流行，是国家上研究和开发智能化应用的主流工具之一。

Visual Prolog 具有模式匹配、递归、回溯、对象机制、事实数据库和谓词库等强大功能。它包含构建大型应用程序所需要的一切特性; 图形开发环境、编译器、链接器和调试器、支持模块化和面向对象的程序设计、支持系统级编程、文件操作、字符串处理、位运算、算数和逻辑运算、以及与其他编程语言的接口。

Visual Prolog 包含一个全部使用 Visual Prolog 语言写成的有效的开发环境，包含对话框，菜单，工具栏等编辑功能。

Visual Prolog 与 SQL 数据库熊，C++ 开发系统，以及 Visual Basic、Delphi 或 Visual Age 等编程语言一样，也可用来轻松地开发各种应用。

1.4 理论基础

本节将简单的介绍一下 Horn 逻辑。Horn 逻辑是 Prolog 的逻辑理论基础。Horn 逻辑是对事物以及其相互关系进行推理的形式系统，它是由 Horn 子句组成的一阶谓词逻辑的子部分。

Horn 子句有两种，一种是有一个非否定的文字，称为有头的；另一种没有非否定的量，称为无头的。Horn 子句的定义如下：

- 一个 Horn 子句时至多包含一个正文字的子句。
- 一个程序子句是只包含一个正文字的子句。一般形式为： $A:-B_1, B_2, \dots, B_n$ 。
- 若一个程序子句包含有负文字，则称它是一个规则。
- 一个事实（或者称元子句）是只包含一个正文字的子句。
- 一个目标子句是不包含正文字的子句。在 Prolog 中目标子句作为一个问题来输入。一般形式为： $?-B_1, B_2, \dots, B_n$ 。
- 一个 Prolog 程序是一个只包含程序子句（规则或事实）的子句集。

Horn 子句或是程序子句或是目标字句，而程序子句或是规则或是事实。即，Horn 子句由事实，规则和目标组成。

1.5 贡献

Prolog 是当代最有影响的人工智能语言之一，由于该语言很适合表达人的思维和推理规则，在自然语言理解、机器定理证明、专家系统等方面得到了广泛的应用，已经成了人工智能领域强有力的开发语言。

第 2 章 Prolog 语言

2.1 特点

1. Prolog 程序没有特定的运行顺序，其运行顺序是由电脑决定的，而不是编写程序的人。从这个意义上来说，Prolog 程序不是真正意义上的程序。所谓程序就是按照一定的步骤运行的计算机指令，而 Prolog 程序的运行步骤不由人来决定。它更像一种描述型的语言，用特定的方法描述一个问题，然后由电脑自动找到这个问题的答案。举个极端的例子，你只需要把某个数学题目告诉它，它就会自动的找到答案，而不像使用其他的语言一样，必须人工的编制出某种算法。

2. Prolog 程序中没有 if、when、case、for 这样的控制流程语句。前面已经说了，程序的运行方式有电脑自己决定，当然就用不到这些控制流程的语句了。通常情况下，程序员不需要了解程序的运行过程，只需要注重程序的描述是否全面，不过 Prolog 也提供了一些控制流程的方法，这些方法和其他语言中的方法有很大的区别。

3. Prolog 程序和数据高度统一。在 Prolog 程序中，是很难分清楚哪些是程序，哪些是数据的。事实上，Prolog 中的所有东西都有相同的形式，也就是说数据就是程序，程序就是数据。举一个其他语言的例子：如果想用 c 语言编写一个计算某个数学表达式的程序很简单 (比如： $a=2+5*4$)，因为这是一段程序。但是如果编写一个计算用户输入的表达式值的程序就很困难了。因为用户输入的是一段数据 (字符串)，如果想让 c 语言处理这个字符串，就需要很多方面的技术。则正是因为 c 语言中，程序和数据是分开的。而在 Prolog 就不存在这个问题，你甚至可以很轻松的编写处理其它 Prolog 程序的程序。

4. Prolog 程序实际上是一个智能数据库。Prolog 的原理就是关系数据库，它是建立在关系数据库的基础上的。使用 Prolog 可以很方便的处理数据。

5. 强大的递归功能。递归是一种非常简洁的方式，它能够有效的解决许多难题。

2.2 基本结构

首先要了解谓词的概念，所谓谓词是 Prolog 语言的基本组成元素，可以是一段程序，一个数据类型或者是一种关系。它由谓词名和参数组成。两个名称相同而参数的数目不同的谓词是不同的谓词。

2.2.1 事实

事实是 Prolog 中最简单的谓词。事实用来说明一个问题中已知的对象和它们之间的关系。在 Prolog 程序中，事实由谓词名及用括号括起来的一个或几个对象组成。谓词和对象可由用户自己定义。事实的语法结构为: `pred (arg1,...argn)`。其中 `pred` 为谓词的名称。`arg1` 等为参数。

例如，谓词 `likes (bill, marry)` . 是一个名为 `like` 的关系，表示对象 `bill` 和 `marry` 之间有喜欢的关系。

2.2.2 规则

规则的实质就是储存起来的查询。规则由几个相互有依赖性的简单句（谓词）组成，用来描述事实之间的依赖关系。从形式上来看，规则由坐标表示结论的后件谓词和右边表示条件的前提谓词组成。它的语法如下: `head: -body`。其中 `head` 是谓词的定一部分，与事实一样，也包括谓词名和谓词的参数说明。`-` 是连接符，一般可以读作如果。`body` 是一个或者多个目标，与查询相同。

例如，规则 `bird (X): -animal (X), has (X, feather)` . 表示凡是动物并且有羽毛，那么它就是鸟。

2.2.3 目标

把事实和规则写进 Prolog 程序中后，就可以向 Prolog 询问有关问题的答案，询问的问题就是程序运行的目标。目标的结构与事实或规则相同，可以是一个简单的谓词，也可以是多个谓词的组合。目标分内、外两种，内部目标写在程序中，外部目标在程序运行时由用户手工键入。

例如问题? `-student (john)` . 表示“john 是学生吗?”。

2.3 数据结构

Prolog 程序是由子句组成的，子句又是由项组成的。Prolog 的数据结构由复合项和表组成，其中表示主要的数据结构。

2.3.1 项

用 BNF 范式的形式表述 Prolog 定义的项：

$$\langle \text{项} \rangle = \langle \text{常量} \rangle \mid \langle \text{变量} \rangle \mid \langle \text{复合项} \rangle$$

项可以为常量、变量、复合项（又叫结构）。

1. 常量

常量包括原子和常数，即

$$\langle \text{常量} \rangle = \langle \text{原子} \rangle \mid \langle \text{常数} \rangle$$

其中原子用来表示个体名，谓词和函数名。

2. 变量

变量分为有名变量和无名变量。有名变量是符合规范的字符串，而无名变量用下划线表示。实际上只是一个占位符。

3. 复合项

复合项又称为结构。事实子句的个体是项，若该个体还含有个体，有嵌套，即为复合项。定义为：

$$\langle \text{复合项} \rangle = \langle \text{函数符} \rangle (\langle \text{项} \rangle \{ \langle \text{项} \rangle \})$$

$$\langle \text{函数符} \rangle = \langle \text{原子} \rangle$$

2.3.2 结构和树

Prolog 数据结构可以用含有若干个项（如结构）来表示，复杂的结构很适合用一颗树来表示，既形象又易于理解。对于 `xihuan(lisi,wangfang(20,nianling(1969,sheng)))` 可以用树表示为：

2.3.3 表

表是用方括号括起来的，它的元素（分量）是有序排列的，它的表示法比复合项简单，如由元素 `a`，`b`，`c` 组成的一个表，写成 `[a,b,c]` 的形式，她从左括号开



始，终止于右括号，方括号里的元素用逗号隔开，一旦安排就绪，便不能任意颠倒。表中的元素可以使常量，变量，还可以是结构或任何其他项，包括其他表，即表中可以用表，表可以多层嵌套，表示递归定义的，下面是表的简单例子：

[1,2,a,b]

[a,b,[1,2],C,[m,n]]

Prolog 把表的处理分为两部分：表头和表尾，表头和规则头毫无关系，表 [a,b,c] 的表头为元素 a，表尾是去掉表头后剩下的表。可见表尾仍然是一个表，若表中无元素则为空表。用 [] 表示。

2.4 基本原理

谓词逻辑下的归结原理是机器推理或自动推理的主要方法，可实现定理的自动证明和问题的求解，是一种可以在计算机上实现的逻辑推理算法，由于 Prolog 是基于 Horn 子句的逻辑程序，因而其运行机理自然就是基于归结原理的演绎推理，程序的执行过程就是从目标子句出发，并不断进行匹配（合一）、归结或回溯，直至目标被完全满足或不能满足为止的归结演绎推理。

2.4.1 SLD 归结

Prolog 的归结演绎方法称为 SLD（Linear resolution with Selection function for Definite clause）归结。在 Prolog 程序的执行过程中，其具体实现方法是：自上而下匹配子句；从左置右选择子目标；归结后产生的新字母表总是插入被消去的目标处。SLD 归结就是 Prolog 程序的运行机理，也就是所谓的 Prolog 语言的过程性语义。

2.4.2 回溯

Prolog 采用深度优先的搜索策略，且使用回溯机制来执行自动搜索以及眼里推理的过程。尽管 SLD 归结是可靠且完全的，但 Prolog 的实现却不是这样。常用的搜索如深度优先搜索，广度优先搜索等搜索算法均可利用 Prolog 应用，不再累述。

2.4.3 控制

逻辑程序设计的基本公式是：算法 = 逻辑 + 控制，其中逻辑部分说明的是让计算机“做什么”，控制部分则说明计算机“怎么做”。一般说来，只需要给出逻辑部分，控制部分可由逻辑程序系统自动处理。

Cut 是 Prolog 中控制搜索过程的一个非常重要的内部谓词。使用 Cut 可以对回溯过程进行剪枝。它没有任何说明性语义，但是可以对程序的运行过程起到控制作用。在一个规则中，它就像是一个目标，它会立即成功，但是不会被重新满足，如果回溯再次到达这一个目标时，不能找到另一个满足目标的成功解。

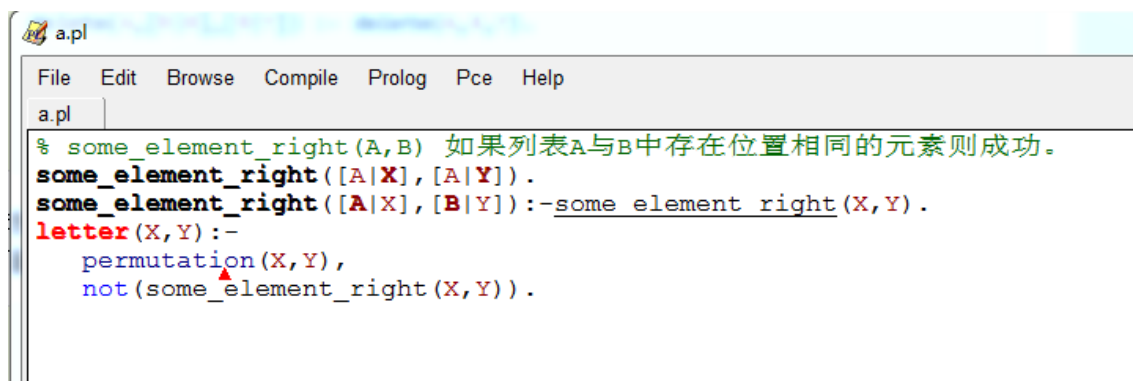
第3章 相关应用

本节将介绍 Prolog 的一些实际应用，这里采用 SWI-PROLOG 对部分项目进行演示。

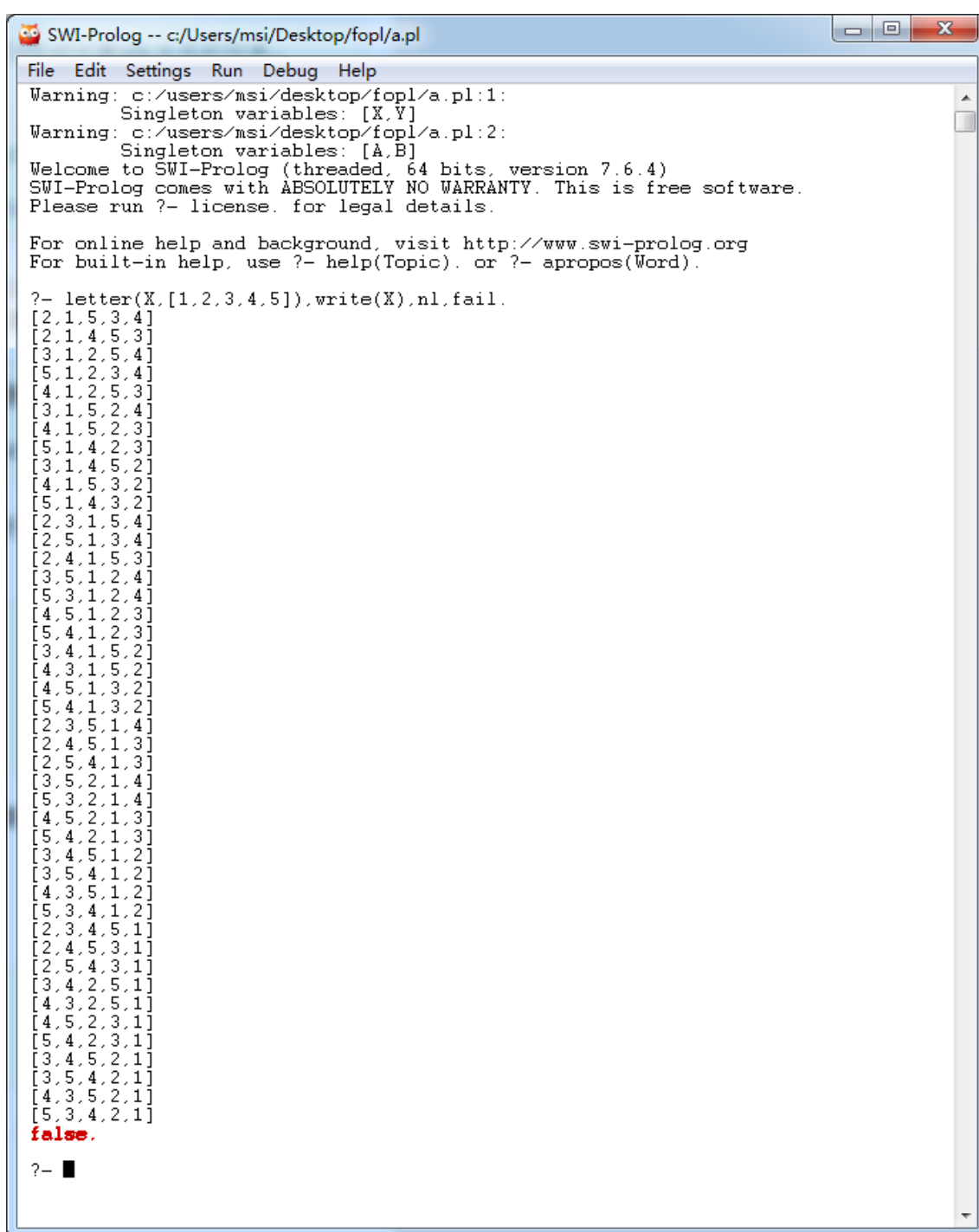
3.1 数学应用

本节我们选取古典概率中经典的问题，装错信封问题进行编程演示。问题的具体描述为：糊涂先生给他的五个朋友写信，他写了五封信，但是当他的朋友收到信后，都告诉他：“你的信寄错了”。那么请你计算一下：出现这种情况的概率有多少？（假设糊涂先生是随机地往信封里装信的），你能不能把所有情况都列出来呢。

代码如下，利用了 Prolog 的谓词进行推理，并且编写循环枚举的主程序。最终进行查询所有的结果。该部分源码将同报告一同上交以供参考。



```
a.pl
File Edit Browse Compile Prolog Pce Help
a.pl
% some_element_right(A,B) 如果列表A与B中存在位置相同的元素则成功。
some_element_right([A|X],[A|Y]).
some_element_right([A|X],[B|Y]):-some_element_right(X,Y).
letter(X,Y):-
    permutation(X,Y),
    not(some_element_right(X,Y)).
```



```

SWI-Prolog -- c:/Users/msi/Desktop/fopl/a.pl
File Edit Settings Run Debug Help
Warning: c:/users/msi/desktop/fopl/a.pl:1:
Singleton variables: [X,Y]
Warning: c:/users/msi/desktop/fopl/a.pl:2:
Singleton variables: [A,B]
Welcome to SWI-Prolog (threaded, 64 bits, version 7.6.4)
SWI-Prolog comes with ABSOLUTELY NO WARRANTY. This is free software.
Please run ?- license. for legal details.

For online help and background, visit http://www.swi-prolog.org
For built-in help, use ?- help(Topic). or ?- apropos(Word).

?- letter(X,[1,2,3,4,5]),write(X),nl,fail.
[2,1,5,3,4]
[2,1,4,5,3]
[3,1,2,5,4]
[5,1,2,3,4]
[4,1,2,5,3]
[3,1,5,2,4]
[4,1,5,2,3]
[5,1,4,2,3]
[3,1,4,5,2]
[4,1,5,3,2]
[5,1,4,3,2]
[2,3,1,5,4]
[2,5,1,3,4]
[2,4,1,5,3]
[3,5,1,2,4]
[5,3,1,2,4]
[4,5,1,2,3]
[5,4,1,2,3]
[3,4,1,5,2]
[4,3,1,5,2]
[4,5,1,3,2]
[5,4,1,3,2]
[2,3,5,1,4]
[2,4,5,1,3]
[2,5,4,1,3]
[3,5,2,1,4]
[5,3,2,1,4]
[4,5,2,1,3]
[5,4,2,1,3]
[3,4,5,1,2]
[3,5,4,1,2]
[4,3,5,1,2]
[5,3,4,1,2]
[2,3,4,5,1]
[2,4,5,3,1]
[2,5,4,3,1]
[3,4,2,5,1]
[4,3,2,5,1]
[4,5,2,3,1]
[5,4,2,3,1]
[3,4,5,2,1]
[3,5,4,2,1]
[4,3,5,2,1]
[5,3,4,2,1]
false.
?- █

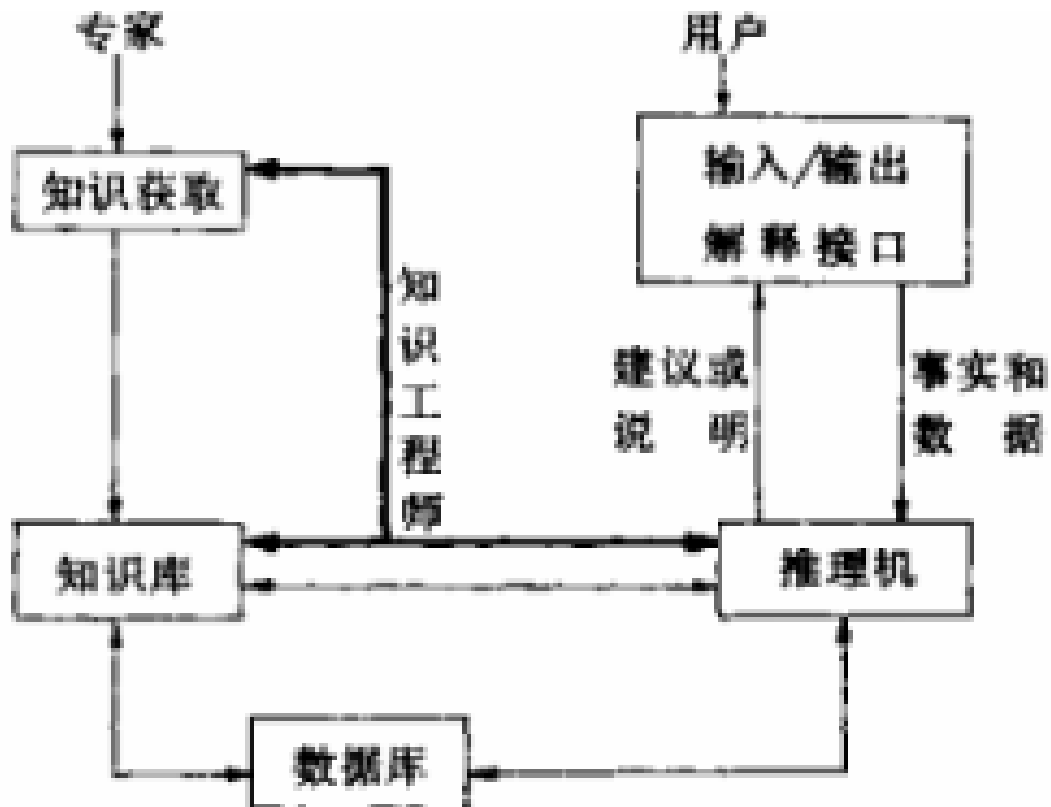
```

3.2 专家系统

专家系统是目前人工智能领域内一个比较活跃的分支。专家系统强调知识的作用，它将人类专家的特殊知识赋予机器，使机器对问题的求解达到专家水平。简单的说，专家系统是一种计算机程序，它在各特定领域中所达到的成效具有人类专家的水平。它主要基于特定领域有关问题的一个知识库，库中知识以某种特定的形式存储，以便程序能够从已经给出的事实、数据中推断出新的成果。

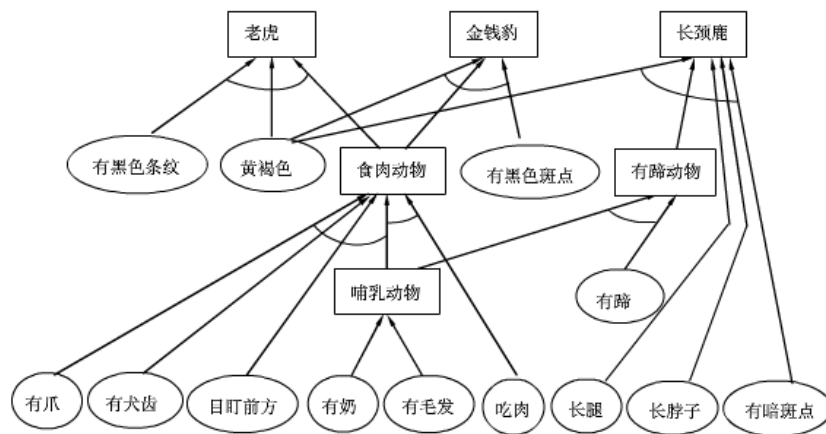
专家系统通常具有下五个部分：

- 知识库：知识库主要用来储存和管理专家系统知识，它所储存的知识主要有两类：领域的事实知识和专家的启发式知识。知识库中的知识必须以一种机器能够识别到的特定形式存放，即要用到知识表示的技术。目前常用的知识表示方法有谓词、产生式规则、语义网、框架以及过程等等。
- 综合数据库：它用于储存领域内的初始数据、证据、推理过程中得到的中间结果等等。所有的这些内容表示了专家系统当前要处理对象的主要状态和特征。
- 推理机：推理机实际上是一组程序，它用来协调控制整个系统以决定如何使用知识库中的知识，根据人类推理的方法和机器的实现方法，目前主要有三种类型的推理机：正向推理、反向推理、和正反混合推理，这三种推理方式中又可以再进行精确推理和不精确推理。
- 解释接口：解释接口是一个人机交互程序，它用来对推理路线和题目的含义给出必要的、清晰的解释，为用户了解推理过程和系统维护提供了方便的手段。
- 知识获取：知识获取部分主要是用来从人类专家哪里获取知识并且为修改和扩充知识库提供手段。简单地说，知识获取部分的建立，主要是设计一组计算机程序，使之能删除知识库中的知识并且能向知识库中加入新的知识，在自动化程度更高的系统中，还能发现原始知识库中知识的错误或能根据事先结果总结出新的知识。



各部分之间的关系可以用上述图来说明。

为了展示专家系统的应用，我们采用比较常见的小型动物分类专家系统进行演示。对于该专家系统，推理网络如下。



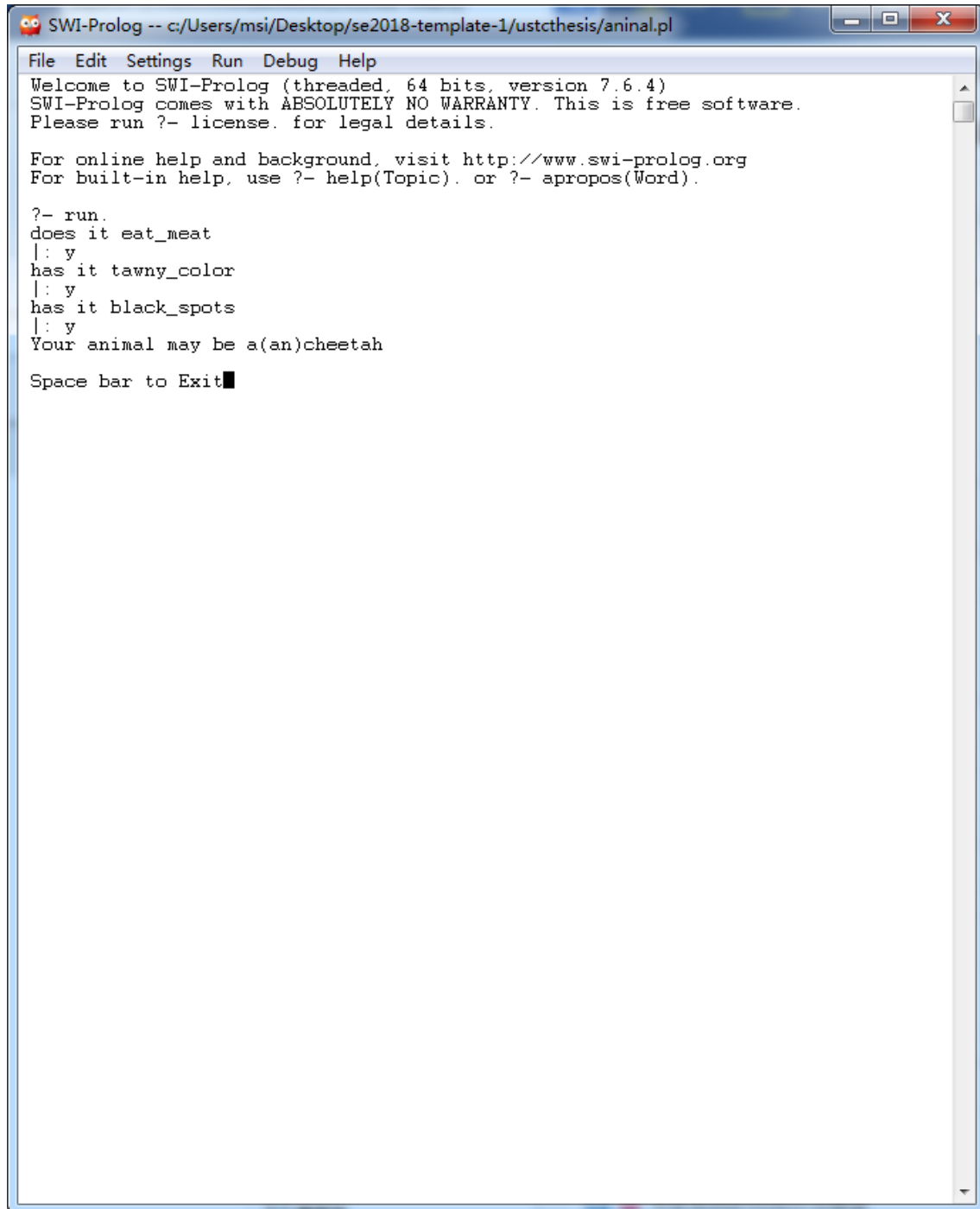
代码和演示结果如下。该部分源码将同报告一同上交以供参考。

3.3 自然语言理解

自然语言是知识的主要载体。在 Prolog 中可以采用专家系统引入自然语言理解，使自然语言文本成为专家系统的知识库，能解决知识获取的瓶颈，且使知识库有统一的表现形式，有利于知识的共享。另一方面，推理技术开发推理与问答机制，通过推理使系统获得更多，更深层次的知识，使系统理解深度的机内表示与机外表示这两种理解深度相等，以进一步增强系统的自然语言理解能力。

3.4 智能知识库

Prolog 程序可以作为一个承前启后的具有推理功能的知识库系统。Prolog 具备简略的语法、语义概念。短句的结构形式和知识表示一致，Prolog 程序的静态模式就是一个包括知识型数据的，基于一阶谓词的数据库。Prolog 的模式匹配、回溯和数据库检索功能融为一体，加上其通过递归定义新数据的功能，用归结算法对目标求解的功能，形成了她独特的功能。其核心是数据库的演绎推理机制。所以 Prolog 程序不是一个简单的数据库，而是一个承上启下的，具有智能机制的知识库系统。Prolog 在数据库方面由于语言的特性，存在着很多的弊端，因此产生了 Datalog 作为加强和补充。具体内容参见下一章。



The screenshot shows a window titled "SWI-Prolog -- c:/Users/msi/Desktop/se2018-template-1/ustcthesis/animal.pl". The window has a menu bar with "File", "Edit", "Settings", "Run", "Debug", and "Help". The main text area contains the following text:

```
Welcome to SWI-Prolog (threaded, 64 bits, version 7.6.4)
SWI-Prolog comes with ABSOLUTELY NO WARRANTY. This is free software.
Please run ?- license. for legal details.

For online help and background, visit http://www.swi-prolog.org
For built-in help, use ?- help(Topic). or ?- apropos(Word).

?- run.
does it eat_meat
|: y
has it tawny_color
|: y
has it black_spots
|: y
Your animal may be a(an)cheetah

Space bar to Exit
```

```
:-dynamic xpositive/2.
:-dynamic xnegative/2.
```

```
run:-
    animal_is(X),!,
    write('Your animal may be a(an)'),
    write(X),nl,clear_facts.

run:-
    nl,
    write('Unable to determine what'),
    write('your animal is.'),clear_facts.
```

```
positive(X,Y):-
    xpositive(X,Y),!.
positive(X,Y):-
    not(xnegative(X,Y)),
    ask(X,Y).
negative(X,Y):-
    xnegative(X,Y),!.
negative(X,Y):-
    not(xpositive(X,Y)),
    ask(X,Y).
ask(X,Y):-
    write(X),
    write(' it '),
    write(Y),
    nl,
    readln(Reply),
    remember(X,Y,Reply).
remember(X,Y,[y|_]):-
    asserta(xpositive(X,Y)).
remember(X,Y,[n|_]):-
    asserta(xnegative(X,Y)),
    fail.
```

```
clear_facts:-
    retract(xpositive(_, _)),
    fail.
clear_facts:-
    retract(xnegative(_, _)),
    fail.
clear_facts:-
    nl,
    write('Space bar to Exit'),
    readln(_).
```

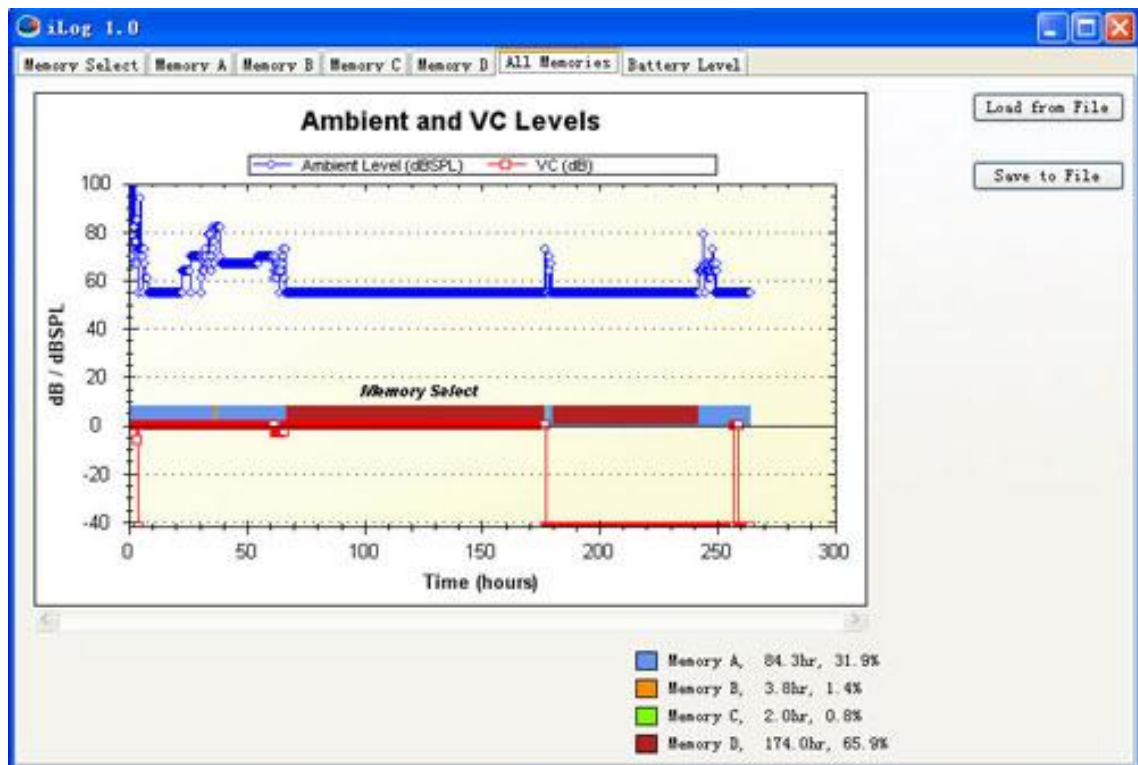
```
animal_is(cheetah):-
    it_is(carnivore),
    positive(has,tawny_color),
    positive(has,black_spots).
```

```
animal_is(tiger):-  
    it_is(bird),  
    positive(has,fly_well).  
  
it_is(mammanl):-  
    positive(has,hair).  
it_is(mammanl):-  
    positive(has,give_milk).  
it_is(bird):-  
    positive(has,feathers).  
it_is(bird):-  
    positive(does,fly),  
    positive(does,lay_eggs).  
it_is(carnivore):-  
    positive(does,eat_meat).  
it_is(carnivore):-  
    it_is(mammal),  
    positive(has,pointed_teeth),  
    positive(has,claws),  
    positive(has,forward_eyes).  
it_is(ungulate):-  
    it_is(mammal),  
    positive(has,hooves).  
it_is(ungulate):-  
    it_is(mammal),  
    positive(does,chew_cud).
```

第 4 章 Datalog

4.1 简介

Datalog 语言是一种基于逻辑编程语言 Prolog 的一种非过程化的语言。就像使用关系演算一样, 用户只需要给出所描述的信息, 不需要给出获取信息的具体过程。Datalog 语言使用声明的方式定义, 简化了简单查询的书写, 使查询优化更容易进行。Datalog 可以从数据库显示表示的事实中推理出未直接存入的新数据或新信息, 以构成内涵数据库。这些新数据或新信息实际上隐含在显示表示的外延数据库之中。Datalog 与关系演算基于同一数学背景, 即一阶逻辑。关系数据库的关系演算实际上可看成逻辑演算的一种特例。不过, Datalog 允许递归, 比关系演算表达能力更强, 因而成为对关系模型的一种极自然的扩充, 是关系数据库管理系统中扩充演绎推理功能的一种重要的研究工具。



4.2 基本结构

Datalog 语言包括了两种基本的原子,即关系原子和算数原子。Datalog 语言是由这些原子按照一定的规则组成的。

在 Datalog 语言中,关系通过成为谓词的符号来表示,每一个谓词都有固定数量的参数。关系原子是由符号谓词和其后的参数组成的,关系原子也经常简称为原子。

算数原子是两个算数表达式的比较。算数原子的值也是布尔值。

4.3 规则

Datalog 中的运算通过规则表示。规则包括如下三部分:

(1) 头部 (Head): 即一个关系原子。

(2) 符号 “ \leftarrow ”: 表示 “if”。

(3) 一个或多个子目标 (Subgoal): 即包括一个或多个原子的体 (Body)。原子可能是关系的,也可能是算术的,都以常量或变量为参数。各子目标用 AND 连接,并且,子目标前有 NOT 的子目标称为求反子目标。

规则的含义描述为: 检查规则变量的所有可能的取值,只有当这些变量具有使所有子目标为 True 时,它们在头部的取值才可以作为谓词头部关系中的结果元组。

4.4 基本关系运算

在了解基本关系运算之前,首先要明确几个概念,当 Datalog 规则运用于关系数据模型后,谓词有外延 (Extensional) 谓词和内涵 (Intensional) 谓词之分。相应地,关系也有两种存在形式: 一种以事实形式存在,称为外延数据库 (EDB), 另一种以规则形式存在,称为内涵数据库 (IDB), 即通过一个或多个 Datalog 规则计算而求得的数据库。如上例中,Student 是一个 EDB 关系,通过它的外延定义,谓词 Student 同样是一个 EDB 谓词。而关系 Newstudent 和谓词 Newstudent 都是 IDB 的。需要注意的是,EDB 谓词永远不能出现在规则的头部,但它可以出现在规则体中。而 IDB 谓词则可出现在规则的任何地方。

Datalog 中的查询是一个或多个规则聚集。如果规则头部只有一个关系出现,那么,就认为该关系的值是查询的答案。如果规则头部有多个关系,那么这些关系中的一个为查询的答案,而其它关系在答案定义中起辅助作用。

4.4.1 并

两个关系的并要用两个规则来构造。每个规则都是一个与其中一个关系相对应的原子作为它唯一的子目标, 并且, 两个规则的头部都有相同的 IDB 谓词。规则头部的参数和它的子目标中的参数完全相同。 $q = r \cup s$ 的规则表示为:

$$Q(x_1, x_2, \dots, x_n) \leftarrow R(x_1, x_2, \dots, x_n) \quad Q(x_1, x_2, \dots, x_n) \leftarrow S(x_1, x_2, \dots, x_n)$$

4.4.2 差

两个关系的差用具有求反子目标的单一规则计算。规则头部的参数和它的子目标中的参数完全相同。 $q = r - s$ 的规则表示为:

$$Q(x_1, x_2, \dots, x_n) \leftarrow R(x_1, x_2, \dots, x_n) \text{ AND NOT } S(x_1, x_2, \dots, x_n)$$

4.4.3 积

两个关系的积可用一个规则表示。规则头部的参数包括两个子目标中的所有参数, 并且参数的前后顺序要一致。 $q = r \times s$ 的规则表示为:

$$Q(x_1, x_2, \dots, x_n, y_1, y_2, \dots, y_m) \leftarrow R(x_1, x_2, \dots, x_n) \text{ AND } S(y_1, y_2, \dots, y_m)$$

4.4.4 选择

如果选择的条件都是与 (AND) 操作, 则用单一规则表示, 而每一个条件作为一个算术子目标, 并且用 AND 相连。例如: $q = \sigma_{x < 10 \text{ and } y > 100}(r)$ 的规则表示为:

$$Q(x_1, x_2, \dots, x) \leftarrow R(x_1, x_2, \dots, x_n) \text{ AND } x > 10 \text{ AND } y < 100$$

如果选择涉及到 p 个条件的或 (OR) 操作, 则需要用 p 条规则表示, 其中每条规则都定义相同的头部谓词。第 i 条规则对 p 个条件中的第 i 个进行选择。例如: $q = \sigma_{x > 10 \text{ or } y < 0}(r)$ 的规则表示为:

$$Q(x_1, x_2, \dots, x) \leftarrow R(x_1, x_2, \dots, x_n) \text{ AND } x < 10 \text{ AND } y < 0 \quad Q(x_1, x_2, \dots, x) \leftarrow R(x_1, x_2, \dots, x_n) \text{ AND } x >$$

4.4.5 投影

关系的投影可以使用具有单一子目标的单一规则实现。头部的参数是按要求的顺序与投影的属性表对应的变量。例如: $q = \pi_{x_1, x_2, x_3}(r)$ 的规则表示为:

$$Q(x_1, x_2, x_3) \leftarrow R(x_1, x_2, \dots, x_n)$$

4.4.6 其他

其它关系运算的 Datalog 规则如:

交: $Q(x_1, x_2, \dots, x_n) \leftarrow R(x_1, x_2, \dots, x_n) \text{ AND } S(x_1, x_2, \dots, x_n)$ 表示 $q = r \cap s$

自然连接: $Q(a, b, c, d) \leftarrow R(a, b) \text{ AND } S(b, c, d)$ 表示 $q = r \bowtie s$

4.5 多重查询

Datalog 规则不仅可以模拟关系代数的单一运算, 实际上还可以模拟任何代数表达式。方法是检查关系代数表达式对应的表达树, 并为树的每个内部节点建立一个 IDB 谓词。每个 IDB 谓词对应的一个或几个规则是把运算符用于树的相应节点所需要的。代表数据库中关系的叶子节点由相应的 EDB 谓词表示, 本身是内部节点的关系也由相应的 IDB 谓词表示。

4.6 递归查询

Datalog 规则不但可以实现关系代数的所有查询, 而且可以表达关系代数不能表达的递归查询。Datalog 规则的递归性在于: 允许一个或多个 IDB 关系通过体中使用相同关系的规则来定义。递归的结果是规则所定义的最小关系元组 (最小固定点), 也就是使规则头部恰好等于规则体所隐含的内容。

张文星 (1989); 李娜 等 (2009); 王湘云 (2008); 赖朝安 等 (2003); 鲁应书 (1987); Bratko (2001); Ceri et al. (2002); De Raedt et al. (2007); Gelfond et al. (2002); Nilsson et al. (2006)

参考文献

- 张文星. 1989. 专家系统原理与设计 [M]. [出版地不详]: 武汉测绘科技大学出版社.
- 李娜, 王湘云. 2009. 基于谓词逻辑的 prolog 程序设计 [J]. 西南大学学报 (社会科学版). 35(6): 48–52.
- 王湘云. 2008. 基于谓词逻辑的知识表示和知识推理及在 Prolog 中的实现 [D]: [PhD]. [出版地不详]: 南开大学.
- 赖朝安, 孙延明, 齐德昱, 等. 2003. 基于自然语言理解的专家系统研究 [J]. 计算机工程. 29 (1):20–22.
- 鲁应书. 1987. Prolog 程序和知识库系统 [J]. 山西大学学报 (自然科学版). (2):24–26.
- Bratko I. 2001. Prolog Programming for Artificial Intelligence[M]. [S.l.]: [s.n.].
- Ceri S, Gottlob G, Tanca L. 2002. What you always wanted to know about datalog (and never dared to ask)[J]. *IEEE Transactions on Knowledge and Data Engineering*. 1(1):146–166.
- De Raedt L, Kimmig A, Toivonen H. 2007. Problog: A probabilistic prolog and its application in link discovery[C]//International Joint Conference on Artificial Intelligence. [S.l.], 2468–2473.
- Gelfond M, Leone N. 2002. Logic programming and knowledge representation—the a-prolog perspective □[J]. *Artificial Intelligence*. 138(1):3–38.
- Nilsson U, Maluszynski J. 2006. Logic programming and prolog[M]: volume 2. [S.l.]: [s.n.], 53–62.