计算机科学 1994 Vol. 21 №. 2

Prolog 语言实用化新进展

高全泉
（中国科学院数学研究所　北京 100080）

摘　要　Prolog has already 20 years history. In its development, in order to make it suit to the requirements of industrial application, people always work for improving its programs excution efficiency and language functions. In this paper, we will introduce two recent advanced prolog programming environments which have considerable improvement in efficiency and language functions.

一、引言

二十年前，一种新型的计算机语言由法国马赛大学首次提出，这就是如今尽人皆知的 Prolog。多年来，英国艾丁堡大学在 Prolog 的理论、实现及推广方面，做出了重要贡献。当然，人们也不会忘记日本所做的大量工作。第五代机计划的实施，在推动和促进 Prolog 语言及知识处理系统的研究与发展方面所产生的作用不可低估。

应用 Prolog 的说明性语义和逻辑特性，可以快速开发出不易出错的程序。简单易学是它的另一优势。这使得它得以广为传播，在一段时期内独领风骚。对它的不足，人们谈得最多的是程序在计算机上的执行效率不高，这也直接影响了 Prolog 的工业化应用。实际上，Prolog 运行效率低（相对于过程型语言）既有语言的内在因素，也有实现技术方面的原因。以往的许多工作都与提高效率不无关系。如，研制专用的 Prolog 计算机及相应的并行系统，采用优化编译及在语言中引入能提高执行或搜索效率的成份等。这些工作曾经有过不可否认的成就。

最近，Prolog 语言在实用化方面又有较大的发展。这里选取 93 年 7 月在法国举行的第十三届国际人工智能大会（即 IJCAI' 93，专门设一 AI 软件展览）上展出的两个具有代表性的 Prolog 环境（已作为商品）在后文分别介绍。其一是比利时 BIM 公司推出的当今执行速度最快的 BIM Prolog V4.0，它在 SPARC station 2 上的运行速度是几年前日本研制的 Prolog 计算机 PSI 的 25 倍。其二是曾经发明了 Prolog 语言的马赛大学及 PROLOGIA 组织的新奉献——Prolog Ⅱ环境。Prolog Ⅱ中引入了约束消解的思想，设计程序时，通过在子句中安排对变量的约束条件而达到更为有效的搜索。Prolog Ⅱ已被成功地用于许多包括工业应用在内的领域。

二、BIM Prolog 的基本情况

BIM Prolog 是一个专业的 Prolog 实现，实现时严格遵循了标准化的原则。它强调性能、健壮性（亦称鲁棒性）、易于开发和灵活集成。为了适应在工业环境中的应用，更加重视性能和健壮性。

1、性能及健壮性

BIM Prolog V4.0 是目前市场上见到的最快专业型 Prolog 实现，在 SPARC station 2 上，执行基本的 reverse 测试程度，执行速度为 750,000LIPS（LIPS 是每秒钟执行逻辑推理的步数）。编译系统严格按照基于 WAM 的方法设计和实现，并采用了一系列先进的运行时优化技术。编译时，源程序首先被编译成中间代码（抽象机指令）；然后，中间代码被翻译成宿主机上的机器指令。编译的第一阶段能提供如同解释程序那样的灵活性，编

高全泉　副研究员，主要从事人工智能和知识工程方面的研究。

译成直接的机器代码又保证了执行速度的无可匹敌。另外，通过在程序中使用模式和索引说明，用户还能进一步使得机器指令的产生优化。

在健壮性方面，它的执行栈可能自动地扩展。在信号处理、错误恢复机制等方面，BIM Prolog 运用了大量独具特色的技术，确保了工业性应用强调的健壮性。例如，精心设计的存贮管理、用户可控的废料回收程序等。废料回收程序作用在由系统维持的所有内部数据结构上。

2. 易于开发和灵活集成

BIM Prolog 是一个程序开发环境。其核心，即 Prolog 机，提供了完整和一致的内部谓词集，数量在 500 个以上。文法上，与艾丁堡（DEC-10）文法兼容，并充分考虑到由标准化协会定义的标准。

BIM Prolog 的窗口环境把 Prolog 机、编译程序、查错程序以及它们对应的浏览器和控制窗口集成为一个平滑的、由鼠标驱动的用户界面。

查错程序包括标准的 Prolog 查错功能以及新的查错、延迟执行与分析技术。由于引入了源级查错思想，因而使得查错过程更加自然和有效。延迟执行分析实现一种特殊的查错算法，允许程序员根据自己程序的具体情况裁剪查错策略，从而使之更合用。

BIM Prolog 程序设计环境在 SUN windows、OPEN windows、X11/R4 以及 OSF/Mutif 之下操作。为了能在工业环境中使用 Prolog，要求 Prolog 具有灵活的各种软件接口。BIM Prolog 中，可通过一个先进的外部语言界面访问外部世界。只需通过一组 Prolog 说明，程序员就能访问用 C、Pascal、Fortran、汇编语言写成的许多已有的函数库。整数、实数（单精度或双精度）、字符串、指针、Prolog 项作为基本类型传递，结构作为表或数组传递。由于外部语言界面是双向的，故也为程序员提供一组外部程序，这组外部程序可以调用 Prolog 谓词，定义回溯程序，或者在所处的外部语言中直接处理 Prolog 项。另外，还应用内部递推机制简化 Prolog 非确定性的管理。

BIM Prolog 的嵌套性保证了 Prolog 应用的集成，允许 main C（即 C 主程序）调用 Prolog 子程序。

在开发完全集成式的工业应用中，BIM Prolog 是一个开放系统，同时它也是一个灵活的构件块。外部语言界面已被成功地用于将 BIM Prolog 与 Xview、Motif、Xlib、Xt、SunView、SunCore、PixRect、SunPHIGS 耦合，以及 BIM Prolog 与 Ingres、Unify、ORACLE、Sybase 的成功耦合。这使得应用系统的建造者可以不受限制地访问图形和窗口功能，也能访问关系数据库中可使用的数据。

3. BIM Prolog 环境的总体结构

BIM Prolog 环境的总体结构如图 1 所示。

三、Prolog 版本Ⅲ

Prolog 版本Ⅲ，即 Prolog Ⅲ，是法国马赛大学和 PROLOGIA 组织（队）合作多年的结晶。由马赛大学提出的 Prolog Ⅰ，是世界上第一个 Prolog 版本。当年（1974），D. Warren 结束了对马赛大学的访问回到艾丁堡大学之后，出于对 Prolog Ⅰ的满腔热情，决定建立一个 Prolog 的编译系统。研制期间，他对 Prolog Ⅰ文法做了少量的修改，主要是变量名字（以大写字母打头）和表文法的改动，形成了后来流传最广的艾丁堡（DEC-10）文法。Prolog Ⅱ是在 1978 年提出的，文法更为清楚和严谨，并首次引入约束程序设计思想，被认为是约束逻辑程序设计语言（Prolog Ⅲ、CLP、CHIP）的先驱。之后，在 Prolog Ⅱ的基础上推出了 Prolog Ⅱ$^+$。从 1990 年起，又提出了他们的第三个 Prolog 版本，并在 Apple MACITOSH 和 SUN 机上开始 Prolog Ⅲ的商品化。

二十年来，马赛大学及 PROLOGIA 一直始终不渝地开展 Prolog 的研究及发展，这种锲而不舍的精神使得 Prolog 的功能和效率都有较大提高。在 SUN 3 上实现的 Prolog Ⅱ$^+$

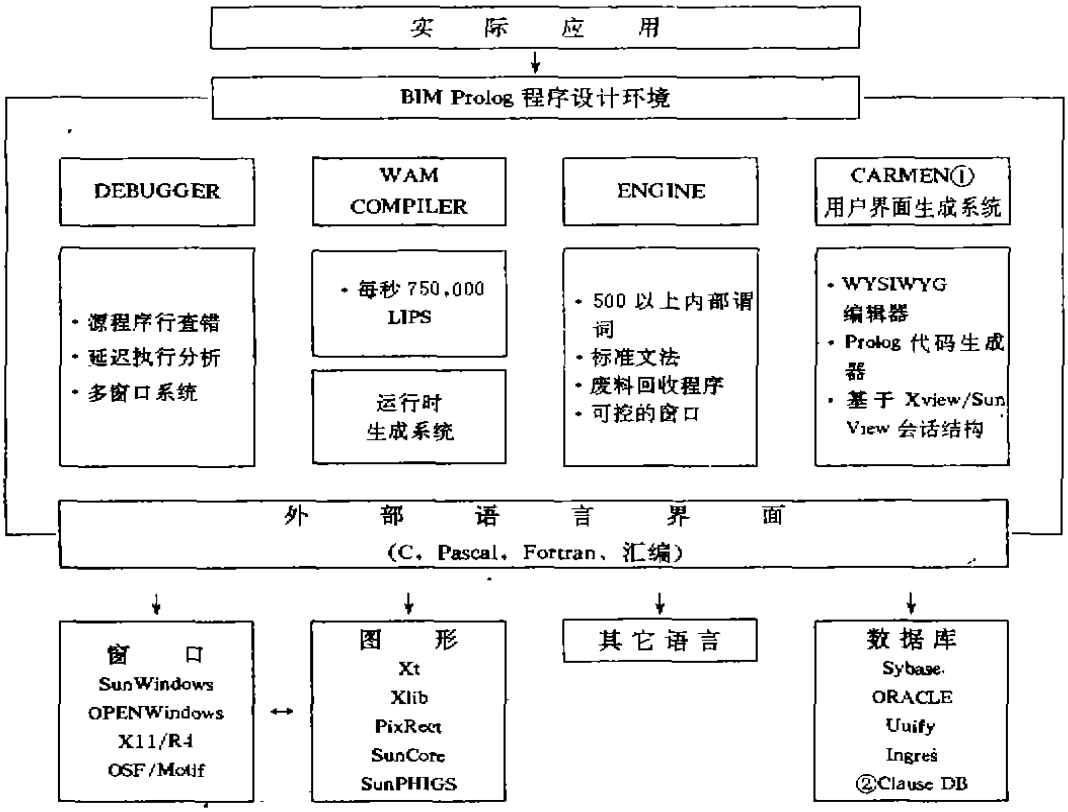

图 1 总体结构示意图

注释：①CARMEN 代表 Computer Aided Representation and Management of Environment Network。是一个基于图形编辑器的交互式用户界面生成系统。②Clause DB 是一个任选的子句数据库管理系统。对于规模适中的基于知识的应用，可通过该系统将这些子句长期保存。通过与 BIM Prolog 的紧耦合，Clause DB 包含了几种经精心设计的、能提供最佳访问速度的索引策略。

编译，代码执行速度为 33K LIPS，Prolog Ⅲ的速度(未见到准确数据)当应不低于此。Prolog Ⅲ也是作为一个程序设计环境开发的，除了有很好的内存管理策略、废料回收程序以外，也具有与图形、窗口及多种外部语言的各种界面，可以适应工业化的应用。事实上，Prolog Ⅲ已被成功地用于多个领域，开发出一批投入实际应用的系统（这些系统都具有一定难度，有的用别的工具很难做到）。比如：

- 声纳性能的优化处理；
- 社会经济学系统仿真；
- 木板材切割优化；
- 默西迪斯六缸发动机（带 Bosch 点火装置）故障诊断；
- 实用空间优化；
- 网络配置的自动生成系统；
- 化学假说推理；
- 车辆运输调度；
- 工业线性优化等。

作为一个环境，Prolog Ⅲ与其它环境有许多大同小异之处。故此，我们在下面只对其核心，即 Prolog Ⅲ语言进行简略的介绍。

1. Prolog Ⅲ的基本思想

Prolog Ⅲ在语言的核心中引入了约束消解，从而引起了逻辑程序设计的变革。这种思想源于为了准确地处理树和表，同时也为了能够把布尔和代数处理引入到 Prolog 语言中。通过在子句里增加一组用布尔和代数形式表示的约束条件，对子句内的变量的例化值进行检测，从而将那些包含不满足约束条件的变量的子句加以排除，即从搜索树中剪掉不可能正确求解的分枝，做到较为有效的

搜索。约束条件也可看作一组广义的有助于提高搜索效率的启发式信息。对它，程序员只需加以说明，其运用由系统自动完成，约束条件的主要形式是相等和不等。

2. 语言的基本元素

1）变量。如同数学中变量的用法，Prolog Ⅱ的变量用于表示未知的值。这些值可以是树、由标识符表示的树的节点、关系、实数、布尔值、字符，以及将它们用〈〉括起来的元组。程序的最基本的功能是确定一组变量的可能值。

2）运算。定义了以下几种范畴的运算

a）树和元组运算

树的构造 X_1（X_2，X_3，…，X_n）

一般树的构造 X_1〔X_2〕

元组的构造〈X_1，X_2，…，X_n〉

元组的连接 $X_1 \cdot X_2$

b）数字运算 N个运算量的取正、求负、加、减、乘、除。

c）布尔运算 非、与、或、蕴含和等价。

3）约束。约束由一定数量的关系构造。关系分为二元关系和N元关系。

二元关系是等于、不等（≠）以及数值关系“大于”、“大于等于”、“小于”，“小于等于”。

N元关系用来约束（或限制）项的，对一组树，可根据N元关系，检测它们是否满足所给性质（如儿子的个数、初始标记性质等）。

我们来看几个约束的例子。

例1 在规则

solve（X，Y）→

{X≥0，Y≥0，X+Y=12，2X+4Y=34，Y≠X−1}；中，花括弧内是数值的约束。“≠”表示不等。这一规则的体为空。“→”等同于“：−”（艾丁堡文法）。

例2 规则

isolate（N，L，X）→{U::=N−1，L=U·〈X〉·V}；中，约束条件U::=N−1表示U是由N−1个元素构成的元组；而L=U·〈X〉·V则表示L是另一个元组，这个元组是由元组U、X之值及V顺序连接而成的。

例3 规则

and（〈〉，1'）→；

and（〈B〉·L，B''）→and（L，B'），{B''=B & B'}；中，第一条规则的and的第一个参数〈〉表示空元组，第二个参数1'表示逻辑真（TRUE），这对两个参数的值进行了限定，是一种简略的表示形式，作用与and（X，Y）→{X=〈〉，Y=TRUE}；相同。第二条规则中，B & B'表示B''是B与B'的逻辑“与”的结果。这里是使用布尔约束的情形。

4）程序。Prolog Ⅱ程序由一组规则组成。每条规则的形式为：$t_0 \rightarrow t_1 t_2 \cdots t_n$，S；其中$t_0$是表示规则头的项，$t_i$（i=1，2，…，n）是要被消去的各个项，S是必须被满足的约束集。

从运算的观点来看，这种规则的含义可被简单地表示为：执行过程t_0的一个分枝时，首先把S加到当前的约束集中，确保该操作不引起任何不一致，并成功地执行过程$t_1 \cdots t_n$。

3. 基本机制

1）约束消解。约束消解是Prolog Ⅱ中的核心成份。在问题处理过程中的每一步，它确定是否存在一个满足当前约束集的解。若这样的解不存在，推理机就搜索求解该问题的别的分枝。

约束消解过程中，变量的例化值被传播到约束集中，推理机根据变量的例化值判定其是否满足约束集中的关系。

我们可以简单地刻画推理机的工作方式。不妨用（W，$t_0 t_1 \cdots t_n$，S）表示在任一给定时刻机器的状态，其中W是变量集，$t_0 t_1 \cdots t_n$是要消去的项的序列，S是需要满足的约束集；用$S_0 \rightarrow S_1 \cdots S_m$，R表示一条规则，对它的运用将得到下一状态；那么，当$S_0 \rightarrow S_1 \cdots S_m$，R被应用后，得到的机器的新状态是：

（W，$S_1 \cdots S_m t_1 t_2 \cdots t_n$，$S \cup R \cup \{S_0 = t_0\}$）

2）未知变量的探测。程序执行期间，Prolog Ⅱ探测那些其可能的值域已被归约到一个元素的变量。这样做的好处是不等约束和延迟机制能够被十分准确地处理。

3）约束集的化简。在大多数代数里，约

束集化简功能通过消去不必要的变量以及一定的冗余元素，简化约束体系。

4)不等关系的综合处理。Prolog Ⅱ能够使得不等约束被应用到任何项上。也就是不论原子、变量还是元组，均可施以不等约束。项的这种一般性处理在约束引起不同代数间的影响时，要求对这些约束做良可调的管理，类似于不同类型的量的运算那样。

4. 关于表和元组

Prolog 中，表概念几乎无所不在。Prolog Ⅱ中，表概念被扩展为元组的概念。元组是带有连接运算的有限树序列。在这些对象上，可以应用不等或相等约束。可以把一个元组加以约束，使其具有确定个数的元素。

元组上的约束的主要限制是：在任一连接运算中，位于连接符左边的元组（子元组）的元素个数必须是已知的。虽然如此，仍使用一个自动的延迟机制处理约束，从而使得元组上的约束不受前述限制。

表是一个在 Prolog 中具有特定含义的数据结构。通过以上提到的元组代数方式，可以将表表示为更为简单并且可更为高效地加以处理的形式，对它，处理速度之快令人吃惊。例如，通过使用适当的约束，可以获得一个元组的任一元素的直接访问，这使得元组更为接近数组（达到按下标访问的效果）。在一些特别强调效率的场合，比方说自然语言处理，就应充分应用元组这种数据结构的特点。

Prolog Ⅱ中，所有先前 Prolog 中使用的各种类型的表处理可以容易地通过元组和连接的方式加以执行。为了与已有程序兼容，Prolog Ⅱ也支持以艾丁堡文法写的标准 Prolog 表的使用（事实上，Prolog Ⅱ同时支持艾丁堡文法）。

5. 程序举例

这里给出一个用 Prolog Ⅱ写的递归排序程序，它应用了元组的特性。

```
sort (〈〉, 〈〉) →;
sort (L, U·〈X〉·V) →partition (L, X, U', V')
                     sort (U', U)
                     sort (V', V);
partition (L, X, U', V') →
    trunc (N/2, N') /*N 是 L 的元素数，除以 2
    regroup (X, L', U', V'), (截尾除) */
    {L::N,
    U::N',
    L=U·〈X〉·V,
    L' =U·V};
regroup (X, 〈〉, 〈〉, 〈〉) →;
regroup (X, 〈X'〉·L, 〈X'〉·U, V) →
    regroup (X, L, U, V),
    {X' <=X};
regroup (X, 〈X'〉·L, U, 〈X'〉·V) →
    regroup (X, L, U, V),
    {X' >X};
```

四、结束语

当今，欧洲及其它一些国家在 Prolog 方面从事着较以往更为务实的工作，力图在实用性方面也使得它能与其它语言媲美，在工业化应用中充分发挥作用。根据最近从国际上了解到的情况来看，Prolog 语言及其开发应用仍然是一个热点。

本文参考了 IJCAI' 93 会议的有关资料。

致谢 作者对陆汝钤教授的帮助诚致谢意。

（接第 67 页）

题，即使是过程性的语言，其程序正确性验证也尚未完全进入实用阶段。作为后起之秀的面向对象的程序设计语言和设计方法，还有很多问题值得探讨和研究。一方面，面向对象的程序虽然在程序结构上有其特点，但其数据操作与过程性的语言是基本一致的，这使得我们可以采用传统的方法进行正确性验证。另一方面，针对面向对象的特点，必然要采取一些与之相适应的方法，要在传统的理论上作一些新的探讨。

参考文献

[1] B. Meyer, Eiffel: The language, Prentice Hall, 1992

[2] 徐家福、王志坚、瞿成祥，对象式程序设计语言，南京大学出版社，1992

[3] 陆汝钤，计算机语言的形式语义，科学出版社，1992

[4] 陈火旺、罗朝晖、马庆鸣，程序设计方法学基础，湖南科学技术出版社，1987

[5] Hoare, C. A. R., An Axiomatic Approach to Computer Programming, Comm. of the ACM, Vol. 12, No. 10, 1969

[6] Dijkstra, E. W., A Displine of Programming, Prentice Hall, Englewood Cliffs, 1976