国防科技大学学报
JOURNAL OF NATIONAL UNIVERSITY OF DEFENSE TECHNOLOGY
一九八七年第三期　总第五十九期　No.3 1987 Sum. 59

顺序推理机研究中的几个重要问题

李良良　张晨曦

（计算机科学系）

摘　要　文章论述了顺序推理机的发展经验和困难，包括数据结构的表示方法，机器指令系统的设计、编译问题、共享与复制问题、存储管理问题以及内部谓词的实现等等，文章还分析了研究顺序推理机在新一代机研究中重要作用，以及在我国研究顺序推理机的必要性和可行性。

一、引　　言

顺序推理机的研究是伴随着Prolog[1]的出现和发展而产生的，在日本的第五代计算机工程[7]中又被进一步明确为主要的初期研究项目，将近十年来已陆续出现了许多实验样机和模型，如美国的PLM[4]，WAM[10,11]，英国的ZIP[2]、以及日本的PSI[9]、HPM[8]和PEK[13]等。这里，顺序推理机系指基于Prolog语言的顺序解释原理，按深度优先搜索策略，对问题求解空间进行归纳和求解，用硬、固件支持Prolog语言实现的专用计算机。

由于均基于几乎相同的顺序Prolog解释原理，迄今提出的各种顺序推理机方案具有很多共同点，本文将讨论这些方案在系统结构设计与实现方面的经验和困难，包括数据项表示方法、指令级别的高低选择、编译问题、结构共享与复制存储管理问题，以及如何实现内部谓词等等，最后在结束语中，谈谈我们对于研究顺序推理机的看法，以及在我国开展这一研究的现实意义和可行性。

二、问题与分析

2.1　带类型标志的数据项表示方法

推理过程中最基本的操作之一是匹配（即一致化操作），为了迅速判断参加匹配变元间的等价或相容关系，必须引入快速数据类型检测机制。数据项设置类型标志能满足这种要求。

本文1986年9月9日收到

标志域	值 域

这样，顺序推理机中的数据项由两部分组成：标志域和值域。原则上，只需定义三种类型标志，便足以支持 Prolog 语言的实现，即引用指针，常量和结构等。然而，为了提高执行速度和效率，常常要对上述划分作进一步加细。例如，引用指针可再分成约束变量和未约束变量，常量可再分为原子，整数、浮点数等。表是 Prolog 应用中极为频繁出现的一种结构，因此值得专门设置表类型，而不是将其简单地表示成变元数为 2 的一般性结构。

再者，为了支持操作系统，存储管理等，还需专门设置一些其它标志。例如 PSI 中每个数据项的标志域里设有 GC 标志两位，专门支持存储单元的回收。

总之，类型标志的定义和设置，是一种简单性与效率间的权衡。

2.2 指令级别的选择及编译问题

现有的顺序推理机模型倾向于选择较高的指令级别。这主要是由所支持的 Prolog语言的基本特性——不确定性——决定的，这种不确定性对于描述应用问题很有用，且有利于开发并行性，然而，这却给编译程序优化生成确定性指令带来困难。其次，在 Prolog的基本操作过程中，象匹配、归约等主要操作隐含着大量的条件转移的可能性，若采用简单机器指令，不仅代码空间会大大增加，而且因频繁地判、转，导致代码执行效率极低、和引入复杂的指令预取先行部件。

较高级指令的实现，通常籍助于微程序设计技术，故速度会受到一些影响，但由于它的一系列优点，反而会提高总体性能：

(a) 高级指令代码紧缀，空间省，效率高；

(b) 微程序技术支持各式、多路条件转移；

(c) 微程序的可修改、可扩充性，这一点对于仍在发展中的Porlog 语言尤为重要。

PSI 采用了与源程序同级的指令格式。实际上就是Prolog 语言的一种结构化的内部表示形式，一个子句对应一条指令。其它机器均采用了抽象指令级。一个 Prolog符号对应一条指令[11]。广义地说来，抽象指令层次系将部分计算 (partial evaluation) 过程施于Prolog程序，使其中的未确定的操作部分，尽可能地予先确定或具体化了的结果，诸如1)隐含的操作（参数传递、变量约束、和内部谓词等）；2)结构型数据的表示和动态构造以及3)程序执行的控制流向等等，都可以不同程度地给予具体化，从而得到不尽相同的抽象指令系统。D.H.D. Warren 提出的 WAM 系统[11] 就是这样的一个有效的例子，被顺序推理机的研究者们广为采纳和借鉴。

诚然，若指望所设计的顺序推理机是一个自主、独立的系统，则指令系统的设计还应提供对系统程序设计的支持，如输入输出，进程管理和控制，存储管理等等。这类要求与机器硬件的较低层次的实现直接相关，指令级别过高，将不利于这种需求，进而言之，象 Prolog 这类描述不确性过程的语言，本身便不适合描述操作系统中的确定性过程。PSI中引入了大量的专门的内部谓词（类源语）来实现之。

下面谈一谈由于编译程序的引入而带来的两个问题。大家知道，实用的 Prolog语言支持许多元级操作和高阶谓词，如assert(X)，retract(X) 等访问和修改过程定义数据库

（代码区）的操作。使用编译程序将源程序翻译成机器指令代码，结果丧失了关于源子句形式的若干信息，使上述数据库操作难以实现。对于算术表达式的处理也遇到同样的问题，在Prolog程序执行时刻。算术表达式中的变量不仅可以约束成值、变量，还可以约束成一个新的表达式项。为了提高计算速度，提出将谓词is的右变元（可计算表达式）编译成算术指令串。但是，动态约束的表达式却具有复合结构形式。因此，须另外设置求值手段，对此类表达式作解释性求值。

2.3 存储区域的划分及存储管理问题

·存储区域的划分　这在各种顺序推理机方案中具有较大的相似性，如图1所示，

代码空间（堆）	数据空间			
	全局数据区（栈）	过程控制区（栈）	子句控制区（栈）	跟踪区（栈）

图 1　顺序推理机存储区域的划分

代码区存放着构成Prolog过程定义的代码，以及关于源程序的其它一些信息，如符号表等；全局数据区专门存放动态生成的结构数据和其它所有全局性的数据项；过程控制区，又称选择点区，存放着控制一次过程调用中可选子句的激活、结果的排序以及过程开始调用时的状态信息；子句控制区存放着维持激活子句中子目标的求解所需的一应信息，如子句中的局部变量等等；另外还有一个区，跟踪区，用于记录或跟踪一个活跃的子句对其执行环境外部的其它变量进行约束、修改的有关信息，以便在回溯时、对其作还原处理，实际上，它属于子句控制信息的一部分。

·存储管理问题　各种模型均无例外地采用了栈式存储管理方式。尽管还有其它机制也能支持Prolog的基本操作如递归调用、归结和回溯等等，但是在现有的技术水平上，用栈机制实现更加实际和自然。

就图1中的划分而言，代码区通常采用堆式结构，通过assert，retract和record等数据库类操作实现存储分配和回收，其它各区均为LIFO栈。其中过程控制区和子句控制区内的单元分配与回收系由一些专门指令完成的，不同的是全局数据区（栈）完全靠回溯的发生实现单元回收，避免了设置专门的单元分配、回收手段的昂贵代价。

然而，这种基于回溯的回收方法尽管实现简单，却可能带来新的问题。首先，有些Prolog程序设计技术是以不回溯为基础的，比如，使用Prolog的目标来模拟一种靠共享变量通信的无休止进程，这种技术愈来愈受到人们的重视，因此完全依靠回溯的回收方法将不能及时回收生成在栈中的多余的数据项。再一个问题是由于栈式管理方法自身引起的。这就是近来出现了一种采用数据结构共享的混合型语言程序设计趋势，在有些系统如Poplog、LM—prolog中，可以在Lisp和Prolog程序之间共享数据结构和调用过程。这样的系统很容易在常规的堆型虚拟机上实现，采用常规的回收方法。而在专门支持Prolog的栈式模型上编译Lisp程序几乎是不可能的，因此，将来的系统可能会围绕着一个常规堆式虚拟机器来设计。

2.4 共享与复制

这是顺序推理机设计中颇费推敲的又一个问题，它包括执行环境和结构数据两方

面。

(1) 执行环境的共享与复制

由于Prolog 语言所支持的求解的不确定性、以及回溯机制的采用，每次过程调用的执行环境均要记录下来（压入栈中），从而导致了存储空间的大量使用，因此提出了如何记录执行环境、节约空间的问题，已有的方法包括目标压栈法和环境压栈法两种。

目标压栈法是一种纯粹的复制方式，无论是归结子目标串，还是动态生成的特例项，都要复制一份，压入栈中，其优点是使许多实现环节简单化，如子句中的变量直接对应硬件寄存器，大大提高处理速度。但它的缺点也是明显的，复制开销（时间和空间）太大，以及难解决“不安全变量”问题。

现在都转向采用环境压栈法。确切地说是动态环境压栈，这里，执行环境分成为1)静态环境，即归结目标串，候选子句等，可设置一些可修改的指针指向过程定义库，实行共享；2)动态环境，包括过程调用变元、动态生成的特例项（子句变量的约束值），以及变量约束情况等，因这部分与每次过程调用及子句激活有关，只能独享。故将其复制，压入栈中。

(2) 结构共享与复制

同样，结构型数据项也可以分成静态部分（结构框架及其常量）和动态部分（结构中的变量）。结构复制系指当激活子句中生成一个源结构的特例时，将该源结构复制到环境中去，从而在以后的处理中可直接在环境栈中访问该结构；而结构共享系指只为结构中变量分配单元。对结构框架及其中常量在不同调用间实现共享。这时所生成的结构特例用一对指针表示，一指向源结构框架，一指向变量单元区(实际是环境栈地址)，源项框架的变量目中给出该变量在栈中的相对位移，即是说，通过间接手段访问结构特例。

比较而言，这是一个时间与空间，复制时间与访问时间之间的权衡问题，作为选择何种存储方式的前提，应首先弄清1)所针对的应用问题中是否频繁使用大型结构数据，以及2)平均复制时间是否大于结构共享时的访同开销。有趣的是日本的PSI采用了结构共享法，而许多其它模型却相继从结构共享改为结构复制。这是因为PSI是解释型的，而其它多半是编译型的，在编译型实现中，采用结构复制，易于实现，且效率不低。

2.5 内部谓词的实现

Prolog 语言发展成为一种功能强、实用性好的逻辑程序设计语言，根本的一条是它含有丰富多彩的内部谓词，包括输入输出类，算术比较类和数据库操作类等等，这些内部谓词，类似于传统程序设计语言中的内部过程，给用户提供了许多方便，然而，正是这些内部谓词给具体的实现带来了很大的不便，这在基于编译实现的推理机研究中，尤为如此，下面的论述主要基于WAM模型。

(1) 一般性实现方法

直接利用现有的指令系统。它包括两个方面：用一串抽象指令实现，或者用一个Prolog 过程实现。

例如，一致化谓词A＝B，可以用get指令、Put指令和unify 指令等实现之，元逻辑谓词，如var，nonvar，list，nonlist structure，atom等，可直接用类型测试转移指令

（如switoh-on-term）实现之，下述序列实现了list(A)（假定A已被取到相应的操作寄存器中了），

```
switch-on-term fail, L, fail
fail
L:
```

OR谓词“；”可以用子句索引指令（try-me-else等）直接实现之。

用一个Prolog过程实现内部谓词的例子有not(A)，和if-then-else等：

not(A)：—call(A)，!，fail；true。

if-then-else(A，B，C)：—A，!B，C。

利用上述方法来处理，均是由编译程序专门识别和完成的，毋须Prolog机提供格外的支持。

·利用推理机的功能部件。借助于推理机中提供的硬件数据通路和功能部件，利用微程序控制，也可以实现许多按第一种方法不能直接实现的内部谓词，例如，一些比较谓词<、>、<=、>=，和 =，以及一些简单的算术运算谓词（如简单的is表达式A is B+C或A is B－C等），可以直接用推理机中的ALU实现之。

这一方法的实现，要求推理机提供专门的支持，如设置escape指令，指示转入特别处理状态，以及调用相应的微子程序。

·使用外部功能部件。这一方法主要是为了解决用上述两种方法不能实现的一些谓词，例如，ALU不能支持一些复杂的算术操作（*、／、和mod等），此外，象I/O操作谓词，代码空间（即定义数据库）操作谓词（如assert，retract）等，实现起来会更加复杂，详见(2)。

使用外加部件（专用功能部件或Host机）时，也可以借助于上述的escape机制，此外，还须设置一些专门手段（微子程序），用于推理机与外加部件间的数据通信。

(2) 数据库访问类内部谓词

Proolg中的许多内部谓词，是以子句，目标或过程作为操作变元，以及要求访问代码空间，如上述的输入输出谓词和数据库访问谓词等（read(X)，write(X)，assert(X)和retract(X)等），它们超出了严格的一阶逻辑范围，属于非逻辑谓词或属于高阶逻辑谓词。

实现这些谓词的难点在于其变元所具有的二重性。变元可以是一般的数据项，也可以是可执行的目标或子句，这一点在解释型实现中是不成问题的，因为在那里程序和数据均具有一致的数据项表示方法，基于同样的解释机制，而对于编译型的推理机模型而言，通常程序与数据项具有不同的数据结构，分存在不同的区域，即是说，子句已按编译后的形式存在代码空间中，与其它结构数据分离，已丧失了其源子句形式。结果，要求实现子句代码与结构数据间动态相互转换，就十分困难了，已提出的解决方法是：

·保留一份源子句，这就像编译型的Lisp所通常实现的那样，将函数定义的S－表达式存放进与该函数名相联的特性表中。其不足之处是：将源子句和编译代码合存在数据库（代码空间）中，增加了存储管理的复杂性；在一致化过程中要复制子句；以及难以用Prolog编写子句搜索程序。

·采用反编译方法，在必要时将编译过的子句体逆编译成一个源形式的项，优点是反编译程序可以用Prolog书写（只需要不多的几个非Prolog支持子程序），每个子句都不用存储额外的内容，当然缺点也不少，例如它要求所有编译生成的代码是可转换的，这会阻碍使用许多有效的优化措施。

·在输入子句被编译的同时，构造一个相应的单位子句source，其中，输入子句被作为一个头变元，再将该单位子句编译后，存入数据库中，这一方法的优点是源子句形式被紧缩地表示为构成该源子句所必须的抽象机器代码；子句的搜索和匹配均利用抽象机的原有机制，不需要其它的子程序，缺点是每个子句实际上都被编译了两次，使编译时间多了一倍。

三、结 束 语

本文分析和论述了顺序推理机(SIM)的发展情况和存在的问题。当今，新型计算机语言，新型系统结构的研究方兴未艾，旨在适应未来社会对计算机性能的甚高要求，日本的五代机工程不过是其中的一个有声有色的例子，作为其主要项目之一的并行推理机(PIM)，正受到人们的高度重视。然而，我们认为顺序推理机的研究也是需要重视的，这不仅仅是因为按照“自底向上”的传统发展原则，顺序推理机的研究可以成为并行推理机的研究基础，而且还因为比起直接的PIM研究来，SIM的研究更具有可行性和现实意义。

众所周知，开发并行性，与因之产生的通信、管理开销的相互制约的矛盾，是并行计算机研究中的一个老生常谈的问题，并行推理机的研究也未能出乎其外，新型的程序设计语言（如Prolog），由于自身的一些特性，给充分开发并行性提供更大的可能性，同时，新的问题也随之出现。譬如，在基于AND/OR树，OR树，或其它描述方法下[3,5,6,12]，并行性的开发至少面临着a）新的冗余和无效计算的产生；b）进程派生数目的组合性剧增；以及c）派生进程的有效生命周期短促等问题，这些因素无疑加剧了并行性开发和开销的制约关系。整个研究问题的难度，是否会由于新型语言的引入而减小或增大，至今尚不得而知。

另一方面，顺序推理机的研究，逐步从对Prolog语言的软件解释、编译仿真，到抽象机模型机，已经积累了很多的经验，渐趋成熟。而且，顺序推理机可以充分借鉴数十年来在发展传统计算机上的实现经验，因此，SIM的研究是可行的、实际的，其次，Prolog等类语言由于它的优点，适合且已经用于人工智能、专家系统、数据库和知识工程等领域，但因为缺乏强有力的硬件支持，其应用范围和规模被限制住了，若能研制出比现有实现手段快一、二个数量级的顺序推理机，必将大大推进目前的应用发展速度，获得较大的社会效益，最后，研究SIM为研究PIM提供基础和工具，正如标量机之成为向量机的基础。

总之，应对顺序推理机的研究给予足够的重视。

我们的顺序推理机研究工作，是在慈云桂教授的指导下进行的，感谢慈教授审阅了本文草稿，并提出修改意见。

参 考 文 献

[1] Clocksin, W.F., and Mellish, C.S., Programming in Prolog, Springer-Verlag, 1981.

[2] Clocksin, W.F., Design and simulation of a sequential Prolog machine, New Generation Computing Vol.3, pp.101-120, 1985.

[3] Conery, J.S and Kibler, D.F., Parallel interpretation of Logic Programs, Proc. of the 1981 Conference on Functional Programming Language and Computer Architecture(1981)163-167.

[4] Dobry, T.P., DesPain, A.M., Patt, Y.N., Performance Studies of a Prolog Machine Architecture, Symposium on Computer Architecture,1985, (June 1985).

[5] Goto, A., Aida, H., Maruyama, T., Yuhara, M., Tanaka, H. and Moto-oka, T.: A Highly Parallel Inference Engine: PIE, Proc. of the Logic Programming Conference (ICOT) (1983).

[6] Kasif, S. and Minker, J., The Intelligent Channel: A Scheme for AND/OR Parallelism in Logic Programs, TR-1414, Computer Science Department, University of Maryland, (June 1984).

[7] Moto-oka, T. (ed.): Fifth Generation Computer Systems (North Holland)(1982).

[8] Nakasaki, R., et al., Design of a High-speed Prolog Machine(HPM), Symposium on Computer Architecture. 1985, (June 1985).

[9] Taki, K., et al., Hardware Design and Implementation of the Personal Sequential Inference Machine(PSI), Proc. of the International Conference of Fifth Generation Computer Systems 1984.

[10] Tick, E., Warren, D.H.D., Towards a Pipelined Prolog Processor, New Generation Computing Vol.2, pp. 323-345, 1984.

[11] Warren, D.H.D., An Abstract Prolog Instruction Set, Technical Report 309 (Artificial Intelligence Center, SRI International) (1983).

[12] Yasuhara, H., Nitadori, K., Orbit: A Parallel Computing Model of Prolog, New Generation Computing, Vol.2, pp. 277-288, 1984.

[13] PEK: Hardware Design, New Generation Computing Vol.4, 1986.

Some Important Issues in the Research of Sequential Inference Machines

Li Liangliang Zhang Chenxi

Abstract

Several important issues of developing SIMs including tagged data structure representation, design of instruction set, compiling problems, sharing and/or copying, memory management and implementation of some critical built-in predicatesJare presented in this pape. The important role of the development of SIM in the new generation computer researches, the feasibility and the necessity for China to do research on this field are also analysed in it.