计算机科学2002Vol.29№.6

一种基于规则的系统的软件体系结构的实现分析*)

An Analysis of Architecture Implementation of a Specific Rule-based System

刘晓建　陈　平　蔡希尧

(西安电子科技大学软件工程研究所　西安710061)

Abstract Rule-based system, or called production system is a widely used kind of application system and important computing model. It occupies some different features from ordinary procedure systems or event-driven systems. From the software architecture point of view, rule-based system is composed of four components, that is knowledge base, working memory, agenda and inference engine. This paper introduces a specific rule-based system—CLIPS, including its rule structure with COOL language, and then gives a detailed analysis of its implementation about these four components with architecture concern.

Keywords Rule-based system, Software architecture, Knowledge base

1. 引言

基于规则的系统或者称为专家系统、产生式系统是一种重要的应用系统，它广泛的应用于医疗诊断、航空航天、实时监控和辅助决策系统等关键领域中。国外在这方面的研究工作开展得较早，并且开发了一些产生式开发工具，如MYCIN, OPS5/OPS83, SOAR, CLIPS 等，其中 MYCIN 是由美国斯坦福大学开发的一个帮助内科医生诊治感染性疾病的专家系统，于1978年最终完成[1]。OPS5是由卡内基·梅隆大学的 J. McDermontt 和 A. Newell 等研发的一种基于规则的通用型工具[2]，自1975年问世以来已有多个版本，如 OPS5, OPS83等。这些系统都是自封闭的，它们有各自的规则表示语言，各自的推理机和集成开发调试环境。但是随着应用的不断发展，这种自封闭的专家系统已不能适应应用的多计算模型特征的要求。比如，一个指挥控制辅助决策系统就是多种计算模型的集成，其中包括雷达原始数据采集，数据处理，辅助决策和命令干预反馈等子系统，从整体来看系统的架构应当是管道-过滤器(pipe-filter)结构，但是在辅助决策子系统中用规则模型来描述更恰当。类似的系统还存在于业务特征明显，需要个性化和信息定制的领域，如电信计费系统，客户关系管理系统和 Web 应用系统等。这些应用要求用规则建模的子系统能够和其它子系统紧密地集成在一起，而不能是独立的、自封闭的工具环境，也就是要求规则的定义能够嵌入到用程序设计语言，如 C/C++, Java 等编写的程序中，使得规则能够引用程序设计语言定义的数据结构，同时应用程序也能通过特定的接口调用推理机的执行[3]。

本文首先论述了基于规则的系统的特点和软件架构，然后分析了 CLIPS 的软件架构的具体实现策略，为以后继续研究和开发推理机和程序设计语言的集成提供基础。

2. 基于规则的系统的特点

规则系统的计算风格不同于用其它编程语言，如过程语言 FORTRAN、C，函数式语言 LISP，面向对象的语言 SMALLTALK, C++等所写的程序的计算风格。在基于规则的系统中，执行控制是基于对当前系统状态数据的匹配，而不是基于静态的程序控制结构，也就是规则系统中的计算是靠"数据驱动的"，而不是指令驱动的。规则系统的基本计算单元是无序的规则，而用过程语言编写的系统的基本计算单元是顺序执行的指令[2]。基于规则的系统的执行也不同于事件驱动的系统，在那里，系统的执行是靠外界的激励，即事件来触发的，而规则系统的执行是靠内部数据状态的改变触发的[12]。规则系统的这种计算特征决定了它的实现是以对系统状态数据的表示和匹配为基础的，因此它的软件体系结构不同于数据流系统，调用返回系统和事件通信系统，而是属于虚拟机系统[4~7]。

3. 基于规则的系统的架构和基本执行过程

基于规则的系统的计算模型和过程计算模型有类似之处，但实现细节不同。它包括三个主要的部分，第一个部分是"程序"，表明了要执行的计算，它包括静态的程序和动态的活动记录；其次是"执行者"，由它来执行计算；最后是"数据"，它描述要解决的特定问题并且存储中间结果状态。如图1所示[4]。从图中可见，基于规则的系统架构由四部分构成：

(1)"工作内存"或称为 data memory，其作用是充当全局数据库，存储关于问题的事实或断定。数据是对象的实例，它可能表示物理对象，也可能表示与特定域相关的事实或与问题解决策略相关的概念对象，如 goals 对象等。工作内存相当于程序的"数据"部分。

(2)"知识库"存储了一组规则和事实，它构成了"程序"的静态部分。每一个规则有条件部分和动作部分。条件部分描述了满足规则的数据配置，动作部分表示改变数据配置的指令。

(3)系统架构的"匹配网络"是对规则集合的预编译，把规则表示为能够进行高效数据匹配的网络结构，也就是程序的伪码表示[12]。"日程"(Agenda)是"程序"的动态部分，它保存程序的执行中间过程，即在当前的工作内存状态下要执行的规则实例。

(4)"推理机"是系统的"执行者"，它执行日程中的规则实例的 RHS，输出结果或改变工作内存的内容。

*)本文受国防预研资助。刘晓建　博士生，主要研究方向为基于规则的系统，形式验证。陈　平　博士生导师，主要研究领域为软件工程。蔡希尧　博士生导师。

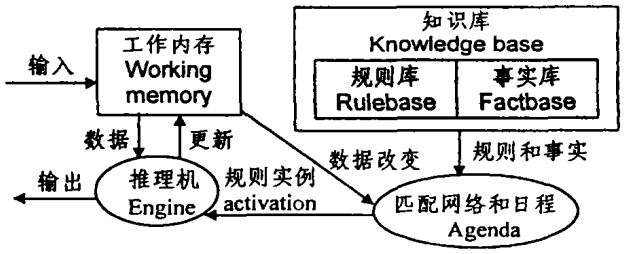

图1 基于规则的系统的体系结构

基于规则的系统的执行过程可以被描述成一个有限状态机，它由三个循环的动作状态构成：匹配规则、选择规则和执行规则。在第一个状态中，状态机根据特定的匹配算法匹配规则，以找到满足当前工作内存内容的所有候选规则实例的集合，我们称之为"冲突集"。匹配会涉及到规则的条件部分或动作部分，或两者都涉及到。同一个规则可能被不同的数据集合匹配，因此它会在冲突集中出现多次。然后状态机把冲突集传到第二个状态，即选择规则。状态机根据特定的冲突解消策略（如元规则）决定哪些规则将被实际执行。状态机迁移到第三个状态，执行被选出的规则实例。执行完之后，状态机返回到第一个状态，准备下一轮的开始。由于规则的执行可能会改变工作内存的内容，因此下一轮可能会匹配不同的规则集合。这个控制机制被称为"识别-动作循环"[2]。统计表明，在整个识别-动作循环中，匹配过程占了90%多的时间[9]，因此提高数据匹配的效率对整个规则系统的执行时间至关重要。

4. CLIPS 中知识的表示和规则的结构

CLIPS 专家系统工具是"C Language Integrated Production System"的缩写，由美国航空航天局约翰逊空间中心的软件技术部(STB)开发。首次发布于1986年。之后经过了不断的细化和改进，现在它被世人所广泛使用。目前我们见到的版本是 V6.1。CLIPS 提供了两种使用方式，一种是把 CLIPS 当作独立的工具来使用，即象上面提到的那些工具一样，必须用特定的语言编写 CLIPS 程序，并且在特定的环境下编译、调试和运行 CLIPS 程序；也可以从过程语言来调用它，完成一定的功能之后把控制返回给调用程序；同样的，用过程语言写的代码也可以被定义为外部函数，从 CLIPS 来调用，当外部代码执行完毕后，控制返回给 CLIPS。另外，在 V6.1中，CLIPS 能够支持对象机制，包括对多重继承和消息传递的支持。因此把 CLIPS 语言称为 COOL（Clips Object - Oriented Language）。

CLIPS 采用框架表示方法来表示事实，用产生式来描述知识。如 CLIPS 中用 deftemplate 结构表示事实的类型，用 defrule 结构定义规则，用 deffacts 结构定义事实库等。每一个事实由属性或"槽"(slot)构成，表明事实的某方面的特征。用规则或产生式，即"条件→动作"的集合来描述知识，其中产生式的"条件"部分称为规则的左手侧 LHS，产生式的动作部分称为规则的右手侧 RHS。当规则的 LHS 所表示的谓词公式对于当前的事实匹配为真时，执行规则的 RHS。我们通常把事实称为"工作内存元素"WME，它们的属性通常用来和规则的 LHS 进行匹配，即进行谓词测试或常量测试[8]。

产生式系统或规则系统通常由多个规则构成，描述了当系统的状态满足 LHS 所规定的约束时，系统所采取的行为。每一个规则的 LHS 都是由若干个条件元素 CE 组成，形成"模式"。每一个 CE 是针对一类 WME 的属性应满足的条件和约束的表达，它由一系列属性测试组成，这些属性测试包括两类性质不同的测试，一类是用 WME 的属性或 slot 所满足的谓词条件组成谓词测试或称为常量测试[8]，它表达了对满足该 CE 的 WMEs 的过滤操作；另一类是用属性的变量绑定表示的约束测试，它表达了匹配不同 CE 的 WMEs 之间应满足的组合条件。在进行 WMEs 的常量测试时不考虑约束测试，只有当满足不同的 CE 的 WMEs 进行组合，即作 join 操作时，才考虑约束测试。例如，下面是 CLIPS 的一段程序片：

```
(deftmplate animal
(slot type (type SYMBOL))
(slot color (type SYMBOL))
(slot size (type SYMBOL)(default lrg))
(slot hair))

(defrule rule-1
(animal (type cat)(color?c)(size lrg)(hair?h))
(animal (type dog)(color?c)(size sm)(hair?h))
⇒
(printout t"color is"?c"hair is"?h))
```

这个模式由两个 CE 组成，其中第一个 CE 包括由 type＝cat，size＝lrg 组成的谓词条件测试，第二个 CE 包括由 type＝dog，size＝sm 组成的谓词条件测试，同时匹配这两个 CE 的事实还应满足由变量绑定?c 和?h 组成的约束测试。因此整个模式表达了这样的含义，我们要找那些大猫和小狗，它们的颜色必须相同，毛发的长度必须相同。

CLIPS 中的条件元素 CE 分为七种不同的类型，如下 BNF 所示：

```
⟨conditional-element⟩::＝⟨pattern-CE⟩|⟨assigned-pattern-CE⟩|
    ⟨not-CE⟩|⟨and-CE⟩|⟨or-CE⟩|
    ⟨logical-CE⟩|⟨test-pattern⟩
⟨test-CE⟩::＝(test⟨function-call⟩)
⟨not-CE⟩::＝(not⟨pattern-CE⟩)
⟨and-CE⟩::＝(and⟨conditional-element⟩＋)
⟨or-CE⟩::＝(or⟨conditional-element⟩＋)
⟨logical-CE⟩::＝(logical⟨conditional-element⟩＋)
```

它们分别表示了不同的测试语义，⟨pattern-CE⟩表示了对一类事实的测试，⟨not-CE⟩，⟨and-CE⟩，⟨or-CE⟩和⟨logical-CE⟩是对 CE 测试进行逻辑连结后所形成的测试，⟨test-CE⟩是用 CLIPS 内部函数表达的约束测试，比如要表达变量?x"小于或等于"变量?y，可用(test＜＝?x?y))来表达。

5. CLIPS 软件架构的实现

CLIPS 是用 C 语言实现的一个运行支撑环境，从使用过程来看，CLIPS 应用程序的执行由三个基本命令组成，即 Load()，Reset()和 Run()。首先用 CLIPS 语言编写程序.clp，然后调用 Load()命令读取程序文本，分析其中的结构，如 defrule，deftemplate，deffacts 等，并且把它们编译成内部数据结构表示，建立匹配网络。Reset()完成知识库的初始化工作，并且针对事实库中的事实完成第一次匹配，把可执行的规则实例放入日程中。Run()命令从日程中取出规则实例并执行。

(1)CLIPS 中模块的实现

在 CLIPS 程序中用 defmodule 定义程序的"模块"，在一个模块内定义的所有结构，如 deffacts，deftemplate 和 defrule 只能在该模块内可见。缺省情况下的模块是"MAIN"。在实现上 CLIPS 也是以 C 语言结构 struct defmodule 来组织在该模块内出现的所有 CLIPS 结构。这些结构被映射到 C 语言的相应结构，即 struct deffacts，struct deftemplate 和 struct defrule。图2例示了在模块缺省情况下的组织结构。

其中 struct defmodule 用 itemArray 指向 MAIN 模块内表示不同结构的子模块数组，并且给每个子模块分配一个确定的索引号以便能够从模块访问子模块。同时每个子模块用

header 指针访问它所在的模块。CLIPS 程序中的每一个特定结构保存在相应的子模块中，如 deftemplate 结构保存在 deftemplateModule 中。

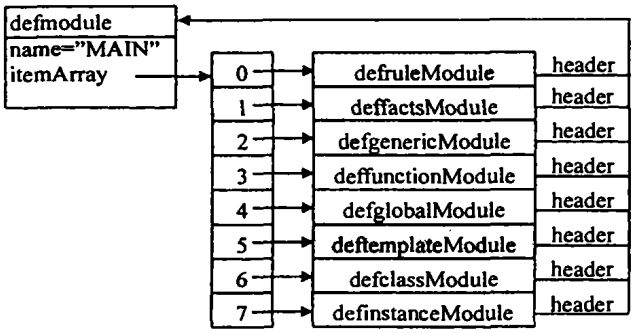

图2　模块结构的实现

(2)事实库的实现

事实库的实现包括两部分，即把 deftemplate 结构和 deffacts 结构关联到相应的子模块，形成模板库和事实库。图3例示了一个定义了三个模板，“context”，“constant”和“region”的内存组织。其中 deftemplateModule 子模块用 firstItem，lastItem 指针访问其中保存的模板，每一个模板用 whichModule 指向保存它的子模块，这样就实现了模块 MAIN 与其中定义的任一模板之间的相互访问。

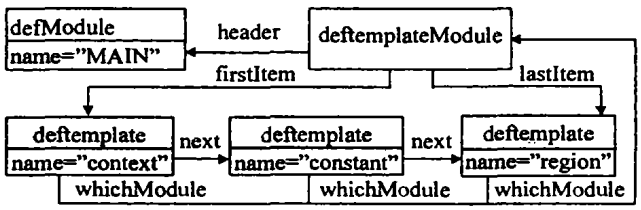

图3　模板库的实现

deffacts 结构与 deffactsModule 子模块的关联与之类似。在 Load()过程中，CLIPS 并没有把 deffacts 定义的事实插入工作内存 WM，而是将其编译成一个类型为 struct expr 的插入列表 assertList，如图4所示。其中，struct expr 的 type 字段 FCALL 表明 value 字段是一个函数调用指针，value 字段指向函数 Assert()的入口地址，argList 表明 Assert()函数要插入的事实，nextArg 字段表明下一个要插入的事实。因此通过 struct deffacts 结构保存了 CLIPS 程序中定义的事实库。

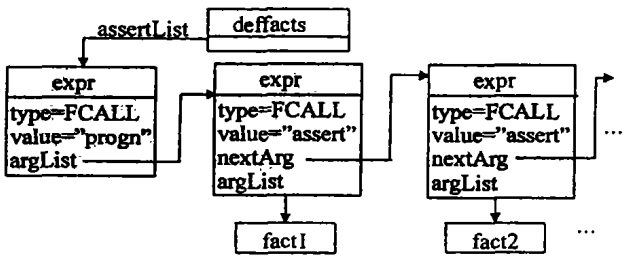

图4　插入列表的实现

(3)CLIPS 程序的 Rete 网络表示和规则库的实现

规则系统的执行是由数据驱动的，它的基础是对大量数据的快速匹配，因此模式匹配的效率直接影响到系统执行的效率。为了提高匹配效率，CLIPS 采用了对规则的预编译技术[2]和 Rete 快速匹配算法[2,10,11]，也就是把 CLIPS 程序中的规则的 LHS 表示成 Rete 网络结构，这是规则库的实现的最复杂的部分也是最重要的部分，限于篇幅这里不详述 Rete 网络的结构。规则的动作部分，即 RHS 被翻译成表达式列表。从规则系统的体系结构来看规则库的实现同事实库的实现一样，也要把 defrule 结构关联到 defruleModule 子模块上，如图5所示。

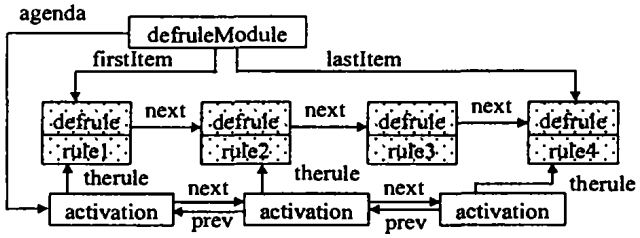

图5　规则库和日程的实现

(4)工作内存的初始化和实现

在 Load()过程中建立了静态的事实库和规则集的 Rete 网络表示，接下来 Reset()调用把事实库中的事实加载到工作内存中。在实现上，工作内存是一个类型为 struct fact 的事实列表 FactList，它维护系统当前的所有事实数据。能够访问事实列表的操作有 Assert()，Retract()和 Modify()，它们分别完成插入事实，取出事实和修改事实的功能。Reset()操作由七个子操作构成，分别完成对 deftemplate，defrule，deffacts 等结构的初始化工作，其中 ResetDeffacts()对(2)中建立的插入列表 assertList 进行求值，即根据 value 字段的 Assert()函数指针和 argList 字段表明的参数把事实插入工作内存。每插入、取出或修改一个事实时，即调用 Assert()、Retract()或 Modify()时，都会把这种数据的更新送到 Rete 匹配网络进行递增匹配，结果是在冲突集中形成匹配当前数据状态的规则实例集合。一般情况下，并非冲突集中的所有规则实例都可以执行，因此采用冲突解消策略挑选可执行的规则实例，将其放在日程中保存。

(5)推理机和日程的实现

规则系统的日程同推理机的执行密切相关。当工作内存的数据有改变时，就会匹配 Rete 网络，并且在冲突集中形成新的规则实例，利用冲突解消策略选择可执行的规则实例，加入日程中。在实现上，日程是一个类型为 struct activation 的双向链表，它和 defruleModule 子模块相关联。图5例示了一个由四个规则构成的规则库和由三个规则实例构成的日程的结构。其中，defruleModule 用 firstItem 和 lastItem 访问规则库，agenda 指针指向可执行的规则实例 activation，每一个 activation 的 therule 指针指向规则，表明这个实例是哪个规则的。推理机从日程中取出规则实例，执行它的 RHS。一般情况下，如果在规则实例的 RHS 有 Assert()、Retract()和 Modify()操作，规则的执行会引起工作内存的改变，这种改变又会引发 Rete 网络的匹配，在日程中形成新的规则实例。推理机不断的从日程中取出规则实例，直到日程为空。

结束语　本文论述了基于规则的系统的基本特征，软件体系结构及其工作流程，并针对一种特殊的规则系统——

(下转第116页)

验，算法的收敛性研究也才开始。但是，大量实验和理论证实，遗传算法用于优化领域效果显著。MGA 使用了遗传算法的基本思想，并根据领域知识设计了遗传算子。虽然遗传算法的效率不是很高，但是 MGA 比 ISODATA 更为简单。MGA 的算子代替了聚类组合、聚类分裂和聚类再计算等复杂的数学计算。

4 实验结果

实验中，我们使用 MGA 来处理一个汽车贸易公司的汽车数据集。数据包含有五个属性：1个数值属性和4个符号属性。数值属性有参考价(万元)，符号属性有编码、品牌、车型和类别。表1中，我们只用了12条记录来说明数据集。

表1 汽车描述

编码	品牌	车型	参考价(万元)	类别
0101001	奔驰	S级	60	进口车
0101002	奔驰	S级	60	进口车
0101003	奔驰	S级	60	进口车
0401001	捷达	1.6升	15	中档车
0401002	捷达	1.6升	15	中档车
0401003	捷达	1.6升	15	中档车
0401004	捷达	1.6升	15	中档车
0201001	富康	1.6升	10	中档车
0202002	富康	1.4升	8	中档车
0203001	富康	1.8升	12	中档车
0301001	派力奥	1.5升	10	经济型车
0302001	派力奥	1.3升	8	经济型车

实验中的初始参数为：$m=6$，$\alpha_c=4$，$\beta_c=0.7$，$\beta_m=0.2$，$\varepsilon=6$。当系统收敛到挖掘聚类规则的命令后，它把通过 SQL 检索到的数据映射到用户感兴趣的子空间。子空间只有两个属性：品牌和参考价。因为品牌不是一个数值属性，它需要正规化。通过一个预先定义的正规化函数 f(x)来进行正规化：f(奔驰)=1，f(富康)=2，f(派力奥)=3，f(捷达)=4，从而得到正规化后的向量(1,60)，(1,60)，(1,60)，(4,15)，(4,15)，(4,15)，(4,15)，(2,10)，(2,8)，(2,12)，(3,10)，(3,8)。

根据这些向量，我们用 MGA 来初始化种群和聚类中心表。然后对种群使用交叉和变异操作，从而产生新的聚类中心表。当一次迭代过程结束时，使用适应度函数来计算种群，并为下一次迭代保留一些结果，直到种群满足适应度函数。最后的聚类中心如表2所示。

表2 结果表

聚类中心	向量	DSE
(2.5,9)	(2,8),(2,10),(3,8),(3,10)	4.47
(1,55)	(1,50)	5.00
(4,14.5)	(4,14),(4,15)	1.00

最后，我们得到如下聚类规则：

$if(\|X-(2.5,9)\|\leqslant 1.12)$ then $X\in C_1$

$if(\|X-(1,55)\|\leqslant 5)$ then $X\in C_2$

$if(\|X-(4,14.5)\|\leqslant 0.5)$ then $X\in C_3$

结束语 数据挖掘技术包含很多知识领域，但是，由于它的发展历史不长，还没有完整的理论，因此，目前的数据挖掘技术还存在很多问题。我们在以前的一些研究成果基础上，提出了一种基于遗传算法的聚类新方法。该方法有两个优点：一是通用性强，它可以对包含数值属性和符号属性的大数据集进行聚类；二是它提高了数据挖掘的效率和质量。

参考文献

1 Weldon J-L. Data Mining and Visualization. Database Programming and Design, 1996, 9 (5): 21～24

2 Michalewicz Z. Genetic Algorithms + Data Structures = Evolution Programs. Springer-Verlag, 1996

3 Robertson G G. A Table of Two Classification Systems. Machine Learning, 1988, 3(23)

4 Jain A K, Dubes R C. Algorithms for Clustering Data. Prentice Hall, 1988

5 Murty M N, Jain A K. Knowledge-based Clustering Schema for Collection Management and Retrieval of Library Books. Pattern Recognition, 1995, 28(7)

6 Shortland R, Scarfe R. Digging for Gold (Data Management). IEE Review, 1995, 41 (5): 213～217

7 Fayyad U. Knowledge Discovery and Data Mining towards a Unifying Framework, KDD'. In: 96 Proc. 2nd Intl. Conf. on Knowledge Discovery & Data Mining, AAAI Press, 1996

8 Han J. Conference tutorial notes: Data Mining Techniques. In: Proc. of ACM SIGMOD Intl. Conf. 96 on Management of Data (SIGMOD' 96). Montreal, Canada, June, 1996

9 Data Mining: Discovering Hidden Value in Your Data Warehouse. Pilot Software. WWW files, 1996 (http:// www. pilot. com/ dmpaper/ dmindex. html)

10 陈栋，徐洁盘. Knight：一个通用知识挖掘工具，计算机研究与发展，1998，35(4)：338～343

(上接第136页)

CLIPS，分析了它的知识表示方式，规则结构和软件架构中四个部件的实现。我们下一步的工作是以此为基础，考虑如何将规则语言和程序设计语言集成，使规则能够在程序设计语言的上下文中引用定义的数据结构。

参考文献

1 Buchanan B G, Shortliffe E H. Rule-Based Expert Systems: The MYCIN Experiments of the Stanford Heuristic Programming Project. Addison-Wesley, 1984

2 Brownston L, et al. Programming Expert Systems in OPS5: An Introduction to Rule-Based Programming. Addison-Wesley, 1985

3 ILOG Rules User's Manual, ILOG Corp, 1996

4 Shaw M, Garlan D. Software Architecture: Perspectives on an emerging discipline. Prentice-Hall, 1996

5 Hayes-Roth F. Rule-based systems. Communications of the ACM, 1985, 28(9): 921～932

6 Hayes-Roth B, Pfleger K, et al. A domain-specific software architecture for adaptive intelligent systems. IEEE Transactions on Software Engineering, special Issue on software Architecture, 1995, 21(4): 288～301

7 Hayes-Roth B. Architectural foundations for real-time performance in intelligent agents. The Journal of Real-Time Systems, Kluwer Academic Publishers, 1990, 2: 99～125

8 Miranker D P. Performance estimates for the DADO machine: A comparison of TREAT and Rete. In Fifth Generation Computer Systems, ICOT, Tokyo, 1984

9 Miranker D P, Lofaso B J. The Orgnization and performance of a TREAT-Based Production System Compiler. IEEE Transactions on Knowledge and Data Engineering, 1991, 3(1): 3～9

10 Scales D. Efficient matching algorithms for the SOAR/OPS5 production system, Knowledge Systems Laboratory, Department of Computer Sciences, Stanford University, Stanford, CA, 1986

11 Forgy C L. Rete: a fast algorithm for the many pattern/many object pattern match problem. Artificial Intelligence, 1982, 19: 17～37

12 Nii H P. Blackboard systems. AI Magazine, 1986, 7(3): 38～53, 7(4): 82～107