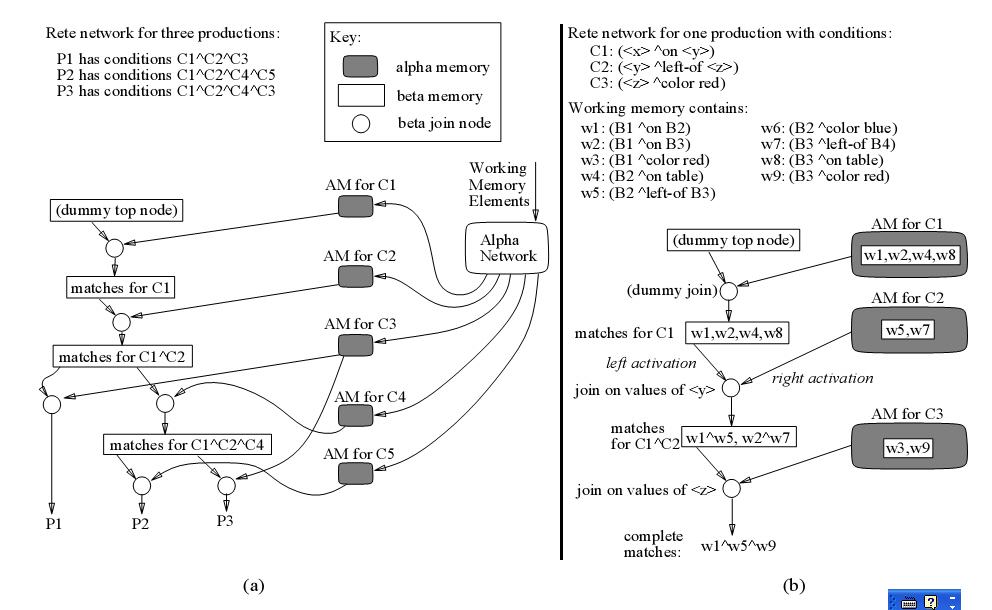
**规则引擎研究（一）——Rete算法（1）**

http://hi.baidu.com/iburu/item/78d44f18d8d7880ce75c36a7

RETE算法介绍  
**一、 rete概述**  
Rete算法是一种前向规则快速匹配算法，其匹配速度与规则数目无关。Rete是拉丁文，对应英文是net，也就是网络。Rete算法通过形成一个rete网络进行模式匹配，利用基于规则的系统的两个特征，即时间冗余性（Temporal redundancy）和结构相似性（structural similarity），提高系统模式匹配效率。  
**二、 相关概念**  
**2.1 事实（fact）：**  
事实：对象之间及对象属性之间的多元关系。为简单起见，事实用一个三元组来表示：（identifier ^attribute value），例如如下事实：  
w1:(B1 ^ on B2) w6:(B2 ^color blue)  
w2:(B1 ^ on B3) w7:(B3 ^left-of B4)  
w3:(B1 ^ color red) w8:(B3 ^on table)  
w4:(B2 ^on table) w9:(B3 ^color red)  
w5:(B2 ^left-of B3)  
**2.2 规则（rule）:**  
由条件和结论构成的推理语句，当存在事实满足条件时，相应结论被激活。一条规则的一般形式如下：  
(name-of-this-production  
LHS /\*one or more conditions\*/  
-->  
RHS /\*one or more actions\*/  
)  
其中LHS为条件部分，RHS为结论部分。  
下面为一条规则的例子：  
(find-stack-of-two-blocks-to-the-left-of-a-red-block  
(^on)  
(^left-of)  
(^color red)  
-->  
...RHS...  
)  
**2.3 模式（patten）：**  
模式：规则的IF部分，已知事实的泛化形式，未实例化的多元关系。  
(^on)  
(^left-of)  
(^color red)  
**三、 模式匹配的一般算法**  
规则主要由两部分组成：条件和结论，条件部分也称为左端（记为LHS, left-hand side），结论部分也称为右端（记为RHS, right-hand side）。为分析方便，假设系统中有N条规则，每个规则的条件部分平均有P个模式，工作内存中有M个事实，事实可以理解为需要处理的数据对象。  
规则匹配，就是对每一个规则r, 判断当前的事实o是否使LHS(r)=True，如果是，就把规则r的实例r(o)加到冲突集当中。所谓规则r的实例就是用数据对象o的值代替规则r的相应参数，即绑定了数据对象o的规则r。  
规则匹配的一般算法：  
1) 从N条规则中取出一条r；  
2) 从M个事实中取出P个事实的一个组合c；  
3) 用c测试LHS(r)，如果LHS(r（c）)=True，将RHS(r（c）)加入冲突集中；  
4) 取出下一个组合c，goto 3；  
5) 取出下一条规则r，goto 2；  
**四、 RETE算法**  
Rete算法的编译结果是规则集对应的Rete网络,如下图。Rete网络是一个事实可以在其中流动的图。Rete网络的节点可以分为四类：根节点（root）、类型节点（typenode）、alpha节点、beta节点。其中，根结点是一个虚拟节点，是构建rete网络的入口。类型节点中存储事实的各种类型，各个事实从对应的类型节点进入rete网络。

**4.1 建立rete网络**  
Rete网络的编译算法如下：  
1) 创建根；  
2) 加入规则1(Alpha节点从1开始，Beta节点从2开始)；  
a. 取出模式1，检查模式中的参数类型，如果是新类型，则加入一个类型节点；  
b. 检查模式1对应的Alpha节点是否已存在，如果存在则记录下节点位置，如果没有则将模式1作为一个Alpha节点加入到网络中，同时根据Alpha节点的模式建立Alpha内存表；  
c. 重复b直到所有的模式处理完毕；  
d. 组合Beta节点，按照如下方式：  
　　　Beta(2)左输入节点为Alpha(1)，右输入节点为Alpha(2)  
　　　Beta(i)左输入节点为Beta(i-1)，右输入节点为Alpha(i) i>2  
并将两个父节点的内存表内联成为自己的内存表；  
e. 重复d直到所有的Beta节点处理完毕；  
f. 将动作（Then部分）封装成叶节点（Action节点）作为Beta(n)的输出节点；  
3) 重复2)直到所有规则处理完毕；  
可以把rete算法类比到关系型数据库操作。  
把事实集合看作一个关系，每条规则看作一个查询，将每个事实绑定到每个模式上的操作看作一个Select操作，记一条规则为P，规则中的模式为c1,c2,…,ci, Select操作的结果记为r(ci),则规则P的匹配即为r(c1)◇r(c2)◇…◇(rci)。其中◇表示关系的连接（Join）操作。

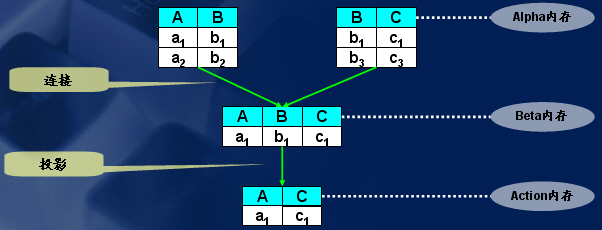
使用一个rete的过程：  
1) 对于每个事实，通过select 操作进行过滤，使事实沿着rete网达到合适的alpha节点。  
2) 对于收到的每一个事实的alpha节点，用Project(投影操作)将那些适当的变量绑定分离出来。使各个新的变量绑定集沿rete网到达适当的bete节点。  
3) 对于收到新的变量绑定的beta节点，使用Project操作产生新的绑定集，使这些新的变量绑定沿rete网络至下一个beta节点以至最后的Project。  
4) 对于每条规则，用project操作将结论实例化所需的绑定分离出来。



4.2 使用rete网络进行匹配

下面为的图示显示了连接（Join）操作和投影（Project）的执行过程。

Rete算法有两个特点使其优于传统的模式匹配算法。  
1、状态保存  
事实集合中的每次变化，其匹配后的状态都被保存再alpha和beta节点中。在下一次事实集合发生变化时，绝大多数的结果都不需要变化，rete算法通过保存操作过程中的状态，避免了大量的重复计算。Rete算法主要是为那些事实集合变化不大的系统设计的，当每次事实集合的变化非常剧烈时，rete的状态保存算法效果并不理想。  
2、节点共享  
另一个特点就是不同规则之间含有相同的模式，从而可以共享同一个节点。Rete网络的各个部分包含各种不同的节点共享。



4.3 Rete算法的特点