分类号： TP311.5 单位代码： 10335

密 级：（注明密级与保密期限） 学 号：

仅限于论文课题来源于国防军工项目，如没有请填：无



硕士学位论文



**中文论文题目 ：** （小二号仿宋体加粗）

**英文论文题目：** （小二号 Time New Roman字体加粗）

四号仿宋

申请人姓名：

指导教师：

合作导师：

专业学位类别： 工程硕士

专业学位领域： 软件工程

所在学院： 软件学院

**论文提交日期 年 月 日**

**学位论文书脊示例**

|  |
| --- |
| **题**  **目**  **作**  **者**  **姓**  **名**  **浙**  **江**  **大**  **学** |

3cm左右

3cm左右

**中文论文题目** （小二号仿宋体加黑）



**论文作者签名:**

**指导教师签名:**

论文评阅人1： (姓名\职称\单位,下同)

评阅人2： (隐名评阅学位论文省略)

评阅人3：

评阅人4：

评阅人5：

答辩委员会主席： (姓名\职称\单位)

委员1：

委员2：

委员3：

委员4：

委员5：

答辩日期：

**英文论文题目**（16pt Time New Roman，Bold）



**Author’s signature:**

**Supervisor’s signature:**

Thesis reviewer 1： (姓名\职称\单位,下同)

Thesis reviewer 2： (隐名评阅学位论文省略)

Thesis reviewer 3：

Thesis reviewer 4：

Thesis reviewer 5：

Chair： (姓名\职称\单位)

(Committee of oral defence)

Committeeman 1：

Committeeman 2：

Committeeman 3：

Committeeman 4：

Committeeman 5：

Date of oral defence：

浙江大学研究生学位论文独创性声明

本人声明所呈交的学位论文是本人在导师指导下进行的研究工作及取得的研究成果。除了文中特别加以标注和致谢的地方外，论文中不包含其他人已经发表或撰写过的研究成果，也不包含为获得 **浙江大学** 或其他教育机构的学位或证书而使用过的材料。与我一同工作的同志对本研究所做的任何贡献均已在论文中作了明确的说明并表示谢意。

学位论文作者签名： 签字日期： 年 月 日

学位论文版权使用授权书

本学位论文作者完全了解 **浙江大学** 有权保留并向国家有关部门或机构送交本论文的复印件和磁盘，允许论文被查阅和借阅。本人授权 **浙江大学** 可以将学位论文的全部或部分内容编入有关数据库进行检索和传播，可以采用影印、缩印或扫描等复制手段保存、汇编学位论文。

（保密的学位论文在解密后适用本授权书）

学位论文作者签名： 导师签名：

签字日期： 年 月 日 签字日期： 年 月 日

摘要

**关键词：**　，

Abstract

**Key Words：**,

目录

[摘要 i](#_Toc264205503)

[Abstract ii](#_Toc264205504)

[图目录 III](#_Toc264205505)

[表目录 IV](#_Toc264205506)

[第1章 绪论 1](#_Toc264205507)

[1.1 课题背景 1](#_Toc264205508)

[1.1.1 1](#_Toc264205509)

[1.2 本章小结 2](#_Toc264205510)

[1.2.1 2](#_Toc264205511)

[第2章 3](#_Toc264205512)

[2.1 第一节 3](#_Toc264205513)

[2.1.1 3](#_Toc264205514)

[2.2 本章小结 4](#_Toc264205515)

[2.2.1 4](#_Toc264205516)

[第3章 5](#_Toc264205517)

[3.1 第一节 5](#_Toc264205518)

[3.1.1 5](#_Toc264205519)

[3.2 本章小结 5](#_Toc264205520)

[3.2.1 5](#_Toc264205521)

[第4章 6](#_Toc264205522)

[4.1 第一节 6](#_Toc264205523)

[4.1.1 6](#_Toc264205524)

[4.2 本章小结 6](#_Toc264205525)

[4.2.1 6](#_Toc264205526)

[第5章 7](#_Toc264205527)

[5.1 第一节 7](#_Toc264205528)

[5.1.1 7](#_Toc264205529)

[5.2 本章小结 7](#_Toc264205530)

[5.2.1 7](#_Toc264205531)

[第6章 8](#_Toc264205532)

[6.1 第一节 8](#_Toc264205533)

[6.1.1 8](#_Toc264205534)

[6.2 本章小结 8](#_Toc264205535)

[6.2.1 8](#_Toc264205536)

[第7章 9](#_Toc264205537)

[7.1 第一节 9](#_Toc264205538)

[7.1.1 9](#_Toc264205539)

[7.2 本章小结 9](#_Toc264205540)

[7.2.1 9](#_Toc264205541)

[第8章 10](#_Toc264205542)

[8.1 第一节 10](#_Toc264205543)

[8.1.1 10](#_Toc264205544)

[8.2 本章小结 10](#_Toc264205545)

[8.2.1 10](#_Toc264205546)

[参考文献 11](#_Toc264205547)

[作者简历 13](#_Toc264205548)

[致谢 14](#_Toc264205549)

图目录

[图 1.1简单的语音信号多描述编解码过程 2](#_Toc164669029)

[图 3.1 流程图 2](#_Toc164669030)

表目录

[表 2.1简单的多描述分配表 2](#_Toc164669160)

# 绪论

## 课题背景

一般绪论中要把自己所做的工作放进去。包括选题背景，必要性，提出问题等。

绪论也要注意规范化。针对现状，有什么问题，本人想做什么？动机，目标说清楚。注意分清哪些是别人做的工作，哪些是自己做的工作。

### 

每个图都应有图说明，图说明包含图编号与图题(即图的名称)；各种图均要求统一编号，图题与图编号空一字距，居中排印在图的下方，图与图说明不能破页。只有1个图也要有图说明。如图2.1，2表示第2章节，1表示本章的图的序号。图中文字、数字应植字[1]。

坐标图纵横标目的量和单位符号应齐全，居中置于纵横坐标的外侧，横坐标的标目自左至右；纵坐标的标目自下而上，右侧纵坐标的标目方式与左侧相同。

照片应标染色方法和放大倍数，必要时应附有表示目的物尺寸大小的标度。图中的量、符号、单位以及缩略词等必须与正文一致，表亦然。图一般随正文，先见文字后见图。

A

B

C

D

图1.1 论文中图的格式要求

## 本章小结

### 

# 

## 第一节

注释可采用脚注或尾注的方式，按照本学科国内外通行的范式，逐一注明本文引用或参考、借用的资料数据出处及他人的研究成果和观点，严禁抄袭剽窃。字体为宋体，

### 

每个表都应有表说明[2]，表说明包含表序与表题，居中排印在表的上方；表序与表题之间空一字距；独表表示也同样要求；表与表说明不能破页。[[1]](#footnote-2)

表中不设“备注”栏，需要说明的事项可排印在表下方，表内用星号“\*”或圈码“①、②”标注在相应内容的右上角。表中的参数应表明量和单位的符号，如表中所有参数的单位相同，可标注在表的右上方或表说明之后；各栏单位不同则应标注在各栏表头内。表中不能用“同上”、“同左”一类词代替具体数字；无某项目则空白；未发现用“...”；结果为零用“0”；同一栏的数字必须按位次上下对齐。

表随正文，先见文字后见表；需要转页的表，应在续表的右上角或左上角注明“续表×”，并应重复排印表头。

图表与上下文之间各空一行。

于是我们得到表2.1中的两个描述[3]。

表2.1 简单的多描述分配表

|  |  |  |
| --- | --- | --- |
| 描述1 | ζR | (1−ζ)R |
| 描述2 | ζR（重复） | (1−ζ)R |

## 本章小结

### 

# 

## 第一节

正文中要用缩写的，第一次出现时必须用全称，题目中不宜出现缩写。

### 

对外统一的出错处理函数

根据错误的代号确定错误种类

根据错误的文件名以及行号确定

错误位置

打印错误信息

清空连接器前申请资源，退出连接器程序

图 3.1 流程图

## 本章小结

### 

# 

## 第一节

### 

数学公式一般另行起排，居中书写，并用阿拉伯数字分章编号。若数学公式前有文字（如"解"、"假定"等），文字空两格写，数学公式仍居中写。数学公式序号按章编排，序号加圆括号，右顶格排。如第1章第1个数学公式序号为"公式（1.1）"。文中引用数学公式时，一般用"见公式（1.1）"或"公式（1.1）"。

## 本章小结

### 

# 

## Rete算法

Rete算法由Charles Forgy博士于1978 - 1979年发表在博士论文，该论文的简化版本于1982年出版（http://citeseer.ist.psu.edu/context/505087/0）。拉丁语“rete”意指“网络”。Rete算法可以分为两部分：规则编译和运行时执行。

编译算法描述如何处理生产内存中的规则以生成有效的识别网络，识别网络是一个有向无环图。在非技术术语中，识别网络用于在数据通过网络传播时对其进行过滤。网络顶部的节点将具有许多匹配，并且当我们沿着网络向下时，匹配将变少。在网络的最底部是终端节点。Forgy博士在其1982年发表的论文中，为Rete算法定义了4个基本节点：根节点，单输入节点，双输入节点和终端节点。

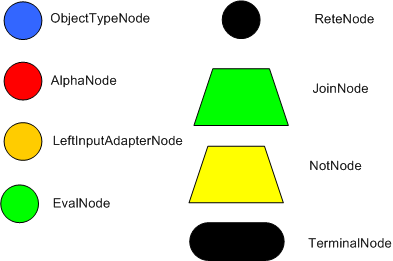


图5.1 Rete Nodes

所有对象从根节点进入网络。对象从根节点从那里传递到ObjectTypeNode。 ObjectTypeNode的目的是尽量减少引擎的工作量,让引擎只做哪些必须要做的事。例如，假设我们有2个对象：Account和Order。如果规则引擎试图针对每个对象进行检测，则会浪费很多时间。为了提高效率，引擎应该将对象传递到与对象类型匹配的节点。最简单的方法是创建一个ObjectTypeNode，并让所有的单输入和双输入节点从它经过。这样，如果应用程序输入一个新的Account，它将不会传递到Order对象对应的节点。在Drools中，通过在对象的类HashMap中检索有效的ObjectTypesNodes的列表来确认对象类型;如果不存在有效的列表，那么它将扫描所有的ObjectTypeNode查找它在列表中缓存的有效匹配项。这使得instanceof匹配的任何类型Drools也都能够匹配。

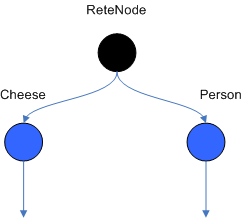


图5.2 ObjectTypeNodes

ObjectTypeNodes的下游节点可以是AlphaNodes，LeftInputAdapterNodes和BetaNodes。AlphaNodes用于字面条件判断。尽管在1982年的文章只涉及等于条件的判断，但许多RETE算法的实现版本都支持其他操作。例如，Account.name ==“Trout”是一个字面条件。 当规则对单个对象类型有多个字面条件判断时，可以使用连接符将他们连接在一起。这意味着如果一个应用程序要判断一个Account对象，它必须首先满足第一个字面条件，然后才能继续下一个AlphaNode。 在Forgy博士的论文中，他将这些称为IntraElement条件。 下图显示了Cheese的AlphaNode组合（name ==“cheddar”，strength ==“strong”）：

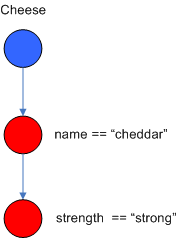


图5.3 AlphaNodes

Drools扩展了Rete算法，通过使用哈希优化从ObjectTypeNode到AlphaNode的传播。每当一个AlphaNode被添加到一个ObjectTypeNode中时，它将字面值作为一个键添加到HashMap中，将AlphaNode作为值。当新实例进入ObjectType节点时，它可以从HashMap检索正确的AlphaNode，而不是传播到每个AlphaNode，从而避免不必要的字面检查。

有两个输入的节点，JoinNode和NotNode，两者都是BetaNodes的类型。 BetaNodes用来比较2个对象及其字段。对象类型可以是相同或不同的。按照惯例，我们将两个输入称为左输入和右输入。 BetaNode的左侧输入通常是对象的列表;在Drools这是一个元组。右侧的输入是单个对象。两个节点都可以用于实现“exists”检查。 BetaNodes也有内存。左输入称为Beta内存，并保存所有传入的元组。右输入称为Alpha存储器，并保存所有传入的对象。 Drools通过对BetaNodes执行索引来扩展Rete。例如，如果我们知道BetaNode正在对字符串字段执行检查，当每个对象进入时，我们可以对该String值进行哈希查找。这意味着当事实从对面进入，我们会做一个查找返回潜在有效的候选人，而不是迭代所有的事实，以找到有效的连接。在任何时候，找到一个有效的连接，元组连接着对象;称为部分匹配;然后传播到下一个节点。

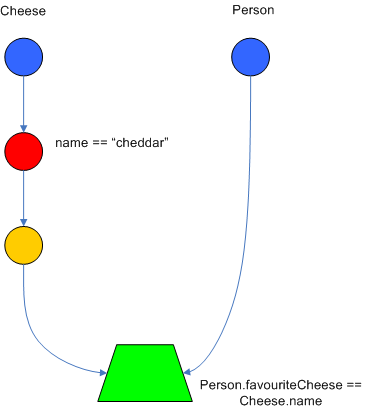


图5.4 JoinNode

要启用第一个对象，如上图中的Cheese对象。要进入网络，我们使用一个LeftInputNodeAdapter - 这需要一个对象作为输入，并传播单个对象元组。

终端节点用于指示具有匹配其所有条件的单个规则; 在这一点上我们说规则有一个完全匹配。 具有“或”条件连接的规则导致每个可能的逻辑分支的子规则生成; 因此一个规则可以具有多个终端节点。

Drools还可以执行节点共享。许多规则重复相同的模式，并且节点共享允许我们重用这些模式，使得它们不必为每个单个实例重新求值。以下示例两个规则共享第一个模式：

rule

when

Cheese( $cheddar : name == "cheddar" )

$person : Person( favouriteCheese == $cheddar )

then

System.out.println( $person.getName() + " likes cheddar" );

end

rule

when

Cheese( $cheddar : name == "cheddar" )

$person : Person( favouriteCheese != $cheddar )

then

System.out.println( $person.getName() + " does not like cheddar" );

end

下图中你可以看到，编译出的Rete网络显示alpha节点是共享的，但beta节点没有共享。 每个beta节点都有自己的TerminalNode。 但如果第二个模式是相同的，那它也会被共享。

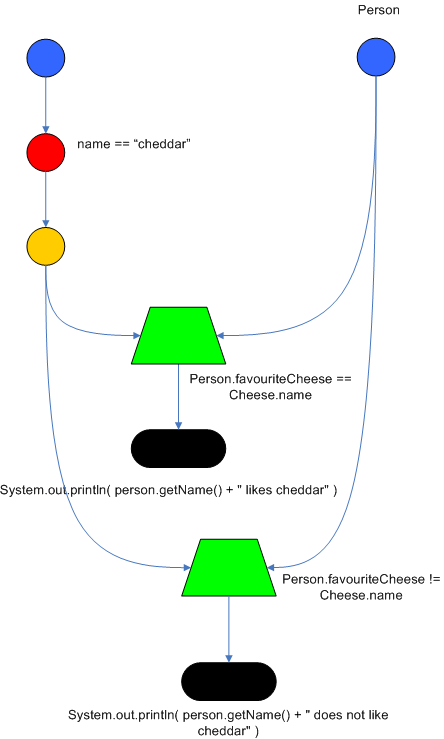


图5.5 Node Sharing

## PHREAK算法

Drools在6.0版本中引入了一种新的算法，称为PHREAK，该算法试图解决RETE算法的一些核心问题。PHREAK算法并不是对RETE算法的重写，该算法包含ReteOO的所有现有代码及其所有增强功能。虽然PHREAK算法是RETE算法的演进，但它不再被归类为RETE实现。在不考虑优化的情况下，由RETE算法演进而来算法一般都有两个关键的RETE特征。第一个特征是完全面向数据，可以理解为算法所有的工作都是在做插入，更新或删除操作;第二个特征是算法致力于快速地生成所有规则的所有部分匹配。相比之下，PHREAK算法并不拥有这两个典型的特征，PHREAK是面向目标的算法，它会推迟部分匹配。

RETE这种致力于快速地生成所有规则的所有部分匹配的做法可能会导致大系统中出现大量的不必要的匹配工作，这些不必要的匹配工作最终并不会导致规则的触发，对系统来说是无用功。

PHREAK受到了许多算法的启发;包括（但不限于）LEAPS，RETE / UL和Collection-Oriented Match。 PHREAK包含了在ReteOO部分中列出的所有增强功能。此外，它添加了以下一组增强功能，在以下段落中将对其进行更详细的说明。

三层上下文存储：节点，段和规则存储。

基于规则，段和节点的链接。

规则懒评估。

规则评估隔离。

面向集合传播。

基于堆栈的评估，并且提供暂停和恢复的功能。

当PHREAK引擎启动时，所有规则都没有被链接，此时不会进行规则评估。在进入到Beta网络之前，插入、更新和删除操作在队列里排队。基于接下来最可能被触发的规则去选择下一条规则来进行评估;这延迟了其他规则的评估和触发。只有当一个规则的所有输入都准备好了之后，规则才被认定为是可链接的。此时一个代表规则的目标被创建，按照定义好的优先级别放入到优先级队列。每个队列与一个AgendaGroup相关联。只有活动的AgendaGroup才会检查其队列，弹出具有最高优先级的规则的目标并提交它进行评估。所以工作从插入，更新，删除阶段转移到fireAllRules阶段。仅评估创建了目标的规则，其他潜在规则的评估将被延迟。虽然评估的是单独的规则，但是仍然通过分割的过程实现节点共享，这将在后面解释。

在RETE算法中的每一次成功的链接接都会得到一个元组（或令牌或部分匹配），它们会被传递到子节点。为此，RETE被表征为面向元组的算法。对于它到达的每个子节点，它将尝试与节点另一侧的节点连接，若连接成功尝试将继续向下传递，这是一个递归的过程。对象从Beta网络的根结点进入到Beta网络，在Beta网络中上下左右运动，最终到达叶节点，也就是中断节点。

PHREAK传播传播方式是面向集合传播（Collection-Oriented），而不是面向元组（Tuple-Oriented）。对于要评估的规则，它将访问第一个节点并处理所有排队的插入，更新和删除操作。将结果添加到集合，并将集合传给子节点。在子节点中，同样是处理所有排队的插入，更新和删除造作，将结果添加到同一集合。一旦完成，该集合被给下一个子节点，以此类推，直到到达终端节点。评估的结果是，创建一个单通道，管道类型的结果，它当前被评估规则隔离；创建了批处理结果，其可以为某些结构的规则提供性能优化，例如累积的子网络。在未来，将倾向于能够以多种方式充分利用多核机器多核的特性，提高效率。

链接和解除链接使用基于网络分段的分层位掩码系统。当规则网络被构建时，会为由相同规则集共享的节点创建段。规则本身由段的路径组成，如果没有共享将是单个段。位掩码偏移量被分配给段中的每个节点，另外一个分层位掩码被分配给规则路径中的每个段。当至少有一个输入时，节点的位被设置为on。当一个段中每个节点的位设置为on时，段的位也设置为on。相反，如果段中任一节点的位被设置为off，则段也被设置为off。如果规则路径中的每个段都被设置为on，则说明规则已链接，并创建目标，计划对规则进行评估。相同的位掩码技术也用于跟踪脏节点，段和规则，如果一条已经链接的规则自上次被评估以来被认为是脏的，这条规则允许被调度用于评估。

如果不会导致规则实例的触发，规则将不会评估部分匹配，因为其中一个联接没有数据。在RETE中部分匹配是可能的，即使最后一个连接为空，也会尝试产生所有节点的部分匹配。

虽然增量规则评估始终从根节点开始，但脏位掩码可用来跳过不脏的节点和段。

使用每个节点至少存在的一个数据项是相当基本的启发式。未来的工作将试图进一步推迟连接操作; 使用诸如弧一致性的技术来确定匹配是否将导致规则实例触发。

此外，RETE只有一级存储，即节点内存，PHREAK有3级存储，节点，段和规则存储。 这允许在评估规则期间更多的上下文理解。

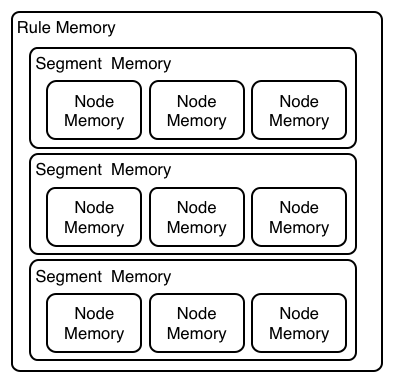


图5.6 PHREAK 3 Layered memory system

示例1显示了一个具有三个模式的单一规则：A，B和C.它形成单个段，节点的位分别是1,2和4。 段的位偏移量为1。

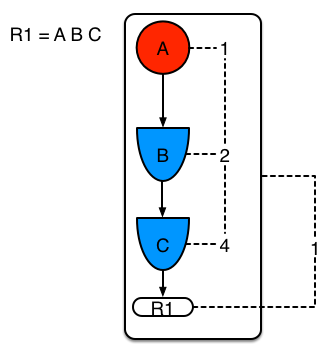


图5.7 Example1: Single rule, no sharing

示例2演示了当另一个规则被添加到共享模式A时发生的情况。模式A被放置在其自己的段中，这导致每个规则两个段。这两个段形成一条路径，用于它们各自的规则。第一个段由两个路径共享。当A链接时，段链接，然后迭代段共享的每个路径，将位1设置为on。如果稍后打开B和C，路径R1的第二个段链接，这将导致R1的位2打开。对于R1，位1和位2均设置为on，此时链接规则并创建目标以调度规则进行后续的评估和触发。

当评估完成的规则，匹配结果成了共享段。每个段都有一个分段内存，用来存储该段的所有插入，更新和删除操作。如果R1被评估，它将处理A并产生一组元组。该算法检测到存在分支，并且将为集合中的插入，更新和删除操作创建对等元组，并将它们添加到R2的分级存储器中。这些元组将与任何现有的分级元组合并，并等待R2最终被评估。

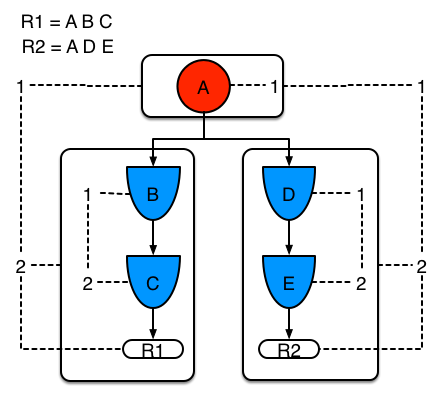
**

图5.8 Example 2: Two rules, with sharing

示例3中有三条规则，规则R4有3个段，R3有3个段，R1有2段。 A和B由R1，R3和R4共享。 而D由R3和R4共享。

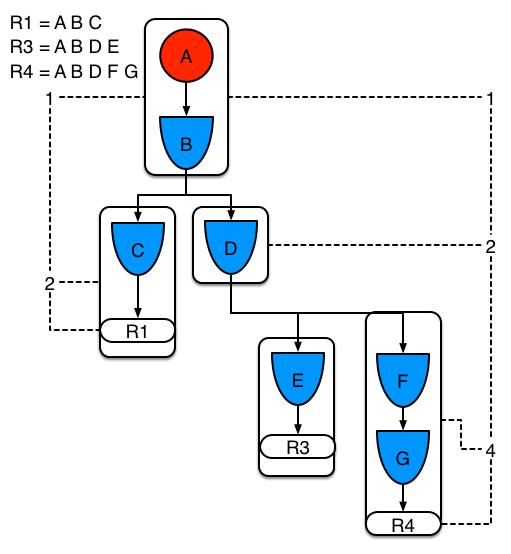


图5.9 Example 3: Three rules, with sharing

所有规则评估都是增量的，并且不会浪费重新计算已经生成的匹配的工作。

评估算法是基于堆栈的，而不是方法递归。可以随时暂停和恢复评估，StackEntry表示正在评估的当前节点。

当规则评估到达子网时，为外部路径段和子网段创建StackEntry。子网段首先被评估，当集合到达子网络路径的末端时，它被合并到其馈入的外部节点的分级列表中。然后恢复先前的StackEntry，其中它可以处理子网络的结果。这具有附加的好处，即所有工作在传播到子节点之前被批量处理;这对于累积节点是更有效的。

相同的堆栈系统可以用于有效的反向链接。当规则评估到达查询节点时，它再次通过将其放置在堆栈上来暂停当前评估。然后评估查询，该查询产生结果集，该结果集保存在用于恢复的StackEntry的存储器位置中以拾取并传播到子节点。如果查询本身调用其他查询，该过程将重复，当前查询被暂停，并为当前查询节点建立新的评估。

关于表演的最后一点。一般来说，一个单一规则不会比使用RETE更快地评估PHREAK。对于给定的规则和相同的数据集（使用根上下文对象来启用和禁用匹配），它们尝试相同数量的匹配并产生相同数量的规则实例，并且大致采用相同的时间。除了具有子网和累积的用例。

然而，PHREAK可以被认为是更宽容的RETE对于写得不好的规则库，并且随着规则的数量和复杂性的增加而更性地降低性能。

RETE也将流失生成在所有连接中没有数据的规则的部分机器;其中PHREAK将避免这一点。

所以这不是说PHREAK比RETE更快，它只是不会放慢与你的系统增长一样多。

AgendaGroups并没有帮助RETE的表现，作为所有规则在任何时候评估，不管组。显着性同样如此。这就是为什么经常使用根上下文对象，以限制匹配尝试。 PHREAK只评估活动的AgendaGroup的规则，并且在该组内将尝试避免评估不导致规则实例触发的规则（通过突出）。

随着PHREAK AgendaGroups和突性现在成为有用的性能工具。根上下文对象不再需要，并且可能对性能产生反作用，因为它们强制刷新和重新创建规则的匹配。

## 本章小结

### 

# 

## 第一节

### 

## 本章小结

### 

# 

## 第一节

### 

## 本章小结

### 

# 

## 第一节

### 

## 本章小结

### 

参考文献

[1]广西壮族自治区林业厅. 广西自然保护区[M].北京:中国林业出版社，1993:35-37.

[2]霍斯尼.谷物科学与工艺学原理[M].李庆龙，译.2版.北京：中国食品出版社，1989: 15-20.

[3]孙玉文. 汉语变调构词研究[D]. 北京：北京大学出版社，2000.

[4]赵耀东. 新时代的工业工程师[M/OL].台北：天下文化出版社, 1998[1998-09-26]. http://www.ie.nthu.edu.tw/info/ie ie.new.htm .

[5]全国信息与文献工作标准化技术委员会出版物格式分委员会. GB/T 12450 —2001 图书书名页 [S]. 北京: 中国标准出版社,2002.

[6]PEEBLES P Z, Jr. Probability, random variable, and random signal principles[M]. 4th ed. New York: McGraw Hill, 2001：100-110.

[7]韩吉人. 论职工教育的特点［G］// 中国职工教育研究 会. 职工教育研究论文集. 北京：人民教育出版社，1985: 90-99.

[8]赵颖力, 曹敏, 王琳.《化工学报》编辑部的人才建设[C] // 第 3 屆中国科技期刊青年编辑学术研讨会论文集. 北京：中国科学技术期刊编辑学会青年工作委员会, 2003：86-88 .

[9]张旭, 张通和, 易钟珍. 采用磁过滤MEVVA 源制类金刚石膜的研究[ J ]. 北京师范大学学报：自然科学版，2002，38(4)：478-481.

[10]周桂莲, 许育彬, 杨智全. 认清市场形势 化解“学报情结”：我国农业学报的现状与发展趋势分析[J]. 编辑学报, 2005, 17(3)：209-211.

[11]萧钰. 出版业信息化迈入快车道[EB/OL] .(2001-12-19) [2002-04-15]. http:∥www.creader.com/news/ 200112190019.htm.

[12]Online Computer Library Center, Inc. History of OCLC［EB/OL］. [2000-01-08]. http:∥www.oclc.org/about/history/default. htm.

[13]Nenad Medvidovic, Richard Taylor. A Classification and Comparison Framework for Software Architecture Description Languages[J] . IEEE Transactions on Software Engineering, 2000, 25(1):70-93

**注意：**

**参考文献的排列按照学位论文中所引用的文献顺序排列，论文中参考文献引用需用上标。**

**文献数量合理，不太少也不滥用，文后列出的参考文献在正文中必须有对应的引用。**

**文献来源正宗权威，是学术文献，出典可查。**

作者简历

教育经历：

×××××××××

×××××××××

工作经历：

×××××××××

×××××××××

攻读学位期间发表的论文和完成的工作简历：

×××××××××

×××××××××

致谢

署名

于浙江大学软件学院

当前日期

1. 数据来源××××× [↑](#footnote-ref-2)