Drools 规则引擎的研究与应用

姜涛
北京邮电大学计算机科学与技术学院，北京 (100876)
E-mail: jiangtao@bupt.cn

摘　要：Drools 是一个使用基于规则的方法实现专家系统的规则引擎，更准确的说属于产生式规则系统。产生式规则系统完全专注于精确表达命题和一阶逻辑的知识表示，不存在含糊不清的定义，它的核心是一个能够处理大量规则和事实的推理引擎。推理引擎将事实、数据与产生式规则进行匹配，以得出结论。本文分析了 Java 规则引擎的工作机制原理及其核心算法——Rete 算法，并对 Drools 规则配置文件的结构与含义做了详细解析，最后介绍了 Drools 规则引擎在商业银行反洗钱大额交易和可疑交易监测中的应用。

关键词： Drools；规则引擎；Rete；反洗钱
中图分类号： TP317

1. 引言

企业管理者对企业级 IT 系统的开发有着如下的要求：(1)为提高效率，管理流程必须自动化，即使现代商业规则异常复杂(2)市场要求业务规则经常变化，IT 系统必须依据业务规则的变化快速、低成本的更新(3)为了快速、低成本的更新，业务人员应能直接管理 IT 系统中的规则，不需要程序开发人员参与。而项目开发人员则碰到了以下问题:(1)程序=算法+数据结构，有些复杂的商业规则很难推导出算法和抽象出数据模型(2)软件工程要求从需求->设计->编码，然而业务规则常常在需求阶段可能还没有明确，在设计和编码后还在变化，业务规则往往嵌在系统各处代码中(3)对程序员来说，系统已经维护、更新困难，更不可能让业务人员管理来。

基于规则的专家系统的出现给开发人员以解决问题的契机。规则引擎是推理擎的一种，起源于基于规则的专家系统。规则引擎能够将业务逻从应用程序代码中分离出来，接受数据输入，解释业务规则，并根据规则做业务决策。规则引擎的核心内容是其所采用的模式匹配算法，Forge 于 1979 年提出了 Rete 算法。Rete 算法牺牲了一部空间来提高算法的时间效率，使该算法有了广泛的应用空间。目前，几乎所有成熟的规则引擎框架的实现都是基于该算法的。但是，绝大多数规则引擎框架都是商业产品，价格非常昂贵。这种情况直到 JsR941，(Java 规范要求，JavaspeeificationRequest)规则引擎标准和开源规则擎的出现才逐渐好转。Drools 是一个基于 JAVA 的开源并且免费的框架，采用 XML 或 DRL 格式的规则描述语言描述业务逻辑，是最近几年才逐渐成熟起来的一个框架，还需要进一步的完善[1]。本文主要研究 Drools 规则引擎的原理及其在实践中的应用。

2. Drools 规则引擎的研究

2.1 Drools 规则引擎的工作原理及运行机制

Drools 是一个使用基于规则的方法实现的专家系统的规则引擎,其系统的核心是一个能够处理大量规则和事实的推理引擎。推理引擎将事实、数据与产生式规则进行匹配，以推出结论[2]。

在 Drools 中规则是二元式结构，由一个左手元（LHS）与一个右手元（RHS）构成，又称为头和尾。规则的 LHS 部分包含条件元素（CE）和数据字段（Column），帮助进行命题编码和一阶逻辑表达。当 Facts 被设置到工作空间（Working Memory）或者在其中被修改时，

规则引擎根据 LHS 的表达式，匹配这些 Facts，当一个规则的所有 LHS 被匹配并且为真值时，规则被激活。当规则被激活后将放入 Agent 中等待执行，被执行的部分是规则的 RHS，也称为推论。LHS 和 RHS 的关系类似于：

If (LHS)
THEN
RHS

规则存在 Production Memory（规则库）中，推理引擎要匹配的 facts（事实）存在 Working Memory（工作内存）中。事实被插入到工作内存中后，可能被修改或删除。一个有大量规则和事实的系统可能会很多规则被满足，这些规则被称为具有冲突性。Agenda 通过（冲突决策策略）管理这些冲突规则的执行顺序。如图 1

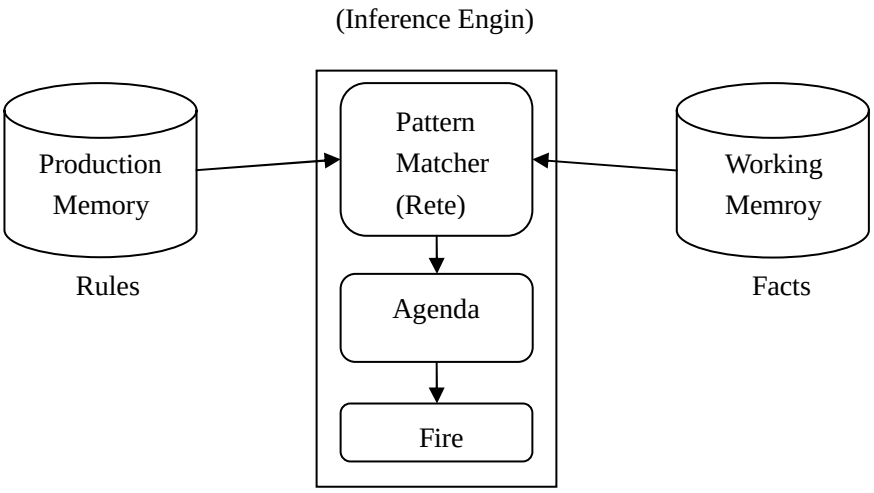

图 1 规则引擎原理结构

推理机的推理步骤如下:

1)将初始事实数据载入工作内存;

2)使用模式匹配器比较规则库中的规则和工作内存中的事实数据;

3)如果执行规则时存在冲突(Conflict)，即同时激活了多个规则，则将冲突的规则放入冲突集合;

4)解决冲突，将激活的规则按顺序放入议程安排器;

5)使用执行引擎执行议程安排器中的规则;

6)重复步骤 2)至 5)，直到议程安排器中的所有规则被执行完毕。

推理机使用的用于模式匹配的算法有很多,Drool 实现了 Rete 和 Leaps 算法；Leaps 是一个新的算法，是试验性质的。下面主要研究一下 Rete 算法。

2.2 Rete 算法

Rete 算法是一个用来实现产生式规则系统的高效模式匹配算法,Rete 是拉丁语中的网络的意思，代表网络的概念。Rete 算法可以被分为两个部分：规则编译(Rule Compilation）和运行时执行（Runtime Execution）。

Rete 算法的规则编译的结果是生成规则集对应的 Rete 网络,网络通过数据在网络的传播过程中来过滤数据。在网络的顶端将会有很多匹配的数据。当顺着网络向下走，匹配的数据将会越来越少,在网络的最底部是终端节点 terminal Node。

Rete 网络的节点可以分为四类：根节点（root）、1-input 结点（也称为 alpha 结点）、2-input

结点（也称为 beta 结点）、终端节点（Terminal Node）。其中 1-input 结点组成了 Alpha 网络，2-input 结点组成了 Beta 网络。

每个 alpha 结点和 beta 结点都有一个存储区。其中 1-input 结点有 alpha 存储区和一个输入口；2-input 结点有 left 存储区和 right 存储区和左右两个输入口，其中 left 存储区是 beta 存储区，right 存储区是 alpha 存储区。存储区储存的最小单位是工作存储区元素（Working Memory Element，简称 WME），WME 是为事实建立的元素，是用于和非根结点代表的模式进行匹配的元素。Token 是 WME 的列表，包含有多个 WME，用于 2-input 结点的左侧输入。事实可以做为 2-input 结点的右侧输入，也可以做为 1-input 结点的输入。

每个非根结点都代表着产生式左部的一个模式，从根结点到终结点的路径表示产生式的左部，如图 2

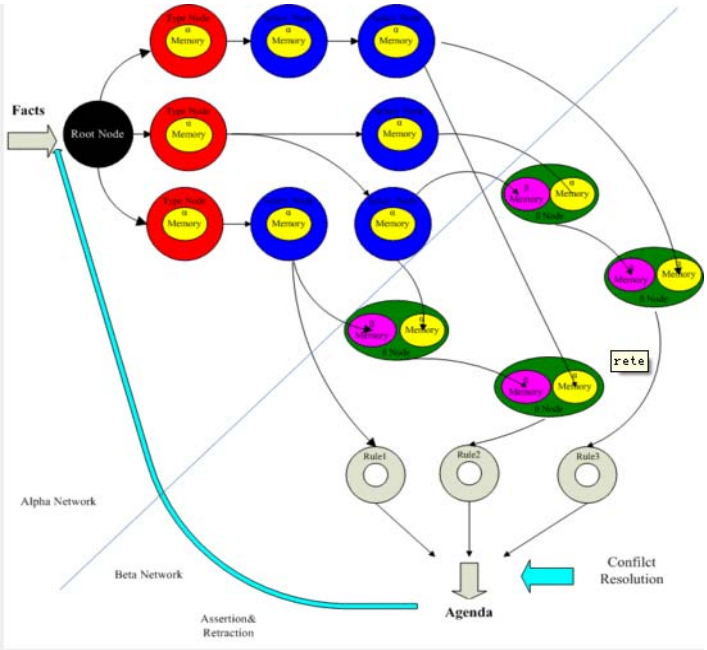

图 2 Rete 网络

2.2.1 规则编译过程[3]

（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；

2.2.2 运行时执行

推理引擎在进行模式匹配时，先对事实进行断言，为每一个事实建立 WME，然后将 WME 从 RETE 鉴别网络的根结点开始匹配，因为 WME 传递到的结点类型不同采取的算法也不同，下面对 alpha 结点和 beta 结点处理 WME 的不同情况分开讨论。

（1）如果 WME 的类型和根节点的后继结点 TypeNode（alpha 结点的一种）所指定的类型相同，则会将该事实保存在该 TypeNode 结点对应的 alpha 存储区中，该 WME 被传到后继结点继续匹配，否则会放弃该 WME 的后续匹配；

（2)如果 WME 被传递到 alpha 结点，则会检测 WME 是否和该结点对应的模式相匹配，若匹配，则会将该事实保存在该 alpha 结点对应的存储区中，该 WME 被传递到后继结点继续匹配，否则会放弃该 WME 的后续匹配；

（3）如果 WME 被传递到 beta 结点的右端，则会加入到该 beta 结点的 right 存储区，并和 left 存储区中的 Token 进行匹配（匹配动作根据 beta 结点的类型进行，例如：join，projection，selection），匹配成功，则会将该 WME 加入到 Token 中，然后将 Token 传递到下一个结点，否则会放弃该 WME 的后续匹配；

（4）如果 Token 被传递到 beta 结点的左端，则会加入到该 beta 结点的 left 存储区，并和 right 存储区中的 WME 进行匹配（匹配动作根据 beta 结点的类型进行，例如：join，projection，selection），匹配成功，则该 Token 会封装匹配到的 WME 形成新的 Token，传递到下一个结点，否则会放弃该 Token 的后续匹配；

（5）如果 WME 被传递到 beta 结点的左端，将 WME 封装成仅有一个 WME 元素的 WME 列表做为 Token，然后按照（4）所示的方法进行匹配；

（6）如果 Token 传递到终结点，则和该根结点对应的规则被激活，建立相应的 Activation，并存储到 Agenda 当中，等待激发。

（7）如果 WME 被传递到终结点，将 WME 封装成仅有一个 WME 元素的 WME 列表做为 Token，然后按照（6）所示的方法进行匹配；

2.2.3 Rete 算法优点

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在 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。

2.3 Drools 规则语言 DRL

JsR941 没有提供规则和动作该如何定义以及该用什么语言定义规则，也没有为规则引擎如何读取和评价规则提供技术性指导。JsR941 规范将上述问题留给了开发规则引擎的厂商。规则语言是规则引擎应用程序的重要组成部分，所有的业务规则都必须用某种语言定义并且存储于规则执行集中，从而规则引擎可以装载和处理他们。由于没有关于规则如何定义

的公用规范，市场上大多数流行的规则引擎都有其自己的规则语言， DRools 则采用系统专用的规则语言 DRL 来描述规则。规则文件通常是以 drl 扩展名结尾。在一个 drl 文件中可以包含多个规则，函数等等。但是你也可以将规则分开到多个规则文件中，图 3 是一个最基本的 DRL 文件示例:

```
package com.sample

import com.sample.DroolsTest.Message;

rule "GoodBye"
    when
        m : Message( status == Message.GOODBYE, message : message )
    then
        System.out.println( message );
        m.setMessage( message );
end
```

图 3 规则文件示例

上述文件中“GoodBye”代表了规则的名称，这个规则在工作内存中查找 com.sample. DroolsTest. Message 对象的实例。如果发现一个 Message 实例，并且满足其属性 status 的值等于 Message.GOODBYE 对应的值，则将 Message 对象的 message 属性绑定到本地变量。然后规则触发结果，打印 message 消息，同时为 Message 的对象实例的 message 属性赋值。

3. Drools 规则引擎在反洗钱监控系统中应用

随着经济的发展，洗钱活动日益猖獗，反洗钱已成为当今国际社会的一个焦点问题。为预防利用金融机构进行洗钱活动，规范金融机构大额交易和可疑交易报告行为，中国人民银行出台了新的《金融机构大额交易和可疑交易报告管理办法》，进一步明确了商业银行需要上报的大额交易和可疑交易的规则。尽管有了反洗钱法，但对于金融机构来说，反洗钱的开展与执行仍然有很大的困难，具体表现在：

数据量大，处理效率低。金融机构每日交易量是非常大，如果通过人工方式或传统的数据库查询方式筛选，几乎是不可能的。

业务规则不明确，监测困难。反洗钱法只规定了大额和可疑交易规则，如何对这些规则进行有效的分析量化，如何从大量的交易中有效地区分出大额和可疑交易，这是要解决的主要问题。

本章主要结合 Drools 规则引擎的技术特点，介绍其在反洗钱监控系统中的应用。

3.1 针对规则要求，建立规则模型，量化指标

以人民银行可疑规则 ”长期闲置的账户原因不明地突然启用或者平常资金流量小的账户突然有异常资金流入，且短期内出现大量资金收付”为例说明监测方法。

首先，针对对公客户我们量化了如下指标：

1）短期内对公本币账户资金转入转出量比；

2）短期内对公本币账户资金转入转出次数比；

3）短期内对公本币账户资金转入金额。

其次，建立以下规则判断模型 ：

IF

（（默认为 90%）<= 短期内对公本币账户资金转入转出量比**(GZ30048)** <=（默认 110

%))

AND

（短期内对公本币账户资金转入转出次数比(**GZ30052**)>=（参数，默认 3））

AND

(短期内对公本币账户资金转入金额(**GZ10108**))>=（默认 200 万）

THEN 预警

针对规则模型建立规则文件如图 4：

```
rule "KY0101"
  when
    #conditions
    #get IndicFact from RuleFact.IndicHash
    #get alertHash from RuleFact
      $T : RuleFact(alerthash : alertHash , indichash : indicHash);
      eval((indichash.containsKey("GZ30048"))
       && (indichash.containsKey("GZ30052"))
       && (indichash.containsKey("GZ10108"))
        && (indichash.containsKey("JC10125")));
      eval( ( (IndicFact)CommonUtil.getObj(indichash,"GZ30048")).getIndicval() >= ((IndicFact)CommonUtil.getObj(i
       && ( (IndicFact)CommonUtil.getObj(indichash,"GZ30048")).getIndicval() <= ((IndicFact)CommonUtil.getObj(ind
       && ( (IndicFact)CommonUtil.getObj(indichash,"GZ30052")).getIndicval() >= ((IndicFact)CommonUtil.getObj(ind
       && ( (IndicFact)CommonUtil.getObj(indichash,"GZ10108")).getIndicval() >= ((IndicFact)CommonUtil.getObj(ind

  then
  #actions
  #set true(alert) into the alerthash
    alerthash.put("KY0101",new Integer(1));

end
```

图 4 规则文件

3.2 指标事实与规则匹配得出预警

借助于 Drools 规则引擎的推理机制，将计算出的指标值与规则库中的规则文件进行模式匹配，若客户的交易行为满足规则，得出可疑预警信息。

4. 结论

本文结合 Drools 规则引擎的工作原理和机制，介绍了它在商业银行反洗钱监控系统中的应用。在项目使用中规则引擎具有如下优点：

（1）逻辑与数据分离

数据保存在系统对象中，逻辑保存在规则中。这样做的结果是，将来逻辑发生改变时更容易被维护，因为逻辑保存在规则中。通过将逻辑集中在一个或数个清晰的规则文件中，取代了之前分散在代码中的局面，使银行业务人员很容易对规则进行维护。

（2）逻辑与数据分离匹配速度快，效率高

Drools 提供了对系统数据对象非常有效率的匹配，满足了商业银行对海量交易数据处理效率的要求。

（3）知识集中化

使用规则，将建立一个可执行的规则库。这意味着规则库代表着现实中的商业政策唯一对应，理想情况下可读高的规则还可以被当作文档使用。

参考文献

[1] 曹永亮. 基于 JAVA 规则引擎的动态数据清洗研究与设计[D]. 武汉:武汉理工大学, 2008.5
[2] JBoss Rules Home. http://www.jboss.org/drools/documentation.html, 2008
[3] 规则引擎研究——Rete 算法. http://blog.sina.com.cn/s/blog_4a7a7aa30100089g.html, 2007.3

Research and Application of Drools Rule Engine

Jiang Tao
Department of Computer Science and Technology, Beijing University of Post and Telecommunication, Beijing (100876)

Abstract

Drools is a Rule Engine that uses the Rule Based approached to implement an Expert System and is more correctly classified as a Production Rule System. A Production Rule System is turing complete with a focus on knowledge representation to expression propositional and first order logic in a concise, non ambigious and declarative manner. The brain of a Production Rules System is an Inference Engine that is able to scale to a large number of rules and facts. The Inference Engine matches facts, the data, against Production Rules, to infer conclusions which result in actions. This paper analysis the basic principles and working mechanism of Java Rule Engine and it'score algorithm--Rete algorithm,also introduced the structure and meanings of its rule configuration file.At the last,this paper present the application of Drools rule Engine in anti money laundering system of commercial bank.

Keywords: Drools; Rule Engine; Rete; Anti-money- Laundering