Research of the Rule Engine based on XML

Zhao Ni[1, a], Lifang Bai[2, b]

[1] Zhengzhou Institute of Information Science and Technology, Henan, China, 450000, China

[2] Zhengzhou Institute of Information Science and Technology, Henan, China, 450000, China

[a]email: biluo_cc@163.com, [b]email:493325178@qq.com

Keywords: Pattern matching algorithm; XML; Rule engine; Rete network

Abstract. As Studying of the current rule engine implementation mechanism, An XML format rules is put forward for the poor ability in the predicate expansion and the rule reuse of the traditional rule description, which can describe of the detail interface information called by entity class in rules, and the improved rete network corresponding to the rule description is proposed, forming a new pattern matching algorithm based on XML format rules.

Introduction

Rules engine originated from production system. It can be regarded as a component of the application, which realizes the separation of the business rules or logic in application from the program code, and writes business rules with predefined semantic module [1]. The application field of rule engine is very extensive, including banking, telecommunications billing, game software, business management, etc. At present, an open source business rules engine Drools (JBoss) is the most representative of the rule engine [2].

Pattern matching algorithm is the core algorithm in the rule engine. This paper focuses on Rete pattern matching algorithm, which is the basis for the shell of many production systems, including the CLIPS (C language integrated production system), JESS, Drools and Soar [3]. Rete algorithm can share nodes to reduce redundant space; the nodes store part matching results to avoid re-evaluating the facts because of a change in the working memory; and the mechanism of the fact efficiently removing from the working memory to achieve the fast rule matching. The pattern matching algorithm directly affects the rate of rule matching in the rule engine, so the research of pattern matching algorithm is very important.

Currently, the Rete-OO algorithm is widely used Rete algorithm. It can describe logic rules in rule description file with multiple programming languages including groovy, Java, python, and etc. [4], which is the object-oriented algorithm used in Drools, but the algorithm can only use the exists logic operation in programming language to describe logic rules, leading to the poor ability in extended predicate of rule engine. Literature [5] added the uncertainty predicates into Rete-OO algorithm to expand its logic expression ability of rules, however its implement is very complex in the existing rules engine. Literature [6] described the use of the rules decompose ways to improve the efficiency of parsing rules without further research on the rule description method. TREAT algorithm improved method for processing variable binding, but its rule matching efficiency is mainly affected by the environment [7]. JESS is the CLIPS realization in the object-oriented language--Java, but its rule description do not support Java programming language, increasing the difficulty in written rules [8]. To solve these problems, this paper proposes a matching algorithm based on the rule description method of XML, which can extend descriptive ability of predicates, realize the flexible rule combination, and improve reusability of rules.

Rete Pattern Algorithm in Rule Engine

Working Mechanism of Rule Engine

Rule engine uses rule description files and interface of rule engine to pull out rules from the application program, to achieve a loosely coupled architecture of business rules and program code.

The structural components of rule engine are shown in figure 1.

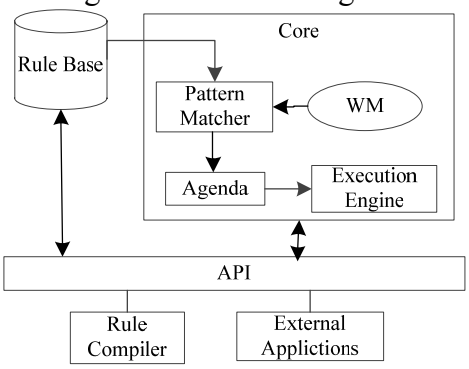

Fig.1. The structural components of rule engine.

Rule Base stores the specific format rules (.drl format in Drools), enabling more efficient use of rules in rule engine. In Rete algorithm, one rule is separated to two parts, an antecedent (LHS, left-hand side) and a rear (RHS, right-hand side), composed of the IF LHS THEN RHS structure. LHS is the condition of triggering the rules, and a separate condition calls a pattern. RHS is the behavior after rule triggering.

The core of rule engine is divided into four components:

1) Working memory (WM), stores the external data (fact), and saves the current state of the system;

2) Pattern matcher, uses facts stored in WM match to the LHS part of rules in the rule base, and the efficiency of pattern matching rules determines the performance of rule engine;

3) Agenda, sorts successfully matching rules in the priority order to prevent executing conflicts;

4) Execution engine, executes the RHS parts of rules in order in the agenda.

The API of rule engine is a communication interface of rules engine and rule compiler or external applications.

The workflow of rule engine can be briefly described as follows: the facts is loaded in WM firstly, the pattern matcher will match the fact data in WM to the LHS part of rules in the rule base. If multiple rules are activated resulting conflicts in executing rules, the conflicting rules will be put in the agenda, and priority ordered by conflict resolution mechanism, until all rules in agenda are executed.

Rete Network and Token

The first step in the Rete algorithm is to establish the Rete network structure. The LHS part of rules are compiled into a form of pattern recognition network similar to the data stream network, and each rule in the rule base corresponds to one or more patterns, such as a simple Rete network structure in figure 2. Token reflects changes in the WM, which can be viewed as a WME (working memory element) list. Token enters into the Rete network from the root node, and transfers between different nodes in different layers of the network. WME contains tags and data elements, in which "+" tab means adding the corresponding data elements to the WM, and "-" tab means removing the corresponding data elements from the WM.

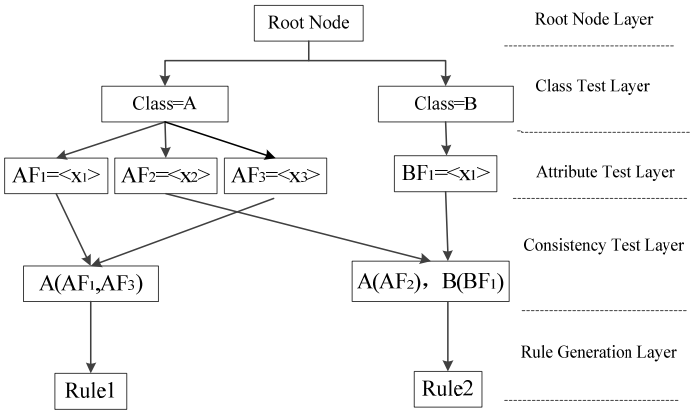

Fig.2. The Rete network structure.

A Rete network structure consists of the following five layers:

1) Root node layer, root node is the entry of token to the Rete network;

2) Class test layer, token is copied into multiple copies and passed to the object type nodes which will check the type of the copies. The copies meeting the node type will be saved to the storage section in the object type nodes, and the other copies that do not meet the node type will be discarded;

3) Attribute test layer, it is composed of the alpha nodes (1-input node) which will test the token matching a pattern, if the match is successful, then the Token can be saved to the storage section in this node;

4) Consistency test layer, it is composed of the beta nodes (2-input node) which test the tokens of two patterns meeting the consistency constraint, if the test is passed, then two token will be combined into one token and saved into the storage section in this node;

5) Rule generation layer, the RHS part of rules are stored in the terminal nodes;

When the token enters into the root node, after class test, attribute test, and consistency test, if existing token flows to the terminal node, then the rule corresponding to this terminal node will be activated, and the corresponding RHS part will be put into the agenda and wait for execution.

Existing Problems

There are two issues mainly considered in pattern matching algorithm executing in rule engine: the first thing is the method of rule description in the rule base; and the second is the Rete network structure of pattern matching. The traditional rule engine has the following problems in rule description:

1) The matching test capability only exists in the attribute test layer, the rule description method cannot directly reflect the dependency between the classes (e.g. CLIPS) [9];

2) The description of logic predicates is limited to the logical operation of programming language itself, which cannot realize the flexible combination of rules, and it limits the scalability of predicates in rules (such as Drools);

3) The rule description files does not support embedding programming language codes (such as JESS);

4) The particle size of rules cannot be effectively controlled and the reusability of rules is relatively poor.

For the traditional rule engine's construct in the Rete network structure, there are two deficiencies as follows:

1) It is very complicated to construct the rete network structure by large number of rules;

2) Since the dependencies between classes cannot reflect until the consistency test layer, it will definitely increase work in the beta node work, especially in the multi-input case, the matching rate will be significantly reduced when there are interoperate between the different classes.

Rule Description Based on XML

In view of the existing rule description problems in executing the Rete algorithm, the paper proposes a rule based on XML format. Currently, XML is the most common self-description text format [10]. The rule description files based on XML can be embedded with all the interfaces of involved entity classes, on the one hand, it achieve the loosely coupled architecture of rule description files and the entity classes, and other hand it can realize the dynamic loading of classes in the application of oriented-object programming language Java.

Rule Description Format

Under normal circumstances, the entity classes involved in rule matching process usually remain unchanged, while the rules reasoning process often needs to be changed. For the characteristics in rule matching, XML description can achieve two important functions: first, when the rules have not changed the methods definition of reasoning involved entity classes remain unchanged, if the logic operations in method are changed, it can simply reason again with recompiling the Java source files; second, when the definition of an entity class remains unchanged, if the rule description files are modified, it can matching reason again without recompiling. Table 1 shows the main subjects, corresponding attributes and significance contained in rule description of XML format.

Tab.1. The main subjects, corresponding attributes and significance in XML rule description .

Subject	Attribute	Significance
rule	name	Rule name
	priority	Rule's priority
precondition	name	Rule's precondition (existing rule name)
	negated	"+" / "-" pattern（true/false）
fact/andfact/orfact		Support logic combination of "and" and "or"
comparator		Logic predicates (such as >、<、=、before、after、between, etc.)
Leftvalue/rightvalue		Left/Right part of the logic predicates in a condition
expression		Variable, numerical, etc.
action		RHS part of activated rules
	classname	The class name in executing RHS part of rules
	methodname	The name of called method
	varname	Variable name
	static	Whether it is static method
argument		The corresponding argument passed to method functions by expression

Advantages of XML Rule Description

The XML rule description is consistent with the basic structure of IF LHS THEN RHS, its main advantages are as follows:

1) The rule description files support embedding detailed interface information of entity classes, and do not support specific logic implementation;

2) The concept of "supporting positive and negative pattern of precondition" is introduced to reasonably control the particle size of rules;

3) It supports the logic combination of "andfact" and "orfact", which can make more flexible structure of logic rules;

4) It supports the use of the comparators, that can improve description ability of the logic predicates in rules;

5) XML format text has a variety of sophisticated parsers which can be conveniently used in rule resolution.

Improved Rete Pattern Algorithm

Aimed at the shortcomings in traditional Rete network structure of rules engine, the paper introduces an improved object-oriented structure of the Rete network.

Improved Rete Network

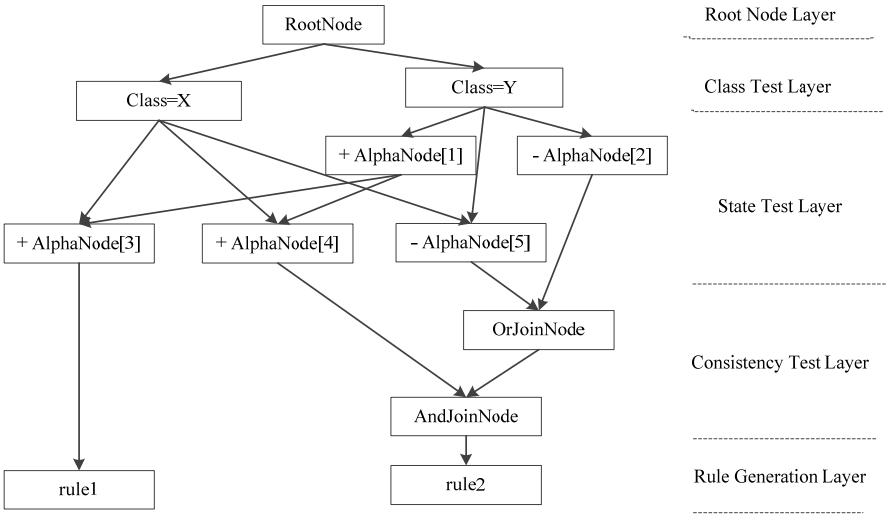

Fig.3. The improved Rete network structure.

Figure 3 shows an improved Rete network structure, which equally divided into 5 levels, mainly contains the following two improvements:

1) The original layer becomes the state test layer, and at this layer the alpha nodes contains both 1-input nodes and 2-input nodes, which is the only node to process the pattern matching. The alpha nodes includes "+" and "-" mode corresponding to the "true" and "false" of " negated" attribute of the precondition in XML rules description. When the token matches the node's pattern, it will be passed to the child nodes, while preserving token to the matching storage section of node, otherwise to the non-matching storage section.

2) The join node connects two or more input nodes in the consistency test layer, and it consists of the "and" and "or" predicate modes. The test is to check whether the token passed from one side can combine with the token passed from the other side, if the combination is successful, the token will be passed to its child nodes.

Token Transmission Optimization

The optimization process of the traditional token includes two aspects as following:

1) The traditional token contains "-" tab which can support revocation operation in the Rete algorithm, but the price is very expensive. This paper use the "asymmetric delete" approach to optimization, namely deleting compound token containing the "-" tab at the 2-input node, rather than passing the compound token to the nodes in the next layer.

2) Adding the "Clear" tab to the token, when a node of the network receives a token with the "Clear" tab, the node will clear all of the contents of its storage section, and when it receives the token with "-" tab, it only clears all tokens involved with this token so as to quickly release the contents of the storage section for the next reasoning.

Advantages of Improved Network

The improved Rete network structure has the following advantages:

1) The alpha nodes contain the 2-input nodes, which can reduce the burden on the under join nodes, and conducive to multiple-input parallel reasoning;

2) The join nodes need not be responsible for other complex logic calculation but the consistency test;

3) The network has two join nodes, "OrJoinNode" and "AndJoinNode", that can support "or" and "And" logic in rules description realizing the flexible combination.

Example Analysis

The example validation and analysis based on the improved algorithm above, if there exists rule1, rule 2:

1) The LHS part of rule1 and rule 2 both includes class X and class Y;

2) Rule1 is the "-" precondition of rule2, is that when rule1 do not be satisfied executing rule2.

The improved Rete network structure shown in Figure 3 is constructed by Rule 1 and Rule 2. The test of class X and class Y is in the class test layer, and Table 2 shows the reasoning process in the state test layer.

Tab.2. Pattern matching operation in alpha nodes.

Alpha Node	Pattern Matching Operation
Alpha Node[3]	test.X. isMethod0(test.Y.getMethod1()) Equals true
Alpha Node[5]	test.X. isMethod0(test.Y.getMethod1()) Not Equals true
Alpha Node[4]	test.X. isMethod0(test.Y.getMethod2()) Equals true
Alpha Node[1]	test.Y. isMethod1() Equals false
AlphaNode[2]	test.Y. isMethod1() Not Equals false

Conclusion

This paper achieves the loosely coupled architecture of description file and specific logic called by entity class, based on the rule description method of XML, which describes the interface information of calling method in rule description file, rather than directly embedding the concrete realization logic code. Through the analysis and discussion of the mechanism and method of the traditional Rete algorithm and the improved Rete algorithm in rule engine, it can be seen that the improved Rete pattern matching algorithm has obvious advantages in the flexibility of logic rule description, granularity control of rules, rules reusability, and etc.

References

[1] Yin Z, Li X, Wu C, et al. Research of Rule Engines in Web Service Environment[M]. Frontiers in Computer Education. Springer Berlin Heidelberg, 2012:659-665

[2] Proctor M. Relational Declarative Programming with JBoss Drools[C]. Symbolic and Numeric Algorithms for Scientific Computing, 2007. SYNASC. International Symposium on. IEEE, 2007:5-5.

[3] Xiaodong Gu, Yang Gao. Rete algorithm: Current Status and Challenges[J]. Computer Science, 2012, 39(11):8-12.

[4] Wulff N, Sottara D. Fuzzy Reasoning with a Rete-OO Rule Engine[M]. Rule Interchange and Applications. Springer Berlin Heidelberg, 2009:337-344.

[5] Sottara D, Mello P, Proctor M. Adding Uncertainty to a Rete-OO Inference Engine[M]. Rule Representation, Interchange and Reasoning on the Web. Springer Berlin Heidelberg, 2008:104-118.

[6] Liu D, Gu T, Xue J P. Rule Engine based on improvement Rete algorithm[C]. Apperceiving Computing and Intelligence Analysis (ICACIA), 2010 International Conference on. IEEE, 2010:346 - 349.

[7] Weihui Wang, Guohua Geng, Mingquan Zhou. Review of Pattern Matching Algorithm in Rules Software System[J]. Journal of Chinese Computer Systems, 2012(5):913-920.

[8] Moskal J, Matheus C J. Detection of Suspicious Activity Using Different Rule Engines -- Comparison of BaseVISor, Jena and Jess Rule Engines[C]. Proceedings of the International

Symposium on Rule Representation, Interchange and Reasoning on the Web. Springer-Verlag, 2008:73-80.

[9] Ding Y, Wang Q, Huang J. The Performance Optimization of CLIPS[C]. Hybrid Intelligent Systems, 2009. HIS '09. Ninth International Conference on. IEEE, 2009:417 - 421.

[10] Kopecký J, Vitvar T, Bournez C, et al. Semantic Annotations for WSDL and XML Schema[J]. IEEE Internet Computing, 2007, 11(6):60-67.