Improvement and Realization of Rete Algorithm for the Dynamic Evolution of Software System

DanFeng WU, GuangPing ZENG, JingYing YAN

School of Computer & Communication Engineering, University of Science & Technology Beijing, Beijing 100083, China, danren_aa120@sina.com

Abstract

In this paper, we focus on the optimization of the Rete algorithm on aspects of memory consumption and time-consuming in matching process. Facing the problem of requirement of limited memory and quick response from users in the rule pattern matching process of dynamic evolution of software system, on the basis of the classic Rete algorithm and from the perspective of the complexity of algorithm space, Rete network structure matching efficiency have been analyzed, combined with characteristics of system dynamic evolution of the strong dynamic and high efficiency, and based on rule weights, the network time and fact adding, through additional node storage space adjustable mechanism and introduction of facts adding process for self-learning. Comparison test shows that the optimized Rete algorithm could complete the self-regulation of the storage space, and the time-consuming of system running have also been reduced, which satisfied the needs of real-time matching of the evolution of system rules.

Keywords: Rete Algorithm, Rules Engine, Pattern Matching, Software Dynamic Evolution

1. Introduction

With the changing of running environment and service requirement, gradually the software has a self-adjusting ability, namely the formation of dynamic evolution of the software [1]. Since the system running environment and user requirement may change over the life cycle of software, the capability of software evolution has become an important indicator to measure the performance of a system. The rule engine that is evolved by the inference engine of rule-based expert system is an effective way to achieve the system dynamic evolution logic [2], which consists of three parts: the pattern match, the agenda, and the execution engine. Among them, pattern matching is responsible for matching the known pattern of system rule set with the changing information in working memory (WM) that is received by matcher. The corresponding rule will be activated and put into agenda, waiting to be executed if the matching is successful. While the pattern matching algorithm is an important tool for the guarantee of shorter operation time and less memory cost during the matching process. So far, the typical pattern matching algorithm has included the algorithms of LFA, Rete, LEAPS and so on [3]. Among them, the Rete which is proposed by Charles L. Forgy of Carnegie Mellon University in 1974 is an algorithm of Forward-Chaining. After that, Forgy had introduced the Rete algorithm in detail in his 1979 ph.D thesis and paper in 1982 [4]. This algorithm which makes the problems of rule-based reasoning mechanism and matching resolved efficiently is one of the most efficient algorithms of the rule engine pattern matching, and in today’s dynamic global business environment, it has now become the leading product of the world's top commercial business rule engine [5].

This paper is on the basis of the 'SoMan' of the distributed Component (SMC) system, we make thorough research on the problems of the requirements of quick response from users and high real-time of rule matching and the shortage of memory resource. Meanwhile, the classic Rete algorithm is improved: The node storage space adjustable mechanism based on rule weights and first in first delete strategy is established by setting the upper limit of utilization of network two-input node storage space; The time consuming of the two-input node matching process and the system running can be reduced through the facts adding process for self-learning in Rete network matching process. Experimental results have shown that the optimized Rete algorithm could achieve the stable running of system when
a new object is added and the reducing utilization of storage space for deleting facts and the save of
time consuming of system running.

2. Related Conception

Considering how to enable the industrial and automated development of application system, the
problem of dynamic behavior and structure flexibility of software[6] [7] have been the key research
issues because of the open, dynamic and ever-changing Internet[8]. A component model-- 'SoftMan'
component (the SMC) can be formed with the properties of construction and evolution by a special
'component' which is a generation of the 'SoftMan' ontology and focused on the core technology of
dynamic evolution supported by component level explicitly. The SMCOSE, namely, SMC-based
Open System Evolution platform, from lower to upper, consists of four layers: the support platform,
the infrastructure, the evolution control, and the application[9]. In this paper, we mainly focus on rule-
based reasoning in the evolution control layer.

In order to improve the matching efficiency of the whole Rete algorithm and to reduce the amount of
calculation, a part of memory space is sacrificed for saving and taking advantage of the information
(facts, rules, matching circumstances, rule patterns and so on) in the previous pattern matching
process[10]. The algorithm will be discussed from two aspects below, identification of network and
pattern matching process.

The Rete algorithm is compiled to generate a network to make pattern matching[4]. Production rule is
composed of two parts, the left LHS (i.e. conditions) and the right RHS (i.e. conclusion). The Rete
algorithm compiles the left side of the rule into an identification network. Dr.Frogy described four basic
points of identification network[10]: root, one-input (also known as alpha node), two-input (also known
as beta node) and terminal. Root node is a point of entry for facts entering into the network; The one-
input node has an alpha storage space and an input port, where the alpha network of identification
network consists of; The two-input node has both the right and left storage spaces and input ports. The
left storage space is beta storage and the right is alpha storage. The beta network of identification
network consists of two-input nodes, where the information is passed to terminal node represents the
rule corresponding to root node in network is activated, and the movements corresponding to the rule
will be activated and put into agenda waiting to be executed. WME (Working Memory Element), which
is established for showing facts, is an element for matching with non-root node, and it is the smallest
storage unit of storage space. WME can be used as an input of one-input node or the right input of two-
input node. Token can be used for left input of two-input node refers to the WME binding list, and the
list contains one or more WME. If WME is passed to the left of two-input node, WME will be
packaged to Token with only one WME as the left input of two-input[6]. The process of pattern matching
between WME/Token and node is presented in Figure 1.

![Figure 1. Process of Rete algorithm pattern matching](image-url)
(1) If the type matching between WME and successor node of root(a kind of one-input node) is succeeded, pass the WME to successor node and continue to match, otherwise, end the matching.(2) If WME is passed to the one-input node, match WME with the corresponding pattern to this node, if the matching is succeeded, WME will be added to Token, and then pass this Token to the next node, otherwise, end the matching.(3) If WME is passed to the right side of two-input node, the WME will be added to the alpha storage space of this node and match with Token of this node's beta storage, and if the matching is succeeded, Token will become the new Token via packaging and matching which is passed to the next node, otherwise, end the matching.(4) If Token is passed to the left side of two-input node, the Token will be added to beta storage space of this two-input node and match with WME of the alpha storage space, if the matching is succeeded, Token will be added to the Token with only one WME element, and then perform step(4). (5) If WME is passed to the left side of two-input node, the WME is packaged as Token with only one WME element, and then perform step(4). (6) If Token is passed to the terminal node, active the rule corresponding to this root node, and put the movement corresponding to the rule into Agenda waiting to be executed. (7) If WME is passed to the terminal node, package this WME as the Token with only one WME element, and execute step(6).

3. The Analysis of Matching Efficiency of Rete Network Structure

Rete identification network is generated by one or more rules. Before analyzing the matching efficiency of Rete network structure, we assume that there is only one rule and affirmative condition is considered.

The alpha network of the identification network gives WME the existing intrinsic characteristics test. If the test is passed, put the result into the storage space A. Assume $\delta_i$ $(0 \leq i \leq n)$ is the object count of the storage space of $A_i$ (object is the corresponding instance after matching); Because two-input node has both the left and right input ports, the part of beta network mainly uses two-input node to make an interaction WME characteristics test and save the intermediate results of the matching with node into the storage space B. Assume $\theta_j$ $(0 \leq j \leq n-1)$ is the count of objects in the storage space $B_j$, $p_k$ $(0 \leq k \leq n-1)$ is conditional probability of satisfaction of the object to two-input node. Figure 2 represents the Rete network which has n+1 original storage spaces.

![Figure 2. Illustration of Rete network](image-url)
When an object enters into network from storage space $A_0$, i.e., $\delta_0 = 1$, the count of objects in storage space $B_0$ is as follows:

$$\theta_0 = \delta_0 p_0$$  \hspace{1cm} (1)

When the object passes the first two-input node, it will be influenced by the probability of $p_0$. Then the object enters the storage space $B_0$ matches with object $\delta_2$ of $A_2$. The count of objects which enters the storage space $B_1$ after being influenced by $p_1$ is as follows:

$$\theta_1 = \delta_1 p_0 \delta_2 p_1$$  \hspace{1cm} (2)

In the same way, the count of objects which enters the storage space $B_{n-1}$ is as follows:

$$\theta_{n-1} = \delta_1 p_0 \delta_2 p_1 ... \delta_{j-1} p_{j-1} ... \delta_n p_n = n \prod_{j=1}^{n} \delta_j p_{j-1}$$  \hspace{1cm} (3)

Hence, the total number of the added object which reaches storage space $B_{n-1}$ from $A_0$ is as follows:

$$\text{Ram} = \sum_{j=0}^{n} \delta_j + \sum_{j=0}^{n} \theta_j$$  \hspace{1cm} (4)

The equation (4) shows that for a system, the total memory space of the Rete network needed is related to the storage space number of $A_n$, and the object number of $\delta_j$ of every storage space. For a system with more number of storage space and objects, it not only requires large memory space, but also takes much time to match each object.

Comparing the process of deleting facts with adding facts in Rete network, there is one more overhead searching the same record with intermediate matching result produced in the process of deleting the facts in storage space $B_j$, but the two-input node is not discuss in detail here.

4. Improvement of Rete Algorithm for Dynamic Evolution of Software System

In the process of dynamic evolution of software system, the component architecture has to evolve dynamically according to the application requirements and changes of network environment, which mainly on the variability of the count of entity elements, the adjustability of relation structure, and the dynamic configurability of structure form. In the dynamic evolution of 'SoftMan' component (SMC) system, the information of current environment status and users' demands are abstracted as the fact objects stored in fact base of system decision-making device. Since the component has strong environment perception, it reacts to the dynamic environment strongly, resulting in a strong dynamic of component system evolution. The decision driven\cite{11-14} of dynamic evolution is the core of Internet ware. When the facts change, the pattern matching module of decision-making device calls rules management module, and then match all the facts data with rules based on Rete algorithm. Since SMC system requires the average response time to users as short as possible, the system evolution has the hard real-time property as well. So the dynamic evolution process of SMC system has strong dynamic and real-time characteristics.

Aiming at the two main characteristics of dynamic evolution of above-mentioned 'SoftMan' components system and analyzing the matching efficiency of Rete network, this paper proposed the improvement of Rete algorithm in the two following aspects.

(1) Establish the adjustable mechanism of node storage space.

Environment information is constantly changing and requiring value from users continues to accumulate as well. In such strong dynamic environment, the facts continue to add, delete, and update, while the total memory space of Rete network is invariable. In order to meet the strong dynamic property of component evolution, we try to manage the occupancy of memory by introducing node storage space adjustability mechanism. Set upper limit of the utilization of two-input node storage space, and users can adjust the upper limit accordingly to the specific conditions. When the utilization rate is greater than 80%, we remove 20%
memory of this storage space in a lower to higher order according to the corresponding rule weight of objects. If the rule weights are the same, the strategy of first-in-first-delete will be executed according to the time order when rules enter the network. The corresponding rules to these objects’ corresponding memory contents to one-input node and other two-input node are also deleted. To delete corresponding memory contents to one-input node in two cases: If this one-input node is shared with other rules which are not deleted, delete the corresponding stored information to the rules which are going to be delete, otherwise, delete this node from Rete network completely.

(2) Introduction of self-learning in facts adding process.

The key to achieve the strong real-time property of dynamic evolution of component system is to improve the matching efficiency of rules. It can reduce the matching time in the matching process of Rete algorithm by introducing the self-learning mechanism. Recording the information of added facts can reduce the running time of deleting the fact. For the traditional Rete algorithm, if wants to delete a fact from working memory area, initially to make a character test by alpha network, if the test is successful, delete the related content to the corresponding storage space A, and pass the alpha records to the corresponding two-input node. The left condition of this two-input node will be matched with alpha records, if the matching is successful, generates the beta records and check out whether there are same beta records by searching corresponding storage space B. If there is same beta record, delete it and pass this beta record to the next two-input node which is executed in the same deleting process and this process continues until all the content associated with the object have been deleted. It is shown that the input records produced in adding process are not used in deleting process, and contrasted with adding process there is an extra searching process. While the fact is adding, we can retain the intermediate calculation results of matching with two-input node in the storage space B. When deleting the fact, the process of matching with storage space B of two-input node is omitted by the intermediate calculation results of the fact adding process records and only need to search each node memory.

5. Experiments and discussion

There are two main standards to evaluate the efficiency of rule pattern matching algorithm: matching time and usage rate of memory. In Matlab 2006 environment, this paper presents the presupposition evolution experiment of comparing both memory area usage rate and system running time of before and after improved algorithm by SMCOSER which was developed based on Eclipse 3.4.

(1) Comparison of usage rate in memory area

Take $B_i$ as the research object. Presupposition evolution experiment has the result of storage area usage rate in different time by 140 trials of matching the objects of $B_i$ in the case of other storage area usage rate did not reach the upper limit. The result is shown in Figure 3.(Pentagram points to the storage area usage rate before optimized, and diamond points to storage area usage rate after optimized)

In the first 50 tests, there were no new added objects in storage space, and the usage rate of system storage area did not change greatly, so the system ran more smoothly. During the later 50 tests, due to objects entered the network, utilization of the storage area in the storage space changed with the matching objects increased, and test ran smoothly before the 100th time. In the 100th time, new added objects led to the utilization of storage space up to 80% which is the upper limit of memory utilization. As we set that 20% memory is reduced in time when the utilization of storage area is up to 80%, the utilization of storage area is reduced sharply to around 60% after optimized and then ran smoothly.
(2) Comparison of matching time-consuming

The running time of system can be got when we observe the storage area usage rate, i.e. the time taken by the object entering Rete network until matching to the storage space. The result is shown in Figure 4. (Pentagram points to time-consuming before optimized, and diamond points to time-consuming after optimized):
In the first 50 tests, the system was running smoothly, so the time-consuming did not change obviously. After the 50th time, due to new objects entered the network, storage area utilization of the storage space \( B_5 \) was increased, at the same time there were some burden to the running system, so time-consuming increased obviously. But before the second time of added object, time-consuming did not change apparently. In the second time of added object, the storage area utilization of storage space \( B_5 \) has reached the upper limit, which causes the storage space to run dramatically slowly and time-consuming to increase obviously. Because the operation to reduce the utilization of storage area was not set for the network before optimized, the system will run slowly after the time-consuming sharply increased. But after optimized, utilization of storage area was reduced 20\%, there was a certain memory spare in this storage space, and 25\% system time-consuming was saved. So the system time-consuming reduced correspondingly with the decrease of storage area usage rate and system can run faster than before.

6. Conclusion

In this paper, we presented the optimized Rete algorithm on the basis of node storage space adjustable mechanism and self-learning approach for facts adding process. The optimized algorithm realized the dynamic control of the Rete network storage space occupancy and the time-consuming reduction of object removing. The running efficiency of pattern matching algorithm is improved obviously, and the problems of low efficiency of rule matching and the resource issues of memory of SMC dynamic evolution system had been solved. However, the complexity and variability of context environment information cause the strong instability of related facts information in the working space. We achieved the adjustment of memory space shared by Rete network, but in terms of memory space usage self-adaption, it still needs to be researched further.

7. References
