Design and Implementation of Rete Node Indexing for Stream Data Processing Rules

Muhammad Habibur Rahman ᴼ Dongjun Lim ᴼ Bonghee Hong ᴼ Woochan Kim
Department of Electrical and Computer Engineering, Pusan National University
Agency for Defense Development South Korea
Mhabiburr17@pusan.ac.kr, dannylim0709@gmail.com, bhhong@pusan.ac.kr, woochankim@add.re.kr

Summary

Latest military vessel is equipped with multiple radar and sonars. When aerial, water and underwater target objects collected in real-time from naval vessels are collected as real-time stream data, continuous execution of space-time rules for risk analysis and weapon response is required. Existing RETE that mostly handle scalar value is extended to support complex event processing rules. Due to urgency to reduce time consumption, we propose multidimensional node indexing and spatial node indexing that extends the node-link structure of the RETE algorithm to support continuous query.

1. INTRODUCTION

When aerial, water and underwater objects position and movement are collected in real-time from naval vessels as real-time stream data, continuous execution of space-time rules for risk analysis and weapon response is required. It is necessary to process the rule using the spatial and temporal query condition for threat analysis on continuously collected target objects. When target objects that are continuously changing in time and space are inputted as the stream data, the performance improvement problem for finding and executing corresponding rules as quickly as possible must be solved.

Latest military vessel is equipped with advanced radar and sonar technology together. By using these technologies, the enemy movement is captured continuously as a primitive event every second. Each sonar and radar have different coverage areas. If we visualize them, it will be figured as **figure 1**.

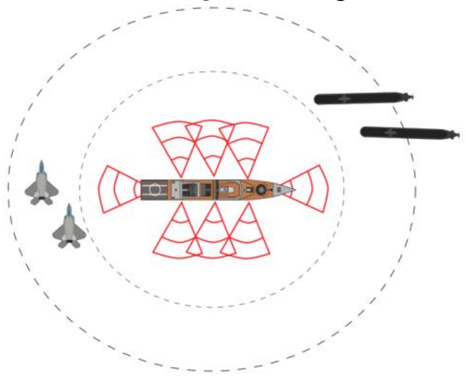


Figure 1: Vessel is equipped with sonar sensor (red) and radar (black dotted)

In this paper, we present the rule-based approach to execute continuous risk analysis. Domain expert need simplest way to express their logic due to lack of technical ability. Continuously changing object's position in space and time make frequent rule modification is unavoidable. The rule might be redundant, affect the next rules, or cascaded. By employing a rule-based approach, domain experts will express their knowledge easier than program the whole code on their own. Also, it will handle rule management to avoid memory leaking.

This research is concerned with risk analysis over moving objects using a rule-based approach. Therefore, enhancement of the rule-based engine to reason the spatiotemporal data is required. The RETE [1] algorithm is a powerful algorithm as a rule-based engine. Originally, RETE is designed to handle scalar value. However, previous works [2, 3, 5] already prove that it is possible to extend the RETE algorithm functionalities to handle spatiotemporal condition.

Spatial and temporal query condition is required for the rule condition must be expressed as a spatiotemporal-parameterized continuous query in which the temporal and spatial range may be vary depending on the time and space. Spatiotemporal condition rules are expressed as follows: IF <Continuous Query Condition> sliding window <time interval> then <event product>.

After all rules are set and compiled, we Index the nodes into multidimensional query indexing. Instead of building index of object's event that keep changing, it is better to keep the queries that more stable. In runtime, each incoming event will be tested with the available rule nodes. This method is called query stabbing.

2. MULTIDIMENSIONAL NODE INDEXING

Our goal is, "*given continuously primitive event, what kind of response that we should execute?*". Hence, we propose four event processing steps. Those are, event filtering, event capturing, continuous query, and complex event processing. To reduce execution time, we propose multidimensional node indexing. Our proposed method is a combination of previous work of RETE algorithm, complex event processing [4], sliding window, and R*-Tree.

2.2 SPATIOTEMPORAL CONDITION FOR RETE

Previous study [5] implements a spatiotemporal rule-engine that based on RETE algorithm to monitor patient's health remotely. They prove that it is possible to extend the spatial and temporal reasoning over RETE algorithm. But their research is conducted over Drools Engine which does not provide processing continuous data stream.

In this paper exploits the benefit of RETE algorithm for processing scalar data, which include node sharing of similar conditions to reduce the number of nodes and repetitive re-computation. However, the rete should be extended to support the spatiotemporal conditions of target objects.

To support spatiotemporal rule condition, we enhance corresponding spatiotemporal node with spatial indexing method which is R*-Tree and sliding windows. By enhancing these two techniques, the node is able to execute spatial query by concentrating on the "given interval". For example, we want to detect if there is an object that is located within a 10 km radius, between certain degree angle with the vessel for 10 seconds, it will be considered a threat. We visualize the stream condition as **figure 2**.

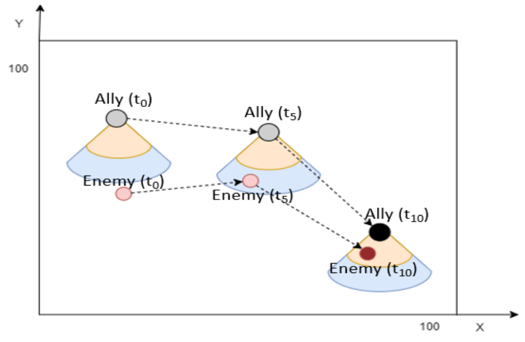


Figure 2: Ally and Enemy position at t_0-t_{10}

Consider we have the following rule,

Rule 1:
IF speed > 3 & elevation < 10 & iff = false
THEN EnemyVessel
Rule 2:
IF Distance (EnemyVessel) < 50 & dir_range(250, 290)
WINDOW range = 10s
THEN CommonThreat, count.objid
Rule 3:
IF Distance (EnemyVessel) < 25 & dir_range(250, 290)
WINDOW range = 10s
THEN MediumThreat, count.objid
Rule 4:
IF approaching (MediumThreat) & MediumThreat.count > 2
WINDOW range = 60s
Then NavalDefenceResponse

From the example above, RETE will compile rule networks. **Figure 3** shows the visualization of the network.

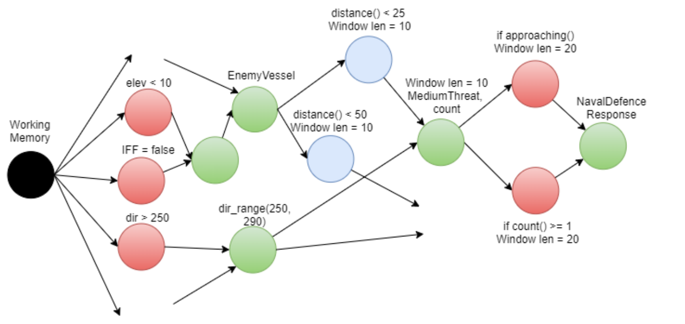


Figure 3: Compiled RETE

Those 4 steps of complex event processing are already covered using RETE algorithm. The first step is event filtering. It is responsible to make sure that there are no duplicate events. Once the events are filtered, it is pushed to working memory. Event capturing is responsible to tracking and refining the simple events from Event Filtering. In this step, each event will be assigned with new meanings. Next is Continuous Query. This step is responsible to doing query continuously. In the end, is Complex Event Processing. It is responsible to query the buffered event that is kept inside the sliding window.

2.3 NODE INDEXING DESIGN AND IMPLEMENTATION

The problem with continuous streaming data is the data is always coming. Previous research [6] they treat the query as data, and their data as query. It means, they keep the query in the database instead of the actual data. It is because query that is more constant size and number than the infinite number of continuous streaming data. Next, when each data is treated as query that will be tested with each of the indexed query. We call this method as query stabbing.

There are two node indexing that we implement here. The multidimensional node will be responsible to index all alpha nodes that placed before the spatial query. Due to different coverage of radar and sonar area, there is a possibility of an enemy object that is located over and overlapping coverage area. Therefore, we implement spatial node indexing to reduce multiple evaluation on multiple spatial nodes. Once the nodes are indexed, then it executes node stabbing. Figure 4 shows the performance comparison.

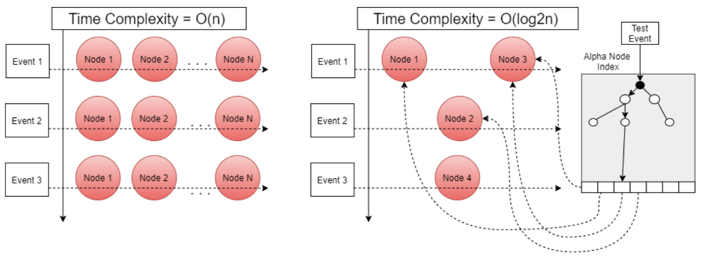


Figure 4: Time complexity of node indexing

There are several steps to insert alpha node into multidimensional indexing, and spatial node into spatial node indexing. Generally, those are decomposing the rule, fetch their range, and put it into the R*-Tree. An example of node Insertion is shown in figure 5 and figure 6 shows the example of spatial node insertion.

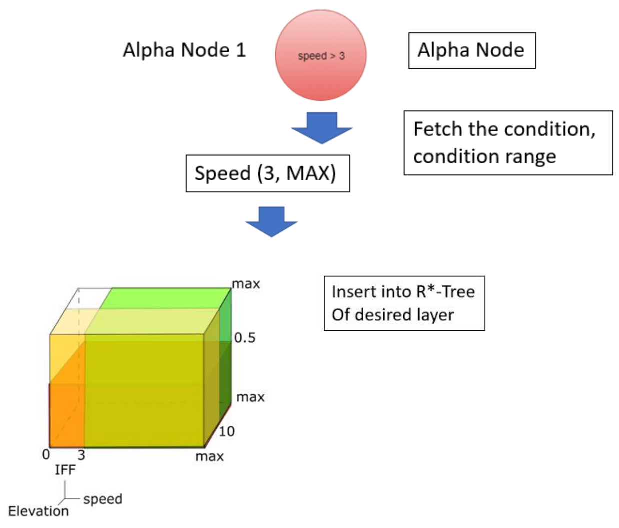
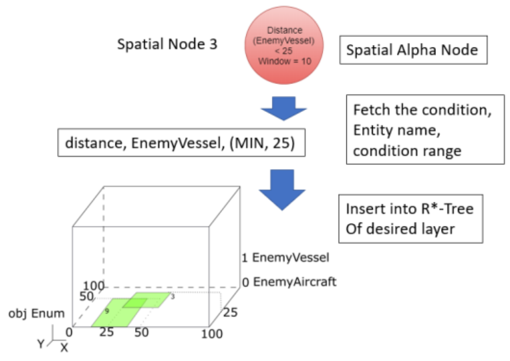


Figure 5: Steps to build the node indexing

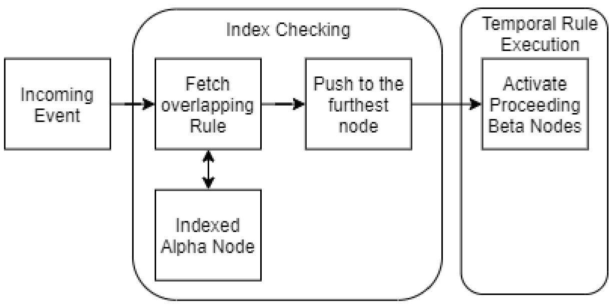


Spatial Node 3
Distance (EnemyVessel) < 25 Window = 10
Spatial Alpha Node
Fetch the condition, Entity name, condition range
distance, EnemyVessel, (MIN, 25)
Insert into R*-Tree Of desired layer
1 EnemyVessel
0 EnemyAircraft
100
50
obj Enum
0
25
50
100
25
Y
X

Figure 6: Steps to build the spatial node indexing

After the nodes are well indexed, we will take advantage of that built index. Simply we convert the index into multi-dimensional point. Next, we will fetch all overlapping rules on that multidimension R*-Tree. At the end, we execute each overlapping rule node so the proceeding rule node will be processed as well. Next, the spatial rule is executed. We also use the indexed spatial node to examine the incoming event. After the final Beta Node is executed, RETE will inform the output to the end-user.

Figure 7: Execution steps



3. EXPERIMENT

In this paper, we are using artificially generated data with several scenario due to data secrecy. The scenario represents if there are several objects coming and retreating with different starting, ending, and trajectories position. figure 8 visualize the scenario. The implementation is done in Windows 10 OS, using 16 GB of RAM, Intel i5 3.2 GHz x64 based processor, and using C++ language. In this experiment, we are going to test the time consumption of different number of object and different number of data length.

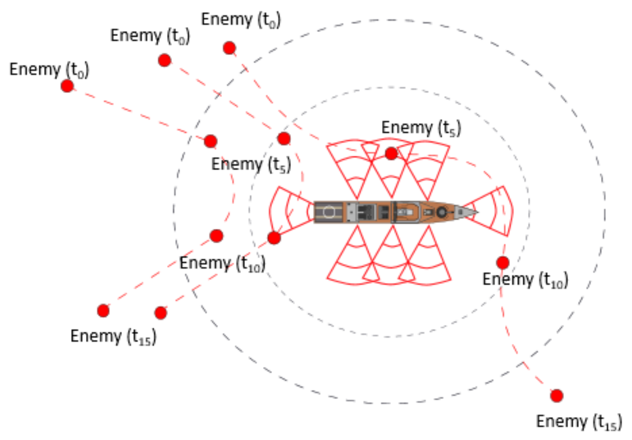


Figure 8: Experiment Scenario

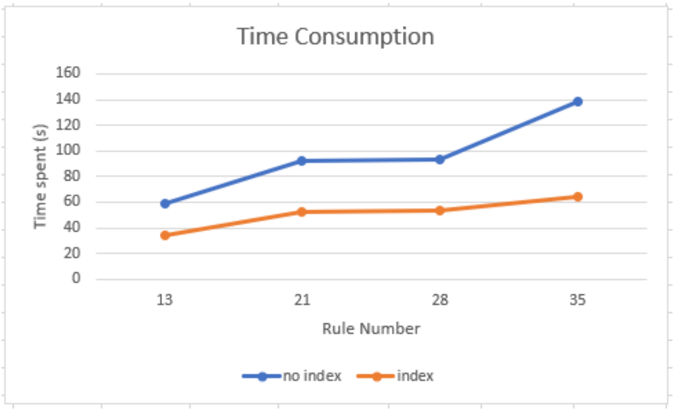


Figure 9: Experiment result with 10.000 events

Table 1: Experiment time result with multiple events and rules

number of rules	1.000 data		5.000 data	
	no indexing	indexing	no indexing	indexing
13	4.105	1.874	24.948	13.687
21	5.992	2.960	37.260	20.394
28	5.989	3.143	37.563	20.750
35	10.308	4.538	65.550	25.457

Figure 9 shows the performance comparison using different number of objects and 10.000 number of data. It is proven that if the node is indexed, it can reduce the time consumption. Table 1 shows the detail of performance experiment.

4. SUMMARY

When aerial, water and underwater objects position and movement are collected in real-time from naval vessels as real-time stream data, continuous execution of space-time rules for risk analysis and weapon response is required. Our goal is, "*given continuously primitive event, what kind of response that we should execute?*". Hence, we propose four event processing steps. Those are, event filtering, event capturing, continuous query, and complex event processing. To reduce time consumption, we implement multidimensional node indexing and spatial node indexing.

We are planning to compare the performance this method with another existing method in the future. We also planning to evaluate our method with various tactical object movement scenario.

Acknowledgments

This work was supported by the Agency of Defense Development (UD190003DD).

REFERENCES

[1] Forgy, Charles L. "Rete: A fast algorithm for the many pattern/many object pattern match problem." Readings in Artificial Intelligence and Databases. Morgan Kaufmann, 1989. 547-559.

[2] Ray, Cyril, et al. "Spatio-temporal rule-based analysis of maritime traffic." 2013.

[3] Merilinna, Janne. "A mechanism to enable spatial reasoning in jboss drools." 2014 International Conference on Industrial Automation, Information and Communications Technology. IEEE, 2014.

[4]. Yihuai Liang, Jiwan Lee, Bonghee Hong, WooChan Kim. "Rule-based Complex Event Processing on Tactical Moving Objects," IEEE 4th International Conference on Computer and Communications, 2018.

[5] Pathak, Ravi, and V. Vaidehi. "Complex event processing based remote health monitoring system." 2014 3rd International Conference on Eco-friendly Computing and Communication Systems. IEEE, 2014.

[6] D Kalashnikov, S Prabhakar, S Hambrusch. "Efficient Evaluation of Continuous Range Queries on Moving Objects," International Conference on Database and Expert Systems Applications, 2002.