

基於規則庫切割的元知識建造方法
Meta-rule Construction based on Rule Base Partitioning

研究生：溫建豪

Student : Chien-Hao Wen

指導教授：曾憲雄

Advisor : Dr. Shian-Shyong Tseng

國立交通大學
資訊科學系
碩士論文



Submitted to Institute of Computer and Information Science
College of Electrical Engineering and Computer Science

National Chiao Tung University

in partial Fulfillment of the Requirements

for the Degree of

Master

in

Computer and Information Science

June 2004

Hsinchu, Taiwan, Republic of China

中華民國九十三年六月


基於規則庫切割的元知識建造方法

研究生：溫建豪

指導教授：曾憲雄博士

國立交通大學電機資訊學院

資訊科學系



摘要

在資訊科學的範疇中，專家系統（Expert System）的應用越來越為廣泛，儼然成為下一代資訊系統的重要特色；而在專家系統的建置技術方面，規則庫（Rule Base）是一種廣為採用的方式，在規則庫中的知識，被設計為人類容易瞭解的邏輯式規則，更使得規則庫無論在設計上、應用上，都能夠符合知識工程的需求。然而，由於資訊系統的進步以及電腦硬體能力的增強，使得規則庫的規模有越來越為龐大的趨勢，因而造成規則庫的規模增加，從此產生了許多管理上的議題。在本論文中，藉著結合規則庫切割（Rule Base Partitioning）和元知識（Meta-knowledge）建造機制，RP-MES 被提出來解決規則庫管理議題，其中透過對於規則庫結構與語意的分析，協助建置更有效率且更容易維護的規則庫。此外，我們也設計及實做一基於 RP-MES 的入侵偵測雛型系統，以及關於系統效能的數個實驗。實驗結果顯示，透過 RP-MES 可以將規則庫分割為合適大小的規則群，並維持知識的正確性，同時顯示最終的執行效能相較於原本不經過分割的規則庫之效能有著明顯的提昇。


Meta-rule Construction based on Rule Base Partitioning

Student : Chien-hao Wen

Advisor : Dr. Shian-Shyong Tseng

**Institute of Computer and Information Science
National Chiao Tung University**

Abstract



Expert system technology becomes more and more important in computer science domain for next generation computer systems. For constructing an expert system, rule base is a widely used approach, where knowledge and expertise are represented as production rules. However, due to the growth of rule base usage, the scale of rule base is increasing and hence many management related issues arise. By designing a new approach combining both rule base partitioning mechanism and meta-rule construction mechanism, RP-MES is proposed to solve these issues in this thesis. An Intrusion Detection System (IDS) prototype is also designed and implemented based on RP-MES, and some experiments have been done to evaluate the system performance. The experimental results show that RP-MES can produce reasonable number of rule clusters and the accuracy of the inference result remains, and that the performance of RP-MES is better than that of original rule base without partitioning.

致謝

能順利完成本篇論文，最重要的必須感謝我的指導教授，曾憲雄博士，曾教授在我碩士班兩年期間相當耐心得指導我論文研究；從他身上我也學習許多領導處事的技巧，這些寶貴的經驗讓我獲益匪淺，感激不盡。同時也感謝我的口試委員，李允中教授、何正信教授、和洪宗貝教授，他們給予了我相當多的寶貴意見，讓本篇論文更有價值。

第二位要感謝的人是林耀聰學長，兩年期間讓我學會許多理論知識及實務技巧，也給予了我許多對於此篇論文的寶貴意見，接受我的詢問和討論，和協助我論文修改工作，深表感激。

此外我必須感謝實驗室學長平日的諸多協助，特別是王慶堯、林順傑、曲衍旭。同時也感謝實驗室同窗夥伴，曾于彰、鄭佩琪、陳家瑜、王威、蘇培綺等人在生活上和課業上互相幫忙的情誼；以及學弟李育松、黃柏智在論文實驗上的支援，深表感激。

最後我要感謝我的家人對於我的支持與鼓勵，讓我在面對挫折的時候能夠繼續向前，也讓我能夠有相當的自信完成本篇論文，深表感激。

Table of Content

ABSTRACT (IN CHINESE)	I
ABSTRACT	II
ACKNOWLEDGEMENT	III
TABLE OF CONTENT	IV
LIST OF FIGURES	V
LIST OF TABLES	VI
CHAPTER 1. INTRODUCTION	1
CHAPTER 2. RELATED WORK	5
2.1 RULE BASE MAINTENANCE.....	5
2.2 RULE BASE PARTITIONING	7
2.3 META-KNOWLEDGE.....	10
CHAPTER 3. RULE BASE PARTITIONING AND META-RULE EXTRACTION SYSTEM (RP-MES)	12
3.1 THE MECHANISMS OF RP-MES	12
3.2 SYSTEM ARCHITECTURE	13
CHAPTER 4. AUTOMATIC META-RULES CONSTRUCTION	18
4.1 OVERVIEW	18
4.2 RULE BASE PARTITIONING PROCESS	18
4.3 META-RULE CONSTRUCTION PROCESS.....	32
CHAPTER 5. EXPERIMENT	40
5.1 EXPERIMENT ENVIRONMENT	40
5.2 EXPERIMENTAL RESULTS	41
CHAPTER 6. CONCLUSION	46
REFERENCE	47

List of Figures

FIGURE 1. TRADITIONAL RULE-BASED SYSTEM VS. META-RULE ADDED RULE-BASED SYSTEM. 11

FIGURE 2. THE SYSTEM ARCHITECTURE OF RP-MES..... 14

FIGURE 3. AUTOMATIC META-RULE CONSTRUCTION PROCESSES. 18

FIGURE 4. COMPONENTS OF RULE BASE PARTITIONER. 19

FIGURE 5. ILLUSTRATION OF THREE CASES OF ATTRIBUTE REFERENCE.23

FIGURE 6. PART OF THE ONTOLOGY IN THE NETWORK MANAGEMENT DOMAIN.....27

FIGURE 7. COMPONENTS OF META-RULE EXTRACTOR.....33

FIGURE 8. AN IDS BASED ON RP-MES.....40

FIGURE 9. PARTITIONING RESULTS BY DIFFERENT SIMILARITY THRESHOLD SETTINGS.43

FIGURE 10. COMPARISON OF NUMBER OF CLUSTERS UNDER THE SAME ACCURACY.....44



List of Tables

TABLE 1. ENCODINGS OF EXPRESSIONS IN RB	36
TABLE 2. THE SUPPORT COUNT OF EACH CANDIDATE IN C_{11}	37
TABLE 3. SUPPORT COUNT OF EACH CANDIDATE IN C_{12}	37
TABLE 4. SUPPORT COUNT OF CANDIDATE IN C_{13}	37
TABLE 5. PARTITIONING RESULTS BY DIFFERENT SIMILARITY THRESHOLD SETTINGS.	42
TABLE 6. ACCURACY COMPARISONS OF PARTITIONED RULE BASE.	43
TABLE 7. COMPARISON OF NUMBER OF CLUSTERS UNDER THE SAME ACCURACY.	44



Chapter 1. Introduction

In recent years, expert system, the system to model expert's decision making process and help build up knowledge systems, becomes more and more important in computer science domain for next generation computer systems. For constructing an expert system, rule base is a widely used approach, where knowledge and expertise are represented as production rules, a well-known logical knowledge representation. It is easy to construct knowledge system using rule base since the representation of knowledge, the storage of knowledge, and the processing of knowledge are well designed in rule base system.

Several kinds of knowledge discovery approaches, such as knowledge acquisition [18], data mining algorithms [1], knowledge fusion [21], are used to extract rules. Designing and maintaining such rule base is a very difficult task, and many researches have been proposed to construct rule base [10][13]. However, due to the growth of rule base usage, the scale of rule base is increasing and hence many management related issues arise about constructing the rule base [5][12][16][30]. Three issues, including complex relationships between rules, structural errors, and performance degradation, are summarized from previous researches [4][7][16][27] about rule base construction. For the first issue, the relationships between rules in a huge rule base can be too complicated for both human to manage and computer to process, and they may cause the difficulty of tracing and explaining the rule base results. Also, the structural errors contained in the rule base which can not be easily detected without well-defined structure of the knowledge can make the rule base inference result incorrect or useless. Moreover, the performance can be dramatically decreased and hence cause the

inference engine of the rule base consuming too many resources when the number of rules in the rule base increases.

Among previous researches proposed to resolve these rule base management issues, rule base partitioning is one of the most popular approaches whose basic idea is grouping rules according to a given criterion, and the grouping process is called modularization [4]. The motivation of modularization is to improve system performance and remain the maintainability of a rule base. That is, rule base partitioning groups rules into smaller rule clusters, which are easier for computer to process and easier for human being to understand and maintain. However, traditional rule base partitioning deals with only the structural relatedness between rules; moreover, rule base partitioning provide few information about partitioned rule partitions and hence causes some problems about management and usage of the rule base. There are some other researches [3][7] of meta-rules aim to solve the issue about selecting the appropriate rule partition, in which the meta-rule is used to store the knowledge about each rule partition.

In this thesis, an RP-MES is proposed to solve these issues by designing a new approach combining both rule base partitioning mechanism and meta-rule construction mechanism. For rule base partitioning, RP-MES not only takes care of the structural relatedness between rules, but also consider the semantic relatedness of rules by calculating the semantic relationship between rules in the rule base. On the other hand, RP-MES extracts the meta-rules from the rule cluster partitioned by the rule base partitioning mechanism, and uses the meta-rules for selecting rule cluster when using the rule base.

There are several components in RP-MES, including Automatic Meta-rule Constructor, Verificator, Application and Management User Interfaces, and Inference Engine. Automatic Meta-rule Constructor is used to partition rules from structureless rule set into different rule clusters according to the rule similarity; Verificator will verify and validate the rules inside each rule cluster, and eliminate the confliction, redundancy, contradiction, circularity, and incompleteness, in the rule cluster. The Automatic Meta-rule Constructor is also used to generate the meta-rules of the rule base using the rule clusters generated, and the meta-rules will be used for rule selection in the usage of the rule base. The Inference Engine is used to process the meta-rules, and according to the result of meta-rule reasoning, the appropriate rule cluster(s) should be selected and inferred. Application and Management User Interfaces provide the GUI for user to manage this system, including rule editing, rule clusters navigation and all the basic functions for user to access the system.



Automatic Meta-rule Constructor, the major component in PR-MES, is to process the rule base into meaningful structure and also extract the meta-information from rule base for better management and usage. Automatic Meta-rule Constructor, which consists of two major components, Rule Base Partitioner and Meta-rule Extractor. Rule Base Partitioner divides set of rules into several rule clusters according to rule similarity, which is calculated based on both structural relatedness and semantic relatedness. For calculating the structural relatedness, the inter-relationship between the rules will be analyzed, including the relations of share-in, share-out, in-out, and not-shared. And for calculating the semantic relatedness, an ontology of the domain knowledge will be used, and also the operators and values used in the rules will be

used as part of our calculation. Moreover, Meta-rule Extractor generates meta-rules from each rule cluster by extracting the embedded information between the rules inside the rule cluster. Using the Meta-Apriori algorithm proposed, the frequent expressions of each rule cluster will be discovered and translated into meta-rules, and also the confidence of each meta-rule will be re-calculated.

In order to evaluate the performance of RP-MES, an Intrusion Detection System (IDS) prototype is designed based on RP-MES, and also some experiments are used to test the system. In the experiments, the accuracy of the meta-rules and rule clusters generated will be analyzed, and also the performance of generating rule base will be evaluated.

The organization of this thesis is as follow: Chapter 2 introduces the related work about rule base maintenance, rule base partitioning and meta-knowledge. RP-MES is introduced in Chapter 3. Chapter 4 detailedly describes the mechanism of Automatic Meta-rule Construction. The implementation of the IDS prototype system based on RP-MES and corresponding experimental results are given in Chapter 5. At the end, concluding remarks are given in Chapter 6.

Chapter 2. Related Work

In this chapter, related work about rule base maintenance and related researches will be introduced first, and also the issues to be solved in rule base maintenance will also be mentioned. Rule base partitioning, one of the approaches to solve these issues, is then described. Besides, the definition of meta-knowledge, which is often used to provide information about partitioned rule base, and some related researches are mentioned, which can help us to understand the usage of meta-knowledge in knowledge base maintenance issues.

2.1 Rule Base Maintenance



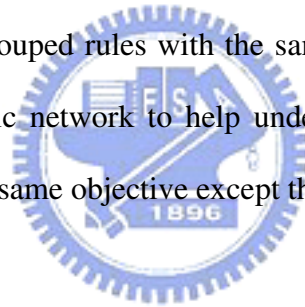
Knowledge-based programming has been widely used in many application fields [13]. With knowledge-based system, the application developers can easily enhance their program ability by modifying the knowledge contained in a knowledge base. In another word, knowledge-based programming provides lots of flexibility for system developers. But for administrators of these knowledge-based systems, the maintenance of the system can be a critical task if they do not familiar with the domain of knowledge base used. The knowledge contained in knowledge base can be represented as several forms: production rules, semantic networks, schemata, frames, and logic [13]. Since production rules are widely used in many real systems, we thus focus on rule base maintenance in this work.

From the previous researches, there are three issues about rule base maintenance,

including complex relationships between rules [16], structural errors [27], and performance degradation [4][7], which are summarized from previous researches, will be described as follows.

(1) *Complex relationships between rules*

Wrong relationships including *duplication*, *subsumption*, *overlap*, and *adjacency* [16] between rules in a rule-based knowledge system are some of the major reasons to make rule base maintenance difficult. In [16], the author proposed an approach to detect and resolve such relationships. Unlike procedural code, traditional rule base has no explicit structures, such like modules, to help administrators understand it. For this reason, some researchers tried to decompose rule base into pieces of modules or groups. Harandi et al. [14] grouped rules with the same atomic topic as a rule group and then generated a semantic network to help understand relationships of groups. Kuo et al. [21] directed to the same objective except that the final result is ontology.

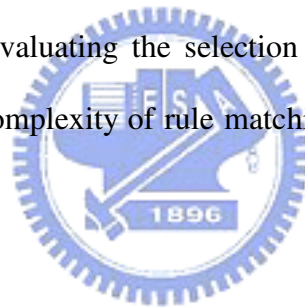


(2) *Structural errors*

For successful deployment of rule based systems, it is important that they be error-free. Nazareth [27] provided taxonomy of errors that rule base design may encounter, called *structural errors*, which consist of *redundancy*, *contradiction* or *conflict*, *circularity*, and *incompleteness*. Verification and validation processes are usually used to detect those structural errors. There are many kinds of verification processes are proposed, such like directed graph [28], Petri net [41][15], hypergraph [39], and directed hypergraph [32]. However, those approaches are not appropriate for large rule bases with thousands of rules due to requiring huge amount of storage and computation resource. Sakar and Ramaswamy suggested decomposing the rule base that allowed verification to be performed in a modular fashion [34].

(3) *Performance degradation*

Most rule-based systems would perform inefficiently when the number of rules in them is huge. For each problem task, the system chooses the appropriate set of rules to apply. However, the number of rules in the set is large when dealing with large rule base which often consists of thousands of rules. Without properly reducing the size of rule set, the system performs poorly. To avoid such situation, some researchers proposed lots of rule selection mechanisms, including neural network learning [1], meta-rule [7], etc. Therefore, for each task, rule-based system applies rule selection mechanism to choose the proper rules, and the experimental result reveals the speedup of inference process. Note that increase of performance does not come free since there is an overhead involved in evaluating the selection mechanism. However, in most cases, the sum reduction in complexity of rule matching process compensates for the increase in the overhead cost.



To solve those issues, rule base partitioning is one of the approaches to achieve it. Therefore, existing rule base partitioning approaches will be reviewed in the next section.

2.2 Rule Base Partitioning

The basic idea of rule base partitioning is grouping rules according to a given criterion, and the grouping process is called *modularization* [4]. The motivation of modularization is to improve system performance and also to remain the

maintainability of a rule base. That is, rule base partitioning groups rules into smaller rule clusters, which is easier for computer to process and easier for human being to understand and maintain. Like most software engineering, the system capability can be enhanced by replacing original rule cluster with new one. In this section, several rule base partitioning techniques are introduced, including the approaches based on probability [26], graph [19][26][21], clustering [19][20][37], information-theoretic [19][20], and genetic algorithm [8][11][17].

There are some researches perform rule base partitioning using clustering analysis approaches, but usually the definitions of distance or relatedness between rules are different in various approaches. In [20], relatedness between rules is measured by the number of facts that are shared in both rules. Lee et al. [22] defined “static distance” between rules. Here “static” refers to the idea that the distance is fixed in terms of the syntactic structure of the rule base. Stanfel et al. adopted shortest path between two components as the weighted connection [37]. Except the definitions of distance can be different, clustering algorithms applied in rule base partitioning can be also different. Raz and Botten mathematically formulated the rule base partitioning problem; that is, the problem of allocation of rules or rule groups among partitions of limited size while minimizing the sum of inter-partition connections, as a 0-1 integer programming problem with a quadratic objective function. Since this type of problems is NP-complete, a clustering algorithm based on the nearest neighbor heuristic is proposed [33].

In [19][20], Jacob et al. proposed a hybrid approach which combining cluster analysis, information-theoretic and graph theory to solve rule base partitioning

problem. In this work, they have proposed a two-phase algorithm; the first phase partitions the rule base graph into a set of trees of rules, where the root of each tree was a fact. That is, divide the rules into groups such that each group of rules produces only one fact that is used by other groups. The group relatedness is defined thereafter. The second phase partitions the original rule base by combining groups produced in the first phase according to the group relatedness measurement. The result revealed that these two algorithms performed better when combined.

Based on the formulation proposed by Raz and Botten, several researches applied evolutionary algorithms to find near optimal solutions for allocation of rules. The representation of a solution is crucial for the performance of a genetic algorithm. Dev et al. [8] designed a genetic algorithm with an integer string representation for the rule base partitioning problem. Dutta [11] used the same string representation and designed an evolutionary heuristic to find valid string (result) as fast as possible. In the works of Ho et al. [17], they proposed an intelligent genetic algorithm which is based on the orthogonal arrays. The experimental results of those evolutionary approaches show that they all perform efficiently than Raz's heuristic clustering algorithm.

Cluster analysis is one of the most frequently used approaches to solve the rule base partitioning problem. The concept of cluster analysis is intuitive and the result can be interpreted by human being. However, many researchers focused on syntactic structure of the rule base, and less concentrated on logical meaning between rules. Also, those techniques discussed above provide less information about the result partitions for further usage. For example, generating meta-knowledge to describe the clusters

generated. Meta-knowledge is used in some researches [14][21] to represent the information of rule cluster, which is proved to be useful in rule base management and processing.

2.3 Meta-knowledge

Knowledge about a particular task domain is called *object-level knowledge*, and information about object-level knowledge is *meta-level knowledge* (or *meta-knowledge*) [7]. For rule base partitioning issue, the rule cluster generated is the object-level knowledge and meta-knowledge can be used to represent the information about rule cluster. Several representations of meta-knowledge are defined in previous researches, such as meta-rule [1][7], semantic network [14], ontology [21], and constraints to restrict the hypothesis space in ILP (inductive logic programming) systems [24]. Meta-knowledge has been used to control the reasoning process of inference engine in many research works [3][7]. In these applications, three functionalities of meta-knowledge are grouping, rules selection, and rules ordering. Meta-knowledge for grouping is used to group related rules in the knowledge base. Meta-knowledge of rule selection and rule ordering can help select and prioritize the rules to be processed when solving a given problem. However, additional mechanism may be needed to process meta-knowledge. Davis recommended encoding meta-knowledge as production rules, called *meta-rules*; thus, the inference engine could be used to deal with meta-rules.

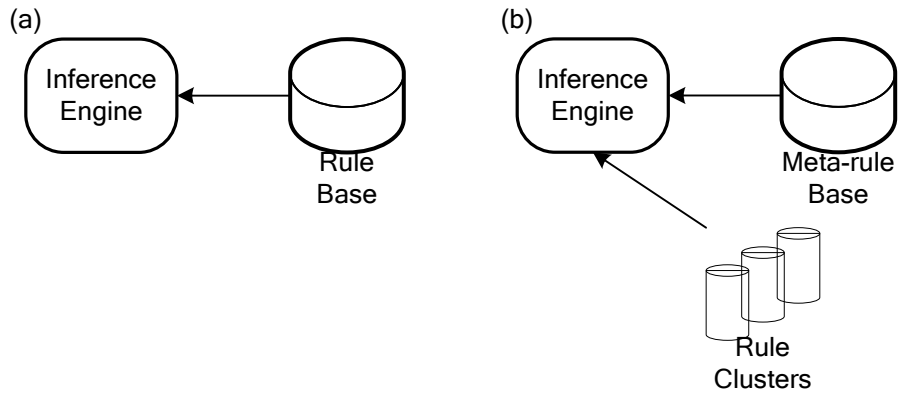


Figure 1. Traditional rule-based system vs. meta-rule added rule-based system.

In the rule base without meta-knowledge, depicted as Figure 1 (a), the rule base is difficult to understand due to lacking of explicit structure. While dealing with the given task, the number of rules needed to evaluate is large since there is no information to guide inference engine. Meta-rules in the rule base system can be used to solve these issues. In the rule base with meta-rules, depicted as Figure 1 (b), each rule cluster is the set of rules and can be described by one or more meta-rules to help human being understand structure of rule base. Besides, meta-rule base is used by inference engine to choose the appropriate rule clusters that are used to solve the given problem task. Such mechanism can greatly reduce the number of rules needed to evaluate.

Chapter 3. Rule Base Partitioning and Meta-rule Extraction System (RP-MES)

To solve the issues mentioned previously, the system called *Rule base Partitioning and Meta-rule Extraction System* (RP-MES) is proposed in this chapter. In the following sections, the mechanisms of RP-MES are described first, and then system architecture of RP-MES is also introduced.

3.1 The Mechanisms of RP-MES

As discussed in Chapter 2, there are three issues about rule base maintenance, including complex relationships between rules, structural errors, and performance degradation. In order to avoid these three issues, RP-MES has following mechanisms.

(1) *Rule base partitioning*

For a rule base containing sophisticated expert knowledge, the amount of rules may be huge, and the relationships between rules may be quite complicated. Modification of such a rule base may cause the rule base to be inconsistent. Unlike procedural programming, rule base does not have explicit structure, and the relationships between rules may be not aware when users try to manage the rule base. For this reason, it is intuitive to group related rules in the same partition (similar to the concept of module in procedural programming) according to atomic topic, subject area, or other criteria; because that the relationships in a smaller rule partition are easier to manage and understand. Hence, rule base with good modularity is the rule base in which rules are

well-partitioned and managed.

(2) Rules verification and validation

A rule base is reliable only in case there are no structural errors inside the rule base and the result of the rule base is always consistent. However, as mentioned before, a rule base may be huge and there may be many structural errors, which cause errors when using the rule base. According to previous researches, these errors can be detected and corrected by some verification and validation processes. However, verification and validation in a large rule base can be very inefficient due to the complication of rule relationships and huge amount of computation resource. An efficient mechanism to verify and validate a rule base is helpful to build up a reliable rule based system.



(3) Rule selection

The processing of rule base may take quite amount of time to match the rules; however, performance is a key index for modern computer systems; a rule selection mechanism to avoid matching all rules during the inference can be very helpful to improve the performance.

3.2 System Architecture

According to the mechanisms defined in Section 3.1, the system architecture of RP-MES is proposed as shown in Figure 2.

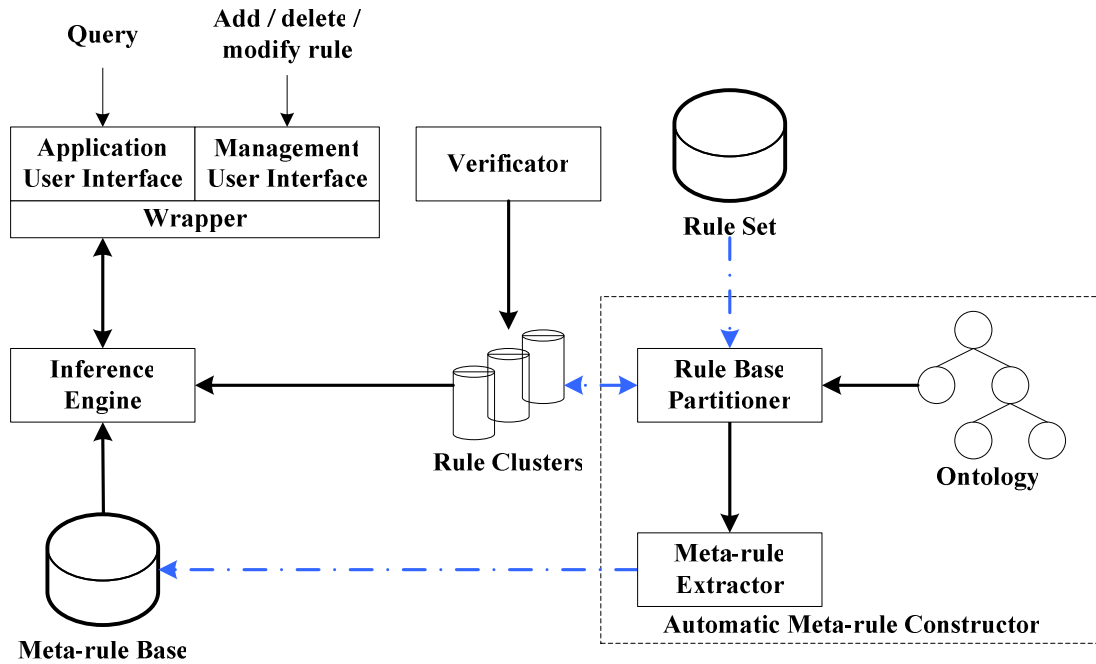


Figure 2. The system architecture of RP-MES.

In RP-MES, *Automatic Meta-rule Constructor* is used to partition rules from structureless rule set into different rule clusters according to the rule similarity; *Verifier* will verify and validate the rules inside each rule cluster, and eliminate the confliction, redundancy, contradiction, circularity, and incompleteness, in the rule cluster. The *Automatic Meta-rule Constructor* is also used to generate the meta-rules of the rule base using the rule clusters generated, and the meta-rules will be used for rule selection in the usage of the rule base. The *Inference Engine* is used to process the meta-rules, and according to the result of meta-rule reasoning, the appropriate rule cluster(s) should be selected and inferred. *Application and Management User Interfaces* provide the GUI for users to manage this system, including rule editing, rule clusters navigation and all the basic functions for users to access the system. In the following paragraphs, these components will be detailedly introduced.

3.2.1 Application and Management User Interfaces

Like most knowledge-based systems, the proposed system architecture provides application user interface for users to execute queries and get results. Management user interface allows users to add, delete, or modify rules with user friendly interface to prevent user from accessing detailed rule base structure and configuration. Management user interface also displays meta-rules of each rule cluster to help user understand the rule base structure.

3.2.2 Verifier

The Verifier of RP-MES is used to verify and validate the rules of the rule base by different rule clusters. Since the domain knowledge may be different due to the growth of domain researches and discovery, the rule base may be also modified to be adapted to new knowledge. However, the modification of rule base may cause some errors and rule based system may obtain unreasonable results, e.g., contradiction or conflict, incompleteness, etc. Also, errors of rule base will cause Automatic Meta-rule Constructor generates incorrect rule clusters and meta-rules. Therefore, it is important to make sure that rule base is consistent and complete. Hence, the Verifier of RP-MES is proposed to verify rule base in order to eliminate the structural errors when rule base is modified. The Verification is used to verify each rule cluster instead of entire rule base in RP-MES, in order to reduce the complexity of verification and validation process and increase the performance.

3.2.3 Inference engine

In order to support the rule selection function by inferring the meta-rules of rule

base, the inference engine used in RP-MES should have the ability to process meta-rules, and also select the rule cluster according to the inference result of meta-rules. In NORM [23], a new knowledge model, proposed to process knowledge modularization and knowledge relations, is used to represent the meta-rule and rule cluster in RP-MES. There are four knowledge relations defined in NORM, including *reference*, *extension*, *acquire*, and *trigger*, in which acquire relation can be used to dynamically link a rule cluster according to given criteria, and the criteria can be the meta-rule in RP-MES. Hence, we use NORM and corresponding inference engine in RP-MES to process the rule inference and rule selection functionality.

3.2.4 Automatic meta-rule constructor

Meta-rules provide some information about each rule cluster. With meta-rules, the structure of the rule base can be easily understood, and in inference process, meta-rules can be used to select appropriate rule clusters which increase the performance of the usage of rule base. However, meta-rules are not easily to obtain. In previous applications about meta-rules [1], the set of meta-rules are usually provided by domain experts; acquiring meta-rules can be time consuming and the expertise may be not available. Therefore, a systematic mechanism is desirable to generate meta-rules. Accordingly, RP-MES incorporates Automatic Meta-rule Constructor, which consists of two major components, *Rule Base Partitioner* and *Meta-rule Extractor*. Rule Base Partitioner divides set of rules into several rule clusters according to rule similarity, and Meta-rule Extractor generates meta-rules from each rule cluster by extracting the embedded information between the rules inside the rule cluster. The detail of Automatic Meta-rule Constructor and corresponding components

will be introduced in Chapter 4.



Chapter 4. Automatic Meta-rules Construction

4.1 Overview

The Automatic Meta-rule Constructor consists of two processes as illustrated in Figure 3. The first is *Rule Base Partitioning Process*; it considers syntactic and semantic structures of given rule base, and then partitions rule base into several rule clusters. The second is *Meta-rule Construction Process*; meta-rules will be extracted from the rule clusters obtained in previous phase. These two phases will be described in detail in the rest of this chapter.

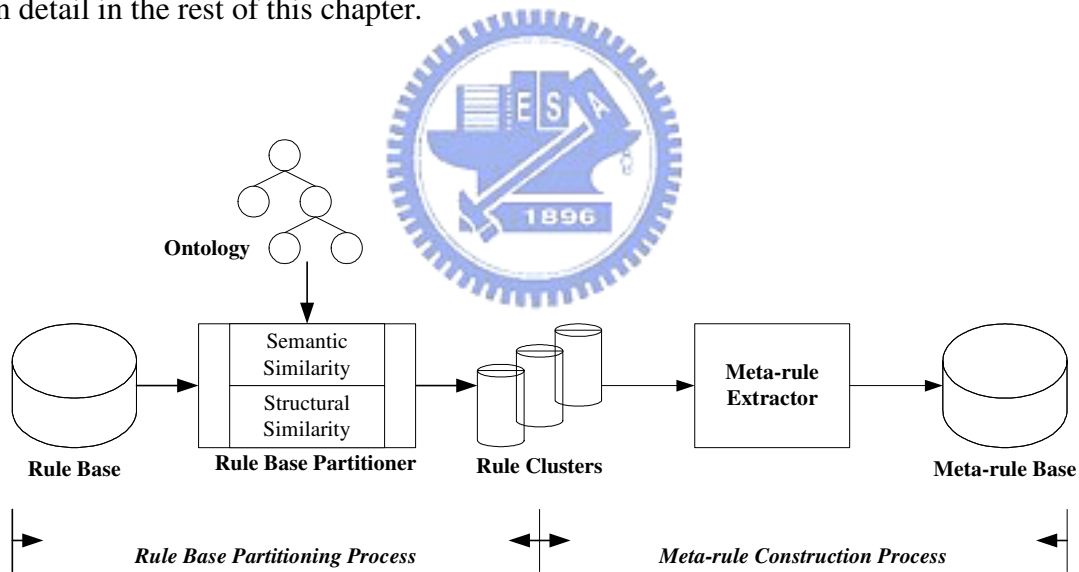


Figure 3. Automatic Meta-rule Construction Processes.

4.2 Rule Base Partitioning Process

In this process, Rule Base Partitioner is used to group rules into rule clusters from a plain rule base without any structure. Figure 4 illustrates the detailed process of Rule

Base Partitioner.

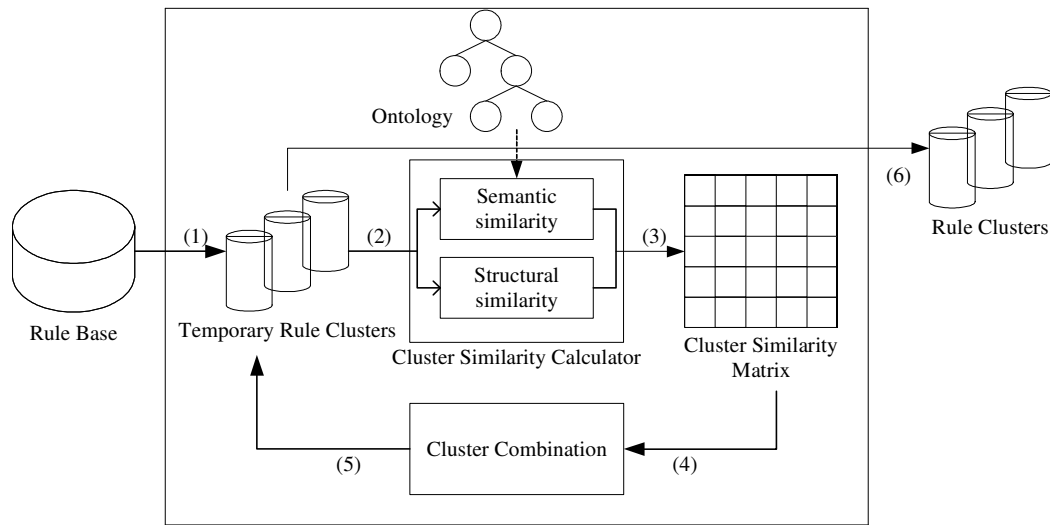


Figure 4. Components of Rule Base Partitioner.

At beginning, each rule of the original rule base is allocated into a single rule cluster. *Cluster Similarity Calculator* calculates cluster similarity of all pairs of two distinct rule clusters and builds a *Cluster Similarity Matrix* (CSM). And the rule clusters will be merged according to the information in Cluster Similarity Matrix. The merge process works iteratively until all similar rule clusters are merged. And the rule clusters generated will be the result of this process.

However, the similarity calculation can seriously affect the result of this process, and the merge process of rule cluster is also an important task. In the following paragraphs of this section, similarity calculation will be introduced first. After that, rule base partitioning algorithm will be detailedly described in the forthcoming section.

4.2.1 Rule Similarity

Rule similarity is a key factor to the clustering result. Well defined rule similarity definition is important for a meaningful rule base partitioning result. Several kinds of rule similarity definitions considering structural relatedness only [20][22], or with semantic relatedness (hybrid approach) [21][38] are defined in previous work. In RP-MES, Rule Base Partitioner incorporates hybrid approach to deal with rule similarity calculation.

Before discussing the rule similarity calculation, some notations will be given to be used in following discussions.

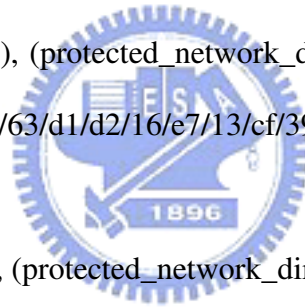
Definition 4.1 Expressions, conditions, actions, rules, and rule base.

- $A = \{attribute_1, attribute_2, \dots, attribute_N\}$: the set of all attributes in the rule base.
- $O = \{=, \neq, >, <, >=, <=\}$: the set of all operators used in the expressions
- $V_{attribute_m}$: is the set of possible values of $attribute_m$, where $attribute_m$ is the corresponding attribute of expression e_m .
- $e_m = (attribute_m operator_m value_m)$ is an expression, where $attribute_m \in A$, $operator_m \in O$, and $value_m \in V_{attribute_m}$.
- $CONDITIONS_i$: a set of expressions of rule r_i , and expressions in the set are connected with conjunction operator (AND).
- $ACTIONS_i$: a set of expressions of rule r_i , and expressions in the set are connected with conjunction operator (AND), where the operator of the expressions must be “=”.
- r_i : a rule of two-tuple ($CONDITIONS_i, ACTIONS_i$) which can be represented as “IF $CONDITIONS_i$ THEN $ACTIONS_i$ ”.

- RB : the set of rules in the rule base.

Example 4.1 Five rules used to detect different network anomalies are showed and form a rule base RB , that is, $RB = \{r_1, r_2, r_3, r_4, r_5\}$.

- r_1 : *IF* {(protocol = TCP), (protected_network_direction = A), (source_port > 8080), (string = NetBus)} *THEN* {(name = NETBUS)};
- r_2 : *IF* {(protocol = TCP), (protected_network_direction = A), (source_port > 1023), (string = NetBus2)} *THEN* {(name = NETBUS2)};
- r_3 : *IF* {(protocol = UDP), (protected_network_direction = A), (destination_port > 1023), (string = /ce/63/d1/d2/16/e7/13/cf/3c/a5/a5/86)} *THEN* {(name = DIR)};
- r_4 : *IF* {(protocol = UDP), (protected_network_direction = A), (destination_port > 1023), (string = /ce/63/d1/d2/16/e7/13/cf/39/a5/a5/86)} *THEN* {(name = INFO)};
- r_5 : *IF* {(protocol = TCP), (protected_network_direction = A), (destination_port = 53), (string = /00/00/ff)} *THEN* {(name = ANY-TCP)}.■



As we have mentioned before, we use hybrid approach to calculate rule similarity. Hence in RP-MES, rule similarity calculation contains two parts, structural relatedness and semantic relatedness, and each will be described as following.

(1) *Structural relatedness*

The structural relatedness considers the reference of attributes between rules, that is, evaluating the same attributes or asserting new values to the same attributes. When considering the reference of attributes, only the name of attribute is considered instead

of attribute value. In the definition of structural relatedness, two rules are related if there is any attribute used by both rules (either on left- or right-hand sides); otherwise they are independent. The structural relatedness between two rules is thus measured by the number of attributes that are mentioned in both rules. There are four situations of rule dependency, including *in-out*, *share-in*, *share-out*, and *not-shared*, and four corresponding functions, *inout()*, *sharein()*, *shareout()*, and *notshared()*, are given in Definition 4.2.

Definition 4.2. *inout()*, *sharein()*, *shareout()*, and *notshared()* Functions.

Given two rules $r_i = (CONDITIONS_i, ACTIONS_i)$, and $r_j = (CONDITIONS_j, ACTIONS_j)$, their definitions are defined as below:

- $inout(r_i, r_j)$: the set of attributes that are used in $CONDITIONS_i$ and $ACTIONS_j$, or $ACTIONS_i$ and $CONDITIONS_j$.
- $sharein(r_i, r_j)$: the set of attribute names that are common to both the $CONDITIONS_i$ and $CONDITIONS_j$.
- $shareout(r_i, r_j)$: the set of attribute names that are common to both the $ACTIONS_i$ and $ACTIONS_j$.
- $notshared(r_i, r_j)$: the set of all attributes used in r_i or r_j but not in $inout(r_i, r_j)$, $sharein(r_i, r_j)$, and $shareout(r_i, r_j)$.

Figure 5 illustrates those three relations mentioned in the Definition 4.2 The counts of attributes of the four sets, generated by *inout()*, *sharein()*, *shareout()*, and *notshared()* functions, are used to calculate the structural relatedness. And the weight of each counts is given as a variable in our rule similarity calculation, for example {1.0, 0.5, 0.75, -0.25}.

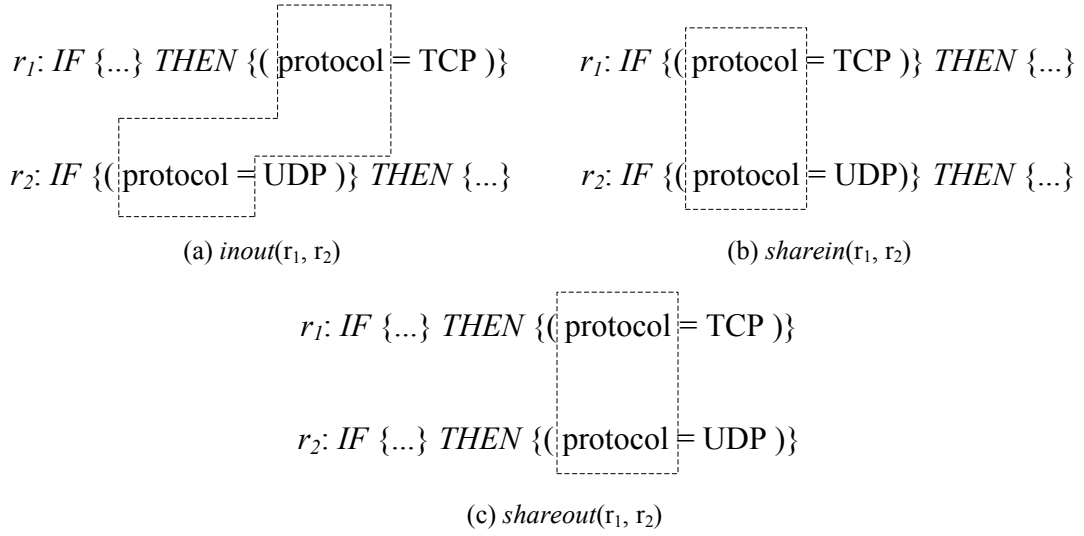


Figure 5. Illustration of three cases of attribute reference.

Example 4.2 Given the rules r_1 and r_2 listed in Example 4.1, the set of common attribute names in both conditions is $sharein(r_1, r_2) = \{\text{protocol, protected_network_direction, source_port, string}\}$, and the set of common attribute names in both actions is $shareout(r_1, r_2) = \{\text{name}\}$; while the $inout(r_1, r_2) = \{\emptyset\}$, and $notshared(r_1, r_2) = \{\emptyset\}$. However, for rules r_1 and r_3 , $sharein(r_1, r_3) = \{\text{protocol, protected_network_direction}\}$, $shareout(r_1, r_3) = \{\text{name}\}$, $inout(r_1, r_3) = \{\emptyset\}$, and $notshared(r_1, r_3) = \{\text{source_port, destination_port}\}$. ■

According to the number of attributes used in both two rules and given weights, the structural relatedness between two rules can be formulated in Definition 4.3.

Definition 4.3 Structural relatedness between two rules.

Given two rules $r_i = (CONDITIONS_i, ACTIONS_i)$, and $r_j = (CONDITIONS_j, ACTIONS_j)$, the structural relatedness between them is measured by $L(r_i, r_j)$ and defined as below:

$$L(r_i, r_j) = |inout(r_i, r_j)| \cdot w_{in-out} + |sharein(r_i, r_j)| \cdot w_{share-in} + |shareout(r_i, r_j)| \cdot w_{share-out} + |notshared(r_i, r_j)| \cdot w_{not-shared}$$

Example 4.3 According to the definition and the corresponding weights, w_{in-out} , $w_{share-in}$, $w_{share-out}$, and $w_{not-shared}$, are 1.0, 0.5, 0.75, and -0.25, respectively; the structural relatedness between r_1 and r_2 listed in Example 4.1 is

$$L(r_1, r_2) = 0 \cdot 1.0 + 4 \cdot 0.5 + 1 \cdot 0.75 - 0.25 \cdot 0 = 2.75 .$$

Also, the structural relatedness between r_1 and r_3 is

$$L(r_1, r_3) = 0 \cdot 1.0 + 2 \cdot 0.5 + 1 \cdot 0.75 - 2 \cdot 0.25 = 1.25 . \blacksquare$$

(2) Semantic Relatedness

In some cases, rules are very similar in syntactic structure, but they may be used to deal with different problems. Considering only structural relatedness between rules cannot effectively distinguish from them and further group related rules in the clusters. As for the rules, r_3 , r_4 , and r_5 , listed in Example 4.1, structural relatedness between every pair of rules is the same, that is, 2.75. Even though the rules are used to detect different network attacks, but with only structural relatedness, no additional information can help separate those rules. Therefore for calculating rule similarity, semantic relatedness is defined and used to complement structural relatedness.

When considering structural relatedness between rules, only the names of attributes are taken in to consideration instead of the values or the operators of attributes. In order to capture the semantic meaning between two rules based on the similarity of expressions of rules, attribute values and operators of the expressions are also considered. The expressions of rules can be divided into two categories, *categorical*

and *numerical* expressions, according to the data type of values. For a given expression, if its value is categorical data, it belongs to categorical expression, e.g., (protocol = TCP); otherwise, if its value is numerical data, it belongs to numerical expression, e.g., (destination_port > 1023). The semantic relatedness calculations of these two types of expressions are different in our definition.

Ontologies of knowledge-based system are often used for content explication or as common dictionary [1][40]. For two categorical expressions, the semantic relatedness can be measured by the conceptual similarity between their values. That is, for two categorical values, x and y , the semantic similarity of two categorical values can be measured by *conceptual similarity function* $s(x,y)$ [6]; the conceptual similarity function depends on both the distance between them in the ontology and their generality. However, some factual information or symbols may not be defined in the ontology; therefore the conceptual similarity function for this case is reduced to the Kronecker delta function by only determining whether the categorical values are the same or not. The definition of conceptual similarity function is given in Definition 4.4.

Definition 4.4 Conceptual Similarity Function.

Given two categorical values, x and y , which can be found on the ontology, the conceptual similarity between x and y is,

$$s(x, y) = \frac{c}{d(x, y) + \log_2(1 + D(x) + D(y))},$$

where $d(x, y)$ is the number of “hops” between x and y , $D(x)$ is the number of all its descendants, and c is the boundary constant. If x and y are not located on the ontology, their conceptual similarity is

$$s(x, y) = \delta(x, y) = \begin{cases} 1 & x = y \\ 0 & x \neq y \end{cases},$$

where $\delta(x, y)$ is Kronecker delta function.

However, different operators used may influence the evaluations of semantic relatedness between two expressions. For categorical expressions, both “=” and “≠” operators can be used, which means we can not just consider values when calculating the semantic relatedness of categorical expressions. There are two cases of operator combinations for two categorical expressions. Hence, the semantic relatedness between two categorical expressions is formulated as $\alpha()$ which is defined in Definition 4.5.

Definition 4.5 Semantic relatedness between two categorical expressions.

Given two expressions in rule action or condition, $e_m = (\text{attribute}_m \text{ operator}_m \text{ value}_m)$ and $e_n = (\text{attribute}_n \text{ operator}_n \text{ value}_n)$, which the attribute names are the same, i.e., $\text{attribute}_m = \text{attribute}_n$; the semantic relatedness $\alpha()$ between the expressions is formulated as below:

$$\alpha(e_m, e_n) = \begin{cases} s(\text{value}_m, \text{value}_n) & , \text{if } \text{operator}_m \neq \text{operator}_n \\ 1 - s(\text{value}_m, \text{value}_n) & , \text{otherwise} \end{cases}.$$

Example 4.4 Given the ontology which is illustrated in Figure 6. Suppose the constant c of conceptual similarity function is set to 0.9, the semantic relatedness between two expressions $e_1 = (\text{name} = \text{NETBUS})$ and $e_2 = (\text{name} = \text{NETBUS2})$ is

$$\begin{aligned} \alpha(e_1, e_2) &= s(\text{NETBUS}, \text{NETBUS2}) \\ &= \frac{c}{d(\text{NETBUS}, \text{NETBUS2}) + \log_2(D(\text{NETBUS}) + D(\text{NETBUS2}) + 1)} \cdot \blacksquare \\ &= \frac{0.9}{1 + \log_2(5 + 0 + 1)} = 0.25 \end{aligned}$$

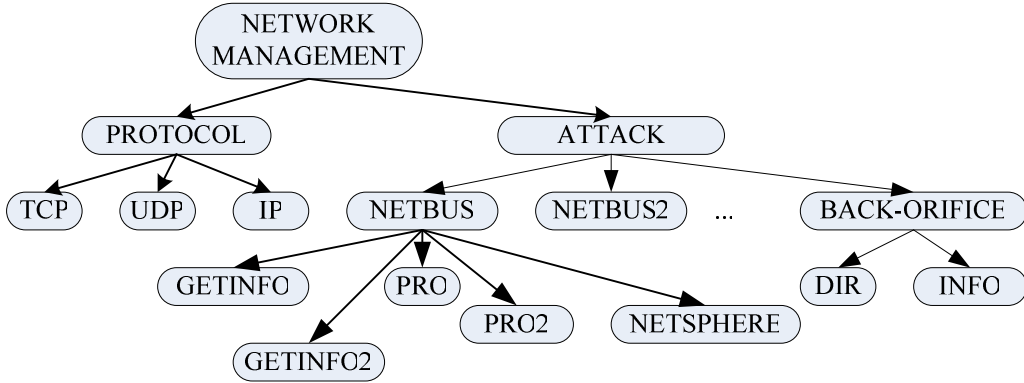


Figure 6. Part of the ontology in the network management domain.

On the other hand, the semantic relatedness calculation between two numerical expressions is different from that of categorical expressions. Before evaluating semantic relatedness between two numerical expressions, both numerical expressions must be transformed into mathematical intervals. For instance, expression $c_1 = (\text{port} > 1023)$ is transformed to $(1023, \max]$, where \max is the maximum value of V_{port} , which is the value range of “*port*”. The semantic relatedness between two numerical expressions is based on the overlapping of two mathematical intervals. The definition is given in Definition 4.6.

Definition 4.6 Semantic relatedness between two numerical expressions.

Given two expressions, $e_m = (\text{attribute}_m \text{ operator}_m \text{ value}_m)$ and $e_n = (\text{attribute}_n \text{ operator}_n \text{ value}_n)$, with the same attribute name, e_m , and e_n will be transformed to two intervals i_m and i_n , where $i_m \in V_{\text{attribute}_m}$ and $i_n \in V_{\text{attribute}_n}$. The semantic relatedness between the expressions is measured by $\beta(e_m, e_n)$ which is defined as:

$$\beta(e_m, e_n) = \frac{|i_m \cap i_n|}{|i_m \cup i_n|}.$$

Example 4.5 Given two numerical expressions $e_1 = (\text{port} > 1023)$ and $e_2 = (\text{port} > 8080)$, and the corresponding intervals are $i_1 = (1023, 65535]$ and $i_2 = (8080, 65535]$ respectively. The semantic relatedness between these two expressions is

$$\beta(e_1, e_2) = \frac{|(1023, 65535] \cap (8080, 65535]|}{|(1023, 65535] \cup (8080, 65535]|} = \frac{57455}{64512} = 0.89 \text{ .} \blacksquare$$

Based on the definition of the relatedness for numerical and categorical expressions, therefore, the semantic relatedness between any two expressions can be formally defined and given in Definition 4.7.

Definition 4.7 Semantic relatedness between two expressions.

Given two expressions, $e_m = (\text{attribute}_m \text{ operator}_m \text{ value}_m)$ and $e_n = (\text{attribute}_n \text{ operator}_n \text{ value}_n)$, the semantic relatedness between e_m and e_n is

$$S(e_m, e_n) = \begin{cases} \alpha(e_m, e_n) & , \text{ both } e_m \text{ and } e_n \text{ are categorical expressions} \\ \beta(e_m, e_n) & , \text{ both } e_m \text{ and } e_n \text{ are numerical expressions .} \\ 0 & , \text{ if } \text{attribute}_m \neq \text{attribute}_n \end{cases}$$

Based on the semantic and structural relatedness defined for expressions of rules, the definition of the similarity of two rules, r_i and r_j , is given in Definition 4.8.

Definition 4.8 Rule similarity between two rules.

Given two rules, $r_i = (\text{CONDITIONS}_i, \text{ACTIONS}_i)$, $r_j = (\text{CONDITIONS}_j, \text{ACTIONS}_j)$, the rule similarity between r_i and r_j is

$$\begin{aligned}
R(r_i, r_j) = & \sum_{e_m \in \text{CONDITIONS}_i, e_n \in \text{ACTIONS}_j, \text{attribute}_m, \text{attribute}_n \in \text{inout}(r_i, r_j)} S(e_m, e_n) \cdot w_{\text{in-out}} \\
+ & \sum_{e_m \in \text{ACTIONS}_i, e_n \in \text{CONDITIONS}_j, \text{attribute}_m, \text{attribute}_n \in \text{inout}(r_i, r_j)} S(e_m, e_n) \cdot w_{\text{in-out}} \\
+ & \sum_{e_m \in \text{CONDITIONS}_i, e_n \in \text{CONDITIONS}_j, \text{attribute}_m, \text{attribute}_n \in \text{sharein}(r_i, r_j)} S(e_m, e_n) \cdot w_{\text{share-in}} \\
+ & \sum_{e_m \in \text{ACTIONS}_i, e_n \in \text{ACTIONS}_j, \text{attribute}_m, \text{attribute}_n \in \text{shareout}(r_i, r_j)} S(e_m, e_n) \cdot w_{\text{share-out}} + \text{notshared}(e_m, e_n) \cdot w_{\text{not-shared}}
\end{aligned}$$

Example 4.6 Given two rules, r_1 and r_2 , listed in Example 4.1. Reviewing Example 4.2, $\text{sharein}(r_1, r_2) = \{\text{protocol, protected_network_direction, source_port, string}\}$, $\text{shareout}(r_1, r_2) = \{\text{name}\}$; $\text{inout}(r_1, r_2) = \{\emptyset\}$, and $\text{notshared}(r_1, r_2) = \{\emptyset\}$. The rule similarity between r_1 and r_2 is

$$R(r_1, r_2) = 0 \cdot 1.0 + (1 + 1 + 0.89 + 0) \cdot 0.5 + 0.25 \cdot 0.75 - 0 \cdot 0.25 = 1.6325 \text{ .}\blacksquare$$

4.2.2 Cluster Similarity



As mentioned before, Rule Base Partitioner iteratively merges the most similar rule clusters to construct the resulting rule clusters. Hence we need to define the similarity between two rule clusters. Besides, in order to avoid too small or too large rule clusters generated, the quantity of rules in rule cluster is also considered when calculating the clusters. The function of rule cluster similarity is given in Definition 4.9.

Definition 4.9 Rule cluster and cluster similarity function.

A rule cluster is a set of rules. Therefore, similarity between two rule clusters, g_s and g_t , is defined as:

$$CS(g_s, g_t) = \sum_{r_i \in g_s, r_j \in g_t} \text{sqrt}(R(r_i, r_j)) + \frac{2}{|g_s| + |g_t|}.$$

Given the set of rule clusters, the similarity between each pair of rule clusters can be calculated in advance and stored in the matrix, called *Cluster Similarity Matrix*.

Definition 4.10 Cluster Similarity Matrix.

The Cluster Similarity Matrix (CSM) is an m -by- m square matrix, where m is the number of rule clusters. Each entry n_{st} is the cluster similarity between rule clusters, g_s and g_t , that is, $CS(g_s, g_t)$. Since the definition of rule similarity is symmetric, the cluster similarity definition is also symmetric. Therefore, CSM is an upper triangular matrix.

Example 4.7 Suppose that there are three rule clusters $g_1 = \{r_1, r_2\}$, $g_2 = \{r_3, r_4\}$, and $g_3 = \{r_5\}$, where $r_1, r_2, r_3, r_4,$ and r_5 are listed in Example 4.1. The cluster similarity between g_1 and g_2 is

$$CS(g_1, g_2) = \text{sqrt}(R(r_1, r_3)) + \text{sqrt}(R(r_1, r_4)) + \text{sqrt}(R(r_2, r_3)) + \text{sqrt}(R(r_2, r_4)) + \frac{2}{2+2} = 0.83;$$

while $CS(g_1, g_3) = 0.86$ and $CS(g_2, g_3) = 1.08$. Therefore, the Cluster Similarity Matrix is

$$CSM = \begin{bmatrix} 0 & 0.83 & 0.86 \\ 0.83 & 0 & 1.08 \\ 0.86 & 1.08 & 0 \end{bmatrix} \blacksquare$$

4.2.3 Rule Base Partitioning Algorithm

Once the similarity for rule clusters can be calculated, the Rule Base Partitioner can use *Rule Base Partitioning Algorithm* to partition a rule base into rule clusters. The algorithm incorporated is a bottom-up approach, which derives a high-level structure for the rule base based on the information of rule similarity. Similar to most clustering algorithm, the algorithm runs iteratively to group rule clusters until the stopping

criterion is met. In this partitioning algorithm, the stopping criterion is to stop when the cluster similarities between all pairs of rule clusters are no longer larger than a *similarity threshold* (st) which is set by users.

The rule base partitioning algorithm is presented more formally below:

Algorithm: Rule Base Partitioning Algorithm

Input: A set of rules, similarity threshold st

Output: A set of rule clusters

Step 1. Group each rule as a single rule cluster.

Step 2. Calculate cluster similarity between each pair of rule clusters and generate corresponding Cluster Similarity Matrix.

Step 3. Choose the entry n_{ij} with the largest value (most similar) from the Cluster Similarity Matrix.

Step 4. Terminate and output the remaining rule clusters, if n_{ij} is less than or equal to st .

Step 5. Combine g_i and g_j into a single rule cluster by merging the rules inside the rule clusters.

Step 6. Go to Step 2.

Example 4.8 In this example, those rules listed in Example 4.1 are partitioned according to the Rule Base Partitioning Algorithm. The similarity threshold st is set to 2.

1. At first, each rule is assigned to a single rule cluster, i.e., $g_1 = \{r_1\}$, $g_2 = \{r_2\}$, $g_3 = \{r_3\}$, $g_4 = \{r_4\}$, and $g_5 = \{r_5\}$. The Cluster Similarity Matrix for this

configuration is

$$CSM_1 = \begin{bmatrix} 0 & 2.09 & 1.77 & 1.77 & 1.8 \\ 2.09 & 0 & 1.89 & 1.89 & 1.92 \\ 1.77 & 1.89 & 0 & 2.31 & 2.08 \\ 1.77 & 1.89 & 2.31 & 0 & 2.08 \\ 1.8 & 1.92 & 2.08 & 2.08 & 0 \end{bmatrix}.$$

2. By reviewing CSM_1 , n_{34} has the largest value of the matrix. Since n_{34} is larger than st , g_3 and g_4 are combined to form a new rule cluster. The remaining rule clusters are $g_1 = \{r_1\}$, $g_2 = \{r_2\}$, $g_3 = \{r_3, r_4\}$, and $g_4 = \{r_5\}$. The Cluster Similarity Matrix in this iteration is

$$CSM_2 = \begin{bmatrix} 0 & 2.09 & 0.77 & 1.80 \\ 2.09 & 0 & 0.89 & 1.92 \\ 0.77 & 0.89 & 0 & 1.08 \\ 1.80 & 1.92 & 1.08 & 0 \end{bmatrix}.$$

3. The largest value within the CSM_2 , n_{12} , can be discovered. Two rule clusters g_1 and g_2 are thus combined since n_{12} is larger than 2. After grouping g_1 and g_2 , the Cluster Similarity Matrix can be generated again, i.e.,

$$CSM_3 = \begin{bmatrix} 0 & 0.83 & 0.86 \\ 0.83 & 0 & 1.08 \\ 0.86 & 1.08 & 0 \end{bmatrix}.$$

4. No more grouping is needed because that all entries of CSM_3 are less than similarity threshold st . The process is terminated and output the result, set of the rule clusters, $\{\{r_1, r_2\}, \{r_3, r_4\}, \{r_5\}\}$. ■

4.3 Meta-rule Construction Process

The second process of Automatic Meta-rule Constructor is meta-rule construction which is used to extract meta-rules from the partitioned rule clusters by Meta-rule

Extractor. Meta-rule Extractor consists of two subcomponents, *Meta Apriori Algorithm* and *Confidence Calculator*. The Meta Apriori algorithm is modified from Apriori algorithm which is used widely to generate frequent large itemsets in data mining algorithms [1]. The Meta Apriori algorithm is used to generate the meta-rules, and Confidence Calculator calculates the confidence value of each meta-rule. The meta-rule generated by Meta-rule Extractor is then stored in the Meta-rule base for further usage. The whole process is illustrated in Figure 7.

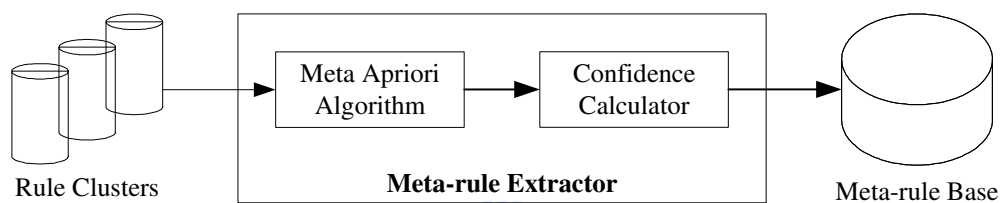


Figure 7. Components of Meta-rule Extractor.

In the following paragraphs, Meta Apriori algorithm is introduced first, and then the process of meta-rule generation will be examined.

4.3.1 Meta Apriori Algorithm

The Meta Apriori algorithm tries to discover the most frequent combinations of expressions to describe the rule cluster. The basic idea is that those most frequent combinations of expressions are used in many rules of the rule clusters, and once the combination is met, those rules may be related to the result. Different from Apriori algorithm, the transactions and itemsets defined in Meta Apriori algorithm are rule conditions and expressions. The notations used in Meta Apriori algorithm is given in Definition 4.11.

Definition 4.11 Transaction and itemset used in Meta-Apriori.

Given a rule cluster $g_i = \{r_{i1}, r_{i2}, \dots, r_{iN}\}$, where N is the number of rules in the rule cluster g_i , the transaction and itemset are defined below:

- $t_{ij} = \text{CONDITIONS}_{ij}$, where $\text{CONDITIONS}_{ij} \in r_{ij}, j \in [1 \dots N]$, is a set of expressions to be used as one transaction.
- d : the itemset of Meta Apriori algorithm is a set of expressions

Example 4.9 From Example 4.8, two rules, r_1 and r_2 , are grouped into the same rule cluster g_1 , i.e. $g_1 = \{r_1, r_2\}$. The corresponding transactions are

- $t_{11} = \{(\text{protocol} = \text{TCP}), (\text{protected_network_direction} = \text{A}), (\text{source_port} > 8080), (\text{string} = \text{NetBus})\}$
- $t_{12} = \{(\text{protocol} = \text{TCP}), (\text{protected_network_direction} = \text{A}), (\text{source_port} > 1023), (\text{string} = \text{NetBus2})\}$ ■



In Meta Apriori algorithm, the support count of the itemset is defined as the number of transactions that the itemset subsumes. That is, the set of expressions of the itemset subsume those of the transactions. Therefore, expression subsumption must be defined. For two expressions, e_m and e_n , e_m subsumes e_n if $\text{sub}(e_m, e_n) = 1$, where $\text{sub}()$ is called *expression subsume function*, which is defined in Definition 4.12.

Definition 4.12 Expression subsumption function.

Given two expressions $e_m = (\text{attribute}_m \text{ operator}_m \text{ value}_m)$ and $e_n = (\text{attribute}_n \text{ operator}_n \text{ value}_n)$,

$$sub(e_m, e_n) = \begin{cases} 1 & , \exists v \in V_{attribute_n}, v \in V_{attribute_m} \\ 0 & , \text{otherwise} \end{cases}.$$

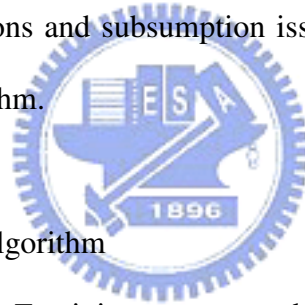
Moreover, an itemset subsumes the transaction if each expression of itemset subsumes at least one expression of transaction. The itemset subsumption is defined in Definition 4.13.

Definition 4.13 Itemset subsumption function.

Given an itemset d and a transaction t , the itemset subsumption function is defined as

$$subsume(d, t) = \begin{cases} 1 & , \forall e_m \in d \exists e_n \in t \mid sub(e_m, e_n) = 1 \\ 0 & , \text{otherwise} \end{cases}.$$

After discussing the notations and subsumption issue, it is appropriate to give the complete Meta Apriori algorithm.



Algorithm: Meta Apriori Algorithm

Input: A set of transactions, T ; minimum support threshold, min_sup .

Output: A set of frequent itemset, D

Step 1. Generate the set of frequent 1-itemsets, D_1 , by scanning T .

Step 2. Set initial value of k to 2.

Step 3. Generate candidate k -itemsets C_{ik} from $D_{i(k-1)}$.

Step 4. For each k -itemset $d_k \in C_{ik}$, compute the support count, that is, $d_k.support$

$$= \sum_{l=1}^N subsume(d_k, t_{il}).$$

Step 5. Remove those k -itemsets that their support counts are less than

$min_sup \cdot N$ from C_{ik} . The remaining itemsets are stored in D_{ik} .

Step 6. If $D_{ik} \neq \{\emptyset\}$, increase k by 1 and repeat step 2.

Step 7. Output D_{ik} .

Example 4.10 In the Example 4.8, rule base $RB = \{r_1, r_2, r_3, r_4, r_5\}$ is partitioned into three rule clusters, $g_1 = \{r_1, r_2\}$, $g_2 = \{r_3, r_4\}$, and $g_3 = \{r_5\}$. The following steps illustrate how to apply Meta Apriori algorithm on g_1 to generate frequent itemsets. The minimum support threshold, min_sup , is set to 0.9. For the sake of simplicity, every expression occurred in the RB is encoded in Table 1.

encoding	expression
e_1	(protocol = TCP)
e_2	(protected_network_direction = A)
e_3	(source_port > 8080)
e_4	(source_port > 1023)
e_5	(string = NetBus)
e_6	(string = NetBus2)

Table 1. Encodings of expressions in RB .

1. Since there are two rules in g_1 , T_1 consists of two transactions, t_{11} and t_{12} . Those are listed below:
 - $t_{11} = \{e_1, e_2, e_3, e_5\}$;
 - $t_{12} = \{e_1, e_2, e_4, e_6\}$.

2. In the first iteration, each expression in the transaction is a member of the set of candidate 1-itemsets, C_{11} . The algorithm scans all of the transactions in order to count the number of transactions that each itemset subsumes and summaries the result in Table 2.

itemset	count
$\{e_1\}$	2
$\{e_2\}$	2
$\{e_3\}$	1
$\{e_4\}$	2
$\{e_5\}$	1
$\{e_6\}$	1

Table 2. The support count of each candidate in C_{11} .

- The minimum support count required is 2 ($|T_1| \cdot \text{min_sup} = 2 \cdot 0.9 = 1.8$). The frequent 1-itemsets, D_{11} , can then be determined. Each itemset in D_{11} must satisfy minimum support count; the content of D_{11} is thus $\{e_1, e_2, e_4\}$.
- The candidate 2-itemsets C_{12} can be generated from D_{11} . To discover frequent 2-itemsets, D_{12} , the transactions in T_1 are scanned and the support count of each candidate is accumulated, as shown in the Table 3. Since support of each candidate is larger than or equal to minimum support count, the set of large 2-itemsets, D_{12} , is $\{\{e_1, e_2\}, \{e_1, e_4\}, \{e_2, e_4\}\}$.

itemset	count
(e_1, e_2)	2
(e_1, e_4)	2
(e_2, e_4)	2

Table 3. Support count of each candidate in C_{12} .

- The candidate 3-itemsets, C_{13} , can be generated in the same way, which is listed in Table 4. Since $|D_{13}| = 1$, no more candidate itemsets of C_{14} can be generated, and the process is thus terminated.

itemset	count
$\{e_1, e_2, e_4\}$	2

Table 4. Support count of candidate in C_{13} .

- Therefore, the final output is $D_{13} = \{\{e_1, e_2, e_4\}\}$. According to Table 1, the frequent combination of expressions is $\{(\text{protocol} = \text{TCP}),$

(protected_network_direction = A), (source_port > 1023)}. ■

4.3.2 Meta-rule Generation

The meta-rule generation is based on the concept of constraint-based association mining [29][31]. That is, the format of meta-rule is specified in advance. The definition of meta-rule is given by Definition 4.14. The meaning of the meta-rule, $mr_j = (CONDITIONS_j, (RULE_CLUSTER = g_i), conf)$, is that if all expressions of $CONDITIONS_j$ are satisfied, the rule cluster g_i will be selected with confidence value, $conf$.

Definition 4.14 Meta-rule representation.

For a given rule cluster g_i , each meta-rule, mr_j , generated from g_i has the form:

- $mr_j = (CONDITIONS_j, (RULE_CLUSTER = g_i), conf)$, where $CONDITIONS_j$ is a set of expressions in the condition part of mr_j , and $conf$ is the confidence value of the meta-rule.

The frequent sets of expressions generated from one rule cluster may be the same with those generated from the other rule clusters. Therefore, once the frequent itemsets of all rule clusters are generated by Meta Apriori algorithm, the next step is to determine the confidence value of each corresponding meta-rule. Within the Meta-rule Extractor, Confidence Calculator is used to calculate the confidence value of each meta-rule generated from frequent itemsets by applying Confidence Calculation Algorithm as below.

Algorithm: Confidence Calculation Algorithm

Input: a set of meta-rules, MRB , in which meta-rules have no confidence values defined

Output: a set of meta-rules, MRB' , in which confidence values from meta-rules specified

Step 1. At first, MRB' is an empty set.

Step 2. Choose the meta-rules from MRB which have the same set of expressions in their condition parts with the first meta-rule of MRB . The first meta-rule of MRB is included in selection result.

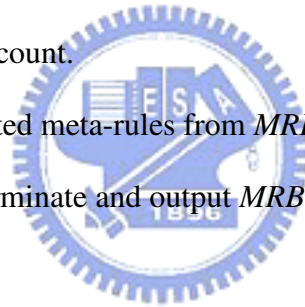
Step 3. Accumulate the total support count of all selected meta-rules.

Step 4. Set the confidence value of each selected meta-rule via dividing its support count by total support count.

Step 5. Remove those selected meta-rules from MRB to MRB' .

Step 6. If MRB is empty, terminate and output MRB' .

Step 7. Go to step 2.



Chapter 5. Experiment

In this chapter, an Intrusion Detection System (IDS) prototype based on RP-MES is proposed, and the partitioning result as well as performance analysis of the prototype system are also introduced.

5.1 Experiment Environment

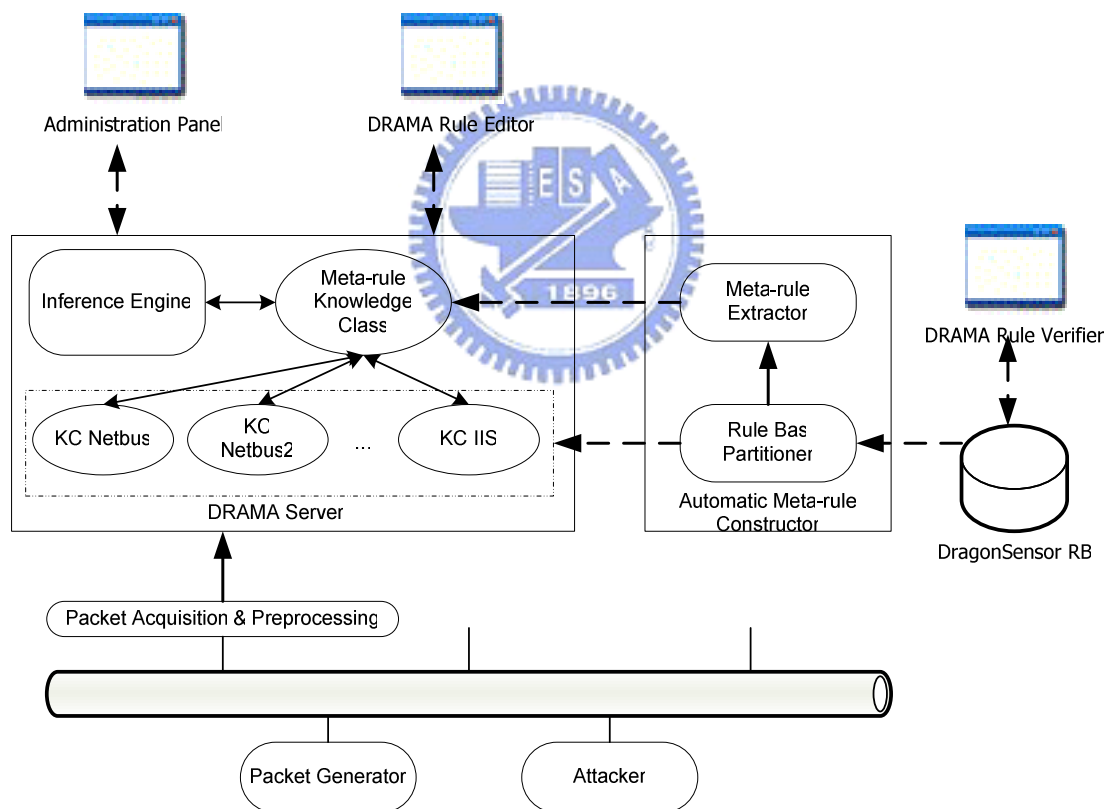


Figure 8. An IDS based on RP-MES.

Figure 8 illustrates the system architecture of the IDS prototype based on RP-MES. There are two major components in the prototype systems, including the DRAMA

System and Automatic Meta-rule Constructor. DRAMA is developed by Knowledge and Data Engineering Laboratory (KDE Lab.) of National Chiao Tung University, Taiwan. DRAMA is implemented using JAVA. In the prototype system, DRAMA is used to store, represent, process the knowledge of the IDS. Moreover, the NORM knowledge model used in DRAMA provides the ability of inferring meta-rules and rule selection by the ACQUIRE knowledge relation, which is a dynamic knowledge relation to include rule cluster only when meta-rule is matched. DRAMA Rule Verifier is used to verify rules within the knowledge classes. DRAMA Rule Editor provides a user friendly GUI interface for editing knowledge classes and rules.

Automatic Meta-rule Constructor is also implemented in JAVA. The knowledge can be the rule base of any other IDS, e.g., Snort [36], Dragon Sensor [9], etc. Automatic Meta-rule Constructor can partition the rule base into rule clusters, which are represented as knowledge classes in DRAMA, and generate meta-rules from those knowledge classes. Those generated knowledge classes can be processed by DRAMA Server for detecting network intrusions.

Packet Preprocessor is implemented in C, which collects the packets from network and translates into the facts for DRAMA to infer. Administration Panel of the prototype system provides a user interface for administrator to monitor the situation of network environment.

5.2 Experimental Results

The rule base of Dragon Sensor consists of 646 rules, which is used as our

experimental knowledge source. In the first experiment, various numbers of rules are partitioned by Automatic Meta-rule Constructor according to different similarity threshold (st) settings, which are used as the criteria to stop clustering process. The partitioning result is shown in Table 5, and Figure 9. It can be observed that similarity threshold = 1.2 produces more reasonable number of rule clusters since the average size of rule clusters is satisfied with Miller's magic number [25]. The Miller's magic number says that human beings have a meaningful chunking size up to 7 ± 2 .

rules \ st	1.0	1.1	1.2	1.3	1.4	1.5
100	2	3	20	29	41	50
200	3	5	28	48	87	100
300	3	5	32	82	136	150
400	3	7	52	112	181	200
500	3	5	57	139	225	246
600	4	7	56	172	272	296
646	3	6	59	188	296	324

Table 5. The number of clusters for different similarity threshold settings and number of rules.

According to the partitioning results obtained from the first experiment, the second experiment is conducted to analyze the accuracy. For each partitioning result in first experiment, Automatic Meta-rule Constructor also generates corresponding meta-rules and stores those meta-rules in a knowledge class. Those knowledge classes are loaded into the IDS prototype system for accuracy experiment. The accuracy is obtained by comparing the result of original rule base (Dragon Sensor) and partitioned rule base, which is calculated in following formula:

$$accuracy = \frac{|F_p|}{|F_{DS}|},$$

where F_p is the set of rules fired in partitioned rule base, and F_{DS} is the set of rules

fired in original rule base. The result is shown in Table 6.

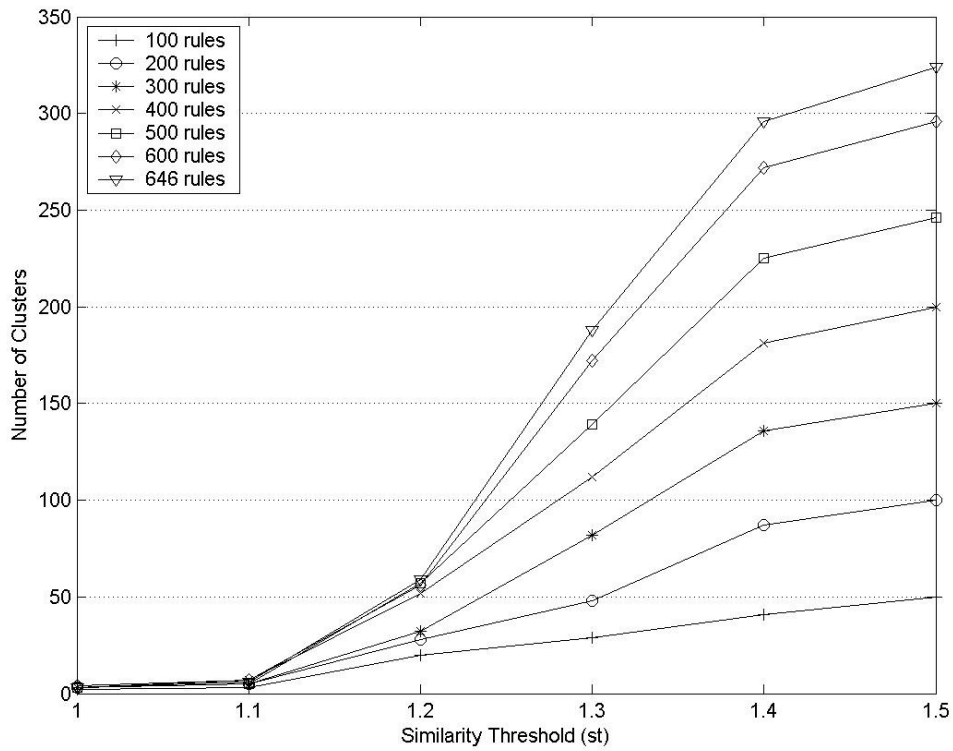


Figure 9. Partitioning results by different similarity threshold settings.

rules \ st	1.0	1.1	1.2	1.3	1.4	1.5
100	95%	97%	100%	100%	100%	100%
200	98%	99%	100%	100%	100%	100%
300	98.33%	100%	100%	100%	100%	100%
400	96.25%	96.75%	98.75%	99.25%	99.25%	99.25%
500	94.20%	94.20%	97%	97.60%	97.60%	97.60%
600	95.33%	95.67%	97.33%	98.00%	98.00%	98.00%
646	95.51%	95.82%	97.37%	98.14%	98.14%	98.14%

Table 6. Accuracy comparisons of partitioned rule base.

We also use the traditional rule base partitioning approach, which only considers structural relatedness of rule, to partition the same rule base used in previous

experiments. The result is listed in Table 7 and illustrated in Figure 10. Table 7 shows the number of rule clusters partitioned for both approaches to meet the same accuracy (the accuracy for RP-MES approach with $st = 1.2$).

rules	100	200	300	400	500	600	646
RP-MES	20	28	32	52	57	56	59
TRADITIONAL	30	51	85	113	139	170	188

Table 7. Comparison of number of clusters under the same accuracy.

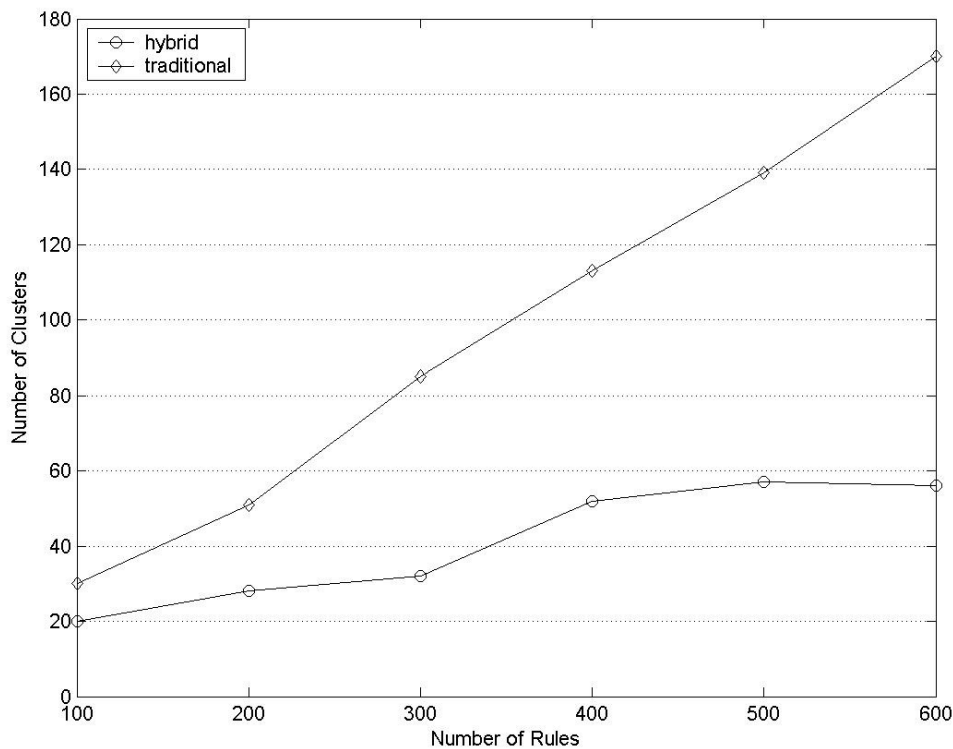


Figure 10. Comparison of number of clusters under the same accuracy.

The final experiment is to evaluate the performance between the rule base not partitioned and the rule base partitioned by RP-MES. The result of the experiment is shown in Figure 11. When the number of rules is small, the performance difference is not obvious. But if the number of rules becomes large, the execution time of original

rule base increases greatly. However, the execution time of RP-MES increases more smoothly than that of original rule base.

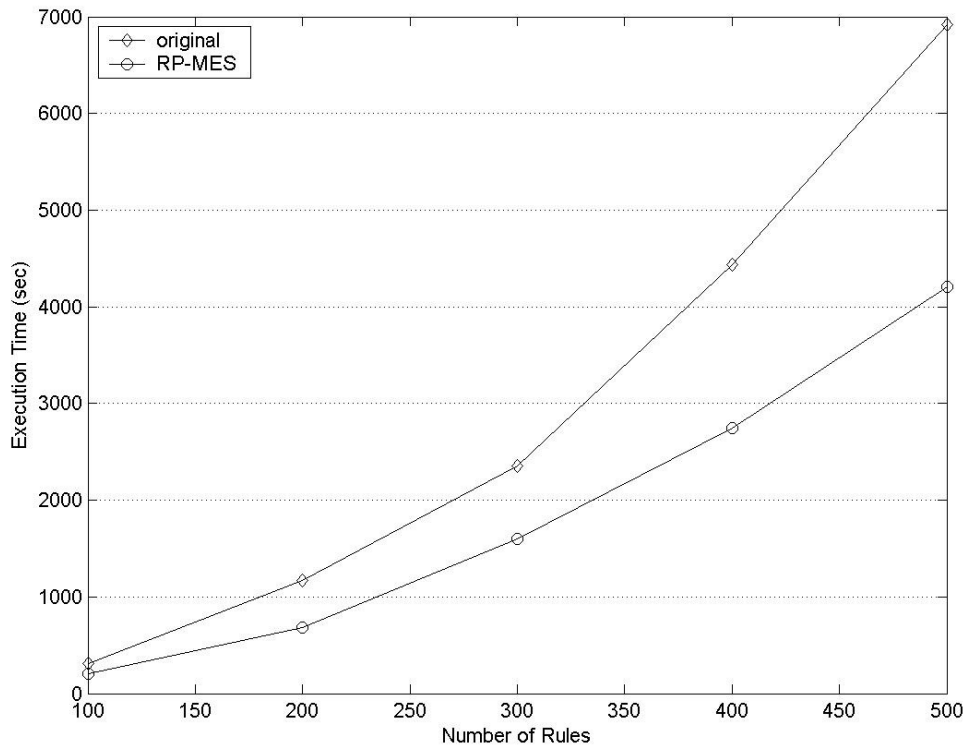


Figure 11. The performance of original rule base and RP-MES.

Chapter 6. Conclusion

Due to the growth of rule base usage, the scale of rule base is increasing and hence many management related issues arise about constructing the rule base. Three are issues about rule base maintenance, including complex relationships between rules, structural errors, and performance degradation. In this thesis, RP-MES is proposed to solve these issues by designing a new approach combining both rule base partitioning mechanism and meta-rule construction mechanism. As for rule base partitioning, RP-MES not only takes care of the structural relatedness between rules, but also considers the semantic relatedness of rules by calculating the semantic relationship between rules in the rule base. On the other hand, RP-MES extracts the meta-rules from the rule clusters partitioned by the rule base partitioning mechanism, and uses the obtained meta-rules for selecting rule cluster when using the rule base.

In order to evaluate the performance of RP-MES, an Intrusion Detection System (IDS) prototype is also designed and implemented based on RP-MES, and some experiments have been done to test the system performance. The experimental results show that RP-MES can produce reasonable number of rule clusters and the accuracy of the inference result remains, and that the performance of RP-MES is better than that of original rule base without partitioning.

Reference

- [1] R. Agrawal and R. Srikant, "Fast algorithms for mining association rules," in *Proceedings of International Conference on Very Large Data Bases (VLDB'94)*, pp. 487-499, 1994.
- [2] C. Baral, S. Kraus, and J. Minker, "Combining multiple knowledge bases," *IEEE Transaction on Knowledge and Data Engineering*, vol. 3, pp. 208-220, 1991.
- [3] R. Bogacz and C. Giraud-Carrier, "Learning meta-rules of selection in expert systems," in: *Proceedings of the Fourth World Congress on Expert Systems*, pp. 576-581, 1998.
- [4] N. Botten, A. Kusiak and T. Raz, "Knowledge bases: Integration, verification, and partitioning," *European Journal of Operational Research*, vol. 42, pp. 111-128, 25 September. 1989.
- [5] C.L. Chang, J.B. Combs, and R.A. Stachowitz, "A report on the expert systems validation associate (EVA)," *Expert Systems with Applications*, vol. 1, pp. 217-230, 1990.
- [6] W.W. Chu, Z. Liu, and W. Mao, "Textual document indexing and retrieval via knowledge sources and data mining," *Communication of the Institute of Information and Computing Machinery*, vol. 5, 2002.
- [7] R. Davis, "Meta-rules: Reasoning about control," *Artificial Intelligence*, vol. 15, pp. 179-222, 1980.
- [8] K. Dev and C. S. R. Murthy, "A genetic algorithm for the knowledge base partitioning problem," *Pattern Recognition Letters*, vol. 16, pp. 873-879, Aug.

1995.

- [9] Dragon Sensor, Enterasys Networks, Inc. <http://www.enterasys.com>.
- [10] J. Durkin, "Expert systems: Design and development," Macmillan Publishing Company, pp. 600-684, 1994.
- [11] P. Dutta, "Evolutionary heuristic for knowledge base partitioning problem," in *Proceedings of the IEEE International Conference on Evolutionary Computation, ICEC'97*, 1997.
- [12] J.R. Geissaman, "Verification and validation for expert systems: A practical methodology," in *Proceedings of the 4th Annual Artificial Intelligence and Advanced Computer Technology Conference*, 1998.
- [13] J. Giarratano, *Expert Systems: Principles and Programming*, Brooks Cole, 1998.
- [14] M.T. Harandi, "Rule base management using meta knowledge," in *Proceedings of the 1986 ACM SIGMOD international conference on Management of data*, pp. 261-267, 1986.
- [15] X. He, W.C. Chu, H. Yang, and S.J.H. Yang, "A new approach to verify rule-based systems using petri nets," *Information and Software Technology*, vol. 45, pp. 663-669, 2003.
- [16] K. Higa, "An approach to improving the maintainability of existing rule bases," *Decision Support Systems*, vol. 18, pp. 23-31, 1996.
- [17] S.Y. Ho, H.M. Chen, and L.S. Shu, "Solving large knowledge base partitioning problems using an intelligent genetic algorithm," in *Proceedings of GECCO-99: the Genetic and Evolutionary Computation Conference*, pp. 1567-1672, 1999.
- [18] G. J. Hwang and S. S. Tseng, "EMCUD: A knowledge acquisition method which

captures embedded meanings under uncertainty,” *International Journal of Human-Computer Studies*, vol. 33, pp. 431-451, 1990.

- [19] R.J.K. Jacob and J.N. Froscher, "Software engineering for rule-based systems," in *Proceedings of 1986 fall joint computer conference on Fall joint computer conference*, pp. 185-189, 1986.
- [20] R.J.K. Jacob and J.N. Froscher, "A software engineering methodology for rule-based systems," *IEEE Transactions on Knowledge and Data Engineering*, vol. 2, pp. 173-189, 1990.
- [21] T.T. Kuo, S.S. Tseng, and Y.T. Lin, "Ontology-based knowledge fusion framework using graph partitioning," *IEA/AIE*, pp. 11-20, 2003.
- [22] O. Lee and P. Gray, "Knowledge base clustering for KBS maintenance," *Journal of Software Maintenance: Research and Practice*, vol. 10, pp. 395-414, 1998.
- [23] Y.T. Lin, S.S. Tseng, and C.F. Tsai, "Design and implementation of new object-oriented rule base management system," *Expert Systems with Applications*, vol. 25, pp. 369-385, 2003.
- [24] E. McCreath, "Extraction of meta-knowledge to restrict the hypothesis space for ILP systems," *Eight Australian Joint Conference on Artificial Intelligence*, pp. 75-82, 1995.
- [25] G. Miller, "The magical number seven, plus or minus two: some limits on our capacity for processing information," *The Psychological Review*, vol. 63, pp. 81-97, 1956.
- [26] G. Myers, *Reliable Software Through Composite Design*, Petrocelli/Charter, New York, 1975.

- [27] D.L. Nazareth, "Issues in the verification of knowledge in rule-based systems," *International Journal of Man-Machine Studies*, vol. 30, pp. 255-271, 1989.
- [28] D.L. Nazareth and M.H. Kennedy, "Verification of rule-based knowledge using directed graphs," *Knowledge Acquisit*, vol. 3, pp. 339-360, 1991.
- [29] R. Ng, L.V.S. Lakshmanan, J. Han, and A. Pang, "Exploratory mining and pruning optimizations of constrained associations rules," in *Proceedings of International Conference on Management of Data (SIGMOD'98)*, pp. 13-24, 1998.
- [30] T.A. Nguyen, W.A. Perkins, T.J. Laffey, and D. Pecora, "Knowledge base verification," *AI Magazine*, 1987.
- [31] J. Pei and J. Han, "Can we push more constraints into frequent pattern mining?" in *Proceedings of International Conference on Knowledge Discovery and Data Mining (KDD'00)*, 2000.
- [32] M. Ramaswamy, S. Sarkar, Y.S. Chen, "Using directed hypergraphs to verify rule-based expert systems," *IEEE Transaction on Knowledge Data Engineering*, vol 9, pp. 221-237, 1997.
- [33] T. Raz and N. Botten, "The knowledge base partitioning problem: Mathematical formulation and heuristic clustering," *Data & Knowledge Engineering*, vol. 8, pp. 329-337, 1998.
- [34] S. Sarkar and M. Ramaswamy, "Knowledge base decomposition to facilitate verification," *Information Systems Research*, vol. 11, No. 3, pp. 260-283, Sep. 2000.
- [35] U.J. Schild and S. Herzog, "The use of meta-rules in rule based legal computer systems," in *Proceedings of the fourth international conference on Artificial*

intelligence and law, pp. 100-109, 1993.

[36] Snort, <http://www.snort.org>.

[37] L.E. Stanfel, “Applications of clustering to information system design,” *Information Processing & Management*, vol. 19, pp. 37-50, 1983.

[38] S. Tsumoto and S. Hirano, “Visualization of rule’s similarity using multidimensional scaling,” in *Proceedings of the Third IEEE International Conference on Data Mining (ICDM’03)*, 2003.

[39] G. Valiente, “Verification of knowledge base redundancy and subsumption using graph transformations,” *International Journal of Expert Systems: Research and Applications*, vol. 6, pp. 341-355, 1993.

[40] H. Wache, T. Vgele, U. Visser, H. Stuckenschmidt, G. Schuster, H. Neumann, and S. Hbner, “Ontology-based integration of information – a survey of existing approaches,” in *Proceedings of the Workshop Ontologies and Information Sharing (IJCAI’01)*, pp. 108-117, 2001.

[41] S.J.H. Yang, A.S. Lee, W.C. Chu, and H. Yang, “Rule base verification using Petri nets,” in *Proc. 22-nd Annual Int. Computer Software and Applications Conf. (COMPSAC-98)*, 1998.