Intelligent Query Approach in IDS Audit Database

Hongyu Yang [1,2,3] and Lixia Xie[1]

[1] Software Technology Research Center, Civil Aviation University of China, 300300, Tianjin, China
Yhyxlx@hotmail.com, Yang_hy@eyou.com
[2] Tianjin Key Lab for Advanced Signal Processing, Civil Aviation University of China, 300300, Tianjin, China
[3] Department of Computer Science and Engineering, Tianjin University, 300072, Tianjin, China

Abstract

With the development of internet, the audit work of IDS is becoming harder. The way of examining log file in text format cannot adapt to the serious situation. Natural Language Process (NLP) technology is a novel approach to solve this problem. In this paper, Functional Unification Grammar (FUG) in NLP was applied to intelligent query in IDS audit system, and XML schema was also employed in expression of accidence, syntax, vocabulary library and grammar. We utilized feature structure to describe the structure of vocabulary, phrase and sentence. SQL query object tree could be translated into SQL sentence easily with translation algorithm. We also took some steps to make the query fuzzy so as to give the query system more intelligence.

Keywords: Intrusion detection System, FUG, NLP, XML, query.

1 Introduction

Information security is an issue of serious global concern. The audit facility of IDS can monitor and keep many records of malicious behaviors to networks and single machines. The more malicious behaviors increase the more complicated and larger the audit records become. The burden of administrator of Intrusion Detection System (IDS) is become heavier. We need a convenient and compact way which could overcome this problem.

LUNAR was a natural language interface to a database, it was capable of translating elaborate natural language expressions into database queries. SHRDLU was a system capable of participating in a dialogue about a microworld (the blocks world) and manipulating this world according to commands issued in English by the user. LUNAR and SHRDLU both exploited the limitations of the domain to make the natural language understanding problem tractable. For instance, disambiguation, compound noun analysis, quantifier scope, pronoun reference.

Unification-based formalisms are increasingly used in linguistic theories and computational linguistics. In particular, one type of unification formalism, functional

unification grammar (FUG) [1], [4], [6] is widely used for text generation and is beginning to be used for parsing. FUG enjoys such popularity mainly because it allies expressiveness with a simple economical formalism. It uses very few primitives, has a clean semantics, is monotonic, and grants equal status to function and structure in the descriptions.

So, NLP is significant to the audit database system and log retrieval in IDSs. Natural language queries (corresponding to a description, or narrative) can be transformed into a well-documented knowledge and meaning representation through NLP approach which can answer the questions put forward by administrators related to the contents of the audit system database. An intelligent query supported by Natural Language Process (NLP) can help administrators greatly. Therefore, administrators don't have to search the database manually. By this means, IDSs can operate the audit database more effectively.

The use of FUG may improve the analysis ability of the Generalized Phrase Structure Grammar, and restricts its generating ability. It also combines with audit system database closely. Feature structure [2], [4], [7] unifies the process of description of vocabulary, phrase, sentence and grammar. XML [3] schema provides a standardized approach to express the accidence, syntax, glossary library and grammar. It can make the query system extend its ability easily. The introduction of NLP technology will eventually realize the target that administrators interact with audit system freely.

This paper includes three sections: section two introduces FUG and Unification Operation; section three gives descriptions of two XML schemas, one describes the vocabulary and the other describes the all kinds of rule predefined; section four describes the SQL query object tree and an algorithm of translation from object tree to SQL query sentence.

2 Functional Unification Grammar

Functional Unification Grammar (FUG) provides an opportunity to encompass within one formalism and computational system the parts of machine translation systems that have usually been treated separately. FUGs are popular for natural language applications because the formalism uses very few primitives and is uniform and expressive [8].

The FUG formalism has rapidly gained acceptance in the field of text generation. FUG traffics in descriptions and there is essentially only one kind of description, whether for lexical items, phrases, sentences, or entire languages. Descriptions do not distinguish among levels in the linguistic hierarchy. This is not to say that the distinctions among the levels are unreal or that a linguist working with the formalism would not respect them. It means only that the notation and its interpretation are always uniform. Either a pair of descriptions is incompatible or they are combinable into a single description.

Within FUG, every object has infinitely many descriptions, though a given grammar partitions the descriptions of the words and phrases in its language into a finite number of equivalence classes, one for each interpretation that the grammar assigns to it. The members of an equivalence class differ along dimensions that are gram-

matically irrelevant--when they were uttered. Each equivalence class constitutes a lattice with just one member that contains none of these grammatically irrelevant properties, and this canonical member is the only one a linguist would normally concern himself with. However, a grammatical irrelevancy that acquires relevance in the present context is the description of possible translations of a word or phrase, or of one of its interpretations, in one or more other languages.

A description is an expression over an essentially arbitrary basic vocabulary. The relations among sets of descriptions therefore remain unchanged under one-for-one mappings of their basic vocabularies. It is therefore possible to arrange that different grammars share no terms except for possible quotations from the languages described.

2.1 Feature Structure

A complex feature set is expressed with Functional Description (FD) in FUG. A FD is composed of a group of descriptors that is a constituent set, a pattern or an attribute with a certain value. Attributes are arbitrary words with no significant internal structure. Values can be of various types, the simplest of which is an atomic value. The most important part is the pair of attribute and value. Descriptor in FD is either an atom or another FD. So the definition of the FD is iterative. The strict definition of FD is given as follows:

Let α a FD, iff α can be expressed as:

$$\begin{bmatrix} f_1 = v_1 \\ \vdots \\ f_n = v_n \end{bmatrix} n \geq 1 \qquad (1)$$

f_i denotes feature name, and v_i denotes feature value, and they should satisfy:

(1) Feature name f_i is an atom, and feature value v_i is either an atom or another FD.

(2) $\alpha(f_i) = v_i (1 \cdots n)$, Which is read as the value of feature f_i is v_i in sets α.

2.2 Unification Operation

Unification operation is used to combine some FD into a single FD. For example, if two or more FD was compatible, they could be combined into a single FD by unification operation by which the object described is the common object described by them. This operation is similar to the union operation in set theory. In set theory atoms in set are always regarded as indiscerptible atom even if the atoms are order pairs without regard to their inner structures. Unification operation must consider the rationality of operation results. If they include inimical information the result set will be empty (Denoted as ϕ). The formalized definition of unification operation is given as follows:

Definition 1. unification operation ($\overline{U}$)

(1) If both a and b are atoms, then

$$a \overline{\cup} b = \left\{ \begin{array}{l} a, (a = b) \\ \phi, (a \neq b) \end{array} \right\}. \quad (2)$$

(2) If both α and β are a complex feature set,

① If $\alpha(f) = v$, but the value of $\beta(f)$ is not defined, then $(f = v) \in \alpha \overline{\cup} \beta$.

② If $\beta(f) = v$, but the value of $\alpha(f)$ is not defined, then $(f = v) \in \alpha \overline{\cup} \beta$.

③ If $\alpha(f)=v_1, \beta(f)=v_2$, v_1 isn't inimical to v_2, then $(f = (v_1 \overline{\cup} v_2)) \in \alpha \overline{\cup} \beta$, else $\alpha \overline{\cup} \beta = \phi$.

There are two roles about unification operation: one is to combine original feature information and construct new feature structures, the other is to examine compatibleness of features and precondition of rules. Unification operation is shown in fig. 1.

$$\text{at:}\begin{bmatrix} PRED = at \\ CAT = \begin{bmatrix} CATEGORY = p \\ SUBCAT = pp1 \end{bmatrix} \\ OBJT = \begin{bmatrix} CAT = \begin{bmatrix} CATEGORY = s \\ SUBCAT = sss \end{bmatrix} \end{bmatrix} \end{bmatrix}$$

$$\text{home:}\begin{bmatrix} PRED = \text{home} \\ CAT = \begin{bmatrix} CATEGORY = s \\ SUBCAT = sss \end{bmatrix} \end{bmatrix}$$

$$\text{at home:}\begin{bmatrix} PRED = at \\ CAT = \begin{bmatrix} CATEGORY = p \\ SUBCAT = pp1 \end{bmatrix} \\ OBJT = \begin{bmatrix} PRED = \text{home} \\ CAT = \begin{bmatrix} CATEGORY = s \\ SUBCAT = sss \end{bmatrix} \end{bmatrix} \end{bmatrix}$$

Fig. 1. Unification operation between "at" and "home"

2.3 Functional Description of Sentence

The most important feature of FUG is that it use complex feature set systematically and roundly in the description of word definition, syntax rule, semantic rule and sentence structure. We only give an example to show the functional description of syntax rule, and other descriptions are similar. Syntax rule provides sentence structure with a pattern that regulates the restrictive conditions in grammar and semantics. Fig.2 shows the functional description of passive voice and active voice.

The value of feature pattern in fig.2 is in the sequence that regulates the order of components in sentence structure with active voice or passive voice. FUG fits for the generation of sentence directly. This process can begin with a simple description, and

then combine with the functional description of grammar, and do some unification operation, and in the end the complete sentence structure is got.

$$FD(2):\begin{bmatrix} CAT = S \\ PATTERN = (\ldots PREDICATE\ DIRECT\_OBJECT \ldots) \\ SUBJ = ACTOR = [CAT = NP] \\ PREDCATOR = \begin{bmatrix} CAT = VERB \\ TRANSITIV = MATER\_PROC \\ VOICE = ACTIVE \end{bmatrix} \\ VOICE = ACTIVE \end{bmatrix} \quad FD(3):\begin{bmatrix} CAT = S \\ PATTERN = (PREDICATOR \ldots BY \ldots ADJUNCT) \\ SUBJ = [CAT = NP] \\ PREDICATOR = \begin{bmatrix} CAT = VERB \\ TRANSITIV = MATER\_PROC \\ VOICE = PASSIVE \end{bmatrix} \\ BY\_ADJUNCT = \begin{bmatrix} CAT = PP \\ PREP = \begin{bmatrix} CAT = PREPOSITION \\ LEX = 'by' \end{bmatrix} \\ OBJECT = < AGENT > \end{bmatrix} \\ VOICE = ACTIVE \end{bmatrix}$$

Fig. 2. Functional description of active and passive voice

3 XML Description

XML indexing is the key to the efficiency of XML based query processing, especially to present audit database systems in IDSs. The semi-structured nature of XML schema and the flexible mechanisms of XML schema introduce new challenges to the existing database indexing methods in audit data querying.

An XML schema is a description of a type of XML document, typically expressed in terms of constraints on the structure and content of documents of that type, above and beyond the basic syntax constraints imposed by XML itself. An XML schema provides a view of the document type at a relatively high level of abstraction.

An XML schema also can be viewed as a directed graph T = (V, E), where V is the set of vertices and E is the set of edges. The vertices correspond to the types of elements and attributes and the edges represent containment (parent-child) relationships. The vertices are labeled with the name of the type and the edges are labeled with the name of the element or attribute. The edges have an additional multiplicity label that can take a value from wildcard character set.

We design two XML schemas, one described the vocabulary library (include word, phrase, and some symbols, etc.), and the other schema described the all kinds of rule predefined by us. We also designed the schema abortively, which could be reused by system users. They can add their own vocabulary library as long as they conform to the uniform description. Especially, the rule schema is the description of meta-rule of grammar rules. All XML instances conform to these two schemas can be introduced into the system to extend the intellect of system. The structures of the two schemas are shown in fig. 3 and fig. 4 respectively.

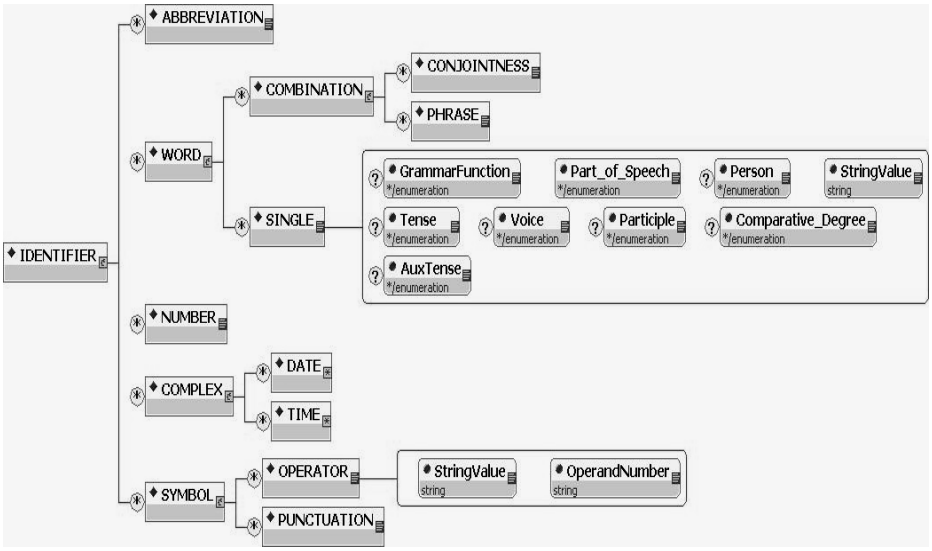

Fig. 3. Schema structure of vocabulary library

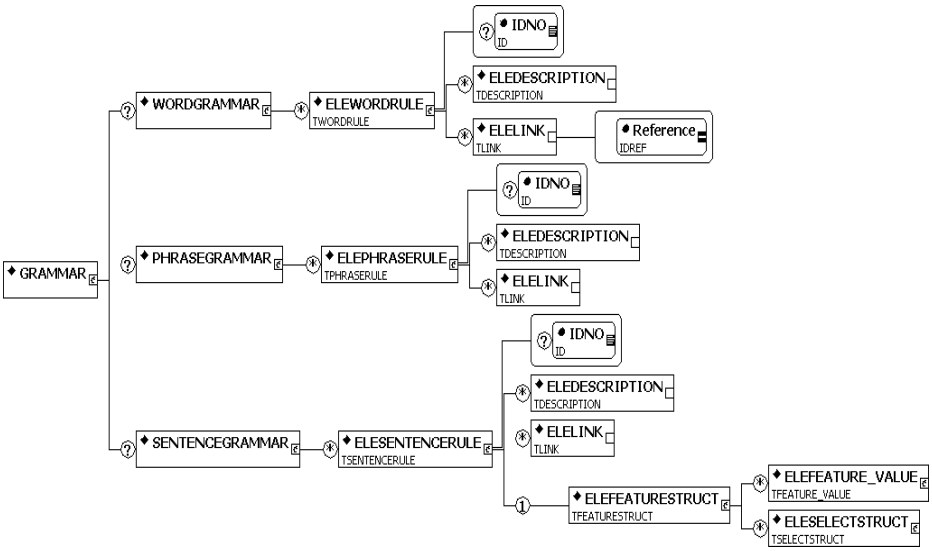

Fig. 4. Schema structure of rule library

4 SQL Query Object Tree

4.1 Query Structure and Classification in Query Target and Condition

Natural language query in audit database is restricted in vocabulary, sentence structure, semantics and usage. In our experiments, we find that optative sentence and question are two most frequent sentence structure used by users of query system. Thus we find the possible sentence structures in query sentence:

(1) Two are about optative sentence:

[1] <Condition1 Condition2…Conditionm>
< Target1 Target2…Targetn> .

[2] (<Condition1 Condition2…Conditionm1>
< Target1>) <conjunction> (<Condition1 Condition2 … Conditionm> <Target2>)…(<Condition1 Condition2…Conditionmn>
<Target n>) .

(2) Two are about judgement question:

[1] (<Condition1, Condition2…Conditionm1>
<Target1>) <Verb> (<Condition1 Condition2 … Conditionm2> <Target2>).For instance, “has the SQL Server TJU_3 been scan in recent three days?”

[2] (<Condition1 Condition2…Conditionm1>) <linkverb> (<Condition1 Condition2…Conditionm2> <Target2><Target1>) . For instance, “was the scan from IP address 202.113.18.45?”

(3) One is about common question:

< Target1 Target2…Targetn> <Condition1 Condition2…Conditionm> .

A query target is classified into three categories: definitive target (given by database object name), question target (given by question word), and collective target (object with a collective function). A query condition is classified into five categories as follows:

- Value condition: For instance, it’s a value of some attribute in the database. For instance, “search the IP address of SERVER TJU_3.” “TJU_3” is the value of SERVER.name, so it is a value condition.
- Table name and attribute name in database.
- Collective condition: for instance, in “search the frequentest IP address which scan 202.113.18.64”, “frequentest” is a collective condition.
- SQL condition: this kind of conditions is similar to WHERE sub clause in SQL.
- Domanial verb condition: the condition occurred includes some domanial verbs.

4.2 Structure of SQL Object Tree

SQL Object Tree is comprised of four kinds of nodes:

- Value node: it corresponds to a description of a specific attribute value and determines a select condition.
- Object node: it corresponds to an object in the database such as table name and column name.
- Relationship node: it corresponds to a relationship in database, and determines a relationship condition.
- Connection node: it corresponds conjunct word (and, or) or comparative word (less then, more than etc.).

Value node gives the restraints, while restrict object should be given by object node. Similarly, relationship nodes must occur with value nodes or object to form a definite semantics. Just because of this, a few of nodes depend on each other to form a clear operational semantics in database. So we give out the formalized definition of Set Chunk as follows.

Definition 2. Set Chunk is a cluster that is a database object with a definite value. For example, it must have some definite restrictive objects (object node) and restrict conditions (value node or relationship node). In detail, it correspond to a sub tree rooted in R_s in object tree, in which:

- Rs must be object node, or
- Rs is connection node, whose two child nodes of operands are the same database object;
- The child nodes of Rs are value nodes, connection nodes or Set Chunk.

4.3 Convert Object Tree to SQL Query Sentence

The translation from the object tree representation to an SQL query to be sent to the audit database is a one-to-one translation. The only difficulty is that it requires a syntactical analysis of the input and the application of transformation rules on the syntactic tree.

A node sequence $NS =< n_1,n_2,\ldots, n_k >$ is a sequence of nodes in the schema that corresponds to a path starting from the node $n_1 = NS_{first}$ and terminating in the leaf node $n_k = NS_{last}$. For any node sequence *NS*, the relational query Query(*NS*) is obtained by combining the conditions on the edges of the sequence and projecting annot(n_k). For any node sequence *NS*, the relational query keyQuery(*NS*) is the same as Query(*NS*), except that the key columns of relation Rel(n_k) are also projected. Query (*NS*) and keyQuery(*NS*) are always conjunctive queries.

The algorithm of translation from object tree to SQL query sentence is shown as follows:

Output: SQL Query Sentence.

Demonstration: SetChunk2 SQL (child 1) deals with the transformation from Set Chunk to SQL query.

Process step:

```
(1)Initialize Set Chunk=(Null, Null, Null).
(2)Post-order traverse rt Tree, and partition maximum Set
   Chunk, Set Current node Sub rt.
```

```
    1) If the parent dependency node of Sub rt is com-
       parative operator:
       [1]Get the first child operand node child1, SC 1=
          SetChunk2SQL(child1);
       [2]Get the second child operand node child2, SC 2=
          SetChunk2SQL(child2);
       [3]Combine operand node and operator node into a
          new Set Chunk, Which is regard as a new leaf
          node;
    2) If current Set Chunk is a verb phrase as the ob-
       ject of the verb v1:
       [1]SC 1= SetChunk2SQL(Sub rt);
       [2]Combine SC1 and its parent node into a new leaf
          node SC;
    3) If current Set Chunk is a verb phrase as the ob-
       ject of the verb v1:
       [1]SC 1= SetChunk2SQL(Sub rt);
       [2]Partition the first verb predication into an-
          other maximum Set ChunkSC2,
          SC 2= SetChunk2SQL(Sub v1);
       [3]Combine SC1 and SC2 into a new leaf node SC;
    4) If current Set Chunk is the object of verb vq
       modified by some universal quantifier, the parent
       dependency node of Vq is E:
       [1]SC 1= SetChunk2SQL(Sub rt);
       [2]Map template qt to a new leaf node SC;
    5) Set current node root, goto (3).
(3) Expand SC to a SQL sentence.
```

5 Conclusions and Future Work

We have realized the query system supported by techniques mentioned above, and made some query experiments. The results proved that the utilization of FUG is effective. FUG can make the realization simplification that generates the inner expression of users' query in natural language. Unified description can make the wrapper of all kinds of objects representing corresponding grammar objects possible. The adoption of XML schema can combine with the FUG closely, and strengthen the extendable ability of the system. The use of SQL Object Tree is proved an effective medial expression that can abstract query object and query condition from the query put forward by users. In the future, we will introduce speech recognition into this query system. We convert the voice into query in character restricted by a certain context. This will make the system more convenient.

Acknowledgement

This work was supported by the National Nature Science Foundation of China under granted Number 60472125, the National High Technology Research and Development Program of China under Grant Number 2002AA142010, the Scientific Research Foundation of CAUC under granted Number K25025 and the Open Fundation of Tianjin Key Lab for Advanced Signal Processing. We would like to thank those organizations and people for their supports.

References

[1] David D., Lauri K.: Natural Language Parsing: Psychological, Computational, and Theoretical Perspectives, Cambridge University Press, UK, Cambridge (1985).
[2] Drabek, E. F., Qiang Z.: Experiments in learning models for functional chunking of Chinese text Systems, IEEE International Conference on Machine, Man, and Cybernetics (2001), pp. 859-864.
[3] Steve H.: Inside XML, Tsinghua University Press, China, Beijing (2002).
[4] Yao T.: Natural Language Understanding—A Kind of Research On How to Make Machine Understand the Language of People. Tsinghua University Press, China, Beijing (2002).
[5] Shan W., Xiao F. M.; Shuang Liu.Nchiql, A Chinese Natural Language Query System to Databases. Proceedings of International Symposium on Database Applications in Non-Traditional Environments, China, Beijing (2000), pp. 453-460.
[6] Cardenas M.A., Navarrete I. M.: Efficient resolution mechanism for fuzzy temporal constraint logic, Proceedings of the Seventh International Workshop on Temporal Representation and Reasoning TIME'2000 (2000), pp. 39-46.
[7] Kasper, R. T.: Feature Structures: A Logical Theory with Application to Language Analysis, University of Michigan, USA, Michigan (1986).
[8] Martin K., Xerox P. A.: Functional Unification Grammar: a formalism for machine translation, Annual Meeting of the ACL Proceedings of the 22nd conference on Association for Computational Linguistics, Stanford, California (1984), pp. 75-78.
[9] David R.: A development environment for multimodal functional unification generation grammars. Proceedings of 3rd International Conference on Natural Language Generation (INLG04), v2, ITRI Technical Report, Brockenhurst, UK (2004).

Hongyu Yang received the B.S. degree in 1992 from Jilin Institute of Technology, and the M.S degree in 1997 from Harbin Engineering University, and the Ph.D. degree in 2003 from Tianjin University, China, all in Computer Science. Since 1997, he has been with the School of Computer Science, the Software Technology Research Center (STRC) and Tianjin Key Lab for Advanced Signal Processing at Civil Aviation University of China, where he is currently Associate Professor and Director of STRC. Since 2003, he has been a postdoctoral researcher in the Department of Computer Science and Engineering at Tianjin University. From 2004 to 2005, he was a senior visiting scholar in the Security and Cryptography Laboratory (LASEC) at the School of Computer and Communication Sciences, Swiss Federal Institute of Technology Lausanne, meanwhile he was a senior visiting scholar in the Computer Engineering and Networks Laboratory (TIK) at the Information Technology and Electrical Engineering Department, Swiss Federal Institute of Technology Zurich.

Lixia Xie received the B.S. degree in Electronic Information Engineering from Harbin Engineering University, China, in 1996, and the M.S. degree in Software Engineering from Nankai University, China, in 2003. Since 1996, she has been with the School of Computer Science at Civil Aviation University of China, where she is currently Senior Lecturer.