Using Domain Knowledge in Knowledge Discovery

Suk-Chung Yoon
Dept. of Computer Science
Widener University
Chester, PA 19013

Lawrence J. Henschen
Dept. of ECE
Northwestern University
Evanston, IL 60208

E. K. Park
Computer Science Telecommunications
University of Missouri
Kansas City, MO 64110

Sam Makki
Computer Science Dept.
Royal Melbourne Inst. of Tech.
Melbourne, 3001, Australia

Abstract

With the explosive growth of the size of databases, many knowledge discovery applications deal with large quantities of data. There is an urgent need to develop methodologies which will allow the applications to focus search to a potentially interesting and relevant portion of the data, which can reduce the computational complexity of the knowledge discovery process and improve the interestingness of discovered knowledge. Previous work on semantic query optimization, which is an approach to take advantage of domain knowledge for query optimization, has demonstrated that significant cost reduction can be achieved by reformulating a query into a less expensive yet equivalent query which produces the same answer as the original one. In this paper, we introduce a method to utilize three types of domain knowledge in reducing the cost of finding a potentially interesting and relevant portion of the data while improving the quality of discovered knowledge. In addition, we propose a method to select relevant domain knowledge without an exhaustive search of all domain knowledge. The contribution of this paper is that we lay out a general framework for using domain knowledge in the knowledge discovery process effectively by providing guidelines.
Keywords: Knowledge discovery, Domain knowledge, Semantic query optimization

1 Introduction

In recent years, there has been an emerging research area, called knowledge discovery or data mining, which addresses the problems in finding implicit, previously unknown, and potentially useful patterns from databases [1, 2, 5, 6, 9, 11, 15, 16, 17, 21]. Currently, several successful tools that analyze databases for intersting and useful patterns have been reported in many areas of business, government, and science. As examples, several banks, using patterns discovered in loan and credit histories, have derived better loan approval and bankruptcy prediction methods, and packaged-goods manufacturers have searched the supermarket scanner data to measure the effects of their promotions and to look for shopping patterns [16]. We believe that data, intelligently analyzed and presented, are a valuable resource to be used for a competitive advantage in many areas.

During the last decade, we have seen an explosive growth of the size and number of databases with the advances in data acquisition and storage technologies. For example, typical marketing databases contain several gigabytes of demographic and purchasing data. When the system has to deal with large databases for knowledge discovery, there are major challenging issues, including computational efficiency and interestingness of patterns. Computational efficiency means that the process of identifying useful patterns should be efficiently implemented on a computer. Interestingness of patterns means the system should not generate too many patterns without focus. In most cases, exhaustive analysis of all the data is infeasible because of unacceptable performance. It is often necessary to perform a relatively constrained search on a specific subset of data for desired knowledge to improve the

efficiency of the knowledge discovery process. The key question is how we can find a relevant portion of the data without sacrificing the discovery of useful knowledge. In this paper, we suggest a method to solve the problem by applying and extending techniques in semantic query optimization.

The semantic query optimization approaches take advantage of domain knowledge about the contents of a database for query optimization. The basic idea is to use domain knowledge to reformulate a query into a less expensive yet equivalent query which produces the same answer as the original one. Previous work on semantic query optimization has demonstrated that significant cost reduction can be achieved by using these techniques. We believe the availability of relatively strong domain knowledge can also improve the efficiency of the knowledge discovery process by reducing the search space and help to focus on the interesting findings. Many current knowledge discovery methods and tools do not incorporate domain knowledge. There has not been any detailed discussion regarding the use of domain knowledge in different aspects of knowledge discovery. This motivates the development of mechanisms using domain knowledge in knowledge discovery. Our approach helps the system find a potentially relevant portion of the data by using domain knowledge which biases the search for interesting knowledge and narrows the focus of the knowledge discovery process.

In this paper, we introduce a method to utilize three types of domain knowledge in reducing the cost of finding a potentially relevant portion of the data while improving the quality of discovered knowledge. The contribution of this paper is that we develop a general framework for using domain knowledge in the knowledge discovery process effectively by providing guidelines.

The remainder of this paper is organized as follows. Section 2 discusses motivating examples. Section 3 surveys related works. Section 4 describes our approach to use domain knowledge in the knowledge discovery process. Section 5 presents our conclusions and possible extensions of our work for future research.

2 Motivating Examples

The following examples illustrate the benifits of domain knowledge in finding subsets of the population which are worthy of focused analysis.

Example 1 Suppose that we have a large ship leasing company database. We are interested in finding useful patterns of all ships whose deadweight is greater than 700 tons and whose speed is greater than 60 mph. Suppose we also have domain knowledge as follows: "all ships whose deadweight is greater than 700 tons have their speed greater than 60 mph" and "the ships whose deadweight is greater than 700 tons are supertankers". According to the domain knowledge, we need to consider only ships whose type is supertanker. We can reformulate the query so that there is no need to check the speed and the deadweight, which may save some execution time. If there exists an index on the shiptype, we can even save more execution time.

Example 2 Suppose that we have a database of car sale transactions. We are interested in finding patterns about domestic sports cars whose price is lower than $15,000. Suppose we also have domain knowledge saying that "there are no domestic sports cars whose price is less than $15,000". If we use the domain knowledge, we do not need to access the database to find the patterns. The domain knowledge is used to detect unsatisfiable conditions in the query, which prevents the exploration of search space.

Example 3 Let us consider a database of credit card transactions. For example, a senior executive at the company may wish to query, "what are the spending patterns of customers ?". If we have domain knowledge to categorize customers into various meaningful groups, e.g. senior, mid-old, middle, and young customers based on age, or high, middle, and low income customers based on income, then we can provide the information to the executive to help him/her make useful and meaningful queries by refining the original query or posing more restrictions on the query while constructing the search space. Now, the executive can ask "what are the spending patterns of senior customers with a high income ?", which can provide more meaningful information to the executive.

Example 4 Let us consider an employee database. We are interested in finding information about employees whose salary is greater than $50,000. Assume that we have domain knowledge describing meaningful correlations among attributes. For example, salary, position, and education are correlated but salary, address, and social security number are not. If we use such domain knowledge, we don't have to consider unrelated attributes such as address and social security number because there is little chance to find useful patterns from those unrelated attributes.

As you see from the above examples, it is useful to take advantage of domain knowledge for knowledge discovery. Domain knowledge can be used to reduce

the size of the database that is being searched for discovery by eliminating data records or irrelevant attributes that are not needed for discovery or refining the original query by posing more restrictions. We can transform a knowledge discovery query with conditions into another query which is more efficiently processed with domain knowledge. In addition, we can infer that a knowledge discovery query is unsatisfiable if it contradicts domain knowledge. In this case, there is no need to access the database to find information. If we use domain knowledge in the knowledge discovery process while evaluating a knowledge discovery query, we can make the discovery process more intelligent and avoid wasting time trying to deal with meaningless data.

3 Related Work

In this section, we discuss some of the works related to our approach in the areas of data mining and semantic query optimization. The purpose of knowledge discovery is to facilitate the understanding of large amounts of data by discovering interesting patterns. Many different methods for knowledge discovery have been proposed in the context of relational database systems [1, 2, 5, 6, 9, 11, 15, 16, 17, 21]. A recent book by Piatetsky-Shapiro and Frawley[16] contains a collection of articles about various approaches to pattern discovery. Frawley et al.[16] define patterns as follows:
Given a set of facts(data) F, a language L, and some measure of certainty C, a pattern S is a statement S in L that describes relationships among a subset F_s of F with certainty C, such that S is simpler than the enumeration of all facts in F_s.

Some databases are so large that even the fastest algorithms for knowledge discovery can be too expensive to apply to all data. There are several approaches that can be utilized in order to minimize the search efforts. One simplest method is to randomly sample a database. However, this method may hinder the discovery of useful knowledge. Even though sampling is appropriate, it is difficult to decide how much to sample. Another possibility is to use other database utilities such as OLAP(On-Line Analytical Processing) to specify a subset of data. However, it is not always possible to use those utilities in order to restrict the set of data.

Klemettinen et al. [13] propose to reduce the number of generated rules by having a user specify templates, which define the structure of interesting association rules (i.e., what attributes occur in the antecedent and what attributes are in the consequent) and restrict the items that can be used in the rule. This is accomplished by organizing the items into a classification hierarchy, and using this information to guide the rule selection process. Shen et al. [19] introduce the notion of metaqueries for guiding the data mining process. Similar to templates, metaqueries work by specifying an abstract form that the rule must satisfy. Metaqueries in their most general form are most useful for mining data from different relations in a database. These approaches require greater interaction from users to identify an interesting portion of data. It is often difficult for users to predict what kinds of patterns should be mined beforehand.

We believe another effective approach is to apply and extend techniques in semantic query optimization to the knowledge discovery process. Semantic query optimization can be regarded as the process of transforming a query into an equivalent form, which can be evaluated efficiently. There have been some interesting studies on semantic query optimization in relational and deductive databases. King[11] describes an algorithm which uses a set of transformation heuristics. These heuristics help to limit the number of transformations by specifying how each heuristic can be used to transform a given query. Xu[25] presents heuristics similar to King's, adding a control strategy for selecting appropriate transformations. Hammer and Zdonik[9] describe how a system can use a knowledge base to perform semantic query optimization. Jarke et al.[10] describe a graph-theoretic approach to semantic query simplication implemented in Prolog. Similarly, Shenoy and Ozsoyoglu[20] suggest a graph-theoretic approach to achieve semantic query optimization by identifying redundant conditions and eliminating them from the query graph. Chakravarthy et al.[4] describe a method to modify query expressions by comparing them with semantic knowledge expressions and forming new expressions. The modified query should then be easier to answer than the original query.

In this paper, we apply and extend the approaches developed for semantic query optimization to the knowledge discovery process. In our approach, we introduce a specific method to represent domain knowledge and suggest strategies to control the size and quality of data explored in the knowledge discovery process by using domain knowledge. These issues are not addressed in [13] and [19]. We believe our approach increases the chances of finding patterns that are of interest to the users and can make the knowledge discovery process more efficient by constraining the search space.

4 Our Discovery System

In this section, we describe our approach to use three types of domain knowledge in the knowledge discovery process. In many discovery applications, a key phase is to find a subset of the population which is worthy of focused analysis. Our approach allows domain knowledge to be incorporated into the key phase. Our approach consists of two phases: preprocessing and execution. The preprocessing phase organizes domain knowledge for fast access and is done once before knowledge discovery queries are issued. The execution phase accesses domain knowledge and applies it to improve the knowledge discovery query. We now consider each phase in detail.

4.1 Preprocessing

In this phase, we collect all domain knowledge and represent it in different forms according to the types. Domain knowledge, not belonging to data, may originate from many sources, including specifications and domain experts. It is often possible for domain experts to provide significant amounts of domain knowledge. Newly discovered patterns can be added to the set of domain knowledge and used in the future as domain knowledge. Our domain knowledge can be classified into three types: interfield, category, and correlation.

Interfield domain knowledge which describes relationships among attributes of a relation can be represented in the following rule form:

If ($A_1 op B_1$) and/or ... and/or ($A_{n-1} op B_{n-1}$)
then ($A_n op B_n$) and/or ... and/or ($A_m op B_m$)

where A_i, $1 \leq i \leq$m, is the name of an attribute, op is normally one of the operators $\{=, <, \leq, >, \geq, \neq\}$, and B_j, $1 \leq j \leq$m, is a constant value from the domain of the attribute A_j or the name of an attribute. The domain knowledge in example 1 shows the relationships among attributes of the relation ship and is therefore interfield domain knowledge. It can be represented as follows:

If deadweight> 700 then speed > 60
If deadweight> 700 then shiptype="supertanker"

The domain knowledge in example 2 is also interfield domain knowledge and can be represented as follows:

If type="domestic sports car" then price>15000

As usual, the conditions between "if" and "then" form the body of the piece of knowledge, and the conditions after "then" form the head.

Category domain knowledge represents useful categories for the domain of an attribute and can be represented in the following rule form:

If ($A_1 op B_1$) and/or ... and/or ($A_{n-1} op B_{n-1}$)
then A_n="abstract concept"

where A_i, $1 \leq i \leq$n, is the name of an attribute, op is normally one of the operators $\{=, <, \leq, >, \geq, \neq\}$, and B_j, $1 \leq j \leq$n-1, is a specific value from the domain of the attribute A_j or an abstract concept which is previously defined in other category domain knowledge. An abstract concept is a high-level concept to summarize the values which satisfy conditions. Different values in the domain may be generalized to the same abstract concept. Category domain knowledge might be views or summaries of data. For example, customers in the example 3 can be divided into four meaningful categories based on age: senior, mid-old, middle, and young. The category domain knowledge can be represented as follows:

If (age$\geq$65) then age=*senior*
If (age$\geq$45) and (age<65) then age=*mid-old*
If (age$\geq$35) and (age<45) then age=*middle*
If (age<35) then age=*young*

Similarly, we can define other abstract concepts. For example, we can also divide customers into three meaningful categories based on income: high, middle, and low. Category domain knowledge can also be formed by combining two or more abstract concepts. That is, abstract concepts across multiple attributes can form category domain knowledge. Such domain knowledge represents global relationships among abstract concepts from different attributes. For example, we can define *loyal* customers with the abstract concepts defined in age and income attributes as follows:

If age=*middle* and income=*high*
then customer-type=*loyal*.

Similarly, we can define the abstract concept *Honor Students* by combining the abstract concept *Undergraduate* in the attribute class and the abstract concept *Excellent* in the attribute GPA in a student relation.

Any existing attribute can be used to define meaningful categories if the attribute has a set of values and finite number of meaningful high-level concepts for those values is existed. That is, if the attribute domain is divided into several meaningful groups and each group can be represented by a meaningful concept, then we can define categories for the attribute. The category domain knowledge provides varying levels of abstraction that can be accessed based on the specific goals of the data analysis.

Correlation domain knowledge suggests correlations among attributes and can be represented in a set form. For example, domain knowledge saying that education, position, and income are correlated can be represented as {education, position, income}. For exam-

ple, if our task is to find patterns related to income, we may focus on position and education, not name or address, because it is unlikely that income and name are correlated. If we have domain knowledge saying that profits, sales, and expenses are related, then we might consider the attributes, sales and expenses to find patterns related to profits. If correlation domain knowledge can rank the relevance of all the related attributes, we select only the highly ranked attributes or leave the selection to the user to decide where to draw the line. If we use this type of domain knowledge, we can eliminate some attributes that are redundant, irrelevant, or unimportant to a given discovery task.

To select relevant domain knowledge without an exhaustive search of domain knowledge, we define a cluster of an attribute P to be a set of domain knowledge related to the attribute P. Each cluster has three parts, one for each domain knowledge type. So, we can easily find each domain knowledge type in the cluster. Grouping domain knowledge into clusters avoids unnecessary search on a large body of irrelevant domain knowledge. We store those clusters in a table and create an index on the attributes in the table to speed up the retrieval of domain knowledge. For example, if we have an attribute income in an employee relation, we might have the following cluster for the attribute in the domain knowledge table.

attribute name	type	domain knowledge
income	interfield	if income > 50000
		then position=manager
	category	if income > 60000
		then income="high"
		else if income > 35000
		then income="middle"
		else income="low"
	correlation	{education, position}

Our approach can save considerable time because searching through domain knowledge to find ones that are applicable to a given query can be done efficiently.

4.2 Execution

In the execution phase, we receive a knowledge discovery query, identify domain knowledge that is relevant to the query, and transform the query with the domain knowledge into another form which is more efficiently processed. During this stage, we can reduce the search space or discard the query itself totally by using only relevant domain knowledge among all domain knowledge. How can relevant domain knowledge be detected ? In our approach, the relevance is decided by a query context. Interfield domain knowledge is relevant to a query if a subset of the conditions in the query implies the body of domain knowledge. Category domain knowledge is relevant to a query if the query includes conditions to be categorized. Correlation domain knowledge is relevant to a query if the query includes attributes which have a list of relevant fields. Upon receiving a query, we bring potentially relevant domain knowledge from the domain knowledge table by using the index created in the preprocessing phase and then check if any domain knowledge is relevant to the query. We can select relevant domain knowledge without an exhaustive search of all domain knowledge. Each selected relevant domain knowledge is processed according to its type as follows:

Case 1: interfield domain knowledge

To process a query with this type of domain knowledge, we use the head of the relevant domain knowledge to transform the query into an equivalent one, which can be evaluated efficiently. The head of the domain knowledge can be used in one of the following two ways.

1) The head of the relevant domain knowledge implies a subset of the conditions in the query: we can remove the subset from the query, which can eliminate the unnecessary and redundant restrictions in the knowledge discovery query.

2) The head of the relevant domain knowledge gives useful additional restrictions to attributes involved in a query: we add the restrictions to the query, which can reduce the processing cost and the number of inner scans of the relation.

In example 1, a user formulates the query, asking "find interesting knowledge of all ships whose deadweight is greater than 700 and whose speed is greater than 60 mph". To process this request, we first look at the domain knowledge table for those two attributes, deadweight and speed. Assume that we have the same domain knowledge as in example 1. The domain knowledge is relevant to the query because the body of the domain knowledge is true. After relevant domain knowledge is found, reformulation process is performed. If we use the head of the first domain knowledge, we can eliminate the second condition in the query which is redundant. If we use the head of the second domain knowledge, the first condition is changed to shiptype=*supertanker*. The original query is reformulated as follows:

"Find interesting knowledge of all ships whose shiptype is supertanker."

In the example 2, a user formulates the query, ask-

ing "find interesting knowledge of all sports cars whose price is less than 15000". Assume that we have the same domain knowledge as in example 2. The domain knowledge is relevant to the query because the body of the domain knowledge is true. If we use the head of the domain knowledge, we don't have to access the database because the condition in the query contradicts the domain knowledge.

Case 2: category domain knowledge

To process a query with this type of domain knowledge, we identify conditions on which additional constraints might be meaningful. Then, we find relevant categories for the conditions in the query and show them to a user. The user can select useful and meaningful categories and refine the original query by posing more restrictions. That is, we can ask a user additional constraints or restrictions to be included as part of conditions in a knowledge discovery query by showing categories related to a given task. This kind of domain knowledge enables the user to specify the level of analysis at which the system should focus and provides an interface for users to specify the set of data of interest easily according to their needs. In the example 3, we have four categories based on age and three categories based on income. We show those categories to the user, and the user can select some interesting categories. The original query can be reformulated as "what are the spending patterns of *senior* customers with a *high* income ?" or "what are the spending patterns of *loyal* customers ?". In this case, the user narrows the search to senior customers with a high income or loyal customers. To process the reformulated query, abstract concepts contained in the reformulated query are identified and then appropriate mappings are performed to transform the abstract concepts into the set of the conditions based on the primitive data.

Case 3: correlation domain knowledge

To process a query with this type of domain knowledge, we look at the domain knowledge table and find a list of attributes related to the attributes in the query. In example 4, we need to find patterns related to income. If we use the domain knowledge in example 4, education and position are relevant fields on which to focus attention. This type of domain knowledge suggests which fields are appropriate for a given task. We can reduce the size of data set by limiting the number of attributes.

Using domain knowledge to bias the search for meaningful patterns leaves some portion of search space unexplored. Our approach allows backtracking to be initiated for further exploration on the unexplored portion of search space if more patterns are needed. For example, if we use more than one category domain knowledge in query reformulation, we can remove some of them and reprocess the query. For example, if the reformulated query has two abstract concepts, say, senior and high income, then we can drop one of the abstract concepts. So, we find the spending patterns of senior customers or high income customers. If correlation domain knowledge is used in the reformulated query, we might add attributes which are excluded in the query reformulation process. For example, we might add another attribute such as number of dependents to the query in example 4.

5 Conclusion

The amount of information stored in databases is exploding. Large amounts of data need to be summarized or reduced to descriptions of a form. Thus, knowledge discovery in databases has been attracting significant attention in the past few years. The challenge of knowledge discovery is to process large quantities of raw data efficiently, while producing the most significant and meaningful patterns for achieving a user's goal. Exhaustive analysis is almost impossible on the megabytes, gigabytes, or even tera-bytes of data in many real-world databases. In these situations, the system should focus its analysis on samples of data by selecting specific fields and/or subsets of records.

In this paper, we have presented a method for using domain knowledge which assists the discovery process by focusing search and helps make the discovered knowledge more meaningful to a user. In particular, we look at the use of domain knowledge to constrain or prune the search space and optimize knowledge discovery queries used to find interesting patterns. In addition, we suggest a method to select relevant domain knowledge without an exhaustive search of domain knowledge.

Our approach provides a simple and reasonable way of using domain knowledge in very large databases in conjuction with the current discovery methods. Our approach can be incorporated as a component in current discovery systems. Our approach increases the chance of finding patterns that are of interest to the user and can make the knowledge discovery process more efficient by constraining the search space.

In the near future, we might need to develop heuristics which help select the most promising set of domain knowledge. It is an interesting topic to measure the quality of each domain knowledge by its discrimination power, its generality, and its interestingness for

future research. Currently, we are building a prototype system to perform experimentation on the proposed methods.

References

[1] R. Agrawal, et.al., "Mining Association Rules between Sets of Items in Large Databases", *Proceedings of ACM SIGMOD*, pp.207-216, 1993

[2] R. Agrawal and R. Srikant, "Mining Sequential Patterns", *Proceedings of the Eleventh International Conference on Data Engineering*, pp.3-14, 1995

[3] F. Bancilhon, et.al., *Building an Object-Oriented Database System*, Morgan Kaufmann Publishers, 1992

[4] U.S. Charkravarthy, et.al., "Logic-based Approach to Semantic Query Optimization", *ACM Transaction on Database Systems*, 15(2): pp.162-207, 1990

[5] V. Dhar and A. Tuzhilin, "Abstract-Driven Pattern Discovery in Databases", *IEEE Transactions on Knowledge and Data Engineering*, vol. 5, no. 6, pp. 926-938, 1993

[6] W.J. Frawley, et.al., "Knowledge Discovery in Databases: An Overview", *Knowledge Discovery in Databases*, G. Piatetsky-Shapiro and W.J. Frawley(Eds), pp. 1-27, AAAI/MIT Press, 1991

[7] J. Freytag, et.al., *Query Processing For Advanced Database Systems*, Morgan Kaufmann Publishers, 1994

[8] H. Gallaire, et.al., "Logic and Database: A Deductive Approach", *Computing Surveys*, vol. 16, no. 1, pp. 154-185, 1984

[9] J. Han, et.al., "Data-Driven Discovery of Quantitative Rules in Relational Datbases", *IEEE Transactions on Knowledge and Data Engineering*, vol. 5, no. 1, pp.29-40, 1993

[10] M. Jarke, "Semantic Query Optimization in Expert Systems and Database Systems", *Proceedings of the First International Conference on Expert Database Systems*, pp. 467-482, 1984

[11] K.A. Kaufman, et.al., "Mining for Knowledge in Databases: Goals and General Description of the INLEN System", *Knowledge Discovery in Databases*, G. Piatetsky and W. Frawley(Eds.), pp.449-462, AAAI/MIT Press, 1991

[12] W. Kim, *Introduction to Object-Oriented Databases*, MIT Press, 1990

[13] M. Klemettinen, et.al., "Finding Interesting Rules from Large Sets of Discovered Association Rules", *Proceedings of the Third ACM International Conference on Information and Knowledge Management*, pp. 401-408, 1994

[14] J. W. Lloyd, *Foundation of Logic Programming*, Springer-Verlag, 1984

[15] J.S. Park, et.al., "An Effective Hash-Based Algorithm for Mining Association Rules", *Proceedings of ACM SIGMOD*, pp.175-186, 1995

[16] G. Piatetsky-Shapiro, and W.J. Frawley, Eds., *Knowledge Discovery in Databases*, AAAI/MIT Press, 1991

[17] G. Piatetsky-Shapiro, "Discovery, Analysis and Presentation of Strong Rules", *Knowledge Discovery in Databases*, G. Piatetsky and W. Frawley(Eds.), pp.229-248, AAAI/MIT Press, 1991

[18] E. Rundensteiner, "A Classification Algorithm for Supporting Object-orienetd Views", *Proceedings of the Third ACM International Conference on Information and Knowledge Management*, pp. 18-25, 1994

[19] W. Shen, et.al., "Metaqueries for Data Mining", *Advanced in Knowledge Discovery and Data Mining*, U. Fayyad, G. Piatetsky-Shapiro, P. Smyth, and R. Uthurusamy (Eds.), pp. 375-398, AAAI/MIT Press, 1996

[20] A. Silberschatz, et.al., "Database Systems: Achievement and Opportunities", *Communication of ACM*, 34:94-109, 1991

[21] M. D. Siegel, "Automatic Rule Derivation for Semantic Query Optimization", *Knowledge Discovery in Databases*, G. Piatetsky-Shapiro and W. Frawley(Eds.), pp.411-427, AAAI/MIT Press, 1991

[22] J. Ullman, *Principles of Database and Knowledge-base Systems*, Vol I and II, Computer Science Press, 1988

[23] S. C. Yoon, et.al., "Intelligent Query Answering in Deductive and Object-Oriented Databases",

Proceedings of the Third ACM International Conferenece on Information and Knowledge Management, pp.244-251, 1994

[24] S. C. Yoon, et.al.,"Semantic Query Processing in Deductive Object-Oriented Databases", *Proceedings of the Fourth ACM International Conference on Information and Knowledge Management*, pp.150-157, 1995

[25] D. Xu, "Search Control in Semantic Query Optimization", University of Massachusetts, Department of Computer Science, Tech Report TR83-09, 1983

[26] S. Zdonik, and D. Maier, *Readings in Object-Oriented Database Systems*, Morgan Kaufmann Publisher, 1990