RELEVANT RULE DERIVATION FOR SEMANTIC QUERY OPTIMIZATION

Junping Sun†, Nittaya Kerdprasop‡, and Kittisak Kerdprasop∗
School of Computer and Information Sciences, Nova Southeastern University
Fort Lauderdale, Florida 33314, USA†
Department of Mathematics, Chulalongkorn University
Bangkok 10330, THAILAND‡
School of Computer Engineering, Suranaree University of Technology
Nakorn Ratchasima, THAILAND∗

ABSTRACT

Semantic query optimization in database systems has many advantages over the conventional query optimization. The success of semantic query optimization will depend on the set of relevant semantic rules available for semantic query optimizer. The semantic query optimizer utilizes a set of available semantic rules to further explore extra query optimization plans for conventional query optimizer to choose. Semantic rules represent the dynamic database state at an instantaneous time point. Finding such set of relevant semantic rules can be very beneficial to support both semantic and conventional query optimizations. In this paper, we will show how to use inductive logic programming approach to derive relevant rules from the data in a database. Language bias as heuristic is used to reduce the search space as well as the costs in the process of inductive rule derivation. Effectiveness and efficiency of our bias generator algorithm are evaluated and their evaluation results are presented in this paper.

1 INTRODUCTION

Query optimization in relational database systems has been an active research subject for many years. The techniques used for relational query optimization can be classified into conventional query optimization and semantic query optimization. The conventional query optimization utilizes syntactic and logic equivalence transformation of a query expression, and various statistical estimation profiles of a database. The capabilities of conventional query optimization are limited to syntactic information in a query expression, however. In comparison, semantic query optimization uses not only syntactic transformations, but also semantic knowledge expressed by semantic integrity constraints or semantic rules to aid the query transformation (semantic equivalence transformation).

Early research work of semantic query optimization can be traced back to knowledge-based query optimization [11], QUIST (QUery Improvement through Semantic Transformation) [14] [15], deductive logic-based approach [3] [5], and others [4] [23] [24]. In QUIST [15], King used a set of integrity constraints as a knowledge base to transform or simplify a query into an optimized one by semantic reasoning. Heuristics such as *index introduction, restriction introduction, join elimination, contradiction detection, etc.* are used to explore more alternatives for conventional query optimizer. Further, if there is contradiction detected between a query and integrity constraints, then it implies that the query cannot be satisfied. So the empty answer to the query can be given directly from the semantic transformation. Of course, the query is semantically optimized since the query can be answered without accessing the physical database. Charkravarthy *et al.* [3] proposed a deductive database-based logic model, which consists of extensional database (EDB), intensional database (IDB), and integrity constraints (ICs), to formalize the semantic query optimization; and resolution refutation method was used to perform semantic query transformation. Siegel *et al.* [21] [22] used automatic rule derivation method based on deduction to learn new rules applicable to semantic query transformation. With the set of user-defined integrity constraints and new derivable rules associated with EDB and IDB predicates, more opportunities can be explored for both semantic query optimization and conventional query optimization. All of these methods mentioned above [3] [5] [11] [14] [15] [20] [21] [22] conduct semantic transformation based on a set of integrity constraints, a set of intensional database predicates(IDB), and a set of extensional database predicates (EDB). But the rich semantics embedded in database instances is overlooked in terms of semantic rule derivation [21] [22] and semantic query optimization. In this paper, we will present a heuristic-based approach to induce a set of relevant semantic rules from not only the set of ICs, IDB and EDB predicates, but also the database instances for semantic query optimization. From this point of view, more semantic rules will provide more alternatives for semantic query optimization.

2 SEMANTIC RULE DERIVATION

The inductive rule derivation is different from the deductive approach [21] such that: inductive rule derivation approach is to induce a set of new semantic rules from both the intensional database (IDB predicates) and extensional database (EDB predicates and instances), whereas the deductive approach is to derive new rules from the intensional database in terms of deductive database model. As far as the semantic query optimization is concerned, the induction-based rule derivation methods can provide more opportunities for semantic query optimization.

Typical rule derivation in a single relational database table can be found in [2] [12] [13]. Han *et al.* developed an attribute-oriented inductive method to discover rules from a relational database table. Their method integrated *learning from examples* techniques from the machine learning paradigm to extract generalized knowledge from actual data in databases. Also an attribute-oriented concept tree was used to guide the learning process and to reduce the computational complexity. The set of rules derived by the methods in [2] [12] [13] are complete by augmenting each of the derived rules with quantitative votes.

Džeroski and Lavrač [10] put inductive learning in the context of a set of relation schemas defined either extensionally or intentionally in the framework of deductive databases. Their approach can induce rules on multiple virtual relations from examples such as positive and negative tuples, and predefined relation schemas in a deductive database. Discovering semantic rules involving multiple relation schemas or predicates makes attractive advances in knowledge discovery in databases comparing with learning rules from a single extensional relation. It also increases the difficulty of the learning task due to the combinatoric complexity in the problem domain of the multiple virtual relations.

Based on the theory of clausal discovery [7], CLAUDIEN (CLAUsal DIscovery ENgine) is the implementation of an inductive logic programming engine that combines data mining principles with inductive logic programming. It discovers clausal regularities (rules) from unclassified data, and it aims to discover maximally general hypothesis, where examples for inductive learning are represented by Herbrand interpretation. In order to reduce the computational complexity in the process of clausal discovery, there is an interface for CLAUDIEN to accept language bias specification. The language bias can be used to reduce the size of the search space in the process of inductive rule derivation and make the process more efficient.

Due to large volume of data in databases, it is very important to find the bias information for inductive rule learning. In this paper we will present bias generating method for knowledge discovery in databases. The bias specification will be used for inductive logic programming engine to derive semantic rules relevant to a query, and these relevant rules can be used for semantic query optimization process.

3 SEMANTIC QUERY OPTIMIZATION

One of major purposes to derive relevant rules is for semantic query optimization. In a deductive database model [3][5], there are three components such as: extensional database (EDB), intensional database (IDB), and integrity constraints (IC) described as follows:

1. **EDB:** a set of function-free unit clauses with no variables;
2. **IDB:** a set of nonrecursive range-restricted function-free definite Horn clauses (i.e., with at least one atom in the body and exactly one atom in head);
3. **IC:** a set of nonrecursive range-restricted Horn clauses.

The following example shows how a database query can be semantically optimized.

Example:

EDB:
EDB1: Employee(Ssno, Salary, Deptno, Age)
EDB2: Department(Deptno, Managerssno, Floorno)
EDB3: Sales(Deptno, Item, Volume)

IDB:
IDB1: Highsalesdept(x1,x2,x3,y2,y3)←
Department(x1,x2,x3),
Sales(x1,y2,y3),
(y3>100000)

IDB2: Highsalesmgrprofile(x2,y2,y4)←
Employee(x2,y2,y3,y4),
Highsalesdept(x1,x2,x3,x4,x5)

IC:
IC1: ←Department(x,y,2)

There are no departments on floor 2.

IC2: (y > 40000)← Employee(x,y,z,u),(u>50)

All employees whose age is greater than 50 have salary greater than 40000.

By the way, the IDB rule such as:

Highsalesmgrprofile(x2,y2,y4)←
Employee(x2,y2,y3,y4),
Highsalesdept(x1,x2,x3,x4,x5)

can be transformed into the following format with body of the rule containing only EDB predicates or extensional relation schemas such as:

Highsalemgrprofile(x2,y2,y4)←
Employee(x2,y2,y3,y4),
Department(y1,x1,y2),
Sales(y1,z2,z3),
(z3>100000)

For a given query such as:

←Highsalesdept(x*,y,2,z*,u)

it can be transformed into the following query by using IDB rule IDB2 to the following equivalent query:

←Department(x*,y,2),Sales(x*,z*,u),(u>100000)

By replacing the IDB predicates with EDB predicates in a query, query optimization strategies can be further explored based on heuristics such as: *index introduction, join elimination, contradiction detection, etc.* Further, with checking the integrity constraints IC1, ←Department(x,y,2), we find out that the transformed query cannot be satisfied, so the query can be answered without accessing the physical database and optimized. In this research, besides the rules given in IDB and IC, we will use inductive logic programming approach to derive a set of semantic rules from data in a database in order to support semantic query optimization.

4 INDUCTIVE LOGIC PROGRAMMING

Some recent inductive learning systems construct logical definitions of relation schemas from examples and background knowledge (other relation schemas) [6]. The restricted forms of program clauses [16] are used to represent training examples, background knowledge, and induced concept descriptions. In this case, learning can be considered as logic program synthesis and has been recently named as *inductive logic programming* (ILP).

ILP can be classified into interactive vs. empirical ILP systems, which learn definitions of single and multiple relation predicates, respectively. Empirical ILP systems require all training examples at the start of the learning process, whereas interactive ILP systems process the example one by one and possibly generate examples in the learning process. For empirical ILP, it can be distinguished between top-down approach and bottom-up approach.

The top-down approach (general-to-specific) starts learning from the empty hypothesis which is the most general hypothesis. The empty hypothesis can imply both positive and negative examples. Since the hypothesis is too general, it needs to be further specialized. The specialization is performed by applying deductive inference rules (the reverse of inductive inference rules) in order to remove or modify part of the hypothesis that implies negative examples. During the search for the correct hypothesis, the ILP systems have to check that the deduced hypothesis implies less negative examples and most of positive examples, otherwise that hypothesis is dropped or modified. The search process is performed repeatedly until the least specialized hypothesis that implies none of the negative examples and most of the positive examples is found and returned as the output of the learning process.

The bottom-up ILP approach starts the learning process from the most specific hypothesis. The selection of the initial hypothesis clause can be, theoretically, an infinite choice [7]. Moreover, pruning part of the search space is more reliable in a top-down search.

The power of ILP systems is traded with its complexity. Fortunately, the complexity can be controlled by the feature of the explicit language bias. If the language bias is specified appropriately, ILP system can discover rules efficiently. The question is how to find an appropriate form of the language bias specification. In later section, we will show our approach to generating language bias for inductive rule derivation.

Based on the above discussions and Figure 1, we can formulate the following problem statement:

Problem Statement:

1. Given a query, a set of EDB and IDB predicates, and a set of ICs, generate a biased grammar as input for CLAUDIEN to derive a set of relevant semantic rules.

2. Based on the biased grammar, have CLAUDIEN perform induction on the set of EDB instances to derive efficiently a set of rules relevant to the query.

5 CLAUDIEN SYSTEM

CLAUDIEN [7] is a top-down, non-interactive batch learner. It operates in the nonmonotonic setting, where each example is a Herbrand interpretation, using only positive examples. The system starts with the empty clause, and uses a downward refinement operator to find the most general clauses true in each of the positive examples. Further, rather than using a hill-climbing search strategy as FOIL [18] [19] does, CLAUDIEN uses a complete depth-first iterative deepening search strategy. Large parts of the search space are pruned by dropping redundant clauses, tautological clauses, or clauses that already implied by the current hypothesis. Besides the training examples and background knowledge, CLAUDIEN also needs a user-specified language bias to limit the search space. The language bias L contains a set of all syntactically well-formed clauses, and the hypothesis space is thus composed of all possible subsets of L. Therefore, the discovery task of CLAIDIEN can be stated as discovering a minimally complete set of rules that satisfies a given database and a language bias. The algorithm of CLAUDIEN can be described in Figure 2:

The CLAUDIEN algorithm starts with the initialization of a set of candidate clauses Q, which is initialized to contain the most general clause in L, i.e., Q = { false ← true }, a null clause which means *false*. The key concept underlying CLAUDIEN is that a clause that is false in the minimal model of the knowledge base KB is overly general because it implies any examples; thus, it should be specialized. Further, the language bias can be utilized to specialize initial set Q. Based on the above discussion, it is not hard to find the important role of bias specification in semantic rule derivation by using CLAUDIEN with respect to the inductive logic programming.

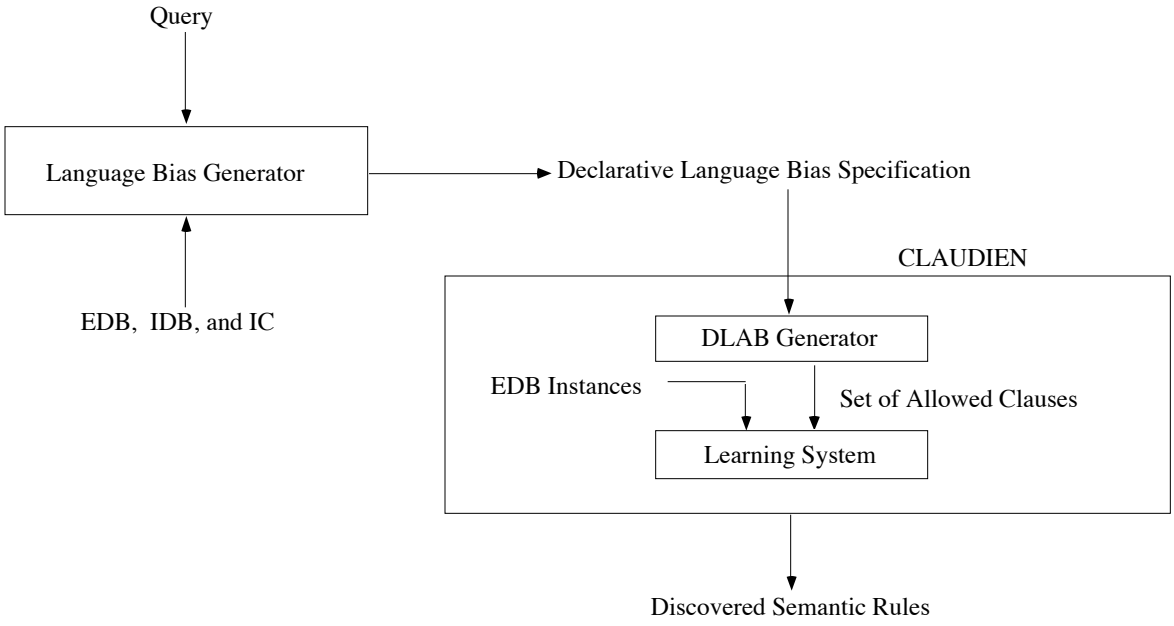

Figure 1: Input and output of the language bias generator

Algorithm 5.1 CLAUDIEN

Input: *Knowledge Base KB*
(It contains training examples)
Language Bias L

Output: *Set of Characteristic Rules R*

begin

1. *Initialization*
   $Q = \{ false \leftarrow true \}$
   / Q is a set of candidate clauses */*
   $R = \emptyset$
2. **while** *Q is not empty* **do**
   (a) *delete a clause C from Q*
   (b) **if** *C is true in the minimal model of KB*
   then *add C to R*
   else *add all specialization of C to Q*
   endif

   endwhile
3. *Scan all clauses in R to remove the redundant clauses*
4. *Return the set of discovered rules R.*

end

Figure 2: The CLAUDIEN system

6 BIAS SPECIFICATION FOR INDUCTIVE RULE DERIVATION

One of the key components in CLAUDIEN system is the explicit specification of the language bias, which specifies the desired format that the clauses induced by the ILP should take. In terms of discovering rules for semantic query optimization, the learning task has to search for all interesting and valid rules. The number of clauses in the search space starting the generalized criterion can be very large, it is critical to specify and use bias to reduce the size of the search space. One of the guidance we use here is to generate bias based on the query, and the target information such as IC (integrity constraint), IDB (intensional database), and EDB (extensional database).

DLAB (Declarative LAnguage Bias) [7] in CLAUDIEN is the extension of syntactic bias format proposed by Adé, De Raedt, and Bruynooghe [1]. In CLAUDIEN [8] [9], the concept of template called *dlab_template* is used to specify the set of clauses allowed in the search space. By doing that, we could limit the search space in the process of semantic rule derivation.

In order to derive relevant rules and limit the search space during the rule derivation, we propose a query driven language bias generator. The language bias generator algorithm takes the query pattern, IC, IDB, and EDB as input and generates language bias in terms of DLAB template as output.

In order to illustrate the **Algorithm 6.1**, we give the following example.

Algorithm 6.1 Language Bias Generator

Input: *A Query Clause Q*
Set of Integrity Constraints ICs
Set of IDB Predicates and Rules
Set of EDB Predicates

Output: *A Biased DLAB Grammar*

begin

1. *read query Q /* Initialization */*
2. **repeat**
   if *Q contains IDB predicates*
   then *substitute each IDB predicate with corresponding EDB predicate(s)*
   endif
   until *Q contains no IDB predicates*
3. **if** *Q contains constants*
   then *substitute each constants with a variable*
   declare a variable in a 'dlab_variable' construct
   else *'dlab_variable' is empty.*
   endif
   / Create HeadPredicateList and BodyPredicateList */*
4. **for** *each query clause obtained from step 2 and 3* **do**
   add query clause to the list of BodyPredicateList
   search for subsuming integrity constraints S from the set of ICs
   / subsuming integrity constraint is an integrity constraint in which */*
   / part of it subsumes the body of a given query clause */*
   rename variables in S to match corresponding variables in Q
   add S predicates to the list of HeadPredicateList
   endfor
   / Compacting HeadPredicateList and BodyPredicateList */*
5. *scan HeadPredicateList and remove the redundant predicates*
6. *scan BodyPredicateList and remove the redundant predicates*
7. *form head of 'dlab_template' from HeadPredicateList*
8. *form body of 'dlab_template' from BodyPredicateList*
9. *return a DLAB grammar G*

end

Figure 3: The language bias generator

Example:

EDB: father(x,y) mother(x,y) female(x)

IDB: { }

IC: ←mother(x,y),¬female(x)

Query: ?-mother('Sue',y),male(y)

EDB Instance:

mother('Sue','Linda') mother('Sue','Ann')
mother('Sue','Alice') mother('Sue','Grace')
female('Sue') female('Linda') female('Ann')
female('Alice') female('Grace')

The execution of the **Algorithm 6.1** is given as follows:

1. Read the query: ?-mother('Sue',y),male(y)
2. No IDB predicate substitution (simplicity of the illustration)
3. Substitute constant with variable in the query:
   ?-mother('Sue', y),male(y)
   ?-mother(name, y),male(y)
   dlab_variable(name,1-1,['Sue']) where 1-1 means the first variable of the first predicate in the query.
4. Search for subsuming integrity constraints:
   ←mother(x,y),¬female(x)
   Create *HeadPredicateList* to contain subsuming IC predicates which are *mother(x,y)* and *female(x)*. The variables in these two predicates are renamed to match the query clause in 3 (X/name). Thus, the *HeadPredicateList* is:
   HeadPredicateList =
   [*mother(name,y),female(name)*]
   Create *BodyPrediateList* to contain the query clause (in 3)
   BodyPrediateList = [*mother(name,y),male(y)*]
5. No redundant predicates removed in both *HeadPredicateList* and *BodyPrediateList*.
6. Form a DLAB grammar G:

   dlab_template('0-len:[*mother(name, y),*
   female(name)]
   ←0-len:[*mother(name,y),male(y)*]')

   dlab_variable(name,1-1,['Sue'])

The grammar G generated by **Algorithm 6.1** can be used as a bias to constrain the search space of the CLAUDIEN system. The semantic rules induced by CLAUDIEN are as follow:
false←male(A)
female(Sue)←true
female(A)←mother(Sue,A)
The last rule derived shows additional integrity constraint that can be used for semantic query optimization since the rule states that *every child of Sue is female.*

7 EFFECTIVENESS OF THE LANGUAGE BIAS GENERATOR ALGORITHM

7.1 Experiments and Results

In order to test the effectiveness and efficiency of our language bias generator algorithm, our experiments were designed to show the degree of relevance of these derived rule sets. The efficiency is evaluated based on the search space reduction with comparison to the exhaustive inductive approach. Effectiveness of the algorithm is evaluated on the basis of the ability of the discovered semantic rules in terms of relevance to a query.

The process of experiments can be summarized as follows:

1. A sample query and a set of data in terms of EDB, IDB, and IC are used as input to the language bias generator algorithm **Algorithm 6.1**.
2. The output from **Algorithm 6.1** - a biased DLAB grammar pattern and the set of EDB instances are used as input to CLAUDIEN.
3. The output from CLAUDIEN system is a set of discovered semantic rules for evaluating the effectiveness and efficiency.
4. A set of discovered semantic rules is also compared against the set of rules which are generated from the weakest-constraint DLAB grammar, i.e., the grammar without any constraints. If the weakest-constraint DLAB grammar feeds to CLAUDIEN, the CLAUDIEN system must conduct exhaustive inductive derivation against the data set. (The weakest-constraint grammar is the one that will guide the CLAUDIEN to search the entire space without any constraint).

7.2 Data Sets for Experiments

The data sets for evaluating effectiveness of **Algorithm 6.1** consist of EDB, IDB, IC, and EDB instances.

EDB:

female(x) male(x)
mother(x,y) father(x,y)
daughter(x,y) son(x,y)
husband(x,y) wife(x,y)
uncle(x,y) aunt(x,y)

IDB:

grandfather(x,y)←father(x,z),parent(z,y)
grandmother(x,y)←mother(x,z),parent(z,y)
parent(x,y)←father(x,y)
parent(x,y)←mother(x,y)
grandson(x,y)←grandfather(y,x),male(x)
grandson(x,y)←grandmother(y,x),male(x)
granddaughter(x,y)←grandfather(y,x),female(x)
granddaughter(x,y)←grandmother(y,x),female(x)
sibling(y,z)←parent(x,y),parent(x,z)
sister(x,y)←sibling(x,y),female(x)
brother(x,y)←sibling(x,y),male(x)
nephew(x,y)←uncle(y,x),male(x)
nephew(x,y)←aunt(y,x),male(x)
niece(x,y)←uncle(y,x),female(x)
niece(x,y)←aunt(y,x),female(x)

IC:

←male(x),female(x)
←mother(x,y),¬female(x)
←father(x,y),¬male(x)
←uncle(x,y),¬male(x)
←aunt(x,y),¬female(x)
←father(x,y),mother(x,y)
←uncle(x,y),aunt(x,y)
←grandfather(x,y),grandmother(x,y)
←grandfather(x,y),¬male(x)
←grandmother(x,y),¬female(x)

Sample Queries

1. List the name(s) of every male child of Sue.
   ?-mother('Sue',y),male(y)
2. List the name(s) of every male child of Mike.
   ?-father('Mike',y),male(y)
3. List the name(s) of every daughter of Laura.
   ?-mother('Laura',y),daughter(y,'Laura')
4. List the name(s) of every son of Alex.
   ?-father('Alex',y)son(y,'Alex')
5. List the name(s) of every husband who is a father.
   ?-husband(x,y),father(x,z)
6. List the name(s) of every person who is a husband of himself.
   ?-husband(x,y),husband(y,x)
7. List the name(s) of every parent who is also an uncle of their children.
   ?-parent(x,y),uncle(x,y)
8. List the name(s) of every sister of Juris.
   ?-sister(x,'Juris')
9. List the name(s) of every brother of Victoria.
   ?-brother(x,'Victoria')
10. List the name(s) of every niece who is also a nephew.
    ?-niece(x,y),nephew(x,y)

For **Query 1:** ?-mother('Sue',y),male(y)
The list of ICs for subsuming is as follows:
←male(x),female(x)
←mother(x,y),¬female(x)
←father(x,y),¬male(x)
←uncle(x,y),¬male(x)
←father(x,y),¬mother(x,y)
←grandfather(x,y),¬male(x)
The grammar G_1 generated from **Algorithm 6.1**:

```
dlab_template('0-len:[mother(name,y),female(name),
                      father(name,y),male(y),
                      female(y),father(y,y),
                      uncle(y,y),grandfather(y,y)]
              ←0-len:[mother(name,y),male(y)]')
```

dlab_variable(name,1-1,['Sue'])

The set of discovered semantic rules DSR_1 is as follows::

female(Sue)←true

female(y)←mother(Sue,y)

false←mother(Sue,y),male(y)

The discovered rule female(y)←mother(Sue,y) implies that all the children of Sue are female. And the discovered rule false←mother(Sue,y),male(y) can be used to answer the query directly without physical database access.

For the query like ?-mother('Sue',y),female(y), the join operation between mother('Sue',y) and female(y) can be eliminated. The original query can be transformed into ?-mother('Sue',y) since the discovered rule female(y)←mother(Sue,y) implies that all of the children of Sue is female. The cost to answer the original query can be saved since the query involving join operation has been replaced by the query involving only select operation (table scan). Further, if there is an index on x of mother(x,y), then the index can be used for scan reduction.

The advantages of introducing Language Bias Generating Algorithm have at least two following aspects:

1. The algorithm **Algorithm 6.1** generates the biased grammar for CLAUDIEN to conduct inductive rule derivation in a much smaller space comparing with the weakest grammar without any constraint and/or bias.
2. The semantic rules discovered based on the biased grammar can be used to optimize a query semantically.

7.3 Analysis of the Experiment Results

The second column in the Table 1 lists the number of rules discovered for each grammar $G_i, 0 \leq i \leq 10$. The third column in the table lists the number of rules relevant to the given query. The set of relevant rule is a set of rules such that the predicates in the query also appear in each rule of the set. The fourth column lists the usefulness of the rule is the ratio of the number of relevant rules to the total number of discovered rules. The grammar G_0 is the weakest grammar in which both the *HeadPredicateList* and *BodyPredicateList* contain all the predicates in EDB, IDB and ICs sets. The process of rule derivation with grammar G_0 incurs the largest cost and search space as well as the largest set of derived rules.

For the rule derivation by using the list of grammars from G_1 to G_{10}, the sizes of their corresponding search spaces are much smaller than the one with grammar G_0. It shows that our algorithm **Algorithm 6.1** does generate the biased grammar that can be utilized for rule derivation in a much smaller space as well as lower cost of CPU. Although there might be the cases in which the large variances in measuring rule usefulness in Table 1 are possible, it still shows that the biased grammars have performed much better than the weakest grammar G_0 without any constraint.

8 CONCLUSIONS

We developed the language bias generator algorithm to generate biased language grammar for inductive logic programming as well as semantic rule derivations. The experiment results demonstrate both effectiveness and efficiency of our developed algorithm. For each biased grammar, the algorithm does generate a set of relevant rules to a corresponding query given from a set of discovered rules. The high percentage values in measuring rule usefulness of the grammars $G_1 - G_{10}$ support our observation of effectiveness. The values such as the size of the search space and the CPU cost also support our observation on efficiency since both the size and the CPU cost with the biased grammars are much smaller than the weakest grammar without constraint. Further experiments can be conducted in order to achieve smooth experimental results with lower variance values in different measurement computation. We believe the richness of the semantics from EDB, IDB, ICs, EDB instances, and a given query can be further explored to support semantic query optimization in intelligent database environments.

9 ACKNOWLEDGMENTS

The authors would like to express sincere thanks to Dean Edward Lieblein, the faculty and staff at School of Computer and Information Sciences, Nova Southeastern University. Thanks also go to Dr. S. Rollins Guild and Dr. Ping Tan.

References

[1] H. Adé, L. De Raedt, and M. Bruynooghe. Declarative Bias for Specific-to-General ILP systems. In *Machine Learning*, Volume 20, Number 1-2, 1995, pp. 119- 154.

[2] Y. Cai, J. Han, and N. Cercone. An Attribute-Oriented Approach for Learning Classification Rules from Relational Databases. In *Proceedings of IEEE the* 6^{th} *International Conference on Data Engineering*, 1990, pp. 281-288.

[3] U. S. Charkravarthy, J. Grant, and J. Minker. Logic-Based Approach to Semantic Query Optimization. In *ACM Transactions on Database Systems*, Volume 15, Number 2, 1990, pp. 162-207.

[4] U. S. Charkravarthy, J. Grant, and J. Minker. Foundations of Semantic Query Optimization for Deductive Databases. In *Foundations of Deductive Databases and Logic Programming*, Edited by J. Minker, Morgan Kaufmann Publishers, Los Altos, CA, USA,1986, pp. 243-273.

[5] U. S. Charkravarthy, *Semantic Query Optimization in Deductive Databases*, Ph.D. Thesis, Department of Computer Science, University of Maryland, College Park, MD, USA, 1985.

[6] Shan-Hwei Nienhuys-Cheng and Ronald de Wolf. *Foundations of Inductive Logic Programming*, Springer-Verlag, 1997.

Grammar	# of Rules Discovered	# of Rules Relevant	Usefulness of the Discovered Rules	CPU Time	Size of the Search Space
G_0	83	0.5*	0.60%	70.1	1.0995×10^{12}
G_1	3	2	66.6%	0.25	1024
G_2	3	2	66.6%	0.20	1024
G_3	2	1	50.0%	0.10	32
G_4	2	1	50.0%	0.12	32
G_5	3	1	33.3%	0.08	32
G_6	1	1	100.0%	0.05	4
G_7	7	2	28.6%	0.87	2048
G_8	10	1	10.0%	1.83	65536
G_9	10	1	10.0%	1.55	65536
G_{10}	9	3	33.3%	5.5	262144

Table 1: Measurement of relevance of derived rules w.r.t. the corresponding queries

[7] L. De Raedt and L. Dehaspe. Clausal Discovery. In *Machine Learning*, Volume 26, Number 2/3, 1997, pp. 99-146.

[8] L. Dehaspe and L. De Raedt. DLAB: A Declarative Language Bias Formalism. In *Proceedings of the International Symposium on Methodologies for Intelligent Systems (ISMIS96)*, Springer-Verlag, 1996.

[9] L. Dehaspe, W. Van Laer, L. De Raedt. CLAUDIEN: The Clausal Discovery Engine User's Guide 3.0. Technical Report CW 239, Department of Computer Science, Katholieke Universiteit Leuven, 1996.

[10] Sašo Džeroski and Nada Lavrač. Inductive Learning in Deductive Databases. In *IEEE Transactions on Knowledge and Data Engineering*, Volume 5, Number 6, 1993, pp. 939-949.

[11] M. M. Hammer and S. B. Zdonik. Knowledge Based Query Processing. In *Proceedings of the* 6^{th} *International Conference on Very Large Data Bases*, 1980, pp. 137-147.

[12] J. Han, Y. Cai, and N. Cercone. Data-Driven Discovery of Quantitative Rules in Relational Databases. In *IEEE Transactions on Knowledge and Data Engineering*, Volume 5, Number 1, 1993, pp. 29-40.

[13] J. Han, Y. Cai, and N. Cercone. Knowledge Discovery in Databases: An Attribute-Oriented Approach. In *Proceedings of the* 18^{th} *Very Large Data Bases Conference*. 1992, pp. 547-559.

[14] J. J. King. QUIST: A System for Semantic Query Optimization in Relational Databases. In *Proceedings of the* 7^{th} *International Conference on Very Large Data Bases*, 1981, pp. 510-517.

[15] J. J. King, *Query Optimization by Semantic Reasoning*, Ph.D. Thesis, Department of Computer Science, Standard University, CA, USA,1981.

[16] J. W. Lloyd. *Foundations of Logic Programming,* 2^{nd} *edition*, Springer-Verlag, 1987.

[17] S. Muggleton and L. De Raedt. Inductive Logic Programming: Theory and Methods. In *Journal of Logic Programming*, Volume 19-20, 1994, pp. 629-679.

[18] J. R. Quinlan and R. M. Cameron-Jones. Induction of Logic Programs: FOIL and Related Systems. In *New Generation Computing*, 13(304), 1995, pp. 287-312.

[19] J. R. Quinlan. Learning Logical Definitions from Relations. In *Machine Learning*, Volume 5, Number 3, 1990, pp. 239-266.

[20] S. T. Shenoy and Z. M. Ozsoyoglo. Design and Implementation of a Semantic Query Optimizer. In *IEEE Transactions on Knowledge and Data Engineering*, Volume 1, Number 3, 1989, pp. 345-361.

[21] M. D. Siegel, E. Sciore, and S. Salveter. A Method for Automatic Rule Derivation to Support Semantic Query Optimization. In *ACM Transactions on Database Systems*, Volume 17, Number 4, 1992, pp. 563-600.

[22] M. D. Siegel, *Automatic Rule Derivation for Semantic Query Optimization*, Ph.D. Thesis, Computer Science Department, Boston University, MA, USA, 1989.

[23] W. Sun and C. T. Yu. Semantic Query Optimization for Tree and Chain Queries. In *IEEE Transactions on Knowledge and Data Engineering*, Volume 6, Number 1, 1994, pp. 136-151.

[24] C. T. Yu and W. Sun. Automatic Knowledge Acquisition and Maintenance for Semantic Query Optimization. In *IEEE Transactions on Knowledge and Data Engineering*, Volume 1, Number 3, 1989, pp. 363-375.