A Framework for Knowledge Discovery and Evolution in Databases

Jong P. Yoon and Larry Kerschberg

***Abstract*—Although knowledge discovery is increasingly important in databases, discovered knowledge is not always useful to users. It is mainly because the discovered knowledge does not fit the user's interests, or it may be redundant or inconsistent with *a priori* knowledge. Knowledge discovery in databases depends critically on how well a database is characterized and how consistently the existing and discovered knowledge is evolved.**

This paper describes a novel concept for knowledge discovery and evolution in databases. The key issues of this work include: using a database query to discover new rules; using not only positive examples (answer to a query) but also negative examples to discover new rules; harmonizing existing rules with the new rules. The main contribution of this paper is the development of a new tool for 1) characterizing the exceptions in databases and 2) evolving knowledge as a database evolves.

***Index Terms*—Active database evolution, database mining, expertise transfer, knowledge discovery, knowledge refinement.**

I. Introduction

Although knowledge discovery is increasingly important in databases, discovered knowledge is not always useful to users. It is mainly because the discovered knowledge does not necessarily fit a user's interests, and may be redundant or inconsistent with *a priori* knowledge. Knowledge discovery critically depends on how well a database is characterized and how consistently the existing and discovered knowledge is evolved.

This paper makes use of database techniques to discover new rules. A database represents facts about the real world and by its nature provides many attractive features. A database models the world in a structured and organized manner (e.g., relational/object-oriented modeling), and implicitly contains knowledge, knowledge which abstracts the data. That is, the organization of data results from codifying knowledge in the data such as relationships among attributes and key constraints. The data can be retrieved in accordance with user's interests by means of queries. A query is the request for a subset of the database.

The key issues addressed in this work include: 1) using a database query to discover new rules; 2) using not only positive examples (the answer to a query) but also negative examples to discover new rules; 3) harmonizing or reconciling both the new rules and the existing rules to maintain consistency of the knowledge and database.

A criterion for knowledge discovery from databases is the determination of how well rules X_i can characterize a database S. For example, suppose S includes the tables, "enrollment" and "transcript." Let X_1, X_2 and X_3 be, respectively, the discovered rules "the students enrolled in CS714 have a GPA over 3.3," "the students enrolled in CS714 have taken CS120," and "the students enrolled in CS714 have taken CS525." This discovery is not always interesting to users, especially for someone who poses the query "List the course names that all students enrolled in CS714 have taken." In this case, only X_2 and X_3 are useful because this query asks for the course names. If there exists in S the rule K such that "Students in the CS department have taken CS120," X_3 is the only useful discovered rule because X_2 is redundant with K.

The problems that motivate our research are: 1) expert user's domain knowledge has not been used to discover new rules. User's interests, intentions, insights, or background knowledge are conveyed by means of a query to support knowledge discovery, which delimits the learning space. Unless a user's interests or intentions are used to discover rules, the discovered rules may not be useful for his applications. 2) Rules are discovered from only positive examples. Without using negative examples, however, "constraints" cannot be discovered efficiently because constraints characterize the exceptions. 3) The discovered rules may be redundant or inconsistent with the existing rules. Discovery of those rules is time-consuming. A rule inconsistent with the existing rules is of no use or may be misleading.

The goal of this paper is 1) to discover the rules which match and support a user's interests, and 2) to harmonize the discovered rules to be consistent with the existing rules. The main contribution of this paper is a new method for 1) characterizing the exceptions in databases, and 2) evolving knowledge as a database evolves.

The organization of this paper is as follows. Section II investigates the related work. Section III reviews the basic terms of relational database systems. In Section IV, a novel concept of knowledge discovery is developed. A knowledge discovery algorithm is also defined. Section V harmonizes the discovered rules with the existing rules. Finally, the contributions of this paper and future work are described in Section VI.

II. Related Work

Until recently, the only methodology available about reasoning from databases has been based on statistical methods [2], [20]. For example, Smyth and Goodman [20] have introduced the form of the probabilistic rule "IF $Y = y$ then $X = x$ with probability p," in which the probability $p(x|y)$ is added to the rule. However, statistical or probabilistic methods are not always efficient for evolutionary systems because of either the inflexible statistical assumptions, such as the adherence to a particular probability distribution model, or the limitation of the statistical approach, such as the inability to recognize the relationships among data.

Structural information about databases is used to discover knowledge [3], [18], [21]. Quinlan [18] has introduced a decision tree for classification. Using a tree structure makes it easier to search the rules. However, this approach does not function efficiently when data are inconclusive, or when there are a few positive data and many more negative data. Cai *et al.* [1] have determined the generalization hierarchy of attribute values. The attribute domains are defined in the given generalization hierarchy and, in turn, used to generalize the attribute values in a table. The strong assumption is the user's predefined domain hierarchy on which the generalization depends.

A few papers describe the assessment of the discovered rules. Rules are measured by calculating probabilities $p(A)$, $p(B)$, $p(A\&B)$ for $A \rightarrow B$ [7], [20]. Schlimmer *et al.* [19] have described a notion of justifying rules to suggest the refinements of the rules. Piatetsky-Shapiro [16] has discussed the expected accuracy of the dis-

Manuscript received October 1, 1992; revised June 15, 1993. This research was conducted in the Center for Artificial Intelligence at George Mason University, and was supported in part by the Defense Research Projects Agency under Grants N00014-91-J-1584 and N0014-92-J-4038, administered by the Office of Naval Research.

The authors are with the Center for Artificial Intelligence, School of Information Technology and Engineering, George Mason University, Fairfax, VA 22030.

IEEE Log Number 9212798.

Algorithm: Knowledge Discovery /* e.g., $a = V$ */
Input: An SQL query.
Output: A set of rules.
Method: (Assume that an SQL query is presented and the positive examples A and the negative examples $\bar{A}$ are generated in the form of table.)
/* A and $\bar{A}$ are generated according to Definition 1. */
Let q be a *disjunctive normal form* of predicates, the condition of the SQL query.
Let S be a set of attributes $\{a_1, a_2, \ldots, a_i, \ldots, a_n\}$ specified in SELECT part of the SQL query.
Let S' be a set of join attributes $\{a'_1, a'_2, \ldots, a'_j, \ldots, a'_m\}$ specified in WHERE part.
/* If there is no join attribute, S is empty, i.e., a query is posed to only a single table. */
Let n be the number of attributes in S

for each a_i in S **do**
begin

1. Let a_i be the "target-attribute";
   /* a'_j can be treated as a target-attribute if the WHERE clause specifies $a_i = a_j$. */
2. **if** S' is not empty
   then let a'_j's in S' be the "reference-attributes" and V'_j be a set of the "instance values" for each a'_j;
3. Group A and $\bar{A}$ by a_i;
4. Let s and t denote the number of decomposed tables from A and $\bar{A}$, respectively. Let each sub-table grouped from A be A_k and each sub-table grouped from $\bar{A}$ be $\bar{A}_l$;
5. **for each** $A_k, (1 \leq k \leq s)$, **do** /* for positive examples */
   begin
   (a) **if** S' is not empty /* if tables are joined */
   then if A_k contains all the instance values V'_j for an a'_j
   then the rule is the form "$q \rightarrow a_i = V_i$"
   else if $|\ A_k\ | = |\ A\ |$ /* if a single table is considered */
   then the rule is the form "$q \rightarrow a_i = V_i$";
   (b) **if** there exists no the value V_i for the attribute a_i in any $\bar{A}_l$
   then the rule is "$a_i = V_i \rightarrow q$"
   end
6. **for each** $\bar{A}_l, (1 \leq l \leq t)$, **do** /* for negative examples */
   if S' is not empty **and** $\bar{A}_l$ contains all the instance values V'_j for an a'_j
   then the rule is the form "$q \rightarrow \neg(a_i = V_i)$"
   else if $|\ \bar{A}_l\ | = |\ \bar{A}\ |$
   then the clause is the form "$q \rightarrow \neg(a_i = V_i)$";

end

Note that $|\ A\ |$ denotes the size of the table A.

Fig. 1. The algorithm of knowledge discovery.

Theorem 1 (Knowledge Discovery from Positive and Negative Examples): Consider a database consisting of a set of tables $\mathfrak{I}$. If a clause (r) is satisfied with an answer $A(\subseteq \mathfrak{I})$ for a query condition (q), then $q \rightarrow r$ holds. If a clause (r') is not satisfied with $A(\subseteq \mathfrak{I})$ for q, then $q \wedge r' \rightarrow$ holds. □

Proof: Let A and $\bar{A}$ be positive and negative database instances respectively, and Q be a query. Because r is satisfied with A which is for q, "$A, q \vdash r$" holds (see [17] for details). That is, query processing in a database yields an answer. Hence, $A \vdash \neg q \vee r$ holds. Positive examples deduce the implication $q \rightarrow r$. For negative instances, because r is not satisfied with A which is for q, it is satisfied with $\bar{A}$ which is not for q. In other words, query processing yields no answer. The deduction of "no" answer is the deduction of the negation of the answer under the closed world assumption. Therefore, $\bar{A}, q \vdash \neg r'$ holds. Hence, $\bar{A} \vdash \neg q \vee \neg r'$ holds, so does $q \wedge r' \rightarrow$. □

A. Knowledge Discovery Algorithm

We specify in Figure 1 the algorithm to discover rules. The motivation behind this algorithm is that *it is simple to characterize a decomposed table if this decomposed table preserves its characterizations.* The *characterization* is an equality predicate describing an attribute (strictly speaking, a target attribute) in a table. The target attributes are the attributes used in a user's query, and, in turn, used to discover a rule interesting to the user. Characterizations are preserved if all the instance values of a join attribute are shown in the decomposed table.

A table of either positive or negative examples is decomposed through grouping by an attribute. When an answer is obtained by a "join" operation between tables (or relations), the join attribute, which is in both tables and used to join them, is considered to check if the decomposed table preserves characterizations. We call these join attributes "reference attributes."

If a table is not decomposed because all the values for the target attribute are the same in the table, denoted as $|A| = |\bar{A}|$, knowledge can be discovered as "target attribute = value" simply, where $|A|$ denotes the size of the table A. If all the instance values of reference attributes are shown in the decomposed positive example, then we may have the rule: $q \rightarrow r$, where q denotes the condition of a query and r is in the form of "target-attribute = value" the charcovered rules by using a statistical function about the number of tuples.

There is a significant difference between our approach and the previous research [2], [3], [6], [9], [11], [20], [21]. The previous research has concentrated on how artificial intelligence can help in knowledge discovery, without considering the characteristics of databases. As shown in the diagonal vector (2′) of the following diagram, logically, if new rules $\mathcal{X}$ are discovered from a database $\mathcal{S}$ (i.e., $\mathcal{S} \vdash \mathcal{X}$), then $\mathcal{X}$ is simply added to the existing rules $\mathcal{K}$, so $\mathcal{K} \cup \mathcal{X}$ will be the knowledge base.

$$\begin{tikzcd}
\mathcal{X} \arrow[r, "(3)"] & \mathcal{K} \arrow[d, dash] \\
\mathcal{A} \arrow[u, "(2)"] & \mathcal{S} \arrow[l, "(1)"] \arrow[ul, "(2')"']
\end{tikzcd}$$

On the other hand, our approach makes use of the database technology. A query is used to access a database. (Notice that rather than developing a new learning command, an ordinary SQL query language is used.) The query answer $\mathcal{A}$ is generated, as depicted by the vector (1) in the diagram. A query embodies the user's interests, as does the answer. Rules are discovered from the answer (refer to (2) in the diagram) and the discovered rules are harmonized, or reconciled, with the existing rules [refer to (3)]. For an answer $\mathcal{A}$, a subset of $\mathcal{S}$, $\mathcal{A} \subseteq \mathcal{S}$, if new rules $\mathcal{X}$ are discovered from $\mathcal{A}$ and $\mathcal{X}$ is satisfied with $\mathcal{K}$ (i.e., $\mathcal{A} \vdash \mathcal{X}$, $\mathcal{X} \models \mathcal{K}$), then the new knowledge base will be $\mathcal{K} \cup \mathcal{X}$.

III. Preliminaries

A relational database is a collection of n-ary relations which are represented as tables. All data are treated as being stored in tables. Each row in the table summarizes information about some object or represents a relationship among the objects. A row is called a *tuple* and a column name is called an *attribute*. For example, the following table "Teaches" contains four tuples, each of which represents the three attributes "instructor", "course," and "dept."

Teaches

instructor	course	dept
Adam	CS714	CS
Fox	CS622	CS
Newman	EE692	EE
Wiseman	EE705	EE

From database tuples, concepts or regularities can be discovered and expressed as rules representing either constraints or deductive rules. A rule is defined as a disjunction of predicates in a first-order Horn clause logic: $L_1 \,\&\, L_2 \,\&\, \cdots \,\&\, L_n \rightarrow L_0$, where L_i is a predicate. For example, in the relational schemes Enrollment(student, course) and Transcript(student, semester, course, GPA), the rule "Students enrolled in CS714 have taken CS525" can be represented as:

Enrollment.course = *CS714* → *Transcript.course* = *CS525*.

Rules can be interpreted by a model-theoretic approach: the rule $p \rightarrow q$ (or $\neg p \vee q$) holds when either both p and q are true, or p is not true (no matter what q is). Therefore, $p \rightarrow q$ results in the three models (three true interpretations): $\{p, q\}$, $\{\neg p, q\}$, and $\{\neg p, \neg q\}$.

Those data and rules investigated above are accessed by a user's query. A query specifies a subset of a database according to user's interests. In SQL, a query specification is of the form "SELECT attributes FROM table names WHERE a conjunction of predicates." As a query conveys the user's intent, the answer to the query is also of interest to the user. During query processing, the tables in the FROM part of a query are joined if necessary. Query processing yields a table, which is an answer, called "positive examples." The other tuples of those tables, not included in the answer, are called "negative examples." This notion is very important in that rules are discovered not only from the true tuples but also from the false tuples [13]. We define positive and negative examples as follows:

Definition 1 (Positive and Negative Examples): Consider a database consisting of a set of tables $\mathfrak{Z} = \{R_1, R_2, \cdots, R_n\}$, *where each table* R_i *is a set of instances. Let A be the answer to a query Q, a set of the instances resulting from query processing among the tables* $\mathcal{R} (\subseteq \mathfrak{Z}) = \{R_1, R_2, \cdots, R_m\}$, *where* $m \leq n$. *The instances in A satisfy the condition q of a query Q:* $A = \sigma_q(R_1 \times R_2 \times \cdots \times R_m)[attri_A]$, *where* σ_q *denotes a selection of the instances satisfying q,* $R_i \times R_j$ *denotes the Cartesian product between* R_i *and* R_j, *and* $[attri_A]$ *denotes a projection of the attributes from the table A. We call A positive examples. The negative examples* $\bar{A}$ *is defined as a set of tables, each table is denoted* $\bar{A}_i$, *such that* $\bar{A}_i$ *is a subset of* R_i *which does not include the instances included in A. That is,* $\bar{A}_i = R_i - (A \bowtie R_i)[attri_{R_i}]$, *where* $R_i \bowtie R_j$ *denotes "join" operation between* R_i *and* R_j. *Clearly,* $\bar{A} = \{\bar{A}_1, \bar{A}_2, \cdots, \bar{A}_m\}$. □

For example, for the following query, selecting all attributes of the table Teaches for the CS department, the positive and negative examples are:

```
SELECT   *
FROM     Teaches
WHERE    dept = CS
```

positive-examples

instructor	course	dept
Adam	CS714	CS
Fox	CS622	CS

negative-examples

instructor	course	dept
Newman	EE692	EE
Wiseman	EE705	EE

IV. Knowledge Discovery from Databases

In our approach, knowledge discovery is performed when a query is posed. For a query, the answer is generated. New rules are discovered from the answer (or true database instances) which is a subset of the database. These rules are satisfied by all the true instances (i.e., positive examples) but not satisfied by any false instances (i.e., negative examples). Knowledge can be discovered from positive and negative examples. This discovered knowledge should be satisfied with all the positive examples but none of negative examples. This notion is defined as follows:

Definition 2 (Discovered Knowledge): Consider a database consisting of a set of tables $\mathfrak{Z}$. *Let A be positive examples and* $\bar{A}$ *be negative examples to a query. The knowledge X discovered from A and* $\bar{A}$ *should have the following property:*

$$\forall t \in A,\ t \models X \quad \textit{and} \quad \forall t \in \bar{A},\ t \models X. \qquad \square$$

Not only the positive examples but also the negative examples may be characterized. The rules characterizing the positive examples represent dependencies among attributes in the table. The rules characterizing the negative examples deal with the exceptions to the query, that is, those database instances that do not satisfy the query conditions.

Enrollment

sname	course
John	CS622
John	CS714
Lance	IS511
Paul	CS714
Paul	CS722
Sam	CS714

Chair

cname	dept	rank
Smith	CS	Visiting Professor
Richard	IS	Professor
Wiseman	EE	Professor

Transcript

sname	semester	course	grade
Andy	SP90	EE120	C
John	FL91	CS722	A
John	FL91	CS525	A
John	SP91	CS611	C
John	SP90	CS120	B
Lance	SS90	CS129	B
Paul	FL90	CS120	A
Paul	FL91	CS525	A
Paul	FL91	CS622	B
Sam	FL91	CS525	B
Sam	SS90	CS120	B

Teaches

instructor	course	dept
Adam	CS714	CS
Brown	CS722	CS
Fox	CS622	CS
Newman	EE692	EE
Richard	IS511	IS
Wiseman	EE705	EE

Fig. 2. A university database example.

$\overset{Answering}{\Longrightarrow}$

positive-examples

t.course	c.dept	c.cname	c.rank
EE705	EE	Wiseman	Professor
IS511	IS	Richard	Professor

negative-examples

t.instructor	t.course	t.dept
Newman	EE692	EE
Fox	CS622	CS
Adam	CS714	CS
Brown	CS722	CS

negative-example

c.cname	c.dept	c.rank
Smith	CS	Visiting Professor

(1)

$\overset{TableDecomp.}{\Longrightarrow}$

positive-example

t.course	c.dept	c.cname	c.rank
EE705	EE	Wiseman	Professor

positive-example

t.course	c.dept	c.cname	c.rank
IS511	IS	Richard	Professor

negative-examples

t.instructor	t.course	t.dept
Newman	EE692	EE

negative-examples

t.instructor	t.course	t.dept
Fox	CS622	CS
Adam	CS714	CS
Brown	CS722	CS

negative-example*

c.cname	c.dept	c.rank
Smith	CS	Visiting Professor

(2)

$\overset{ClauseGen.}{\Longrightarrow}$ r_2 or r_2'

Fig. 3. The knowledge discovery process 1.

acterizations of the decomposed table. If the value of a target attribute does not appear in any negative examples, the rule $r \rightarrow q$ can be generated. These two rules, $q \rightarrow r$ and $r \rightarrow q$, explain that any instance satisfying q satisfies r. However, the rule $r \rightarrow q$ cannot be generated from negative examples because the safety problem.[1]

The decomposition of a table is performed efficiently using sort algorithms. We observe that using the *quick sort* for the inner loop (#3 in the algorithm) takes the average time complexity of $O(n \log n)$, where n is the number of tuples. The outer loop takes only $O(m)$, where m is the number of attributes, and $m \ll n$ in databases.

B. Example

Consider a university database example in Figure 2. Suppose the following query $Q1$ is presented to list List the course names, department name, and a department chairperson who teaches courses.

```
SELECT   t.course, c.dept, c.name, c.rank
FROM     Teaches t, Chair c
WHERE    t.instructor = c.cname
         AND c.dept = t.dept
```

As the query $Q1$ is processed on database S shown in Figure 2, the answer is generated from a natural join of *Teaches* and *Chairs* shown in the upper table of column (1) of Figure 3. Negative examples are then the remainder of the tables *Teaches* and *Chairs* as in the lower two tables of column (1). The discovered rule X can be of the form according to Theorem 1 and the clauses r_2 and r_2' are discovered from the positive example and negative examples, respectively, by applying the above discovery algorithm.

$$(c.cname = t.instructor) \wedge (c.dept = t.dept) \rightarrow r_2$$

$$(c.cname = t.instructor) \wedge (c.dept = t.dept) \rightarrow \neg r_2'$$

Consider the attribute "c.dept" (or "t.dept") as the target attribute while "t.instructor" and "c.name" as the reference attributes. The set of the domain values for *c.dept* of the positive examples are {EE, IS} in the top table while the domain values of the negative examples are {EE, CS} in the middle table and {CS} in the bottom table. Now, the tables in column (1) in Figure 3 are decomposed into subtables in terms of the target attribute. The decomposed tables are shown in column (2). The bottom table (marked by "*") preserves the domain values {CS} for the reference attribute and has the same value "Smith" for the target attribute. The remaining tables become primitive and do not preserve its characteristics. Therefore, the negative example marked by "*" characterizes: $r_2' = \{c.dept = CS\}$.

By substituting r_2' in the X above, the discovered rule is:

$$(c.cname = t.instructor) \wedge (c.dept = t.dept) \rightarrow \neg(c.dept = CS)$$

meaning that "the chairperson of the CS department does not teach any course offered by the CS department."

Furthermore, consider the attribute "c.rank" as the target attribute while "t.instructor" and "c.cname" as the reference attributes. The set of the domain values for *c.dept* are the same as above. Now, the tables in column (1) in Figure 4 are decomposed into subtables in terms of the target attribute "c.rank." The decomposed tables are shown in column (2). Notice that the middle

[1]The rule $r \rightarrow \neg q$ does not guarantee that the instances satisfying r do not satisfy only q. There are infinitely many other cases than the condition q.

$\xRightarrow{Answering}$

(1)

positive-examples

t.course	c.dept	c.cname	c.rank
EE705	EE	Wiseman	Professor
IS511	IS	Richard	Professor

negative-examples

t.instructor	t.course	t.dept
Newman	EE692	EE
Fox	CS622	CS
Adam	CS714	CS
Brown	CS722	CS

negative-example

c.cname	c.dept	c.rank
Smith	CS	Visiting Professor

$\xRightarrow{TableDecomp.}$

(2)

positive-examples*

t.course	c.dept	c.cname	c.rank
EE705	EE	Wiseman	Professor
IS511	IS	Richard	Professor

negative-example*

c.cname	c.dept	c.rank
Smith	CS	Visiting Professor

$\xRightarrow{ClauseGen.}$ r_2 or r_2'

Fig. 4. The knowledge discovery process 2.

table cannot be decomposed because it does not have the target attribute. No rule can be discovered from this table. We know that the decomposed tables (marked by "*") preserve the domain values of reference attributes and has the same value of the target attribute. Therefore, the negative example marked by "*" characterizes: $r_2 = \{c.rank = Professor\}$ and $r_2' = \{c.rank = VisitingProfessor\}$. Therefore, the following two rules are discovered.

$$(c.cname = t.instructor) \wedge (c.dept = t.dept) \rightarrow (c.rank = Professor)$$

$$(c.cname = t.instructor) \wedge (c.dept = t.dept) \rightarrow \neg(c.rank = VisitingProfessor)$$

V. Knowledge Harmonization

Although rules can be discovered from a database, only those deemed to be "useful" are considered for database knowledge. Rules are useful when they are consistent and non-redundant. The usefulness of rules depends on the version of database instances. First, the knowledge versions are defined with respect to database versions as follows.

Definition 3 (Knowledge Version): Let S_i denote a version i of database S. That is, the versions of database S are $S_1 > S_2 > \cdots > S_i > S_{i+1}$. Note that S_{i+1} is a latest version. Suppose that $S_i \cap S_{i+1} \neq \{\ \}$. Since the knowledge K_i is discovered from database S_i at a particular instant in time, $S_i \vdash K_i$, and thereafter $S_{i+1} \vdash K_{i+1}$, the version of the knowledge is correspondingly $K_1 > K_2 > \cdots > K_i > K_{i+1}$. □

As a database evolves, so do the rules characterizing the database. Both the discovered rules and the existing rules are evolved to be consistent.

In general, the concept of "*models*" of clauses are defined as true interpretation. Since a clause is interpreted by value assignment to the all variable arguments of the clause, the models of the clause contain no variable symbols as argument. Now, we extend the concept of models in order to interpret rules and constraints. Without value assignment to the variable arguments in a rule or a constraint, we can create a truth table. Notice that we do not emphasize the distinction between rules and constraints when constructing the truth table. Consider $P(x, Y) \rightarrow Q(x, Y)$ (denotes either a rule or a constraint), where x and y are variable symbols.

	$P(x, Y)$	$Q(x, Y)$
Case 1	T	T
Case 2	F	T
Case 3	F	F

Those three cases make $P(x, Y) \rightarrow Q(x, Y)$ hold. That is, the three conjunctions, $P(x, Y) \wedge Q(x, Y)$, $\neg P(x, Y) \wedge Q(x, Y)$, $\neg P(x, Y) \wedge \neg Q(x, Y)$, represent all possible models of $P(x, Y) \rightarrow Q(x, Y)$ as the variable symbols x and y denote the values in a database. We call these conjunctions *model schemas*. A model schema determines true interpretation of rules and constraints as they apply to a database.

The idea is that rules are harmonized by removing inconsistent models (e.g., [4], [5]). To resolve inconsistencies, the following theorem is proposed.

Method 1 (Knowledge Harmonization Method): Let $\Delta = \{K, X, Q\}$ be a closed theory of the possible world containing the existing rules K and the discovered rules X which correspond to a query Q. Let $E(K)$ and $E(X)$ be the models of K and X, respectively. If $\exists j E_i(K) \models E_j(X)$, then $\cup_i E_i(K) \cup_j E_j(X)$ is a set of the harmonized models. □

Rationale: Consider two versions of a database: $S_t > S_{t'}$, versions $t \neq t'$. Suppose the existing rule K was discovered from S_t: $S_t \vdash K$, similarly, a new rule X is discovered from $S_{t'}$: $S_{t'} \vdash X$. Since $t \neq t'$ and $S_t - S_{t'} = \delta(\neq \{\ \})$ by evolution, $\delta \not\subseteq S_{t'}$ and $\delta \not\vdash X$. Thus, k, for $\delta \vdash k \in K$ is outdated and so it should be eliminated. □

It is needed to obtain only the models of those existing rules consistent with at least one model of the newly discovered rules. For any model of the new rule, the models of each existing rule are ensured to be consistent. Those consistent models are shown to be evolved by the new rules.

A. Example

Consider once again the new rule (X) discovered in the previous section, "the chairperson of the CS department does not teach any course offered by the CS department:"

$$(c.cname = t.instructor) \wedge (c.dept = t.dept) \rightarrow \neg(c.dept = CS).$$

This rule can be rewritten as $Chair(x, CS), Teaches(x, y, z) \rightarrow$. Consider also rules ($K$) in Table I.

The initial clauses (Δ_K, Δ_X) for both the existing rules and the discovered rule are:

$$\Delta_K = \{\neg Faculty(x, CS) \vee \neg Teaches(x, y) \vee Offered(y, CS), \neg Chair(x, CS) \vee Faculty(x, CS)\}$$

$$\Delta_X = \{\neg Chair(x, CS) \vee \neg Teaches(x, y, CS)\}$$

These initial clauses are converted to possible models. The four model schemas $E_i(K)$ and the three model schemas $E_j(X)$ respec-

TABLE I
UNIVERSITY DATABASE RULES

$Transcript(x, CS714, A) \rightarrow Can\_Be\_TA(x_1, CS714)$
$Faculty(x_1, CS) \wedge Teaches(x_1, x_3) \rightarrow Offered(x_3, CS)$
$Chair(x_1, x_2) \rightarrow Faculty(x_1, x_2)$

tively for the existing rules and the discovered rule are obtained:

$$E_1(K) = \{Chair(x, CS), Teaches(x, y), Faculty(x, CS), Offered(y, CS)\}$$

$$E_2(K) = \{Chair(x, CS), \neg Teaches(x, y), Faculty(x, CS), Offered(y, CS)\}$$

$$E_3(K) = \{Chair(x, CS), \neg Teaches(x, y), Faculty(x, CS) \neg Offered(y, CS)\}$$

$$E_4(K) = \{\neg Chair(x, CS), Teaches(x, y), Faculty(x, CS), Offered(y, CS)\}$$

$$E_5(K) = \{\neg Chair(x, CS), \neg Teaches(x, y), Faculty(x, CS), Offered(y, CS)\}$$

$$E_6(K) = \{\neg Chair(x, CS), \neg Teaches(x, y), Faculty(x, CS), \neg Offered(y, CS)\}$$

$$E_7(K) = \{\neg Chair(x, CS), Teaches(x, y), \neg Faculty(x, CS), Offered(y, CS)\}$$

$$E_8(K) = \{\neg Chair(x, CS), Teaches(x, y), \neg Faculty(x, CS), \neg Offered(y, CS)\}$$

$$E_9(K) = \{\neg Chair(x, CS), \neg Teaches(x, y), \neg Faculty(x, CS), Offered(y, CS)\}$$

$$E_{10}(K) = \{\neg Chair(x, CS), \neg Teaches(x, y), \neg Faculty(x, CS), \neg Offered(y, CS)\}$$

$$E_1(X) = \{Chair(x, CS), \neg Teaches(x, y)\}$$

$$E_2(X) = \{\neg Chair(x, CS), Teaches(x, y)\}$$

$$E_3(X) = \{\neg Chair(x, CS), \neg Teaches(x, y)\}$$

Until the new rule was discovered, the model schemas $E_1(K), \cdots, E_{10}(K)$ were themselves consistent and well accepted in the database. However, it is likely that an inconsistency may take place as the new rule is added to the existing rules. The model $E_1(K)$ of the new rule is not consistent with any one of model schemas $E_1(X)$, $E_2(X)$, or $E_3(X)$. By applying Method 1: $E_1(K)$ is removed because it does not satisfy any model of $E_i(X)$ for $i = 1, \cdots, 3$. Therefore, the consistent model schemas, $E_i(K)$ for $i = 2, \cdots, 10$, are obtained as follows:

$$E_2(K) = \{Chair(x, CS), \neg Teaches(x, y), Faculty(x, CS), Offered(y, CS)\}$$

$$E_3(K) = \{Chair(x, CS), \neg Teaches(x, y), Faculty(x, CS) \neg Offered(y, CS)\}$$

$$E_4(K) = \{\neg Chair(x, CS), Teaches(x, y), Faculty(x, CS), Offered(y, CS)\}$$

$$E_5(K) = \{\neg Chair(x, CS), \neg Teaches(x, y), Faculty(x, CS), Offered(y, CS)\}$$

$$E_6(K) = \{\neg Chair(x, CS), \neg Teaches(x, y), Faculty(x, CS), \neg Offered(y, CS)\}$$

$$E_7(K) = \{\neg Chair(x, CS), Teaches(x, y), \neg Faculty(x, CS), Offered(y, CS)\}$$

$$E_8(K) = \{\neg Chair(x, CS), Teaches(x, y), \neg Faculty(x, CS), \neg Offered(y, CS)\}$$

$$E_9(K) = \{\neg Chair(x, CS), \neg Teaches(x, y), \neg Faculty(x, CS), Offered(y, CS)\}$$

$$E_{10}(K) = \{\neg Chair(x, CS), \neg Teaches(x, y), \neg Faculty(x, CS), \neg Offered(y, CS)\}$$

$E_2(K)$ means that the CS department chairperson does not teach courses offered by the CS department. $E_3(K)$ means that the CS department chairperson does not teach courses offered by other departments. $E_4(K)$ means that faculty in the CS department who is not the chairperson teaches the courses offered by the CS department. □

B. Discussion

A new rule and the existing rules are integrated by the knowledge consistency checking method. The outcome of this checking is a set of the model schemas each of which represents a possible world as seen above. One advantage of this approach is that rather than removing one or more rules, inconsistent model schemas are removed. As a query is presented later on, the model schemas of the query should be satisfied by those model schemas in the database. For example, suppose the model schemas above are set as constraints in the current database. Consider the insertion of a new instance "$Chair(x, CS) \wedge Teaches(x, y)$." Since any of the model schemas, $E_2(K), E_3(K), \cdots, E_{10}(K)$, does not satisfy the insertion, the insertion is not allowed so that the database consistency is maintained.

VI. CONCLUSION

This paper describes a novel concept for knowledge discovery from both positive and negative examples. The rules discovered from positive examples characterize the answer to an expert user's query. The rules discovered from negative examples characterize the exceptions in a database. These rules are used to evolve the existing rules and make the database consistent.

The benefits described in this paper include: 1) the rules more suitable to user's interests are discovered because discovery process is initiated by a query which conveys the user's interests; 2) a higher performance of discovery processing results from using answers to queries rather than using the entire database; 3) constraints which handle the exceptions of a database can be discovered when negative examples are used in knowledge discovery process; and 4) the existing rules are evolved as they are harmonized with the discovered rules.

Although this paper uses a small example, the discovery process can scale up to handle the real world large databases obtained from network fault/alarm data, satellite data, aeronautical data, or meteorological data. Some analysis of the model-theoretic interpretation approach and experimentation on other large databases will be our future research. This paper has focused on discovering rules with equality predicates. Discovery of the rules with range restricted predicates will be our future research. Moreover, the rules discovered in this paper can be represented in a concise and elegant form by generalization techniques. Discovered rules may be refined [8], [10] as a database is evolved. The approach may also be useful for research in "cooperative answers" [14], [5] in which the system responds with knowledge rather than simply providing the tuples satisfying the query.

References

[1] Y. Cai, N. Cercone, and J. Han, "An attribute-oriented approach for learning classification rules from relational databases," in *Proc. 6th Int. Conf. Data Eng.*, Los Angeles, CA, 1990, pp. 281-288.

[2] K. C. Chan and A. K. Wong, "A statistical technique for extracting classificatory knowledge from databases," in *Knowledge Discovery in Databases*, G. Piatetsky-Shapiro and W. J. Frawley, Eds. Cambridge, MA: MIT Press, 1991.

[3] D. K. Chiu, A. K. Wong, and B. Cheung, "Information discovery through hierarchical maximum entropy discretization and synthesis," in *Knowledge Discovery in Databases*, G. Piatetsky-Shapiro and W. J. Frawley, Eds. Cambridge, MA: MIT Press, 1991.

[4] N. Eisinger, H. J. Ohlbach, and A. Pracklein, "Reduction rules for resolutionbased systems," *Artificial Intell.*, vol. 50, pp. 141-181, 1991.

[5] T. Gaasterland, P. Godfrey, and J. Minker, "Relaxation as a platform for cooperative answering," *J. Intell. Inform. Syst.*, vol. 1, pp. 293-321, 1992.

[6] B. Gaines, "Extracting knowledge from data," in *Proc. IJCAI-89 Workshop Knowledge Disc. in Databases*, 1989, pp. 109-116.

[7] —, "The trade-off between knowledge and data in knowledge acquisition," in *Knowledge Discovery in Databases*, G. Piatetsky-Shapiro and W. J. Frawley, Eds. Cambridge, MA: MIT Press, 1991, pp. 491-506.

[8] A. Ginsberg, "Knowledge base refinement and theory revision," in *Proc. 6th Int. Conf. Machine Learning*, 1989, pp. 260-265.

[9] K. Kaufman, R. Michalski, and L. Kerschberg, "Mining for knowledge in databases: Goals and general description of the inlen system," in *Proc. IJCAI-89 Workshop Knowledge Disc. in Databases*, 1989, pp. 158-172.

[10] D. Kulkarni and H. Simon, "The role of experimentation in scientific theory revision," in *Proc. 6th Int. Conf. Machine Learning*, 1989, pp. 278-283.

[11] D. Lubinsky, "Discovery from databases: A review of ai and statistical techniques," in *Proc. IJCAI-89 Workshop Knowledge Disc. in Databases*, 1989, pp. 204-218.

[12] R. S. Michalski and J. B. Larson, "Incremental generation of VL1 hypotheses: The underlying methodology and the description of the program AQ11," Univ. Illinois, Dep. Comput. Sci., Tech. Rep. VIVCDCS-F-83-905, 1983.

[13] R. S. Michalski, J. G. Carbonell, and T. M. Mitchell, *Machine Learning*. Los Altos: Morgan Kaufmann, 1983.

[14] A. Motro, "FLEX: A tolerant and cooperative user interface to database," *IEEE Trans. Knowledge and Data Eng.*, vol. 2, 1990.

[15] L. M. Pereira, J. J. Alfere, and J. N. Aparicio, "Contradiction removal within well founded semantics," in *Proc. 1st Int. Workshop Logic Programming and Non-Monotonic Reasoning*, A. Nerock, W. Marek and V. S. Subrahmanian, Eds. Cambridge, MA: MIT Press, 1991, pp. 105-119.

[16] G. Piatetsky-Shapiro, "Discovery, analysis, and presentation of strong rules," in *Knowledge Discovery in Databases*, G. Piatetsky-Shapiro and W. J. Frawley, Eds. Cambridge, MA: MIT Press, 1991, pp. 229-248.

[17] A. Pirotte and D. Roelants, "Constraints for improving the generation of intentional answers in a deductive database," in *Proc. 5th Int. Conf. Data Eng.*, 1989, pp. 652-659.

[18] J. R. Quinlan, "Induction of decision trees," *Machine Learning*, vol. 1, no, 1, pp. 81-106, 1986.

[19] J. C. Schlimmer, T. M. Mitchell, and J. McDermott, "Justification-based refinement of expert knowledge," in *Knowledge Discovery in Databases*, G. Piatetsky-Shapiro and W. J. Frawley, Eds. Cambridge, MA: MIT Press, 1991, pp. 397-409.

[20] P. Smyth and R. M. Goodman, "Rule induction using information theory," in *Knowledge Discovery in Databases*, G. Piatetsky-Shapiro and W. J. Frawley, Eds. Cambridge: MA, MIT Press, 1991, pp. 159-176.

[21] W. Ziarko, "The discovery, analysis, and representation of data dependencies in databases," in *Knowledge Discovery in Databases*, G. Piatetsky-Shapiro and W. J. Frawley, Eds. Cambridge, MA: MIT Press, 1991, pp. 195-212.

Induction of Rules Subject to a Quality Constraint: Probabilistic Inductive Learning

Özden Gür-Ali and William A. Wallace

***Abstract*—Organizational databases are being used to develop rules or guidelines for action that are incorporated into decision processes. Tree induction algorithms of two types, total branching and subset elimination, used in the generation of rules, are reviewed with respect to their treatment of the issue of quality. Based upon this assessment, a hybrid approach, Probabilistic Inductive Learning (PrIL), is presented that provides a probabilistic measure of goodness for an individual rule, enabling the user to set maximum misclassification levels, or minimum reliability levels, with predetermined confidence that each and every rule will satisfy this criterion. The user is able to quantify the reliability of the decision process, i.e., the invoking of the rules, which is of crucial importance in automated decision processes. PrIL is described with its associated algorithm. An illustrative example based upon the claims process at a workers' compensation board is also presented.**

***Index Terms*—Decision support system, decision tree classifiers, induction, knowledge acquisition, machine learning, statistical quality control.**

I. Introduction

Machine learning algorithms for providing guidelines or rules for action are receiving increasing attention. The combination of more readily accessible large databases and disenchantment with the heuristic development of rules has made users realize that databases can provide knowledge in a form that can guide or even replace human decision making in routine situations. However, in order for the user to incorporate this technology into organizational decision processes, there must be a means to determine the reliability of the rules, and the capability to evaluate this reliability with respect to the costs (benefits) of incorrect (correct) actions.

In today's uncertain environment, the user needs to be able to determine when the rules are applicable, and when these rules, which were derived from past data encoded and stored in the database, cannot meet the prescribed reliability. Developing a reliability measure for a *set* of rules is not sufficient or meaningful, since an individual rule which forms the basis for a decision may not satisfy this global measure. The reliability of each rule must be greater than the minimum level required by the user.

In the following sections of the paper, various tree induction algorithms are reviewed with respect to how they treat the issue of

Manuscript received October 1, 1992; revised June 15, 1993.

The authors are with Decision Sciences and Engineering Systems, Rensselaer Polytechnic Institute, Troy, NY 12180.

IEEE Log Number 9212799.