An Efficient Algorithm for Deriving Compact Rules from Databases*

Show-Jane Yen and Arbee L.P. Chen

Department of Computer Science
National Tsing Hua University
Hsinchu, Taiwan 300, R.O.C.
Email: alpchen@cs.nthu.edu.tw

Abstract

Owing to the rapid growth in the sizes of databases, potentially useful information may be embeded in a large amount of data. Knowledge discovery is the search for semantic relationships in databases. One of the main problems for knowledge discovery is that the number of possible relationships is very large, thus reducing the search complexity is important. The relationships can be represented as rules which can be used in efficient query processing. We present a knowledge discovery technique to analyze relationships and to derive compact rules. Data are first generalized to reduce their sizes, which makes them easier to be characterized in terms of rules. A mechanism and some heuristics are then proposed to alleviate the computational complexity of the rule derivation process. Finally, an evaluation model is presented to evaluate the quality of the derived rules.

1 Introduction

Because of increasing sizes of databases, it is difficult to analyze a large amount of data to find relationships among them by human beings. Techniques have been proposed to find such relationships (or knowledge) from databases [1, 2, 4, 10, 11, 14]. The knowledge discovered can be used to answer cooperative queries [3], handle null values [12] and facilitate semantic query optimization [6, 10, 13].

In this work, we analyze relationships among data to derive rules which have full confidence. These rules describe the dependencies between an attribute (called a *target attribute*) and other attributes (called *condition attributes*) in a relation. These rules can be used to provide high-level answers to user requests [5] and to improve query processing in databases. However, if many rules are derived or if the antecedent of each rule involves many condition attributes, efficient use of these rules becomes difficult. As a compact rule set can help to reduce the cost of query processing, the derived rules are expected to be compact.

The approach of Han, et al. [4] first identifies the condition attributes for a target attribute manually and projects them out for rule derivation. This approach replaces lower-level concepts (or attribute values) of each attribute with higher-level concepts to reduce the number of distinct values in each attribute in the projected relation. The number of tuples in the projected relation can be reduced, if some tuples have the same value in each attribute due to the above-mentioned replacements. From each tuple, a rule is derived. Since the cardinality of the reduced projected relation may still be large, relatively complex rules may remain. However, if the number of rules is required to be small, there may result in over-generalization and loss of valuable information. Also, the antecedent of each rule involves all condition attributes, which may be unnecessary.

Agrawal, et al. [1] presented another approach to derive rules from a relation. This approach derives rules for some combinations of values from different attributes, which appear in the same tuple in a relation with a frequency larger than a certain *threshold*. The number of rules derived depends on the setting of the threshold. If the threshold is small, many rules may be derived. Conversely, if the threshold is large, only a few rules are generated, which may result in loss of useful information about dependencies between attributes. It is difficult to find a suitable threshold to prevent problems of extreme rule sizes. Moreover, this approach requires many relation scans to derive rules, which is computationally inefficient.

Ziarko [14] analyzed data dependencies based on *rough set theory* [8]. A *minimal* subset is found from these subsets of which the dependencies between these

Proceedings of the Fourth International Conference on Database Systems for Advanced Applications (DASFAA'95)
Ed. Tok Wang Ling and Yoshifumi Masunaga
Singapore, April 10-13, 1995

*This work was partially supported by the Republic of China National Science Council under Contract No. NSC 84-2213-E-007-007

subsets and the target attribute are functional dependency. This minimal subset and the target attribute are projected to derive rules. However, if there is no better mechanism to analyze the dependency between each subset of condition attributes and the target attribute, to find the minimal subset is time consuming. Moreover, no algorithm was provided in [14] to show how to derive rules; it may be inefficient to separate data dependencies analysis and rule derivation.

In this paper, we extract *semantic association relationships* [12] from databases and use these relationships to derive compact rules of which the number of attributes involved in the antecendent of each rule and the number of derived rules can be diminished. Thereby, the complexity of rules becomes alleviated and the dependencies between attributes can be better understood; that is, the rules can make general descriptions for the dependencies between attributes.

The rest of this paper is organized as follows. Section 2 describes the basic concepts and the relationships in databases. Section 3 presents how to reduce relation sizes, proposes a mechanisms for learning process and presents the learning algorithm. The analysis of computational complexity of our learning algorithm and a comparison with other approaches are discussed in Section 4. Finally, we conclude this paper and present directions for future research in Section 5.

2 Background Knowledge

In order to make the data easier to characterize in terms of rules, one must diminish the number of distinct values in each attribute. The domain knowledge is provided to diminish the number of distinct values in an attribute. Also, an important relationship that is used throughout the learning process are introduced in this section.

2.1 Domain concept hierarchy

For each database attribute, if an attribute domain can be represented by higher-level concepts, then a *domain concept hierarchy* can be constructed for this attribute. The domain concept hierarchy is a general-to-specific structure that organizes varied levels of abstractions relevant to an attribute. The most specific values are domain elements of an attribute and the remaining values in the hierarchy are concepts specified by domain experts.

For a numerical attribute, according to the meaning of a numerical domain, we can divide its elements into groups, and represent each group by a concept. Some concepts can be further represented by higher-level concepts.

For a nonnumerical attribute, the domain concept hierarchy can be an 'IS-A' hierarchy, such as "a junior is an undergraduate student." The domain concept hierarchy can also be a taxonomy, such as "there are many countries in a continent and there are many cities in a country."

2.2 Semantic association relationships

Semantic association relationships are associations between domain elements or concepts in different attributes. For example, suppose the relation EMPLOYEE contains attributes Degree and Salary and there exists a statistic "if degree is PHD, then salary is large," in which "PHD" is the domain element of attribute Degree and "large" is the concept specified in the domain concept hierarchy for the attribute Salary. Then, there is a semantic association from "Degree is PHD" to "Salary is large."

Semantic association degrees (SAD) are degrees of semantic associations between domain elements or concepts. In our previous research [12], we suppose the SADs were specified by database designers. In this work, we provide a mechanism to efficiently obtain the SAD from a relation by using conditional probability and set theory.

For convenience, we discuss semantic association relationships in a relation. The relationships in separate relations can be found by joining these relations and applying the same approach with a single relation.

3 Knowledge Discovery

In this section, we describe a knowledge discovery process that serves to derive compact rules about a target attribute in a relation. The knowledge discovery process includes a *generalization process* that can reduce the size of a relation and a learning process that can derive compact rules from the reduced relation.

3.1 Generalization algorithm

In a relation, if each attribute contains a large set of distinct values in the relation, the rules that are derived from the relation are not useful, because rather complex rules may be derived and each rule may characterize few tuples. Hence, in order to avoid the above situation, the number of distinct values in each attribute needs to be limited.

Generalization is a process that ascends more specific values to higher-level concepts by climbing a domain concept hierarchy. The first step of the knowledge discovery process is to perform generalization on each attribute in an *initial relation*. An initial relation is a relation formed by projecting a target attribute and the condition attributes for the target attribute from a base relation.

Suppose IR is an initial relation, A_i is an attribute in IR, k_i is the number of distinct values in attribute A_i in IR, and n is the total number of tuples in IR. A *generalization threshold* is used to control the level of domain concept hierarchy ascension. The generalization algorithm is described as follows:

1. If the ratio of the number of distinct values in attribute A_i to the total number of tuples in IR, i.e. $\frac{k_i}{n}$, is greater than a generalization threshold, then execute step 2. Otherwise, the generalization algorithm on attribute A_i terminates.

2. If there exist higher-level concepts in the domain concept hierarchy for the values in A_i, all values of A_i in IR are replaced according to their higher-level concepts by ascending the hierarchy one level. Otherwise, attribute A_i is removed from relation IR, because there is a large set of distinct values in A_i but it cannot be generalized using higher-level concepts.

3. Since distinct values may be replaced by the same higher-level concepts, the number of distinct values in A_i may be diminished and k_i may be modified.

   The algorithm repeats Steps 1-3 until the ratio is less than or equal to the generalization threshold.

♯	displace	fuelcap	mass	speed	cyl	cost
1	large	high	medium	medium	6	expensive
2	large	low	heavy	fast	6	expensive
3	medium	medium	light	fast	6	medium
4	small	low	light	slow	6	cheap
5	large	medium	medium	medium	6	expensive
6	large	medium	light	medium	6	expensive
7	small	low	light	medium	6	cheap
8	small	medium	light	slow	4	cheap
9	medium	low	medium	medium	6	medium
10	medium	high	medium	fast	6	expensive
11	medium	high	light	fast	6	expensive
12	small	high	heavy	medium	4	expensive
13	small	high	light	medium	4	cheap
14	medium	low	heavy	medium	4	expensive
15	medium	medium	medium	medium	4	medium
16	medium	high	medium	medium	4	medium
17	small	medium	medium	fast	4	medium
18	medium	medium	heavy	slow	4	expensive

Table 1: relation GCAR

After performing generalization on each attribute in an initial relation, as distinct attribute values can be substituted by the same higher-level concepts, a set of tuples may be generalized to the same tuple. Redundant tuples are eliminated to generate a *generalized relation* (GR). Suppose after completing the generalization process on relation CAR, the generalized relation GCAR is as shown in Table 1, where attribute ♯ will be explained later.

3.2 Information for learning

We propose a mechanism by which the learning process needs to scan GR only once. First of all, each tuple in GR is assigned a unique *tuple number*. The attribute ♯ in Table 1 shows the numbers of the tuples.

displace	
large	{ 1,2,5,6 }
medium	{ 3,9,10,11,14,15,16,18 }
small	{4,7,8,12,13,17 }

(a)

cost	
expensive	{ 1,2,5,6,10,11,12,14,18 }
medium	{3,9,15,16,17}
cheap	{4,7,8,13}

(b)

Table 2: (a) The tuple numbers for the same attribute value in **displace**
(b) The tuple numbers for the same attribute value in **cost**

Then, from the scan of GR, the tuple numbers for the same attribute value in an attribute, and for each combination of attribute values in the same tuple from two different attributes are recorded. For example, in the generalized relation GCAR (Table 1), the set of tuples with "medium" as a value in attribute **displace** includs {3, 9, 10, 11, 14, 15, 16, 18}, and the set of tuples with "medium" and "light" as a combination of values from attributes **displace** and **mass** includes {3, 11}. Part of the information recorded from scanning GCAR is shown in Table 2 and Table 3.

3.3 A mechanism for computing SAD

Suppose X is an *event* which is an attribute-value pair; for example, event Y=(**displace**, medium) denotes the value "medium" of attribute **displace**. A function f maps one or more events into the frequency of these events appearing in the same tuple in GR. For example, in relation GCAR, if Y=(**displace**, medium) then $f(Y)$=8.

Let N be the total number of tuples in GR. A function P maps one or more events into the probability of these events appearing in the same tuple in GR. We compute the probability $P(X)$ as:

$$P(X) = \frac{f(X)}{N} \qquad (1)$$

Let Z=(**mass**, heavy). Continuing the above example, $P(Y)$ and $P(Y,Z)$ are equal to $\frac{8}{18}$ and $\frac{2}{18}$ respectively, in which the total number of tuples in GCAR is 18.

Suppose E and H are two events. Let p be a SAD from E to H. That is, there is a degree p associated

displace / mass	large	medium	small
heavy	{2}	{14,18}	{12}
medium	{1,5}	{9,10,15,16}	{17}
light	{6}	{3,11}	{4,7,8,13}

(a)

mass / speed	heavy	medium	light
fast	{2}	{10,17}	{3,11}
medium	{12,14}	{1,5,9,15,16}	{6,7,13}
slow	{18}	{}	{4,8}

(b)

speed / cost	fast	medium	slow
expensive	{2,10,11}	{1,5,6,12,14}	{18}
medium	{3,17}	{9,15,16}	{}
cheap	{}	{7,13}	{4,8}

(c)

Table 3: (a) The tuple numbers for each combination of attribute values in the same tuple from displace and mass

(b) The tuple numbers for each combination of attribute values in the same tuple from mass and speed

(c) The tuple numbers for each combination of attribute values in the same tuple from speed and cost

with "if E then H" as discussed in Section 2.2. The degree p is a conditional probability which can be represented as:

$$p(H|E) = \frac{P(E,H)}{P(E)} \qquad (2)$$

where event E is called a *condition event* and event H a *target event*.

By expression (1), $p(H|E)$ defined in the above can be computed as:

$$p(H|E) = \frac{f(E,H)}{f(E)} \qquad (3)$$

Suppose E_1, E_2, ..., and E_k are events. A function TS maps two events into a set of tuples in which the two events appear simultaneously in GR. The frequency of events E_1, E_2, ..., and E_k appearing in the same tuple in GR can be obtained as:

$$f(E_1, E_2, ..., E_k) = Card(\bigcap\{ TS(E_1, E_2), TS(E_2, E_3), ..., TS(E_{k-1}, E_k)\}) \qquad (4)$$

where the symbol "$\bigcap$" denotes set intersection and $Card(S)$ denotes cardinality of set S.

Let H be a target event and E_1, E_2, ..., and E_k be condition events in GR. The SAD from events E_1, E_2, ..., and E_k to event H can be obtained according to expression (3):

$$p(H|E_1, E_2, ..., E_k) = \frac{f(E_1, E_2, ..., E_k, H)}{f(E_1, E_2, ..., E_k)} \qquad (5)$$

For example, in relation GCAR, let events E_1 = (**displace**, medium), E_2 = (**mass**, medium) and E_3 = (**speed**, medium) be condition events and event H = (**cost**, medium) be a target event. Then, the frequency of events E_1, E_2 and E_3 appearing in the same tuple in relation GCAR can be computed according to expression (4):

$$\begin{aligned} & f(E_1, E_2, E_3) \\ = \; & Card(\bigcap\{TS(E_1, E_2), TS(E_2, E_3)\}) \qquad (6) \end{aligned}$$

From Tables 3(a) and 3(b), TS(E_1,E_2) = {9,10,15,16} and TS(E_2,E_3) = {1,5,9,15,16}.

$$\begin{aligned} & f(E_1, E_2, E_3) \\ = \; & Card(\bigcap\{\{9, 10, 15, 16\}, \{1, 5, 9, 15, 16\}\}) \\ = \; & Card(\{9, 15, 16\}) \\ = \; & 3 \qquad (7) \end{aligned}$$

Similarly, from Table 3(c), TS(E_3,H) = {9,15,16}; the frequency of events E_1, E_2, E_3 and H appearing in the same tuple in relation GCAR is:

$$\begin{aligned} & f(E_1, E_2, E_3, H) \\ = \; & Card(\bigcap\{TS(E_1, E_2), TS(E_2, E_3), TS(E_3, H)\}) \\ = \; & Card(\bigcap\{\bigcap\{TS(E_1, E_2), \\ & TS(E_2, E_3)\}, TS(E_3, H)\}) \\ = \; & Card(\bigcap\{\{9, 15, 16\}, \{9, 15, 16\}\}) \\ = \; & Card(\{9, 15, 16\}) \\ = \; & 3 \qquad (8) \end{aligned}$$

Therefore, the SAD from events E_1, E_2 and E_3 to event H can be computed according to expression (5):

$$p(H|E_1, E_2, E_3) = \frac{f(E_1, E_2, E_3, H)}{f(E_1, E_2, E_3)} = 1 \qquad (9)$$

This result signifies that the SAD from combination (**displace**, medium), (**mass**, medium) and (**speed**, medium) of condition events to event (**cost**, medium) is 1. This semantic association is later regarded as a rule "IF **displace**=medium AND **mass**=medium AND **speed**=medium THEN **cost**=medium"; the set of tuples corresponding to this rule is {9, 15, 16}.

3.4 Learning algorithm

Suppose there are k condition attributes and a target attribute in GR. Initially, in the first iteration of the learning algorithm, the SAD from each condition event to each target event is computed. Subsequently, in the lth iteration, the SAD from each combination of l condition events to each target event is computed, in which $1 \leq l \leq k$.

Suppose there are k condition attributes CA_1, CA_2, ... and CA_k and a target attribute TA in GR. Let CE denotes the combination (CA_i, ca_i), (CA_{i+1}, ca_{i+1}), ..., (CA_j, ca_j) of condition events. We have the following strategies to be used in the learning algorithm.

Strategy 1. If the SAD from CE to event (TA, ta) is 1, a rule: "IF $CA_i = ca_i$ AND $CA_{i+1} = ca_{i+1}$ AND ... AND $CA_j = ca_j$ THEN $TA = ta$" is derived. Otherwise, no rule will be derived.

The SAD can be regarded as a confidence associated with the semantic association. As we seek to derive rules which have full confidence, only the semantic association of which degree is 1 becomes a rule in each iteration.

Suppose $\Im_R$ is a set of tuples corresponding to rule R and R_1, R_2, ... and R_{v-1} are previously derived rules.

Strategy 2. When rule R_v is derived, if the union of $\Im_{R_1}$, $\Im_{R_2}$, ..., $\Im_{R_v}$ contains all tuples in GR, the learning algorithm terminates.

If there are k condition attributes in GR, there are at most k iterations in the learning algorithm.

Strategy 3. When rule R_v is derived, if $\Im_{R_i}$ $(1 \leq i \leq v-1)$ is a proper subset of $\Im_{R_v}$, rule R_i is discarded.

Because all tuples corresponding to rule R_i are characterized completely by rule R_v, rule R_i is redundant which should be discarded.

Strategy 4. If $\Im_{R_v}$ is a subset of the union of $\Im_{R_1}$, $\Im_{R_2}$, ... and $\Im_{R_{v-1}}$, rule R_v is discarded.

Because all tuples corresponding to rule R_v are characterized completely by previously derived rules, rule R_v is redundant.

Strategy 5. If the SAD from CE to event (TA, ta) is 0, the SADs from the combinations which contain CE to event (TA, ta) need not be computed in later iterations.

In this case, CE is independent of target event (TA, ta). Hence, any combination which contains CE is also independent of target event (TA, ta). The SAD from the combination to target event (TA, ta) must be 0.

Strategy 6. Continuing the case in Strategy 1, CE need not be combined with other events to find SADs from these combinations which contain CE to any target event in later iterations.

If CE is combined with other events such that some rules are derived, the sets of tuples corresponding to these rules must be subsets of the set of tuples corresponding to the rule derived in Strategy 1. Moreover, if the SAD from CE to a target event is 1, then the SADs from CE to other target events must be 0.

Strategy 7. When rule R_v is derived, if all tuples in which target event (TA, ta) appears in GR are contained in the union of $\Im_{R_1}$, $\Im_{R_2}$, ..., $\Im_{R_v}$, the SAD from any combination of condition events to target event (TA, ta) need not be computed later.

When all tuples which contain a certain target attribute value in GR are characterized completely by derived rules, no rules about this target attribute value need to be further derived.

Strategies 5, 6 and 7 are heuristics to decrease the number of computations of SADs. The learning algorithm follows:

Algorithm LCR (Learning Compact Rules)
for $l = 1$ **to** k **do**
begin
for each combination $CA_i, ..., CA_j$ selected from $CA_1, CA_2, ..., CA_k$ of size l **do**
DeriveRules($CA_i, ..., CA_j, TA$)
if all tuples in GR are characterized by derived rules
then terminate (Strategy 2)
end

DeriveRules($CA_i, ..., CA_j, TA$)
for each combination $ca_i, ..., ca_j$, denoted φ, of values in attributes $CA_i, ..., CA_j$, respectively, **do**
begin
for each value ta in attribute TA **do**
begin
if a combination which is contained in φ, value ta, or the semantic association from a combination which is contained in φ to value ta has not been removed
then
begin
compute SAD from φ to ta and
if SAD is 1
then
begin
derive a rule "IF $CA_i = ca_i$ AND ... AND $CA_j = ca_j$ THEN $TA = ta$" (Strategy 1), record the set of tuples corresponding to this rule, remove φ (Strategy 6), and
if the set of tuples corresponding to a previously derived rule is a proper subset of the set of tuples corresponding to this rule
then discard the previously derived rule (Strategy 3)
else
if the set of tuples corresponding to this rule is a subset of the union of the sets of tuples corresponding to previously derived rules
then discard this rule (Strategy 4)
end
if the union of the sets of tuples corresponding to this rule and previously derived rules of which the consequentsare "$TA = ta$" contains all tuples in which the value ta appears in GR
then remove value ta from attribute TA (Strategy 7)
else
if SAD is 0
then remove the semantic association from φ to value ta (Strategy 5)
end
end
end

A simple example is given below to demonstrate the LCR algorithm.

Example: Consider the relation GCAR shown in Table 1. We want to analyze the relationships between target attribute **cost** and other attributes in GCAR and derive compact rules about attribute **cost** from GCAR.

In the first iteration of the LCR algorithm, the SAD from each condition event to each target event

in GCAR is obtained. Tables 4(a) and 4(b) show SADs from each value in condition attributes **displace** and **mass** to each value in target attribute **cost**, respectively. The SADs from event (**displace**, large) and from event (**mass**, heavy) to event (**cost**, expensive) are both 1. Hence, the rules "IF **displace**=large THEN **cost**=expensive" and "IF **mass**=heavy THEN **cost**=expensive" are derived and recorded as rules 1 and 2, respectively. The set of tuples corresponding to rule 1 is {1, 2, 5, 6} and the set of tuples corresponding to rule 2 is {2, 12, 14, 18}. Condition events (**displace**, large) and (**mass**, heavy) are then removed, because the two events need not be combined with other condition events in later iterations according to Strategy 6.

Table 4(c) shows the SAD from each combination of values in **displace** and **mass** to each value in **cost**. Because events (**displace**, large) and (**mass**, heavy) have been removed, there are four combinations of values in **displace** and **mass**.

Similarly, the rules "IF **displace**=small AND **mass**=medium THEN **cost**=medium" and "IF **displace**=small AND **mass**=light THEN **cost**=cheap" are derived and recorded as rules 3 and 4. The set of tuples corresponding to rule 3 is {17} and the set of tuples corresponding to rule 4 are {4, 7, 8, 13}. From Table 2(b), the set of tuples in which event (**cost**,cheap) appears is also {4, 7, 8, 13}. The target event (**cost**,cheap) is then removed according to Strategy 7, because all tuples which contain attribute value (**cost**,cheap) in GR are characterized completely by rule 4. Also, SADs from any combination of condition events to target event (**cost**, cheap) need not be computed later.

In the first iteration, the SADs from event (**displace**, medium) and from event (**mass**, medium) to event (**cost**, cheap) are both 0, respectively. Hence, the SAD from each combination which contains event (**displace**, medium) or event (**mass**, medium) to event (**cost**, cheap) need not be computed in later iterations according to Strategy 5. The entry of SADs from any combination of condition events which contains event (**displace**, medium) or event (**mass**, medium) to event (**cost**, cheap) is marked "×", as shown in Table 4(c).

	cost		
displace	expensive	medium	cheap
large	1	0	0
medium	$\frac{1}{2}$	$\frac{1}{2}$	0
small	$\frac{1}{6}$	$\frac{1}{6}$	$\frac{2}{3}$

(a)

	cost		
mass	expensive	medium	cheap
heavy	1	0	0
medium	$\frac{3}{7}$	$\frac{4}{7}$	0
ight	$\frac{2}{7}$	$\frac{1}{7}$	$\frac{4}{7}$

(b)

		cost		
displace	**mass**	expensive	medium	cheap
medium	medium	$\frac{1}{4}$	$\frac{3}{4}$	×
medium	light	$\frac{1}{2}$	$\frac{1}{2}$	×
small	medium	0	1	×
small	light	0	0	1

(c)

Table 4: Part of SADs

Continuing the LCR algorithm, the rule "IF **fuelcap**=medium
AND **speed**=fast THEN **cost**=medium" is derived; the set of tuples corresponding to this rule is {3, 17}. The set of tuples corresponding to rule 3 is a proper subset of the set of tuples corresponding to this rule. Hence, rule 3 is discarded according to Strategy 3.

After completing algorithm LCR on GCAR , the derived rules are shown in Figure 4.

Rule 1: IF **displace**=large THEN **cost**=expensive
Rule 2: IF **mass**=heavy THEN **cost**=expensive
Rule 3: IF **fuelcap**=high AND
speed=fast THEN **cost**=expensive
Rule 4: IF **fuelcap**=medium AND
speed=fast THEN **cost**=medium
Rule 5: IF **displace**=small AND
mass=light THEN **cost**=cheap
Rule 6: IF **displace**=medium AND **mass**=medium
AND **speed**=medium THEN **cost**=medium

Figure 1: A set of rules derived according to the LCR algorithm

4 Evaluation

In this section, we analyze the computational complexity of the LCR algorithm. An evaluation model is proposed to show that the set of rules derived according to the LCR algorithm is more compact than that of other approaches [1, 4, 9, 11]. Other extensions of the LCR algorithm are also discussed.

4.1 Computational complexity of the LCR algorithm

Suppose N is the total number of tuples in GR and there are k condition attributes in GR. It takes time $O(N)$ to record the tuple numbers for the same attribute value in an attribute, and for each combination of attribute values in the same tuple from two different attributes. After recording this information, the learning algorithm need not scan GR again. However, learning techniques proposed in Ziarko [9] and Agrawal, et al. [1] may need to scan a relation repeatedly.

We analyze the computational complexity of a SAD. Assume that A_1, A_2, ... and A_i are condition events and that A_{i+1} is a target event. Suppose the average cardinality of $TS(A_j, A_{j+1})$ is m, for all j within 1 and i. In order to obtain a SAD from the combination A_1, A_2, ... and A_i to event A_{i+1}, the intersections of tuple sets $TS(A_1, A_2)$, ... and $TS(A_i, A_{i+1})$ need to be computed, as discussed in Section 3.3. The intersection of two tuple sets requires at most $2m$ comparisons to find the same elements in the two tuple sets and in total, $i-1$ intersections are needed to find the same elements in the i tuple sets. Hence, in the worst case, $2m(i-1)$ comparisons are required to obtain a SAD in iteration i. As i is less than or equal to k and numbers m and k are both small numbers, few comparisons are needed to obtain a SAD.

Suppose, the average number of distinct values in each attribute is h. For iteration i, there are $C_i^k = \frac{k!}{(k-i)!i!}$ combinations of i attributes selected from k condition attributes, and there are h^i combinations of values from i condition attributes. Hence, there are at most $h^{i+1} \times C_i^k$ SADs to compute in iteration i. As there are heuristics proposed in the LCR algorithm, the number of computations of SADs can be decreased. Moreover, the LCR algorithm needs to scan a relation only once. After this relation scan, the efficiency of the LCR algorithm is independent of the size of the relation. However, most learning algorithms suffer from inefficiency problems in a large database environment [1, 7, 14].

4.2 An evaluation model

An *evaluation model* is constructed in the following to evaluate how compact a rule set about a target attribute is. There are three observations: the smaller the cardinality of the rule set is, the more compact is the rule set; the more tuples are corresponding to each rule in the rule set, the more compact is the rule set; and the smaller the number of condition attributes in the antecedent of a rule in the rule set is, the more compact is the rule set. Suppose the cardinality of the rule set about a certain target attribute is r, the number of the tuples corresponding to rule i in the rule set is t_i, and the number of condition attributes in the antecedent of rule i is C_i, $1 \leq i \leq r$. The evaluation model is:

$$E = \frac{1}{r}\sum_{i=1}^{r} \frac{t_i}{n} \times \frac{1}{C_i} \qquad (10)$$

According to this evaluation model, we see that the larger the value E is, the more compact is the rule set. Notice that for a relation, the largest E value for all possible rule sets may not be 1. The value E for the rule set about target attribute **cost** in GCAR according to the LCR algorithm is 0.12.

Here, we illustrate two other approaches to show that the rule set derived according to the LCR algorithm is more compact than those derived by these two approaches.

Quinlan [9] proposed a learning method ID3 for classification. *Decision trees* are built to assign instances to a single class based on values of a certain attribute. Applying this method to derive rules about attribute **cost** from relation GCAR, the result is shown in Figure 5.

According to the evaluation model in expression (10), the value E of the rule set in Figure 5 is 0.062, which is smaller than that of the rule set derived according to LCR algorithm.

We apply the learning algorithm presented by Han, et al. [4] to derive rules about attribute **cost** from relation GCAR. The relevant attributes are **displace**, **mass**, **fuelcap** and **speed**. Suppose the relation GCAR is a *final generalized relation* for relation CAR after applying their learning algorithm. Each tuple in the relation GCAR becomes a rule, and there are 18 rules in the rule set. According to the evaluation model, the value E of this rule set is 0.014.

Rule 1: IF **displace**=large THEN **cost**=expensive
Rule 2: IF **displace**=medium AND **mass**=heavy THEN **cost**=expensive
Rule 3: IF **displace**=small AND **mass**=heavy THEN **cost**=expensive
Rule 4: IF **displace**=small AND **mass**=medium THEN **cost**=medium
Rule 5: IF **displace**=small AND **mass**=light THEN **cost**=cheap
Rule 6: IF **displace**=medium AND **mass**=medium AND **speed**=fast THEN **cost**=expensive
Rule 7: IF **displace**=medium AND **mass**=medium AND **speed**=medium THEN **cost**=medium
Rule 8: IF **displace**=medium AND **mass**=light AND **fuelcap**=high THEN **cost**=expensive
Rule 9: IF **displace**=medium AND **mass**=light AND **fuelcap**=medium THEN **cost**=medium

Figure 2: A set of rules derived according to ID3

4.3 Variations of the learning algorithm

Each tuple in a generalized relation may contain many tuples in an initial relation. The more tuples in an initial relation a tuple in GR contains, the more representative is the tuple. If the number of tuples in the initial relation contained in a tuple in GR is greater than some threshold, the tuple in GR is said to be a *representative tuple*. This threshold can be specified according to the size of the initial relation.

The LCR algorithm cannot terminate until all tuples in GR are corresponding to some derived rules. However, if a tuple is not a representative tuple, this tuple can be regarded as *exceptional data* and may not have to be corresponding to a derived rule. Hence, the

LCR algorithm can be revised to terminate when all representative tuples in GR are corresponding to some derived rules.

In real-world databases, there may exist conflicting data. For example, assume after completing the generalization algorithm, relation GEMP (Table 5) contains two tuples:

Sex	Age	Salary	Position
male	old	medium	professor
male	old	medium	instructor

Table 5: relation GEMP

Suppose the learning task is to derive rules about attribute Position; attributes Age and Salary are condition attributes for target attribute Position. As the two tuples in GEMP conflict, no rules which have full confidence can be derived. The LCR algorithm can be revised to extract semantic association of which the degree reaches some large threshold as a rule. These rules with large confidences are useful for decision support and to give approximate answers to statistical queries.

5 Conclusion and Future Work

In order to make data more regular and easier to characterize in terms of rules, we present a generalization algorithm that can generalize the data and diminish their sizes. Domain concept hierarchies are introduced to support the generalization algorithm.

Semantic association relationships are important knowledges in databases. A learning algorithm is presented to identify important relationships and to derive compact rules. A mechanism that makes use of conditional probability and set theory to obtain the SADs is proposed such that the learning algorithm needs to scan a relation only once. Heuristics that are used to decrease the number of computations of SADs are also provided to improve the efficiency of the learning algorithm.

Finally, we evaluate the efficiency of our learning technique and compare it with other approaches. An evaluation model is developed to evaluate how compact a rule set about a target attribute is. The rule set derived according to the LCR algorithm are demonstrated to be more compact than the rule set derived according to other approaches.

We shall extend the learning algorithm to extract relationships in databases with uncertain or incomplete data. Also we shall consider resource discovery which extracts knowledge from multidatabase system. Since there exist incompatibilities in distinct databases, we need to consider resolving conflicts in the rule sets derived from these databases.

References

[1] R. Agrawal and et al. Mining Association Rules Between Sets of Items in Large Databases. In *SIGMOD*, pages 207–216, 1993.

[2] R. Agrawal and et al. An Interval Classifier for Database Mining Applications. In *VLDB-92*, pages 560–573, Vancouver, British Columbia, 1992.

[3] W. Chu and et al. Using Type Inference and Induced Rules to Provide Intensional Answers. In *Data Eng.*, pages 396–403, 1991.

[4] J. Han and et al. Data-Driven Discovery of Quantitative Rules in Relational Databases. In *IEEE Trans. Knowl. Data Eng.*, pages 29–40, 1993.

[5] T. Imielinski. Intelligent Query Answering in Rule-Based Systems. *Journal of Logic Programming*, 4:229–257, 1987.

[6] M. Jarke. Semantic Query Optimization in Expert Systems and Database Systems. In *Expert Database Systems*, pages 467–482, 1984.

[7] R.S. Michalski. A theory and Methodology of Inductive Learning. *Machine Learning: An Artifical Intelligence Approach*, 1:83–134, 1983.

[8] Z. Pawlak. Rough Sets. *Computer and Information Sciences*, 11(5):341–356, 1982.

[9] J. R. Quinlan. Induction of decision trees. *Machine Learning*, 1:81–106, 1986.

[10] M.S.E. Sciore and et al. A Method for Automatic Rule Derivation to Support Semantic Query Optimization. In *ACM Transactions on Database Systems*, pages 563–600, 1992.

[11] P.H. Winston. Learning by Building Identification Trees. In *Artificial Intelligence*, pages 423–432, 1992.

[12] Show-Jane Yen and Arbee L.P. Chen. Neighborhood/Conceptual Query Answering with Imprecise/Incomplete Data. In *Entity-Relationship Approach*, pages 151–162, 1993.

[13] C. Yu and W. Sun. Automatic Knowledge Acquisition and Maintenance for Semantic Query Optimization. In *IEEE Trans. Knowl. Data Eng.*, pages 362–375, 1989.

[14] W. Ziarko. The Discovery, Analysis, and Representation of Data Dependencies in Databases. In *Knowledge Discovery in Databases*, pages 195–209, 1991.