Computational Intelligence, Volume 11, Number 2, 1995

LEARNING IN RELATIONAL DATABASES: A ROUGH SET APPROACH

XIAOHUA HU, NICK CERCONE

Department of Computer Science, University of Regina, Regina, Saskatchewan, Canada, S4S 0A2

Knowledge discovery in databases, or data mining, is an important direction in the development of data and knowledge-based systems. Because of the huge amount of data stored in large numbers of existing databases, and because the amount of data generated in electronic forms is growing rapidly, it is necessary to develop efficient methods to extract knowledge from databases. An attribute-oriented rough set approach has been developed for knowledge discovery in databases. The method integrates machine-learning paradigm, especially learning-from-examples techniques, with rough set techniques. An attribute-oriented concept tree ascension technique is first applied in generalization, which substantially reduces the computational complexity of database learning processes. Then the cause-effect relationship among the attributes in the database is analyzed using rough set techniques, and the unimportant or irrelevant attributes are eliminated. Thus concise and strong rules with little or no redundant information can be learned efficiently. Our study shows that attribute-oriented induction combined with rough set theory provide an efficient and effective mechanism for knowledge discovery in database systems.

Key words: knowledge discovery in databases, machine learning, rough set, attribute-oriented induction.

1. INTRODUCTION

Knowledge discovery is the extraction of implicit, useful information from data. Knowledge discovery in databases is a form of machine learning which discovers interesting knowledge from databases and represents that knowledge in a high-level language. The growth in the size and number of existing databases far exceeds a human's ability to analyze this data, which creates both a need and an opportunity to extract knowledge from databases.

A major challenge to learning in databases is computational efficiency. In our previous studies (Cai *et al.* 1991; Han *et al.* 1992), we developed an attribute-oriented induction approach for discovering knowledge in relational databases. The approach strives for efficiency in two aspects: *knowledge-directed learning*, and *attribute-oriented induction*. The former is achieved by providing knowledge about the learning task, the concept hierarchies, and the expected rule forms. The latter is achieved by attribute-oriented concept tree ascension. These techniques substantially reduce the search space and improve the efficiency in a database learning process.

The attribute-oriented induction method integrates a machine learning paradigm, especially *learning from example* techniques, with database operations and extracts generalized data from actual data in databases. Induction is performed attribute by attribute, using attribute removal and concept ascension. As a result, undesirable attributes may be removed and different tuples may be generalized to identical ones, and the final generalized relation may consist of only a small number of distinct tuples. Then the final generalized relation can be transformed into logical rules. In the final generalized relation, all the attributes are treated as equally important, although this may not be true in real life. For example, to determine the (gas) mileage of a car, the weight and the (horse) power of the car are more important than its color. Therefore, it is important to retain the set of relevant attributes and eliminate the set of irrelevant or unimportant attributes according to the learning task without losing essential information about the original data in the databases, that is, to find a minimal subset of interesting attributes that have the same power as the original set of attributes in the database for distinguishing different classes. This reduction of the set of attributes simplifies the final generalized relation and produces a set of concise and meaningful rules.

Our previous attribute-oriented induction method did not analyze data dependency relationships among attributes, so the rules generated may not be concise and strong since they may contain some redundant information or unnecessary constraints. Thus, a technique is needed to perform comprehensive analysis of properties of the generalized data prior to the generation of rules. On the other hand, the rough set technique introduced by Pawlak (1982) provides the necessary tools to analyze the set of attributes globally but may not be feasible to apply to a large database directly because of its computational complexity, which is NP-hard (Ziarko 1990). The two approaches are apparently different. But in both methods, objects are assumed to be characterized by attributes and attribute values. Our study finds that a close connection exists between attribute-oriented induction and rough set approach. A natural approach would combine the advantages of both techniques. In this paper, a new method for knowledge discovery is proposed which generalizes the data by performing attribute-oriented induction. It then applies the rough set techniques to the generalized relation to find data dependency among the generalized attributes and choose the best minimal attribute set to represent the final generalized relation. Finally, the final generalized relation is transformed into the logical rule form.

This paper is organized as follows. In Section 2, the rough set theory and information system are introduced. The principle and algorithm for learning from database are presented in Section 3. In Section 4, a comparison with other knowledge discovery methods is set forth, and the concluding remarks are in Section 5.

2. ROUGH SET AND INFORMATION SYSTEM

Rough set was first introduced in Pawlak (1982). In this section we introduce some principal notions of rough set cited from Pawlak (1982) and Ziarko (1990), and present a new method to find the best reduct.

2.1. Information System

By an information system S, we mean that $S = \{U, A, V\}$, where U is a finite set of objects, $U = \{x_1, x_2, \ldots, x_n\}$, A is a finite set of attributes, the attributes in A are further classified into two disjoint subsets—*condition* attributes C and *decision* attributes D, $A = C \cup D$,

$$V = \bigcup_{p \in A} V_p$$

and V_p is a *domain* of attribute p.

Let $P \subset A, x_i, x_j \in$ U. We define a binary relation $\tilde{P}$, called an *indiscernibility relation*, as follows:

$$\tilde{P} = \{(x_i, x_j) \in U \times U : \text{ for every } p \in P \; p(x_i) = p(x_j)\}$$

We say that x_i and x_j are indiscernible by set of attributes P in S iff $p(x_i) = p(x_j)$ for every $p \in P$. One can check that $\tilde{P}$ is an equivalence relation on U for every $P \subset A$. Equivalence classes of relation are called P-elementary sets in S. A-elementary sets are called atoms of S. Information system S is selective iff all atoms in S are one element set, i.e., A is an identity relation.

A relational database may be considered an information system in which columns are labeled by attributes, rows are labeled by objects, and the entry in column p and row x has the value $p(x)$. Each row in the relational table represents *information* about some object in U. The difference is that the entities of the information systems do not need to be distinguished by their *attributes* or by their relationship to entities of another type. In the relational database, one attribute is identified as a *decision* attribute, and the other attributes are the *condition* attributes. In this paper, we adopt the view that a relational database is a selective information system and use the term *relational database* and *information system* interchangeably.

2.2. Approximation Space

For the information system $S = \{U, A, V\}$, and every subset $IND \subset A$ generates an equivalence relation (*indiscernibility relation*; we also use IND to denote the corresponding indiscernibility relation) on U. An order pair $AS = (U, IND)$ is called an *approximation space*. For any element x_i of U, the equivalence class of x_i in relation to IND is represented as $[x_i]_{IND}$. Equivalence classes of IND are called *elementary sets in AS* because they represent the smallest discernible groups of objects.

Any finite union of elementary sets in AS is called a *definable set in AS*.

Let $X \subset U$. We want to define X in terms of *definable sets* in AS; thus we need to introduce the following notions cited from Pawlak (1982):

(i) The lower approximation of X in AS is defined as

$$\underline{IND}X = \{x_i \in U \mid [x_i]_{IND} \subset X\}$$

$\underline{IND}$X is the union of all those elementary sets each of which is contained by X. For any $x_i \in \underline{IND}$X, it is certain that x_i belongs to X.

(ii) The upper approximation of X in AS is defined as

$$\overline{IND}X = \{x_i \in U \mid [x_i]_{IND} \cap X \neq \emptyset\}$$

$\overline{IND}$X is the union of those elementary sets each of which has a non-empty intersection with X. For any $x_i \in \overline{IND}$X, we can only say that x_i could possibly belong to X.

(iii) The set $\overline{IND}X - \underline{IND}X$ is called the *IND*-doubtful region of *IND* in (U, IND). For any $x_i \in U$, if x_i is in $\overline{IND}X - \underline{IND}X$, it is impossible to determine that x_i belongs to X based on the descriptions of the elementary sets of *IND*.

Example 2.1. Let us consider a generalized car relation in Table 1. $U = \{1, 2, 3, \dots, 14\}$ is the collection of cars. Suppose we choose $IND = \{cyl, power, weight\}$, and $D = mileage$ is the decision attribute. Thus the decision attribute consists of two concepts: D_{MEDIUM} = "*mileage = MEDIUM*" and D_{HIGH} = "*mileage = HIGH*."

$$D_{MEDIUM} = \{1, 2, 3, 4, 5, 6, 7\}$$

$$D_{HIGH} = \{8, 9, 10, 11, 12, 13, 14\}$$

We have the equivalence classes of *IND* as shown:

$$E_1 = \{1, 6\},\ E_2 = \{2\},\ E_3 = \{3, 4, 10, 13, 14\},$$
$$E_4 = \{5, 7, 11\},\ E_5 = \{8\},\ E_6 = \{9\},\ E_7 = \{12\}$$

The corresponding lower approximation and upper approximation of D are as follows

$$\underline{IND}(D_{MEDIUM}) = \{E_1, E_2\} = \{1, 6, 2\}$$
$$\overline{IND}(D_{MEDIUM}) = \{E_1, E_2, E_3, E_4\} = \{1, 6, 2, 3, 4, 10, 13, 14, 5, 7, 11\}$$
$$\underline{IND}(D_{HIGH}) = \{E_5, E_6, E_7\} = \{8, 9, 12\}$$
$$\overline{IND}(D_{HIGH}) = \{E_3, E_4, E_5, E_6, E_7\} = \{3, 4, 10, 13, 14, 5, 7, 11, 8, 9, 12\}$$

TABLE 1. A Generalized Car Relation.

Object#	Make_model	cyl	displace	compress	power	trans	weight	mileage
1	USA	6	MEDIUM	HIGH	HIGH	AUTO	MEDIUM	MEDIUM
2	USA	6	MEDIUM	MEDIUM	MEDIUM	MANUAL	MEDIUM	MEDIUM
3	USA	4	SMALL	HIGH	MEDIUM	AUTO	MEDIUM	MEDIUM
4	USA	4	MEDIUM	MEDIUM	MEDIUM	MANUAL	MEDIUM	MEDIUM
5	USA	4	MEDIUM	MEDIUM	HIGH	MANUAL	MEDIUM	MEDIUM
6	USA	6	MEDIUM	MEDIUM	HIGH	AUTO	MEDIUM	MEDIUM
7	USA	4	MEDIUM	MEDIUM	HIGH	AUTO	MEDIUM	MEDIUM
8	USA	4	MEDIUM	HIGH	HIGH	MANUAL	LIGHT	HIGH
9	JAPAN	4	SMALL	HIGH	LOW	MANUAL	LIGHT	HIGH
10	JAPAN	4	MEDIUM	MEDIUM	MEDIUM	MANUAL	MEDIUM	HIGH
11	JAPAN	4	SMALL	HIGH	HIGH	MANUAL	MEDIUM	HIGH
12	JAPAN	4	SMALL	MEDIUM	LOW	MANUAL	MEDIUM	HIGH
13	JAPAN	4	SMALL	HIGH	MEDIUM	MANUAL	MEDIUM	HIGH
14	USA	4	SMALL	HIGH	MEDIUM	MANUAL	MEDIUM	HIGH

2.3. Core and Reducts of Attributes

In many applications, the set of objects is classified into a disjointed family of classes based on the values of the decision attribute. We want to determine each class in terms of features of corresponding condition attributes belonging to this class. In most cases, classes are determined by several or even one attribute, not by small differences in all the attributes in the databases. This is consistent with the cognitive process of human discovery, because people often have difficulty taking more than a few attributes into account and tend to focus on a few important attributes. The rough set theory provides the tool to deal with this problem. Core and reduct are the two fundamental concepts of rough set. A reduct is the essential part of an information system which can discern all objects discernible by the original information system. A core is the common parts of all the reducts. In this section we present a method to distinguish the core from the discernibility matrix and to construct the best reduct from core based on the dependency relationship and significant value of attributes.

Let $S = \{U, A, V\}$ be an information system, with $A = C \cup D$ and B $\subset$ C. We define a positive region B in $IND(D)$, $POS_B(D)$, as

$$POS_B(D) = \cup\{\underline{B}X : X \in IND(D)\}.$$

The positive region $POS_B(D)$ includes all objects in U which can be sorted into classes of $IND(D)$ without error, based solely on the classification information in $IND(B)$.

We say that the set of attributes D depends in degree k $(0 \leq k \leq 1)$ on the subset R of C in S if

$$k(R, D) = card(POS_R(D))/card(POS_C(D))$$

The value $k(R, D)$ provides a measure of dependency between R and D. If R is a reduct of C with respect to D, then $k(R, D) = 1$.

Definition 1. An attribute $p \in B$ is superfluous in B with respect to D if $POS_B(D) = POS_{B-\{p\}}(D)$; otherwise p is indispensable in B with respect to D.

If an attribute is superfluous in the information system, it can be removed from the system without changing the dependency relationship of the original system. While an indispensable attribute carries the essential information about objects of the information system, it should be kept if one does not want to change the dependency relationship of the original system.

Definition 2. If every attribute of B is indispensable with respect to D, then B is indispensable with respect to D.

Definition 3. The set of all indispensable attributes in C with respect to D is called the core of C and is denoted by $CORE(C, D)$.

$$CORE(C, D) = \{a \in C : POS_C(D) \neq POS_{C-\{a\}}(D)\}.$$

Definition 4. $B \subset C$ is defined as a reduct in S if B is independent with respect to D and $POS_C(D) = POS_B(D)$.

The reduct of C is a minimal subset of attributes that discerns all objects discernible by the whole set of attributes. Usually C may have more than one reduct.

Significant Value of Attributes. Different attributes may play different roles in determining the dependency relationship between the condition and decision attributes. The significance of an individual attribute a added to the set R with respect to the dependency between R and D is represented by significant factor SGF, given by

$$SGF(a, R, D) = k(R + \{a\}, D) - k(R, D)$$

$SGF(a, R, D)$ reflects the degree of increase of dependency level between R and D as a result of the addition of the attribute a to R. In practice, the stronger the influence of the attribute a is on the relationship between R and D, the higher the value of the $SGF(a, R, D)$ is. For example, in Table 1, if $R = \{Make\_model, trans\}$, $D = \{mileage\}$, then $SGF(cyl, R, D) = 0.07$, $SGF(displace, R, D) = 0.07$, $SGF(compress, R, D) = 0.36$, $SGF(power, R, D) = 0.00$, $SGF(weight, R, D) = 0.07$.

Criteria for Best Reduct. Quite often an information system has more than one reduct. Each reduct can be used instead of the whole group of attributes in the original system in the decision-making procedure without changing the dependency relation in the original system. So a natural question is which reduct is the best. The selection depends on the optimality criterion associated with attributes. If it is possible to assign a cost function to attributes, then the selection can be naturally based on the combined minimum cost criteria. In the absence of an attribute cost function, the only source of information to select the reduct is the contents of the table (Ziarko 1990). In this paper we adopt the criteria that the best reduct is the one with the minimal number of attributes and that if there are two or more reducts with the same minimal number of attributes, then the reduct with the least number of combinations of values of its attributes is selected.

Modified Discernibility Matrix. In this subsection, we give a modified definition of discernibility matrix based on Pawlak (1990). Using *the modified discernibility matrix*, we can compute the core of the information system easily.

Definition 5. A *modified discernibility matrix* of C in S, $M(C) = \{m_{i,j}\}_{n\times n}$ is defined as

$$(m_{ij}) = \begin{cases} \emptyset & x_i, x_j \in \text{the same equivalence class of } IND(D) \\ \{c \in C : f(c, x_i) \neq f(c, x_j)\} & x_i, x_j \in \text{the different equivalence class of } IND(D) \end{cases}$$

TABLE 2. Modified Discernibility Matrix for the Generalized Car Relation.

	1	2	3	4	5	6	7	8	9	10	11	12	13	14
1														
2														
3														
4														
5														
6														
7														
8	bfg	bdeg	cefg	deg	dg	bdfg	dfg							
9	abcefg	abcdfg	acfg	acdeg	acdeg	abcdefg	acdefg							
10	abdef	ab	acdf	a	ae	abef	aef							
11	abcf	abcde	aef	acde	acd	acdf	acdf							
12	abcdef	abce	adef	ace	ace	abcef	acef							
13	bcef	abcd	af	acdf	acdf	abcdef	acdef							
14	abcef	bcd	f	cd	bcde	bcdef	cdef							

Abbreviations: a, Make_model; b, cyl; c, displace; d, compress; e, power; f, trans; g, weight.

The entry m_{ij} contains the attributes whose values are not identical on both x_i and x_j (x_i, x_j belong to different classes of $IND(D)$, that is, x_i, x_j represent different concepts). In other words, m_{ij} represents the complete information to distinguish x_i, x_j. $M(S) = (m_{ij})$ is symmetric; we need only compute the entries m_{ij} for $1 \leq j < i \leq n$.

Example 2.2. For the generalized car relation in Table 1, the modified discernibility matrix is computed in Table 2. (Suppose the attribute "mileage" is a decision attribute; the other attributes are condition attributes.)

Core and Modified Discernibility Matrix. A core has the common attributes of all the reducts. So a core can be used as a basis for computing a reduct. From the modified discernibility matrix, we can easily compute the core of the information system based on the following observation. (Note: a core of an information system may be empty.)

For $S = \{U, A, V\}$, $A = C \cup D$, $M(S) = \{m_{ij}\}$, for any $c \in C$, $c \in CORE(C, D)$ iff there exists i, j, $1 \leq j < i \leq n$ such that $m_{ij} = \{c\}$.

For example, examine the modified discernibility matrix in Table 2 for the generalized car relation in Table 1, $m_{10,4} = \{a\}$ and $m_{14,3} = \{f\}$, so the core of the attributes is $\{Make\_model, trans\}$.

Compute the Best Reduct. The general problem of finding all reducts is NP-hard (Ziarko 1990), but in most cases it is usually not necessary to find all the reducts. The user is often more interested in finding the best reduct with respect to his problem; moreover some user usually knows better about the decision task. Based on the dependency relation and significant value of attributes, it is very easy and efficient to find a "best" or "optimal" reduct that has the same discernibility as the original relation. Here we present an algorithm to construct the best reduct by using core as the starting point. The algorithm is very simple and straightforward.

Algorithm 1. Compute the best reduct.
Input: (i) The task-relevant generalized relation R'; (ii) a set of attributes AR for relation R', which is classified into condition attributes C, and decision attributes D; (iii) the core CO of AR computed from the discernibility matrix of R' (CO may be empty).

Output. The best reduct *REDU*.
Method
Step 1: $REDU = CO$.
Step 2: $AR = AR - REDU$.
Step 3: Find attribute a in AR which has the maximal value $SGF(a, REDU, D)$.
Step 4: If there are several attributes a_i $(i = 1, \dots, m)$ with the same maximal value, choose the attribute a_j which has the least number of combination value with *REDU*.
Step 5: $REDU = REDU \cup \{a_j\}; AR = AR - \{a_i\}$ $(i = 1, \dots, m)$.
Step 6: If $K(REDU, D) = 1$, then terminate, otherwise go to Step 3.

The best reduct of the generalized car relation in Table 1 is {*Make_model, compress, trans*} is generated by using this algorithm. We can find the best reduct or user minimal attribute subset in $N_A \times O(N' \times N')$ in the worst case, where N_A is the number of attributes in the generalized relation R', and N' is the number of tuples in R'. Usually N' is not big in the generalized relation R'. In the next section we discuss how to generalize data in the database.

3. PRINCIPLES AND ALGORITHMS FOR LEARNING IN RELATIONAL DATABASES

Rough set theory has been successful in many areas; witness the knowledge-based system in medical diagnosis, machine learning, and industry (Grzymala-Busse 1988). In this section, a new algorithm DBDeci is presented which can be used in database to derive knowledge rules. The method is based on the attribute-oriented concept ascension techniques proposed in Han *et al.* (1992) and on rough set techniques (Pawlak 1982). The method integrates generalization and reduction and provides a simple, efficient, and effective way to extract knowledge from database. To simplify our discussion, the following assumptions are made in our study.

Assumption 1. A database stores a large amount of information-rich and relatively reliable data.

Data in a database may be incomplete, redundant, or incorrect, in certain applications (Zytkow and Baker 1990), which makes knowledge discovery a challenging task. The assumption of data reliability facilitates the development of a knowledge discovery mechanism in relatively simple environments and then the evolution of the techniques toward more complicated situations.

Assumption 2. A knowledge discovery process is initiated by a user's learning request.

Ideally, a knowledge discovery system performs interesting discovery autonomously, without human interaction. However, since learning can be performed in many different ways on any subset of data in the database, a huge amount of knowledge may be generated from even a medium-sized database by unguided, autonomous discovery, whereas much of the discovery knowledge could be outside the user's interests. In contrast, a command-driven discovery may confine the study to the relevant set of data and the desired form of the rules, so that it represents a relatively constrained search for the desired knowledge. Thus command-driven discovery is adopted in this study.

Assumption 3. Generalized rules are expressed in terms of high-level concepts.

Without concept generalization, discovered knowledge is expressed in terms of primitive data (data stored in the database), often in the form of functional or multivalued dependency

{Honda_civic, Honda_acura,..., Honda_accord} $\subset$ Honda
{Toyota_tercel,...,Toyota_camry} $\subset$ Toyota
{Mazda_323, Mazda_626,..., Mazda_939} $\subset$ Mazda
{Toyota , Honda , ..., Mazda } $\subset$ Japan(car)
{Ford_escort, Ford_probe,..., Ford_taurus } $\subset$ Ford
{Chervolet_corvette, Chervolet_camaro,...,Chervolet_corsica } $\subset$ Chervolet
{Dodge_stealth, Dodge_daytona,..., Dodge_dynasty } $\subset$ Dodge
{Ford, Dodge, ..., Chervolet } $\subset$ USA(Car)
{Japan(Car), ..., USA(Car)} $\subset$ Any(Make_model)
{0..800} $\subset$ Light
{801..1200} $\subset$ Medium
{1201..1600} $\subset$ Heavy
{Low, Medium, High} $\subset$ Any(Weight)

FIGURE 1. Concept Hierarchy Table.

rules or primitive-level integrity constraints. On the other hand, with concept generalization, discovered knowledge can be expressed in terms of concise, expressive, and higher level abstractions in the form of generalized rules or generalized constraints and be associated with statistical information. Obviously, it is often more desirable for large databases to have rules expressed at concept levels higher than the primitive ones.

Assumption 4. Background knowledge is generally available for the knowledge discovery process.

Discovery may be performed with the assistance of relatively strong background knowledge (such as conceptual information, etc.) or with little support from background knowledge. The discovery of conceptual hierarchy information itself can be treated as a part of the knowledge discovery process. However, the availability of a relatively strong background knowledge not only improves the efficiency of the discovery process but also expresses the user's preference for guided generalization, which may lead to an efficient and desirable generalization process.

Based on these assumptions, a learning task can be specified by three primitives: *task-relevant data, background knowledge*, and *expected representation of learning results*. Our method is performed in three steps: First, an attribute-oriented, concept tree–ascending technique is applied to the task-relevant data relation, which removes some undesirable attributes (Cai *et al.* 1991) and generalizes the concepts of desirable attribute to a certain level based on the conceptual bias and the tuple threshold values. Then the rough set technique is applied to analyze the data dependency of the attribute and find the best reduct. A reduced relation is obtained from the generalized relation by removing those attributes not in the reduct. Finally rule extraction is performed on the reduced relation, which generates concise, expressive, and stronger rules.

Example 3.1. Suppose we have a collection of data on Japanese and American cars which includes such attributes **Plate#**, **Make_model**, type of fuel system (**fuel**), engine displacement (**disp**), **weight**, number of cylinders (**cyl**), **power**, presence of turbocharge (**turbo**), compression ratio (**comp**), transmitter (**tran**) and **mileage** as depicted in Table 3. It also includes the concept hierarchy table for the car relation as in Fig. 1, the concept hierarchy tree for the attribute Make_model depicted in Fig. 2.

A large data relation usually contains a huge set of distinct attribute values. To obtain a simple and concise scheme and derive a decision rule for each class, we should first generalize

TABLE 3. Car Relation.

Plate#	Make_model	fuel	disp	weight	cyl	Power	turbo	comp	tran	mileage
BCT89U	Ford Escort	EFI	Medium	876	6	High	yes	high	auto	medium
UI89P0	Dodge Shadow	EFI	Medium	1100	6	High	no	medium	manu	medium
P0967H	Ford Festival	EFI	Medium	1589	6	High	no	high	manu	medium
LKIPO8	Chevrolet Corvette	EFI	Medium	987	6	High	no	medium	manu	medium
IUTY56	Dodge Stealth	EFI	Medium	1096	6	High	no	high	manu	medium
ERTW34	Ford Probe	EFI	Medium	867	6	High	no	medium	manu	medium
TYUR45	Ford Mustang	EFI	Medium	1197	6	High	no	high	manu	medium
0987UO	Dodge Daytona	EFI	Medium	798	6	High	yes	high	manu	high
9876T	Chrysler Le B	EFI	Medium	1056	4	Medium	no	medium	manu	medium
UYTHG7	Dodge Spirit	EFI	Medium	1557	6	High	no	medium	manu	low
OPLSAD	Honda Civic	2-BBL	Small	786	4	Low	no	high	manu	high
KMN897	Ford Escort	2-BBL	Small	1098	4	Low	no	high	manu	medium
LKOPL0	Ford Tiempo	2-BBL	Small	1187	4	Medium	no	high	auto	medium
WEQ546	Toyota Corolla	EFI	Small	1023	4	Low	no	high	manu	high
PLMNH7	Mazda 323	EFI	Medium	698	4	Medium	no	medium	manu	high
QAS453	Dodge Daytona	EFI	Medium	1123	4	Medium	no	medium	manu	medium
PLMJH9	Honda Prelude	EFI	Small	1094	4	High	yes	high	manu	high
DSA321	Toyota Paseo	2-BBL	Small	1023	4	Low	no	medium	manu	high
GHF6UY	Chevrolet Corsica	EFI	Medium	980	4	High	yes	medium	manu	medium
KNM876	Chevrolet Beretta	EFI	Medium	1600	6	High	no	medium	auto	low
IKLO90	Chevrolet Cavalier1	EFI	Medium	1002	6	High	no	medium	auto	medium
IKHTY8	Chrysler Le B	EFI	Medium	1098	4	High	no	medium	auto	medium
OPL876	Mazda 626	EFI	Small	1039	4	Medium	no	high	manu	high
UYT342	Chervolet Corsica	EFI	Small	980	4	Medium	no	high	manu	high
UYTBNG	Chevrolet Lumina	EFI	Small	1000	4	Medium	no	high	manu	medium

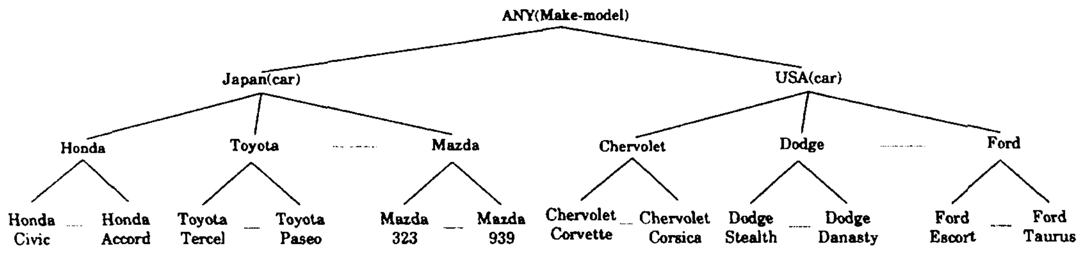

FIGURE 2. Concept Hierarchy Tree for Make_model. The process is demonstrated using Example 3.1.

the primitive data instances to higher level concepts. This is realized by an attribute-oriented generalization (Cai *et al.* 1991) on the task-relevant relation. The first attribute, Plate#, is the key to the relation. The key value is distinct for each tuple in the relation. If no higher level concept is provided for such an attribute in the concept tree, the values of the attribute cannot be generalized and they should be removed in the generalization. Also, other candidate key attributes or nonkey attributes can be eliminated under a similar condition. We then examine the remaining attributes. By examining the values attribute by attribute and substituting the values in each attribute with their higher level concepts in the corresponding concept tree, e.g., from *Mazda 323* to *Mazda 798* (weight of car) to *light*, the relation is generalized.

Strategy 1 (*Attribute-Oriented Generalization*). The generalization is performed on each attribute by removing the attribute if there is a large set of distinct values for the attribute and no higher-level concept hierarchy available, or by otherwise ascending the concept tree (i.e., substituting the values of the attribute in each tuple by its corresponding higher level concept).

Rationale: Attribute removal corresponds to the generalization rule of *dropping conditions* (Michalski 1983). Consider a tuple as a set of conjuncts in the logical forms; an attribute value together with its attribute name form one of the conjuncts. By removing a conjunct, a constraint is eliminated and the concept is generalized. If there are a large set of distinct values for an attribute, the large set of values must be generalized. However, if there is no higher level concept provided for the attribute, it can not be further generalized by ascending the concept tree. Therefore, the attribute should be eliminated in generalization. Attribute removal can also be viewed as a generalization of all attribute values to the most general concept ANY and then removed from the generalization.

Concept tree ascension corresponds to the generalization rule of *climbing generalization trees* (Michalski 1983). If there exists a higher level concept for the value in the concept tree, the substitution of the value in each tuple in the relation by the corresponding higher level concepts makes the tuple cover more cases than the original one; thus it generalizes the tuple. ■

As a result of generalization, different objects may be generalized to equivalent ones where two (generalized) objects are *equivalent* if they have the same corresponding attribute values without considering a special internal attribute *vote*, which registers the number of tuples in the initial working relation that are generalized to the tuple in the current resulting relation. The *vote* accumulated in the generalized relation incorporates quantitative information in the learning process.

Strategy 2 (*Vote Propagation*). The value of the vote of a tuple should be carried to its corresponding generalized tuple, and the vote should be accumulated when merging equivalent tuples in generalization.

Rationale: Based on the definition of *vote*, the vote of each generalized tuple must register the number of tuples in the initial data relation generalized to the current one. Therefore, to keep the correct number of votes registered, the vote of each tuple should be carried in the generalization process, and such votes should be accumulated when merging identical tuples. ■

Notice it is often necessary for some attribute values to be generalized by climbing up a concept tree several times in order to derive a simple class scheme. To control the generalization process and integrate it with the classification process, it is necessary to provide a class threshold value, which is the upper bound on the number of classes to be partitioned. Then the generalization can be repeatedly performed on an attribute (*without examining other attributes*) until the number of distinct values is under the specified threshold value. For instance, if the threshold value is set at three in this example (because there are only three distinct values for the decision attribute *mileage*), the concepts in the attribute Make_model should be generalized on two levels (*Mazda 323* to *Mazda* then to *Japanese car*).

Strategy 3 (Threshold Control on Generalized Relations). If the number of tuples of a generalized relation is larger than the generalization threshold value, further generalization on the relation should be performed.

Rationale: Based on the definition of the generalization threshold, further generalization should be performed if the number of tuples in a generalization relation is larger than the threshold value. By further generalization on selected attribute(s) and the merging of identical tuples, the size of the generalized relation is reduced. Generalization should continue until the number of remaining tuples is no greater than the threshold value. ■

Since different attribute values can be substituted by the higher level concepts in the generalization process, a set of tuples may be generalized to the same generalized tuples in the generalized relation and the size of the generalized relation is reduced.

A detailed discussion of the attribute-oriented generalization can be found in Han *et al.* (1992). For the car relation, if the class threshold relation is set to 3, the attribute-oriented generalized will derive the generalized relation, as shown in Table 4 (with redundant tuples eliminated). After the generalization process, rough set method is performed on the generalized relation. The best minimal set of attributes can be found, and the further reduction of the generalized relation is generated.

Taking mileage as the decision attribute, we examine how to compute the reduct of the condition attributes.

Strategy 4 (Find the best reduct and reduce the generalized relation). By using Algorithm 1 in section 2, we can find the best reduct {*Make_model, fuel, disp, weight*} for the condition attributes in Table 4. A reduced relation as shown in Table 5 is obtained by removing those attributes (*cyl, Power, turbo, comp, tran*) not in the reduct without changing the dependency relationship between the mileage and the condition attributes.

Strategy 5 (Combine the similar tuples). In the same class, two tuples can be combined into one if the values of condition attributes differ in only one attribute; this corresponds to the *closing interval rule* in Michalski (1983). If the data values appearing in the merged tuples cover all the possible values of the attribute in the corresponding generalization hierarchy, then this attribute should be dropped from the tuple. For example, if the first tuple and second tuple in Table 5 differ only in *fuel*, these two can be combined into *(USA,* {*EFL,2-BBL*}, *Medium, Medium, medium)*, which can be further simplified to *(USA, , Medium, Medium,*

TABLE 4. The Car Relation after Generalization.

Make_model	fuel	disp	weight	cyl	Power	turbo	comp	tran	mileage	vote
USA	EFI	Medium	Medium	6	High	yes	high	manu	medium	1
USA	EFI	Medium	Medium	6	High	no	medium	manu	medium	3
USA	EFI	Medium	Medium	6	High	no	medium	auto	medium	1
USA	EFI	Medium	Medium	6	Medium	no	Medium	manu	medium	1
USA	EFI	Medium	Medium	6	High	no	high	manu	medium	2
USA	EFI	Medium	Light	4	High	yes	high	manu	high	1
USA	EFI	Small	Medium	4	Medium	no	high	manu	high	1
USA	EFI	Medium	Medium	6	High	no	high	manu	medium	2
USA	EFI	Medium	Heavy	6	High	yes	high	auto	low	1
USA	EFI	Medium	Medium	6	Medium	no	medium	manu	medium	1
USA	EFI	Medium	Heavy	6	High	no	medium	manu	low	1
Japan	2-BBL	Small	Light	4	Low	no	high	manu	high	1
USA	2-BBL	Small	Medium	4	Medium	no	high	auto	medium	1
Japan	EFI	Small	Medium	4	Low	no	high	manu	high	1
Japan	EFI	Medium	Light	4	Medium	no	medium	manu	high	1
USA	2-BBL	Medium	Medium	4	Medium	no	medium	manu	medium	1
Japan	EFI	Small	Medium	4	High	yes	high	auto	high	1
Japan	2-BBL	Small	Medium	4	Low	no	medium	manu	high	1
USA	EFI	Medium	Medium	4	High	yes	midium	manu	medium	1
USA	EFI	Medium	Medium	6	High	no	medium	manu	medium	1
Japan	EFI	Small	Medium	4	Medium	no	high	auto	high	1

TABLE 5. Reduced Car Relation.

Make_model	fuel	disp	weight	mileage	vote
USA	EFI	Medium	Medium	medium	13
USA	2-BBL	Medium	Medium	medium	1
USA	2-BBL	Small	Medium	medium	1
USA	EFI	Medium	Heavy	low	2
USA	EFI	Medium	Light	high	1
USA	EFI	Small	medium	high	1
Japan	2-BBL	Small	Light	high	1
Japan	EFI	Small	Medium	high	3
Japan	EFI	Small	Light	high	1
Japan	2-BBL	Small	Medium	high	1

medium). After examining the distribution of the values for each attribute, the reduced relation is further simplified in Table 6.

Strategy 6 (Transform the tuples in the reduced relation into logical rules). The value of *vote* is the number of cases in the original relation that support each given rule, since the tuples in the original relation may be distributed along the full spectrum of the possible values. Without using quantitative information, it is impossible to obtain a meaningful set of rules for such data. We would not trust a rule based on just a few cases. However, using the

TABLE 6. Reduced Car Relation after Combination.

Make_model	fuel	disp	weight	mileage	vote
USA		Medium	Medium	medium	14
USA	2-BBL		Medium	medium	2
USA	EFI	Medium	Heavy	low	2
USA	EFI	Medium	Light	high	1
USA	EFI	Small	Medium	high	1
Japan		Small	Light,Medium	high	6

TABLE 7. Reduced Car Relation with Frequency Ratio.

Make_model	fuel	disp	weight	mileage	f_ratio
USA		Medium	Medium	medium	56%
USA	2-BBL		Medium	medium	8%
USA	EFI	Medium	Heavy	low	8%
USA	EFI	Medium	Light	high	4%
USA	EFI	Small	Medium	high	4%
Japan		Small	Light, Medium	high	24%

quantitative information, various kinds of techniques can be developed to extract meaningful rules. A simple method is introduced here.

The method treats the tuples which occur rarely in the reduction relation as exceptional or noise case and filters them out before the rule generalization. This is accomplished by calculating the *frequency ratio* for each tuple and filtering out the *exceptional* tuples using the *noise filter threshold.*

Definition 6. The *frequency ratio* is the ratio of the tuple vote value versus the summation of the vote in the reduced relation, that is (where n is the number of tuples in the reduction relation),

$$r_{frequency} = vote\_of\_the\_tuple / \sum_{i=1}^{n}(vote\_of\_tuple_i)$$

The *noise filter threshold* is a small percentage number used to filter out those tuples in the reduced relation with a very low frequency ratio (i.e., below the specified noise filter threshold).

Suppose the *noise filter threshold* is set to 5%. The noise-filtering operation is performed as follows. If the frequency ratio of a tuple is less than 5%, then the tuple is cleared, otherwise the tuple is transformed to a logical rule.

According to Table 7, we can derive the following rules.

(1) *if* (Make_model=USA(car)) & (disp=Medium) & (weight=Medium)
then (mileage=medium).

(2) *if* (Make_model=USA(car)) & (fuel=2-BBL) & (weight=Medium)
then (mileage=medium).

(3) *if* (Make_model=Japan(car)) & (disp=Small) & (weight=Light ∪ Medium)
then (mileage=high).

(4) *if* (Make_model=USA(car)) & (fuel=EFL) & (disp=Medium) & (Weight=Heavy)
then (mileage=low).

In summary, we present the algorithm below:

Algorithm 2. (DBDeci—An Attribute-Oriented Rough Set Knowledge Discovery Algorithm in Databases).
Input: (i) A set of task-relevant data R (assume they are obtained by a relation query and are stored in a relation relation), a relation of arity n with a set of attributes A_i ($1 \leq i \leq n$); (ii) a set of concept hierarchies, H_i, where H_i is a hierarchy on the generalized attribute A_i, if available; (iii) the threshold value T; and (iv) decision attribute name.
Output. A set of decision rules.
Method

```
Step 1. Attribute-oriented induction.
    /* Suppose $d_i$ is the number of distinct values in attributes $A_i$ */.
begin

  for each attribute $A_i$ ( $1 \leq i \leq n$) of $R$ do {
      while $T \leq d_i$ do
          if there is no higher level concepts of $A_i$
          then remove $A_i$;
          else substitute the values by its corresponding minimal generalized concept; }
      eliminate redundant tuples and register the number of identical tuples in vote.
    /* Generalization is implemented as follows. Collect the distinct values in the relation and
      compute the lowest level $L$ on which the number of distinct values will be no more than
      $T$ by synchronously ascending the concept hierarchy from these values. Then generalize
      the attribute to this level $L$ by substituting for each value $A_i$'s with its corresponding
      concept $H_i$ at level $L$. */ }

end { attribute-oriented induction }
Step 2. Find the desired reduct.
begin

  construct the modified discernibility matrix $M(R)$ and find the core
  using core as the starting point to compute the desired reduct (apply Algorithm 1)

end
Step 3. Generate the reduced relation from the generalized relation by removing those
attributes not in the desired reduct and combine the similar tuples.
Step 4. Transform the tuples in the reduced relation into logic rules.
```

4. DISCUSSION

The algorithms presented in this paper are based on the research in Cai *et al.* (1991) and influenced by the ideas presented in Ziarko (1990). Our algorithms adopt the attribute-oriented conceptual ascension technique combined with the rough set technique; attribute-oriented induction provides a simple and efficient way for the generalization process, and rough set provides an effective tool to find out the reduction form of the generalized relation. The major difference of our approach from others is attribute-oriented vs. tuple-oriented induction. The former techniques perform generalization, attribute by attribute, while the latter, tuple by tuple. The two approaches involve significantly different search spaces.

Our algorithms use rough set to analyze the attribute dependency and choose the reduct form of the best minimal subset for rule generalization. Our method eliminates undesirable attributes in the generalization process and removes irrelevant attributes in the reduction process. The rules generated are concise, expressive, and strong because it is in the most generalized form. Besides, with concept generalization, the derived rules are represented in higher level abstraction.

Another advantage of our approach to many other learning algorithms is our integration of the learning process with database operations. Relational systems store a large amount of information in a structured and organized manner and are implemented by well-developed storage and accessing techniques (Han *et al.* 1992). In contrast to most existing learning algorithms, which do not take full advantage of these database facilities (Diettrich and Michalski 1983), our approach primarily adopts such relational operations as selection, join, projection (extracting relevant data and removing attributes), tuple substitution (ascending concept trees), and intersection (discovering common tuples among classes).

5. CONCLUSION

Based on our previous implementation of the database learning system DBLEARN (Cai *et al.* 1991), which performs attribute-oriented generalization to extract knowledge rules in relational database, we are currently working on the implementation and testing of the DBDeci for the large database environment. We plan to test it on a large database—that of the Research Grants Information System of Natural Science and Engineering Research Council of Canada—hope to report our results in the future.

ACKNOWLEDGMENTS

The authors are members of the Institute for Robotics and Intelligent Systems (IRIS) and wish to acknowledge the support of the Networks of Centres of Excellence Program of the Government of Canada, the Natural Sciences and Engineering Research Council, and the participation of PRECARN Associates Inc.

REFERENCES

CAI, Y., N. CERCONE, and J. HAN. 1991. Attribute-oriented induction in relational databases. *In* Knowledge Discovery in database. *Edited by* G. Piatetsky-Shapiro and W. J. Frawley. AAAI/MIT Press, Cambridge, MA, pp. 213–228.

DIETTERICH, T. G., and R. S. MICHALSKI. 1983. A theory and methodology of inductive learning. *In* Machine learning: an artificial intelligence approach, vol. 1. *Edited by* R. S. Michalski *et al.* Morgan Kaufmann, San Mateo, CA, pp. 43–82.

FRAWLEY, W. J., G. PIATETSKY, and C. J. MATHEUS. 1991. Knowledge discovery in database: an overview. *In* Knowledge discovery in database. *Edited by* G. Piatetsky-Shapiro and W. J. Frawley. AAAI/MIT Press, Cambridge, MA, pp. 1–27.

GRZYMALA-BUSSE, J. W. 1988. Knowledge acquisition under uncertainty: a rough set approach. Journal of Intelligent and Robotic Systems, **1**:3–16.

HAN, J., Y. CAI, and N. CERCONE. 1992. Knowledge discovery in databases: an attribute-oriented approach. *In* Proceedings of the 18th VLDB Conference, Vancouver, B.C., Canada, pp. 340–355.

KAUFMAN, K. A., R. S. MICHALSKI, and L. KERSCHBERG. 1991. Mining for knowledge in databases: goals and general descriptions of the INLEN system. *In* Knowledge discovery in database. *Edited by* G. Piatetsky-Shapiro and W. J. Frawley. AAAI/MIT Press, Cambridge, MA, pp. 449–462.

MANAGO, M. V., and Y. KODRATOFF. 1987. Noise and knowledge acquisition. *In* Proceedings of the 10th International Joint Conference on Artificial Intelligence, Milan, Italy, pp. 348–354.

MICKALSKI, R. S. 1983. A theory and methodology of inductive learning. *In* Machine learning: an artificial intelligence approach, vol. 1. *Edited by* R. S. Michalski *et al.* Morgan Kaufmann, San Mateo, CA, pp. 41–82.

PAWLAK, Z. 1982. Rough sets. International Journal of Information and Computer Science, **11**(5):341–356.

PAWLAK, Z. 1990. Anathomy of conflicts. *In* ICS Research Report 11/92. Warsaw University of Technology, Warsaw, Poland.

SILBERSCHATZ, A., M. STONEBRAKER, and J. D. ULLMAN. 1991. Database systems: achievements and opportunities. Communications of the ACM, **34**(10):94–109.

ZIARKO, W. 1990. The discovery, analysis, and representation of data dependencies in databases. *In* Knowledge discovery in databases. *Edited by* G. Piatetsky-Shapiro and W. J. Frawley. AAAI/MIT Press, Cambridge, MA, pp. 213–228.

ZYTKOW, J., and J. BAKER. 1990. Interactive mining of irregularities in databases. *In* Knowledge discovery in databases. *Edited by* G. Piatetsky-Shapiro and W. J. Frawley. AAAI/MIT Press, Cambridge, MA, pp. 31–54.