An Overview of a Tool for Extracting Rules from Databases with Incomplete Information

BMFSA(2001-07-01-12)

CONTRIBUTED ARTICLE

Biomedical Soft Computing and Human Sciences, Vol.7, No.1, pp. 77-83, 2001

An Overview of a Tool for Extracting Rules from Databases with Incomplete Information

Hiroshi SAKAI

Department of Computer Engineering, Faculty of Engineering, Kyushu Institute of Technology

(*received April 2, 2001*)

Abstract: *The problem of knowledge discovering in the form of rules from databases with incomplete information is studied. At first, the rule extraction from databases without incompleteness is surveyed according to the rough sets theory. Then, databases with incomplete information and the rule extraction from these databases are outlined. We briefly survey our previous research, and apply it for realizing some programs of the rule extraction. The implemented programs and the real execution of these programs are shown, too. In this way, a tool for extracting rules from databases with incomplete information is proposed.*

Keywords: *Rule extraction, Incomplete information, Database, Data dependency, Rough sets, Tool.*

1. Introduction

Databases with incomplete information were proposed by Lipski in 1981 [1,2]. This is well known as a framework to handle the uncertain information in relational databases. In usual relational databases, there is an attribute value for every object (or a primary key) and every attribute [3]. However in Lipski's database, there may be a set of attribute values for some objects and some attributes. This set of attributes is interpreted as that the actual value exists in this set, but it is not known for the lack of information [1]. Codd's database with null value [4] is a special case of Lipski's database, since the null value shows the whole attribute value domain. Lipski mainly discussed the modal question-answering and the axiomatic system for this question-answering.

In this paper, the rule extraction from databases with incomplete information is studied. The motivation of this research is due to the recent works on the KDD (Knowledge Discovery and Data Mining) [5]. There are several works on the KDD from very large databases [6], but the knowledge discovery from the uncertain information will also be an important work. The rule extraction from databases with incomplete information is a framework for the new variation of KDD.

Here, we remark that we deal with non-numerical data, and we do not apply the fuzzy analysis, the statistical analysis nor the multivariate analysis. We mainly depend upon the data dependency. This is a way of rough sets analysis [7,8].

Tobata, Kitakyushu 804, Japan
Phone and Fax: +81-93-884-3258
sakai@comp.kyutech.ac.jp

2. Basic Concepts in Rough Sets Theory

Every *DIS* (*Deterministic Information System*) [7] is a quadruplet

$DIS=(OB, AT, \{VALa \mid a \in AT\}, f)$,

where OB is a finite set whose elements are called *objects*, AT is a finite set whose elements are called *attributes*, $VALa$ ($a \in AT$) is a finite set whose elements are called *attribute values*, and f is such a mapping as $f: OB \times AT \rightarrow \cup aVALa$. The mapping f is called a *classification function*.

For $x, y (x \neq y) \in OB$, if $f(x,a)=f(y,a)$ for every $a \in AT$, we see there is a relation for x and y, which becomes an equivalence relation over OB. Every element of an equivalence relation is called an *equivalence class*, and $[x]$ denotes an equivalence class with object x. For every equivalence class, either $[x]=[y]$ or $[x] \cap [y]=\{\}$ holds. Furthermore, $\cup_{x \in OB}[x] = OB$ holds. If a set $X \subset OB$ is the union of some equivalence classes, we say X is *definable* in *DIS*. Otherwise, we say X is *rough*.

Let's consider two sets $CON \subset AT$ which is called *condition attributes* and $DEC \subset AT$ which is called *decision attributes*. We say objects x and y are *consistent*, if $f(x,a)=f(y,a)$ for every $a \in CON$ then $f(x,a)=f(y,a)$ for every $a \in DEC$. If every two objects are consistent each other in a *DIS*, we say *DIS* is *consistent* for *CON* and *DEC*. We may say *there is a strict dependency* between *CON* and *DEC*.

3. Rule Extraction based on Dependency

If a *DIS* is consistent for *CON* and *DEC*, every tuple restricted to *CON* and *DEC* is a rule [7,8]. Namely for a fixed *DEC* in a *DIS*, such *CON* as *DIS* is consistent for

CON and *DEC* is obtained at first. Then, tuples restricted to *CON* and *DEC* are extracted. They are rules from this *DIS*. This is the simplest case of the rule extraction based on a dependency.

Les's consider two equivalence relations eq_1 and eq_2 over *OB*. $eq_1 \subset eq_2$ denotes that there exists such $M \in eq_2$ as $L \subset M$ for every $L \in eq_1$. Here, an important proposition connecting the dependencies in *DISs* with equivalence relations over *OB* is shown.

Proposition 1 [7]. For any *DIS*, suppose *eq*(*CON*) be an equivalence relation for the condition attributes and *eq*(*DEC*) be an equivalence relation for the decision attributes. In this case, the following (1) and (2) are equivalent.
(1) *There is a strict dependency between CON and DEC.*
(2) $eq(CON) \subset eq(DEC)$.

If there is not any strict dependency between *CON* and *DEC*, a set $POS_{CON}(DEC) = \cup \{L \in eq(CON) \mid$ there is such $M \in eq(DEC)$ as $L \subset M\}$ is applied to characterize the dependency. This is a set of all consistent objects for *CON* and *DEC*, and it is called (*CON*-)*positive region* of *DEC*. The ratio $|POS_{CON}(DEC)|/|OB|$ is called the *degree of dependency* between *CON* and *DEC*. Intuitively, the high degree of dependency implies there exist many rules between *CON* and *DEC*.

Example 1. Let's consider DIS_1 in Table 1.

Table 1. A Table for DIS_1

OB	*color*	*shape*	*size*
1	*red*	*square*	*small*
2	*blue*	*square*	*large*
3	*red*	*square*	*small*
4	*blue*	*round*	*medium*

There are four objects and three attributes. In Table 1, $eq(\{color\})=\{\{1,3\},\{2,4\}\}$. A set $\{1,2,3,4\}=\{1,3\} \cup \{2, 4\}$ is definable for the attribute *color*, but a set $\{1,2,3\}$ is rough. For an attribute set {*shape*, *size*}, $eq(\{shape, size\})=\{\{1,3\},\{2\},\{4\}\}$. Therefore in this case, both $\{1,2, 3,4\}$ and $\{1,2,3\}$ are definable. Let's discuss the dependency. For $CON=\{shape, size\}$ and $DEC=\{color\}$, $\{1,3\} \subset \{1,3\}$, $\{2\} \subset \{2,4\}$ and $\{4\} \subset \{2,4\}$ hold. Namely, the set $POS_{CON}(DEC) = \{1,2,3,4\} = OB$, and it is known there exists a strict dependency between {*shape*, *size*} and {*color*}. As for $CON=\{shape\}$ and $DEC=\{color\}$, the positive region is $\{4\}$. Namely, the attribute *size* can't be reduced from {*shape*, *size*}. As for $CON=\{size\}$ and $DEC=\{color\}$, the positive region is equal to *OB*. The degree of dependency between {*size*} and {*color*} is 1, therefore the attribute *shape* can be reduced from {*shape*, *size*}. Finally, rules are extracted. Here, *rule*(*x*) denotes the rule from object *x*.

rule(1)(=*rule*(3)): [*size*=*small*]→[*color*=*red*],
rule(2):[*size*=*large*]→[*color*=*blue*],
rule(4):[*size*=*medium*]→[*color*=*blue*].

Like this, rules are extracted by using the dependency in a *DIS*. For numerical data, there are some other ways of data analysis. However in DIS_1, there do not exist any numerical data. In such data, some ways of numerical data analysis are not applicable. We depend upon the way by a data dependency, namely the way of rough sets theory.

4. Databases with Incomplete Information

There may be some frameworks for databases with incomplete information, so we clarify our framework in this section. We identify the following *Non-deterministic Information System* (*NIS*) with a database with incomplete information. Every *NIS* is also a quadruplet

$NIS=(OB, AT, \{VALa \mid a \in AT\}, g)$,

where g is such a mapping as $g: OB \times AT \rightarrow P(\cup aVALa)$ (Power set) [1,2]. For every set $M \in P(\cup aVALa)$, M is interpreted as that there exists an actual value in this set but it is not known for the lack of information. If the actual value is not known at all, $g(x,a)$ is equal to *VALa*. A *NIS* is seen as a *DIS* with incomplete information.

Orlowska and Pawlak discussed the logical formalism for representing incomplete knowledge in *NISs* [7,8]. They proposed logic *NIL* by using modal operators like [*] and <*>.

5. An Example and Some Remarks

An example of *NIS* is shown, and the problem in this paper is clarified.

Example 2. Let's consider next NIS_1, where $VAL_A=\{0,1, 2\}$, $VAL_B=\{0,1,2,3\}$, $VAL_C=VAL_D=\{0,1,2,3,4,5\}$. Random number programs are applied for producing this table. In such a table, how do we deal with the rule extraction?

Table 2. A Table for NIS_1

OB	*A*	*B*	*C*	*D*
1	0	{2,3}	5	3
2	0	{0,1,3}	4	5
3	2	3	5	{0,4,5}
4	{0,1,2}	3	1	1
5	1	{0,1}	5	0
6	2	1	{3,5}	4
7	1	0	2	2
8	1	3	{1,2,4}	4
9	0	1	5	4
10	2	1	0	1

In this table, there are three selections for $g(4,A)=\{0,1,2\}$, namely 0, 1 and 2. For other sets of attribute values, there are two or three selections of an attribute value. Therefore in NIS_1, $648(=2^3 \times 3^4)$ *DISs* can be derived. We generally call such *DISs derived DISs from NIS*. Table 3 shows a derived *DIS* for attributes $\{A,B,C\}$. For attributes $\{A,B,C\}$, there are $216(=2^3 \times 3^3)$ derived *DISs* in NIS_1. According to the interpretation of $g(x,a)$, it is known there is a derived *DIS* with actual information, but it is not known.

Table 3. A Table for a derived DIS from NIS_1 for attributes {A,B,C}

OB	*A*	*B*	*C*
1	0	2	5
2	0	0	4
3	2	3	5
4	0	3	1
5	1	0	5
6	2	1	3
7	1	0	2
8	1	3	1
9	0	1	5
10	2	1	0

Now we discuss three cases of the rule extraction in Table 2.

(CASE 1) Suppose $CON=\{B\}$ and $DEC=\{D\}$.
In this case, there are $36(=2^2\times 3^2)$ derived *DISs*. As for the object 1, there are two rules,

$rule(1)_1$:$[B=2]\rightarrow[D=3]$,

$rule(1)_2$:$[B=3]\rightarrow[D=3]$.

The $rule(1)_1$ appears in 18 derived *DISs* and it is always consistent in 18 derived *DISs*. On the contrary, $rule(1)_2$ also appears in other 18 derived *DISs*. But the $rule(1)_2$ is always inconsistent with the *rule*(4). Furthermore, the *rule*(9) and the *rule*(10) appear in all derived *DISs*, and they are always inconsistent. Namely, every derived *DIS* is inconsistent for $CON=\{B\}$ and $DEC=\{D\}$, and there exists no strict dependency in any derived *DISs*.

(CASE 2) Suppose $CON=\{A, B, C\}$ and $DEC=\{D\}$.
In this case, there are $648(=2^3\times 3^4)$ derived *DISs*. The $rule(1)_1$ and the $rule(1)_2$ are always consistent with any other rules in all derived *DISs*, respectively. Rules only

$rule(4)_2$:$[A=1]\wedge[B=3]\wedge[C=1]\rightarrow[D=1]$,

$rule(8)_1$:$[A=1]\wedge[B=3]\wedge[C=1]\rightarrow[D=4]$

are inconsistent. These two rules appear in 72 derived *DISs* at the same time. Therefore, 576(648-72) derived *DISs* are consistent. In this case, there exist less inconsistent *DISs*, and we see every tuple is a rule. Especially, the *rule*(7), the *rule*(9) and the *rule*(10) appear in all derived *DISs*, and they are always consistent with any other rules, therefore we see such rules are certain rules. As for other rules, we see they are possible rules.

(CASE 3) Suppose $CON=\{A, B, D\}$ and $DEC=\{C\}$.
In this case, there are $648(=2^3\times 3^4)$ derived *DISs* and every derived *DIS* is consistent. Namely, any two rules are consistent. This is the most preferable case. Like CASE 2, we see the *rule*(7), the *rule*(9) and the *rule*(10) are certain rules. For the object 1, we see the $rule(1)_1$ and the $rule(1)_2$ are possible rules, respectively. Furthermore, we see the $rule(1)_1$ or the $rule(1)_2$ certainly holds.

In CASE 2, the almost rules are consistent, but in CASE 1 the almost rules are inconsistent. In CASE 3, every rule is consistent with other rules. According to the discussion, every rule is divided to the following sorts.

1. **Global-Consistent rule:** This is such a rule as that it appears in all derived *DISs* and it is always consistent with other rules, like the *rule*(7), in CASE 2.
2. **Local-Consistent rule:** This is such a rule as that it appears in some derived *DISs* and it is consistent with other rules in these *DISs*, like the $rule(1)_1$ in CASE 1.
3. **Global-Inconsistent rule:** This is such a rule as that it appears in all derived *DISs* and it is inconsistent with other rules in all derived *DISs*, like the *rule*(9) in CASE 1.
4. **Local-Inconsistent rule:** This is such a rule as that it appears in some derived *DISs* and it is inconsistent with other rules in these *DISs*, like the $rule(1)_2$ in CASE 1.
5. **Global rule, Local rule:** They are rules except the above rules.

In these sorts, the global-consistent rule is the most preferable, and the global-inconsistent rule is the worst. In any *DIS*, there are only two sorts of rules, i.e., the consistent rule and the inconsistent rule, but there are five sorts of rules in every *NIS*.

In the subsequent section, the next issues related to five sorts of rules are discussed.

(Issue 1) *How do we deal with the dependency among attributes in every NIS?*

(Issue 2) *How do we get CON, which cause a high degree of the dependency for fixed DEC?*

(Issue 3) *How do we extract rules from every NIS?*

As for Issue 1, a new data dependency in every *NIS* is proposed. As for Issue 2 and Issue 3, some useful algorithms are proposed for the implementation.

6. Surveys of our Research

This section surveys our research [9,10]. In every *DIS*, there exists an equivalence relation for any set of attributes. We call every equivalence relation in a derived *DIS* a *possible equivalence relation* (*pe-relation*), and we call every element in a possible equivalence relation a *possible equivalence class* (*pe-class*).

As for the definability of a set in *NIS*, we rely on the definition by Orlowska. Namely a set X is *definable* in *NIS* if X is definable in some derived *DISs*. It implies that if we pick up some appropriate derived *DISs* then the set is definable in those *DISs*.

The simple way to check the definability of a set is to examine the definability of a set for all derived *DISs*. However, this way will not be suitable for the *NIS* with large number of derived *DISs*. The order to check the definability of a set in a *DIS* depends on $|\mathrm{OB}|^2$, and it is necessary to repeat this check for all derived *DISs*. The problem to check the definability of a set is in *NP* class. More effective way for the simple way is needed, and another way is proposed by the next proposition.

Proposition 2. For $NIS=(OB,AT,\{VALa\,|a\in AT\},g)$ and a set $X\subset OB$, if there exist subsets of OB, $CL_1,\cdots, CL_m$ satisfying (1) and (2), then X is definable in *NIS*. Vice versa.

(1) $\cup_i CL_i=X$.

(2) $\{CL_1,\cdots, CL_m\}$ *is a subset of a pe-relation.*

By finding subsets of OB, $CL_1,\cdots,CL_m$, we check the definability of a set in every *NIS*. The following is the overview of an algorithm for solving it.

Algorithm 1 [9,10].
Input: A *NIS* and a set $X \subset OB$.
Output: X is definable in *NIS* or not.
(1) $X^*=X$ and $eq=\{\}$.
(2) For any element $x \in X^*$, find such a CL as
(CL-1) $x \in CL$, $CL \subset X^*$,
(CL-2) $eq \cup \{CL\}$ is a subset of a pe-relation.
(2-1) If there exists a CL, $[x]=CL$, $eq=eq \cup \{[x]\}$ and $X^*=X^*-CL$. If $X^* \neq \{\}$, go to (2). Otherwise, X is definable in *NIS*.
(2-2) If there exists no CL, backtrack. If there is no branch to backtrack, the set X is not definable.

This algorithm is similar to Grzymala-Busse's LEM1 and LEM2 algorithms [11]. The proposing algorithm is an extension of Grzymala-Busse's algorithm to the non-deterministic information systems. In the proposing algorithm, it is very difficult to realize (CL-1) and (CL-2). Some properties for solving those problems are in [9,10].

Example 3. Let's show an example of *NIS*, and simulate the transaction of Algorithm 1.

Table 3. A Table for a NIS

OB	A
1	5
2	7
3	{5, 7}

In Table 3, there are clearly two pe-relations $\{\{1,3\},\{2\}\}$ and $\{\{1\},\{2,3\}\}$. In this table, the following *internal expressions* are picked up at first.

$inf(1,(5),\{A\})=\{1\}$, $sup(1,(5),\{A\})=\{1,3\}$,
$inf(2,(7),\{A\})=\{2\}$, $sup(2,(7),\{A\})=\{2,3\}$,
$inf(3,(5),\{A\})=sup(3,(5),\{A\})=\{1,3\}$,
$inf(3,(7),\{A\})=sup(3,(7),\{A\})=\{2,3\}$.

The $inf()$ implies certain pe-class and the $sup()$ implies the possible pe-class. Only such a set X as $inf() \subset X \subset sup()$ can be a pe-class [10]. The CL in Algorithm 1 must satisfy this condition. Furthermore in order to manage the sequence of sets $CL_1,\cdots, CL_m$, two lists *PLIST* and *NLIST* are employed [10]. *PLIST* keeps selected tuples and *NLIST* keeps rejected tuples. For the first element $1 \in X^*=\{1,2,3\}$, there are two pe-classes $\{1\}$ and $\{1,3\}$. For the selection of $\{1\}$, $X^*=\{2,3\}$, $PLIST=\{[1,(5)]\}$, $NLIST=\{[3,(5)]\}$ and $eq=\{\{1\}\}$ are derived. The tuple (5) for the object 3 is rejected for being the pe-class $\{1\}$. For the first element $2 \in X^*=\{2,3\}$, there are two pe-classes $\{2\}$ and $\{2,3\}$. However, the pe-class $\{2\}$ can't be applicable. Because in this case, $NLIST$ = {[3,(5)], [3,(7)]} holds, and it implies all tuples for the object 3 are rejected. For $X^*=\{2,3\}$, only pe-class $\{2,3\}$ is applicable, and $X=\{\}$, $PLIST$ = {[1,(5)],[2,(7)],[3,(7)]}, $NLIST$ = {[3,(5)]} and $eq=\{\{1\},\{2,3\}\}$ are derived. Let's consider another pe-relation. For the first element $1 \in X^*=\{1,2,3\}$, suppose a pe-class $\{1,3\}$ is selected. Then, $X^*=\{2\}$, $PLIST=\{[1,(5)],[3,(5)]\}$, $NLIST=\{\}$ and $eq=\{\{1,3\}\}$ are derived. For the $2 \in X^*=\{2\}$, there are two pe-classes $\{2\}$ and $\{2,3\}$, and $\{2\}$ is only applicable to $X^*=\{2\}$. Finally, $X^*=\{\}$, $PLIST=\{[1,(5)],[2,(7)],[3,(5)]\}$, $NLIST=\{[3,(7)]\}$ and $eq=\{\{1,3\},\{2\}\}$ are derived.

For realizing Algorithm 1, at first a prolog program *candidate* is implemented, which picks up CL satisfying conditions (CL-1) and (CL-2). The following program *class0* is the main program for checking the definability of a set.

```
class0(ATR,X,Y,EQ,Ppre,Pres,Npre,Nres)
  :-X==[],EQ=Y,Pres=Ppre,Nres=Npre.
class0(ATR,[X|X1],Y,EQ,Ppre,Pres,Npre,Nres)
  :-candidate(ATR,[X|X1],CAN,Ppre,Pres1,Npre,Nres1),
  minus([X|X1],CAN,REST),
  class0(ATR,REST,[CAN|Y],EQ,Pres1,Pres,Nres1,Nres).
```

In this program, variables EQ, $Pres$ and $Nres$ manage the subset of pe-relations, $PLIST$ and $NLIST$, respectively.

The following three solutions show the definability of a set {7,8,9} for condition attributes $CON=\{A,B,C\}$ in Example 2. The positive selection shows the tuple of every object. For example, the first solution shows that if we select 1 from a set $g(8,C)=\{1,2,4\}$ and we do not select 1 from a set $g(4,A)=\{0,1,2\}$, the set {7,8,9} is definable in NIS_1. The positive and negative selections show the conditions for being definable in *NISs*.

```
?-class(con,[7,8,9]).
[1] RELATION: [[7],[8],[9]]
    POSITIVE SELECTION
        CONDITION OF 7: [1,0,2]
        CONDITION OF 8: [1,3,1] *
        CONDITION OF 9: [0,1,5]
    NEGATIVE SELECTION
        CONDITION OF 4: [1,3,1] *
[2] RELATION: [[7],[8],[9]]
    POSITIVE SELECTION
        CONDITION OF 7: [1,0,2]
        CONDITION OF 8: [1,3,2] *
        CONDITION OF 9: [0,1,5]
    NEGATIVE SELECTION
[3] RELATION: [[7],[8],[9]]
    POSITIVE SELECTION
        CONDITION OF 7: [1,0,2]
        CONDITION OF 8: [1,3,4] *
        CONDITION OF 9: [0,1,5]
    NEGATIVE SELECTION
EXEC_TIME = 0.016(sec)
```

7. Algorithm 1 and Pe-relations

In Algorithm 1, the side effect of this algorithm is very useful. If a set X is definable, Algorithm 1 assigns a set of pe-classes $\{CL_1,\cdots, CL_m\}$ to the variable eq. Here, $\cup_i CL_i=X$ and $\{CL_1,\cdots, CL_m\}$ is a subset of a pe-relation. If $X=OB$, every pe-relation satisfies these conditions.

Namely, if X=OB is solved by Algorithm 1, every pe-relation can be obtained as a side effect. By using this property, we are obtaining all pe-relations over OB.

8. Proposal of a Dependency in Databases with Incomplete Information

In this section, our previous research is applied to the rule extraction from databases with incomplete information. At first, a new dependency between *CON* and *DEC* is proposed [10].

A proposal of a dependency in every NIS

Let's consider a *NIS*, all derived $DIS_1,\cdots,DIS_m$ from *NIS*, the condition attributes *CON*, the decision attributes *DEC*, two threshold values val_1 and val_2 ($0 \leqq val_1, val_2 \leqq 1$). If the following conditions D1 and D2 hold, we see *there exists a dependency between CON* and *DEC*.

(D1) [(number of derived consistent *DISs*) / m] $\geqq$ val_1.

(D2) The minimal degree of dependency $\geqq$ val_2.

In this proposal, the condition (D1) requires the almost derived *DISs* are consistent. The condition (D2) requires that the degree of dependency is more than a threshold value. Suppose CON=$\{A,B\}$, DEC=$\{C\}$, val_1 =0.8 and val_2=0.8 in Example 2. In this case, there are 216(=$2^3 \times 3^3$) *DISs*. The condition (D1) requires 173 *DISs* must be consistent in 216 *DISs*, and the condition (D2) requires the minimal degree of dependency in 216 *DISs* must be more than 0.8.

This proposal is an extension of a dependency in *DISs* to *NISs*. According to these criteria, we handle the dependency in every *NIS*. Namely for the condition attributes *CON* and decision attributes *DEC* with high degree of dependency, we see the tuples restricted to *CON* and *DEC* are rules. We call this rule extraction the *rule extraction based on the dependency* in every *NIS*.

However, to calculate these criterion values requires the degree of dependency in every derived *DIS*. Of course, it is able to get every degree of dependency by examining 648 *DISs* sequentially, but this way is ineffective. We really implemented programs by another way, i.e., by using the pe-relations and Proposition 1.

In NIS_1, let's consider a case that CON=$\{A,B,C\}$ and DEC=$\{D\}$. There are two kinds of pe-relations for *CON* and three kinds of pe-relations for *DEC*. For pe-relations $\{\{1\},\{2\},\{3\},\{4,8\},\{5\},\{6\},\{7\},\{9\},\{10\}\}$ in *CON* and $\{\{1\},\{2,3\},\{4,10\},\{5\},\{6,8,9\},\{7\}\}$ in *DEC*, $POS_{CON}(DEC)$=$\{1\} \cup \{2\} \cup \{3\} \cup \{5\} \cup \{6\} \cup \{7\} \cup \{9\} \cup \{10\}$=$\{1,2,3,5,6,7,9,10\}$. Except objects 4 and 8, other objects are consistent with each other. The degree of dependency is 0.8=(8/10). In this way, all degrees of dependency are calculated by repeating 6(=2×3) combinations of *CON* and *DEC*.

9. Total Procedure to Extract Rules

In this section, the total procedure to extract rules is presented. The sequence consists of the following steps.

(Step 1) Make a data file and an attribute file.

(Step 2) Execute a program and produce the internal expressions.

(Step 3) Execute a program to obtain all pe-relations for *CON* and *DEC*, respectively.

(Step 4) Execute a program for checking the dependency between *CON* and *DEC*. If there exists a dependency, go to the Step 5 else go to the Step 1.

(Step 5) Execute a program to extract rules.

By using NIS_1, the execution of every step is shown. These programs are implemented on a workstation with 167MHz UltraSparc CPU. Prolog and C language are employed for the implementation. Prolog was necessary for a search with backtracking, especially for getting pe-relations. In the following execution, the prompt of prolog is '?-' , and the prompt of C is '%'.

The syntax of the data file and attribute file is very simple. This is the data file data.pl in Example 2.

```
object(10,4),
data(1,[0,[2,3],5,3]), data(2,[0,[0,1,3],4,5]), ···
data(9,[0,1,5,4]), data(10,[2,1,0,1]).
```

This is the attribute file attrib.pl in Example 2. The 1, 2, 3 and 4 in the attribute file correspond to attributes A, B, C and D, respectively.

```
condition([1,2,3]).
decision([4]).
```

The following is the real execution in Step 2.

```
?- step2.
File Name :'data.pl'.
Attribute File Name:'attrib.pl'.
Output File Name:'data.rs'.
EXEC_TIME = 0.135 (sec)
```

After this translation, new file data.rs is created, which stores the internal expressions for data.pl and attrib.pl.

```
?- relall(con).
[1] [[1],[2],[3],[4,8],[5],[6],[7],[9],[10]]   24
[2] [[1],[2],[3],[4],[5],[6],[7],[8],[9],[10]]   192
POSSIBLE CASES   216
EXEC_TIME = 0.535 (sec)
yes
?- relall(dec).
[1] [[1],[2,3],[4,10],[5],[6,8,9],[7]]   1
[2] [[1],[2],[4,10],[5,3],[6,8,9],[7]]   1
[3] [[1],[2],[4,10],[5],[6,8,9,3],[7]]   1
POSSIBLE CASES   3
EXEC_TIME = 0.014 (sec)
```

The first solution of a query relall(con) shows there are 216 derived *DISs* and two kinds of pe-relations for condition attributes $\{A, B, C\}$. The 24 derived *DISs* have the same pe-relation $\{\{1\},\{2\},\{3\},\{4,8\},\{5\},\{6\},\{7\},\{9\},\{10\}\}$ and the 192 derived *DISs* have the same pe-relation $\{\{1\},\{2\},\{3\},\{4\},\{5\},\{6\},\{7\},\{8\},\{9\},\{10\}\}$. In this execution, two files are created, i.e., eqall.con with

pe-relations for *CON* and eqall.dec with pe-relations for *DEC*. In Step 4, a program for calculating the dependency is executed.

```
% step4
CRITERION 1
   Number of Derived DISs: 648
   Number of Derived Consistent DISs: 576
   Degree of Consistent DISs: 0.889
CRITERION 2
   Minimal Degree of Dependency: 0.800
   Maximal Degree of Dependency: 1.000
EXEC_TIME = 0.200 (sec)
```

It is known that there are 576 derived consistent *DISs* and the minimal degree of dependency is 0.8. For threshold values val_1=0.8 and val_2=0.8, there exists a dependency between $\{A,B,C\}$ and $\{D\}$ in NIS_1. Namely, it is able to discuss the rule extraction by this dependency. For such *CON* and *DEC*, the program step5 is applied.

```
?- step5.
Rules from [A,B,C] to [D]
[2,3,5] => [0]   (1, 3) from 3    local-consistent
[1,0,5] => [0]   (2, 1) from 5    local-consistent
[1,1,5] => [0]   (2, 1) from 5    local-consistent
[0,3,1] => [1]   (3, 1) from 4    local-consistent
[1,3,1] => [1]   (3, 1) from 4    local
[2,3,1] => [1]   (3, 1) from 4    local-consistent
[2,1,0] => [1]   (1, 1) from 10   global-consistent
      :    :     :     :
[1,3,1] => [4]   (3, 1) from 8    local
      :    :     :     :
[0,3,4] => [5]   (3, 1) from 3    local-consistent
[2,3,5] => [5]   (1, 3) from 3    local-consistent
EXEC_TIME = 0.032 (sec)
```

In the execution of the program step5, local rules [1,3,1] →[1] from the object 4 and [1,3,1]→[4] from the object 8 are inconsistent. They appear in 72 derived *DIS* at the same time, namely there are 72 inconsistent derived *DISs*. However, any other rule is either a global-consistent rule or a local-consistent rule in all derived DISs. There is less inconsistency of rules in this case. Like this, the new dependency is an important measure for discussing the rule extraction.

Now, let's consider other case that *CON*=$\{B\}$ and *DEC*=$\{D\}$ in NIS_1, which is CASE 1 in Section 5. The following is the result of program step4.

```
% step4
CRITERION 1
   Number of Derived DISs: 36
   Number of Derived Consistent DISs: 0
   Degree of Consistent DISs: 0.000
CRITERION 2
   Minimal Degree of Dependency: 0.000
   Maximal Degree of Dependency: 0.200
EXEC_TIME = 0.190 (sec)
```

In this case, all derived *DISs* are inconsistent. The maximal degree of dependency is 0.2. This result shows there is less dependency between $\{B\}$ and $\{D\}$ in NIS_1. The extracted rules for *CON*=$\{B\}$ and *DEC*=$\{D\}$ are almost inconsistent rules.

```
?- step5.
Rules from [B] to [D]
[3] => [0]   (1, 3) from 3    local-inconsistent
[0] => [0]   (2, 1) from 5    local-inconsistent
[1] => [0]   (2, 1) from 5    local-inconsistent
[3] => [1]   (1, 1) from 4    global-inconsistent
      :    :     :     :
[3] => [5]   (1, 3) from 3    local-inconsistent
EXEC_TIME = 0.016(sec)
```

There is a local-consistent $rule(1)_1$: [2]→[3] from the object 1. Other rule is either local-inconsistent or global-inconsistent.

We have shown the real sequence of execution. For *CON*=$\{A,B,C\}$ and *DEC*=$\{D\}$, good criterion values existed, and the almost rules are better than the local-consistent. However for *CON*=$\{B\}$ and *DEC*=$\{D\}$, two criterion values are bad, and the almost rules are worse than local-inconsistent.

10. Remarks on the Execution Time

Now in this section, we refer to the execution time from Step 2 to Step 4. Four *NISs* in Table 4 are automatically produced based on the random number program. Let's consider these four *NISs*, *CON*=$\{A,B,C\}$ and *DEC*=$\{D\}$.

Table 5 shows the execution time, and the 6-th column shows the time by the simple method, namely (number of derived *DISs*) × (execution time to obtain an equivalence relation in a derived *DIS* by the simple method). Here, the simple method compares each two tuples, and extracts an equivalence relation in a *DIS*.

Table 4. Conditions in Four NISs

	$\|OB\|$	$\|AT\|$	Derived *DISs*
NIS_1	10	4	864
NIS_2	100	4	1944
NIS_3	300	4	3888
NIS_4	1000	4	7776

Table 5. Execution Time (sec) in Every Step

	Step 2	Step 3	Step 4	Total	Simple method
NIS_1	0.127	0.259	0.000	0.386	-
NIS_2	1.430	2.397	0.100	3.927	-
NIS_3	15.329	23.780	0.200	39.309	388.800
NIS_4	56.876	137.147	0.700	194.723	8553.600

In Table 5, Step 2 and Step 3 take the almost total execution time. On the other hand, Step 4 takes less time. In Step 3, the program *step3* depending upon Algorithm 1 is executed, and the search with backtracking is necessary in this program. Therefore, prolog language is employed. Prolog is useful for problems solved by search procedures, but it usually takes much execution time. In

Step 4, the program *step4* is implemented by C language, because the search with backtracking is not necessary in Step 4. According to such reasons, it is known that to obtain all pe-relations in a *NIS* is the most time-consuming step in the total procedure.

The simple method may be useful for small size *NISs*, but the simple method is not suitable for *NIS* with large number of derived *DISs* like NIS_4. In NIS_1 and NIS_2 in Table 5, the execution time was 0.000(sec) for a derived *DIS*, respectively. Therefore, the symbol – is placed in Table 5.

11. Concluding Remarks

An overview of a tool for extracting rules from databases with incomplete information is presented. In such databases with non-numerical data, it is not able to use the fuzzy analysis, the statistical analysis nor the multivariate analysis. We relied on the data dependency, and discussed a way to extract rules. This is also a way based on the rough sets theory. Even if the information is not strict, it is able to get some useful rules, like global-consistent rules and local-consistent rules.

We will not apply these programs to *NISs* with very large size of objects, but apply them to *NISs* with small size of objects and large number of derived *DISs*. These programs will fill the role of a hypotheses generator from *NISs*, too.

We have examined the correctness of the programs by several *NISs*, which we got by using random number programs. However, we need to deal with actual data toward the real application, and need to discuss the combination of our way and typical fuzzy way, too.

Acknowledgment

The author would be grateful to anonymous referees for their useful comments, and wishes to express his gratitude to Prof. Torao YANARU.

References

[1] Lipski, W. (1981): "On databases with Incomplete Information", J.ACM, Vol.28, No.1, pp.41-70.

[2] Lipski, W. (1979): "On Semantic Issues Connected with Incomplete Information Data Base", ACM Transaction on Database Systems, Vol.4, pp.269-296.

[3] Date, C.J. (1995): "An Introduction to Databases Systems", Addison-Wesley.

[4] Codd, E.F. (1979): "Extending the Database Relational Model to Capture More Meaning", ACM Trans. Database Systems, Vol.4, No.4, pp.397-434.

[5] Komorowski, J. and Zytkow, J. (Eds.) (1997): "Principles of Data Mining and Knowledge Discovery", Lecture Notes in AI, Springer-Verlag, Vol.1263.

[6] Agrawal, A., Imielinski, T. and Swami, A. (1993): "A Database Mining", IEEE Trans. on Knowledge and Data Engineering, Vol.5, pp.914-925.

[7] Pawlak, Z. (1991): "Rough Sets", Kluwer Academic Publisher.

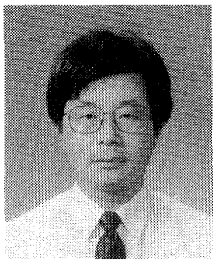
[8] Orlowska, E. and Pawlak, Z. (1984): "Representation of Nondeterministic Information", Theoretical Computer Science, Vol.29, pp.27-39.

[9] Sakai, H. and Okuma, A. (1999): "An Algorithm for Finding Equivalence Relations from Tables with Non-deterministic Information", Proc.RSFDGrC99, Lecture Notes in AI, Springer-Verlag, Vol.1711, pp. 64-72.

[10] Sakai, H. and Okuma, A. (2000): "An Algorithm for Checking Dependencies of Attributes in A Table with Non-deterministic Information: A Rough Sets Based Approach", Proc. PRICAI2000, Lecture Notes in AI, Springer-Verlag, Vol.1886, pp.219-229.

[11] Grzymala-Busse, J. (1997): "A New Version of the Rule Induction System LERS", Fundamenta Informaticae, Vol.31, pp.27-39.

* * *



Hiroshi SAKAI

He received the Dr. degree in computer science from Kyushu University in 1988. He is working as an associate professor of Department of Computer Engineering, Faculty of Engineering at Kyushu Institute of Technology. He is interested in incomplete information systems, logic programming, knowledge-based system, rough set theory and soft computing.