ON A METHOD TO EXTRACT RULES FROM A TABLE WITH NON-DETERMINISTIC INFORMATION: A ROUGH SETS BASED APPROACH

Sakai, Hiroshi
Department of Computer Engineering, Kyushu Institute of Technology

https://doi.org/10.5109/13506

出版情報：Bulletin of informatics and cybernetics. 34 (1), pp.13-28, 2002-10. Research Association of Statistical Sciences
バージョン：
権利関係：

Bulletin of Informatics and Cybernetics, Vol. 34, No. 1, 2002

ON A METHOD TO EXTRACT RULES FROM A TABLE WITH NON-DETERMINISTIC INFORMATION: A ROUGH SETS BASED APPROACH

By

Hiroshi SAKAI*

Abstract

Rough sets theory is now becoming a mathematical foundation of soft computing. This theory makes use of equivalence relations defined for each set of attributes in any table, and applies the concept like definability of a set, dependency among attributes, reduction of data, rule extraction, etc., to data analysis.

In this paper, a problem of knowledge discovering in the form of rules from any table with non-deterministic information is discussed. At first, the rough sets based concept including rule extraction is surveyed, and this concept is extended to new one related to non-deterministic information. Then, a framework of rule extraction from tables with non-deterministic information is proposed, and some algorithms for handling such new concept are presented. Also implemented programs and a real execution of these programs are shown.

Key Words and Phrases: Rule extraction, Rough sets, Non-deterministic information, Data dependency, Possible equivalence relation.

1. Introduction

Rough sets theory is seen as a mathematical foundation of soft computing. The theory covers important areas of research in AI such as knowledge, imprecision, vagueness, learning and induction (Pawlak, 1991;1996; Orlowska and Pawlak, 1984; Nakamura et al., 1996). It has been applied to machine learning and knowledge discovery by Polkowski and Skowron (1998a;1998b), Grzymala-Busse (1997), Ras and Joshi (1997), Tsumoto (2000) and Zhong et al. (1998).

Rough sets theory usually handles a table with deterministic information which we call DIS(*Deterministic Information System*). Recently for handling incomplete information like null values (Codd, 1970), unknown values (Lipski, 1981; Grzymala-Busse, 1991; Kryszkiewicz, 1998;1999; Sakai, 1998), etc., rough sets in a table with non-deterministic information, which we call NIS(*Non-deterministic Information System*), attract interest of researchers.

Already some important work on $NISs$ has been conducted, but most work has focused on logic, in particular, an axiomatization of logic in $NISs$ (Orlowska and Pawlak, 1984; Lipski, 1981). There is only few work that deals with real data analysis in $NISs$. For example, Lipski (1981) showed modal question-answering systems

* Department of Computer Engineering, Kyushu Institute of Technology, Tobata Kitakyushu 804 Japan. tel +81-93-884-3258 sakai@comp.kyutech.ac.jp

besides the axiomatization of logic. Grzymala-Busse (1991;1997;1998) surveyed the unknown attribute values, and studied the learning from examples with unknown attribute values. Kryszkiewicz (1998;1999) discussed the theoretical aspect of rules in incomplete information systems. These are the most important work related to $NISs$.

The research of rough sets based data analysis in $NISs$ has still been untouched. Rough sets based data analysis in $NISs$ could also be a mathematical foundation of knowledge discovery and data mining from incomplete information. In such a situation, we are now investigating rough sets based algorithms for data analysis, namely algorithms using equivalence relations in $DISs$ and algorithms using possible equivalence relations in $NISs$. In (Sakai and Okuma, 1999; Sakai, 2001b), a manipulation of possible equivalence relations in $NISs$ was studied. This work proposed a method to obtain all possible equivalence relations in any NIS. A new data dependency among attributes in $NISs$ and algorithms for checking this dependency were also proposed (Sakai and Okuma, 2000; Sakai, 2001a). Algorithms using possible equivalence relations drastically reduced the execution time for checking this data dependency.

In this paper, the framework of rule extraction from $DISs$ is extended to the framework of rule extraction from $NISs$. In every NIS, a rule is defined by a formula satisfying some kinds of constraints. Six groups of rules are introduced into $NISs$ as a constraint, and the proposed dependency in $NISs$ is applied as another constraint to formulas. The problem is reduced to pick up formulas satisfying constraints, and some effective algorithms are needed. For this problem, we applied algorithms using possible equivalence relations, and realized a tool to obtain rules from $NISs$. This framework follows the framework of rule extraction from $DISs$. Thus, it is possible to extract rules not only from $DISs$ but also from $NISs$.

2. A Survey of Rough Sets Theory

This section surveys basic concept in rough sets theory.

2.1. Some Definitions in Deterministic Information Systems

A *Deterministic Information System* (DIS) is a quadruplet $(OB, AT, \{VAL_a | a \in AT\}, f)$, where OB is a finite set whose elements are called *objects*, AT is a finite set whose elements are called *attributes*, VAL_a is a finite set whose elements are called *attribute values* and f is such a mapping that $f : OB \times AT \rightarrow \cup_{a \in AT} VAL_a$ which is called a *classification function*. If $f(x,a)=f(y,a)$ for every $a \in ATR \subset AT$, we see there is a relation between x and y for ATR. This relation is an equivalence relation over OB. $eq(ATR)$ denotes this equivalence relation, and $[x]_{ATR}$ denotes an equivalence class with x. In this case either $[x]_{ATR}=[y]_{ATR}$ or $[x]_{ATR} \cap [y]_{ATR}=\emptyset$ holds for any $x, y \in OB$. Furthermore, $\cup_{x \in OB}[x]_{ATR}=OB$ also holds. If a set $X \subset OB$ is the union of some equivalence classes, we say X is *definable* (for ATR) in DIS. Otherwise we say X is *rough*.

Let us consider two sets $CON \subset AT$ which we call *condition attributes* and $DEC \subset AT$ which we call *decision attributes*. Two distinct objects $x, y \in OB$ are *consistent* (*between CON and DEC*), if $f(x,a)=f(y,a)$ for every $a \in CON$ implies $f(x,a)=f(y,a)$ for every $a \in DEC$. A DIS is *consistent* (*between CON and DEC*), if every pair of objects is consistent. In this case, we may say *DEC depends totally on CON*.

Now, let us show an important proposition connecting dependencies with equiv-

alence relations. Let eq_1 and eq_2 be two equivalence relations over OB. A formula $eq_1 \subset eq_2$ means that for every equivalence class $L \in eq_1$ there exists such an equivalence class $M \in eq_2$ that $L \subset M$.

PROPOSITION 2.1. *(Pawlak, 1982;1991) For any DIS, condition attributes CON and decision attributes DEC, 1 and 2 in the following are equivalent.*

1. *DEC depends totally on CON.*
2. $eq(CON) \subset eq(DEC)$. □

Positive region $POS_{CON}(DEC)=\cup\{L \in eq(CON)|$ there exists such $M \in eq(DEC)$ that $L \subset M\}$ is applied to characterize dependencies. Each degree is measured by a ratio $|POS_{CON}(DEC)|/|OB|$, and this ratio is called *the degree of dependency from CON to DEC*. The degree of dependency is 1 if and only if DEC depends totally on CON.

2.2. Rules in Deterministic Information Systems

Let us survey important work for extracting rules in $DISs$. For any DIS, let CON and DEC be condition and decision attributes, respectively. For any $x \in OB$, a function $d_x : CON \cup DEC \rightarrow \cup_{a \in CON \cup DEC} VAL_a$ such that $d_x(a)=f(x,a)$ is introduced in $DISs$ (Pawlak, 1991). $d_x|CON$ denotes a formula $\wedge_{a \in CON}[a,f(x,a)]$, and $d_x|DEC$ denotes a formula $\wedge_{a \in DEC}[a,f(x,a)]$. The formula $[a, f(x,a)]$ is called a *descriptor*, and it means that $f(x,a)$ is the value of the attribute a. Generally, a rule is defined by an implication $d_x|CON \Rightarrow d_x|DEC$ satisfying some kinds of constraints.

The most familiar constraint is a dependency from CON to DEC. If the degree of dependency is more than a threshold value, each implication in DIS is a rule (Pawlak, 1991). Some criteria are applied to each implication as other constraint. Three criteria in the following are applied to each implication from x (Tsumoto, 2000).

$support(x) = |\{y \in OB|\ d_y|CON=d_x|CON$ and $d_y|DEC=d_x|DEC\ \}|\ /\ |OB|$,
$accuracy(x) = |[x]_{CON} \cap [x]_{DEC}|\ /\ |[x]_{CON}|$,
$coverage(x) = |[x]_{CON} \cap [x]_{DEC}|\ /\ |[x]_{DEC}|$.

If each criterion value of an implication is more than each threshold value respectively, this implication is a rule.

EXAMPLE 2.2. Let us consider a DIS in Table 1.

OB	A	B	C
1	2	2	1
2	1	2	2
3	2	2	1
4	1	3	3

Table 1: A deterministic information system

Here, $eq(\{A\})=\{\{1,3\},\{2,4\}\}$ and $eq(\{C\})=\{\{1,3\},\{2\},\{4\}\}$ hold. A set $\{1,2,3,4\}$ is definable for $\{A\}$, because this set is equal to $\{1,3\} \cup \{2,4\}$. However, a set $\{1,2,3\}$ is rough for $\{A\}$. Both sets are definable for attributes $\{C\}$. Since $eq(\{C\}) \subset eq(\{A\})$, $\{A\}$ depends totally on $\{C\}$, and every implication $d_x|\{C\} \Rightarrow d_x|\{A\}$ $(x \in OB)$ is seen

as a rule. This dependency generates rules in the following.

$d_1|\{C\} \Rightarrow d_1|\{A\}$ $(= d_3|\{C\} \Rightarrow d_3|\{A\})$:[C,1]⇒[A,2],

$d_2|\{C\} \Rightarrow d_2|\{A\}$:[C,2]⇒[A,1], $d_4|\{C\} \Rightarrow d_4|\{A\}$:[C,3]⇒[A,1].

Suppose CON=$\{A\}$ and DEC=$\{C\}$ hold. In this case $POS_{\{A\}}(\{C\})$=$\{1,3\} \neq OB$, and $\{C\}$ does not depend on $\{A\}$. However, two implications from object 1 and 3 are seen as rules under the constraint that *support* ≥ 0.25, *accuracy* ≥ 1.0 and *coverage* ≥ 1.0. Since *accuracy*=0.5 holds for other implications, each other implication is not seen as a rule under this constraint. □

2.3. A Definition of Non-deterministic Information Systems

A *Non-deterministic Information System* (*NIS*) is also a quadruplet $(OB, AT, \{VAL_a | a \in AT\}, g)$, where g is a mapping from $OB \times AT$ to a power set of $\cup_{a \in AT} VAL_a$, i.e., $g : OB \times AT \rightarrow P(\cup_{a \in AT} VAL_a)$ (Orlowska and Pawlak, 1984; Orlowska, 1998). $g(x, a)$ is interpreted as if there is an actual value in this set but it is not known. This is the unknown interpretation for incomplete information (Lipski, 1981; Grzymala-Busse, 1991; Kryszkiewicz, 1998). Especially if the actual value is not known at all, $g(x, a)$ is equal to VAL_a. This is the same as null value interpretation (Codd, 1970).

3. Rules in Non-deterministic Information Systems

This section defines possible implications in *NISs* and two kinds of constraints for them. A rule is defined by a possible implication satisfying two kinds of constraints.

3.1. Possible Implications in Non-deterministic Information Systems

An example of a *NIS* clarifies problems in *NISs*, and possible implications in *NISs* are proposed.

EXAMPLE 3.1. Let us consider a *NIS* such that $OB = \{1, \cdots, 10\}$, $AT = \{A, B, C, D\}$, VAL_1=$\{0, 1, 2\}$, VAL_2=$\{0, 1, 2, 3\}$, VAL_3=VAL_4=$\{0, 1, 2, 3, 4, 5\}$. Table 2 shows the function g in *NIS*. In such a table, how do we deal with the dependency among attributes and rule extraction ? □

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

Table 2: A non-deterministic information system

DEFINITION 3.2. Let $NIS=(OB, AT, \{VAL_a|a \in AT\}, g)$ hold and ATR be a set $\{a_1, \cdots, a_n\} \subset AT$. For any $x \in OB$, $PT(x, ATR)$ denotes Cartesian product $g(x, a_1) \times \cdots \times g(x, a_n)$. We call an element a *possible tuple (for ATR) of x*. Every possible tuple $\zeta=(\eta_{a_1}, \cdots, \eta_{a_n}) \in PT(x, ATR)$ is identified with a formula $\bigwedge_{1 \leq i \leq n}[a_i, \eta_{a_i}]$. □

DEFINITION 3.3. For any NIS, let CON be condition attributes and DEC be decision attributes. For any $x \in OB$, $IMP(x, CON, DEC)$ denotes a set $\{\zeta \Rightarrow \eta | \zeta \in PT(x, CON), \eta \in PT(x, DEC)\}$. We call an element $\zeta \Rightarrow \eta$ a *possible implication from x*. □

Let us consider a case that $CON=\{A, B, C\}$ and $DEC=\{D\}$ in Example 3.1. $PT(1, \{A, B, C\})=\{(0, 2, 5), (0, 3, 5)\}$, $PT(1, \{D\})=\{(3)\}$ and $IMP(1, \{A, B, C\}, \{D\})=\{[A, 0] \wedge [B, 2] \wedge [C, 5] \Rightarrow [D, 3], [A, 0] \wedge [B, 3] \wedge [C, 5] \Rightarrow [D, 3]\}$ hold. A rule in a NIS is defined by a possible implication satisfying some kinds of constraints. Some kinds of constraints to possible implications characterize rules in $NISs$. In the subsequent sections, two kinds of constraints are proposed in $NISs$, and they are applied to possible implications.

3.2. Constraint 1: Data Dependencies

Data dependencies in $DISs$ are extended to dependencies in $NISs$.

DEFINITION 3.4. Let us consider a $NIS=(OB, AT, \{VAL_a|a \in AT\}, g)$, a set of attributes $ATR \subset AT$ and a mapping $h : OB \times ATR \rightarrow \cup_{a \in ATR} VAL_a$ such that $h(x, a) \in g(x, a)$ for any $x \in OB$ and any $a \in ATR$. We call a $DIS=(OB, ATR, \{VAL_a|a \in ATR\}, h)$ a *derived DIS (for ATR form NIS)*. We call an equivalence relation in a derived DIS a *possible equivalence relation (pe-relation)*, and call an element in a pe-relation a *possible equivalence class (pe-class)*. □

In Table 2, there are $648(=2^3 \times 3^4)$ derived $DISs$ for $\{A, B, C, D\}$, and there are $6(=2 \times 3)$ derived $DISs$ for $\{C\}$, According to the interpretation of $g(x, a)$, it is known that there is a derived DIS with real information, but it is not known.

DEFINITION 3.5. Let us consider a NIS, condition attributes CON, decision attributes DEC and all derived $DIS_1, \cdots, DIS_m$ for $CON \cup DEC$. For two threshold values val_1 and $val_2(0 \leq val_1, val_2 \leq 1)$, if conditions 1 and 2 hold then we see *DEC depends on CON* in NIS.

1. $|\{DIS_i|DIS_i(1 \leq i \leq m) \text{ is consistent }\}|/m \geq val_1$.
2. $min_i\{\text{degree of dependency in } DIS_i(1 \leq i \leq m)\} \geq val_2$. □

In Definition 3.5, condition 1 requires most derived $DISs$ are consistent, and condition 2 specifies the minimum value of degrees of dependency. This dependency is applied as a constraint to defining rules in $NISs$.

3.3. Constraint 2: Six Groups of Possible Implications

Six groups of possible implications are introduced in $NISs$. These groups are useful for characterizing each possible implication.

	Globally_consistent	*Marginal*	*Globally_inconsistent*
Definite	*DGC*	*DM*	*DGI*
Indefinite	*IGC*	*IM*	*IGI*

Table 3: Six groups of possible implications in $NISs$

DEFINITION 3.6. Let us consider a NIS, condition attributes CON, decision attributes DEC and any object $x \in OB$. For any possible implication $\psi \in IMP(x, CON, DEC)$, $DDIS(\psi, x, CON, DEC)$ denotes a set $\{\phi |\ \phi$ is such a derived DIS for $CON \cup DEC$ that an implication $d_x|CON \Rightarrow d_x|DEC$ in ϕ is equal to $\psi\}$. □

DEFINITION 3.7. Let us consider a NIS, condition attributes CON, decision attributes DEC and any object $x \in OB$. If $IMP(x, CON, DEC)$ is a singleton set $\{\psi\}$, we say ψ is *definite*. Otherwise we say each $\psi \in IMP(x, CON, DEC)$ is *indefinite*. If $DDIS(\psi, x, CON, DEC) = \{\phi \in DDIS(\psi, x, CON, DEC) |\ x$ is consistent with other objects in $\phi\}$, we say ψ is *globally consistent*. If $\{\phi \in DDIS(\psi, x, CON, DEC) |\ x$ is consistent with other objects in $\phi\} = \emptyset$, we say ψ is *globally inconsistent*. Otherwise we say ψ is *marginal*. Finally, six groups in Table 3 are defined. □

Let us consider a possible implication $\psi : [B, 1] \Rightarrow [C, 5] \in IMP(5, \{B\}, \{C\})$ in Table 2. There exist $72 (= 2^3 \times 3^2)$ derived $DISs$ for $\{B, C\}$, and $DDIS(\psi, 5, \{B\}, \{C\})$ consists of $36 (= 72/2)$ derived $DISs$. Because object 5 is inconsistent with object 10 in all 36 derived $DISs$, ψ belongs to IGI group. There exists no information incompleteness for each possible implication in definite groups. Possible implications in DGC or IGC groups are preferable, and possible implications in DGI or IGI groups are inappropriate. These six groups are also applied as another constraint to defining rules in $NISs$.

3.4. A Definition of Rules in Non-deterministic Information Systems

DEFINITION 3.8. For any NIS, let DEC be decision attributes. A *rule* (*for* DEC) *from* an *object* x is a possible implication $\psi \in IMP(x, CON, DEC)$ satisfying conditions below:

1. There exists a dependency from CON to DEC.
2. ψ belongs to DGC or IGC groups in Table 3. □

Condition 1 follows data dependencies in $DISs$, and information incompleteness in $NISs$ causes condition 2. Even though there may be other definitions of rules in $NISs$, rules in Definition 3.8 are dealt with in this paper.

In Definition 3.8 the most important problem is to find appropriate CON for specified DEC. In order to find appropriate CON, it is necessary to know each degree of dependency in every derived DIS. For $CON = \{A, B, C\}$ and $DEC = \{D\}$ in Example 3.1, it is necessary to apply a procedure 648 times. The number of all derived $DISs$ is equal to the number of products $\prod_{x \in OB, a \in CON \cup DEC} |g(x, a)|$, and it increases in exponential order. Therefore a method, which sequentially applies the same procedure to all derived $DISs$, is not appropriate for handling $NISs$ with a large number of derived $DISs$. For solving such a problem, we propose a method using *pe*-relations in $NISs$. Some other problems are also solved by applications of *pe*-relations.

4. A Method to Extract Rules in Non-deterministic Information Systems

A method to extract rules (for specified decision attributes DEC) in any NIS is proposed, which consists of seven steps.

(Step 1) Make a *data file.*

(Step 2) Execute a program *transall* for translating the data file to *internal expressions* in each attribute.

(Step 3) Execute a program *pe* to obtain all *pe*-relations in each attribute.

(Step 4) Execute a program *merge* to obtain all *pe*-relations for specified decision attributes DEC.

(Step 5) Fix condition attributes CON, and execute a program *merge* to obtain all *pe*-relations for CON.

(Step 6) Execute a program *dependency* to check the dependency from CON to DEC. If there exists a dependency, go to Step 7 else go to Step 5.

(Step 7) Execute a program *extract* to pick up possible implications belonging to DGC or IGC groups. □

In order to clarify this method, a real execution handling Example 3.1 is presented. A workstation with 450MHz UltraSparc is employed for this experiment. The contents of data file *data8.pl* in Example 3.1 are as follows:

```
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]).
```

The following is a real execution in Step 2.

```
?-transall.
File Name:'data8.pl'.
EXEC_TIME = 0.112 (sec)
yes
```

After this translation, four files A.rs, B.rs, C.rs and D.rs are produced. Step 3 is applied to these four files.

```
?-pe.
File Name:'data8.pl'.
EXEC_TIME = 0.182 (sec)
yes
```

After this execution, four files A.pe, B.pe, C.pe and D.pe are produced. Every file stores all *pe*-relations in a format. In Step 4, suppose DEC be $\{C\}$. All *pe*-relations for $\{C\}$ are stored in a file C.pe. In Step 5, all *pe*-relations for $\{A, B, D\}$ are produced by a program *merge*.

```
% merge
Merging A.pe ...
Merging B.pe ...
Merging D.pe ...
EXEC_TIME = 0.020 (sec)
%
```

After this execution, a file ABD.pe is produced. It is known from this file that there exist 108 derived $DISs$ for $\{A, B, D\}$, and there is only a kind of *pe*-relation, i.e., $\{\{1\},\{2\},\{3\},\{4\},\{5\},\{6\},\{7\},\{8\},\{9\},\{10\}\}$. Since any distinct objects $x, y \in OB$ are different each other, it is known every derived DIS (for CON=$\{A, B, D\}$ and DEC=$\{C\}$) is consistent. A program *dependency* is executed in Step 6. Here, val_1 and val_2 in Definition 3.5 are fixed to 0.5 and 0.6, respectively.

```
% dependency
CRITERION 1
   Number of Derived DISs:648
   Number of Derived Consistent DISs:648
   Degree of Consistent DISs:1.000
CRITERION 2
   Minimum Degree of Dependency:1.000
   Maximum Degree of Dependency:1.000
EXEC_TIME = 0.000 (sec)
%
```

This execution shows a preferable result, namely each degree of dependency from $\{A, B, D\}$ to $\{C\}$ is 1. By repeating Step 5 and Step 6, some kinds of dependencies are checked, and CON is fixed to a set $\{A, D\}$. In Step 7, CON=$\{A, D\}$ and DEC=$\{C\}$ are specified in a file *attrib.pl*, and either a program *extract* or *extractall* is executed. Because there exists a dependency from $\{A, D\}$ to $\{C\}$, most possible implications belong to DGC or IGC groups.

```
?-extractall.
Rules from [A,D] to [C]
[1] [2,1] => [0] (0.667= 36/54, 54, DM ) from object 10
[2] [0,1] => [1] (1.000= 18/18, 54, IGC ) from object 4
[3] [1,1] => [1] (1.000= 18/18, 54, IGC ) from object 4
[4] [2,1] => [1] (0.000= 0/18, 54, IGI ) from object 4
     :          :          :
[16] [2,4] => [5] (1.000= 27/27, 54, IGC ) from object 6
[17] [0,4] => [5] (1.000= 54/54, 54, DGC ) from object 9
EXEC_TIME = 0.025 (sec)
yes
```

In this way it is possible to obtain rules in any NIS. Even though there exists work handling logic in $NISs$, there seems no work processing these rules in $NISs$. Of course, there exists a lot of work handling rules in $DISs$.

In the subsequent sections algorithms of programs are examined. Every equivalence relation is effectively applied to extracting rules in any DIS, and every *pe*-relation is analogically applied to extracting rules in any NIS. Every program is realized according to *pe*-relations.

5. Algorithms for Obtaining *pe*-relations

In this section, algorithms in Step 2 and Step 3 are examined. An algorithm for solving the definability of a set in any NIS is first proposed, and an algorithm to obtain all *pe*-relations is also proposed.

5.1. An Algorithm for Solving the Definability of a Set in any NIS

The definability of a set in $DISs$ is extended to the definability of a set in $NISs$.

DEFINITION 5.1. Let us consider a NIS and a set $X \subset OB$. X is *definable* in NIS, if X is definable in some derived $DISs$ from NIS. □

To check this, a method which sequentially examines the definability of a set for every derived DIS may be employed. However, this method will not be suitable for $NISs$ with a large number of derived $DISs$. We propose another method according to Proposition 5.2.

PROPOSITION 5.2. *(Sakai, 2001b) Suppose there exist a NIS and a set $X \subset OB$. X is definable in NIS, if and only if there exist subsets $CL_1, \cdots, CL_m$ of OB, satisfying 1 and 2 below:*

1. $\cup_i CL_i = X$,
2. $\{CL_1, \cdots, CL_m\}$ *is a subset of a pe-relation.* □

According to this proposition, it is possible to check the definability of a set by finding $CL_1, \cdots, CL_m$. An algorithm in the following checks the definability of a set.

ALGORITHM 5.3. (Sakai, 2001b)
Input: A NIS and a set $X \subset OB$.
Output: X is definable in this NIS or not.

(1) $X^*=X$ and $eq=\emptyset$.
(2) For any element $x \in X^*$, find a set CL satisfying conditions (CL-1) and (CL-2).
  (CL-1) $x \in CL$ and $CL \subset X^*$,
  (CL-2) $eq \cup \{CL\}$ is a subset of a *pe*-relation.
(2-1) If there is a set CL, then $[x]=CL$, $eq=eq \cup \{[x]\}$ and $X^*=X^* - CL$. If $X^* \neq \emptyset$ then go to (2). If $X^*=\emptyset$ then X is definable.
(2-2) If there is no CL, then backtrack. If there is no branch for backtracking, then X is not definable in this NIS. □

This algorithm is similar to LEM1 and LEM2 algorithms (Grzymala-Busse and Werbrouck, 1998). The set CL is unique in every DIS, but in NIS this set CL may not be unique. Therefore, a search with backtracking is applied. At this point, Algorithm 5.3 is different from LEM1 and LEM2. In order to realize Algorithm 5.3, two extra properties of *pe*-relations are necessary (Sakai, 2001b).

5.2. Basic Programs candidate and class

A program *class* which checks the definability of a set in NIS is as follows:

```
class(ATR,X):-class0(ATR,X,[],EQ,[],Pres,[],Nres).
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).
```

A program *candidate*, which assigns a *pe*-class to the variable CAN, is realized due to the clarification of two properties for *pe*-relations. A program *translate* in Step 2 produces data for the program *candidate*. The following is a real execution for $ATR=\{A, B\}$, which is specified in a file *attrib.pl*.

```
?-translate.
File Name for Read Open:'data8.pl'.
Attribute Definition File:'attrib.pl'.
EXEC_TIME = 0.025 (sec)
yes
?-classall(con,[2,3,4,5]).
[1] (EQUIVALENCE)RELATION:[[2],[3,4],[5]]
POSITIVE SELECTION
CONDITION OF 2:[0,0] *
CONDITION OF 3:[2,3]
CONDITION OF 4:[2,3] *
CONDITION OF 5:[1,1] *
     :          :          :
[2] (EQUIVALENCE)RELATION:[[2],[3],[4],[5]]
     :          :          :
[3] (EQUIVALENCE)RELATION:[[2,4],[3],[5]]
     :          :          :
[4] (EQUIVALENCE)RELATION:[[2],[3,4],[5]]
POSITIVE SELECTION
CONDITION OF 2:[0,3] *
CONDITION OF 3:[2,3]
CONDITION OF 4:[2,3] *
CONDITION OF 5:[1,1] *
NEGATIVE SELECTION
CONDITION OF 1:[0,3] *
CONDITION OF 4:[0,3] *
EXEC_TIME = 0.016 (sec)
yes
?-class(con,[2,3,4,5,6]).
EXEC_TIME = 0.005 (sec)
yes
```

It is known that there are four kinds of relations which make a set $\{2, 3, 4, 5\}$ definable. Every relation is stored in the variable EQ in a clause *class*. A set $\{2, 3, 4, 5, 6\}$ is not definable in this NIS, because $[6]_{\{A,B\}}=\{6, 10\}$ holds in any *pe*-relation.

5.3. Side Effect of Algorithm 5.3

Algorithm 5.3 has the following merit. After checking the definability of a set, a subset of a *pe*-relation is stored in the variable EQ as a side effect of this algorithm. If $X=OB$, it is possible to obtain a *pe*-relation from this EQ. By applying this algorithm repeatedly until there is no branch for backtracking, all *pe*-relations can be obtained.

6. A Data Structure of *pe*-relations

Usually, a number i is identified with an object located in the i-th row of a table. By the order of row, it is possible to introduce the total order into OB. Suppose there exist N kinds of *pe*-relations for ATR, and let $peq(ATR, k)$ denote the k-th *pe*-relation for ATR. For handling $peq(ATR, k)$, two arrays $head_{ATR}[k][i]$ and $succ_{ATR}[k][i]$ $(1 \leq k \leq N, 1 \leq i \leq |OB|)$ are employed. A set $[i]_{ATR,k} \in peq(ATR, k)$ denotes a *pe*-class including object i. $head_{ATR}[k][j]$ for any $j \in [i]_{ATR,k}$ is the first element of the *pe*-class $[i]_{ATR,k}$, and $succ_{ATR}[k][j]$ is the successor to j in $[i]_{ATR,k}$. For the last element $j \in [i]_{ATR,k}$, $succ_{ATR}[k][j]$=0. In such a data structure, every *pe*-class $[i]_{ATR,k}$ can easily be obtained. For every i, $[i]_{ATR,k}$=$\{head_{ATR}[k][i], succ_{ATR}[k][head[k][i]], \cdots, succ_{ATR}[k][\cdots succ_{ATR}[k][head_{ATR}[k][i]]]\}$ holds.

For ATR=$\{A\}$ in Example 3.1, there exist three *pe*-relations, and $peq(\{A\}, 3)$=$\{\{1, 2, 9\}, \{3, 6, 10\}, \{4, 5, 7, 8\}\}$ is identified with two arrays below:

$head_{\{A\}}[3][1]$=1, $succ_{\{A\}}[3][1]$=2, $head_{\{A\}}[3][2]$=1, $succ_{\{A\}}[3][2]$=9,
$head_{\{A\}}[3][3]$=3, $succ_{\{A\}}[3][3]$=6, $head_{\{A\}}[3][4]$=4, $succ_{\{A\}}[3][4]$=5,
$head_{\{A\}}[3][5]$=4, $succ_{\{A\}}[3][5]$=7, $\cdots$, $head_{\{A\}}[3][10]$=3, $succ_{\{A\}}[3][10]$=0.

For object 2, $head_{\{A\}}[3][2]$=1, $succ_{\{A\}}[3][1]$=2, $succ_{\{A\}}[3][2]$=9 and $succ_{\{A\}}[3][9]$=0 hold, so it is known $[2]_{\{A\},3}$=$\{1, 2, 9\}$.

7. An Algorithm for Checking Inclusion of *pe*-classes

Let us show an important proposition.

PROPOSITION 7.1. *For a NIS and an $ATR \subset AT$, let us consider t-th pe-relation $peq(ATR, t)$, i.e., two arrays $head_{ATR}[t][i]$ and $succ_{ATR}[t][i]$ $(1 \leq i \leq |OB|)$. For any $j \in OB$, $j \in [i]_{ATR,t}$ holds if and only if $head_{ATR}[t][j]$=$head_{ATR}[t][i]$ holds.*

PROOF. If $j \in [i]_{ATR,t}$ holds, clearly $head_{ATR}[t][j]$=$head_{ATR}[t][i]$ holds. If $j \notin [i]_{ATR,t}$ holds, j belongs to other different class $[i']_{ATR,t}$. The first element of $[i]_{ATR,t}$ and the first element of $[i']_{ATR,t}$ are clearly different each other, and $head_{ATR}[t][j] \neq head_{ATR}[t][i]$ is derived. Therefore if $head_{ATR}[t][j]$=$head_{ATR}[t][i]$ holds, $j \in [i]_{ATR,t}$ holds by contraposition. □

By applying Proposition 7.1 to each $j \in [i]_{CON,t}$, it is possible to check $[i]_{CON,t} \subset [i]_{DEC,s}$ or not. This inclusion implies every object in $[i]_{CON,t}$ is consistent with other objects. Algorithm 7.2 solves this inclusion relation.

ALGORITHM 7.2. (Algorithm for checking the inclusion of *pe*-classes)
Input: Any object i, a *pe*-class $[i]_{CON,t}$ for CON and a *pe*-class $[i]_{DEC,s}$ for DEC.
Output: $[i]_{CON,t} \subset [i]_{DEC,s}$ or not.

```
begin
    mark:=0; point:=head_CON[t][i];
    while point≠0 do
       if head_DEC[s][point]=head_DEC[s][i] then point:=succ_CON[t][point]
          else begin mark:=1; point:=0 end;
    if mark=0 then [i]_CON,t ⊂[i]_DEC,s else [i]_CON,t ⊄[i]_DEC,s
end.
```

□

Algorithm 7.2 sequentially picks up $point \in [i]_{CON,t}$, and applies Proposition 7.1 to checking $point \in [i]_{DEC,s}$. The complexity of the worst case, which is a case such

that $[i]_{CON,t} \subset [i]_{DEC,s}$, is $o(|[i]_{CON,t}|)$. Algorithm 7.2 is a basic algorithm for realizing a program *dependency* in Step 6.

8. An Algorithm for Merging *pe*-relations

A program *merge* in Step 5 is examined in this section.

PROPOSITION 8.1. *(Pawlak, 1991) Suppose $A, B \subset AT$ hold in a DIS. An equivalence relation $eq(A \cup B)$ is $\{M \subset OB | M = [i]_A \cap [i]_B$ for $[i]_A \in eq(A)$ and $[i]_B \in eq(B)(1 \leq i \leq |OB|)\}$.* □

Proposition 8.1 shows us a way to merge two equivalence relations. Algorithm 8.2 produces $peq(A \cup B, \_)$ from $peq(A, t)$ and $peq(B, s)$.

ALGORITHM 8.2. (Sakai, 2001b)
Input: $peq(A, t)$ for $A \subset AT$ and $peq(B, s)$ for $B \subset AT$ respectively,
i.e., $head_A[t][i]$, $succ_A[t][i]$, $head_B[s][i]$ and $succ_B[s][i]$ $(1 \leq i \leq |OB|)$.
Output: $peq(A \cup B, \_)$ for $A \cup B \subset AT$,
i.e., $head_{A \cup B}[\_][i]$ and $succ_{A \cup B}[\_][i]$ $(1 \leq i \leq |OB|)$.

```
begin
    for i:=1 to |OB| do begin head_{AUB}[_][i]:=i; succ_{AUB}[_][i]:=0 end;
    for i:=1 to |OB| do if head_{AUB}[_][i]=i then
       begin
       pre:=i; point:=succ_A[t][i];
       while point≠ 0 do
          begin if head_B[s][point]=head_B[s][i] then
             begin succ_{AUB}[_][pre]:=point;
                   head_{AUB}[_][point]:=i; pre:=point
             end;
          point:=succ_A[t][point]
          end
       end
end.
```
□

In Algorithm 8.2, if $peq(A, t)$=$\{\{1, 2, \cdots, |OB|\}\}$ and $peq(B, s) = \{\{1\}, \{2\}, \cdots, \{|OB|\}\}$, it is necessary to check $head_B[s][point]$=$head_B[s][i]$ for every $point \in [i]_{A,t}$= $\{1, 2, \cdots, |OB|\}$ and every $i \in OB$. Thus, computational complexity in the worst case is $o(|OB|^2)$. On the contrary, if $peq(A, t) \subset peq(B, s)$ holds, $[i]_{A,t} \subset [i]_{B,s}$ holds for any $i \in OB$. Therefore $head_B[s][point]$=$head_B[s][i]$ always holds for any $point \in [i]_{A,t}$, and $head_B[s][point]$=$head_B[s][i]$ is checked only once for any $point \in [i]_{A,t}$. Thus computational complexity in the best case is $o(|OB|)$.

Algorithm 8.2 is more effective for merging several kinds of attributes. Suppose there exists a sequence of attributes $A_1, A_2, \cdots, A_m \subset AT$, and it is necessary to obtain $eq(A_1 \cup \cdots \cup A_m)$. In this case, Algorithm 8.2 is sequentially applied to $eq(A_1 \cup \cdots \cup A_k)$ and $eq(A_{k+1})$ $(1 \leq k \leq m-1)$. Even though the order may be $o(|OB|^2)$ in the first application of Algorithm 8.2, the order is sequentially near to $o(|OB|)$. Because, every object is sequentially discerned by merging attributes.

9. An Algorithm for Calculating the Dependencies

A program *dependency* in Step 6 is examined in this section. Algorithm 9.1 calculates the degree of dependency by using *pe*-relations.

ALGORITHM 9.1. (Algorithm for calculating the degree of dependency)
Input: $peq(CON, t)$ for attributes CON and $peq(DEC, s)$ for attributes DEC.
Output: The degree of dependency based on $peq(CON, t)$ and $peq(DEC, s)$.

```
begin
    count:=0;
    for i:=1 to |OB| do if head_CON[t][i]=i then
       if [i]_CON,t ⊂[i]_DEC,s then count:=count+|[i]_CON,t|;
    degree:=count/|OB|
end.
```

□

In Algorithm 9.1, inclusion $[i]_{CON,t} \subset [i]_{DEC,s}$ is checked for any object i satisfying $head_{CON}[t][i] = i$. Since the order of checking this inclusion is less than $o(|[i]_{CON,t}|)$, the order of Algorithm 9.1 is less than $o(\Sigma_{\{i|head_{CON}[t][i]=i\}}|[i]_{CON,t}|)$, which is equal to $o(|OB|)$. Because $[i]_{CON,t}$ is an equivalence class, i.e., $\cup_{i \in OB}[i]_{CON,t} = OB$ and $[i]_{CON,t} \cap [j]_{CON,t} = \emptyset$ for any two distinct classes hold.

Suppose there exist N_{CON} kinds of *pe*-relations for CON and N_{DEC} kinds of *pe*-relations for DEC. By applying Algorithm 9.1 to these pairs of *pe*-relations $N_{CON} \times N_{DEC}$ times, it is possible to obtain each degree of dependency in all derived $DISs$. As stated before, the number of pairs of *pe*-relations are much smaller than the number of all derived $DISs$. If $N_{CON}=1$ and $N_{DEC}=1$, it is possible to check the dependency by only one time application of Algorithm 9.1.

10. Execution Time for Checking Dependencies

In order to examine the execution time, four $NISs$ in Table 4 are prepared, and each degree of dependency from $CON=\{A, B, C\}$ to $DEC=\{D\}$ is calculated. Four $NISs$ are produced by using a random number program.

NIS	$\|OB\|$	$\|AT\|$	$\|VAL_a\|(a \in AT)$	$Derived\_DISs$
NIS_1	10	4	10	864
NIS_2	100	4	10	1944
NIS_3	300	4	10	3888
NIS_4	1000	4	10	7776

Table 4: Definitions of $NISs$

Table 5 shows each execution time. The column of simple method shows expected values for obtaining each degree of dependency in all derived $DISs$. An execution time for calculating the degree of dependency is measured in a derived DIS, and expected values

NIS	$Step\_2$	$Step\_3$	$Step\_5$	$Step\_6$	$Total$	$simple\_method$
NIS_1	0.087	0.241	0.000	0.000	0.328	$0.000(= 0.000 * 864)$
NIS_2	0.752	2.746	0.080	0.000	3.578	$0.000(= 0.000 * 1944)$
NIS_3	3.799	6.840	0.420	0.000	11.059	$116.640(= 0.030 * 3888)$
NIS_4	32.548	50.001	0.090	0.080	82.719	$2643.840(= 0.340 * 7776)$

Table 5: Results of the execution time(sec) for dependency $[A, B, C] \Rightarrow [D]$

NIS	$[A,D] \Rightarrow [B,C]$	$[A,C] \Rightarrow [B,D]$	$[D] \Rightarrow [A,B,C]$
NIS_1	0.000	0.000	0.000
NIS_2	0.600	0.040	0.070
NIS_3	0.380	0.440	0.420
NIS_4	0.520	0.480	0.150

Table 6: Results of the execution time in Step 5 + Step 6(sec) for every dependency

are defined by this execution time multiplied the number of derived $DISs$. According to Table 5, Step 5 and Step 6 take less execution time. Step 3, which obtains all *pe*-relations for each attribute, is the most time-consuming.

Table 6 shows each execution time in Step 5 and Step 6 for three cases. In order to find appropriate CON for specified DEC, it may be necessary to calculate some kinds of dependencies. Suppose N kinds of dependencies are calculated in a NIS. Total execution time is [execution time of (Step 1 + Step 2 + Step 3 + Step 4) + $N\times${execution time of (Step 5 + Step 6)}]. Because, Step 5 and Step 6 take less execution time, it is convenient for finding dependencies in a NIS. If simple method is employed in this case, total execution time is $N\times$(the number of derived $DISs$)$\times$(execution time of checking a dependency in a DIS). In this way simple method is not suitable for finding appropriate attributes CON. Proposed method is more effective than simple method.

11. Program extractall in Step 7

A program *extractall* in Step 7 is examined in this section. This program is now implemented by prolog, and this program sequentially compares possible implications $\psi_x \in IMP(x, CON, DEC)$ and $\psi_y \in IMP(y, CON, DEC)$. The computational order is $o((\sum_{1\leq i\leq |OB|} |IMP(i, CON, DEC)|)^2)$. This program took about 5(sec) in NIS_2, about 100(sec) in NIS_3 and over 1000(sec) in NIS_4 in Table 6. In this program, it is also necessary to apply *pe*-relations, however there is a problem. In Example 3.1 suppose CON={A, B} and DEC={D} hold, and let us consider all *pe*-relations for CON and DEC. By using these *pe*-relations, it is easy to obtain the following response.

```
% ratio
File Name for Condition:AB.pe
File Name for Decision:D.pe
Consistent ratio of object 1:0.722(= 78 / 108)
Consistent ratio of object 2:0.444(= 48 / 108)
Consistent ratio of object 3:0.667(= 72 / 108)
     :          :          :
Consistent ratio of object 9:0.667(= 72 / 108)
Consistent ratio of object 10:0.000(= 0 / 108)
EXEC_TIME = 0.040(sec)
%
```

According to this response it is known object 1 is consistent with other objects in 78 derived $DISs$. This information is related to every object, and is not related to each possible implication. According to data structure of *head*[*][*] and *succ*[*][*], it is possible to reach all *pe*-relations from any object. However, there is no relation between

pe-relations and possible implications, and it is impossible to reach *pe*-relations from any possible implication. A new data structure connecting *pe*-relations and possible implications is necessary for improving the program *extractall*. It is also important to extend three criteria $support(\psi)$, $accuracy(\psi)$ and $coverage(\psi)$ in $DISs$ to new criteria in $NISs$. An application of such criteria to rule extraction is under consideration.

12. Concluding Remarks

The rough sets based concept in $DISs$ is extended to new concept in $NISs$ for handling incomplete information, and a framework of rule extraction from non-deterministic information systems is proposed. Some algorithms using possible equivalence relations are also proposed for realizing programs. These programs may be useful tools for data analysis in non-deterministic information systems.

Up to now, there is only few work that deals with real data analysis in $NISs$. For handling not only certain information but also incomplete information, the research of rough sets based data analysis in $NISs$ will be important. Furthermore, rough sets based data analysis in $NISs$ could also be a mathematical foundation of knowledge discovery and data mining from incomplete information. Our work is toward real data analysis in $NISs$, and will extend the application area of rough sets theory.

References

Codd, E. (1970). A relational model of data for large shared data banks, *Communication of the ACM*, **13**, 377–387.

Grzymala-Busse, J. (1991). On the unknown attribute values in learning from examples, *Lecture Notes in AI*, Springer-Verlag, **542**, 368–377.

Grzymala-Busse, J. (1997). A new version of the rule induction system LERS, *Fundamenta Informaticae*, IOS Press, **31**, 27–39.

Grzymala-Busse, J. and Werbrouck, P. (1998). On the best search method in the LEM1 and LEM2 algorithms, *Incomplete Information: Rough Set Analysis, Studies in Fuzziness and Soft Computing*, Physica-Verlag, **13**, 75–91.

Kryszkiewicz, M. (1998). Rough set approach to incomplete information systems, *Information Sciences*, **112**, 39–49.

Kryszkiewicz, M. (1999). Rules in incomplete information systems, *Information Sciences*, **113**, 271–292.

Lipski, W. (1981). On databases with incomplete information, *Journal of the ACM*, **28**, 41–70.

Nakamura, A., Tsumoto, S., Tanaka, H. and Kobayashi, S. (1996). Rough set theory and its applications, *Journal of Japanese Society for AI*, **11**, 209–215.

Orlowska, E. and Pawlak, Z. (1984). Representation of nondeterministic information, *Theoretical Computer Science*, **29**, 27–39.

Orlowska, E. (1998). What you always wanted to know about rough sets, *Incomplete Information: Rough Set Analysis, Studies in Fuzziness and Soft Computing*, Physica-Verlag, **13**, 1–20.

Pawlak, Z. (1982). Rough sets, *Int'l. Journal of Information and Computer Sciences*, **11**, 341–356.

Pawlak, Z. (1991). *Rough Sets*, Kluwer Academic Publisher.

Pawlak, Z. (1996). Data versus logic a rough set view, *Proc. 4th Int'l. Workshop on Rough Set, Fuzzy Sets and Machine Discovery*, 1–8.

Polkowski, L. and Skowron, A.(Eds.) (1998a). *Rough Sets in Knowledge Discovery 1, Studies in Fuzziness and Soft Computing*, Physica-Verlag, **18**.

Polkowski, L. and Skowron, A.(Eds.) (1998b). *Rough Sets in Knowledge Discovery 2, Studies in Fuzziness and Soft Computing*, Physica-Verlag, **19**.

Ras, Z. and Joshi, S. (1997). Query approximate answering system for an incomplete DKBS, *Fundamenta Informaticae*, IOS Press, **30**, 313–324.

Sakai, H. (1998). Some issues on nondeterministic knowledge bases with incomplete and selective information, *Lecture Notes in AI*, Springer-Verlag, **1424**, 424–431.

Sakai, H. and Okuma, A. (1999). An algorithm for finding equivalence relations from tables with non-deterministic information, *Lecture Notes in AI*, Springer-Verlag, **1711**, 64–72.

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, *Lecture Notes in AI*, Springer-Verlag, **1886**, 219–229.

Sakai, H. (2001a). Two procedures for dependencies among attributes in a table with non-deterministic information: a summary, *Lecture Notes in AI*, Springer-Verlag, **2253**, 301–305.

Sakai, H. (2001b). Effective procedures for handling possible equivalence relations in non-deterministic information systems, *Fundamenta Informaticae*, IOS Press, **48**, 343–362.

Tsumoto, S. (2000). Knowledge discovery in clinical databases and evaluation of discovered knowledge in outpatient clinic, *Information Sciences*, **124**, 125–137.

Zhong, N., Dong, J., Fujitsu, S. and Ohsuga, S. (1998). Soft techniques to rule discovery in data, *Transactions of Information Processing Society of Japan*, **39**, 2581–2592.

Received March 16, 2001
Revised April 10, 2002