# An Advancement of a Rough Sets based Rule Generator - Minimal Certain Rules and Discernibility Functions -

Hiroshi Sakai
Department of Mathematics and Computer Aided Sciences
Faculty of Engineering, Kyushu Institute of Technology
Tobata, Kitakyushu 804, Japan
e-mail: sakai@mns.kyutech.ac.jp

*Abstract*— **A rough sets based method to obtain minimal certain rules in information systems is presented. This method employs discernibility functions, and minimal certain rules are obtained as solutions of a discernibility function. Some manipulations on discernibility functions are proposed, and these manipulations are implemented for not only deterministic information systems but also non-deterministic information systems. An execution of realized programs is also shown.**

## I. Introduction

Rough set theory is seen as a mathematical foundation of soft computing. This theory usually handles tables with deterministic information, which we call *Deterministic Information Systems* (*DISs*). Many applications of this theory to rule generation, machine learning and knowledge discovery have been presented [1,2,3,4].

*Non-deterministic Information Systems* (*NISs*) and *Incomplete Information Systems* have been proposed for handling information incompleteness in *DISs*, like null values, unknown values, missing values and etc. In this extension, the concept of modality is introduced into *NISs*, and an axiomatization of logic in *NISs* has been studied [5,6,7,8]. Most of work related to *NISs* is research of logic with modal operators [ ] and <>, and focuses on valid formulas.

Very few work deals with algorithms for handling *NISs* on computers. In [7,8], Lipski showed a question-answering system besides an axiomatization of logic. Grzymala-Busse surveyed the unknown attribute values, and studied the learning from examples with unknown attribute values [9,10]. In [11,12], Kryszkiewicz discussed rules in incomplete information systems. These are the most important work for handling incomplete information on computers.

According to such previous work, we have already proposed rule generation in *NISs* [13,14], and in this paper we focus on minimal rule generation in *NISs*.

## II. Preliminary

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*. Such a relation that $f(x, A)$=$f(y, A)$ for every $A \in ATR \subset AT$ is an equivalence relation over $OB$. Let $[x]_{ATR}$ denote an equivalence class $\{y \in OB | f(y, A)$=$f(x, A)$ for every $A \in ATR\}$.

Let us consider two sets $CON \subset AT$ which we call *condition attributes* and $DEC \subset AT$ which we call *decision attributes*. An object $x \in OB$ is *consistent* with an object $y$, if $f(x, A)$=$f(y, A)$ for every $A \in CON$ implies $f(x, A)$=$f(y, A)$ for every $A \in DEC$. An object $x \in OB$ is *consistent*, if $x$ is consistent with any $y \in OB$.

For any $x \in OB$, let $imp(x, CON, DEC)$ denote a formula called an *implication*:

$\wedge_{A \in CON}[A, f(x, A)] \Rightarrow \wedge_{A \in DEC}[A, f(x, A)]$,

where a formula $[A, f(x, A)]$ implies that $f(x, A)$ is the value of the attribute $A$. This is called a *descriptor* in [7,12].

## III. Basic Definitions on NISs

A *Non-deterministic Information System* (*NIS*) is also a quadruplet $(OB, AT, \{VAL_A | A \in AT\}, g)$, where $g : OB \times AT \rightarrow P(\cup_{A \in AT} VAL_A)$ (a power set of $\cup_{A \in AT} VAL_A$) [5,7]. Every set $g(x, A)$ is interpreted as that there is an actual value in this set but this value is not known. Especially if the actual value is not known at all, $g(x, A)$ is equal to $VAL_A$. Codd has already dealt with such information incompleteness in relational databases, and relational algebra handling this information incompleteness was proposed. This interpretation to information incompleteness was named *null value* interpretation [15]. In [16], new relational algebra handling non-deterministic information has also been proposed.

Let us consider a $NIS$=$(OB, AT, \{VAL_A | A \in AT\}, g)$, a set $ATR \subset AT$ and a mapping $h : OB \times ATR \rightarrow \cup_{A \in ATR} VAL_A$ such that $h(x, A) \in g(x, A)$. We call a $DIS$=$(OB, ATR, \{VAL_A | A \in ATR\}, h)$ a *derived DIS* (*for ATR*) *from NIS*.

Let us consider a $NIS$ and a set $ATR$=$\{A_1, \cdots, A_n\} \subset AT$. For any $x \in OB$, let $PT(x, ATR)$ denote the Cartesian product $g(x, A_1) \times \cdots \times g(x, A_n)$. We call every element a *possible tuple* (*for ATR*) *of x*. For a possible tuple $\zeta$=$(\zeta_1, \cdots, \zeta_n) \in PT(x, ATR)$, let $[ATR, \zeta]$ denote a formula $\bigwedge_{1 \leq i \leq n}[A_i, \zeta_i]$.

For a $NIS$, let $CON$ be condition attributes and let $DEC$ be decision attributes. For any $x \in OB$, let $PI(x, CON, DEC)$ denote a set $\{[CON, \zeta] \Rightarrow [DEC, \eta] | \zeta \in PT(x, CON), \eta \in PT(x, DEC)\}$. We call an element of

| $OB$ | $A$ | $B$ | $C$ | $D$ | $E$ | $F$ | $G$ | $H$ |
|---|---|---|---|---|---|---|---|---|
| 1 | 3 | 1,3,4 | 3 | 2 | 5 | 5 | 2,4 | 3 |
| 2 | 2 | 3,4 | 1,3,4 | 4 | 1,2 | 2,4,5 | 2 | 2 |
| 3 | 4,5 | 5 | 1,5 | 5 | 2 | 5 | 1,2,5 | 1 |
| 4 | 1 | 3 | 4 | 3 | 1,2,3 | 1 | 2,5 | 1,2 |
| 5 | 4 | 1 | 2,3,5 | 5 | 2,3,4 | 1,5 | 4 | 1 |
| 6 | 4 | 1 | 5 | 1 | 4 | 2,4,5 | 2 | 1,2,3 |
| 7 | 2 | 4 | 3 | 4 | 3 | 2,4,5 | 4 | 1,2,3 |
| 8 | 4 | 5 | 4 | 2,3,5 | 5 | 3 | 1,2,3 | 1,2,3 |
| 9 | 2 | 3 | 5 | 3 | 1,3,5 | 4 | 2 | 3 |
| 10 | 4 | 2 | 1 | 5 | 2 | 4,5 | 3 | 1 |

TABLE I
A Table of $NIS_1$

$PI(x, CON, DEC)$ a *possible implication (for CON and DEC) of x*. If $PI(x, CON, DEC)$ is a singleton set $\{\tau\}$, we say $\tau$ (from $x$) is *definite*. Otherwise we say $\tau$ (from $x$) is *indefinite*.

For a possible implication $\tau \in PI(x, CON, DEC)$, let $DD(\tau, x, CON, DEC)$ denote a set $\{\varphi |\ \varphi$ is such a derived $DIS$ for $CON \cup DEC$ that an implication $imp(x, CON, DEC)$ in $\varphi$ is equal to $\tau\}$.

If a set $\{\varphi \in DD(\tau, x, CON, DEC) |\ x$ is consistent in $\varphi\}$ is equal to $DD(\tau, x, CON, DEC)$, we say $\tau$ is *globally consistent* ($GC$). If this set is equal to $\emptyset$, we say $\tau$ is *globally inconsistent* ($GI$). Otherwise we say $\tau$ is *marginal* ($MA$). By combining two cases, i.e., 'D(efinite) or I(ndefinite)' and '$GC$ or $MA$ or $GI$', we define six classes, *D-GC*, *D-MA*, *D-GI*, *I-GC*, *I-MA*, *I-GI*, for possible implications.

**Example 1.** Let us consider $NIS_1$ in Table I, which is an artificial data and symbols { and } are omitted. There are 7346640384 derived $DISs$ for all attributes. As for $CON=\{A, B\}$ and $DEC=\{C\}$, there are $216 (=2^3 \times 3^3)$ derived $DISs$. Here, $PT(1, \{A, B\})=\{(3, 1), (3, 3), (3, 4)\}$, $PT(1, \{C\})=\{(3)\}$ and $PI(1, \{A, B\}, \{C\})$ consists of three possible implications $[A, 3] \wedge [B, 1] \Rightarrow [C, 3]$, $[A, 3] \wedge [B, 3] \Rightarrow [C, 3]$ and $[A, 3] \wedge [B, 4] \Rightarrow [C, 3]$. Since there exists no possible tuple $(3, \_) \in PT(x, \{A, B\})$ $(x \neq 1)$, each possible implication $\tau$ is consistent in $DD(\tau, 1, \{A, B\}, \{C\})$. Therefore, every possible implication belongs to *I-GC* class. A possible implication $\tau : [A, 4] \wedge [B, 2] \Rightarrow [C, 1]$ from object 10 belongs to *D-GC* class.

A possible implication $\tau$ belonging to *D-GC* class is consistent in all derived $DISs$, and this $\tau$ is not influenced by the information incompleteness, therefore we especially say $\tau$ is a *certain rule*. A possible implication $\tau$' belonging to either *I-GC*,*D-MA* or *I-MA* class is consistent in some derived $DISs$, therefore we also say $\tau$' is a *possible rule*.

## IV. Rough Sets based Properties in NISs

This section presents necessary and sufficient conditions for characterizing $GC$, $MA$ and $GI$ classes [14].

**Definition 1.** Let us consider a $NIS$ and a set $ATR \subset AT$. For any $\zeta \in PT(x, ATR)$, we fix the tuple of $x$ to $\zeta$, and define (1) and (2) below.
(1) $inf(x, \zeta, ATR)=\{y \in OB | PT(y, ATR)=\{\zeta\}\}$,
(2) $sup(x, \zeta, ATR)=\{y \in OB | \zeta \in PT(y, ATR)\}$.

In Definition 1, $inf(x, \zeta, ATR)$ implies a set of objects whose tuples are $\zeta$ and definite. A set $sup(x, \zeta, ATR)$ implies a set of objects whose tuples may be $\zeta$. In $DISs$, $[x]_{ATR}=inf(x, \zeta, ATR)=sup(x, \zeta, ATR)$ holds, and $\{x\} \subset inf(x, \zeta, ATR) \subset sup(x, \zeta, ATR)$ holds in $NISs$. Although $sup$ specifies the same set defined by the similarity relation $SIM$ [5,12], $inf$ is newly introduced in $NISs$.

**Theorem 1.[14]** For any $NIS$, let $CON$ be condition attributes, let $DEC$ be decision attributes and let us consider a possible implication $\tau$:$[CON, \zeta] \Rightarrow [DEC, \eta] \in PI(x, CON, DEC)$. Then, the following holds.
(1) $\tau$ belongs to $GC$ class if and only if
$sup(x, \zeta, CON) \subset inf(x, \eta, DEC)$.
(2) $\tau$ belongs to $MA$ class if and only if
$inf(x, \zeta, CON) \subset sup(x, \eta, DEC)$.
(3) $\tau$ belongs to $GI$ class if and only if
$inf(x, \zeta, CON) \not\subset sup(x, \eta, DEC)$.

Here, we show a useful property for calculating $inf$ and $sup$. This property is effective for examining conditions in Theorem 1.

**Proposition 2.** For any $NIS$, let $ATR \subset AT$ be $\{A_1, \cdots, A_n\}$, and let a possible tuple $\zeta \in PT(x, ATR)$ be $(\zeta_1, \cdots, \zeta_n)$. Then, the following holds.
(1) $inf(x, \zeta, ATR)=\cap_i inf(x, (\zeta_i), \{A_i\})$.
(2) $sup(x, \zeta, ATR)=\cap_i sup(x, (\zeta_i), \{A_i\})$.

By means of Theorem 1 and Proposition 2, it is possible to decide a class of each possible implication. Namely, we first prepare $inf(x, (\zeta_{i,j}), \{A_i\})$ and $sup(x, (\zeta_{i,j}), \{A_i\})$ for any $x \in OB$, any $A_i \in AT$ and any $(\zeta_{i,j}) \in PT(x, \{A_i\})$. Then, we produce $inf(x, \zeta, CON)$, $sup(x, \zeta, CON)$, $inf(x, \eta, DEC)$ and $sup(x, \eta, DEC)$ according to Proposition 2. Finally, we apply Theorem 1 to them. Especially, if $\cap_i sup(x, (\zeta_i), \{A_i\}) \subset inf(x, \eta, DEC)$ holds for a set of definite descriptors $[A_i, \zeta_i]$ in $x$, it is possible to conclude that a certain rule can be generated from object $x$. In the subsequent sections, a minimal certain rule is defined in every object satisfying this condition. It is impossible to generate any certain rules from object $x$, which does not satisfy the above condition.

## V. Minimal Certain Rules in NISs

Now, let us consider minimal certain rules in $NISs$. According to the usual definition in $DISs$ [1,2,10], we give the definition of minimal certain rules in $NISs$.

**Definition 2.** Let us consider a possible implication $\tau : [CON, \zeta] \Rightarrow [DEC, \eta]$, which belongs to *D-GC* class. We say $\tau$ is a *minimal certain rule*, if there is no proper non-empty subset $CON^* \subset CON$ such that $[CON^*, \zeta^*] \Rightarrow [DEC, \eta]$ ($\zeta^*$ is a tuple restricted to $CON^*$) belongs to *D-GC* class.

Now it is possible to define the problem in this paper.

**Problem.** For any $NIS$, let $DEC$ be decision attributes and let $\eta$ be a tuple of decision attributes values for $DEC$. Then, find all minimal certain rules in the form of $[CON, \zeta] \Rightarrow [DEC, \eta]$.

This problem has already been investigated in $DISs$. An important problem for finding minimal rules in $DISs$ lies in such a fact that the minimal rule may not be unique. Some minimal rules may be generated from an object. In [2,17], a *discernibility function* in *DISs* has been proposed, and some algorithms including reduction of attributes were investigated by means of discernibility functions. To find a minimal reduct in a *DIS* is proved to be NP-hard, too. A discernibility function in *DISs* is extended to a discernibility function in incomplete information systems [12].

In [18,19], the problem of finding minimal reducts is discussed. For a disjunctive normal form $DNF{=}(a \vee b) \wedge c \wedge (b \vee c)$, $c$ must be true for assigning true to $DNF$. Then, true is automatically assigned to $(b \vee c)$, and $DNF$ is reduced to $(a \vee b)$. Such procedure is called the *absorption law* in [19]. For a set $ATT$ of attributes which appear in a discernibility function, a power set $p(ATT)$ of $ATT$ is defined. All minimal reducts are elements in $p(ATT)$. The lower and upper bounds of minimal reducts are defined by a set of *core* attributes and $ATT$, respectively. All minimal reducts exist between these two bounds. According to these properties, a depth-first search algorithm to find minimal reducts in *DISs* is proposed [19]. However, the implementation of these results on computers are not clear. There exists no statement of an implementation.

In such a situation, we investigate a method to generate minimal certain rules in *NISs*, and implement some programs on computers. We define a discernibility function in *NISs*, and propose some manipulations on a discernibility function. In this case also, we apply the absorption law to reducing the discernibility function. Moreover, we employ an interactive method. Generally, there may exist lots of minimal certain rules in *NISs*. For obtaining more suitable rules, we interactively select descriptors in a discernibility function. The generated minimal certain rules are based on the selection of descriptors. Thus, the interactive method may be better than another method, which simply generates all minimal certain rules.

## VI. Minimal Certain Rules and a Discernibility Function in NISs

Let us suppose a certain rule can be generated from object $x$, whose conclusion is $[DEC, \eta]$. According to Theorem 1, the problem is to find such a minimal conjunction $[CON, \zeta]$ that $sup(x, \zeta, CON) \subset inf(x, \eta, DEC)$. Here, $sup(x, \zeta, CON){=}\cap_i sup(x, (\zeta_i), \{A_i\})$ $(A_i \in CON)$ holds. Therefore, a minimal set of definite descriptors, which discriminate every object in $OB\text{-}inf(x, \eta, DEC)$ from $inf(x, \eta, DEC)$, becomes a minimal conjunction $[CON, \zeta]$. According to this property, we give the following definitions.

**Definition 3.** Let us consider an object $x \in OB$. Any distinct $y \in OB$ is *discriminated from x* by a definite descriptor $[A, \zeta]$ $(A \in AT\text{-}DEC)$ in $x$, if $y \notin sup(x, (\zeta), \{A\})$. Let $DISC(x, y)$ denote a disjunction of such definite descriptors in $x$. If $DISC(x, y)$ is a disjunction consisting of a descriptor [A,$\zeta$], we say $[A, \zeta]$ is a *core* descriptor.

**Definition 4.** For any $NIS$, let us suppose that it is possible to generate a certain rule from object $x$. A *discernibility function* $DF(x)$ (in *D-GC* class) is a formula, $DF(x){=}\wedge_{y \in OB-inf(x,\eta,DEC)} DISC(x, y)$.

**Definition 5.** For a discernibility function $DF(x)$, let us identify every descriptor in $DF(x)$ with a propositional variable. If a set $SOL$ of descriptors assigns true to $DF(x)$, we say *SOL satisfies DF(x)*.

**Example 2.** In Table I, let us consider possible implications $[CON, \zeta]$ ⇒[H,1]. Since a definite descriptor [H,1] appears in objects 3, 5 and 10, it may possible to generate certain rules from these three objects. A discernibility function $DF(3)$ is

$DF(3)=$

$([B,5] \vee [D,5] \vee [E,2]) \wedge ([B,5] \vee [D,5]) \wedge ([B,5] \vee [D,5] \vee [F,5])$
$\wedge ([B,5] \vee [D,5] \vee [E,2]) \wedge ([B,5] \vee [D,5] \vee [E,2]) \wedge ([E,2] \vee [F,5])$
$\wedge ([B,5] \vee [D,5] \vee [E,2] \vee [F,5])$.

Since $DISC(3, y) \neq \emptyset$ holds for every $y \in OB - inf(3, (1), \{H\}){=}OB - \{3, 5, 10\}{=}\{1, 2, 4, 6, 7, 8, 9\}$, it is possible to generate a certain rule from object 3. Here, the first disjunction ([B,5]∨[D,5]∨[E,2]) implies that descriptors [B,5], [D,5] and [E,2] discriminate object 1 from object 3. A set {[B,5],[E,2]} satisfies this function, and neither {[B,5]} nor {[E,2]} satisfy this function. Thus, we obtain a minimal certain rule [B,5]∧[E,2]⇒[H,1].

## VII. Manipulations on Discernibility Functions

This section proposes some manipulations on discernibility functions.

At first, a simple method to obtain all minimal certain rules is proposed in Algorithm 1, which we name *enumeration method.* In this method, we enumerate every subset of all descriptors in $DF(x)$, then we sequentially examine the satisfiability of $DF(x)$.

**Algorithm 1. (Enumeration method)**

```
Input: Such an object $x$ that a certain rule [CON,$\zeta$]⇒
    [DEC,$\eta$] is generated from $x$.
Output: All minimal certain rules from $x$.
begin
    generate DF(x);
    enumerate every subset SUB of all descriptors in
    DF(x) according to an order of the set inclusion;
    repeat the following;
      if SUB satisfies DF(x), SUB is a minimal subset,
      namely $\wedge_{DESC \in SUB} DESC$ ⇒[DEC,$\eta$] is a minimal
      certain rule, and remove every subset SUB1
      satisfying SUB⊂SUB1;
end.
```

This enumeration method can generate all minimal certain rules from object $x$. However, there exist $2^{|ALL\_DESC|}$ kinds of subsets for a set of all descriptors $ALL\_DESC$ in $DF(x)$. Therefore, this method works well just for small size *NISs*.

Now, let us consider another method to obtain a minimal subset of descriptors, which satisfy $DF(x)$. Namely, we sequentially select a descriptor in $DF(x)$, and we reduce $DF(x)$ to new $DF$'$(x)$. By repeating this procedure, it is possible to obtain a set of descriptors satisfying $DF(x)$.

Let us consider the discernibility function $DF(3)$ in Table I, again. If we select a descriptor [B,5] in the first disjunction, it is possible to remove all disjunctions with [B,5] from $DF(3)$. Therefore, we have only to solve $DF$'(3)=[E,2]∨[F,5]. This is the application of the absorption law, and we name new discernibility function a *revised* discernibility function. This revision will be effective for reducing descriptors in $DF(x)$.

According to such consideration, we have proposed *interactive selection method* for obtaining minimal certain rules [20]. However, we have to pay attention to this interactive selection method. Because some sets of selected descriptors, which satisfy $DF(x)$, may not be minimal. Let us consider the following example.

**Example 3.** Let $a$,$b$,$c$,$d$ be descriptors in a $NIS$, and let us suppose $DF$=$(a \vee b) \wedge (b \vee c) \wedge (c \vee d)$ be a discernibility function for a class. If we select descriptor $a$ in $DF$, $DF$ is revised to $DF$'=$(b \vee c) \wedge (c \vee d)$. Similarly if we select descriptor $b$ in $DF$', $DF$' is revised to $DF$"=$(c \vee d)$. Finally, we select descriptor $c$ and we obtain a set $\{a, b, c\}$. This set satisfies $DF$, but this set is not minimal. Because, both sets $\{a, c\}$ and $\{b, c\}$ satisfy $DF$.

For solving the problem in Example 3, we combine interactive selection method with enumeration method, and propose Algorithm 2.

**Algorithm 2. (Interactive selection and enumeration method)**

```
Input: Such an object $x$ that a certain rule [CON,$\zeta$]⇒
    [DEC,$\eta$] is generated from $x$.
Output: Interactively selected minimal certain rules from
    $x$.
begin
    generate DF(x); CORE={}; SOL={};
    for (every descriptor DESC in DF(x))
      if ([A$_i$,$\zeta_i$] is a core) CORE=CORE∪{[A$_i$,$\zeta_i$]} and
      remove each disjunction with [A$_i$,$\zeta_i$] from DF(x);
    suppose DF'(x) be the revised discernibility function;
    while (DF'(x)$\neq \emptyset$) do
      select a descriptor DESC in DF'(x), SOL=SOL∪
      {DESC} and assign the revised discernibility
      function to DF'(x);
    apply Algorithm 1 for descriptors CORE∪SOL, and
    generate all minimal certain rules based on SOL;
end.
```

In Algorithm 2, we sequentially fix a branch of tree search, and a set of descriptors satisfying $DF(x)$ is also decided. Then, Algorithm 1 is applied, so Algorithm 2 generates only minimal certain rules depending upon $SOL$. We also propose Algorithm 3.

**Algorithm 3.(Interactive selection and enumeration method with a threshold value)**

```
Input: Such an object $x$ that a certain rule [CON,$\zeta$]⇒
    [DEC,$\eta$] is generated from $x$, and a threshold value $\alpha$.
Output: Interactively selected minimal certain rules from
    $x$.
begin
    fix a value $\alpha$; generate DF(x); CORE={}; SOL={};
    for (every descriptor DESC in DF(x))
      if ([A$_i$,$\zeta_i$] is a core) CORE=CORE∪{[A$_i$,$\zeta_i$]} and
      remove each disjunction with [A$_i$,$\zeta_i$] from DF(x);
    suppose DF'(x) be the revised discernibility function,
    and let LIST_DESC be a set of all descriptors in DF'(x);
    while (|SOL|+|LIST_DESC| $> \alpha$) do
      select a descriptor DESC in DF'(x), SOL=SOL∪
      {DESC} and assign the revised discernibility
      function to DF'(x);
      if DF'(x)=$\emptyset$ exit while loop;
    apply Algorithm 1 for either CORE∪SOL∪
    LIST_DESC or CORE∪SOL, and generate all
    minimal certain rules;
end.
```

In Algorithm 3, the threshold value $\alpha$ adjusts the application of Algorithm 1. For relatively small size $\alpha$, like 4 or 5, it may be necessary to specify several number of descriptors in $DF'(x)$, and small numbers of minimal certain rules are generated. In this case, Algorithm 1 works well for such small size of $|SOL| + |LIST\_DESC|$. On the other hand, for relatively large size $\alpha$, for example more than 15, it may not be necessary to specify any descriptors in $DF'(x)$, and most of minimal certain rules are generated. In this case, Algorithm 1 takes much execution time.

## VIII. Real Execution

This section shows real execution by a tool. Programs are implemented in prolog respectively, and they are realized on a workstation with 450 MHz UltraSparc CPU.

```
% more data.pl [Operation 1]
object(10,8).
data(1,[3,[1,3,4],3,2,5,5,[2,4],3]).
data(2,[2,[3,4],[1,3,4],4,[1,2],[2,4,5],2,2]).
data(3,[[4,5],5,[1,5],5,2,5,[1,2,5],1]).
data(4,[1,3,4,3,[1,2,3],1,[2,5],[1,2]]).
data(5,[4,1,[2,3,5],5,[2,3,4],[1,5],4,1]).
data(6,[4,1,5,1,4,[2,4,5],2,[1,2,3]]).
data(7,[2,4,3,4,3,[2,4,5],4,[1,2,3]]).
data(8,[4,5,4,[2,3,5],5,3,[1,2,3],[1,2,3]]).
data(9,[2,3,5,3,[1,3,5],4,2,3]).
data(10,[4,2,1,5,2,[4,5],3,1]).
total_cases(7346640384).
% more attrib.pl [Operation 2]
decision([8]).
decval([1]).
condition([1,2,3,4,5,6,7]).
```

In Operation 1, the contents in data file are displayed. The statement object(10,8) implies that there are 10 objects and 8 attributes. Every tuple of data is denoted by means of a list structure. In Operation 2, the contents of the attribute definition file are displayed. Here for the simplicity, we identify an attribute with the ordinal number of this attribute, for example an attribute B is identified with 2. The contents indicate the 8th attribute(=$\{H\}$) is the decision attribute and this attribute value is 1, namely rules, whose decision is [8,1], are specified. Attributes 1 to 7(=$\{A, B, \cdots, G\}$) are applied as condition attributes. According to such syntax, it is possible to define any $NIS$

and any rule.

```
% plc [Operation 3]
?-consult(rule.pl).  [Operation 4]
yes
?-translate.  [Operation 5]
File Name for Read Open:'data.pl'.
Decision Definition File:'attrib.pl'.
File Name for Write Open:'data.rs'.
EXEC_TIME=0.147(sec)
yes
?-init.  [Operation 6]
Certain Rules come from [3,5,10]
EXEC_TIME=0.003(sec)
yes
```

In Operation 3, prolog complier is invoked. In Operation 4, a prolog program is included. In Operation 5, the contents in the data file and the attribute definition file are translated into internal expressions, which are stored in data.rs file. Operation 6 shows it is possible to generate a certain rule from each of objects 3, 5 and 10. These are examined by means of checking such a formula like $\cap_{i=1,7,definite} sup(3,(\zeta_i),\{i\}) \subset inf(3,(1),\{8\})$. Therefore, minimal certain rules are only generated from these three objects.

```
?-minimal.  [Operation 7]
<<Minimal Certain Rules from object 3>>
DF:[[1,[2,5],[4,5],[5,2]],[2,[2,5],[4,5]],···,
[9,[2,5],[4,5],[5,2],[6,5]]]
<<Minimal Certain Rules from object 5>>
DF:[[1,[1,4],[4,5]],[2,[1,4],[2,1],[4,5],···,
[8,[2,1],[7,4]],[9,[1,4],[2,1],[4,5],[7,4]]]
<<Minimal Certain Rules from object 10>>
[2,2]=>[8,1] [324/324(=6/6,54/54),D-GC:Common]
Rule covers objects [10],Coverage=0.333
EXEC_TIME=0.015(sec)
yes
```

In Operation 7, all minimal certain rules from object 3, 5 and 10 are handled. The unique minimal certain rule, which appears in 324 derived *DISs* and belongs to *D-GC* class, exists just for object 10. As for objects 3 and 5, there is no unique minimal certain rule, and every discernibility function in each object is displayed. The discernibility function $DF(3)$ is internally in the form of

[[1,[2,5],[4,5],[5,2]],[2,[2,5],[4,5]],[4,[2,5],[4,5],[6,5]],[6,[2,5], [4,5],[5,2]],[7,[2,5],[4,5],[5,2]],[8,[5,2],[6,5]],[9,[2,5],[4,5], [5,2],[6,5]]].

In every list, the first element denotes an object, and the rest elements denote descriptors discriminating the object from $inf(3,(1),\{8\})$.

```
?-solall(3).  [Operation 8]
Number of Descriptors:5.  [Operation 9]
Exhaustive Search for less than 32 Cases !!
<<Minimal Certain Rules from object 3>>
Core Descriptors:[]
DF without Cores:[[1,[2,5],[4,5],[5,2]],···,
[8,[5,2],[6,5]],[9,[2,5],[4,5],[5,2],[6,5]]]
Currently Selected Descriptors:[]
[Loop:1]
Descriptors in DF:[[2,5],[4,5],[5,2],[6,5]]
Exhaustive Search([[2,5],[4,5],[5,2],[6,5]]),
16 Cases !!
Finally Selected Descriptors:[]
[4,5]&[6,5]=>[8,1] [17496/17496,D-GC]
Rule covers objects [3],Coverage=0.333
[4,5]&[5,2]=>[8,1] [8748/8748,D-GC]
Rule covers objects [3,10],Coverage=0.667
[2,5]&[6,5]=>[8,1] [34992/34992,D-GC]
Rule covers objects [3],Coverage=0.333
[2,5]&[5,2]=>[8,1] [17496/17496,D-GC]
Rule covers objects [3],Coverage=0.333
EXEC_TIME(for Exhaustive Search)=0.013(sec)
yes
```

In Operation 8, minimal certain rules from object 3 are handled. This program *solall* simulates Algorithm 3. A threshold value $\alpha$ is fixed to 5 in Operation 9. In Loop 1, there are four descriptors in this discernibility function, and this value is less than $\alpha$=5. Therefore, an exhaustive search begins for all subsets of four descriptors. Four minimal certain rules are generated. The 2nd rule $[4,5]\&[5,2]{=}{>}[8,1]$ comes from not only object 3 but also object 10.

```
?-solall(3).  [Operation 10]
Number of Descriptors:2.  [Operation 11]
Exhaustive Search for less than 4 Cases !!
<<Minimal Certain Rules from object 3>>
Core Descriptors:[]
DF without Cores:[[1,[2,5],[4,5],[5,2]],···,
[8,[5,2],[6,5]],[9,[2,5],[4,5],[5,2],[6,5]]]
Currently Selected Descriptors:[]
[Loop:1]
Descriptors in DF:[[2,5],[4,5],[5,2],[6,5]]
Select a Descriptor:[5,2].  [Operation 12]
DF without Cores:[[2,[2,5],[4,5]],[4,[2,5],
[4,5],[6,5]]]
Currently Selected Descriptors:[[5,2]]
[Loop:2]
Descriptors in DF:[[2,5],[4,5],[6,5]]
Common Descriptors in DF: [[2,5],[4,5]]
Exhaustive Search([[5,2],[2,5]]),4 Cases !!
Finally Selected Descriptors:[[5,2]]
[2,5]&[5,2]=>[8,1] [17496/17496,D-GC]
Rule covers objects [3],Coverage=0.333
Exhaustive Search([[5,2],[4,5]]),4 Cases !!
Finally Selected Descriptors:[[5,2]]
[4,5]&[5,2]=>[8,1] [8748/8748,D-GC]
Rule covers objects [3,10],Coverage=0.667
EXEC_TIME(for Exhaustive Search)=0.006(sec)
yes
```

In Operation 10, minimal certain rules from object 3 are handled again. In Operation 11, the number of specified descriptors is fixed to 2. In Loop 1, there are four descriptors in this discernibility function, and we select a descriptor [5,2]. In Loop 2, the revised discernibility function is displayed. The common descriptors [2,5] and [4,5] satisfy this new discernibility function. Thus, we have two sets of descriptors [[5,2],[2,5]] and [[5,2],[4,5]] for the original discernibility function. Both sets consists of less than

2 descriptors, so an exhaustive search begins for each set, respectively. Finally, we obtain two minimal certain rules, which are generated according to the selection [5,2].

## IX. Execution Time for Other NISs

Data $NIS_1$ in the previous section is very small, so we define other $NISs$ and examine the execution time for other $NISs$. Table II shows the details of three $NISs$. These $NISs$ are also artificial data. The column of derived $DISs$ indicates the number of all derived $DISs$ from $NIS$, respectively. Intuitively, it seems hard to pick up every derived $DISs$ sequentially.

| $NIS$ | $\|OB\|$ | $\|AT\|$ | $\|VAL_A\|$ | $derived\_DISs$ |
|---|---|---|---|---|
| $NIS_2$ | 50 | 10 | 10 | $1.57 \times 10^{18}$ |
| $NIS_3$ | 100 | 10 | 10 | $7.01 \times 10^{35}$ |
| $NIS_4$ | 300 | 10 | 10 | $6.74 \times 10^{86}$ |

TABLE II
Definitions of NISs

| $NIS$ | $translate$ | $minimal$ | $object$ | $solall$ |
|---|---|---|---|---|
| $NIS_2$ | 0.896 | 0.723 | 7 | 0.764 |
| $NIS_3$ | 6.503 | 3.589 | 16 | 1.370 |
| $NIS_4$ | 49.892 | 35.345 | 21 | 2.943 |

TABLE III
Execution time(sec) of programs

In Table III, the *object* column implies the number of objects, in which some minimal certain rules are generated. The execution time of *minimal* depends upon the number of objects. A program *solall* are also applied to an object in each $NIS$. In this execution, the threshold value $\alpha$ was fixed to 10. Since $|AT|$=10, this program began to enumerate $1024(= 2^{10})$ subsets without specifying any descriptors, and generated all minimal certain rules. According to Table III, we may employ $\alpha$=10 for $NISs$ as large as $NIS_4$. For $NISs$ with more large number of attributes, we sequentially specify some descriptors in $DF(x)$, and $DF(x)$ is sequentially reduced. When the sum of specified descriptors and descriptors in the revised $DF(x)$ is smaller than $\alpha$, Algorithm 1 is invoked.

## X. Concluding Remarks

A tool for generating minimal certain rules in $NISs$ is presented. Minimal certain rules are not influenced by the information incompleteness, and they are always consistent in every derived $DIS$. We will know the property and the tendency of data through generated minimal certain rules.

In Example 3, we stated a problem of generating minimal certain rules, i.e., interactive selection method may not generate minimal certain rules. In order to solve this problem, we applied enumeration method to a set of descriptors satisfying $DF(x)$. Since such problem has also been discussed in [18,19], it is necessary to include some theoretical results on monotonic Boolean functions to our programs. We have to clarify the computational complexity of proposed algorithms, too.

As we have discussed in section VIII, it is possible to control minimal rule generation in $NISs$ by means of adjusting a threshold value $\alpha$. For relatively large size of threshold $\alpha$, we obtain most minimal certain rules from objects but it may take much execution time. On the other hand for relatively small size of $\alpha$, it is necessary to specify some descriptors, and we may obtain small number of minimal certain rules based on specified descriptors. Although, this will take less execution time. According to this control, it is possible to generate minimal certain rules for every user's purpose. We will apply this tool to real data with much missing values and uncertain values.

The author is grateful to anonymous referees for their useful comments.

## References

[1] Z.Pawlak: Rough Sets, Kluwer Academic Publisher, 1991.
[2] J.Komorowski, Z.Pawlak, L.Polkowski and A.Skowron: Rough Sets: a tutorial, Rough Fuzzy Hybridization, Springer, pp.3-98, 1999.
[3] A.Nakamura, S.Tsumoto, H.Tanaka and S.Kobayashi: Rough Set Theory and Its Applications, J.JSAI, Vol.11, No.2, pp.209-215, 1996.
[4] S.Tsumoto: Knowledge Discovery in Clinical Databases and Evaluation of Discovered Knowledge in Outpatient Clinic, Information Sciences, Vol.124, pp.125-137, 2000.
[5] E.Orłowska and Z.Pawlak: Representation of Nondeterministic Information, Theoretical Computer Science, Vol.29, pp.27-39, 1984.
[6] E.Orłowska (Ed.): Incomplete Information: Rough Set Analysis, Physica-Verlag, 1998.
[7] W.Lipski: On Semantic Issues Connected with Incomplete Information Data Base, ACM Trans. DBS, Vol.4, pp. 269-296, 1979.
[8] W.Lipski: On Databases with Incomplete Information, Journal of the ACM, Vol.28, pp.41-70, 1981.
[9] J.Grzymala-Busse: On the unknown attribute values in learning from examples, Proc. ISMIS'91, Lecture Notes in AI, Springer-Verlag, Vol.542, pp.368-377, 1991.
[10] J.Grzymala-Busse and P.Werbrouck: On the Best Search Method in the LEM1 and LEM2 Algorithms, Incomplete Information: Rough Set Analysis, Phisica-Verlag, pp.75-91, 1998.
[11] M.Kryszkiewicz: Rough Set Approach to Incomplete Information Systems, Information Sciences, Vol.112, pp.39-49, 1998.
[12] M.Kryszkiewicz: Rules in Incomplete Information Systems, Information Sciences, Vol.113, pp.271-292, 1999.
[13] H.Sakai: Effective Procedures for Handling Possible Equivalence Relations in Non-deterministic Information Systems, Fundamenta Informaticae, Vol.48, pp.343-362, 2001.
[14] H.Sakai: A Framework of Rough Sets based Rule Generation in Non-deterministic Information Systems, Lecture Notes in AI, Vol.2871, Springer-Verlag, pp.143-151, 2003.
[15] E.Codd: A Relational Model of Data for Large Shared Data Banks, Communication of the ACM, Vol.13, pp.377-387, 1970.
[16] M.Nakata and S.Miyamoto: Databases with Non-deterministic Information, Bulletin of Int'l. Rough Set Society, Vol.7, pp.15-21, 2003.
[17] A.Skowron and C.Rauszer: The Discernibility Matrices and Functions in Information Systems, In Intelligent Decision Support - Handbook of Advances and Applications of the Rough Set Theory, Kluwer Academic Publishers, pp. 311-369, 1992.
[18] M.Kryszkiewicz: Maintenance of Reducts in the Variable Precision Rough Sets Model, ICS Research Report 31/94, Warsaw University of Technology, 1994.
[19] M.Kryszkiewicz and H.Rybinski: Computation of Reducts of Composed Information Systems, Fundamenta Informaticae, Vol.27, pp.183-195, 1996.
[20] H.Sakai: An Interactive Tool for Generating Minimal Certain Rules in Non-deterministic Information Systems, Proc. International Workshop on Fuzzy Systems and Innovational Computing, Japan Society for Fuzzy Theory and Intelligent Informatics, D4-2, pp.1-6, 2004.