On Two Apriori-Based Rule Generators: Apriori in Prolog and Apriori in SQL

Paper: jc*-**-**-****

On Two Apriori-Based Rule Generators: Apriori in Prolog and Apriori in SQL

Hiroshi Sakai[†], Kao-Yi Shen[††], Michinori Nakata[†††]

[†]Department of Basic Sciences, Graduate School of Engineering,
Kyushu Institute of Technology,
Tobata, Kitakyushu 804-8550, Japan
email: sakai@mns.kyutech.ac.jp
[††]Department of Banking and Finance,
Chinese Culture University (SCE), Taipei, Taiwan
email: kyshen@sce.ppcu.edu.tw
[†††]Faculty of Management and Information Science,
Josai International University,
Gumyo, Togane, Chiba 283-8555, Japan
email: nakatam@ieee.org

This paper focuses on two Apriori-based rule generators. The first is the rule generator in Prolog and C, and the second is the one in SQL. They are named *Apriori in Prolog* and *Apriori in SQL*, respectively. Each rule generator is based on the Apriori algorithm. However, each rule generator has its own properties. Apriori in Prolog employs the equivalence classes defined by table data sets and follows the framework of rough sets. On the other hand, Apriori in SQL employs a search for rule generation and does not make use of equivalence classes. This paper clarifies the properties of these two rule generators and considers effective applications of each to existing data sets.

Keywords: Rule generation, Apriori algorithm, Table data analysis, Association rules, Prolog, SQL

1. Introduction

Under conditions of uncertainty, we have been investigating the Apriori algorithm [1], rough sets [10, 21], non-deterministic information [9], incomplete information databases [7], missing values [5], among others, for rule generation, data mining, and decision support [19].

Our interest comprises three issues. The first is improving the theoretical framework of rule generation. The second is implementing rule generators with proven properties. The third is applying the rule generators to existing data sets. Each obtained rule becomes fundamental knowledge for decision support. In the theoretical framework, we already proposed the framework of *Rough Sets Non-deterministic Information Analysis* (RNIA), which combines rough sets and non-deterministic information for rule generation under uncertainty [8, 14, 17, 18]. In the application of rule generators, the second author handled existing data sets like credit card evaluation and banking data sets [20].

This paper is closely related to the second issue: the implementation of rule generators. Apriori in Prolog employs the concept of rough sets and makes use of equivalence classes defined in tables. There are, of course, rule generation systems based on rough sets [2, 12]. However, Apriori in SQL follows the framework of relational data bases and sequentially manipulates tables to obtain the final table, including rules. We clarify the property of these two rule generators and consider more effective uses of them.

We are also coping with the Apriori algorithm for table data sets with non-deterministic information. This is an extension from the Apriori algorithm handling deterministic information to that handling non-deterministic information. This paper is the foundation for NIS-Apriori in Prolog [13] and NIS-Apriori in SQL [15]. Currently, Apriori in SQL may be familiar, but NIS-Apriori in SQL is a new framework for uncertain information.

This paper is organized as follows. Section 2 considers the framework of Apriori-based rule generation. Section 3 investigates the properties of equivalence classes defined by table data sets and describes Apriori in Prolog by using the actual execution logs. Section 4 clarifies the functionality of SQL and investigates Apriori in SQL. Section 5 extends Apriori in SQL to NIS-Apriori in SQL for rule generation from uncertain information. Section 6 compares Apriori-based rule generation with rough set-based rule generation. Section 7 concludes this paper.

2. Apriori-Based Rule Generation in Table Data Sets

This section describes some definitions for Apriori-based rule generation. We follow the framework of rough sets and handle each table data set as a *Deterministic Information System* (DIS).

2.1. Table Data Sets and Rules

DIS ψ is a quadruplet

$$\psi = (OB, AT, \{VAL_A | A \in AT\}, f), \tag{1}$$

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 $f : OB \times AT \to \cup_{A \in AT} VAL_A$. An attribute $Dec \in AT$ is usually predefined and is called a *decision attribute*. A subset CON of $AT \setminus \{Dec\}$ is called a set of *condition attributes*. In ψ, a pair $[A, val]$ ($A \in AT$, $val \in VAL_A$) is called a *descriptor*, and a formula τ in the following is called an *implication*.

$$\begin{array}{l} \tau : \wedge_{A \in CON}[A, val_A] \Rightarrow [Dec, val], \\ val_A \in VAL_A, val \in VAL_{Dec}. \end{array} \tag{2}$$

Definition 1. [10, 21] For DIS ψ, two given threshold values $0 < \alpha$, $\beta \leq 1.0$, an implication τ satisfying (1) and (2) is called (a candidate of) a *rule* in ψ.

$$\begin{array}{l} (1)\ support(\tau) = \frac{|OBJ(\wedge_{A \in CON}[A, val_A] \wedge [Dec, val])|}{|OB|} \geq \alpha, \\ (2)\ accuracy(\tau) = \frac{|OBJ(\wedge_{A \in CON}[A, val_A] \wedge [Dec, val])|}{|OBJ(\wedge_{A \in CON}[A, val_A])|} \geq \beta. \end{array} \tag{3}$$

Here, $OBJ(*)$ is a set of the objects satisfying the formula $*$, and $|M|$ (M is a set of objects) is the cardinality of the set M. If $|OBJ(\wedge_{A \in CON}[A, val_A])|$=0, we define $support(\tau)$=$accuracy(\tau)$=0.

Table 1. An exemplary DIS ψ_1.

OB	$color$	$shape$	$price$
1	red	$square$	low
2	red	$triangle$	low
3	$blue$	$round$	$high$
4	$blue$	$square$	$high$
5	$blue$	$square$	low

Example 1. (1) For an exemplary DIS ψ_1 in Table 1, we suppose two threshold values α=0.2 and β=0.7, and an implication $\tau_1 : [color, blue] \Rightarrow [price, high]$. Since $|OBJ([color, blue])|$=$|\{3,4,5\}|$=3 and $|OBJ(\tau_1)|$=$|\{3,4\}|$=2, $support(\tau_1)$=$2/5 > 0.2$ and $accuracy(\tau_1)$=$2/3 < 0.7$ hold. Thus, we see τ_1 is not a rule.
(2) For $\tau_2 : [color, blue] \wedge [shape, round] \Rightarrow [price, high]$, we have $|OBJ([color, blue] \wedge [shape, round])|$=$|\{3\}|$=1 and $|OBJ(\tau_2)|$=$|\{3\}|$=1, and $support(\tau_2)$=$1/5 \geq 0.2$ and $accuracy(\tau_2)$=$1/1 > 0.7$ hold. Thus, we see τ_2 is a rule.

The $support(\tau)$ value means the occurrence ratio of τ in ψ, and the $accuracy(\tau)$ means the consistency ratio of τ in ψ. The purpose of rule generation in ψ is to obtain all rules specified in Definition 1.

2.2. The Adjusted Apriori Algorithm to Table Data Sets

The *Apriori algorithm* was proposed by Agrawal, and it is known as the representative algorithm in data mining [1, 3]. This algorithm originally handled transaction data sets instead of table data sets. However, if we identify each descriptor in a table data set with an item, we can consider the Apriori algorithm in tables [13, 14]. For example, we see the tuple of the object 1 in Table 1 implies an item set $\{[color, red], [shape, square], [price, low]\}$. We name this algorithm an *adjusted Apriori algorithm*.

The Apriori algorithm does not take such a method that each implication τ is sequentially created and two values $support(\tau)$ and $accuracy(\tau)$ are evaluated for examining (1) and (2) in Definition 1. Instead of it, the Apriori algorithm enumerates each implication τ, which may satisfies (1) in Definition 1. This method reduces the number of implications to be handled. In order to clarify the adjusted Apriori algorithm, we consider the following proposition.
Proposition 1. In DIS ψ, let P be an attribute in $(AT \setminus CON)$. For two implications below:

$$\begin{array}{l} \tau : \wedge_{A \in CON}[A, val_A] \Rightarrow [Dec, val], \\ \eta : ([P, val_P] \wedge (\wedge_{A \in CON}[A, val_A])) \Rightarrow [Dec, val]. \end{array} \tag{4}$$

we have the following.
(1) $support(\eta) \leq support(\tau)$ always holds.
(2) Regarding $accuracy(\tau)$ and $accuracy(\eta)$, we theoretically do not have one unique inequality. The inequality depends on each data set.
(Proof) (1) In η, we need to add another condition $[P, val_P]$ to each object, so the number of objects satisfying η is less than that of τ. This causes the unique inequality.
(2) In Table 1, $accuracy(\tau_1)(= 2/5) \leq accuracy(\tau_2)$ (=1.0) holds. For $\tau_3 : [color, blue] \wedge [shape, square] \Rightarrow [price, high]$, $accuracy(\tau_3)$=1/2 holds. So in this case, $accuracy(\tau_3) \leq accuracy(\tau_1)$. This is an example, and we cannot conclude one inequality.
Remark 1. For τ and η in Proposition 1, we say η is *redundant* to τ. If τ is a rule, we say η is a *redundant rule*.

$$\begin{array}{l} (A \wedge B \Rightarrow Dec) = (\neg(A \wedge B) \vee Dec) = \\ (\neg A \vee \neg B \vee Dec) = ((\neg A \vee Dec) \vee (\neg B \vee Dec)) = \\ ((A \Rightarrow Dec) \vee (B \Rightarrow Dec)). \end{array} \tag{5}$$

Due to the above formula (5), we can conclude $A \wedge B \Rightarrow Dec$ from $A \Rightarrow Dec$. For simplicity, we do not consider any redundant rule. We only handle each rule whose condition part is minimum, which may be called a *prime implicant*.
Definition 2. For DIS ψ, a threshold value α, and each implication $\tau : \wedge_{A \in CON}[A, val_A] \Rightarrow [Dec, val]$, we define IMP_i=$\{\tau \mid |CON| = i$ and $support(\tau) \geq \alpha\}$. The subscript i in IMP_i means the number of descriptors in the condition part.

For example,
IMP_1=$\{\tau : [A, val_A] \Rightarrow [Dec, val]$, $support(\tau) \geq \alpha\}$,
IMP_2=$\{\tau : [A, val_A] \wedge [B, val_B] \Rightarrow [Dec, val]$,
$support(\tau) \geq \alpha\}$,

IMP_3=$\{\tau : [A, val_A] \wedge [B, val_B] \wedge [C, val_C] \Rightarrow [Dec, val]$, $support(\tau) \geq \alpha\}$.

Since every implication τ is an element in $IMP_{|CON|}$, we do no miss any τ if we sequentially examine IMP_1, IMP_2, IMP_3, $\cdots$. On the other hand, we need to handle a large number of meaningless implication, if we consider each IMP_1, IMP_2, IMP_3, $\cdots$, independently. However, it is enough to consider implications related to IMP_1 based on the next Proposition 2. This is an effective property for handling IMP_1, IMP_2, IMP_3, $\cdots$.

Proposition 2 For each implication $\tau : [P, val_P] \wedge [Q, val_Q] \Rightarrow [Dec, val] \in IMP_2$, $[P, val_P] \Rightarrow [Dec, val] \in IMP_1$ and $[Q, val_Q] \Rightarrow [Dec, val] \in IMP_1$ must hold.

(Proof) If $[P, val_P] \Rightarrow [Dec, val] \notin IMP_1$, this implication does not satisfy the condition of *support*. Due to (1) in Proposition 1, we have $support(\tau) \leq \alpha$.

Since Proposition 2 is recursively applicable, we can reduce the elements of IMP_i by using Proposition 2. It is enough for us to consider IMP_1, IMP_2, $\cdots$, whose elements are the combination of implications in IMP_1.

Algorithm 1 is the adjusted Apriori algorithm to table data sets and employs Propositions 1 and 2.

Algorithm 1. The adjusted Apriori algorithm.

Input: DIS ψ, the decision attribute *Dec*, the threshold values α and β.

Output: A set $Rule(\psi)$ consisting of all rules.

begin
$Rule(\psi)$:={}; i:=1; generate IMP_i;
while ($|IMP_i| \geq 1$)
{ $Rest$:={};
for each $\tau \in IMP_i$, if $accuracy(\tau) \geq \beta$ then add τ to $Rule(\psi)$ else add τ to $Rest$;
i:=i+1;
generate IMP_i by using $Rest$ and Propositions 1 and 2, where $\tau(\in IMP_i)$ is not any redundant implication to rules in $Rule(\psi)$ }
end.

Regarding the adjusted Apriori algorithm, we have the following theorem. Thus, Algorithm 1 is necessary and sufficient for obtaining rules (excepting the redundant rules) in Definition 1.

Theorem 1. [14] The adjusted Apriori algorithm is *sound* and *complete* for a set of rules (excepting the redundant rules) defined by $support(\tau) \geq \alpha$ and $accuracy(\tau) \geq \beta$ in DIS ψ. Namely, each $\tau \in Rule(\psi)$ is a rule (soundness) and any rule τ belongs to $Rule(\psi)$ (completeness).

3. Apriori-Based Rule Generator in Prolog and Its Property

This section considers Apriori in Prolog and its implementation by using rough set-based concepts.

3.1. Equivalence Classes in Tables

In DIS ψ, we see there is a relation between two objects x and y for ATR, if $f(x,A)$=$f(y,A)$ holds for every $A \in ATR \subseteq AT$. This relation is known as an equivalence relation over OB [10]. Let $eq(ATR)$ denote a set of the equivalence classes with respect to ATR, and let $[x]_{ATR} \in eq(ATR)$ denote an equivalence class below:

$$\{y \in OB \mid f(y,A) = f(x,A) \text{ for every } A \in ATR\}. \quad (6)$$

We connect the calculation of $support(\tau)$ and $accuracy(\tau)$ with the equivalence classes. For an object x, CON and Dec, we automatically have an implication $\tau^x : \wedge_{A \in CON}[A, f(x,A)] \Rightarrow [Dec, f(x,Dec)]$. (We employ the notation τ^x, if we need to specify the object x.) Since $OBJ(\wedge_{A \in CON}[A, f(x,A)])$ is a set of objects with attribute values $f(x,A)$ for $A \in CON$, it is the same set of $[x]_{CON}$. Similarly, $OBJ(\tau^x)$ is an equivalence class $[x]_{CON \cup \{Dec\}}$.

Proposition 3. [10] For A, $B \in AT$, let $[x]_{\{A\}}$ and $[x]_{\{B\}}$ be two equivalence classes with an object x, then we have $[x]_{\{A,B\}}$=$[x]_{\{A\}} \cap [x]_{\{B\}}$.

Proposition 3 shows us that any equivalence class $[x]_{ATR}$ is equal to $\cap_{A \in ATR}[x]_{\{A\}}$, and we have the formulas in the following:

$$\begin{aligned} support(\tau^x) &= |[x]_{CON \cup \{Dec\}}| \,/\, |OB| \\ &= |(\cap_{A \in CON}[x]_{\{A\}}) \cap [x]_{\{Dec\}}| \,/\, |OB|, \\ accuracy(\tau^x) &= |[x]_{CON \cup \{Dec\}}| \,/\, |[x]_{CON}| \\ &= |(\cap_{A \in CON}[x]_{\{A\}}) \cap [x]_{\{Dec\}}| \,/\, |\cap_{A \in CON}[x]_{\{A\}}|. \end{aligned} \quad (7)$$

Remark 2. One of the original property in Prolog is the manipulation of lists. Prolog has the good performance for the list processing. Here, we express each equivalence class $[x]_{\{A\}}$ by a list and make use of the list processing functionality in Prolog.

3.2. Apriori in Prolog with Equivalence Classes

This subsection considers the calculation of $support(\tau)$ and $accuracy(\tau)$ in Algorithm 1. We employ the list processing functionality in Prolog. For simplicity, we use Table 1 below, which is expressed in Prolog.

```
support(0.2).  accuracy(0.7).
decision(3).  condition([1,2]).
data(1,[red,square,low]).
data(2,[red,triangle,low]).
data(3,[blue,round,high]).
data(4,[blue,square,high]).
data(5,[blue,square,low]).
```

The following is the execution logs. The step1 command generates rules from IMP_1, and three rules are generated. There is no rule from IMP_2 nor IMP_3.

```
?- step1.
File Name for Read Open:jaciii.rs.
SUPPORT:0.2,ACCURACY:0.7
[1] MINSUPP=0.0,MINACC=0.0
[2] MINSUPP=0.4,MINACC=1.0
 [color,red] => [price,low][1,2]
     :        :        :        :
[8] MINSUPP=0.2,MINACC=1.0
 [shape,triangle] => [price,low][2]
[9] MINSUPP=0.2,MINACC=1.0
 [shape,round] => [price,high][3]
[10] MINSUPP=0.0,MINACC=0.0
```

```
(Next Candidates are Remained)
[[[1,2],[3,1]],[[1,2],[3,2]],[[2,1],[3,1]],
[[2,1],[3,2]]]
EXEC_TIME=0.0(sec)
yes
?- step2.
[1] MINSUPP=0.2,MINACC=0.5
[2] MINSUPP=0.2,MINACC=0.5
(Next Candidates are Remained)
[[[1,2],[2,1],[3,1]],[[1,2],[2,1],[3,2]]]
EXEC_TIME=0.0(sec)
yes
?- step3.
(System Terminated)
EXEC_TIME=0.0(sec)
```

Internally, the following lists (In Prolog, they are called *predicates*) are generated at first. Each descriptor satisfies the condition $|OBJ(*)| \geq 0.2$. The pair of the first and the second arguments is the index of each descriptor, and the fourth argument is the equivalence class. We name each predicate *eq* with four arguments a *primitive predicate of a descriptor*.

```
eq(1,1,[color,red],[1,2]).
eq(1,2,[color,blue],[3,4,5]).
eq(2,1,[shape,square],[1,4,5]).
eq(2,2,[shape,triangle],[2]).
eq(2,3,[shape,round],[3]).
eq(3,1,[price,high],[3,4]).
eq(3,2,[price,low],[1,2,5]).
```

By Proposition 2, In step1, the following list (the Cartesian products) is internally generated.

```
conjunction([[[1,1],[3,1]],[[1,1],[3,2]],
[[1,2],[3,1]],[[1,2],[3,2]],[[2,1],[3,1]],
[[2,1],[3,2]],[[2,2],[3,1]],[[2,2],[3,2]],
[[2,3],[3,1]],[[2,3],[3,2]]]).
```

The first list $[[1,1],[3,1]]$ means an implication $[color, red] \Rightarrow [price, high]$ by the index of each descriptor. Each implication is expressed by such list. After step2, we see [[1,2],[2,1],[3,1]] in the next candidates. This means $\tau : [color, blue] \wedge [shape, square] \Rightarrow [price, high]$, and this is an element of *Rest* in Algorithm 1. By Proposition 3, $OBJ([color, blue] \wedge [shape, square])$ is equal to $OBJ([color, blue]) \cap OBJ([shape, square])$, so $\{3,4,5\} \cap \{1,4,5\}=\{4,5\}$ is derived, and $OBJ(\tau)=\{4,5\} \cap OBJ([price, high])=\{4\}$. In this way, we have $support(\tau)=2/5>0.2$ and $accuracy(\tau)=1/2<0.7$. Like this, we can calculate $support(\tau)$ and $accuracy(\tau)$ in Algorithm 1.

3.3. Implementation and Some Properties of Apriori in Prolog

Apriori in Prolog is coded by K-Prolog, and each predicate is compiled by using C. This work is extended to the system *getRNIA* [24] in Python. We also extended Apriori in Prolog to NIS-Apriori in Prolog for handling not only DISs ψ but also *Non-deterministic Information Systems* (NISs) [13].

Now, we enumerate some advantages and disadvantages of Apriori in Prolog.

(Advantage AP (Apriori in Prolog) 1) In the execution, Apriori in Prolog generates a set of primitive predicates of a descriptor below:

$$eq(index_1, index_2, descriptor, OBJ(descriptor)). \quad (8)$$

Furthermore, it is possible to remove such descriptors that $|OBJ(descriptor)| \,/\, |OB| < \alpha$ by using Proposition 1. By using Propositions 2 and 3, it is possible to have IMP_1, IMP_2, $\cdots$ and to calculate $support(\tau)$ and $accuracy(\tau)$. Namely, it is enough for us to consider the set of primitive predicates of a descriptor for rule generation.

(Advantage AP2) Prolog has the good performance for the list processing, so Prolog will be suitable for handling the equivalence classes. For small size data sets, Apriori in Prolog will be effective.

(Disadvantage AP1) The description of the original data must follow the syntax of Prolog. It is not easy to prepare this description.

(Disadvantage AP2) For large size data sets, the length of the list expressing $OBJ(descriptor)$ may become large. In the Mammographic data set [4] in UCI machine learning repository, we have below: (In [18], we opened the execution logs by Apriori in Prolog.)

```
?- eq(5,3,P,Q),length(Q,T).
P=[density,3],
Q=[2,3,5,6,7,8,10,11,12,15,16,17,18,23,24,25,27,
     :       :       :       :
951,952,953,954,955,956,957,958,959,960],
T=797
```

The list Q consists of 797 objects for totally 960 objects. Even though there is no restriction on the length of each list, we sometimes had stack overflow in the execution. Actually, we could handle lists which consist of less than 1000 objects, but we had the execution errors for lists consisting of more than 2000 objects. We increased the local stack and the global stack for execution in such case. Furthermore, Apriori in Prolog handles such plural lists at the same time. Due to this constraint for the execution environment, Disadvantage AP2 seems a very important problem, and we figure out that Apriori in Prolog may not be suitable for large size data set.

4. Apriori-Based Rule Generator in SQL and Its Property

This section considers Apriori in SQL and its implementation. Some execution logs are in the web page [18].

4.1. The Functionality of SQL and a Data Format

SQL was originally developed for information retrieval, and SQL easily manipulates table data. Even though rule generation is different from information retrieval, the functionality of SQL is useful for rule generation. Some frameworks of rule generation depending on SQL are proposed [6, 22, 23].

In data sets, we usually have the csv format. This is very familiar, however the number of all attributes and the name of each attribute are different. So, if the rule generation program is depending upon each data set, we need to have one program for one data set. In order to escape from this situation, we need another unified format handling various types of data sets uniformly.

4.2. RDF Format for Apriori in SQL

This subsection considers the RDF (or EAV) format [22, 23] as the unified format handling various types of data sets, and we consider some procedures in SQL.

Each program is implemented by using the procedures in SQL. Therefore, it is easy to execute Apriori in SQL in any PC. Figure 1 shows the rdf table generated from Table 1. This rdf table stores each tuple as a set of descriptors. After this translation from table data set to rdf table, there are only three attributes *object*, *attrib*, *value*. Each attribute in a data set is stored as a value of *attribute*. All procedures in SQL handle this translated rdf table for rule generation. By using this rdf table, it is possible to handle any data set.

```
mysql> select * from rdf where object<=3;
+--------+--------+----------+
| object | attrib | value    |
+--------+--------+----------+
|      1 | color  | red      |
|      1 | price  | low      |
|      1 | shape  | square   |
|      2 | color  | red      |
|      2 | price  | low      |
|      2 | shape  | triangle |
|      3 | color  | blue     |
|      3 | price  | high     |
|      3 | shape  | round    |
+--------+--------+----------+
9 rows in set (0.00 sec)
```

Fig. 1. Some parts of the rdf table for Table 1.

```
mysql> show tables;
+------------------+
| Tables_in_jaciii |
+------------------+
| rdf              |
| table 1          |
+------------------+
2 rows in set (0.00 sec)

mysql> call apriv2('price',5,0.2,0.7);
Query OK, 0 rows affected (2.08 sec)

mysql> show tables;
+------------------+
| Tables_in_jaciii |
+------------------+
| con1             |
| con2             |
| con3             |
| condi            |
| deci             |
| rdf              |
| rest1            |
| rest2            |
| rest3            |
| rule1            |
| rule2            |
| rule3            |
| table 1          |
+------------------+
13 rows in set (0.00 sec)
```

Fig. 2. Rule generation by Apriori in SQL.

Figure 2 is the actual execution log for rule generation from Table 1. The procedure $apriv2('price', 5, 0.2, 0.7)$ means that the decision attribute is 'price', 5 objects, $support(\tau) \geq 0.2$ and $accuracy(\tau) \geq 0.7$. This is the same condition in Section 3. In Figure 2, the table rule1 stores rules in IMP_1 and the table rest1 stores implication in *Rest* in Algorithm 1. Figure 3 shows tables rule1 and rest1. From the table rest1, Apriori in SQL generates the table con2 consisting of one conjunction $[color, blue] \wedge [shape, square]$, then two implications $\tau : [color, blue] \wedge [shape, square] \Rightarrow [Dec, high]$ and $\eta : [color, blue] \wedge [shape, square] \Rightarrow [Dec, low]$ are examined. Two implications mutually do not satisfy the condition, so we have the table rest2 consisting τ and η. Since it is impossible to have any implication with three descriptors in the condition part, IMP_3 is an empty set, and Apriori in SQL ends.

The procedure *apriv2* below simulates the adjusted Apriori algorithm. The following is the overview of the series (for IMP_1, IMP_2, and IMP_3) of the SQL procedures. Especially, we describe the SQL code for rule generation in Figure 3. This part seems to be comprehensible, and the most complicated part is to manage the redundant implications and to generate the condition part by Rest in Algorithm1 .

```
delimiter //
create procedure apriv2
begin
create table condi(); /* a table of
  Dec, |OB|, α, β */
create table deci(); /* a table of Dec */
create table con1(); /* a table of CON */
create table rule1(); /* a table of the rules */
create table rest1(); /* a table of the rest */
create table con20(), con21(), con2();
  /* a table of p1∧p2 from rest1 */
create table con2_eq0(), con2_eq();
  /* a table of eq information by search */
create table rule21(), rule2();
  /* a table of rule2 */
create table rest2(); /* a table of rest */
create table con30(), con31(), con3();
  /* a table of p1∧p2∧p3 from rest2 */
create table con3_eq0(), con3_eq();
  /* a table of eq information by search */
create table rule31(), rule3();
  /* a table of rule3 */
end //
```

4.3. Properties of Apriori in SQL

Now, we refer to the properties of Apriori in SQL. Apriori in Prolog makes use of the equivalence classes, but Apriori in SQL uses search for calculating $support(\tau)$

```
mysql> select * from rule1;
+------------+----------+-------+------------+---------+----------+
| att1       | val1     | deci  | deci_value | support | accuracy |
+------------+----------+-------+------------+---------+----------+
| color      | red      | price | low        |   0.400 |    1.000 |
| shape      | round    | price | high       |   0.200 |    1.000 |
| shape      | triangle | price | low        |   0.200 |    1.000 |
| end_attrib | NULL     | NULL  | NULL       |    NULL |     NULL |
+------------+----------+-------+------------+---------+----------+
4 rows in set (0.00 sec)

mysql> select * from rest1;
+-------+--------+-------+------------+---------+----------+
| att1  | val1   | deci  | deci_value | support | accuracy |
+-------+--------+-------+------------+---------+----------+
| color | blue   | price | high       |   0.400 |    0.667 |
| color | blue   | price | low        |   0.200 |    0.333 |
| shape | square | price | high       |   0.200 |    0.333 |
| shape | square | price | low        |   0.400 |    0.667 |
+-------+--------+-------+------------+---------+----------+
4 rows in set (0.00 sec)
```

The following is the SQL code for generating table rule1.

```
create table rule1 (att1 varchar(15),val1 varchar(15),deci varchar(15),
   deci_value varchar(15),support decimal(15,3),accuracy decimal(15,3))
select att1,val1,deci,deci_value,count(*)/ob as support,
   count(*)/(con1.support) as accuracy
from con1,deci,rdf t1,rdf t2
where att1 <> decision and t1.attrib=att1 and t1.value=val1 and
   t2.attrib=decision and t2.value=deci_value and t1.object=t2.object
group by att1,val1,deci,deci_value
having support>=alpha and accuracy>=beta;
insert into rule1 (att1) values ('end_attrib');
```

The following is the SQL code for generating table rule2. The procedure rule2 removes redundant rules from the table rule21 (IMP_2) and generates the table rule2.

```
create table rule2 (att1 varchar(15),val1 varchar(15),att2 varchar(15),
   val2 varchar(15),deci varchar(15),deci_value varchar(15),
   support decimal(15,3),accuracy decimal(15,3))
select att1,val1,att2,val2,deci,deci_value,support,accuracy
from rule21
where not exists (
   select *
   from rule1 t
   where
      (t.att1 <> 'end_attrib' and rule21.att1=t.att1 and rule21.val1=t.val1
      and rule21.deci=t.deci and rule21.deci_value=t.deci_value)
   or
      (t.att1 <> 'end_attrib' and rule21.att2=t.att1 and rule21.val2=t.val1
      and rule21.deci=t.deci and rule21.deci_value=t.deci_value)
    )
group by att1,val1,att2,val2,deci,deci_value;
insert into rule2 (att1) values ('end_attrib');
```

Fig. 3. Three rules and four implications in the list Rest, and the SQL codes for tables rule1 and rule 2.

and $accuracy(\tau)$. This is the most different point between two generators, and we have the following with respect to Apriori in SQL.

(Advantage AQ (Apriori in SQL) 1) The environment of SQL is familiar in PC, and it is easy to execute the implemented procedures. The system Xampp [11] with the user interface is effective for execution. After the translation from a csv data set to a rdf table, it is possible to handle any data set uniformly. This seems much better than that of Apriori in Prolog.

(Advantage AQ2) Apriori in SQL employs search instead of the equivalence classes. Therefore, we can es-

cape from (Disadvantage AP2), and it will be possible to handle large size data sets. Actually, we could easily have rule generation for some data sets [18].

(Disadvantage AQ1) For a small size data sets, Apriori in Prolog seems much faster than Apriori in SQL. In the execution of Table 1 by Apriori in Prolog, step1, step2, and step3 took 0.0 seconds, respectively. However, Apriori in SQL took about 2 seconds. In Mammographic data set [18], NIS-Apriori in Prolog took about 5 seconds for step1, step2, and step 3. However NIS-Apriori in SQL took about 200 seconds. Even though it is necessary to brush up the code of NIS-Apriori in SQL, NIS-Apriori in Prolog seems better than NIS-Apriori in SQL with respect to the Mammographic data set [18].

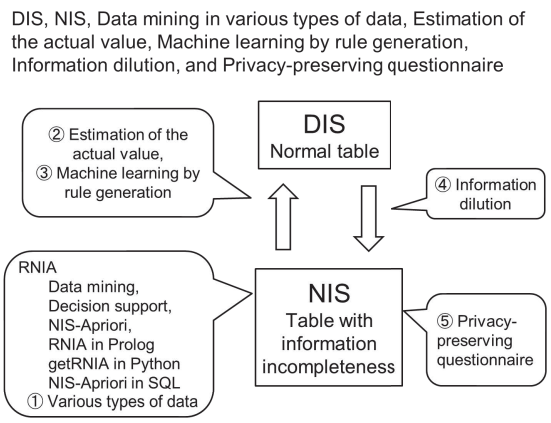

Fig. 4. An overview of new topics [17].

5. From Table Data Analysis to Table Data Analysis with Uncertainty

This paper focuses on rule generators handling DIS ψ, however we have lots of tables with missing values like the Mammographic data set, the Congressional Voting data set, the Hepatitis data set, etc. We see such a table as a *Non-deterministic Information System* (NIS).

In NIS Φ, we usually apply the discretization procedure and handle a finite number of possible values. By replacing each ? symbol with a possible value, we have a standard table, which we name a *derived DIS*. Let $DD(\Phi)$ denote the set of all derived DISs from NIS Φ. In rule generation, we employ the usual definition of a rule in DIS [10] and extend it to a certain rule and a possible rule in NIS below [14, 17]:

(A certain rule in NIS) An implication τ is a *certain rule*, if τ is a rule in each derived DIS for given α and β.

(A possible rule in NIS) An implication τ is a *possible rule*, if τ is a rule in at least one derived DIS for given α and β.

If τ is a certain rule, we can conclude τ is also a rule in the unknown actual DIS ψ^{actual}. (We see there is one derived DIS $\psi^{actual} \in DD(\Phi)$ which contains the actual values.) This property is also described in Lipski's incomplete information databases [7]. In DIS, the same set of rules are defined by two definitions, so two definitions seem to be a natural extension from rules in DIS. However, the number of $DD(\Phi)$ increases exponentially, and there are more than 10^{100} derived DISs for the Congressional Voting data set. It will be hard to examine the certain rules and the possible rules by checking each derived DIS sequentially. For this problem, we afford a solution by showing some properties on rules [14, 17]. In [18], some examples are uploaded.

We are now proposing the new framework related to NIS and missing values [17]. We focused on how we obtain rules in DIS and NIS in the previous research. Since we gave a solution for rule generation in DIS and NIS, we think that it will be able to cope with next new topics in Figure 4. The ② and ③ consider the estimation of uncertain values by using the machine learning strategy, and the ④ aims the dilution of information. We intentionally change deterministic information to non-deterministic information with preserving some constraints. This strategy seems to be similar to the pay-per-view scrambled broadcasting. The ⑤ tries to apply NIS to privacy preserving system. We agree with that each respondent answers two choices for questionnaire consisting of multiple choice question. Then, we have NIS instead of DIS, and some personal answers are diluted for privacy preserving. In our RNIA, we can obtain certain rules and possible rules for knowing the properties of data sets. Namely, a privacy preserving questionnaire system can be realized by using RNIA [16].

6. Apriori-Based Rule Generation and Rough Set-Based Rule Generation

This section compares Apriori-based rule generation with rough set-based rule generation. In rough sets, the following property takes the important role.

Proposition 4. [10] In DIS and an implication $\tau : \wedge_{A \in CON}[A, val_A] \Rightarrow [Dec, val]$, the following (1) and (2) are equivalent:

(1) $accuracy(\tau)$=1,

(2) $OBJ(\wedge_{A \in CON}[A, val_A]) \subseteq OBJ([Dec, val])$.

At the beginning, (2) in Proposition 4 are employed for detecting the implication τ satisfying $accuracy(\tau)$=1. Since the condition $accuracy(\tau)$=1 is too strict, the research to handle $accuracy(\tau) \geq \beta$ has been investigated. Due to this historical background, rough set-based rule generation generally takes the next strategy.

(Rough set-based rule generation)

(1) For one descriptor $[Dec, val]$, we fix a target set X= $OBJ([Dec, val])$ and two threshold values α and β.

(2) After finding a set LOW $(=OBJ(\wedge_{A \in ATTL}[A, val_A]))$ for $ATTL \subset AT$, an implication $\tau : \wedge_{A \in ATTL}[A, val_A] \Rightarrow [Dec, val]$ with $accuracy(\tau)$=1 is obtained as a side effect. We recognize τ is a rule, if $support(\tau) \geq \alpha$.

(3) After finding a set UPP $(=OBJ(\wedge_{A \in ATTU}[A, val_A]))$ for $ATTU \subset AT$, an implication $\tau : \wedge_{A \in ATTU}[A, val_A] \Rightarrow [Dec, val]$ is obtained similarly. We recognize τ is a rule,

if τ satisfies $support(\tau) \geq \alpha$ and $accuracy(\tau) \geq \beta$.

In rough sets, the most important issue is how we find two sets LOW and UPP. The problem of rule generation is converted to the problem of finding two appropriate sets LOW and UPP. Here, Propositions 3 and 4 take the important role.

Remark 3. In rough set-based rule generation, it may be easy to find one LOW and one UPP by using equivalence classes. However, to find all LOW and all UPP may not be easy. Skowron showed that the problem to find all LOW is NP-hard [21]. Since each rule is connected with one LOW or UPP, we may miss some rules if we do not find some LOW or UPP. The completeness in Theorem 1 may not be assured. In order to solve this problem, Skowron introduced the discernibility function method [21]. Even though the use of the equivalence classes is effective, some specific methods are necessary for preserving the completeness of rule generation.

On the other hand, the original Apriori algorithm does not consider decision attribute Dec. There is no concept of decision attribute. In order to handle table data sets, we adjusted the Apriori algorithm and introduced decision attribute Dec in Algorithm 1. Apriori-based rule generation does not consider a target set X defined by the descriptor $[Dec, val]$. This is the difference between Apriori-based rule generation and rough set-based rule generation. Apriori-based rule generation generates all implications τ satisfying $support(\tau) \geq \alpha$ and $accuracy(\tau) \geq \beta$ for decision attribute Dec, and rough set-based rule generation does all implications τ satisfying the same condition for the descriptor $[Dec, val]$.

7. Concluding Remarks

This paper clarified the properties of Apriori in Prolog and Apriori in SQL, and described the difference between Apriori-based rule generation and rough set-based rule generation. Two generators have their properties, and two types of rule generation also have their properties. For handling several types of tables, it is necessary to consider which generator and which rule generation are suitable for existing data sets. We think the following:

(1) For large data sets, generally rough set-based rule generation will reduce the execution time, because the decision part of rules is predefined.

(2) Apriori-based rule generation seems to include the functionality of rough set-based rule generation. Namely, rules defined by a set of descriptors $\{[Dec, val] \mid val \in VAL_{Dec}\}$ are generated in one rule generation. Rough set-based rule generation repeats rule generation for each descriptor $[Dec, val]$.

(3) Apriori in Prolog may have constraint on PC environment for execution, but the use of the equivalence classes is effective. So, Apriori in Prolog will be better for small size data sets (the object size is about less than 1000). On the other hand, the description of data set may not be easy.

(4) Apriori in SQL will be applicable to large size data sets, and PC environment for execution is stable, for example Xampp system supports PC environment. Even though it is necessary to prepare one procedure to translate a csv data set to rdf data set, we can uniformly handle any csv data set after this translation.

In this paper, we handled Table 1 for describing the frameworks, however the conclusion in this paper depends upon the actual execution of data sets, like the Mammographic data set, the Lenses data set, the Car evaluation data set, the Credit approval data set, the Congressional voting data set, the Hepatitis data set, etc. [18]. It is necessary to consider the effective use of the several proposed frameworks toward the intelligent analysis of table data sets.

Acknowledgments The authors are grateful to the anonymous reviewers and Professor Junzo Watada (Waseda University and Universiti Teknologi PETRONAS, Malaysia) for their useful comments. This paper is an extended version of a paper presented at FIM2017.

This work is supported by the Grant-in-Aid for Scientific Research (C) (No.26330277), Japan Society for the Promotion of Science.

References:

[1] Agrawal, R., Srikant, R.: Fast algorithms for mining association rules in large databases, *Proc. VLDB'94*, pp. 487–499, 1994.

[2] Bazan, J., Szczuka, M.: The rough set exploration system, *Transactions on Rough Sets*, **3**, pp. 37–56, 2005.

[3] Ceglar, A., Roddick, J.F.: Association mining, *ACM Computing Survey*, **38**(2), 2006.

[4] Frank, A., Asuncion, A.: UCI Machine Learning Repository, Irvine, CA: University of California, School of Information and Computer Science, 2010. `http://mlearn.ics.uci.edu/MLRepository.html`

[5] Grzymała-Busse, J.W., Werbrouck, P.: On the best search method in the LEM1 and LEM2 algorithms, *in: Incomplete Information: Rough Set Analysis, Studies in Fuzziness and Soft Computing* **13**, pp. 75–91, 1998.

[6] Kowalski, M., Stawicki, S.: SQL-based heuristics for selected KDD tasks over large data sets, *In: Proc. FedCSIS 2012*, pp. 303–310, 2012.

[7] Lipski, W.: On semantic issues connected with incomplete information databases, *ACM Transactions on Database Systems*, **4**(3), pp. 262–296, 1979.

[8] Nakata, M., Sakai, H.: Twofold rough approximations under incomplete information, *International Journal of General Systems*, **42**(6), pp. 546–571, 2013.

[9] Orłowska, E., Pawlak, Z.: Representation of nondeterministic information, *Theoretical Computer Science*, **29**(1-2), pp. 27–39, 1984.

[10] Pawlak, Z.: *Rough Sets: Theoretical Aspects of Reasoning About Data*, Kluwer Academic Publishers, 1991.

[11] phpMyAdmin Web Page `http://www.phpmyadmin.net/`, 2016.

[12] Riza, L.S. et al.: Implementing algorithms of rough set theory and fuzzy rough set theory in the R package RoughSets, *Information Sciences*, **287**(10), pp. 68–89, 2014.

[13] Sakai, H., Nakata, M., Ślęzak, D.: A NIS-Apriori based rule generator in Prolog and its functionality for table data, *in: Proc. RSKT 2011*, Springer LNAI, **6954**, pp. 226-231, 2011.

[14] Sakai, H., Wu, M., Nakata, M.: Apriori-based rule generation in incomplete information databases and non-deterministic information systems, *Fundamenta Informaticae*, **130**(3), pp. 343–376, 2014.

[15] Sakai, H., Liu, C., Zhu, X., Nakata M.: On NIS-Apriori based data mining in SQL, *in: Proc. Int'l. Conf. on Rough Sets*, LNCS **9920**, Springer, pp. 514–524, 2016.

[16] Sakai, H., Liu, C., Nakata, M., Tsumoto, S.: A proposal of a privacy-preserving questionnaire by non-deterministic information and its analysis, *in: Proc. IEEE Big Data Conference*, pp. 1956-1965, 2016.

[17] Sakai, H., Nakata, M., Yao, Y.: Pawlak's many valued information system, non-deterministic information system, and a proposal of new topics on information incompleteness toward the actual application, *Studies in Computational Intelligence*, **708**, Springer, pp. 187-204, 2017.

[18] Sakai, H.: Execution logs by RNIA software tools, 2016. http://www.mns.kyutech.ac.jp/~sakai/RNIA

[19] Shen, K.Y., Tzeng, G.H.: Contextual improvement planning by fuzzy-rough machine learning: A novel bipolar approach for business analytics, *International Journal of Fuzzy Systems* **18**(6), pp. 940–955, 2016.

[20] Shen, K.Y., Sakai, H., Tzeng, G.H.: Stable rules evaluation for rough-set-based bipolar model: A preliminary study for credit loan evaluation, *in: Proc. Int'l. Conf. on Rough Sets*, LNCS **10313**, pp. 317–328, 2017.

[21] Skowron, A., Rauszer, C.: The discernibility matrices and functions in information systems, *Intelligent Decision Support - Handbook of Advances and Applications of the Rough Set Theory*, pp. 331–362, Kluwer Academic Publishers, 1992.

[22] Ślęzak, D., Sakai H.: Automatic extraction of decision rules from non-deterministic data systems: Theoretical foundations and SQL-based implementation, *in: Database Theory and Application*, CCIS **64**, Springer, pp. 151–162, 2009.

[23] Swieboda, W., Nguyen, S.: Rough set methods for large and spare data in EAV format, *in: Proc. IEEE RIVF 2012*, pp. 1–6, 2012.

[24] Wu, M., Nakata, M., Sakai, H.: An overview of the *getRNIA* system for non-deterministic data, *Procedia Computer Science*, **22**, pp. 615-62, Elsevier, 2013.

Name:
Hiroshi Sakai

Affiliation:
Graduate School of Engineering, Kyushu Institute of Technology

Address:
Tobata, Kitakyushu 804-8550, Japan

Brief Biographical History:
Hiroshi Sakai received B.S., M.S. and D.S. degrees in applied mathematics and computer science from Kyushu University, in 1982, 1984 and 1988. He is a professor of Graduate School of Engineering, Kyushu Institute of Technology.

Main Works:
• Rough non-deterministic information analysis for uncertain information, The Handbook on Reasoning-Based Intelligent Systems, pp. 81–118, World Scientific, 2013.
• Apriori-based rule generation in incomplete information databases and non-deterministic information systems, Fundamenta Informaticae, Vol. 130, No.3, pp.343-376, 2014.

Membership in Learned Societies:
• Japan Society for Fuzzy Theory and Intelligent Informatics
• International Rough Set Society (Senior member)
• The Mathematical Society of Japan

Name:
Kao-Yi Shen

Affiliation:
Department of Banking & Finance, Chinese Culture University (SCE)

Address:
Da'an Dist., Taipei City Taiwan, R.O.C

Brief Biographical History:
Kao-Yi Shen received his Bachelor and Master degrees in Industrial Engineering from the Tunghai University; in 2002, he received the Ph.D. degree in Business Administration from the Chengchi University in Taiwan. He worked as a senior analyst in the venture capital industry as well as a Marketing Manager and Head of Project Management in a Taiwan-based international IT company. He is an Associate Professor of Department of Banking & Finance, Chinese Culture University (SCE) in Taipei.

Main Works:
• Financial modeling and improvement planning for the life insurance industry by using a rough knowledge based hybrid MCDM model, Information Sciences, Vol. 375, pp. 296–313, 2017.
• New concepts and trends of hybrid multiple criteria decision making, CRC Press, New York, US, 2017, ISBN: 978-1-4987-7708-7.

Membership in Learned Societies:
• Taiwan Fuzzy System Associations
• International Rough Set Society
• International MCDM Society

Name:
Michinori Nakata

Affiliation:
Faculty of Management & Information Science, Josai International University

Address:
1, Gumyo, Togane, Chiba, 283-0003, Japan

Brief Biographical History:
1985-1990 Assistant Professor, Chiba-Keizai College, 1991-1998 Associate Professor, Chiba-Keizai College, 1999-2000 Professor, Chiba-Keizai College, 2001- Professor, Josai International University

Main Works:
• Applying rough sets to information tables containing possibilistic values, Trans. Computational Science, Vol. 2, pp. 180–204, 2008.
• Twofold rough approximations under incomplete information, International Journal of General Systems, Vol. 42(6), pp. 546–571, 2013.

Membership in Learned Societies:
• Japan Society for Fuzzy Theory and Intelligent Informatics(SOFT)
• American Association of Artificial Intelligence(AAAI)
• Association for Computing Machinery(ACM)
• The Institute of Electrical and Electronics Engineers(IEEE)