BULGARIAN ACADEMY OF SCIENCES

CYBERNETICS AND INFORMATION TECHNOLOGIES • Volume **13**, No 1

Sofia • 2013 Print ISSN: 1311-9702; Online ISSN: 1314-4081
DOI: 10.2478/cait-2013-0007

A Method for Extracting Knowledge from Decision Tables in Terms of Functional Dependencies

Vu Duc Thi, Nguyen Long Giang

Institute of Information Technology, VAST, Viet Nam
Emails: vdthi@ioit.ac.vn nlgiang@ioit.ac.vn

Abstract: *The problem of extracting knowledge from decision tables in terms of functional dependencies is one of the important problems in knowledge discovery and data mining. Based on some results in relational database, in this paper we propose two algorithms. The first one is to find all reducts of a consistent decision table. The second is to infer functional dependencies from a consistent decision table. The second algorithm is based on the result of the first. We show that the time complexity of the two algorithms proposed is exponential in the number of attributes in the worst case.*

Keywords: *Relational database, functional dependency, rough set theory, decision table, reduct.*

1. Introduction

In rough set theory, for the given consistent decision table $\mathrm{DS} = \left(U, A \cup \{d\}\right)$, an attribute set R is called a *reduct* of A if R is a minimal set, which satisfies the functional dependency $R \to \{d\}$. Consequently, the problem of determining all functional dependencies $R_i \to \{d\}$ where $R_i \subseteq A$ can be solved by searching all reducts of the decision table *DS*. This problem can be considered as the problem of extracting all possible knowledge from a given consistent decision table in terms of functional dependencies. This is one of the important problems in the field of knowledge discovery and data mining.

In relational database theory, if $\mathcal{R}$ is a relation over an attribute set A then an attribute set B is called a *minimal set* of an attribute $a \in A$ if B is a minimal attribute set, which satisfies the functional dependency $B \to \{a\}$. If any consistent decision table $\mathrm{DS} = (U, A \cup \{d\})$ can be treated as a relation $\mathcal{R} = \{u_1, ..., u_n\}$ over the set of all attributes $A \cup \{d\}$, then the concept of *reduct* in *DS* is equivalent to the concept of a *minimal set* of the attribute *d* over $\mathcal{R}$. Consequently, the problem of searching all reducts of a consistent decision table can be solved by using some results concerning the minimal sets of an attribute in relational database theory.

In this paper we propose two algorithms. The first one is to find all reducts of a consistent decision table. The second is to infer functional dependencies from a consistent decision table. The second algorithm is based on the result of the first. The two algorithms are constructed based on some results about Sperner system and minimal sets of an attribute of J. D e m e t r o v i c s and V u D u c T h i [1, 2, 4]. We show that the time complexity of the two algorithms is exponential in the number of attributes in the worst case. However, the proposed algorithms are effective in some special cases.

The paper is structured as follows. Section 2 presents some basic concepts in relational database and rough set theory. Section 3 proposes an algorithm to search all reducts of a consistent decision table. Section 4 proposes an algorithm to construct a relation scheme from a consistent decision table based on the algorithm in Section 3. The conclusions are presented in the last section.

2. Basic concepts

2.1. Basic concepts of rough set theory

In this section we introduce some basic concepts in rough set theory [7, 8, 9].

An information system is a pair $\mathrm{IS} = (U, A)$, where the set U denotes the *universe of objects* and A is the set of *attributes*, i.e., mappings of the form: $a: U \to V_a$; V_a is called the value set of attribute *a*. A decision table is an information system $\mathrm{DS} = (U, A \cup \{d\})$ where *d* is a distinguished *decision attribute*. The remaining attributes are called *conditions* or *conditional attributes.*

Every attribute subset $B \subseteq A$ determines a binary discernibility relation $\mathrm{IND}(B)$:

$$\mathrm{IND}(B) = \{(u,v) \in U \times U \mid \forall a \in B,\ a(u) = a(v)\},$$

$\mathrm{IND}(B)$ determines a partition of U which is denoted by U/B. Any element $[x]_B = \{y \in U \mid (x,y) \in \mathrm{IND}(B)\}$ in U/B is called equivalent class. For $B \subseteq A$ and $X \subseteq U$, *B-upper approximation* of *X* is the set $\overline{B}X = \{u \in U \mid [u]_B \cap X \neq \varnothing\}$, *B-lower approximation* of X is the set $\underline{B}X = \left\{ u \in U \left| [u]_B \subseteq X \right. \right\}$, *B-boundary* of X is the set $BN_B(X) = \overline{B}X \setminus \underline{B}X$ and *B-positive region* of $\{d\}$ is the set $\mathrm{POS}_B(\{d\}) = \bigcup_{X \in U/\{d\}} (\underline{B}X)$. A decision table *DS* is consistent iif $\mathrm{POS}_A(\{d\}) = U$, in other words the functional dependency $A \to \{d\}$ is true. Conversely, DS is inconsistent, and then $\mathrm{POS}_A(\{d\})$is the maximum subset of U that the functional dependency $A \to \{d\}$ is true**.**

One of the crucial concepts in rough set theory is *reducts* or *decision reducts* [8]. In general, reducts are minimal subsets (with respect to the set inclusion relation) of the attributes which contain a necessary portion of information about the set of all attributes.

Definition 1. [8] Let $\mathrm{DS} = (U,\ A \cup \{d\})$be a decision table. If $R \subseteq A$ satisfies:

1) $\mathrm{POS}_R(\{d\}) = \mathrm{POS}_A(\{d\})$,

2) $\forall r \in R$, $\mathrm{POS}_{R-\{r\}}(\{d\}) \neq \mathrm{POS}_A(\{d\})$,

then *R* is called a *reduct* of DS.

If DS is a consistent decision table, it is known from the above definition that *R* is a reduct of *A* if *R* satisfies $R \to \{d\}$ and $\forall R' \subset R$, $R' \not\to \{d\}$. Let RED(DS) be the set of all reducts of DS.

2.2. Basic concepts and algorithms of the relational database

Let us give some necessary definitions and results of the theory of relation database that can be found in [1-6].

Let $A = \{a_1, ..., a_k\}$ be a finite set of attributes and let $D(a_i)$ be the set of all possible values of the attribute a_i, for $i = 1, ..., k$. Any subset of the Cartesian product

$$\mathcal{R} \subseteq D(a_1) \times D(a_2) \times ... \times D(a_k)$$

is called the relation over *A*. In other words, relation over *A* is the set of tuples $\{h_1, ..., h_m\}$ where $h_j : A \to \bigcup_{a_i \in A} D(a_i), 1 \leq j \leq n,$ is a function that $h_j(a_i) \in D(a_i)$.

Let $\mathcal{R} = \{h_1, ..., h_n\}$ be a given relation over $A = \{a_1, ..., a_k\}$. Any pair of attribute sets $B, C \subseteq A$ is called functional dependency (FD for short) over *A*, and denoted by $B \to C$, if and only if for any pair of tuples $h_i, h_j \in \mathcal{R}$:

$$\forall_{a \in B} \left(h_i(a) = h_j(a) \right) \Rightarrow \forall_{a \in C} \left(h_i(a) = h_j(a) \right).$$

The set $\mathcal{F}_{\mathcal{R}} = \{(B, C) : B, C \subset A; B \to C\}$is called a full family of functional dependencies in $\mathcal{R}$.

Let $\mathbb{P}(A)$ be the power set of attribute set *A*. A family $\mathcal{R} \subset \mathbb{P}(A) \times \mathbb{P}(A)$ is called an *f-family* over *A* if and only if for all subsets of attributes $P, Q, S, T \subseteq A$, the following properties hold:

$\text{R1}: (P, P) \in \mathcal{R}$;

$\text{R2}: (P, Q) \in \mathcal{R}, (Q, S) \in \mathcal{R} \Rightarrow (P, S) \in \mathcal{R}$;

$\text{R3}: (P, Q) \in \mathcal{R}, P \subseteq S, T \subseteq Q \Rightarrow (S, T) \in \mathcal{R}$;

$\text{R4}: (P, Q) \in \mathcal{R}, (R, S) \in \mathcal{R} \Rightarrow (P \cup R, Q \cup T) \in \mathcal{R}$.

Clearly $\mathcal{F}_{\mathcal{R}}$ is an *f*-family over *A*. It is also known that if $\mathcal{F}$ is an *f*-family over *A*, then there is a relation $\mathcal{R}$ such that $\mathcal{F}_{\mathcal{R}} = \mathcal{F}$.

A pair $\mathbb{S} = (A, \mathcal{F})$, where *A* is a set of attributes and $\mathcal{F}$ is a set of FDs on *A*, is called a *relation scheme*. Let us denote by $\mathcal{F}^+$ the set of all FDs, which can be derived from $\mathcal{F}$ by using the rules R1-R4. For any $B \subseteq A$, the set $B^+ = \{a \in A : B \to a \in \mathcal{F}^+\}$ is called the *closure* of *B* on $\mathbb{S}$. It is clear that $B \to C \in \mathcal{F}^+$ if and only if $C \subseteq B^+$.

Recall that a family $\mathcal{K} \in \mathbb{P}(A)$ is a Sperner system if for any $K_1, K_2 \in \mathcal{K}$ implies $K_1 \not\subseteq K_2$. Let $\mathcal{K}$ be a Sperner system. We defined the set $\mathcal{K}^{-1}$, as follows:

$$\mathcal{K}^{-1} = \left\{B \subseteq A : \left(\forall_{C \in \mathcal{K}} C \not\subseteq B\right) \wedge \left(B \subseteq D \to \exists_{C \in \mathcal{K}} C \subseteq D\right)\right\}.$$

It is easy to see that $\mathcal{K}^{-1}$ is the set of subsets of *A*, which does not contain the elements of $\mathcal{K}$ and which is maximal for this property. Clearly, $\mathcal{K}^{-1}$ is also a Sperner system.

For the given relation $\mathcal{R} = \{h_1, \ldots, h_n\}$ over $A = \{a_1, \ldots, a_k\}$, let $\mathbb{E}(\mathcal{R}) = \{E_{ij} : 1 \le i < j \le n\}$, where $E_{ij} = \{a \in A : h_i(a) = h_j(a)\}$. Thus, it is called the *equality set* of $\mathcal{R}$. It is known [1] that for each subset of attributes $B \subseteq A$, the following property holds:

$$B^+ = \begin{cases} \bigcup_{i,j=1}^{n} E_{ij} & \text{if } B \subseteq E_{ij} \text{ for some } E_{ij} \in \mathbb{E}(\mathcal{R}); \\ A & \text{otherwise.} \end{cases}$$

In the next content we introduce some definitions about the family of all minimal sets of an attribute over a relation and a relation scheme.

Definition 2. [4] Let $\mathbb{S} = (A, \mathcal{F})$ be a relation scheme over *A*. For any attribute $a \in A$, the set $K_{\mathbb{S}}(a) = \{B \subseteq A : B \to a \wedge \nexists_{C \subset B} C \to a\}$ is called *a family of all minimal sets of the attribute a over* $\mathbb{S}$.

Similarly, we define the family of minimal sets of an attribute over a relation

Definition 3. Let $\mathcal{R}$ be a relation over *A*. For any attribute $a \in A,$ the set $K_{\mathcal{R}}(a) = \left\{B \subseteq A : B \to a \wedge \nexists_{C \subset B} C \to a\right\}$ is called *a family of all minimal sets of the attribute a over* $\mathcal{R}$.

It is known that $A \notin K_{\mathcal{S}}(a), A \notin K_{\mathcal{R}}(a),\ a \in K_{\mathcal{S}}(a), a \in K_{\mathcal{R}}(a)$ and $K_{\mathcal{S}}(a), K_{\mathcal{R}}(a)$ are Sperner systems over *A*.

In the relational database theory, two algorithms which are related to Sperner system have been proposed [1, 2].

Algorithm 1. [1] Finding $\mathcal{K}^{-1}$ from a given Sperner-system $\mathcal{K}$.

Input: Let $\mathcal{K} = \{B_1, ..., B_m\}$ be a Sperner-system over *A*.

Output: $\mathcal{K}^{-1}$

Step 1. We set $\mathcal{K}_1 = \left\{R - \{a\} : a \in B_1\right\}$. It is obvious that $\mathcal{K}_1 = \{B_1\}^{-1}$.

Step *q*+1. (*q* < *m*). Assume that $\mathcal{K}_q = F_q \cup \left\{X_1, ..., X_{tq}\right\}$, where $X_1, ..., X_{tq}$ are elements of $\mathcal{K}_q$ containing B_{q+1} and $F_q = \left\{A \in \mathcal{K}_q : B_{q+1} \not\subseteq A\right\}$. For all *i* , *i* = 1, …, t_q , we calculate $\left\{B_{q+1}\right\}^{-1}$ on X_i in an analogous way as $\mathcal{K}_1$, which are the maximal subsets of X_i not containing B_{q+1}. We denote them by $A_1^i, ..., A_{ri}^i$. Let:

$$\mathcal{K}_{q+1} = F_q \cup \left\{A_p^i : A \in F_q \Rightarrow A_p^i \not\subset A,\ 1 \le i \le t_q,\ 1 \le p \le r_i\right\}.$$

Finally, let $\mathcal{K}^{-1} = \mathcal{K}_m$.

Suppose that I_q is the number of elements of $\mathcal{K}_q$. According to [1], the time complexity of Algorithm 1 in the worst case is $O\left(|R|^2 \sum_{q=1}^{m-1} t_q u_q\right)$, where $u_q = I_q - t_q$ if $I_q > t_q$ and u_q = 1 if $I_q = t_q$. This time complexity cannot be more than exponential in the number of attributes. In cases, for which $I_q \le I_m \quad \forall q : 1 \le q \le m-1,$ the time complexity of Algorithm 1 is not greater than $O\left(|R|^2 |\mathcal{K}| \left|\mathcal{K}^{-1}\right|^2\right)$. Thus, in these cases the algorithm finds $\mathcal{K}^{-1}$ in polynomial time in $|R|, |\mathcal{K}|$ and $|\mathcal{K}|^{-1}$. Especially, when $|\mathcal{K}|, |\mathcal{K}|^{-1}$ is small, Algorithm 1 is effective.

Algorithm 2 [2]. Finding a set $D \in \mathcal{K}$ from $\mathcal{K}^{-1}$.

Input: Let $\mathcal{K}^{-1}$ be a Sperner-system over *A*, $C = \{b_1, ..., b_m\} \subseteq A$ such that $\exists B \in \mathcal{K}^{-1} : B \subseteq C.$

Output: $D \in \mathcal{K}$.

Step 1. We set $T(0) = C$.

Step i+1. We set

$T(i+1) = T(i) - b_{i+1}$ if $\forall B \in \mathcal{K}^{-1}$, there is not $T \subseteq B$,

$T(i+1)=T(i)$ otherwise.

Finally, we set $D=T(m)$.

Algorithm 3 [2]. Finding $\mathcal{K}$ from $\mathcal{K}^{-1}$

Input: Let $\mathcal{K}^{-1}=\{B_1,...,B_m\}$ be a Sperner-system over *A*.

Output: $\mathcal{K}$.

Step 1. Using Algorithm 2, we construct A_1. We set $\mathcal{H}_1=A_1$.

Step *i*+1. If there is a $B\in\mathcal{H}_i^{-1}$ such that $B\not\subseteq B_j\ (\forall j:1\le j\le m)$, then using Algorithm 2 we calculate A_{i+1}, where $A_{i+1}\in\mathcal{K}$, $A_{i+1}\subseteq B$. We set $\mathcal{H}_{i+1}=\mathcal{H}_i\cup A_{i+1}$. In the converse case we set $\mathcal{K}=\mathcal{H}_i$.

According to [2], the time complexity of Algorithm 3 is $O\left(|R|\left(\sum_{q=1}^{|\mathcal{K}|-1}\left(\left|\mathcal{K}^{-1}\right|I_q+|R|t_qu_q\right)+\left|\mathcal{K}^{-1}\right|^2+|R|\right)\right)$ where I_q,t_q,u_q are defined as in Algorithm 1. Consequently, the worst-case time of Algorithm 3 cannot be more than exponential in the number of attributes. In cases, for which $I_q\le\left|\mathcal{K}^{-1}\right|,\ q=1,...,m-1,$ the time complexity of Algorithm 3 is $O\left(|R|^2\left|\mathcal{K}^{-1}\right|^2|\mathcal{K}|\right)$, this complexity is polynomial in $|R|,\left|\mathcal{K}^{-1}\right|$ and $|\mathcal{K}|$. Clearly, if $|\mathcal{K}|$ is polynomial in $|R|,\left|\mathcal{K}^{-1}\right|$ then the algorithm is effective. Especially, when $|\mathcal{K}|$ is small.

3. An algorithm for searching all reducts of a consistent decision table

Let $\mathrm{DS}=(U,A\cup\{d\})$ be a consistent decision table, it is known from the above Definiton 1 and 3 that $R\subseteq A$ is a reduct of A if $\mathrm{POS}_R(\{d\})=\mathrm{POS}_A(\{d\})=U$ (or $R\to\{d\}$) and for $\forall R'\subset R$, $\mathrm{POS}_{R'}(\{d\})\ne\mathrm{POS}_A(\{d\})$ (or $R'\not\to\{d\}$). Hence, the problem of finding the set of all reducts of A in the consistent decision table $\mathrm{DS}=(U,A\cup\{d\})$ where $U=\{u_1,u_2,...,u_n\}$ becomes the problem of finding the family of all minimal sets of the attribute {d} for the relation $\mathcal{R}=\{u_1,u_2,...,u_n\}$ over the attribute set $A\cup\{d\}$. Denote $\mathrm{RED}(\mathrm{DS})$ as the set of all reducts of DS, then we have $\mathrm{RED}(\mathrm{DS})=K_{\mathcal{R}}(d)-\{d\}$ where $K_{\mathcal{R}}(d)$ is the family of all minimal sets of the attribute {*d*} over $\mathcal{R}$.

Algorithm 4. Finding the set of all reducts in a consistent decision table.

Input: Let $\mathrm{DS}=(U,A\cup\{d\})$ be a decision table, where $\mathrm{POS}_A(\{d\})=U$, $A=\{a_1,a_2,...,a_n\}$, $U=\{u_1,u_2,...,u_m\}$.

Output: $\mathrm{RED}(\mathrm{DS})$.

Let us consider the decision table DS as the relation $\mathcal{R} = \{u_1, u_2, ..., u_m\}$ over the set of attributes $R = A \cup \{d\}$.

Step 1. From $\mathcal{R}$, we construct the equality system $\mathcal{E}_r = \{E_{ij} : 1 \le i < j \le m\}$ where $E_{ij} = \{a \in R : u_i(a) = u_j(a)\}$.

Step 2. From $\mathcal{E}_r$, we construct

$$\mathcal{M}_d = \{A \in \mathcal{E}_r : d \notin A \ \nexists B \in \mathcal{E}_r : d \notin B, A \subset B\}.$$

Step 3. Using Algorithm 3, we calculate $\mathcal{K}$ from $\mathcal{M}_d$ $\left(\mathcal{M}_d = \mathcal{K}^{-1}\right)$.

Step 4. We set $\mathrm{RED}(\mathrm{DS}) = \mathcal{K} - \{d\}$.

According to the method to construct $\mathcal{M}_d$ at Step 2 and the method to calculate the closure of an attribute set over a relation, $\forall A \in \mathcal{M}_d$ we have $A^+ = A$ and *A* does not contain *d*, so A^+ does not contain *d*, then $A \to \{d\} \notin \mathcal{F}^+$. Moreover, if there exists *B* such that $A \subset B$ then there are two cases: (1) if *B* does not contain *d*, then $B^+ = R$; (2) if *B* contains *d* then obviously B^+ contains *d*. For both cases we have $B \to \{d\} \in \mathcal{F}^+$. Hence, $\mathcal{M}_d = \mathrm{MAX}\left(\mathcal{F}^+, d\right)$ where $\mathrm{MAX}\left(\mathcal{F}^+, d\right) = \{A \subseteq R : A \to \{d\} \notin \mathcal{F}^+, A \subset B \Rightarrow B \to \{d\} \in \mathcal{F}^+\}$. According to [4], $\mathrm{MAX}\left(\mathcal{F}^+, d\right) = \left(K_{\mathcal{R}}(d)\right)^{-1}$ where $K_{\mathcal{R}}(d)$ is the Sperner-system over *R* as the family of all minimal sets of the attribute *d*, so $\mathcal{M}_d = \left(K_{\mathcal{R}}(d)\right)^{-1}$. Consequently, at Step 3, $\mathcal{K} = K_{\mathcal{R}}(d)$ is the family of all minimal sets of the attribute *d*. At Step 4, $\mathrm{RED}(\mathrm{DS}) = \mathcal{K} - \{d\}$ is the set of all reducts of DS.

It is easy to see that the time complexity of Step 1 and Step 2 is polynomial in the size of $\mathcal{R}$. So the time complexity of Algorithm 4 is the same as that of Algorithm 3 at Step 3. Consequently, the worst-case time of Algorithm 4 cannot be more than exponential in the number of attributes. As Algorithm 3, Algorithm 4 is also effective in some special cases.

Example 1. Let us consider the consistent decision table $\mathrm{DS} = (U, A \cup \{d\})$, where $U = \{u_1, u_2, u_3, u_4, u_5, u_6, u_7\}$, $A = \{a, b, c\}$ (Table 1).

Table 1. An example decision table

U	*a*	*b*	*c*	*d*
u_1	6	6	0	6
u_2	0	2	2	0
u_3	0	0	0	0
u_4	0	0	3	0
u_5	4	4	0	0
u_6	5	0	5	5
u_7	1	0	0	0

We perform the following steps of Algorithm 4 to calculate $\mathrm{RED}(\mathrm{DS})$.

Step 1. We calculate

$\mathcal{E}_r = \{\{a, b, d\}, \{b, c, d\}, \{a, d\}, \{b, d\}, \{c, d\}, \{b\}, \{c\}, \{d\}\}$.

Step 2. We calculate $\mathcal{M}_d = \{\{b\}, \{c\}\}$.

Step 3. Using Algorithm 3 we calculate $\mathcal{K}$ from $\mathcal{K}^{-1}$ $\left(\mathcal{K}^{-1} = \mathcal{M}_d\right)$.

We have $B_1 = \{b\}, B_2 = \{c\}$.

Using Algorithm 2 we calculate $A_1 = \mathcal{K}_1 = \{d\}$. Using Algorithm 1 we calculate $\mathcal{K}_1^{-1} = \{d\}^{-1} = \{a, b, c\}$.

We choose $B = \{a, b, c\}, B \not\subseteq B_1, B \not\subseteq B_2$. Using Algorithm 2 we calculate $A_2 = \{b, c\}$. So $\mathcal{K}_2 = \{\{d\}, \{b, c\}\}$. Using Algorithm 1 we calculate $\mathcal{K}_2^{-1} = \{\{a, b\}, \{a, c\}\}$.

It is known that $\{a, b\} \not\subseteq B_1$ and $\{a, b\} \not\subseteq B_2$, so we choose $B = \{a, b\}$. Using Algorithm 2 we calculate $A_3 = \{a\}$. So $\mathcal{K}_3 = \{\{d\}, \{b, c\}, \{a\}\}$. Using Algorithm 1 we calculate $\mathcal{K}_3^{-1} = \{\{b\}, \{c\}\}$.

Clearly, $\forall B \in \mathcal{K}_3^{-1}, B \in B_1$ or $B \in B_2$, the algorithm stops and $\mathcal{K} = \mathcal{K}_3 = \{\{d\}, \{b, c\}, \{a\}\}$.

Step 4. $\mathrm{RED}(\mathrm{DS}) = \mathcal{K} - \{d\} = \{\{a\}, \{b, c\}\}$.

4. An algorithm for inferring functional dependencies from a consistent decision table

Given a consistent decision table $\mathrm{DS} = (U, A \cup \{d\})$ as a relation $\mathcal{R}$ over an attribute $R = A \cup \{d\}$. In this section we propose an algorithm to construct the relation scheme $\mathbb{S}_d = (R, \mathcal{F})$, where $\mathcal{F}$ is the set of functional dependencies $A_i \to \{d\},\ A_i \subseteq A,\ 1 \le i \le t$, such that $K_{\mathbb{S}}(d) = \mathrm{RED}(\mathrm{DS}) \cup \{d\}$, where $K_{\mathbb{S}}(d)$ is the family of all minimal sets of the attribute *d* over $\mathbb{S}_d$ and $\mathrm{RED}(\mathrm{DS})$ is the set of all reducts of DS.

Algorithm 5. Inferring functional dependencies from a consistent decision table.

Input: Let $\mathrm{DS} = (U, A \cup \{d\})$ be a decision table, where $\mathrm{POS}_A(\{d\}) = U$.

Output: $\mathbb{S}_d = (R, \mathcal{F})$ such that $K_{\mathbb{S}}(d) = \mathrm{RED}(\mathrm{DS}) \cup \{d\}$.

Step 1. Using Algorithm 4 we obtain $\text{RED}(\text{DS})$. Assume that $\text{RED}(\text{DS}) = \{K_1, K_2, ..., K_t\}$. It is easy to see that $\text{RED}(\text{DS})$ is a Sperner-system over *A*.

Step 2. For each $K_i \in \text{RED}(\text{DS})$, $1 \le i \le t$, we construct the functional dependency $K_i \to \{d\}$. The obtained relation scheme $\mathcal{S}_d = (R, \mathcal{F})$, where $R = A \cup \{d\}$ and $\mathcal{F} = \{K_i \to \{d\} : K_i \in \text{RED}(\text{DS})\}$, is the result of Algorithm 5.

P r o o f: $K_{\mathcal{S}}(d) = \text{RED}(\text{DS}) \cup \{d\}$:

(1) For any $K \in \text{RED}(\text{DS}) \cup \{d\}$ we have $K \to \{d\}$ and there does not exist $K' \subset K$ such that $K' \to \{d\}$. Consequently, *K* is a minimal set of the attribute *d* with respect to $\mathcal{S}_d = (R, \mathcal{F})$, that is $K \in K_{\mathcal{S}}(d)$.

(2) Conversely, assume that there exists $K \in K_{\mathcal{S}}(d)$, $K \ne \{d\}$, such that $K \notin \text{RED}(\text{DS})$, then we have $K \to \{d\}$ and there does not exist $K' \subset K$ such that $K' \to \{d\}$. It is easy to see that for any $K_i \in \text{RED}(\text{DS})$, $K \not\subset K_i$ because: (i) if $K \subset K_i$ then K_i is not a reduct of *A*; moreover, for any $K_i \in \text{RED}(\text{DS})$, $K_i \not\subset K$; (ii) if $K_i \subset K$ then *K* is not a minimal set of the attribute *d*. From (i), (ii) we can conclude that $\mathcal{K} = \{K, K_1, K_2, ..., K_t\}$ is a Sperner-system and for any $A \subset \mathcal{K}$ we have $A \to \{d\}$. According to the definition, $\mathcal{K}$ is the set of all reducts of DS, that is $K \in \text{RED}(\text{DS})$. This is in contradiction with the condition $K \notin \text{RED}(\text{DS})$. Therefore we have $K \in \text{RED}(\text{DS})$.

From (1) and (2) we conclude $K_{\mathcal{S}}(d) = \text{RED}(\text{DS}) \cup \{d\}$.

It is easy to see that the time complexity of Step 1 is the time complexity of Algorithm 4. The time complexity of Step 2 is $O(|\text{RED}(\text{DS})|)$. Consequently, the worst-case time of Algorithm 5 cannot be more than exponential in the number of attributes. As Algorithm 4, Algorithm 5 is also effective in some special cases.

Example 2. From the decision table in Example 1 we have $\text{RED}(\text{DS}) = \{\{a\}, \{b, c\}\}$. Consequently, the obtained relation scheme is $\mathcal{S}_d = (R, \mathcal{F})$ where $R = \{a, b, c, d\}$ and $\mathcal{F} = \{\{a\} \to \{d\}, \{b, c\} \to \{d\}\}$.

5. Conclution

Based on some results of J. Demetrovics and Vu Duc Thi, concerning Sperner system and minimal sets of an attribute in relational database theory [1, 2, 4], we propose two algorithms on consistent decision tables in rough set theory. The first one is to find all reducts of a consistent decision table. The second is to construct a relation scheme from a consistent decision table. As Algorithm 3 [2], the worst-case time of the two algorithms is exponential in the number of conditional attributes. However, both algorithms proposed are effective in many special cases. These results have significant contribution in extracting knowledge from data tables.

Acknowledgments: *This work was funded by Vietnam's National Foundation for Science and Technology Development (NAFOSTED) via a research grant for fundamental sciences, Grant number 102.01-2010.09.*

References

1. D e m e t r o v i c s, J., V. D. T h i. Keys, Antikeys and Prime Attributes. – Ann. Univ. Sci. Budapest Sect. Comp., Vol. **8**, 1987, 37-54.
2. D e m e t r o v i c s, J., V. D. T h i. Relations and Minimal Keys. – Acta Cybernetica, Vol. **8**, 1998, No 3, 279-285.
3. D e m e t r o v i c s, J., V. D. T h i. Algorithms for Generating Armstrong Relation and Inferring Functional Dependencies in the Relational Datamodel. – Computers and Mathematics with Applications, Great Britain, Vol. **26**, 1993, No 4, 43-55.
4. D e m e t r o v i c s, J., V. D. T h i. Some Remarks on Generating Armstrong and Inferring Functional Dependencies Relation. – Acta Cybernetica, Vol. **12**, 1995, 167-180.
5. D e m e t r o v i c s, J., V. D. T h i. Describing Candidate Keys by Hyper-Graphs. – Computers and Articial Intelligence, Vol. **18**, 1999, No 2, 191-207.
6. D e m e t r o v i c s, J., V. D. T h i. Some Computational Problems Related to Boyce-Codd Normal Form. – Ann. Univ. Sci. Budapest Sect. Comp., Vol. **19**, 2000, 119-132.
7. P a w l a k, Z. Rough Sets. – International Journal of Information and Computer Sciences, Vol. **11**, 1982, No 5, 341-356.
8. P a w l a k, Z. Rough Sets – Theoritical Aspects of Reasoning about Data. Dordrecht, Kluwer Academic Publishers, 1991.
9. S k o w r o n, A., Z. P a w l a k, J. K o m o r o w s k i, L. P o l k o w s k i. A Rough Set Perspective on Data and Knowledge. – In: W. Kloesgen, and J. Zytkow, Eds. Handbook of KDD. Oxford, Oxford University Press, 2002, 134-149.
10. T h i, V. D., N. H. S o n. Some Problems Related to Keys and the Boyce-Codd Normal Form. – Acta Cybernetica, Vol. **16**, 2004, 473-483.
11. T h i, V. D., N. H. S o n. On Armstrong Relations for Strong Dependencies. – Acta Cybernetica, Vol. **17**, 2006, No 3, 521-531.
12. T h i, V. D., N. L. G i a n g. A Method to Construct Decision Table from Relation Scheme. – Cybernetics and Information Technologies, Vol. **11**, 2011, No 3, 32-41.