Structure of Set of Association Rules Based on Concept Lattice

Article · January 2010

DOI: 10.1007/978-3-642-12090-9_19

2 authors:

Tin C Truong
University of Dalat

Anh Tran
University of Dalat

Structure of Set of Association Rules Based on Concept Lattice

Tin C. Truong and Anh N. Tran

Faculty of Mathematics and Computer Science, University of Dalat,
01, Phu Dong Thien Vuong, Dalat, Vietnam
truongchitin@gmail.com, anhtndalat@gmail.com

Abstract. It is important to propose effective algorithms that find basic association rules and generate all consequence association rules from those basic rules. In this paper, we propose the *new* concept of *eliminable itemset* to show how to represent itemset by generators and eliminable itemsets. Using algebraic approach based on equivalence relations, we propose a new approach to partition the set of association rules into basic and consequence sets. After describing their strict relations, we propose two ways to derive all consequence association rules from the basic association rules. These two ways satisfy the properties: sufficiency, preserved confidence. Moreover, they *do not derive repeated consequence rules*. Hence, we save much time for discovering association rule mining.

Keywords: association rule, generator, eliminable itemset, closed itemset.

1 Introduction

Consider the problem of discovering association rules from databases introduced by Agrawal et al. (1993). It is necessary to obtain a set of basic rules such that it can derive all other consequence rules. Recently, some authors (Zaki, 2004; Pasquier et al., 2005) used the lattice of frequent closed itemsets and generators (without the need to find all frequent itemsets) for extracting basic rules. All other rules can derive from them needlessly accessing database. Zaki [8] considered the set of association rules in two subsets: rules with confidence equal to 1 and those with confidence less than 1. He showed that in each subset, there are generalized (or basic) rules and the consequence rules. He also showed how to find the basic rules. However, his method generates many candidate rules. Moreover, he did not present algorithms for deriving the consequence rules. Pasquier et al. [7], split association rule set into set of exact rules and set of approximate rules. They proposed the algorithms for finding basic and consequence sets. However, the one for finding consequence rules is not sufficient (it cannot find out all association rules) and spends much time to generate repeated rules.

In this paper, based on two equivalence relations in the class of itemsets and the association rule set, and on the new concept of "eliminable itemset", we present the structures of basic set, consequence set and their strict relations. We also

N.T. Nguyen et al. (Eds.): Adv. in Intelligent Inform. and Database Systems, SCI 283, pp. 217–227.
springerlink.com

propose a rather smooth partition of the association rule set in which the consequence set is sufficient, non-repeated, confidence-preserved and easy to use.

The rest of the paper is as follows. Section 2 recalls some primitive concepts of concept lattice, frequent itemset and association rule. Sections 3 and 4 present the structures of the class of itemsets and the association rule set. We experimentally validate out theoretical results in section 5 and conclude in section 6.

2 Concept Lattice, Frequent Itemset and Association Rule

Given set $\mathcal{O}$ ($\neq\emptyset$) contained objects (or records) and $\mathcal{A}$ ($\neq\emptyset$) contained items related to each of object $o \in \mathcal{O}$ and $\mathcal{R}$ is a binary relation in $\mathcal{O}$ x $\mathcal{A}$. Now, consider two set functions: λ: $2^{\mathcal{O}} \rightarrow 2^{\mathcal{A}}$, ρ: $2^{\mathcal{A}} \rightarrow 2^{\mathcal{O}}$ are determined in the following: $\forall A \subseteq \mathcal{A}$, $O \subseteq \mathcal{O}$: $\lambda(O)=\{a \in \mathcal{A} \mid (o, a) \in \mathcal{R}, \forall o \in O\}$, $\rho(A)=\{o \in \mathcal{O} \mid (o, a) \in \mathcal{R}, \forall a \in A\}$. In which, $2^{\mathcal{O}}$, $2^{\mathcal{A}}$ are the classes of all subsets of $\mathcal{O}$ and $\mathcal{A}$. Assign that, $\lambda(\emptyset)=\mathcal{A}$, $\rho(\emptyset)=\mathcal{O}$. Defining two set functions h, h' in $2^{\mathcal{O}}$, $2^{\mathcal{A}}$ respectively by: $h = \lambda \circ \rho$, $h' = \rho \circ \lambda$, we say that h'(O), h(A) are the closures of O and A. $A \subseteq \mathcal{A}$ ($O \subseteq \mathcal{O}$) is a closed set if h(A)=A (h'(O)=O). If A, O are closed, $A = \lambda(O)$ and $O = \rho(A)$ then the pair $C=(O, A) \in \mathcal{O}$ x $\mathcal{A}$ is called a concept. In the class of concepts $\mathcal{C} = \{C \in \mathcal{O}$ x $\mathcal{A}\}$, if defining the order relation $\leq$ as relation $\supseteq$ between subsets of $\mathcal{O}$ then $\mathcal{L} \equiv (\mathcal{C}, \leq)$ is the concept lattice [3].

Let minsup s_0 be the minimum support. Let $A \subseteq \mathcal{A}$ be an itemset. The support of A is defined as follows: $sup(A) \equiv |\rho(A)|/|\mathcal{O}|$. If $sup(A) \geq s_0$ then A is frequent itemset [1, 4]. Let $\mathcal{CS}$ be the class of all closed itemsets.

Let c_0 be the minimum confidence (minconf). For any frequent itemset S (with threshold s_0), we take a non-empty, strict subset L from S ($\emptyset \neq L \subset S$), $R = S \backslash L$. Denote that r: $L \rightarrow R$ is the rule created by L, R (or by L, S). Then, $c(r) \equiv |\rho(L) \cap \rho(R)|/|\rho(L)| = |\rho(S)|/|\rho(L)|$ is called the confidence of r. The rule r is an association rule if $c(r) \geq c_0$ [1]. Let $\mathcal{AR} \equiv \mathcal{AR}(s_0, c_0)$ be the set of all association rules with threshold c_0.

For two non-empty itemsets G, A: $G \subseteq A \subseteq \mathcal{A}$, G is a generator of A if h(G)=h(A) and $\forall G' \subset G \Rightarrow h(G') \subset h(G)$ [6]. Let Gen(A) be the class of all generators of A.

The algorithms for finding frequent concept lattice can be found in [2], [4], [5], [6], etc. In this paper, we only concentrate on researching the structure of the association rule set based on the structure of the class of itemsets. Since the size of the paper is limited, so we do not show the proofs of some propositions and theorems, and the codes of some algorithms. We also display briefly the examples.

3 Structure of Itemsets

In this section, we *partition* all *itemsets* into the *disjointed equivalence classes*. *Each class* contains the itemsets that their *supports* are the *same*. Their closures are also the *same*. Theorem 1 in 3.3 shows how to represent itemset by *generators* *and eliminable itemsets* in their closure. *Only based* on *generators, eliminable itemsets,* and *frequent closed itemsets* (some other methods use *maximal* frequent *itemsets*), we *still* can *generate sufficiently* and *quickly all frequent itemsets*.

3.1 Equivalence Relation in the Class of Itemsets

Definition 1. Closed mapping h: $2^{\mathcal{A}} \rightarrow 2^{\mathcal{A}}$ generate a binary relation $\sim_h$ in class $2^{\mathcal{A}}$: $\forall$ A, B $\subseteq \mathcal{A}$: $A \sim_h B \Leftrightarrow h(A) = h(B)$.

Proposition 1. $\sim_h$ is an equivalence relation ([h(A)]=[A], where [A] denotes the equivalence class contained A) and generates the partition of $2^{\mathcal{A}}$ into disjointed classes (the supports of all itemsets in a class are the same). We have $2^{\mathcal{A}} = \sum_{A \in CS} [A]$ [1].

3.2 Eliminable Itemsets

Definition 2. In class $2^{\mathcal{A}}$, a non-empty set R is *eliminable* in S if $R \subset S$ and $\rho(S) = \rho(S \backslash R)$, i.e., when *deleting* set *R in S*, *ρ(S) will not change.* Denote that $\mathcal{N}(S)$ is the class that contains all eliminable itemsets in S.

Proposition 2 (*Criteria of recognizing an eliminable itemset*).

a. $R \in \mathcal{N}(S) \Leftrightarrow \rho(S\backslash R) \subseteq \rho(R) \Leftrightarrow c(r: S\backslash R \rightarrow R) = 1 \Leftrightarrow h(S) = h(S\backslash R)$.
b. $\mathcal{N}(S) = \{A: \emptyset \neq A \subseteq S\backslash GenS,\ GenS \in Gen(S)\}$.

Proof: a. By the properties of ρ, h, and c (see [3, 4, 6, 7]), we easily prove this. b. - $\forall \emptyset \neq A \subseteq S\backslash GenS$, GenS $\in$ Gen(S), we have h(GenS) $\subseteq$ h(S\A) $\subseteq$ h(S). Since GenS is a generator of S so h(GenS) = h(S). Thus, h(S\A) = h(S). By a), A is in $\mathcal{N}$(S).

- If A is in $\mathcal{N}$(S) then there exists GenS $\in$ Gen(S): A $\subseteq$ (S\GenS). Indeed, assume inversely that there exists $a_0 \in$ A and a_0 belong to every generator GenS of S. Let $Gen_0 \in$ Gen(S\A) be a generator of S\A. Thus, *a_0 is not in Gen_0*. $h(Gen_0)$ = h(S\A) = h(S), i.e., Gen_0 is also a generator of S, so $a_0 \in Gen_0$. It is the contradiction! □

3.3 Representation of Itemsets by Generators and Eliminable Itemsets

Theorem 1 (*Representative theorem of itemset*). $\forall$ $\emptyset \neq$A $\in$ $\mathcal{CS}$, $\forall$ X $\in$ [A], $\exists$ Gen_A $\in$ Gen(A), X'$\in \mathcal{N}$(A): *X = Gen_A + X'*.

Proof: If X $\in$ Gen(A) then X'=∅. If X $\notin$ Gen(A), let $Gen_0 \in$ Gen(X) and X'=X\$Gen_0$ $\subseteq$ A\$Gen_0$, then $h(Gen_0)$=h(X)=A, so $Gen_0 \in$ Gen(A), X'$\in \mathcal{N}$(A) and X=Gen_0+X'. □

---

[1] A + B is the union of two disjointed sets A, B. $\sum_{i \in I} A_i = \bigcup_{i \in I} A_i : A_i \cap A_j = \emptyset,\ \forall i, j \in I,\ i \neq j$.

Example 1. Consider the database T in Table 1. Figure 1 shows the lattice of closed itemsets and generators in corresponding with T. This lattice will support for the examples in the rest of this paper. Attribute subsets of {1, 2, 3, 4, 5, 6, 7, 8} are partitioned into the disjointed equivalence classes: [A], [B] ... [I]. Consider class [A], $\mathcal{M}$(A) is {37[2], 3, 7, 5, 35}. Since Gen(A) is {15, 17} so the itemsets in [A] are 15, 17, 135, 137, 157 and *1357*. Their supports are the same.

Table 1. Database T

Record ID (object ID)	Items							
	1	2	3	4	5	6	7	8
1	1	0	1	0	1	0	1	0
2	1	0	1	0	0	1	0	1
3	1	0	0	1	0	1	0	1
4	0	1	1	0	1	0	1	0

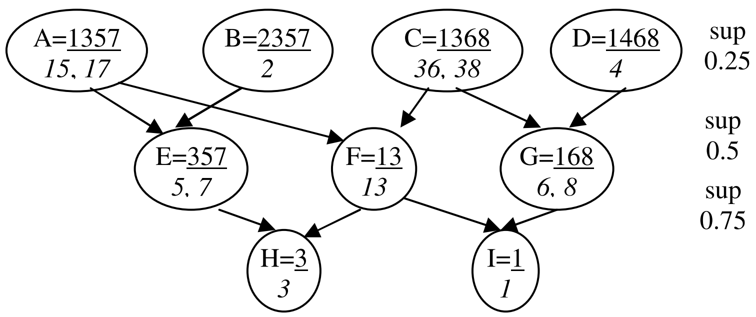

Fig. 1. The lattice of *closed itemsets* and *their generators*. The closed itemsets are underlined, their *supports* are the outside numbers, and their generators are italicized.

4 Structure and Partition of Association Rule Set

Based on the equivalence relation in 4.1, we will *partition* the *set* of association *rules* $\mathcal{AR}$ into *disjointed equivalence classes*. Without loss of general, we only consider an *equivalence class* $\mathcal{AR}$*(L, S)* in corresponding with *(L, S)*[3]. The rules in $\mathcal{AR}$(L, S) are in the form: r_i:$L_i \rightarrow S_i \backslash L_i$ (where $\emptyset \neq L_i \subset S_i$, and $h(L_i)=L$, $h(S_i)=S$). They have the *same confidence*. Then, $\mathcal{AR}$*(L, S)* is also partitioned into two disjointed sets: the basic set $\mathcal{RAR}$(L, S) and the consequence set $\mathcal{CAR}$(L, S). The *basic rules* of $\mathcal{RAR}$(L, S) are r_i:$L_i \rightarrow S \backslash L_i$, where $L_i \in Gen(L)$. In [6, 7], for deriving consequence rules from the basic rule r_i, they *delete all subsets* of the *right-hand*

[2] For brief, we replace {a_1, a_2 ... a_k} with "$a_1a_2...a_k$", where, $a_i \in \mathcal{A}$. For example, 37 is {3, 7}.

[3] From here to the end of this paper, we denote (L, S) for pair of L and S: L, S $\in \mathcal{CS}$ and L $\subseteq$ S.

side of r_i or *move their subsets to* the *left-hand* side of r_i. This method can generate *a large amount* of *repeated consequence rules*. The *repeated* consequence *rules* can be *in their equivalence class* $\mathcal{AR}$(L, S) or *in* the *different classes*.

To remove this repeat, based on the *eliminable itemset* concept, in 4.3, we *only delete* (or *move* to left-hand side) the *subsets* of the *right-hand* side that are *eliminable subsets of S* (or *of L* respectively). *Our method* does *not* generate *repeated* consequence rules. It *still derives sufficiently all consequence rules* from the basic rules in their equivalence class.

4.1 Equivalence Relation $\sim_r$ in the Rule Set $\mathcal{AR}$

Definition 3. Let $\sim_r$ be a binary relation in the rule set $\mathcal{AR}$ that is determined as follows: $\forall$ L, S, Ls, Ss $\subseteq \mathcal{A}$, $\emptyset \neq L \subset S$, $\emptyset \neq Ls \subset Ss$, r:L$\rightarrow$S\L, s:Ls$\rightarrow$Ss\Ls

$$s \sim_r r \Leftrightarrow (h(Ls) = h(L) \text{ and } h(Ss) = h(S)) \Leftrightarrow (Ls \in [L] \text{ and } Ss \in [S]).$$

Proposition 3 (*Disjointed partition of the association rule set*)**.** *$\sim_r$ is an equivalence relation*. For each (L, S), we often take any rule r_0: GenL$\rightarrow$S\GenL (GenL$\in$Gen(L)) to represent the equivalence class $\mathcal{AR}$(L, S) that denotes $[r_0]_{\sim r}$. Hence, the relation $\sim_r$ partition the rule set $\mathcal{AR}$ into disjointed equivalence classes (*the supports of rules in each class are the same*): $\mathcal{AR} = \sum_{L,S \in CS : L \subseteq S} \mathcal{AR}(L, S)$.

4.2 Basic and Consequence Rule Sets in Each Rule Class

Let $\mathcal{RAR}$(L, S) be the *basic rule set* and BARS is the algorithm that generate it. All rules in $\mathcal{RAR}$(L, S) are in the form: GenL$\rightarrow$S\GenL, where: *minimal* left-hand side *GenL* is a *generator of L* and *maximal* right-hand side is *S\GenL*. To derive the set $\mathcal{NRAR}$(L, S) contained all *consequence* rules of $\mathcal{RAR}$(L, S), the previous results used the ways in similar to the following ways. For any r: L$\rightarrow$Right $\in$ $\mathcal{RAR}$(L, S):

- *W_1. Delete subsets* R of *Right* to create the consequence rules r_d: L$\rightarrow$R;
- *W_2. Move* subsets R' of Right or R (in the result of the way *W_1) to* the *left* sides respectively to create consequence rules r_m: L+R'$\rightarrow$ Right\R' or r_m: L+R'$\rightarrow$ R\R'.

Let *$SNRAR_1$* be the algorithm for *finding* consequence rules in $\mathcal{NRAR}$(L, S) by two above ways. This algorithm generates sufficiently the consequence rules. However, it *does not determine immediately* their *confidences* and *generates many repeated rules*.

Pasquier et al. [7] presented the algorithm for finding consequence rules and their *confidences* (figure 9, page 50 and figure 12, page 53). It is unfortunately the algorithm *does not generate sufficiently the consequence rule set*. For example, consider the database T in example 1. Their algorithm *does not discover* the following consequence rules: 25$\rightarrow$7, 27$\rightarrow$5, 23$\rightarrow$7, etc. To overcome this disadvantage, we *correct it* in order to obtain the *$SNRAR_2$ algorithm*. However, although using many conditional checks, *$SNRAR_2$* still derives *many repeated* consequence rules. The repeat can take place in the *same* equivalence rule class or in the *different classes*. For example, the consequence rule 57→3 (is derived from the basic rule 5→37) coincides with one consequence rule of the basic rule 7→35. All of them are in the rule class $[(357, 357)]_{\sim r}$. In *4.3*, we will *overcome all disadvantages* of $SNRAR_1$ and $SNRAR_2$.

4.3 A Preserved-Confidence Non-repeated Partition of Association Rule Set $\mathcal{AR}$

In this section, we will present a *rather-smooth* partition of each equivalence rule class based on two set functions for generating consequence rules $\mathcal{R}_d$, $\mathcal{R}_m$. However, $\mathcal{R}_m$ *still* generates some repeats. To eliminate them, we propose *proposition 6*. Then, *theorem 3* presents a confidence-preserved, non-repeated partition of the rule set $\mathcal{AR}$.

Proposition 4 (*Relation about confidence of basic rules and their consequences*). Suppose that $\varnothing \neq L, R_1, R_2$: $R_1 + R_2 = R$, $S=L+R$, consider rules $r: L \to R$, $r_d: L \to R_1$, $r_m: L+R_2 \to R_1$, $sup(L+R_2) > 0$. We have:

a. $c(r) = 1 \Leftrightarrow \rho(L) \subseteq \rho(R) \Leftrightarrow \rho(L) = \rho(S) \Leftrightarrow h(L) = h(S) \Leftrightarrow R \in \mathcal{M}(S)$.

b. $c(r_d) = c(r) \Leftrightarrow R_2 \in \mathcal{M}(S)$.

c. $c(r_m) = c(r) \Leftrightarrow R_2 \in \mathcal{M}(h(L)) \Leftrightarrow R_2 \subseteq (h(L)\backslash L) \cap R,\ R_2 \neq R$.

Definition 4. Consider two set functions from $\mathcal{AR}$ to $2^{\mathcal{AR}}$ for generating rules (let W_d, W_m be two ways in corresponding with them): $\forall r{:}L\to R \in \mathcal{AR}$: $\mathcal{R}_d(r) = \{s{:}L\to R\backslash R' \mid \varnothing \subset R' \subset R,\ R' \in \mathcal{M}(L+R)\}$, $\mathcal{R}_m(r) = \{s{:}L+R' \to R\backslash R' \mid \varnothing \subset R' \subset R,\ R' \in \mathcal{M}(h(L))\}$.

Proposition 5. $\forall r{:}L\to R \in \mathcal{AR}$, two functions $\mathcal{R}_d$, $\mathcal{R}_m$ satisfy:

$$\mathcal{R}_d(r) \subseteq [r]_{\sim r},\ \mathcal{R}_m(r) \subseteq [r]_{\sim r} \text{ and } \mathcal{R}_m o \mathcal{R}_d(r) \subseteq [r]_{\sim r},\ \mathcal{R}_d o \mathcal{R}_m(r) \subseteq [r]_{\sim r}.$$

Two above functions generate *sufficiently* and *only generate* consequence rules *in the same equivalence rule class*. Thus, their confidences are preserved. They are *different totally* from the consequence rules *in the different* equivalence rule *classes*.

For each (L, S), let us call: $\mathcal{RAR}(L, S) \equiv \{r_0: GenL \to S\backslash GenL \mid GenL \in Gen(L)\}$, $\mathcal{CAR}(L, S) \equiv \mathcal{R}_d(\mathcal{RAR}(L, S)) + \mathcal{R}_m(\mathcal{RAR}(L, S)) + \mathcal{R}_m(\mathcal{R}_d(\mathcal{RAR}(L, S)))$, and $\mathcal{AR}(L, S) \equiv \{r: L' \to R' \mid h(L')=L,\ h(L'+R')=S\}$.

Theorem 2 (*Partition and structure of each equivalence rule class*)**.** For each (L, S):

a. $\mathcal{AR}(L, S) = \mathcal{RAR}(L, S) + \mathcal{CAR}(L, S)$.

b. $\forall r \in \mathcal{CAR}(L, S)$, $\exists r_0 \in \mathcal{RAR}(L, S)$: either $r \in \mathcal{R}_d(r_0)$ or $r \in \mathcal{R}_m(r_0)$ or $r \in \mathcal{R}_m o \mathcal{R}_d(r_0)$.

Proof: a. - "$\supseteq$": Obviously by definition and proposition 5. It is easy to see that $\mathcal{RAR}(L, S)$ is disjointed with $\mathcal{CAR}(L, S)$.

- "⊆": $\forall r$: $L' \to R' \in \mathcal{AR}(L, S)$, $L' \in [L]$, $(L'+R') \in [S]$. Consider three following cases. *(1)* If $L' \in Gen(L)$, $R' = S \backslash L'$ then $r \in \mathcal{RAR}(L, S)$. *(2)* If $L' \in Gen(L)$, $R' \subset (S \backslash L')$ then there exists $\emptyset \neq Rd \in \mathcal{N}(S)$: $S \backslash L' = R' + Rd$ and $r \in \mathcal{R}_d(L' \to S \backslash L') \subseteq \mathcal{R}_d(\mathcal{RAR}(L, S))$. *(3)* If $L' \notin Gen(L)$ then there exists $L_0 \in Gen(L)$, $R_0 \in \mathcal{N}(L)$: $L' = L_0 + R_0$. Let $r_0: L_0 \to R_1 \equiv S \backslash L_0 \in \mathcal{RAR}(L, S)$, $R_1 \equiv (S \backslash L') + (L' \backslash L_0) = (S \backslash L') + R_0$. Since $R' \subseteq S \backslash L'$ so $S \backslash L' = R' + R''$, where $R'' = S \backslash L' \backslash R' = S \backslash (L' + R')$ and $R_1 = R' + R_0 + R''$. *(3a)* If $R'' = \emptyset$ then $r: L_0 + R_0 \to R' \in \mathcal{R}_m(r_0) \subseteq \mathcal{R}_m(\mathcal{RAR}(L,S))$. *(3b)* If $R'' \neq \emptyset$ then: $R'' \in \mathcal{N}(S)$, $r_d: L_0 \to R' + R_0 \in \mathcal{R}_d(r_0)$ and $r: L_0 + R_0 \to R' \in \mathcal{R}_m(r_d) \subseteq \mathcal{R}_m(\mathcal{R}_d(\mathcal{RAR}(L,S)))$.

b. Proposition b is proved while we prove proposition a. □

Example 2. Consider (L, S)=(357, 1357) in the closed lattice in figure 1. The rule class $\mathcal{AR}(L, S)$ contains rules with confidence ½. The rule $r_1: 5 \to 137$ is a basic rule. In [6, 7], for example, if deleting the subset {1} of the right-hand side of r_1 (or moving it to the left-hand side) then we have the consequence rule $r': 5 \to 37$ with $c(r') = 1 \neq ½$ (or $r'': 15 \to 37$, $c(r'') = 1 \neq ½$ respectively). The rule r' (or r'') *coincides* with one rule in $\mathcal{AR}(L, L)$ (or $\mathcal{AR}(S, S)$). *The reason of this repeat* is "*{1}* is *non-eliminable in S"*.

For (L, S), let $\mathcal{R}_d'(\mathcal{RAR}(L, S)) \equiv \mathcal{RAR}(L, S) + \mathcal{R}_d(\mathcal{RAR}(L, S))$. We see that the rules in $\mathcal{RAR}(L, S)$ and $\mathcal{R}_d(\mathcal{RAR}(L, S))$ are different. However, the rules derived from set $\mathcal{R}_d'(\mathcal{RAR}(L, S))$ by function $\mathcal{R}_m$ (or the way W_m) can be repeated. *The following proposition 6 will overcome this last disadvantage*.

Let $\mathcal{S}_m(L, S) \equiv \{r_i: L_i + R' \to R \backslash R' \mid h(L_i + R) = S,\ L_i \in Gen(L),\ \emptyset \neq R' \subseteq R \cap L,\ R' \neq R,\ (i = 1 \text{ or } (i > 1 \text{ and } \forall k: 1 \leq k < i: L_k \not\subseteq L_i + R'))\}$. Different from W_m, let W_m' be the *moving way* that *generates* the set $\mathcal{S}_m$.

Proposition 6 *(Non-repeated generating of consequence rules)***.** For *each (L, S)*:

a. $\mathcal{S}_m(L, S) = \mathcal{R}_m(\mathcal{R}_d'(\mathcal{RAR}(L, S)))$.

b. *Rules in* $\mathcal{S}_m$(L, S) *are different*. Hence, *the consequence rules are also different:*

$$\mathcal{CAR}(L, S) \equiv \mathcal{R}_d(\mathcal{RAR}(L, S)) + \mathcal{S}_m(L, S).$$

Proof: a. - "⊆": $\forall r_i \in \mathcal{S}_m(L,S)$, let $r_0: L_i \to S \backslash L_i \in \mathcal{RAR}(L, S)$. Since $L_i + R \subseteq S$ so $R'' = S \backslash (L_i + R) \in \mathcal{N}(S) \cup \{\emptyset\}$ and $r_d: L_i \to R \in \mathcal{R}_d'(r_0: L_i \to R + R'')$, where $R \cap L_i = \emptyset$. Thus, $L_i + R' \subseteq L$, $L = h(L_i) \subseteq h(L) = L$. Hence, $h(L_i) = h(L_i + R')$: $\emptyset \neq R' \in \mathcal{N}(L_i + R')$. Therefore $r_i: L_i + R' \to R \backslash R' \in \mathcal{R}_m(r_d) \subseteq \mathcal{R}_m(\mathcal{R}_d'(r_0)) \subseteq \mathcal{R}_m(\mathcal{R}_d'(\mathcal{RAR}(L, S)))$.

- "⊇": $\forall r$: $L' \to R''' \in \mathcal{R}_m(\mathcal{R}_d'(\mathcal{RAR}(L, S)))$: $R''' \neq \emptyset \neq L' \subseteq L \subseteq S = h(L' + R''') \supseteq L' + R'''$, so $R''' \subseteq S \backslash L'$. Let i be the minimum index such that: $L' = L_i + R'$, $L_i \in Gen(L)$ and $\emptyset \neq R' \subseteq L \backslash L_i$. Let $R = R''' + R'$, we have $h(L_i + R) = S$, $R' = R \backslash R''' \subseteq R \cap L$ and $L_k \not\subseteq L_i + R'$, $\forall k < i$ (because its inverse will contradict with the way selected i-index).

b. Suppose that $\exists i>j$, r_i: $L_i+R_i' \to R_i \backslash R_i'$, r_j: $L_j+R_j' \to R_j \backslash R_j'$ and $r_i = r_j$. Then, $L_i+R_i' = L_j+R_j' \supset L_j$: the contradiction happen! Therefore, the rules in $\mathcal{S}_{m'}$(L, S) are different. □

Let us call: $\mathcal{RAR} \equiv \sum_{L,S \in CS: L \subseteq S} \mathcal{RAR}(L, S)$, $\mathcal{CAR} \equiv \sum_{L,S \in CS: L \subseteq S} \mathcal{CAR}(L, S)$.

Theorem 3 (*A preserved-confidence, non-repeated partition of the association rule set*)**.** The partition $\mathcal{AR} = \mathcal{RAR} + \mathcal{CAR}$ satisfies the following properties: *sufficiency* (finding sufficiently all association rules), *non-repeating* (consequence association rules are derived from different basic rules are different) and *preserving confidence* (basic rules and their consequence rules have the same confidence).

Based on theorem 3, the following *algorithm CARS* (Consequence Association Rule Set) will generate the set $\mathcal{CAR}$ from the set $\mathcal{RAR}$:

Input: $\mathcal{RAR}$. *Output:* $\mathcal{CAR}$.

$R_{d}\_AR = \varnothing$; $R_{m}\_AR = \varnothing$;
forall (<r_0: $L_i \to$Right, $c(r_0)$, L, S> $\in \mathcal{RAR}$) do {
  $R_{m}\_AR = R_{m}\_AR + MA$ (L_i, Right, $c(r_0)$);
  if ($c(r_0)=1$) then forall ($\varnothing \neq R \subset$ Right) do {
      $R_{d}\_AR = R_{d}\_AR + \{r_d$: $L_i \to R$, $c(r_d)=1\}$;
      $R_{m}\_AR = R_{m}\_AR + MA$ (L_i, R, 1);
    }
  else forall ($\varnothing \neq R \subset$ Right and $R \in \mathcal{M}$(S)) do { // (*)
      $R_{d}\_AR = R_{d}\_AR + \{r_d$: $L_i \to$ Right$\backslash$R, $c(r_d) = c(r_0)\}$;
      $R_{m}\_AR = R_{m}\_AR + MA$ (L_i, Right$\backslash$R, $c(r_0)$);
    }
}
$\mathcal{CAR} = R_{d}\_AR + R_{m}\_AR$;
return $\mathcal{CAR}$

In which, MA (Move Appropriately) is the algorithm that generates the different consequence rules by the moving way W_m'.

Example 3. Consider the rule class $\mathcal{AR}$(L=357, S=1357) in corresponding with (L, S) in the closed lattice in figure 1. Two basic rules in corresponding with two generators L_1=5, L_2=7 of L are <r_1:5→137, ½, L, S>, <r_2:7→135, ½, L, S>. By the ways W_d, W_m', the algorithm CARS only generate non-repeated consequence rules as follows:

- *Consider r_1: 5→137.* The sets R (R ⊂ 137) that are *eliminable itemsets* in S are 3, 7, and 37. *Delete* each R from 137 in order to create the following consequence rules 5→17, 5→13, and 5→1. The results of *moving*: *on* the basic rule r_1 are 35→17, *57→13, 357→1; on* the consequence rule, *5→17* is 57→1; and *on 5→13* is 35→1.

• *Consider r_2: 7→135* (with L_2). The sets R (R ⊂ 135) that are *eliminable itemsets* in S are 3, 5, and 35. *Deleting* each R from 135, we have 7→15, 7→13, 7→1. The results of *moving*: *on r_2* is 37→15 *(57→13, 357→1 are removed because 57 ⊇ L_1, 357 ⊇ L_1)*; *on 7→15* is empty because *57 ⊇ L_1*; and *on 7→13* is 37→1.

5 Experimental Results

We use four databases in [9] during these experiments. Table 2 shows their characteristics.

Table 2. Database characteristics

Database (DB)	# Records (Objects)	# Items	Average Record size
Pumsb (P)	49046	7117	74
Mushroom (M)	8124	119	23
Connect (Co)	67557	129	43
Chess (C)	3196	75	37

Table 3 contains the results of finding all association rules upon them with the different minconfs *(MC)*, by BARS+SNRAR$_1$ (*SBN_1*), BARS+SNRAR$_2$ (*SBN_2*) and BARS+CARS (*SBC*). In which, FCS is the number of frequent closed itemsets (G is the number of generators), FS is the number of frequent itemsets (GFC = G/FCS), AR is the number of all association rules, and ER is the ratio of the number of basic rules to *AR*. *With SBN_1,* CC is the ratio of the number of repeated consequence rules to the size of the basic rule set. *With SBC*, NS and RC are the ratios of the numbers of eliminable and non-moved subsets *to the number of all subsets* (of the right-hand sides of basic rules). Finally, T_1, T_2 and T_3 are in turn the running times (by seconds) of *SBN_1, SBN_2* and *SBC*.

In most of the results, the amount of repeated consequence rules (CC) is large (it ranges from 641% to 5069%), and the number of the eliminable itemsets (NS) is small (from 5% to 51%). Hence, *the time for using two ways W_d, W_m' in CARS is small*. Moreover, the number of generators is small (GFC ranges from 1.0 to 1.4). Therefore, *checking the eliminable property (*)* of a subset R of S in CARS will *reduce significantly* not only the *cost* for *considering the repeated* rules *generated* by the *way W_1* in *$SNRAR_1$* but also the *one* to *traverse the subsets of the set of association rules* in *$SNRAR_2$*. Furthermore, the *time* for *checking conditional* in *MA* (see the definition of S_m) will be *smaller* than *the one* for *generating* (by the way *W_2* in *$SNRAR_1$*) *redundantly* the consequence rules and *deleting* them, or *the one* for *traversing subsets* and *checking repeats* in *$SNRAR_2$* many times.

Experimental results show that the reduction in running time by our approach ranges from a factor of 2 to 368 times. Let us recall that SNRAR$_1$ does not determine immediately the confidences of the consequence rules.

Table 3. Experimental results of SBN_1, SBN_2 and SBC upon P, M, Co and Ch

DB (MS%)	MC (%)	FCS (G)	FS (GFC)	AR	ER (%)	CC (%)	NS (%)	RC (%)	T_3 (s)	T_1/T_3	T_2/T_3
P (90%)	95	1465	2607	46143	52	1040	20	50	22	2	85
	50; 5	(2030)	(1.4)	71474	51	1966	15	50	7	7	*368*
M (30%)	95	427	2735	14366	7	641	51	10	4	1	5
	50	(558)	(1.3)	79437	7	4290	33	11	5	5	85
	5			94894	8	5069	29	9	5	6	90
Co (95%)	95; 50; 5	811 (811)	2201 (1.0)	78376	33	3429	14	0	19	4	49
Ch (87%)	95	1183	1553	19963	71	758	8	0	129	1	2
	50; 5	(1183)	(1.0)	41878	74	1731	5	0	10	4	16

6 Conclusion

The theoretical affirmations in this paper *show clearly* the *structures* of the class of *itemsets* and the *association rule set*, based on the *proposal of* the *equivalence relations* on them and on the "*eliminable set*" concept. We propose the partition of the class of itemsets into disjointed classes and show how to represent itemset by *generators and eliminable itemsets.* Then, we also propose the rather-smooth disjointed *partition* of the *association rule set* into the basic and consequence sets and their *strict relation*.

As a result, we build the *CARS algorithm* that derives *sufficiently* and *quickly* all *consequence* rules from the corresponding basic rules. This algorithm satisfies the following properties: *preserving confidence*, *non-repeating*. Hence, it *reduces significantly* the time for discovering all association rules. Moreover, *two ways* W_d, W_m' used *in CARS* are *convenient* and *close* to *user.*

References

1. Aggarwal, C.C., Yu, P.S.: Online Generation of Association Rules. In: Proceedings of the International Conference on Data Engineering, pp. 402–411 (1998)
2. Agrawal, R., Srikant, R.: Fast algorithms for mining association rules. In: Proceedings of the 20th Very Large Data Bases Conference Santiago, Chile, pp. 478–499 (1994)
3. Bao, H.T.: An approach to concept formation based on formal concept analysis. IEICE trans, Information and systems E78-D(5) (1995)
4. Bastide, Y., Pasquier, N., Taouil, R., Stumme, G., Lakhal, L.: Mining minimal Non-Redundant Association Rules Using Frequent Closed Itemsets. In: 1st International Conference on Computational Logic (2000)
5. Godin, R., Missaoul, R., Alaour, H.: Incremental concept formation algorithms based on Galois lattices. Magazine of computational Intelligence, 246–247 (1995)
6. Pasquier, N., Bastide, Y., Taouil, R., Lakhal, L.: Efficient Mining of association rules using closed item set lattices. Information systems 24(1), 25–46 (1999)

7. Pasquier, N., Taouil, R., Bastide, Y., Stumme, G., Lakhal, L.: Generating a condensed representation for association rules. J. of Intelligent Information Systems 24(1), 29–60 (2005)
8. Zaki, M.J.: Mining Non-Redundant Association Rules. Data Mining and Knowledge Discovery 9, 223–248 (2004)
9. Frequent Itemset Mining Dataset Repository (2009), `http://fimi.cs.helsinki.fi/data/`