

Incremental Maintenance of Association Rule Bases

Petko Valtchev¹⁾, Rokia Missaoui²⁾, Mohamed Rouane Hacene¹⁾, Robert Godin³⁾

¹⁾ DIRO, Université de Montréal, C.P. 6128, Succ. “Centre-Ville”, Montréal (Qc), Canada, H3C 3J7

²⁾ Département d’informatique et d’ingénierie, UQO, C.P. 1250, succursale B, Gatineau (Qc), Canada, J8X 3X7

³⁾ Département d’informatique, UQAM, C.P. 8888, succ. “Centre Ville”, Montréal (Qc), Canada, H3C 3P8

Abstract

Association rule mining from transaction databases (TDB) is a classical data mining task, whereby the most computationally intensive step is the detection of frequently occurring patterns, called frequent itemsets (FIs), from which the rules are further extracted. The number of FIs may be potentially large, leading to an even greater number of rules. Approaches based on closure operators, Galois connections and Galois (concept) lattices have been proposed in an attempt to reduce the size of the resulting rule set. Thus, the search for frequent patterns has been limited to closed itemsets, while looking for a representative and reduced set of association rules, called a basis, which nevertheless conveys all the relevant information. In these approaches, the minimal generators of a closed itemset play a key role for the itemset/rule construction. In our paper, we present a straightforward method for the maintenance of an association rule basis when a new transaction is added to the TDB. To that end, we utilize results on the incremental update of lattices and extend them with new properties to form a complete framework for association rule base maintenance. In particular, we define a simple and efficient method for on-line computation of the generators for closed itemsets and show how its output is used in the update of the rule basis.

Key Words: Association rule mining, Galois (concept) lattices, closed itemsets, rule bases, minimal generators, incremental methods.

1 Introduction

Association rule mining from a transaction database [2] is a classical data mining topic, whereby the most challenging problem is the detection of frequent patterns in the transaction sets

(*frequent itemsets*) [1, 5, 14]. A major difficulty with association rules is the prohibitive number of frequent itemsets (and hence association rules) that can be generated even from a reasonably large data set. The frequent *closed* itemsets (*FCIs*) research topic [17, 18, 27, 26] constitutes a promising solution to the problem of reducing the number of the reported association rules. A step further in this direction is the construction of association rule bases out of the *FCIs* [13, 15, 17], an operation which largely relies on the notion of generator of a closed itemset (*CI*). Yet another difficulty arises with dynamic databases where the transaction set is frequently updated. Although the necessity of processing volatile data in an incremental manner has been repeatedly emphasized in the general data mining literature (see for example [11]), a few incremental algorithms for association rule generation (and hence frequent itemset detection) have been reported so far [11, 6, 20, 9, 7, 3].

Our own approach to incremental rule mining is motivated by the belief that *FCIs* and bases provide the key to compact rule sets and low storage requirements. Therefore, we have been investigating the potential benefits of using Galois lattice theory and formal concept analysis as a formal basis for the resolution of this problem. In previous studies, we have tackled the problem of incrementally mining *FCIs* as a separate task [23, ?]. In this paper, we present an integrated approach to incremental association rule mining, which covers the entire process from *FCI* extraction to rule generation, allows flexibility and efficiency and produces various rule bases [15, 17] that help reduce the size of the output. More specifically, we will focus on two main tasks: incremental extraction of *FCIs* and incremental identification of generators. Then, we show how the evolution in the generator set propagates to the rule bases.

The paper starts with a short recall on associa-

tion rule mining problem (Section 2) followed by a brief summary of relevant results from Galois lattice theory and algorithms (Section 3). The theoretical foundations of our generator-based framework are presented in Section 4 while details about generator set update and rule basis maintenance are given in Section 5.

2 Association rule mining problem

Given a database of transactions, the problem of mining association rules consists in generating all association rules that have certain user-specified minimum support and confidence.

Let $\mathcal{I} = \{i_1, i_2, \dots, i_m\}$ be a set of m distinct items. A transaction T contains a set of items in \mathcal{I} , and has an associated unique identifier called *TID*. A subset X of \mathcal{I} where $k = |X|$ is referred to as a k -itemset (or simply an itemset), and k is called the length of X . A transaction database (*TDB*), say \mathcal{D} , is a set of transactions. The fraction of transactions in \mathcal{D} that contain an itemset X is called the support of X and is denoted $supp(X)$. For example, the support of bcd in Table 1 is 33%¹. Thus, an itemset is frequent (or large) when $supp(X)$ reaches at least a user-specified minimum threshold called *minsupp*.

As a running example, let us consider Table 1 which shows a supermarket database with a sample set of transactions $\mathcal{D} = \{1, \dots, 9\}$ involving items from the set $\mathcal{I} = \{a, \dots, h\}$. The itemsets whose support is higher than 30% of $|\mathcal{D}|$ are given in Table 2.

Trans.	Items
1	b, c, d, g, h,
2	a, c,
3	d, e, f, g, h
4	g
5	e, f, h
6	a, b, c, d
7	b, c, d
8	d
9	c, d, f, g, h

Table 1: A sample transaction database.

¹In the rest of the paper, we shall use the number, rather than the proportion, of the transactions supporting X .

Itemset	Supp.	Itemset	Supp.
b	3	c	5
d	6	f	3
g	4	h	4
bc	3	bd	3
cd	4	dg	3
dh	3	fh	3
gh	3	-	-
bcd	3	dgh	3

Table 2: The itemsets of support greater than 30%.

2.1 Association rule generation

An association rule is an implication of the form $X \Rightarrow Y$, where X and Y are subsets of \mathcal{I} , and $X \cap Y = \emptyset$ (e.g., $d \Rightarrow c$). The support of a rule $X \Rightarrow Y$ is defined as $supp(X \cup Y)$ while its confidence is computed as the ratio $supp(X \cup Y)/supp(X)$. For example, the support and confidence of $d \Rightarrow c$ are 44% and 66% respectively.

The problem of mining association rules with given minimum support and confidence (called *minconf*) can be split into two steps:

- Detecting all frequent (large) itemsets (*FIs*) (i.e., itemsets that occur in the database with a support $\geq minsupp$),
- Generating association rules from large itemsets (i.e., rules whose confidence $\geq minconf$).

The second step is relatively straightforward. However, the first step presents a great challenge because the set of frequent itemsets may grow exponentially with $|\mathcal{I}|$.

2.2 Frequent closed itemsets

Since the most time consuming operation in association rule generation is the computation of frequent itemsets, some recent studies have proposed a search space pruning based on the computation of frequent *closed* itemsets only, without any loss of information. In particular, approaches inspired by *Galois lattices* [4] have been suggested to that end [26, 17]. Thus, only a subset of the *FIs* is produced and stored, which is made up of *frequent closed itemsets* (*FCIs*). An itemset X is a closed itemset if adding an arbitrary item i from $\mathcal{I} - X$ to X results in an itemset whose support is lower than the support of X (see

next section for a formal definition):

$$\forall i \in \mathcal{I} - X, \text{supp}(X \cup \{i\}) < \text{supp}(X).$$

The following table provides the set of all *CI*s, both frequent (more than 30%) and infrequent ones, related to the TDB of nine transactions (see Table 2 and Figure 2).

Set of CIs	Closed itemsets
<i>FCI</i>	c, d, g, h, cd, fh, bcd, dgh
<i>CI - FCI</i>	ac, cdgh, dfgh, efh, abcd, bcdgh, cdgh, defgh, abcdefgh

The key property in the *CI* framework states that any itemset has the same support as its closure, and hence is as frequent as its closure. For example, the closure of the itemset *b* is *bc* and both sets have a support of 4.

Previous work [11, 17] has shown that *CI*s and *FCI*s may be used in the generation of all *FIs* and rules, whereby there is no need to further access the *TDB*. Another important aspect of the rule generation problem is the enormous number of rules that can be generated even for high support and confidence thresholds. Producing minimal covers, or basis, for the entire rule sets is more useful from the user point of view. Again, previous work has shown that such minimal covers can be generated directly from the set of *FCI*s [19] or the lattice of *CI*s [26] (see the next section). Furthermore, the lattice structure provides a context for the efficient generation of rules limited to any given frequent item subset [11].

2.3 Incremental generation

In real-life applications, databases tend to emerge as collections of very large and dynamic sets which present a real challenge for mining association rules in a reasonable computation time. In particular, the possibility of constructing the *FIs* from such a dynamic set incrementally, i.e., without restarting from scratch on each change in the data, is a highly sought feature.

To motivate our study of the algorithmic problems which arise with dynamic data, let us consider the following example. Assume that the *initial TDB*, \mathcal{D} , includes only transactions $\{1, 2, 4 \dots, 8\}$ while the *increment* is made up of transaction #9. The following table provides the sets of *CI* for both the initial *TDB* and the increment. The augmented *TDB*, i.e., \mathcal{D}^+ , is the union of \mathcal{D} and the increment.

Set of CIs	Closed itemsets
<i>CI</i>	c, d, g, h, ac, bcd, dgh, efh, abcd, bcdgh, defgh, abcdefgh
<i>Increment</i>	cd, fh, cdgh, dfgh, cdgh

While a batch algorithm would have to start the computation of the *CI*s in \mathcal{D}^+ from scratch, an incremental method will use both the new transaction and the existing set of *CI*s from \mathcal{D} to compute the new *CI*s in *Increment* and thus obtain the complete set of *CI*s from \mathcal{D}^+ .

3 Background on Galois (concept) lattices

The following is a summary of the key results from the Galois lattice theory [4], which provides the basis of our approach toward incremental generation of frequent closed itemsets and incremental maintenance of generators.

3.1 Formal concept analysis basics

The domain focuses on the partially ordered structure², known under the name of *Galois lattice* [4] or *concept lattice* [25, 10], which is induced by a binary relation *R* over a pair of sets *O* (*objects*) and *A* (*attributes*). For example, Table 1 shows the binary relation $\mathcal{K} = (O, A, R)$ (or *context*) drawn from the TDB of Table 1 where transactions are taken as objects, items as attributes, and *oRa* is to be read as “transaction *o* has an item *a*”. Two functions, *f* and *g*, summarize the links between subsets of objects and subsets of attributes induced by *R*.

Definition 1. *The function *f* maps a set of objects into a set of common attributes, whereas *g*³ is the dual for attribute sets:*

- $f : \mathcal{P}(O) \rightarrow \mathcal{P}(A), f(X) = X' = \{a \in A \mid \forall o \in X, oRa\},$
- $g : \mathcal{P}(A) \rightarrow \mathcal{P}(O), g(Y) = Y' = \{o \in O \mid \forall a \in Y, oRa\}.$

For example, $f(13) = dgh^4$.

Furthermore, the compound operators $g \circ f(X)$ and $f \circ g(Y)$ (denoted by $''$) are *closure operators* over

²An excellent introduction to partial orders and lattices may be found in [8].

³Hereafter, both *f* and *g* are expressed by $'$.

⁴We use a separator-free form for sets, e.g. 127 stands for $\{1, 2, 7\}$ and $g(abc) = 127$.

	<i>a</i>	<i>b</i>	<i>c</i>	<i>d</i>	<i>e</i>	<i>f</i>	<i>g</i>	<i>h</i>
1		X	X	X			X	X
2	X		X					
3				X	X	X	X	X
4							X	
5					X	X		X
6	X	X	X	X				
7		X	X	X				
8				X				
9			X	X		X	X	X

Table 3: Binary table $\mathcal{K} = (O = \{1, 2, \dots, 8\}, A = \{a, b, \dots, h\}, R)$ and the object 9.

$\mathcal{P}(O)$ and $\mathcal{P}(A)$ respectively. This means, in particular, that $Z \subseteq Z''$ and $(Z'')'' = Z''$ for any $Z \in \mathcal{P}(A)$ or $Z \in \mathcal{P}(O)$. Thus, each of the $''$ operators induces a family of *closed* subsets, further denoted $\mathcal{C}_{\mathcal{K}}^a$ (for *attributes*) and $\mathcal{C}_{\mathcal{K}}^o$ (for *objects*) respectively.

Figure 1: The Hasse diagram of the lattice derived from \mathcal{K} .

Using the example in Table 3, the attribute set in $\mathcal{C}_{\mathcal{K}}^a$, represents the *CI*s for the first eight transactions in Table 1.

A key result of the domain states that, when ordered with set inclusion, both $\mathcal{C}_{\mathcal{K}}^o$ and $\mathcal{C}_{\mathcal{K}}^a$ form complete lattices which are sub-lattices of $\mathcal{P}(O)$ and $\mathcal{P}(A)$ respectively. Moreover, f and g constitute bijective mappings between $\mathcal{C}_{\mathcal{K}}^o$ and $\mathcal{C}_{\mathcal{K}}^a$, and isomorphisms between the corresponding lattices.

Definition 2. A *closed pair* or *concept* is a pair of sets (X, Y) where $X \in \mathcal{P}(O)$, $Y \in \mathcal{P}(A)$, $X = Y'$ and $Y = X'$. X is called the *extent* and Y the *intent* of the concept.

For example, $(167, bcd)$ is a closed pair, but $(35, fh)$ is not. Within the *CI* mining framework, the closed pairs are useful as they contain both a closed itemset Y and the (closed) set X of all transactions which share exactly Y , i.e., the supporting transaction set.

Furthermore, the set $\mathcal{C}_{\mathcal{K}}$ of all closed pairs/concepts of $\mathcal{K} = (O, A, I)$ is partially ordered by intent/extent inclusion:

$$(X_1, Y_1) \leq_{\mathcal{K}} (X_2, Y_2) \Leftrightarrow X_1 \subseteq X_2, Y_2 \subseteq Y_1.$$

Theorem 1. The partial order $\mathcal{L} = \langle \mathcal{C}_{\mathcal{K}}, \leq_{\mathcal{K}} \rangle$ is a complete lattice, called **Galois** or **concept** lattice, with joins and meets defined as follows:

- $\bigvee_{i=1}^k (X_i, Y_i) = ((\bigcup_{i=1}^k X_i)'', \bigcap_{i=1}^k Y_i)$,
- $\bigwedge_{i=1}^k (X_i, Y_i) = (\bigcap_{i=1}^k X_i, (\bigcup_{i=1}^k Y_i)'')$.

The Hasse diagram of the lattice \mathcal{L} drawn from $\mathcal{K} = (\{1, 2, \dots, 8\}, A, R)$ is shown in Figure 1 where intents and extents are drawn on both sides of a node representing a concept. For example, the join and the meet of the closed pairs $c_1 = (26, ac)$ and $c_2 = (67, bcd)$ are $(1267, c)$ and $(6, abcd)$ respectively.

The lattice provides a hierarchical organization of all concepts which may be used to speed-up their computation and subsequent retrieval. It is particularly useful when the set of closed pairs (closed itemsets) is to be generated incrementally, a problem which is addressed in the next section.

3.2 Rules and order

As indicated earlier, association rules can be advantageously generated from *F*CI*s* rather than *F*I*s*. However, even in this case, there is still a large set of generated rules with information redundancy. It is therefore more useful and relevant to provide a non-redundant rule set.

For example, suppose the following rules are valid with the same support and confidence levels:

$ab \rightarrow cde$	$ab \rightarrow c$	$ab \rightarrow d$
$abc \rightarrow d$	$abcd \rightarrow e$	$abde \rightarrow c$

The first rule $ab \rightarrow cde$ is sufficient because the others do not give additional information. A subset of rules S of a rule set R which preserves the information of R is called a *cover* of R .

Minimal covers for exact rules (confidence = 1) have been extensively studied in formal concept analysis (for example the *Guigues-Duquenne* basis [13,

10]) and database theory. The *Guigues-Duquenne* basis is minimal with respect to the number of rules, which is a relevant criterion from a data mining perspective. Rules of the basis are of the form $p \rightarrow p''$ where p is a *pseudo-closed* set (also called a pseudo-intent). An item set p is pseudo-closed if it is not closed and contains the closure of all its subsets that are pseudo-closed.

The notion of *generator* of a closed itemset plays a key role in the rule set construction as indicated in [17, ?] and defined in the following.

Definition 3. A *generator itemset* (or *generator*) Z for a closed set X is a minimal subset of X such that $Z'' = X$.

Definition 4. A *generic basis* for exact rules is a collection of rules of the form $Z \rightarrow Z'' - Z$ such that Z is a generator for Z'' and $Z \neq Z''$.

The generic basis can hence be constructed from *FCIs* and generators only. For example, concept #7 (see Figure 1) leads to the generation of rules $b \rightarrow cd$ and $cd \rightarrow b$ with a support of 0.37.

Definition 5. A *Luxenberger cover basis* is a set of rules of the form $X \rightarrow Y - X$ where X and Y are closed itemsets and Y covers X [15]. Support and confidence of a rule r in such basis are $\text{supp}(Y)$ and $|Y'|/|X'|$ respectively.

The covering relationship corresponds to the Hasse diagram of the *CI*s, which hence needs to be updated whenever new transactions occur. For example, rules $c \rightarrow a$ ($\text{supp.} = 0.25, \text{conf.} = 0.5$) and $c \rightarrow bd$ ($\text{supp.} = 0.37, \text{conf.} = 0.75$) are extracted from concept #2 (see Figure 1) as a part of the Luxenburger basis.

However, the rules in this basis are not necessarily maximally informative (i.e., the premise is minimized and the consequence is maximized). This has motivated the definition of the informative basis [16] where every rule is maximally informative.

Definition 6. An *informative basis* is a set of rules of the form: $Z \rightarrow Y_2 - Z$ where Z is a generator for Y_1 such that $C_1 = (X_1, Y_1)$, $C_2 = (X_2, Y_2)$ and C_1 covers C_2 . Support and confidence of a rule r in such basis are $\text{supp}(Y_2)$ and $|Y_2'|/|Z'|$ respectively.

Based on Figure 1, rules such as $b \rightarrow acd$ ($\text{supp} = 1/9, \text{conf} = 1/3$) and $e \rightarrow dfgh$ ($\text{supp} = 0.25, \text{conf} = 0.5$) are part of the informative basis.

From the previous definitions, one can see that incremental maintenance of concepts (and hence *CI*s),

their covering relationships and their generators can be used to produce both the Luxenburger cover and the informative basis.

3.3 Incremental lattice update

Incremental methods construct the lattice \mathcal{L} starting from $\mathcal{L}_0 = \langle \{(\emptyset, A)\}, \emptyset \rangle$ and gradually incorporating a new object o_i into the lattice \mathcal{L}_{i-1} which corresponds to a table $\mathcal{K}_{i-1} = (\{o_1, \dots, o_{i-1}\}, A, I)$. Each incorporation involves a set of structural updates [21].

3.3.1 Principles of the incremental approach

The basic approach initially described in [12], follows a fundamental property of the *Galois connection* established by f and g on $(\mathcal{P}(O), \mathcal{P}(A))$: both families of closed subsets are themselves closed under set intersection [4]. Thus, the integration of a new object/transaction is mainly aimed at the insertion into \mathcal{L}_{i-1} , or simply \mathcal{L} , of all concepts (further called *new* concepts) whose intent is the intersection of $\{o_i\}'$ with the intent of an existing concept but not an already existing intent.

Hence, three groups of concepts in \mathcal{L} are distinguished: *generator* concepts (denoted $\mathbf{G}(o)$) which give rise to new concepts and help compute the respective new intents and extents; *old* concepts (denoted $\mathbf{U}(o)$) which remain completely unchanged; and *modified* concepts (labelled $\mathbf{M}(o)$) which evolve by integrating o_i into their extents while their intents remain stable. The main tasks for a reconstruction algorithm include the identification of the three sets together with the creation of the new concepts ($\mathbf{N}^+(o)$), and their subsequent integration into the existing lattice structure. These jointly leads to the construction of the target lattice \mathcal{L}^+ .

The above rather intuitive framework has been thoroughly examined and reestablished upon a solid theoretical background from lattice theory, initially in [22] and then in [23]. In the following paragraph, a summary of the key concepts and structural results from this work is provided.

3.3.2 Theoretical framework for incremental lattice construction

Intuitively, in order to update a lattice \mathcal{L} (taken as a data structure, e.g., the graph of its precedence relation) with respect to an object o , one has to produce the new nodes and properly “connect” them into the graph, i.e., compute their respective upper and lower

covers and complete the precedence relation appropriately. The members of $\mathbf{M}(o)$ must be detected and properly updated as well. The exact location of a new element being tagged by a generator, a member of $\mathbf{G}(o)$, must be first identified within \mathcal{L} .

In the rest of the paper, the members of the set $\mathbf{G}(o)$ will be referred to as *c-generators* to avoid mixing it with the (minimal) generator of a closed itemset: the former is a concept, whereas the latter is an itemset (comparable to a concept intent).

From Godin *et al.* [12] we know that generators are maximal for the attribute set that results from the intersection between the concept intent and the description of the new object. Moreover, the authors proved that in \mathcal{L}^+ every generator is a lower cover of the generated new concept. Later work on a broader class of lattices called *type lattices* [21] has established that the set of all generator concepts induces a meet sub-semi-lattice of both \mathcal{L} and \mathcal{L}^+ , and that this structure is isomorphic to the sub-order of \mathcal{L}^+ induced by $\mathbf{N}(o)$. However, few results have been provided about the precedence relation between new elements. Finally, a recent work [22] has explicitly characterized the precedence relation in \mathcal{L}^+ and the way it is obtained from the precedence in \mathcal{L} . A complete theoretical framework, including strict definitions for the sets $\mathbf{G}(o)$ and $\mathbf{M}(o)$ from \mathcal{L} , as well as their counterparts from \mathcal{L}^+ , $\mathbf{G}^+(o)$ and $\mathbf{M}^+(o)$, together with the set $\mathbf{N}^+(o)$ has been developed in [23]. The following remarkable facts from this framework will be used in the present study.

First, $\mathbf{G}(o)$ and $\mathbf{M}(o)$ are the maximal concepts in the equivalence classes induced by the function $\mathcal{Q} : \mathcal{C} \rightarrow 2^A$ defined as $\mathcal{Q}(c) = Y \cap \{o\}'$ where $c = (X, Y)$. Then, the order in \mathcal{L}^+ is such that:

- given a new concept c_n in $\mathbf{N}^+(o)$, and its respective *c-generator* c_g , c_n has c_g as a lower cover whereas all its other lower covers, if any, are in $\mathbf{N}^+(o)$.
- given a new concept c_n in $\mathbf{N}^+(o)$, all its upper covers are in $\mathbf{M}^+(o) \cup \mathbf{N}^+(o)$,
- given a modified concept c_m in $\mathbf{M}^+(o)$, its lower covers in \mathcal{L}^+ differ from the lower covers of the corresponding concept⁵ in \mathcal{L} by: (i) the inclusion of a (possibly empty) subset of $\mathbf{N}^+(o)$, and (ii) the drop out of all the concepts from $\mathbf{G}^+(o)$.

We define a mapping χ^+ of a concept $c = (X, Y)$

⁵i.e., $(X - \{o\}, Y)$, if $c_m = (X, Y)$.

in \mathcal{C} to the concept in \mathcal{L}^+ whose intent is $Y \cap \{o\}'$, where o' is the new object.

To sum up, from all the concepts that behold their intents from \mathcal{L} to \mathcal{L}^+ , only *c-generators* have their upper covers changed, and only modified may have their lower-covers changed. Consequently, given a concept c from $\mathcal{C}^+ - \mathbf{N}^+(o)$ and a new concept c_n from $\mathbf{N}^+(o)$, $c \leq_{\mathcal{L}^+} c_n$ if and only if $c \leq_{\mathcal{L}^+} c_g$ where c_g is the *c-generator* of c_n .

3.3.3 Description of our lattice algorithm.

In the following, we present an algorithm which update a given lattice when a new object is added. It handles the identification of the above three concept categories and the creation of new concepts. More details about the algorithm can be found in [?].

```

1: procedure ADD-OBJECT(In/Out:  $\mathcal{L} = \langle \mathcal{C}, \leq \rangle$  a lattice; In:  $o$  an object)
2:
3: SORT( $\mathcal{C}$ )
4: for all  $c$  in  $\mathcal{C}$  do
5:    $new-max \leftarrow \operatorname{argmax}(\{|\mathcal{Q}(\bar{c})| \mid \bar{c} \in \operatorname{Cov}^u(c)\})$ 
6:   if  $|\mathcal{Q}(c)| \neq |\mathcal{Q}(new-max)|$  then
7:     if  $|\mathcal{Q}(c)| = |\operatorname{Intent}(c)|$  then
8:        $\operatorname{Extent}(c) \leftarrow \operatorname{Extent}(c) \cup \{o\}$  {  $c$  modified }
9:        $\mathbf{M}_o \leftarrow \mathbf{M}_o \cup \{c\}$ 
10:       $new-max \leftarrow c$ 
11:     else
12:        $\hat{c} \leftarrow \operatorname{new-concept}(\operatorname{Extent}(c) \cup \{o\}, \mathcal{Q}(c))$  {  $c$  generator }
13:       NEW-LINK( $c, \hat{c}$ )
14:       COMPUTE-GENERATORS( $c, \hat{c}$ )
15:        $\operatorname{Candidates} \leftarrow \{\operatorname{ChiPlus}[\bar{c}] \mid \bar{c} \in \operatorname{Cov}^u(c)\}$ 
16:       for all  $\bar{c}$  in MIN( $\operatorname{Candidates}$ ) do
17:         NEW-LINK( $\hat{c}, \bar{c}$ )
18:         if  $\bar{c} \in \mathbf{M}_o$  then
19:           DROP-LINK( $c, \bar{c}$ )
20:          $new-max \leftarrow \hat{c}$ 
21:          $\mathcal{L} \leftarrow \mathcal{L} \cup \{\hat{c}\}$ 
22:          $\operatorname{ChiPlus}[c] \leftarrow new-max$ 

```

Algorithm 1: Update of a Galois (concept) lattice upon an insertion of a new object.

Example 3.1 (Insertion of object 9). Assume \mathcal{L} is the lattice induced by the object set 12345678 (see Figure 1) and consider 9 as the new object whose intent is $\{o\}' = \operatorname{cdfgh}$. The three categories are:

- $\mathbf{U}(o) = \{c_{\#7}, c_{\#9}\}$,
- $\mathbf{M}(o) = \{c_{\#1}, c_{\#2}, c_{\#4}\}$,

- $\mathbf{G}(o) = \{c_{\#3}, c_{\#5}, c_{\#6}, c_{\#8}, c_{\#10}, c_{\#11}, c_{\#12}\}$.

The set of new concepts (identified by their extents) is: $\{13, 123, 134, 1378, 1346, 12378, 12346\}$.

The trace of the algorithm is given in the following table which provides the intent intersection and the χ^+ image for each concept. Concepts in \mathcal{L}^+ are underlined to avoid confusion with their counterparts in \mathcal{L} .

c	$\mathcal{Q}(c)$	$\chi^+(c)$	Cat.
$c_{\#1}$	\emptyset	$\underline{c_{\#1}}$	mod.
$c_{\#2}$	c	$\underline{c_{\#2}}$	mod.
$c_{\#3}$	d	$\underline{c_{\#3}}$	mod.
$c_{\#4}$	g	$\underline{c_{\#4}}$	mod.
$c_{\#5}$	h	$\underline{c_{\#5}}$	mod.
$c_{\#6}$	c	$\underline{c_{\#2}}$	old
$c_{\#7}$	cd	$\underline{c_{\#14}}$	gen.
$c_{\#8}$	dgh	$\underline{c_{\#8}}$	mod.
$c_{\#9}$	fh	$\underline{c_{\#15}}$	gen.
$c_{\#10}$	cd	$\underline{c_{\#14}}$	old
$c_{\#11}$	$cdgh$	$\underline{c_{\#16}}$	gen.
$c_{\#12}$	$dfgh$	$\underline{c_{\#17}}$	gen.
$c_{\#13}$	$cdfgh$	$\underline{c_{\#18}}$	gen.

To illustrate the way our algorithm proceeds, consider the processing of concept $c_{\#12} = (3, defgh)$ (see Figure 1) when object 9 is added. The value of $\mathcal{Q}(c_{\#12})$ is $dfgh$ whereas *Candidates* contains the images by χ^+ of the upper covers of $c_{\#12}$, i.e., $c_{\#8}$ and $c_{\#9}$: *Candidates* = $\{\underline{c_{\#8}} = (139, dgh), \underline{c_{\#15}} = (359, fh)\}$. Obviously, neither of the intents is as big as $\mathcal{Q}(c_{\#12})$, so $c_{\#12}$ is a maximum, more precisely a (concept) generator. The new concept, $\underline{c_{\#17}}$ is $(39, dfgh)$ and its upper covers are both concepts in *Candidates* (since both are incomparable). Finally, as $\underline{c_{\#8}}$ is in $\mathbf{M}(o)$, its link to $c_{\#12}$ is dropped out.

4 Theoretical foundations

In this section, we provide the foundations of our approach, including an intuitive definition of the generators and their evolution upon transaction insertions.

4.1 Definitions and notations

In this part, we consider only the family of closed itemsets \mathcal{C}^a and its natural order based on set-theoretical inclusion. First, \mathcal{C}^a is a Moore family and as such, induces an equivalence relation on $\mathcal{P}(A)$ where the classes are induced by the closure of an itemset.

Figure 2: The Hasse diagram of the concept (Galois) lattice derived from \mathcal{K} with $O = \{1, 2, 3, \dots, 9\}$.

Definition 7. Given a set $Y \in A$, its closure class is $[Y]_{\mathcal{C}^a} = \{\bar{Y} | \bar{Y}'' = Y''\}$

Obviously, the closed itemsets are the (unique) maximal elements in their own classes, whereas the sets of generator itemsets for each closed itemset are the minima.

For example, the equivalence class of the closed set $bcdgh$ (see Figure 3) contains a set of elements, including the following generators: bg, bh, cg , and ch .

To denote the various generator sets, we shall use the following notations:

- $gen : \mathcal{P}(\mathcal{P}(A)) \rightarrow \mathcal{P}(\mathcal{P}(A))$ assigns the set of all generators to a given Moore family,
- $gen_{\mathcal{C}^a} : \mathcal{C} \rightarrow \mathcal{P}(\mathcal{P}(A))$ assigns to a closed set in a Moore family \mathcal{C}^a , the set of its generators, i.e., the minima of its equivalence class,
- $\Delta gen_{\mathcal{C}_1^a \rightarrow \mathcal{C}_2^a}$ is a shortcut for $gen_{\mathcal{C}_1^a}(Y) - gen_{\mathcal{C}_2^a}(Y)$.

Finally, we shall use a particular notation for the one-to-one correspondence between c-generators and new concepts in :

Definition 8. Given a lattice \mathcal{L} and a new object o , the function $\gamma : \mathbf{G}^+(o) \rightarrow \mathbf{N}^+(o)$ maps each c-generator to its respective new element:

$$\gamma(X, Y) = (X \cup \{o\}, Y \cap o').$$

To avoid confusion between contexts, we shall denote the closure of a set by a dedicated operator rather than $''$. Thus, the closure with respect to the Moore family \mathcal{C}^a will be an operator $\varphi_{\mathcal{C}^a} : \mathcal{P}(A) \rightarrow \mathcal{P}(A)$.

4.2 Structural results

We are interested in the evolution of the set of generators in every concept from \mathcal{C} along the transition from \mathcal{L} to \mathcal{L}^+ . The question is first tackled in a global manner. Thus, we consider the status of a generator from $gen(\mathcal{C}^a)$ in \mathcal{C}^{a+} . A first result states that whenever Y is a generator in \mathcal{C}^a , its status either remains steady or it can change so that Y becomes a closed set in \mathcal{C}^{a+} . In the later case, Y is a *new* closed set, i.e., corresponds to the intent of a concept from $\mathbf{N}^+(o)$. More generally speaking, Y has a different closure in \mathcal{C}^+ , i.e., $\varphi_{\mathcal{C}^a}(Y) \neq \varphi_{\mathcal{C}^{a+}}(Y)$. The examination of all the cases of sets changing their closures after o has been inserted leads to the observation that the old closure is necessarily the intent of a c-generator concept and the new one is the intent of a new concept.

Lemma 1. *For a set $Y \subseteq A$, whenever its closure changes in \mathcal{C}^{a+} , i.e., $\varphi_{\mathcal{C}^a}(Y) \neq \varphi_{\mathcal{C}^{a+}}(Y)$, the closed sets involved may be characterized as follows:*

1. $\varphi_{\mathcal{C}^a}(Y)$ is the intent of a concept from $\mathbf{G}(o)$,
2. $\varphi_{\mathcal{C}^{a+}}(Y)$ is the intent of a concept from $\mathbf{N}^+(o)$.

Proof. The above property could be easily established by first observing that no previously existing closed set could “pick” in \mathcal{C}^{a+} , a member of another equivalence class in \mathcal{C}^a . Otherwise, this would contradict the stability of the lattice order along the insertion. Moreover, only c-generators change their lower covers, so they are good candidates for losing some elements of their equivalence classes from \mathcal{C}^a to \mathcal{C}^{a+} . Such “lost” elements include the respective new closure as well as a possibly empty set of other non-closed sets. To see that no other closed set Y can verify $[Y]_{\mathcal{C}^{a+}} \subset [Y]_{\mathcal{C}^a}$, observe that the old closure of any set that is “picked” by a new closure Y_n is necessarily the intent of the c-generator of the concept $c = (Y'_n, Y_n)$, i.e., $Y_g = Int(\gamma(c))$. Indeed, assume Y is such that $\varphi_{\mathcal{C}^a}(Y) \neq \varphi_{\mathcal{C}^{a+}}(Y)$ and let Y_n be the closure of Y in \mathcal{C}^{a+} . Then, $Y_g = Int(\gamma((Y'_n, Y_n)))$ is the minimal closure from \mathcal{C}^a that includes Y_n , i.e., $Y_g = \varphi_{\mathcal{C}^a}(Y_n)$. Therefore, there cannot be another closure for Y in \mathcal{C}^a than Y_g since $\varphi_{\mathcal{C}^a}(Y)$ should have the following two properties: larger than Y_n and minimal in \mathcal{C}^a . \square

Further to the above property, one may express the evolution of the entire equivalence classes from \mathcal{C}^a to \mathcal{C}^{a+} . In fact, the equivalence class of a new intent is exactly the part of the class of its c-generator

intent which is made up of elements that are themselves included in the new intent. Conversely, the new equivalence class of a c-generator is included in the set difference between the old class and the elements mentioned before, i.e., all those included in the respective new intent. In all other cases, the class remains the same both in \mathcal{C}^a and \mathcal{C}^{a+} .

Corollary 1. *For any $c \in \mathcal{C}^{a+}$, the following holds:*

$$\begin{aligned} c \in \mathbf{G}^+(o) &: [Int(c)]_{\mathcal{C}^{a+}} \subseteq [Int(c)]_{\mathcal{C}^a} - \mathcal{P}(Int(\gamma(c))) \\ c \in \mathbf{N}^+(o) &: [Int(c)]_{\mathcal{C}^{a+}} = [Int(\gamma^{-1}(c))]_{\mathcal{C}^a} \cap \mathcal{P}(Int(c)) \\ else &: [Int(c)]_{\mathcal{C}^{a+}} = [Int(c)]_{\mathcal{C}^a} \end{aligned}$$

Example 4.1. If now we assume that the intent of object 9 is ab (instead of $cdfgh$), then $c = (6, abcd)$ is a c-generator for the new concept $\gamma(c) = (69, ab)$. The equivalence class for the intent of c in \mathcal{C}^a is $[Int(c)]_{\mathcal{C}^a}$ and includes the generators ab and ad while the generator set of $Int(\gamma(c))$ and $Int(c)$ are $\{ab\}$ and $\{abc, ad\}$ respectively.

Another remarkable fact is that any attribute set Y which is minimal in its respective class $[Y]_{\mathcal{C}^a}$ is still minimal in its new class $[Y]_{\mathcal{C}^{a+}}$.

Lemma 2. *For all $Y \in A$, $Y \in \min([Y]_{\mathcal{C}^a})$ implies $Y \in \min([Y]_{\mathcal{C}^{a+}})$.*

Proof. With respect to corollary 1, the only interesting case is when Y moves to a new class in \mathcal{C}^{a+} , which means $\varphi_{\mathcal{C}^{a+}}(Y)$ is the intent of a new concept. Moreover, if $c = (g(\varphi_{\mathcal{C}^{a+}}(Y)), \varphi_{\mathcal{C}^{a+}}(Y))$, then according to the same corollary 1, $Y \in [Int(\gamma^{-1}(c))]_{\mathcal{C}^a}$ since all the elements of the class $[Y]_{\mathcal{C}^{a+}}$ come from the former class. Thus, following the hypothesis of the lemma, $Y \in \min([Int(\gamma^{-1}(c))]_{\mathcal{C}^a})$. Finally, there cannot be other sets in $[Y]_{\mathcal{C}^{a+}}$ that are smaller than Y since all those come from $[Int(\gamma^{-1}(c))]_{\mathcal{C}^a}$ as well and Y is a minimum there. \square

It is noteworthy that the above lemma holds even in case of a non-closed Y becoming closed in \mathcal{C}^{a+} . As a result, one may assert that all the generators in \mathcal{C}^a either remain generators in \mathcal{C}^{a+} or become closed sets, i.e., members of \mathcal{C}^{a+} .

Corollary 2. $gen(\mathcal{C}^a) \subseteq gen(\mathcal{C}^{a+}) \cup \mathcal{C}^{a+}$.

Now that we know that all the generators can only become closed or stay generators, the complementary question comes to the light: where do new generators come from?

First, it is noteworthy that only in the case of a c -generator concept, new generators can emerge in \mathcal{C}^{a+} . In fact, as the elements in the new classes in \mathcal{C}^{a+} , i.e., those corresponding to new intents, preserve their inclusion-based order, any minimal element Y of such a class is minimal in its class in \mathcal{C}^a . Thus, to characterize the new generators, one has to focus on the minima of the c -generator classes in \mathcal{C}^{a+} . The real difference is then a sum of all differences between actual and all minima over the set of all c -generators. Obviously, $gen(\mathcal{C}^{a+}) - gen(\mathcal{C}^a) = \cup_{c \in \mathbf{G}^+(o)} \Delta gen_{\mathcal{C}^{a+} \rightarrow \mathcal{C}^a}(c)$, where, following previous results, $\Delta gen_{\mathcal{C}^{a+} \rightarrow \mathcal{C}^a}(c)$ may be written as $(\min([Int(c)]_{\mathcal{C}^a} - \mathcal{P}(Int(\gamma(c)))) - \min([Int(c)]_{\mathcal{C}^a}))$. We shall now characterize the sets $\Delta gen_{\mathcal{C}^{a+} \rightarrow \mathcal{C}^a}(c)$ where c is in $\mathbf{G}^+(o)$. In fact, we show that all new generators come from former generators for the same c to which an attribute from the difference between the c -generator intent and the new intent ($Int(c) - Int(\gamma(c))$) is added. This difference is called the “face” [?] of $Int(c)$ with respect to $Int(\gamma(c))$.

Property 1. *For any $Y \in \Delta gen_{\mathcal{C}^{a+} \rightarrow \mathcal{C}^a}(c)$ for a given c from \mathcal{L} with c -generator, the following holds:*

$$Y = Y_o \cup \{a\}$$

where $Y_o \in \min([Int(\gamma(c))]_{\mathcal{C}^{a+}})$ and $a \in Int(c) - Int(\gamma(c))$

This property states that every new generator in $\Delta gen_{\mathcal{C}^{a+} \rightarrow \mathcal{C}^a}(c)$ is the union of a generator that went to $\gamma(c)$ and an attribute that belongs to $Int(c)$ but not to $Int(\gamma(c))$.

Proof. First, observe that $Y \in \Delta gen_{\mathcal{C}^{a+} \rightarrow \mathcal{C}^a}(c)$ implies that $Y \in [Int(c)]_{\mathcal{C}^a}$, but $Y \notin \min([Int(c)]_{\mathcal{C}^a})$.

Thus, there is a set Y_n such that $Y_n \subseteq Y$ and $Y_n \in gen_{\mathcal{C}^a}(c)$. Obviously, $Y_n \in gen_{\mathcal{C}^{a+}}(c)$ would be a contradiction. Therefore, Y_n must be a generator that moved to the new concept, i.e., $Y_n \in gen_{\mathcal{C}^{a+}}(\gamma(c))$. Moreover, for any generator Y_s of c that has remained in the same class, i.e., $Y_s \in gen_{\mathcal{C}^a}(c) \cap gen_{\mathcal{C}^{a+}}(c)$, Y_s is incomparable with $Y = Y_n \cup \Delta Y$. In order to show that $|\Delta Y| = 1$, one may observe that $\Delta Y \not\subseteq Int(\gamma(c))$, whereas $\Delta Y \subseteq Int(c)$. Consequently, $\exists a \in \Delta Y$ such that $a \in Int(c)$. To prove $\Delta Y = \{a\}$, recall that $Y = Y_n \cup \Delta Y$ is minimal for its class. Thus, it is enough to show that $Y_a = Y_n \cup \{a\}$ has the same profile as Y in its own class, i.e., it is minimal. The only thing one needs to show is that Y_a is in $[Int(c)]_{\mathcal{C}^{a+}}$, which is trivially true. Thus, no element Y_g could have been less than Y_a in $[Int(c)]_{\mathcal{C}^{a+}}$ since it

would have been a contradiction with $Y \in gen_{\mathcal{C}^{a+}}(c)$. Consequently, Y_a is a minimum and $Y_a = Y$, the latter being a minimum too. \square

5 A scheme for incremental generator construction

The structural results from the previous paragraphs underlie a procedure (see Algorithm 2) which, given a generator concept c and its corresponding new concept \bar{c} computed using Algorithm 1, updates the set of generators associated with the intent of c and identifies the set of generators attached to the intent of \bar{c} .

5.1 Principles of the method

The proposed procedure for generator extraction relies on the properties of a generator [?] of a closet itemset as well as the structural results we defined in the previous section.

```

1: procedure COMPUTE-GENERATORS(In/Out:  $c, \bar{c}$  concepts)
2:
3: for all  $g$  in  $c.gens$  do
4:   if  $g \subseteq \bar{c}.Intent$  then
5:      $\bar{c}.gens \leftarrow \bar{c}.gens \cup \{g\}$ 
6:    $c.gens \leftarrow c.gens - \bar{c}.gens$ 
7: SORT( $\bar{c}.gens$ )
8: for all  $\bar{g}$  in  $\bar{c}.gens$  do
9:    $new-gens \leftarrow \emptyset$ 
10:  for all  $a$  in  $(c.Intent - \bar{c}.Intent)$  do
11:     $gen-cond \leftarrow \mathbf{true}$ 
12:    for all  $g$  in  $c.gens$  do
13:      if  $g \subseteq \bar{g} \cup \{a\}$  then
14:         $gen-cond \leftarrow \mathbf{false}$ 
15:      if  $gen-cond$  then
16:         $new-gens \leftarrow new-gens \cup \{\bar{g} \cup \{a\}\}$ 
17:    $c.gens \leftarrow c.gens \cup new-gens$ 

```

Algorithm 2: Computation of the generators of a new concept and the generators of its c -generator.

The algorithm includes two main tasks: (i) identify the generators of the intent of a new concept \bar{c} from the generators g in $c.gens$ (i.e., the generators of the CI related to concept c that created \bar{c}), and (ii) update the generators of the CI related to concept c by discarding any generator g that is included in the intent of \bar{c} and augmenting any generator \bar{g} (identi-

fied at the first step) with individual items a from $c.Intent - \bar{c}.Intent$ whenever the produced itemset is minimal (i.e., the condition of line 13 does not hold).

5.2 An illustration

Figure 3: The equivalence class of the closet set $bcdgh$ prior to the insertion of object 9 into the context.

The evolution of the equivalence class associated with the closet itemset $bcdgh$ is depicted in Figures 3 and 4. As it may be seen in Figure 3, the generators of $bcdgh$ before the addition of object 9 are $bg, bh, cg,$ and ch . Once object 9 is added, a new concept (19, $cdgh$) is derived from the c-generator (1, $bcdgh$) and the generators attached to its intent $cdgh$ (see Figures 4) are now cg and ch picked from the original equivalence class of $bcdgh$. The generators associated with $bcdgh$ in the new set \mathcal{C}^{a+} are now limited to bg and bh .

5.3 Incremental rule base update

In the following, we will use the Luxenburger basis as an illustration of incremental association rule update. Two procedures are proposed for updating a given rule set when a new object is inserted into the TDB. They can be adapted to construct the informative basis too. Procedure *New-link* generates new rules while procedure *Drop-link* removes obsolete ones. The former is called using the parameters c_1 and c_2 whenever one of the following cases holds:

- c_1 is a c-generator and c_2 is the new generated concept
- both c_1 and c_2 are new concepts

Figure 4: Evolution of the equivalence class of $bcdgh$ when object 9 is added.

- c_1 is a new concept while c_2 is a modified one

Procedure *Drop-link* is called for any pair (c_1, c_2) in \mathcal{L}^+ such that c_1 is a c-generator and c_2 is a modified concept directly covering (i.e., an immediate successor of) c_1 in \mathcal{L} .

```

1: procedure NEW-LINK(In: Concepts  $c_1, c_2$  such that  $c_1$ 
   is an immediate predecessor/subconcept of  $c_2$ ).
2:
3: for all  $g$  in  $c_2.gens$  do
4:    $r \leftarrow g \Rightarrow c_1.intent - g$ 
5:    $r.supp \leftarrow |c_1.extent|/|O|$ 
6:    $r.conf \leftarrow |c_1.extent|/|c_2.extent|$ 
7:    $c_2.rules \leftarrow c_2.rules \cup \{r\}$ 

```

Algorithm 3: New rule identification in the Luxenburger basis.

```

1: procedure DROP-LINK(In: concepts  $c_1, c_2$  such that  $c_1$ 
   is a c-generator and  $c_2$  a modified concept)
2:
3: for all  $r$  in  $c_2.rules$  do
4:   if  $r.conseq \cup r.premise = c_1.intent$  then
5:      $c_2.rules \leftarrow c_2.rules - \{r\}$ 

```

Algorithm 4: Rule elimination in the Luxenburger basis.

As an illustration, let us consider concepts $c_1 = (35, efh)$ (a new concept) and $c_2 = (359, fh)$ (a c-generator) from Figure 2. Since f is the unique generator for fh , then the rule $f \Rightarrow eh$ (supp=2/9, conf=2/3) is added to the existing Luxenburger basis. As (1359, h) is a modified concept that was a

direct successor of $c_1 = (35, efh)$ in \mathcal{L} (see node #5 Figure 1), rule $h \Rightarrow ef$ will be removed.

6 Conclusion

We believe that providing a complete framework for the incremental generation of rule bases by combining appropriate computation methods together with a theoretical framework for compact output production (e.g., rule bases) will increase the efficiency and flexibility of the association rule mining process.

As a starting point for the present study, we have chosen a less complex version of both problems, which only considers adding one individual (transaction) to the data set. This allows a body of available results about lattices to be explored and completed in the definition of an initial framework for *FCI* mining. Indeed, the results presented in this paper already provide a solution for the general problem of dynamic *FCI* mining. However, just as in the lattice case, we are currently generalizing our findings to the multi-transaction context, expecting significant efficiency gains.

In its most general form (i.e., a multi-transaction context), the problem of dynamic *FCI* mining would require the “merge” of two families of *FCIs* corresponding to two TDB that share the same global set of items \mathcal{I} . In terms of Galois lattices, this could be modelled as the assembly of two upper semi-lattices (obtained from both *FCI* families). In a recent work, we have already addressed the problem of assembling entire Galois lattices [24]. The extension of the proposed approach to “truncated” lattices is under investigation.

To the best of our knowledge, the proposed incremental framework for *FCI* and generator extraction pioneers the work on incremental association rule mining. Such a framework can fruitfully be used for exploratory data mining to identify rules that have a specific pattern.

References

- [1] R. Agrawal, H. Mannila, R. Srikant, H. Toivonen, and A.I. Verkamo. Fast Discovery of Association Rules. In U. Fayyad, G. Piatetsky-Shapiro, and P. Smyth, editors, *Advances in Knowledge Discovery and Data Mining*, pages 307–328. AAAI Press, Menlo Park (CA), USA, 1996.
- [2] R. Agrawal and R. Srikant. Fast algorithms for mining association rules. In *Proceedings of the 20th International Conference on Very Large Data Bases (VLDB'94)*, pages 487–499, Santiago, Chile, September 1994.
- [3] N. Ayan, A. Tansel, and M. Arkun. An efficient algorithm to update large itemsets with early pruning. In *Proceedings, KDD-99*, pages 287–291, San Diego (CA), USA, 1999. ACM Press.
- [4] M. Barbut and B. Monjardet. *Ordre et Classification: Algèbre et combinatoire*. Hachette, 1970.
- [5] R.J. Bayardo and R. Agrawal. Mining the Most Interesting Rules. In *Proceedings, KDD-99*, pages 145–154, San Diego (CA), USA, 1999. ACM Press.
- [6] D. W. Cheung, J. Han, V. Ng, and C.Y. Wong. Maintenance of Discovered Association Rules in Large Databases: An Incremental Updating Technique. In *Proceedings, ICDE-96*, pages 106–114, New Orleans (LA), USA, 1996.
- [7] D. W. Cheung, S. D. Lee, and B. Kao. A General Incremental Technique for Maintaining Discovered Association Rules. In *Proceedings, DASFAA-97*, pages 185–194, Melbourne, Australia, 1997.
- [8] B. A. Davey and H. A. Priestley. *Introduction to lattices and order*. Cambridge University Press, 1992.
- [9] R. Feldman, Y. Aumann, A. Amir, and H. Mannila. Efficient Algorithms for Discovering Frequent Sets in Incremental Databases. In *Proceedings, ACM SIGMOD Workshop DMKD'97*, pages 59–70, Tucson (AZ), USA, 1997.
- [10] B. Ganter and R. Wille. *Formal Concept Analysis, Mathematical Foundations*. Springer-Verlag, 1999.
- [11] R. Godin and R. Missaoui. An Incremental Concept Formation Approach for Learning from Databases. *Theoretical Computer Science*, 133:378–419, 1994.
- [12] R. Godin, R. Missaoui, and H. Alaoui. Incremental concept formation algorithms based on galois (concept) lattices. *Computational Intelligence*, 11(2):246–267, 1995.
- [13] J.L. Guigues and V. Duquenne. Familles minimales d'implications informatives résultant d'un tableau de données binaires. *Mathématiques et Sciences Humaines*, 95:5–18, 1986.
- [14] J. Hipp, U. Guentzer, and G. Nakhaeizadeh. Algorithms for Association Rule Mining - A General Survey and Comparison. *SIGKDD Explorations*, 2(1):58–64, 2000.
- [15] M. Luxenburger. Implications partielles dans un contexte. *Mathématiques et Sciences Humaines*, 29(113):35–55, 1991.

- [16] N. Pasquier. Extraction de bases pour les règles d'association à partir des itemsets fermés fréquents. In *Proceedings of the 18th INFORSID'2000*, pages 56–77, Lyon, France, 2000.
- [17] N. Pasquier, Y. Bastide, T. Taouil, and L. Lakhal. Efficient Mining of Association Rules Using Closed Itemset Lattices. *Information Systems*, 24(1):25–46, 1999.
- [18] J. Pei, J. Han, and R. Mao. CLOSET: An Efficient Algorithm for Mining Frequent Closed Itemsets. In *Proceedings, ACM SIGMOD Workshop DMKD'00*, pages 21–30, Dallas (TX), USA, 2000.
- [19] R. Taouil, N. Pasquier, Y. Bastide, and L. Lakhal. Mining bases for association rules using closed sets. In *Proceedings, ICDE-00*, page 307, San Diego (CA), USA, February 2000. IEEE Computer Society.
- [20] S. Thomas, S. Bodagala, K. Alsabti, and S. Ranka. An Efficient Algorithm for the Incremental Updation of Association Rules in Large Databases. In *Proceedings, KDD-97*, pages 263–266, New Port Beach (CA), USA, 1997.
- [21] P. Valtchev. An algorithm for minimal insertion in a type lattice. *Computational Intelligence*, 15(1):63–78, 1999.
- [22] P. Valtchev and R. Missaoui. Building concept (Galois) lattices from parts: generalizing the incremental methods. In H. Delugach and G. Stumme, editors, *Proceedings, ICCS-01*, volume 2120 of *Lecture Notes in Computer Science*, pages 290–303, Stanford (CA), USA, 2001. Springer-Verlag.
- [23] P. Valtchev, R. Missaoui, R. Godin, and M. Meridji. Generating Frequent Itemsets Incrementally: Two Novel Approaches Based On Galois Lattice Theory. *Journal of Experimental & Theoretical Artificial Intelligence*, 14(2-3):115–142, 2002.
- [24] P. Valtchev, R. Missaoui, and P. Lebrun. A partition-based approach towards building Galois (concept) lattices. *Discrete Mathematics*, 256(3):801–829, 2002.
- [25] R. Wille. Restructuring the lattice theory: An approach based on hierarchies of concepts. In I. Rival, editor, *Ordered sets*, pages 445–470, Dordrecht-Boston, 1982. Reidel.
- [26] M.J. Zaki. Generating Non-Redundant Association Rules. In *Proceedings, KDD-00*, pages 34–43, Boston (MA), USA, 2000.
- [27] M.J. Zaki and C.-J. Hsiao. CHARM: An Efficiently Algorithm for Closed Itemset Mining. In R. Grossman, J. Han, V. Kumar, H. Mannila, and R. Motwani, editors, *Proceedings of the 2nd SIAM International Conference on Data Mining (ICDM'02)*, 2002.