Formal Concept Analysis for Knowledge Discovery and Data Mining: The New Challenges

Petko Valtchev[1], Rokia Missaoui[2], and Robert Godin[3]

[1] DIRO, Université de Montréal, C.P. 6128, Succ. "Centre-Ville", Montréal, Québec, Canada, H3C 3J7
[2] Département d'informatique et d'ingénierie, UQO, C.P. 1250, succursale B, Gatineau, Québec, Canada, J8X 3X7
[3] Département d'informatique, UQAM, C.P. 8888, succ. "Centre Ville", Montréal (Qc), Canada, H3C 3P8

Abstract. Data mining is the activity of extracting potentially interesting regularities out of the raw data, which are further transformed within the wider process of knowledge discovery into non-trivial facts intended to support decision making. Formal concept analysis (FCA) offers an appropriate framework for knowledge discovery, whereby our focus here is on its potential for data mining support. Indeed, a variety of methods powered by FCA results have been published and the numbers grow steadily, especially in the association rule mining (ARM) field. However, an analysis of current practices in this latter field indicates that the impact of FCA has not reached its limits, i.e., that a number of situations exist where appropriate FCA-based methods could successfully apply. As a contribution to the understanding of the current needs in ARM, we discuss the existing methods for itemset mining and provide a set of guidelines for the design of novel algorithms. As a first step, we propose on-line methods computing the generator family of a closure system and compare the practical performances to those of existing batch procedures.

1 Introduction

Knowledge discovery in databases (KDD) is the process of discovering useful knowledge from data, possibly stored in a large database or a data warehouse. Data mining is its main step, i.e., the one that consists in extracting potentially interesting regularities out of the raw data. According to Han et Kamber [13], the main challenges faced by researchers on data mining in the late 90s were the specific characteristics of the target datasets, i.e., "large, noisy, of unknown generation laws, of high dimensionality, distributed, etc." that did not allow the conventional methods for data analysis / machine learning to apply. Thus, the key features of data mining tools and methods were efficiency and scalability to large datasets, robustness to missing and incorrect items in data, visualization and exploration of the mining results.

The core of FCA is an approach towards the design of conceptual hierarchies, the concept lattices, from observations organized in a formal context. Therefore, the process of concept formation in FCA is a knowledge discovery from data *par excellence*, whereby the construction of the concept set constitutes the "mining" phase. A key advantage of the FCA-like mining lays in the fact that due to the closure properties only patterns of maximal size are extracted, thus reducing the exploration/interpretation burden for the analyst and, possibly, increasing efficiency.

Among classical mining paradigms, the association rule mining (ARM) has seen the largest number of successful methods that can be qualified as FCA-based [40, 26].

These basically take advantage of the reduced size of the closed pattern set as opposed to the one of plain patterns. As a result, some of the methods from that group are listed among the most efficient ones in the field.

Nowadays, the KDD faces new challenges in more realistic situations, e.g., the dynamicity in databases, the distribution of the datasets over the Web and the resulting need for integration of partial results, the processing of rich data formats (XML documents, DNA sequences, OO data), etc. The ARM discipline is on its front-line since associations are among the most universal patterns that are easily interpreted and may even support other mining activities such as clustering and classification. We believe that FCA can still offer suitable theoretical and algorithmic tools for the resolution of the underlying mining problems.

The goal of the present study is first to put the existing "FCA-aware" work on ARM in the perspective of what are some open problems, e.g., the on-line rule base mining, as well as new research directions such as mining of distributed or structured data. We thus define a set of new challenges by first clarifying the underlying practical motivations together with the existing conventional mining methods that address them. We then show how the related difficulties translate into concrete algorithmic problems within the FCA framework and provide general guidelines for the design of effective mining tools. For some of the problems, solutions are proposed, with particular attention paid to the on-line computation of the generator family of a closure system. Generators are key elements in the FCA-based approach to ARM since they compose several non redundant bases of rules while underlying the computation in several closed itemset (CI) miners. We propose two on-line algorithms that combine closure and generator maintenance and compare practical performances to CLOSE and ACLOSE, two batch CI miners that also rely on generators.

The paper starts with a short recall on formal concept analysis theory (Section 2) and association rule mining problem (Section 3). The relevant work on ARM with closed patterns is then summarized followed by a discussion on the lessons that may be drawn from past experience (Section 3.3). We then suggest translations for the set of algorithmic problems into the FCA domain, before presenting recent algorithmic results on some of the identified problems (Section 5).

2 Background on concepts, lattices and implications

Formal concept analysis (FCA) [9] is a discipline that studies the hierarchical structures induced by a binary relation between a pair of sets. The structure, made up of the closed subsets (see below) ordered by set-theoretical inclusion, satisfies the properties of a complete lattice[4]. It appeared, under different names, in the work of Öre [20], Birkhoff [3], and Barbut and Monjardet [2]. In FCA, it is known as the *concept lattice* [34] of a context.

2.1 FCA basics

FCA considers a binary relation I (*incidence*) over a pair of sets O (*objects*, further denoted by numbers) and A (*attributes*, denoted by lower-case letters). The relation is given by the matrix of its incidence relation (oIa means that object o has the attribute a) which is called a *formal context* or simply *context* (see Figure 1, on the left, for an example). Following standard FCA notations, sets notations are separator-free, e.g., *127* stands for $\{1, 2, 7\}$, and *abdf* for $\{a, b, d, f\}$).

[4] An excellent introduction to partial orders and lattices may be found in [6].

Two set-valued functions, f and g, summarize the links established by the context:

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

For example, w.r.t. the context in Figure 1, $134' = fgh$ and $abc' = 127$. The pair of $'$ functions induces a *Galois connection* [2] between $\mathcal{P}(O)$ and $\mathcal{P}(A)$.

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

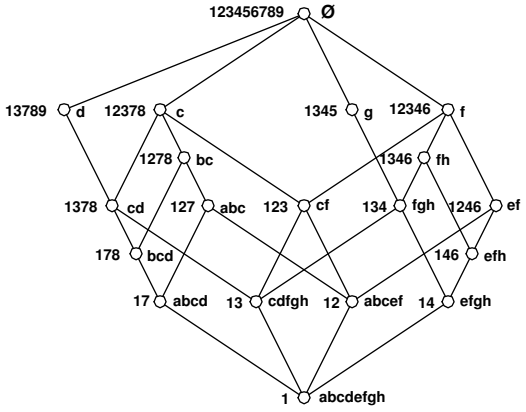

Fig. 1. Left: Context $\mathcal{K}$ (adapted from [9]) with $O = \{1, 2, ..., 9\}$ and $A = \{a, b, ..., h\}$. **Right:** The Hasse diagram of the concept (Galois) lattice derived from $\mathcal{K}$.

Furthermore, the composite operators $''$, which map each of the sets $\mathcal{P}(O)$ and $\mathcal{P}(A)$ into itself (e.g., $36'' = 1346$ and $a'' = abc$), constitute *closure operators* and therefore each of them induces a family of *closed* subsets over the respective power-set ($\mathcal{C}^o_\mathcal{K}$ and $\mathcal{C}^a_\mathcal{K}$). A pair (X, Y), of mutually corresponding subsets, i.e., $X = Y'$ and $Y = X'$, is called a *(formal) concept* in [34] whereby X is referred to as the concept *extent* and Y as the concept *intent* (e.g., $c = (134, fgh)$ is a concept of the context in Figure 1).

The set $\mathcal{C}_\mathcal{K}$ of all concepts of the context $\mathcal{K} = (O, A, I)$ is partially ordered by the set-theoretic inclusion of extents:

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

In fact, the inclusion order induces two *complete lattices* over $\mathcal{C}^o_\mathcal{K}$ and $\mathcal{C}^a_\mathcal{K}$, respectively, which are dually isomorphic with $'$ operators as dual isomorphisms. Both lattices can thus be merged into a unique structure called the *Galois lattice* [2] or the *(formal) concept lattice* of the context $\mathcal{K}$ [9].

Property 1 *The partial order $\mathcal{L}_\mathcal{K} = \langle \mathcal{C}_\mathcal{K}, \leq_\mathcal{K} \rangle$ is a complete lattice with joins and meets as follows:*

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

Figure 1 on the right shows the Hasse diagram of the concept/Galois lattice of the context $\mathcal{K}$. For example, the *join* and the *meet* of the concepts $c_1 = (123, cf)$ and $c_2 = (1246, ef)$ are $(12346, f)$ and $(12, abcef)$, respectively.

2.2 Implications

Dependencies among attributes in the dataset constitute important information and may be the goal of separate analysis process for a context. Indeed, a fact like "the attribute h is met in an object each time the attributes f and g are met" may convey a deep knowledge about the regularities that exist in the domain where the data comes from.

FCA offers a compact representation mode for this type of knowledge, the *implication rules*, which follows the same pattern as its logical counterpart, with two attribute sets, premise and conclusion, $X \rightarrow Y$ ($X, Y \subseteq A$).

Intuitively, an implication is *valid* in a context if none of the objects violates it which basically means $X \rightarrow Y$ is valid iff $Y \subseteq X''$ (e.g., $cg \rightarrow h$ is valid in $\mathcal{K}$ from Figure 1, whereas $cd \rightarrow f$ is not).

Moreover, a rule is *informative* [12] if its premise is minimal and its consequence maximal for set inclusion (e.g., $ab \rightarrow c$ is a valid but non informative rule whereas $a \rightarrow bc$ is informative). The minimal sets that may be put in the premise part of a rule for a given conclusion are called (minimal) *generators* (or *key sets*). Formally, $Y \subseteq A$ is a generator iff $\forall \bar{Y} \subset Y, \bar{Y}'' \subset Y''$.

The set $\Sigma_{\mathcal{K}}$ of all valid implications in a context $\mathcal{K}$ may be of a large size. Even the number of informative rules may be larger than the number of concepts in $\mathcal{C}_{\mathcal{K}}$. Fortunately, a lot of redundant information is stored in these sets in the sense that inference mechanisms close to natural deduction in logics may help retrieve the same content from subsets. A popular system due to Armstrong is made out of three *inference rules* (the *Armstrong axioms* [17]): ($\emptyset \models Y \rightarrow Y$), ($Y \rightarrow Z, U \rightarrow V \models Y \cup U \rightarrow Z \cup V$), and ($Y \rightarrow Z, U \rightarrow V, U \subseteq Z \models Y \rightarrow V$).

The above axioms underly the definition of a number of compact representations, or bases, for $\Sigma_{\mathcal{K}}$. For example, the basis defined by Guigues and Duquenne [12] (the *Duquenne-Guigues basis*) is known to be minimal in the number of rules. It relies on *pseudo-closed* sets: $Y \subseteq A$ is pseudo-closed if it is not closed and for any other pseudo-closed Z, $Z \subset Y$ entails $Z'' \subset Y$ [9]. The Duquenne-Guigues basis of $\mathcal{K}$, denoted $\mathcal{B}_{\mathcal{K}}$, is made out of all the rules $Y \rightarrow Y''$ where Y is pseudo-closed. The entire set $\Sigma_{\mathcal{K}}$ may be obtained as the *implicational hull* of $\mathcal{B}_{\mathcal{K}}$.

Implication rules as closely related to functional dependencies in the database field (see [17]) made their way into data mining, where approximative functional dependencies inspired the association rule mining (see Section 3). In fact, approximative associations correspond to *partial*, i.e., not necessarily overall valid, implications, whereas exact associations are the counterpart of (valid) implications. For example, the rule $ab \rightarrow cd$ is valid in $66,67\%$ of the cases since three objects have ab, but only two of them have also cd. In [16], a basis for partial implications is provided, later called the *Luxenburger cover basis*. It is made out of all rules of the form $\bar{Y} \rightarrow Y - \bar{Y}$ where $\bar{Y}$ and Y are closed attribute sets (intents) and (Y'', Y) is a lower cover of $(\bar{Y}'', \bar{Y})$ in the respective lattice. For example, rules $c \rightarrow b$ and $c \rightarrow d$ are extracted from concept $(12378, c)$ and its lower covers (see Figure 1) as parts of the Luxenburger basis, both valid in 80% of all cases.

2.3 Lattice construction

A variety of efficient algorithms exist for constructing the concept lattice of a binary table [8, 4, 11, 19]. We only recall key algorithms here, interested readers are referred to [15]. A classical distinction between lattice algorithms is made upon two axes: whether or not the lattice order (precedence) relation is computed, on the one hand, and whether or not the context evolves during construction, on the other hand.

Batch construction The algorithm NEXTCLOSURE designed by Ganter [8] uses deep insights into the structure of the concept set to avoid generating concepts more than once. The concepts are thus listed according to the *lectic* order on their intents, each time considering the closure of a current candidate set.

Batch algorithms constructing both concepts and order have been proposed first by Bordat [4] and then by Nourine and Raynaud [19]. The first algorithm uses structural properties of the precedence relation in $\mathcal{L}$ to generate concepts from their upper covers. The second one represents an efficient procedure for constructing a family of open sets[5] and it was shown that it may be used to construct the lattice. The core step of the method is a fast mechanism for the computation of the precedence relation that exploits cardinality properties of neighbor concepts in the lattice.

On-line construction On-line or incremental algorithms fit to the context of rapidly evolving datasets, e.g., dynamic databases. They construct the lattice $\mathcal{L}_{\mathcal{K}}$ starting from a single object o_1 and gradually incorporating any new object o_i into the lattice $\mathcal{L}_{i-1}$ (over a context $\mathcal{K}_{i-1} = (\{o_1, ..., o_{i-1}\}, A, I)$), each time carrying out a set of structural updates. An early incremental procedure was proposed by Godin *et al.* [11], which locally modifies the lattice structure while keeping large parts of the lattice unaltered.

The basic approach follows a fundamental property of the *Galois connection* stating that both families $\mathcal{C}^o$ and $\mathcal{C}^a$ are closed under intersection. Thus, the updates are aimed at the integration into $\mathcal{L}_{i-1}$ of all concepts whose intents correspond to intersections of $\{o_i\}'$ with intents from $\mathcal{C}^a_{i-1}$, which are not themselves in $\mathcal{C}^a_{i-1}$ (such concepts are further called *new*). Basically, three groups of concepts in $\mathcal{L}_{i-1}$ are distinguished in the approach of Godin *et al.*: *generator* concepts (denoted $\mathbf{G}_{i-1}(o)$) which give rise to new concepts and help compute the respective new intents and extents; *old* concepts (denoted $\mathbf{U}_{i-1}(o)$) which remain completely unchanged; and *modified* concepts (labelled $\mathbf{M}_{i-1}(o)$) which evolve by integrating o_i into their extents while their intents remain stable. Here, we suggest a change in the standard terminology: the term *generator* will be further used for minimal generators of concept intents (see previous paragraph) whereas the specific members of $\mathcal{L}_{i-1}$ that play the markers for the insertion of the new concepts will be further referred to as *genitors*[6].

Given a lattice $\mathcal{L}$ and a new object o, the main tasks for a reconstruction algorithm include the identification of the two important sets, $\mathbf{G}(o)$ and $\mathbf{M}(o)$, together with the creation of the new concepts ($\mathbf{N}^+(o)$), and their subsequent integration into the existing lattice structure. These jointly lead to the construction of the target lattice $\mathcal{L}^+$ (where the counterparts of $\mathbf{G}(o)$ and $\mathbf{M}(o)$ are denoted $\mathbf{G}^+(o)$ and $\mathbf{M}^+(o)$, respectively).

In [31], we have presented a detailed study of the lattice substructures relevant to incremental construction with characterizations of the sets $\mathbf{G}(o)$ and $\mathbf{M}(o)$, as well as of the evolution in the precedence relation in the final lattice $\mathcal{L}^+$. A key fact says that concepts from $\mathbf{G}(o)$ and $\mathbf{M}(o)$ are the unique maximal elements in the respective equivalence classes in $\mathcal{L}$ induced by the function $\mathcal{Q} : \mathcal{C} \rightarrow 2^A$ that maps a concept into the intersection of its intent with o' ($\mathcal{Q}(X, Y) = Y \cap \{o\}'$). A further distinction between both sorts of concepts is based on the inclusion of the corresponding intents in o': this holds only for members of $\mathbf{M}(o)$. Another remarkable property that will be used here is that when connecting a new concept c into the lattice under reconstruction the only lower cover of c in $\mathcal{L}^+$ that has a counterpart in $\mathcal{L}$ (i.e., concept with the same intent) is the respective genitor from $\mathbf{M}^+(o)$. The complete set of results was embedded

[5] i.e., the dual to a family of closed sets as in the lattice

[6] The American Heritage Dictionary of the English Language: Fourth Edition. 2000: **genitor**: 1. One who produces or creates. 2. *Anthropology* A natural father or mother.

in a generic scheme for incremental lattice methods as described in [29]. Algorithm 1 provides a viewpoint on the scheme limited to tasks that are relevant here.

```
 1: procedure ADD-OBJECT(In/Out: L = ⟨C, ≤⟩ a lattice; In: o an object)
 2:
 3: for all c in C do
 4:   if c = max([c]_Q) then
 5:     if Intent(c) ⊆ o' then
 6:       Extent(c) ← Extent(c) ∪ {o}
 7:     else
 8:       ĉ ← NEW-CONCEPT(Extent(c) ∪ {o}', Q(c))
 9:       UPDATE-ORDER)(ĉ, c)
10:       C ← C ∪ {c}
```

Algorithm 1: Generic scheme for the insertion of a new object into a concept (Galois) lattice.

To sum up the structural changes in $\mathcal{L}^+$, from all the concepts that behold their intents from $\mathcal{L}$ to $\mathcal{L}^+$, only genitors *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 genitor of c_n.

As database increments rarely happen to be object-wise, i.e., sets of objects are added/deleted at a time, we have investigated a generalization of the incremental approach that considers merging the lattices obtained from two fragments of the same database. The key results are summarized in the following paragraphs.

2.4 Table splits and lattice assembly

FCA has studied the relations between the lattice of a context and those of a set of sub-contexts, further called the *factors*, based on subposition/apposition.

Subposition of contexts and semi-product of lattices Subposition is the horizontal assembly of contexts[7] sharing the same set of attributes [9].

Definition 1 *Let $\mathcal{K}_1 = (O_1, A, I_1)$ and $\mathcal{K}_2 = (O_2, A, I_2)$ be two contexts sharing the attribute set A, the context $\mathcal{K}_3 = (O_1 \dot{\cup} O_2, A, I_1 \dot{\cup} I_2)$ is called their subposition, denoted $\mathcal{K}_3 = \frac{\mathcal{K}_1}{\mathcal{K}_2}$.*

For example, for the context $\mathcal{K} = (O, A, I)$ as given in Figure 1, let $O_1 = \{1, 2, 3, 4\}$ and $O_2 = \{5, 6, 7, 8, 9\}$. The factor lattices corresponding to $\mathcal{K}_1$ and $\mathcal{K}_2$, say $\mathcal{L}_1$ and $\mathcal{L}_2$, are given in Figure 2.

The lattices $\mathcal{L}_1$ and $\mathcal{L}_2$ are related to the lattice $\mathcal{L}_3$ of the subposition context in very specific way. In the extreme case, $\mathcal{L}_3$ will be isomorphic to the direct product of $\mathcal{L}_1$ and $\mathcal{L}_2$, $\mathcal{L}_1 \times \mathcal{L}_2$ (denoted shortly $\mathcal{L}_{1,2}$ in the sequel). However, in the general case, $\mathcal{L}_3$ is only a meet sub-semi-lattice of $\mathcal{L}_{1,2}$. The lattice operator that sends couples of lattices of "subposed" contexts into the lattice of the resulting context is called the *semi-product* in [9].

[7] Apposition is dual assembly upon O; both generalize to n-splits.

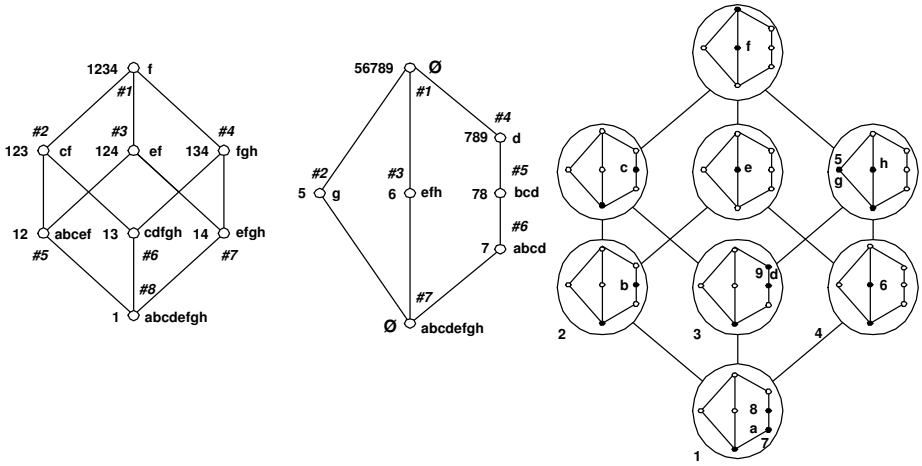

Fig. 2. Left: Partial lattices $\mathcal{L}_1$ and $\mathcal{L}_2$ built from a horizontal decomposition of the context in Figure 1. **Right**: The nested line diagram of the lattice $\mathcal{L}_{1,2}$.

Furthermore, two mappings link the factor lattices to the semi-product and *vice versa*. The composition of two concepts from the factors is based on the fact that the attribute dimension in both remains steady and thus the semi-product may be seen as the merge of two Moore families on A. Therefore, each pair of concepts (c_1, c_2) from $\mathcal{L}_{1,2}$ is sent to the concept c from $\mathcal{L}_3$ such that the intent of c is the intersection of the intents of c_1 and c_2. For example, the couple $(c_{\#7}, c_{\#3})$ (see Figure 2) is sent into the concept $(146, efh)$ since efh is the intersection of the respective intents. The resulting function from $\mathcal{L}_{1,2}$ to $\mathcal{L}_3$ is a surjective order morphism that preserves lattice meets (see [32] for details). Conversely, $\mathcal{L}_3$ is surjectively projected on each factor by the simple projection of the concept intent on the respective subset of A. In other words, the projections of a closed attribute set (intent) from $\mathcal{L}_3$ both on O_1 and O_2 are themselves closed in $\mathcal{K}_1$ and $\mathcal{K}_2$, respectively. For example, the concept $(127, abc)$ is projected to the node $(c_{\#5}, c_{\#6})$.

A framework for the effective transition from a given pair of lattices $\mathcal{L}_1$, $\mathcal{L}_2$ to their semi-product, as studied in [32, 30], is briefly described in the reminder of this section.

Merge of factor lattices Our merge method basically filters the direct product of the factor lattices $\mathcal{L}_1$ and $\mathcal{L}_2$, and keeps only the product nodes that belong to the join/meet sub-semi-lattices isomorphic to the semi-product $\mathcal{L}_3$. These nodes, like in the reference case, are the canonical representative of their respective equivalence classes, which arise through intersection between intents from the factor lattices. In fact, for subposition a global concept (X, Y) is projected into a pair of concepts $((X_1, Y_1), (X_2, Y_2))$ where: $X = X_1 \cup X_2$; $Y = Y_1 \cap Y_2$.

As the function that maps concept pairs from the direct product $\mathcal{L}_{1,2}$ to concepts from $\mathcal{L}$ by preserving intent intersection is not injective, a further property states that the pair $((X_1, Y_1), (X_2, Y_2))$, where $X = X_1 \cup X_2$ is the canonical representative for its intersection class. In fact, it is the maximal node of the direct product among all those which satisfy $Y = Y_1 \cap Y_2$.

The above facts together with further properties characterizing the order in $\mathcal{L}_3$ with respect to that in $\mathcal{L}_{1,2}$ underlie a straightforward algorithm for lattice merges. The algorithm performs a top-down traversal of all combinations of factor concepts in $\mathcal{L}_{1,2}$

with successive canonicity tests for each combination. An overview of the method is provided by the following Algorithm 2.

```
1: procedure MERGE(In: L1, L2 lattices; Out: L3 a lattice)
2:
3: L ← ∅
4: SORT(C1); SORT(C2) {Decreasing order}
5: for all (ci, cj) in L1 × L2 do
6:    I ← Intent(ci) ∩ Intent(cj)
7:    if CANONICAL((ci, cj), E) then
8:       c ← NEW-CONCEPT(Extent(ci) ∪ Extent(cj),I)
9:       UPDATE-ORDER(c, L3)
10:      L3 ← L3 ∪ {c}
```

Algorithm 2: Assembling the global Galois lattice from a pair of partial ones.

It is noteworthy that the canonicity test in the initial MERGE method is based on a comparison of intent intersection on a node (c_i, c_j) (variable I) to the intersection on node's immediate successors (upper covers) in $\mathcal{L}_{1,2}$. For example, the node $(c_{\#7}, c_{\#3})$ is canonical for its class since the extent intersection of both factor concepts, efh, is strictly greater than the intersection on immediate successors $(c_{\#7}, c_{\#1})$, $(c_{\#3}, c_{\#3})$ and $(c_{\#4}, c_{\#3})$ in $\mathcal{L}_{1,2}$ ($\emptyset$, ef, and fh, respectively).

Algorithm 2 is completed to a first-class lattice construction procedure of a *divide-and-conquer* type. The resulting method performs recursive context splits until the basic case of a single column is reached, followed by the reverse merges that end up by computing the lattice of the entire context.

3 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 $\mathcal{I} = \{i_1, i_2, \cdots, i_m\}$ 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 the TDB represented by the context in Figure 1 is 33%[8]. 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 Figure 1 which may be thought of as showing a supermarket database with a sample set of transactions $\mathcal{D} = \{1, \cdots, 9\}$ involving items from the set $\mathcal{I} = \{a, \cdots, h\}$. The itemsets whose support is higher or equal to 40% of $|\mathcal{D}|$ are given in Table 1 hereafter.

3.1 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

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

Itemset	Supp.	Itemset	Supp.	Itemset	Supp.	Itemset	Supp.
b	4	c	5	d	5	f	5
g	4	h	4	e	4	-	-
bc	4	cd	4	fh	4	ef	4

Table 1. The itemsets of support greater than 40%.

$supp(X \cup Y)$ 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 80% 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 (*FI*s) (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 $|\mathcal{I}|$.

3.2 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, FCA-powered approaches have been suggested to that end in [39, 23]. The gain is in producing and storing only a subset of the *FIs*, which is made up of *frequent closed itemsets* (*FCIs*), i.e., the intents of concepts that have their extents above the size-limit. The subset of the concept set corresponding to the FCIs is usually referred to as the *iceberg* (concept) lattice [26].

In [24], an itemset X is considered to be closed 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:

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

This is clearly equivalent to the definition given in Section 2.

A 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 an absolute support of 4.

3.3 Relevant results from the FCI framework

As indicated earlier, association rules can be advantageously generated from *FCIs* rather than *FIs*. 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.

The notion of *generator* of a closed itemset plays a key role in the construction of non redundant rule set as indicated in [23, 25].

Definition 2 *A **generic basis** for exact or implication 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, the concept $(178, bcd)$ (see Figure 1) leads to the generation of rule $bd \rightarrow c$ with a support of 44%.

A similar basis is defined for approximative association rules, in particular as a remedy of the problem of informativeness in the Luxenburger basis. Indeed, the rules in the latter basis are not necessarily *maximally* informative (i.e., the premise is minimized and the consequence is maximized). Thus, the informative basis has been defined in [22], where every rule is maximally informative.

Definition 3 *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 $C1 = (X_1, Y_1)$, $C_2 = (X_2, Y_2)$ and C_1 covers C_2. Support and confidence of a rule r in such basis are $supp(Y_2)$ and $|Y_2'|/|Z'|$ respectively.*

Based on Figure 1, rules such as $b \rightarrow ac$ ($supp = 44\%, conf = 75\%$) and $fg \rightarrow eh$ ($supp = 33\%, conf = 67\%$) are part of the informative basis.

4 Computation of FCI families and rule bases

Like lattice algorithms, ARM methods can be divided into batch and on-line, whereby on-line here covers a broader range of mining settings in which the input data evolves. Moreover, distinctions are made on the target structure: FCI, complete sets of rules, rule bases, etc. Finally, the input format is a criterion too: standard itemsets are opposed to sequences of items or itemlists or to even richer descriptions. In the following, we present a set of methods from the ARM field that either aim at FCIs or address a key issue for our presentation.

4.1 FCI miners

Historically, the first efficient FI mining algorithm is the APRIORI algorithm [1]. It performs a level-wise generation of FIs within the powerset lattice of the items, starting by singleton sets and moving upwards in the lattice levels. At each level, the candidates are generated by joining FIs from the previous level that differ by a single element. Candidates which have at least one non-frequent subset are prunned a priori, i.e., before looking at the database to estimate frequencies.

ACLOSE is probably the first FCI miner. Like APRIORI, it performs level-wise computation within the powerst lattice, but this time based on the generators of the FCIs. Generators replace candidates in the APRIORI framework; they guide the FCI lookup in the database. TITANIC [26] is a descendent of ACLOSE, but relies on advanced features of generators to avoid redundant computation, e.g., cardinality reasoning for closure computation and minimalness checks for the filtering of non-generator sets.

CHARM [40] is another closed pattern miner which generates FCIs in a tree organized by inclusion. Closure and support computation of relies on storage and intersection of *TID-sets* (i.e., the set of transactions per item, the equivalent of concept extent). To speed-up closure computation, it use *diffsets*, the set difference on the TID-sets of a given node and of its unique parent node in the tree.

CLOSET and its recent modification CLOSET+ [33] both generate FCIs as maximal branches of a *FP-tree*, a structure that is basically a prefix tree (or *trie*) augmented with with transversal lists of pointers. The global FP-tree of a database is projected into a set of conditional FP-trees that organize patterns sharing the same suffix. Cardinality resoning is applied to compute the closure of a given branch in the FP-tree.

Computation of various rule bases has been addressed in [23] and in [39]. However, to the best of our knowledge, there is no efficient method for their construction. An interesting insight on rule bases and their effective construction is provided in [14].

Finally, a very recent trend in closure-based mining is the discovery of closed patterns on richer descriptions, i.e., patterns that range over a collection of structured objects such as sequences [21, 37] or graphs [36] and that represent sub-objects common to some of the collection members.

4.2 On-line FI miners

Realistic databases often evolve in time, with regular add or remove of bunches of transactions. On-line mining algorithms were introduced to cope with data evolution at low cost, i.e., without starting from scratch.

Early incremental FI miners were based on the APRIORI framework. FUP [5] updates the set of association rules whenever some new transactions are added. The candidates for the incremental transaction set are generated with respect to their frequencies in the initial database which are in turn deduced from some pre-computed support information for the database. The descendant of the FUP method, FUP-2 admits a larger set of operations on the database, including insertion, removal and modification of transactions. An alternative paradigm for on-line FI mining relies on the notion of *negative border* [18], i.e., the set of all infrequent itemsets that are minimal for inclusion (see [7] and [27] for concrete methods). More recently, the UWEP method was proposed which performs a look-ahead pruning to drop any unfrequent candidate as early as possible.

To the best of our knowledge, the problem of on-line FCI mining has not been addressed within the data mining community. This is mainly due to the fact that along the data evolution, patterns may repeatedly change their status between frequent and infrequent. The underlying difficulties are easily summarized in the one-case, i.e., where a single transaction is added: given the initial and the target families of FCIs, some new FCIs may be produced by genitors that are not themselves frequent enough and therefore lay out of the initial iceberg. Thus, the classical incremental reasoning about genitors and modified concepts holds no more. Besides, the application of algorithms computing incrementally the set of all CIs, to the above task is described in the next section.

4.3 Parallel and distributed FI mining

Parallelism and distribution constitute a yet different computation paradigm that has been largely explored in data mining, in particular for cost reduction purposes (see [38] for a survey). It is noteworthy that most of the strategies for parallel mining rely on a distribution of the dataset among the available processors.

4.4 Observations

Despite the apparent proximity between the respective targets, algorithms for lattice/implication basis construction and association miners are based on diverging principles which impede easy adaptation of FCA methods to the mining tasks. To summarize the distinctions, one could say that ARM methods are designed to work on very large datasets that do not hold in the main memory of a computer and therefore keep the access to the raw data (in the database) to a minimum. For instance, CLOSET only scans the database twice, whereas TITANIC would make a number of scans that equals the size

of the largest generator. In contrast, NEXTCLOSURE will look at the data table on each closure computation, i.e., a prohibitively large number of times.

Moreover, the potentially huge size of the mining results limits the quantity of information that can be stored per FI or FCI. Typically, the itemsets are stored with their support, a number, and rules with their premises, conclusion, support and confidence. Unlike that, in addition to concept intents, lattice algorithms may store extents and order as well, most of the time because these are relied on at a later step of the calculations. Space limits forced the ARM methods to use advanced data structures that enable speedy computation of the basic operations, i.e., set inclusion and intersection, as well as direct access to sub-patterns. For example, prefix-tree-like structures, e.g., FP-trees, are massively used in the compact storage of itemset families, whereas hash tables provide direct access to sets of patterns.

4.5 Lessons learned

Based on our experience both in FCA and KDD, we strongly believe that FCA is a suitable paradigm for KDD [35] since it offers valuable features such as the strong mathematical background, the availability of algorithmic methods that can perform conceptual clustering, implication and association rule generation, the effective framework for data fragmentation (e.g., apposition, concatenation) and the reverse fusion together with the corresponding lattice operations, etc. Moreover, line diagrams and scaling can be seen as tools for visualization/browsing and data preprocessing mechanisms, respectively. Moreover, when focusing on the data mining step and on the particular activity of ARM, basic theoretical results from FCA have been successfully used to reduce the size of the output in both ARM tasks, i.e., FI mining and rule extraction. However, the work done by the FCA community [12, 10, 16, 22] do not seem to have a significant echo in the DM field. This is partly due to the fact that the proposed FCA algorithms do not compete with the leading algorithms in a particular subfield, or, in some cases, to the lack of sufficient empirical evidence on how those methods scale over large datasets.

We firmly believe that the potential of FCA is much bigger than the current status, in particular in what concerns managing data evolution, distributive and parallel computation, and coping with structure in the data items. To put it in more fancy way, in order to attract stronger interest from the general KDD and DM community, FCA should (also) stand for **F**acility, **C**ost-effectiveness and **A**daptability.

Facility is the ease of use, in particular on various data formats (expressiveness). This aspect covers both the availability of user-friendly FCA tools and the compatibility with various mining settings such as constraints, multi-level conceptual hierarchies, or other structure defining domain knowledge.

Cost-effectiveness amounts to insuring high performance and good scalability. It can be achieved through an effective and efficient implementation of DM algorithms, and, whenever applicable, a well-organized decomposition of the mining effort using parallel/distributed architectures.

Adaptability is the faculty to easily change or be changed in order to fit evolving situations such as modified user needs, changes in input data, new system constraints. To reach **a**daptability, FCA should offer on-line mining (e.g., incremental procedures for adding/removing transactions or itemsets), and provide mechanisms for adapting the DM step to the needs of the user.

In the following section, we present a set of results intended to clear the way for new FCA-based mining methods that fit the above requirements.

5 Flexible mining of FCIs and association bases

In the following paragraphs, we summarize past and present work on flexible mining methods based on FCA.

5.1 Mining scenarios and related FCA problems

As a starting point, we have identified a set of mining scenarios that correspond to practical situations and considered the underlying algorithmic problems. The following is a first attempt to draw systematic list of the FCA problems whose solution is of potential interest to the data mining community. In fact, Table 2 presents each situation with the respective operations on contexts, CI/FCI families, and rule bases, whereby algorithmic problems are given unique identifiers pX.

Scenario	**Context**	**CI/FCI**	**Rule bases**
ADD transactions	Subposition	Increment/Assembly ($p1/p2$) of FCI families on ground set A	Merge of Duquenne-Guigues ($p3$), Luxenburger cover ($p4$), informative/generic ($p5$) bases
REMOVE transactions	Horizontal Split	Contraction of an FCI family ($p2'$)	Filtering of Duquenne-Guigues ($p3'$), Luxenburger cover ($p4'$), informative/generic ($p5'$) bases
JOIN database fragments (views)	Apposition	Assembly ($p6$) of disjoint FCI families	Merge of disjoint Duquenne-Guigues ($p7$), Luxenburger cover ($p8$), informative/generic ($p9$) bases
PROJECT a table on a subset of attributes	Vertical Split	Projection of an FCI family ($p6'$)	Factoring of Duquenne-Guigues ($p7'$), Luxenburger cover ($p8'$), informative/generic ($p9'$) bases

Table 2. The translation of mining scenarios into algorithmic problems.

Before going on, it is noteworthy to underline the fact that in order to compute the various bases, one needs the concept intents, the pseudo-closed (Duquenne-Guigues), the generators (generic, informative), order relation (Luxenburger).

We have paid significant attention to the problem $p1$, i.e., the incremental maintenance of the FCI family upon insertions of single transaction. The problem is the basic building block for the study of larger increments, on the one hand, and is close in spirit to the lattice incremental construction, on the other hand. However, as noted in the previous section, substantial differences exist between input structures, in particular in the availability of genitor/modified information for all new concepts that need to be created. Indeed, in the iceberg case, there might be new FCI in the target structure whose genitors are "hidden" below the "sea-level", i.e., the border of the iceberg. Clearly, the discovery of those new FCIs by an updating procedure cannot rely on the standard genitor/modified framework.

The problem of the incremental FCI computation was therefore approached with strategies relying on the computation of all the CI of the database. The approach effectively removes the obstacle created by hidden genitors/modified, but the price to pay is the storage of the entire CI family whereas usually only a tiny part of it will be examined by the analyst. The extra storage increases substantially memory consumption and algorithmic cost of the method. To avoid processing the entire concept set, two separate

techniques have been applied (see [31]). In the first method, GALICIA-P, an index (item x CI) is used to avoid producing all the the empty intersections between an existing concept intent and the new object description. The second algorithm, GALICIA-LBU, uses a bottom-up lattice traversal which is not typical for object increments. Thus, it looks for a quick jump from any concept which is not maximal in its class $[]_{\mathcal{Q}}$, to the effective maximum of that class. The jump mechanism has its own cost since it forces the lattice order to be available and hence to be maintained. When compared to CLOSED, the former algorithm has shown satisfactory performances.

Another sort of problems that we have studied carefully is the maintenance of implication bases upon the insertion of one new object. The following section explains how generators may be efficiently extracted upon single-object increments and how the corresponding procedure generalizes to the lattice assembly case. In contrast, the computation of all pseudo-closed in the incremental settings, has not been tackled so far since very complex.

Problems related to object removal (REMOVE line of the above table) have the opposite effect on the mining results when compared to their ADD counterpart. Consequently, the respective methods could simply reverse the incremental scheme. Therefore, we do not insist on the theoretical foundations of the remove operations.

In the dual case, i.e., with attributes evolving, the reasoning is unfortunately not symmetric. Thus, whereas the iceberg maintenance does not pose serious challenges, unlike in the object case, the computation of all generators is tricky. Moreover, the Duquenne-Guigues basis may be obtained in an indirect manner, as shown in [28]. The underlying algorithm produces both closed and pseudo-closed. Like in the object case, the removal problems just follow the reverse scheme of their ADD counterpart, that is why we do not focus on them here.

5.2 On-line computation of the generator family for $\mathcal{C}^a$

Definitions and notations Given a context $\mathcal{K} = (O, A, I)$, consider the family of its concept intents, $\mathcal{C}^a$, ordered by inclusion and the corresponding equivalence relation induced by the associated closure operator $''$. The latter will be denoted explicitly $\varphi_{\mathcal{C}^a}$ in order to avoid confusion. Moreover, given an object $o \notin O$, a specific mapping from new concepts to their genitors in the augmented lattice $\mathcal{L}^+$ is defined:

$$\gamma : \mathbf{G}^+(o) \rightarrow \mathbf{N}^+(o); \gamma(X, Y) = (X \cup \{o\}, Y \cap o').$$

Our first goal is to clarify the evolution of the set of generators in every concept from $\mathcal{C}$ along the transition from $\mathcal{L}$ to $\mathcal{L}^+$. 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 family,
- $gen_{\mathcal{C}^a} : \mathcal{C} \rightarrow \mathcal{P}(\mathcal{P}(A))$ assigns to a closed set in a 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 $gen_{\mathcal{C}_1^a}(Y) - gen_{\mathcal{C}_2^a}(Y)$.

Structural results The generator evolution 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 latter 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 genitor 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 $\mathbf{N}^+(o)$.*

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 genitor intent which is made out 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:*

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

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 $Y \in \min([Y]_{\mathcal{C}^{a+}})$.*

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 genitor 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" [25] of $Int(c)$ with respect to $Int(\gamma(c))$.

Property 2 *For any $Y \in \Delta gen_{\mathcal{C}^{a+} \rightarrow \mathcal{C}^a}(c)$ for a given c from $\mathcal{L}$ where c is a 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 $Int(\gamma(c))$.

An example In the following, we illustrate the evolution of an equivalence class associated with a closet itemset of a c-generator when a new object is added, and the computation of the set of generators for that *CI* as well as the generators of the new generated closet itemset.

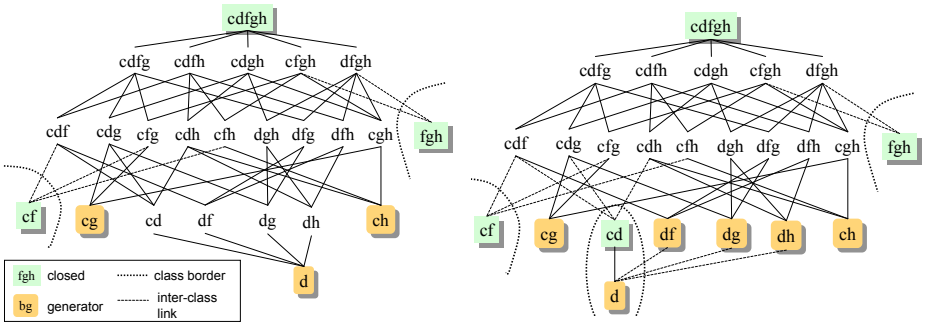

Fig. 3. The equivalence class of the closet set $cdfgh$ in 2^A prior to the insertion of object 7 ($abcd$) into the context $\mathcal{K}_1$ (left) and after the insertion (right).

The evolution of the equivalence class associated with the closet itemset $cdfgh$ attached to the genitor $(13, cdfgh)$ is depicted in Figures 3. As it may be seen in left part of the figure, the generators of $bcdgh$ before the insertion of object 7 are d, cg, and ch. Once object 7 is added, a new concept $(137, cd)$ is derived from the genitor $(13, cdfgh)$ and the generator d moves to the class of its intent cd (see the right part of the figure). This migration leaves three new minima in the class of $cdfgh$: df, dg, and dh. Thus, the set of all generators associated to $cdfgh$ in the new family $\mathcal{C}^{a+}$ increases to $\{cg, ch, df, dg, dh\}$.

5.3 A scheme for incremental generator construction

The structural results from the previous paragraphs underlie a procedure (see Algorithm 3) 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 $\bar{c}$.

Principles of the method The proposed procedure for generator extraction relies on the properties of a generator [25] of a closet itemset as well as the structural results we defined in the previous section.

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

```
 1: procedure COMPUTE-GENERATORS(In/Out: c, c̄ concepts)
 2:
 3: for all g in c.gens do
 4:    if g ⊆ c̄.Intent then
 5:       c̄.gens ← c̄.gens ∪ {g}
 6: c.gens ← c.gens - c̄.gens
 7: SORT(c̄.gens)
 8: for all ḡ in c̄.gens do
 9:    new-gens ← ∅
10:    for all a in (c.Intent - c̄.Intent) do
11:       gen-cond ← true
12:       for all g in c.gens do
13:          if g ⊆ ḡ ∪ {a} then
14:             gen-cond ← false
15:       if gen-cond then
16:          new-gens ← new-gens ∪ {ḡ ∪ {a}}
17:    c.gens ← c.gens ∪ new-gens
```

Algorithm 3: Computation of the generators of a new concept and of its genitor.

to c, the genitor of $\bar{c}$), 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 $\bar{g}$ (identified 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).

5.4 Implementation and performance tests

The INC-GEN algorithm was implemented in `Java`, within the 1.1 version of the Galicia platform[9].

The method has been tested as a stand-alone application and its performances were compared to those of two basic algorithms for FCI mining that rely on generator computation, CLOSE and ACLOSE. The experiments were done on a Windows PC station (Pentium III 996 MHz with 512 MB of RAM) using various subsets of the IBM transaction database T25I10D10K. This dataset is made out of 10 000 transactions over a set of 10 000 items. It is known to be a sparse one, with an average of 25 items per transaction. The graphs drawn in Figure 4 summarize our findings so far. They clearly indicate that the incremental method, INC-GEN, is not competitive as a batch miner for support thresholds of 2% and up. This fact is not surprising given the large number of concepts that the algorithm must examine on each insertion of a new transaction and the even larger number of generators. For example, the first 2 500 transactions produce 900 000 generators in the entire concept set. It is more surprising to see that for supports of 1% and lower, the incremental method scores better than the batch ones, at least for the first quarter of the dataset. This result is even more surprising given the large discrepancy in the number of closed sets and generators produced (ca. 3 000 for CLOSE/ACLOSE versus 900 000 for INC-GEN) and the fact that the overwhelming part of the CPU time is spent on generator computation.

Further tests will be necessary to fully understand the behavior of the INC-GEN) method. However, the current track seems promising, especially if combined with an efficient on-line miner of iceberg lattices.

[9] See the website at: `http://www.iro.umontreal.ca/~galicia`.

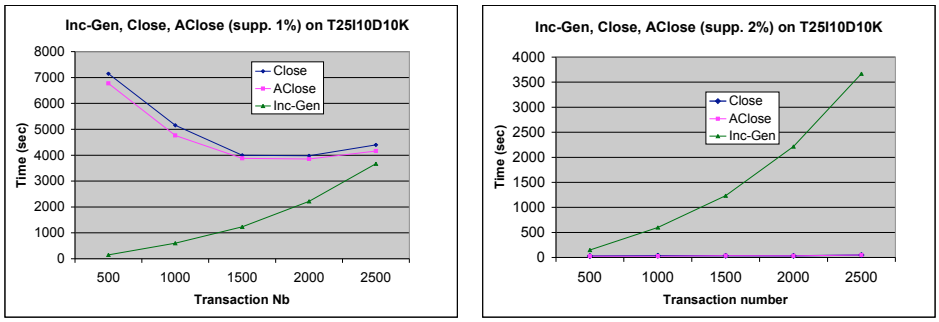

Fig. 4. Evolution of the CPU-time for all three algorithms on transaction batches up to 2500 drawn from the dataset T25I10D10K. CLOSE and ACLOSE were given min. supp. thresholds of 1% (left) and 2% (right).

5.5 Generator evolution along the factor lattice assembly

Generator computation can easily extend to the construction of the subposition-based semi-product of factor lattices $\mathcal{L}_1$ and $\mathcal{L}_2$. First, recall that any concept in the semi-product $\mathcal{L}_3$ is created by a pair of factor concepts that play symmetric role (call them the genitors). We nevertheless adopt an asymmetric view on factors and set $\mathcal{L}_1$ to the initial lattice where generators already exit whereas $\mathcal{L}_2$ is seen as the surrogate for "new" concepts that constitute $\mathcal{L}_3$. Consequently, when such a new concept is detected by the assembly algorithm, its generators will be computed with respect to its genitor in $\mathcal{L}_1$, i.e., the respective component of the canonical representative in $\mathcal{L}_{1,2}$. Obviously, unlike the object-wise increment, there can be several new concepts per genitor (these are exactly the concepts $c_3 = (X, Y)$ from $\mathcal{L}_3$ where $Y^{11} = Intent(c_1)$). The challenge will be to determine their generators without an interference between those.

Next, observe that new concepts corresponding to a genitor c_1 have intents that lay in the equivalence class $[Intent(c_1)]_{11}$. Moreover, they define a partition of this class into finer classes according to the ^{33} closure, whereas a unique new concept has the same intent as c_1.

A safe strategy for consecutive insertions of the new concepts is to insure that whenever a new c_3 is inserted, there is a larger intent above $Intent(c_3)$ in the class $[Intent(c_1)]_{11}$ whose generators are known and can be filtered in the way described in Algorithm 3. The straightforward way to do this is to perform insertions in an order compatible with $\leq_3$, i.e., starting with smaller intents and then proceeding with larger ones. Given the set of the intents of new concepts generated by c_1, say $\mathcal{C}_3^a \cap [Intent(c_1)]_{11}$, the previous condition imposes that at any time the set of already inserted concepts corresponds to an order ideal of that set, provided with inclusion order.

A noteworthy fact about the concrete computation method is that it perfectly fits Algorithms 2 and 3 (with parameters c_1 and c_3). Moreover, all along the insertions, the temporary set of generators that are to be considered for the next insertion is stored at the genitor node within $\mathcal{L}_3$. Indeed, the concept corresponding to c_1 in $\mathcal{L}_3$ (i.e., with the same intent) will be the last one to be created since its intent is the greatest element of the equivalence class. For example, the evolution of the equivalence class associated to $cdfgh$ (from $\mathcal{L}_1$) is depicted in Figure 5. Indeed, the inner loop of Algorithm 2 discovers three new concepts with intents d, cd, and $cdfgh$, respectively, and in this order. These are gradually "inserted" in the class of $cdfgh$: the generators of the new intent are computed and those of $c_1 = (13, cdfgh)$ are updated in $\mathcal{L}_1$. Thus, the new intent d which corresponds to an initial generator of the class forces the creation of

four new generators (cd, df, dg, dh). In contrast, the closed cd merely converts a former generator into a closure.

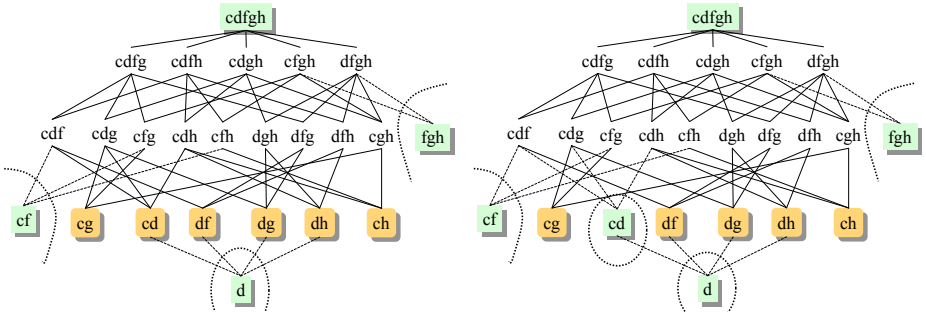

Fig. 5. The evolution of the equivalence class of the closet set $cdf\,gh$ in $\mathcal{P}(A)$ during the assembly process (see initial state in Figure 3, on the left): after the creation of the new closed d (*left*) and after the creation of cd (*right*).

6 Conclusion

A fundamental problem with association rule mining (ARM) is the enormous amount of information that needs to be managed. FCA has already had an important impact on this problem with the introduction of FCI mining and related minimal covers for rules including the Duquenne-Guigues, generic, Luxenberger and informative basis. Several important algorithmic contributions are rooted in these concepts.

Improvement to the flexibility of ARM can be achieved through approaches that adapt to dataset evolution. Incremental approaches for mining CI, FCI and related basis is one direction that we have tackled with some success. Furthermore, this work has revealed some important insights on the properties of these objects from an evolutionary point of view including the more general divide and conquer context fusion problem. Another direction that could contribute to the flexibility of ARM tools is the adaptation to user needs. In the same spirit as OLAP analysis tools, the incremental and fusion approaches are also relevant to this aspect of the problem by providing techniques to dynamically handle several levels of details in the analysis process as expressed by user needs.

Finally, a direction that could contribute to the flexibility of ARM tools but that has had little impact up to now is the ability to handle more expressive data representations (many-valued contexts, objects, scaling, etc.). Much work remains to be done in order to fully exploit the power of FCA in the overall knowledge discovery process.

References

[1] 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.

[2] M. Barbut and B. Monjardet. *Ordre et Classification: Algèbre et combinatoire*. Hachette, 1970.

[3] G. Birkhoff. *Lattice Theory*, volume XXV of *AMS Colloquium Publications*. AMS, 3rd edition, 1967.

[4] J.-P. Bordat. Calcul pratique du treillis de Galois d'une correspondance. *Mathématiques et Sciences Humaines*, 96:31–47, 1986.

[5] 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.

[6] B. A. Davey and H. A. Priestley. *Introduction to lattices and order*. Cambridge University Press, 1992.

[7] 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.

[8] B. Ganter. Two basic algorithms in concept analysis. preprint 831, Technische Hochschule, Darmstadt, 1984.

[9] B. Ganter and R. Wille. *Formal Concept Analysis, Mathematical Foundations*. Springer--Verlag, 1999.

[10] R. Godin and R. Missaoui. An Incremental Concept Formation Approach for Learning from Databases. *Theoretical Computer Science*, 133:378–419, 1994.

[11] R. Godin, R. Missaoui, and H. Alaoui. Incremental concept formation algorithms based on galois (concept) lattices. *Computational Intelligence*, 11(2):246–267, 1995.

[12] J.L. Guigues and V. Duquenne. Familles minimales d'implications informatives résultant d'un tableau de données binaires. *Mathématiques et Sciences Sociales*, 95:5–18, 1986.

[13] J. Han and M. Kamber. *Data Mining : Concepts and Techniques*. Morgan Kaufmann, 2001.

[14] M. Kryszkiewicz. Concise representations of association rules. *Pattern Detection and Discovery*, pages 92–109, 2002.

[15] S. Kuznetsov and S. Ob'edkov. Comparing the performance of algorithms for generating concept lattices. *Journal of Experimental & Theoretical Artificial Intelligence*, 14(2-3):189–216, 2002.

[16] M. Luxenburger. Implications partielles dans un contexte. *Mathématiques et Sciences Humaines*, 29(113):35–55, 1991.

[17] D. Maier. *The theory of Relational Databases*. Computer Science Press, 1983.

[18] H. Mannila, H. Toivonen, and A. Verkamo. Efficient algorithms for discovering association rules. In U. Fayyad and R. Uthurusamy, editors, *Proceedings, AAAI Workshop on Knowledge Discovery in Databases*, pages 181–192, Seattle (WA), USA, 1994. AAAI Press.

[19] L. Nourine and O. Raynaud. A Fast Algorithm for Building Lattices. *Information Processing Letters*, 71:199–204, 1999.

[20] O. Öre. Galois connections. *Transactions of the American Mathematical Society*, 55:493–513, 1944.

[21] F. Pan, G. Cong, A. Tung, J. Yang, and M. Zaki. Carpenter: Finding closed patterns in long biological datasets. In *Proceedings of the 9th International Conference on Knowledge Discovery and Data Mining (KDD'03), Washington (DC), August*, 2003.

[22] 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.

[23] 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.

[24] 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.

[25] J. Pfaltz and C. Taylor. Scientific discovery through iterative transformations of concept lattices. In *Proceedings of the 1st International Workshop on Discrete Mathematics and Data Mining*, pages 65–74, Washington (DC), USA, April 2002.

[26] G. Stumme, R. Taouil, Y. Bastide, N. Pasquier, and L. Lakhal. Computing Iceberg Concept Lattices with Titanic. *Data and Knowledge Engineering*, 42(2):189–222, 2002.

[27] 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.

[28] P. Valtchev and V. Duquenne. Towards scalable divide-and-conquer methods for computing concepts and implications. In E. SanJuan, A. Berry, A. Sigayret, and A. Napoli, editors, *Proceedings of the 4th Intl. Conference Journées de l'Informatique Messine (JIM'03): Knowledge Discovery and Discrete Mathematics, Metz (FR), 3-6 September*, pages 3–15. INRIA, 2003.

[29] P. Valtchev, M. Rouane Hacene, and R. Missaoui. A generic scheme for the design of efficient on-line algorithms for lattices. In B. Ganter A. de Moor, W. Lex, editor, *Proceedings of the 11th Intl. Conference on Conceptual Structures (ICCS'03)*, volume 2746 of *Lecture Notes in Computer Science*, pages 282–295, Berlin (DE), 2003. Springer-Verlag.

[30] P. Valtchev and R. Missaoui. Building concept (Galois) lattices from parts: generalizing the incremental methods. In H. Delugach and G. Stumme, editors, *Proceedings of the ICCS'01*, volume 2120 of *Lecture Notes in Computer Science*, pages 290–303, 2001.

[31] 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.

[32] P. Valtchev, R. Missaoui, and P. Lebrun. A partition-based approach towards building Galois (concept) lattices. *Discrete Mathematics*, 256(3):801–829, 2002.

[33] J. Wang, J. Han, and J. Pei. CLOSET+: Searching for the Best Strategies for Mining Frequent Closed Itemsets. In *In Proceedings of the Ninth ACM SIGKDD International Conference on Knowledge Discovery and Data Mining (KDD'03)*, Washington, DC, USA, 2003.

[34] R. Wille. Restructuring lattice theory: An approach based on hierarchies of concepts. In I. Rival, editor, *Ordered sets*, pages 445–470, Dordrecht-Boston, 1982. Reidel.

[35] R. Wille. Why can concept lattices support knowledge discovery in databases. *Journal of Experimental & Theoretical Artificial Intelligence*, 14(2-3):81–92, 2002.

[36] X. Yan and J. Han. CloseGraph: Mining Closed Frequent Graph Patterns. In *Proceedings of the 9th International Conference on Knowledge Discovery and Data Mining (KDD'03), Washington (DC)*, 2003.

[37] X. Yan, J. Han, and R. Afshar. CloSpan: Mining Closed Sequential Patterns in Large Datasets. In R. Grossman, J. Han, V. Kumar, H. Mannila, and R. Motwani, editors, *Proceedings of the 3rd SIAM International Conference on Data Mining (ICDM'03), San Fransisco (CA)*, 2003.

[38] M.J. Zaki. Parallel and Distributed Association Mining: A Survey. *IEEE Concurency*, 7(4):14–25, december 1999.

[39] M.J. Zaki. Generating Non-Redundant Association Rules. In *Proceedings, KDD-00*, pages 34–43, Boston (MA), USA, 2000.

[40] 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.