Discovering Frequent Closed Itemsets for Association Rules

Nicolas Pasquier, Yves Bastide, Rafik Taouil, and Lotfi Lakhal

Laboratoire d'Informatique (LIMOS)
Université Blaise Pascal - Clermont-Ferrand II
Complexe Scientifique des Cézeaux
24, av. des Landais, 63177 Aubière Cedex France
{pasquier,bastide,taouil,lakhal}@libd1.univ-bpclermont.fr

Abstract. In this paper, we address the problem of finding frequent itemsets in a database. Using the closed itemset lattice framework, we show that this problem can be reduced to the problem of finding frequent closed itemsets. Based on this statement, we can construct efficient data mining algorithms by limiting the search space to the closed itemset lattice rather than the subset lattice. Moreover, we show that the set of all frequent closed itemsets suffices to determine a reduced set of association rules, thus addressing another important data mining problem: limiting the number of rules produced without information loss. We propose a new algorithm, called A-Close, using a closure mechanism to find frequent closed itemsets. We realized experiments to compare our approach to the commonly used frequent itemset search approach. Those experiments showed that our approach is very valuable for dense and/or correlated data that represent an important part of existing databases.

1 Introduction

The discovery of association rules was first introduced in [1]. This task consists in determining relationships between sets of items in very large databases. Agrawal's statement of this problem is the following [1, 2]. Let $\mathcal{I} = \{i_1, i_2, \ldots, i_m\}$ be a set of m items. Let the database $\mathcal{D} = \{t_1, t_2, \ldots, t_n\}$ be a set of n transactions, each one identified by its unique TID. Each transaction t consists of a set of items I from $\mathcal{I}$. If $\|I\| = k$, then I is called a *k-itemset*. An itemset I is *contained* in a transaction $t \in \mathcal{D}$ if $I \subseteq t$. The support of an itemset I is the percentage of transactions in $\mathcal{D}$ containing I. Association rules are of the form $r : I_1 \xrightarrow{c} I_2$, with $I_1, I_2 \subset \mathcal{I}$ and $I_1 \cap I_2 = \emptyset$. Each association rule r has a support defined as $support(r) = support(I_1 \cup I_2)$ and a confidence c defined as $confidence(r) = support(I_1 \cup I_2) \ / \ support(I_1)$. Given the user defined minimum support *minsup* and minimum confidence *minconf* thresholds, the problem of mining association rules can be divided into two sub-problems [1]:

1. Find all *frequent itemsets* in $\mathcal{D}$, i.e. itemsets with support greater or equal to *minsup*.

2. For each frequent itemset I_1 found, generate all association rules $I_2 \xrightarrow{c} I_1 - I_2$ where $I_2 \subset I_1$, with confidence c greater or equal to *minconf*.

Once all frequent itemsets and their support are known, the association rule generation is straightforward. Hence, the problem of mining association rules is reduced to the problem of determining frequent itemsets and their support.

Recent works demonstrated that the frequent itemset discovery is also the key stage in the search for episodes from sequences and in finding keys or inclusion as well as functional dependencies from a relation [12]. All existing algorithms use one of the two following approach: a levelwise [12] bottom-up search [2, 5, 13, 16, 17] or a simultaneous bottom-up and top-down search [3, 10, 20]. Although they are dissimilar, all those algorithms explore the *subset lattice* (itemset lattice) for finding frequent itemsets: they all use the basic properties that *all subsets of a frequent itemset are frequent* and that *all supersets of an infrequent itemset are infrequent* in order to prune elements of the itemset lattice.

In this paper, we propose a new efficient algorithm, called A-Close, for finding frequent closed itemsets and their support in a database. Using a closure mechanism based on the *Galois connection*, we define the *closed itemset lattice* which is a sub-order of the itemset lattice, thus often much smaller. This lattice is closely related to the *Galois lattice* [4, 7] also called *concept lattice* [19]. The closed itemset lattice can be used as a formal framework for discovering frequent itemsets given the basic properties that *the support of an itemset I is equal to the support of its closure* and that *the set of maximal frequent itemsets is identical to the set of maximal frequent closed itemsets.* Then, once A-Close has discovered all frequent closed itemsets and their support, we can directly determine the frequent itemsets and their support. Hence, we reduce the problem of mining association rules to the problem of determining frequent closed itemsets and their support.

Using the set of frequent closed itemsets, we can also directly generate a reduced set of association rules without having to determine all frequent itemsets, thus lowering the algorithm computation cost. Moreover, since there can be thousands of association rules holding in a database, reducing the number of rules produced without information loss is an important problem for the understandability of the result [18]. Empirical evaluations comparing A-Close to an optimized version of Apriori showed that they give nearly always equivalent results for weakly correlated data (such as synthetic data) and that A-Close clearly outperforms Apriori for correlated data (such as statistical or text data).

The rest of the paper is organized as follows. In Section 2, we present the closed itemset lattice. In Section 3, we propose a new model for association rules based on the *Galois connection* and we characterize a reduced set of association rules. In Section 4, we describe the A-Close algorithm. Section 5 gives experimental results on synthetic data[1] and census data using the PUMS file for Kansas USA[2] and Section 6 concludes the paper.

[1] `http://www.almaden.ibm.com/cs/quest/syndata.html`
[2] `ftp://ftp2.cc.ukans.edu/pub/ippbr/census/pums/pums90ks.zip`

2 Closed Itemset Lattices

In this section, we define *data mining context*, *Galois connection*, *Galois closure operators*, *closed itemsets* and *closed itemset lattice*. Interested readers should read [4, 7, 19] for further details on order and lattice theory.

Definition 1 (Data mining context). *A data mining context*[3] *is a triple* $\mathcal{D} = (\mathcal{O}, \mathcal{I}, \mathcal{R})$. $\mathcal{O}$ *and* $\mathcal{I}$ *are finite sets of objects and items respectively.* $\mathcal{R} \subseteq \mathcal{O} \times \mathcal{I}$ *is a binary relation between objects and items. Each couple* $(o, i) \in \mathcal{R}$ *denotes the fact that the object* $o \in \mathcal{O}$ *is related to the item* $i \in \mathcal{I}$.

Definition 2 (Galois connection). *Let* $\mathcal{D} = (\mathcal{O}, \mathcal{I}, \mathcal{R})$ *be a data mining context. For* $O \subseteq \mathcal{O}$ *and* $I \subseteq \mathcal{I}$, *we define:*

$$f(O)\colon 2^{\mathcal{O}} \rightarrow 2^{\mathcal{I}} \qquad f(O) = \{i \in \mathcal{I} \mid \forall o \in O, (o, i) \in \mathcal{R}\}$$

$$g(I)\colon 2^{\mathcal{I}} \rightarrow 2^{\mathcal{O}} \qquad g(I) = \{o \in \mathcal{O} \mid \forall i \in I, (o, i) \in \mathcal{R}\}$$

$f(O)$ *associates with* O *the items common to all objects* $o \in O$ *and* $g(I)$ *associates with* I *the objects related to all items* $i \in I$. *The couple of applications* (f, g) *is a Galois connection between the power set of* $\mathcal{O}$ *(i.e.* $2^{\mathcal{O}}$*) and the power set of* $\mathcal{I}$ *(i.e.* $2^{\mathcal{I}}$*). The following properties hold for all* $I, I_1, I_2 \subseteq \mathcal{I}$ *and* $O, O_1, O_2 \subseteq \mathcal{O}$:

(1) $I_1 \subseteq I_2 \Rightarrow g(I_1) \supseteq g(I_2)$ *(1')* $O_1 \subseteq O_2 \Rightarrow f(O_1) \supseteq f(O_2)$

(2) $O \subseteq g(I) \Longleftrightarrow I \subseteq f(O)$

Definition 3 (Galois closure operators). *The operators* $h = f \circ g$ *in* $2^{\mathcal{I}}$ *and* $h' = g \circ f$ *in* $2^{\mathcal{O}}$ *are Galois closure operators*[4]. *Given the Galois connection* (f, g), *the following properties hold for all* $I, I_1, I_2 \subseteq \mathcal{I}$ *and* $O, O_1, O_2 \subseteq \mathcal{O}$ *[4, 7, 19]:*

Extension :	*(3)* $I \subseteq h(I)$	*(3')* $O \subseteq h'(O)$
Idempotency :	*(4)* $h(h(I)) = h(I)$	*(4')* $h'(h'(O)) = h'(O)$
Monotonicity :	*(5)* $I_1 \subseteq I_2 \Rightarrow h(I_1) \subseteq h(I_2)$	*(5')* $O_1 \subseteq O_2 \Rightarrow h'(O_1) \subseteq h'(O_2)$

Definition 4 (Closed itemsets). *An itemset* $C \subseteq \mathcal{I}$ *from* $\mathcal{D}$ *is a closed itemset iff* $h(C) = C$. *The smallest (minimal) closed itemset containing an itemset* I *is obtained by applying* h *to* I. *We call* $h(I)$ *the closure of* I.

Definition 5 (Closed itemset lattice). *Let* $\mathcal{C}$ *be the set of closed itemsets derived from* $\mathcal{D}$ *using the Galois closure operator* h. *The pair* $\mathcal{L}_{\mathcal{C}} = (\mathcal{C}, \leq)$ *is a complete lattice called closed itemset lattice. The lattice structure implies two properties:*

i) There exists a partial order on the lattice elements such that, for every elements $C_1, C_2 \in \mathcal{L}_{\mathcal{C}}$, $C_1 \leq C_2$, *iff* $C_1 \subseteq C_2$[5].

ii) All subsets of $\mathcal{L}_{\mathcal{C}}$ *have one greatest lower bound, the Join element, and one lowest upper bound, the Meet element.*

---

[3] By extension, we call database a data mining context afterwards.

[4] Here, we use the following notation: $f \circ g(I) = f(g(I))$ and $g \circ f(O) = g(f(O))$.

[5] C_1 is a sub-closed itemset of C_2 and C_2 is a sup-closed itemset of C_1.

Below, we give the definitions of the Join and Meet elements extracted from the basic theorem on Galois (concept) lattices [4, 7, 19]. For all $S \subseteq \mathcal{L}_\mathcal{C}$*:*

$$Join\ (S) = h(\bigcup_{C \in S} C), \qquad Meet\ (S) = \bigcap_{C \in S} C$$

OID	Items
1	A C D
2	B C E
3	A B C E
4	B E
5	A B C E

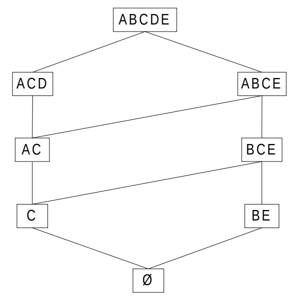


Fig. 1. The data mining context $\mathcal{D}$ and its associated closed itemset lattice.

3 Association Rule Model

In this section, we define *frequent* and *maximal frequent* itemsets and closed itemsets using the Galois connection. We then define *association rules* and *valid association rules*, and we characterise a *reduced set of valid association rules* in a data mining context $\mathcal{D}$.

3.1 Frequent Itemsets

Definition 6 (Itemset support). *Let* $I \subseteq \mathcal{I}$ *be a set of items from* $\mathcal{D}$*. The support count of the itemset* I *in* $\mathcal{D}$ *is:*

$$support(I) = \frac{\|g(I)\|}{\|\mathcal{O}\|}$$

Definition 7 (Frequent itemsets). *The itemset* I *is said to be frequent if the support of* I *in* $\mathcal{D}$ *is at least minsup. The set* L *of frequent itemsets in* $\mathcal{D}$ *is:*

$$L = \{I \subseteq \mathcal{I} \mid support(I) \geq minsup\}$$

Definition 8 (Maximal frequent itemsets). *Let* L *be the set of frequent itemsets. We define the set* M *of maximal frequent itemsets in* $\mathcal{D}$ *as:*

$$M = \{I \in L \mid \nexists I' \in L,\ I \subset I'\}$$

Property 1. All subsets of a frequent itemset are frequent (intuitive in [2]).

Proof. Let $I, I' \subseteq \mathcal{I}$, $I \in L$ and $I' \subseteq I$. According to Property (1) of the Galois connection: $I' \subseteq I \Longrightarrow g(I') \supseteq g(I) \Longrightarrow \text{support}(I') \geq \text{support}(I) \geq minsup$. So, we get: $I' \in L$.

Property 2. All supersets of an infrequent itemset are infrequent (intuitive in [2]).

Proof. Let $I, I' \subseteq \mathcal{I}$, $I' \notin L$ and $I' \subseteq I$. According to Property (1) of the Galois connection: $I \supseteq I' \Longrightarrow g(I) \subseteq g(I') \Longrightarrow \text{support}(I) \leq \text{support}(I') \leq minsup$. So, we get: $I \notin L$.

3.2 Frequent Closed Itemsets

Definition 9 (Frequent closed itemsets). *The closed itemset C is said to be frequent if the support of C in $\mathcal{D}$ is at least minsup. We define the set FC of frequent closed itemsets in $\mathcal{D}$ as:*

$$FC = \{C \subseteq \mathcal{I} \mid C = h(C) \ \wedge support(C) \geq minsup\}$$

Definition 10 (Maximal frequent closed itemsets). *Let FC be the set of frequent closed itemsets. We define the set MC of maximal frequent closed itemsets in $\mathcal{D}$ as:*

$$MC = \{C \in FC \mid \ \nexists C' \in FC,\ C \subset C'\}$$

Property 3. The support of an itemset I is equal to the support of its closure: $support(I) = support(h(I))$.

Proof. Let $I \subseteq \mathcal{I}$ be an itemset. The support of I in $\mathcal{D}$ is: $support(I) = \dfrac{\|g(I)\|}{\|\mathcal{O}\|}$

Now, we consider $h(I)$, the closure of I. Let's show that $h'(g(I)) = g(I)$. We have $g(I) \subseteq h(g(I))$ (extension property of the Galois closure) and $I \subseteq h(I) \Rightarrow g(h(I)) \subseteq g(I)$ (Property (1) of the Galois connection). We deduce that $h'(g(I)) = g(I)$, and therefore we have:

$$support(h(I)) = \frac{\|g(h(I))\|}{\|\mathcal{O}\|} = \frac{\|h'(g(I))\|}{\|\mathcal{O}\|} = \frac{\|g(I)\|}{\|\mathcal{O}\|} = support(I)$$

Property 4. The set of maximal frequent itemsets M is identical to the set of maximal frequent closed itemsets MC.

Proof. It suffices to demonstrate that $\forall I \in M$, I is closed, i.e. $I = h(I)$. Let $I \in M$ be a maximal frequent itemset. According to Property (3) of the Galois connection $I \subseteq h(I)$ and, since I is maximal and $support(h(I)) = support(I) \geq minsup$, we conclude that $I = h(I)$. I is a maximal frequent closed itemset. Since all maximal frequent itemsets are also maximal frequent closed itemsets, we get: $M = MC$.

3.3 Association Rule Semantics

Definition 11 (Association rules). *An association rule is an implication between itemsets of the form* $I_1 \xrightarrow{c} I_2$ *where* $I_1, I_2 \subset \mathcal{I}$ *and* $I_1 \cap I_2 = \emptyset$*. Below, we define the support and confidence* c *of an association rule* $r : I_1 \xrightarrow{c} I_2$ *using the Galois connection:*

$$support(r) = \frac{\|g(I_1 \cup I_2)\|}{\|\mathcal{O}\|}, \qquad confidence(r) = \frac{support(I_1 \cup I_2)}{support(I_1)} = \frac{\|g(I_1 \cup I_2)\|}{\|g(I_1)\|}$$

Definition 12 (Valid association rules). *A valid association rules is an association rules with support and confidence greater or equal to the minsup and minconf thresholds respectively. We define the set* $\mathcal{AR}$ *of valid association rules in* $\mathcal{D}$ *using the set MC of maximal frequent closed itemsets as:*

$$\mathcal{AR}(\mathcal{D}, minsup,\ minconf) = \{r : I_2 \xrightarrow{c} I_1 - I_2,\ I_2 \subset I_1 \mid I_1 \in L = \bigcup_{C \in MC} 2^C \text{ and } confidence(r) \geq minconf\}$$

3.4 Reduced Set of Association Rules

Let $I_1, I_2 \subset \mathcal{I}$ and $I_1 \cap I_2 = \emptyset$. An association rule $r : I_1 \xrightarrow{c} I_2$ is an *exact association rule* if $c = 1$. Then, r is noted $r : I_1 \Rightarrow I_2$. An association rule $r : I_1 \xrightarrow{c} I_2$ where $c < 1$ is called an *approximate association rule*. Let $\mathcal{D}$ be a data mining context.

Definition 13 (Pseudo-closed itemsets). *An itemset* $I \subseteq \mathcal{I}$ *from* $\mathcal{D}$ *is a pseudo-closed itemset iff* $h(I) \neq I$ *and* $\forall I' \subset I$ *such as* I' *is a pseudo-closed itemset, we have* $h(I') \subseteq I$*.*

Theorem 1 (Exact association rules basis [8]). *Let* P *be the set of pseudo-closed itemsets and* $\mathcal{R}$ *the set of exact association rules in* $\mathcal{D}$*. The set* $\mathcal{E} = \{r : I_1 \Rightarrow h(I_1) - I_1 \mid I_1 \in P\}$ *is a basis for all exact association rules.* $\forall r' \in \mathcal{R}$ *where confidence*$(r') = 1 \geq$ *minconf we have* $\mathcal{E} \models r'$*.*

Corollary 1 (Exact valid association rules basis). *Let* FP *be the set of frequent pseudo-closed itemsets in* $\mathcal{D}$*. The set* $\mathcal{BE} = \{r : I_1 \Rightarrow h(I_1) - I_1 \mid I_1 \in FP\}$ *is a basis for all exact valid association rules.* $\forall r' \in \mathcal{AR}$ *where confidence*$(r') = 1$ *we have* $\mathcal{BE} \models r'$*.*

Theorem 2 (Reduced set of approximate association rules [11]). *Let* C *be the set of closed itemsets and* $\mathcal{R}$ *the set of approximate association rules in* $\mathcal{D}$*. The set* $\mathcal{A} = \{r : I_1 \xrightarrow{c} I_2 - I_1 \mid I_2 \subset I_1 \wedge I_1, I_2 \in C\}$ *is a correct reduced set for all approximate association rules.* $\forall r' \in \mathcal{R}$ *where minconf* $\leq$ *confidence*$(r') < 1$ *we have* $\mathcal{A} \models r'$*.*

Corollary 2 (Reduced set of approximate valid association rules). *Let* FC *be the set of frequent closed itemsets in* $\mathcal{D}$*. The set* $\mathcal{BA} = \{r : I_1 \xrightarrow{c} I_2 - I_1 \mid I_2 \subset I_1 \wedge I_1, I_2 \in FC\}$ *is a correct reduced set for all approximate valid assocition rules.* $\forall r' \in \mathcal{AR}$ *where confidence*$(r') \leq 1$ *we have* $\mathcal{BA} \models r'$*.*

4 A-Close Algorithm

In this section, we present our algorithm for finding frequent closed itemsets and their supports in a database. Section 4.1 describes its principle. In Section 4.2 to 4.5, we give the pseudo-close of the algorithm and the sub-functions it uses. Section 4.6 provides an example and the proof of the algorithm correctness.

4.1 A-Close Principle

A closed itemset is a maximal set of items common to a set of objects. For example, in the database $\mathcal{D}$ in Figure 1, the itemset BCE is a closed itemset since it is the maximal set of items common to the objects $\{2,3,5\}$. BCE is called a frequent closed itemset for *minsup* = 2 as support$(BCE) = \|\{2,3,5\}\| = 3 \geq$ *minsup*. In a basket database, this means that 60% of customers (3 customers on a total of 5) purchase *at most* the items B, C and E. The itemset BC is not a closed itemset since it is not a maximal group of items common to some objects: all customers purchasing the items B and C also purchase the item E. The closed itemset lattice of a finite relation (the database) is dually isomorphic to the Galois lattice [4, 7], also called concept lattice [19].

Based on the closed itemset lattice properties (Section 2 and 3), using the result of A-Close we can generate all frequent itemsets from a database $\mathcal{D}$ through the two following phases:

1. Discover all frequent closed itemsets in $\mathcal{D}$, i.e. itemsets that are closed and have support greater or equal to *minsup*.
2. Derive all frequent itemsets from the frequent closed itemsets found in phase 1. That is generate all subsets of the maximal frequent closed itemsets and derive their support from the frequent closed itemset supports.

A different algorithm for finding frequent closed itemsets and algorithms for deriving frequent itemsets and generating valid association rules are presented in [15].

Using the result of A-Close, we can directly generate the reduced set of valid association rules defined in Section 3.4 instead of determining all frequent itemsets. The procedure is the following:

1. Discover all frequent closed itemsets in $\mathcal{D}$.
2. Determine the exact valid association rule basis: determine the pseudo-closed itemsets in $\mathcal{D}$ and then generate all rules $r : I_1 \Rightarrow I_2 - I_1 \mid I_1 \subset I_2$ where I_2 is a frequent closed itemset and I_1 is a frequent pseudo-closed itemset.
3. Construct the reduced set of approximate valid association rules: generate all rules of the form: $r : I_1 \stackrel{c}{\to} I_2 - I_1 \mid I_1 \subset I_2$ where I_1 and I_2 are frequent closed itemsets.

In the two cases, the first phase is the most computationally intensive part. After this phase, no more database pass is necessary and the later phases can be solved easily in a straightforward manner. Indeed, the first phase has given us all information needed by the next ones.

A-Close discovers the frequent closed itemsets as follows. Based on the closed itemset properties, it determines a set of *generators* that will give us all frequent closed itemsets by application of the Galois closure operator h. An itemset p is a generator of a closed itemset c if it is one of the smallest itemsets (there can be more than one) that will determine c using the Galois closure operator: $h(p) = c$. For instance, in the database $\mathcal{D}$ (Figure 1), BC and CE are generators of the closed itemset BCE. The itemsets B, C and E are not generators of BCE since $h(C) = C$ and $h(B) = h(E) = BE$. The itemset BCE is not a generator of itself since it includes BC and CE: BCE is not one of the smallest itemsets for which closure is BCE.

The algorithm constructs the set of generators in a levelwise manner: $(i+1)$-generators[6] are created using i-generators in G_i. Then, their support is counted and the useless generators are pruned. According to their supports and the supports of their i-subsets in G_i, infrequent generators and generators that have the same closure as one of their subsets are deleted from G_{i+1}. In the previous example, the support of the generator BCE is the same as the support of generators BC and CE since they have the same closure (Property 3).

Once all frequent useful generators are found, their closures are determined, giving us the set of all frequent closed itemsets. For reducing the cost of the closure computation when possible, we introduce the following optimization. We determine the first iteration of the algorithm for which a $(i+1)$-generator was pruned because it had the same closure as one of its i-subsets. In all iterations preceding the i^{th} one, the generators created are closed and their closure computation is useless. Hence, we can limit the closure computation to generators of size greater or equal to i. For this purpose, the *level* variable indicates the first iteration for which a generator was pruned by this pruning strategy.

4.2 Discovering Frequent Closed Itemsets

As in the Apriori algorithm, items are sorted in lexicographic order. The pseudo-code for discovering frequent closed itemsets is given in Algorithm 1. The notation is given in Table 1. In each of the iterations that construct the candidate generators, one pass over the database is necessary in order to count the support of the candidate generators. At the end of the algorithm, one more pass is needed for determining the closures of generators that are not closed. If all generators are closed, this pass is not made.

First, the algorithm determines the set G_1 of frequent 1-generators and their support (step 1 to 5). Then, the *level* variable is set to 0 (step 6). In each of the following iterations (step 7 to 9), the AC-Generator function (Section 4.4) is applied to the set of generators G_i, determining the candidate $(i+1)$-generators and their support in G_{i+1} (step 8). This process takes place until G_i is empty. Finally, closures of all generators produced are determined (step 10 to 14). Using the *level* variable, we construct two sets of generators. The set G which contains generators p for which size is less than $level-1$, and so that are closed ($p = h(p)$).

[6] A generator of size i is called an i-generator.

Set	Field	Contains
G_i	*generator*	A generator of size i.
	support	Support count of the generator: $support = count(generator)$
G, G'	*generator*	A generator of size i.
	closure	Closure of the generator: $closure = h(generator)$.
	support	Support count of the generator and its closure: $support = count(closure) = count(generator)$ (Property 3).
FC	*closure*	Frequent closed itemset (closed itemset with support $\geq minsup$).
	support	Support count of the frequent closed itemset.

Table 1. Notation

The set G' which contains generators for which size is at least $level - 1$, among which some are not closed, and so for which closure computation is necessary. The closures of generators in G' are determined by applying the *AC-Generator* function (Section 4.4) to G' (step 15). Then, all frequent closed itemsets have been produced and their support is known (see Theorem 3).

Algorithm 1 A-Close algorithm

```
1)  generators in G_1 ← {1-itemsets};
2)  G_1 ← Support-Count(G_1);
3)  forall generators p ∈ G_1 do begin
4)      if (support(p) < minsup) then delete p from G_1;    // Pruning infrequent
5)  end
6)  level ← 0;
7)  for (i ← 1; G_i.generator ≠ ∅; i++) do begin
8)      G_{i+1} ← AC-Generator(G_i);                       // Creates (i+1)-generators
9)  end
10) if (level > 2) then begin
11)     G ← ⋃{G_j | j < level-1};                          // Those generators are all closed
12)     forall generators p ∈ G do begin
13)         p.closure ← p.generator;
14)     end
15) end
16) if (level ≠ 0) then begin
17)     G' ← ⋃{G_j | j ≥ level-1};          // Some of those generators are not closed
18)     G' ← AC-Closure(G');
19) end
20) Answer FC ← {c.closure,c.support|c ∈ G ∪ G'};
```

4.3 Support-Count Function

The function takes the set G_i of frequent i-generators as argument. It returns the set G_i with, for each generator $p \in G_i$, its support count: $support(p) = \|\{o \in \mathcal{O} \mid p \subseteq f(\{o\})\|$. The pseudo-code of the function is given in Algorithm 2.

Algorithm 2 Support-Count function

```
1) forall objects o ∈ O do begin
2)     G_o ← Subset(G_i.generator,f({o}));      // Generators that are subsets of f({o})
3)     forall generators p ∈ G_o do begin
4)         p.support++;
5)     end
6) end
```

The Subset function quickly determines which generators are contained in an object[7], i.e. generators that are subsets of f({o}). For this purpose, generators are stored in a *prefix-tree* structure derived from the one proposed in [14].

4.4 AC-Generator Function

The function takes the set G_i of frequent i-generators as argument. Based on Lemma 1 and 2, it returns the set G_{i+1} of frequent $(i+1)$-generators. The pseudo-code of the function is given in Algorithm 3.

Lemma 1. *Let I_1, I_2 be two itemsets. We have:*

$$h(I_1 \cup I_2) = h(h(I_1) \cup h(I_2))$$

Proof. Let I_1 and I_2 be two itemsets. According to the extension property of the Galois closure operators:

$$\begin{aligned} I_1 \subseteq h(I_1) \text{ and } I_2 \subseteq h(I_2) &\Longrightarrow I_1 \cup I_2 \subseteq h(I_1) \cup h(I_2) \\ &\Longrightarrow h(I_1 \cup I_2) \subseteq h(h(I_1) \cup h(I_2)) \qquad (1) \end{aligned}$$

Obviously, $I_1 \subseteq I_1 \cup I_2$ and $I_2 \subseteq I_1 \cup I_2$. So $h(I_1) \subseteq h(I_1 \cup I_2)$ and $h(I_2) \subseteq h(I_1 \cup I_2)$. According to the idempotency property of the Galois closure operators:

$$h(h(I_1) \cup h(I_2)) \subseteq h(h(I_1 \cup I_2)) \Longrightarrow h(h(I_1) \cup h(I_2)) \subseteq h(I_1 \cup I_2) \qquad (2)$$

From (1) and (2), we conclude that $h(I_1 \cup I_2) = h(h(I_1) \cup h(I_2))$.

Lemma 2. *Let I_1 be an itemset and I_2 a subset of I_1 where support(I_1) = support(I_2). Then we have $h(I_1) = h(I_2)$ and $\forall I_3 \subseteq \mathcal{I}$, $h(I_1 \cup I_3) = h(I_2 \cup I_3)$.*

Proof. Let I_1, I_2 be two itemsets where $I_2 \subset I_1$ and support(I_1) = support(I_2). Then, we have that $\|g(I_1)\| = \|g(I_2)\|$ and we deduce that $g(I_1) = g(I_2)$. From this, we conclude $f(g(I_1)) = f(g(I_2)) \Longrightarrow h(I_1) = h(I_2)$. Let $I_3 \subseteq \mathcal{I}$ be an itemset. Then according to Lemma 1:

$$h(I_1 \cup I_3) = h(h(I_1) \cup h(I_3)) = h(h(I_2) \cup h(I_3)) = h(I_2 \cup I_3)$$

[7] We say that an itemset I is contained in object o if o is related to all items $i \in I$.

Corollary 3. *Let I be an i-generator and $\mathcal{S} = \{s_1, s_2, \ldots, s_j\}$ a set of $(i-1)$-subsets of I where $\bigcup_{s \in \mathcal{S}} s = I$. If $\exists s \in \mathcal{S}$ such as support(s) = support(I), then $h(I) = h(s)$.*

Proof. Derived from Lemma 2.

The AC-Generator function works as follows. We first apply the combinatorial phase of Apriori-Gen [2] to the set of generators G_i in order to obtain a set of candidate $(i+1)$-generators: two generators of size i in G_i with the same first $i-1$ items are joined, producing a new potential generator of size $i+1$ (step 1 to 4). Then, the potential generators produced that will lead to useless computations (infrequent closed itemsets) or redundancies (frequent closed itemsets already produced) are pruned from G_{i+1} as follows.

First, like in Apriori-Gen, G_{i+1} is pruned by removing every candidate $(i+1)$-generator c such that some i-subset of c is not in G_i (step 8 and 9). Using this strategy, we prune two kinds of itemsets: first, all supersets of infrequent generators (that are also infrequent according to Property 2); second, all generators that have the same support as one of their subset and therefore have the same closure (see Theorem 3). Let's take an example. Suppose that the set of frequent closed itemsets G_2 contains the generators AB, AC. The AC-Generator function will create $ABC = AB \cup AC$ as a new potential generator in G_3 and the first pruning will remove ABC since $BC \notin G_2$.

Next, the supports of the remaining candidate generators in G_{i+1} are determined and, based on Property 2, those with support less than *minsup* are deleted from G_{i+1} (step 7).

The third pruning strategy works as follows. For each candidate generator c in G_{i+1}, we test if the support of one of its i-subsets s is equal to the support of c. In that case, the closure of c will be equal to the closure of s (see Corollary 3), so we remove c from G_{i+1} (step 10 to 13). Let's give another example. Suppose that the final set of generators G_2 contains frequent generators AB, AC, BC and their respective supports $3, 2, 3$. The AC-Generator function will create $ABC = AB \cup AC$ as a new potential generator in G_3 and suppose it determines its support is 2. The third prune step will remove ABC from G_3 since support(ABC) = support(AC). Indeed, we deduce that $closure(ABC) = closure(AC)$ and the computation of the closure of ABC is useless. For the optimization of the generator closure computation in Algorithm 1, we determine the iteration at which the second prune suppressed a generator (variable *level*).

4.5 AC-Closure Function

The AC-Closure function takes the set of frequent generators G, for which closures must be determined, as argument. It updates G with, for each generator $p \in G$, the closed itemset p.closure obtained by applying the closure operator h to p. Algorithm 4 gives the pseudo-code of the function. The method used to compute closures is based on Proposition 1.

Algorithm 3 AC-Generator function

insert into G_{i+1}
select $p.\text{item}_1, p.\text{item}_2, \ldots, p.\text{item}_i, q.\text{item}_i$
from $G_i\ p$, $G_i\ q$
where $p.\text{item}_1 = q.\text{item}_1, \ldots, p.\text{item}_{i-1} = q.\text{item}_{i-1}, p.\text{item}_i < q.\text{item}_i$;
forall candidate generators $c \in G_{i+1}$ **do begin**
  forall i-subsets s of c **do begin**
    if $(s \notin G_i)$ **then delete** c from G_{i+1};
  end
end
$G_{i+1} \leftarrow$ Support-Count(G_{i+1});
forall candidate generators $c \in G_{i+1}$ **do begin**
  if (support(c) < minsup) **then delete** c **from** G_{i+1}; // Pruning infrequent
  else do begin
    forall i-subsets s of c **do begin**
      if (support(s) = support(c)) **then begin**
        delete c from G_{i+1};
        if $(level = 0)$ **then** $level \leftarrow i$; // Iteration number of the first prune
      endif
    end
  end
end
Answer $\leftarrow \bigcup$ {c $\in G_{i+1}$};

Proposition 1. *The closed itemset $h(I)$ corresponding to the closure by h of the itemset I is the intersection of all objects in the database that contain I:*

$$h(I) = \bigcap_{o \in O} \{f(\{o\}) \mid I \subseteq f(\{o\})\}$$

Proof. We define $H = \bigcap_{o \in S} f(\{o\})$ where $S = \{o \in O \mid I \subseteq f(\{o\})\}$. We have $h(I) = f(g(I)) = \bigcap_{o \in g(I)} f(\{o\}) = \bigcap_{o \in S'} f(\{o\})$ where $S' = \{o \in O \mid o \in g(I)\}$. Let's show that $S' = S$:

$$I \subseteq f(\{o\}) \Longleftrightarrow o \in g(I)$$
$$o \in g(I) \Longleftrightarrow I \subseteq f(g(I)) \subseteq f(\{o\})$$

We conclude that $S = S'$, thus $h(I) = H$.

Using Proposition 1, only one database pass is necessary to compute the closures of the generators. The function works as follows. For each object o in $\mathcal{D}$, the set G_o is created (step 2). G_o contains all generators in G that are subsets of the object itemset $f(\{o\})$. Then, for each generator p in G_o, the associated closed itemset p.closure is updated (step 3 to 6). If the object o is the first one containing the generator, p.closure is empty and the object itemset $f(\{o\})$ is assigned to it (step 4). Otherwise, the intersection between p.closure and the object itemset gives the new p.closure (step 5). At the end, the function returns

Algorithm 4 AC-Closure function

```
forall objects o ∈ O do begin
    G_o ← Subset(G.generator, f({o}));        // Generators that are subsets of f({o})
    forall generators p ∈ G_o do begin
        if (p.closure = ∅) then p.closure ← f({o});
        else p.closure ← p.closure ∩ f({o});
    end
end
Answer ← ⋃ {p ∈ G | ∄p' ∈ G, closure(p')=closure(p)};
```

for each generator p in G, the closed itemset p.closure corresponding to the intersection of all objects containing p.

4.6 Example and Correctness

Figure 2 gives the execution of A-Close for a minimum support of 2 (40%) on the data mining context $\mathcal{D}$ given in Figure 1. First, the algorithm determines the set G_1 of 1-generators and their support (step 1 and 2), and the infrequent generator D is deleted form G_1 (step 3 to 5). Then, generators in G_2 are determined by applying the AC-Generator function to G_1 (step 8): the 2-generators are created by union of generators in G_1, their support is determined and the three pruning strategies are applied. Generators AC and BE are pruned since support(AC) = support(A) and support(BE) = support(B), and the *level* variable is set to 2.

Calling AC-Generator with G_2 produces 3-generators in G_3. The only generator created in G_3 is ABE since only AB and AE have the same first item. The three pruning strategies are applied and the second one removes ABE form G_3 as $BE \notin G_2$. Then, G_3 is empty and the iterative construction of sets G_i terminates (the loop in step 7 to 9 stops).

The sets G and G' are constructed using the *level* variable (step 10 and 11): G is empty and G' contains generators from G_1 and G_2. The closure function AC-Closure is applied to G' and the closures of all generators in G' are determined (step 15). Finally, duplicates closures are removed from G' by AC-Closure and the result is returned to the set FC which therefore contains AC,BE,C,$ABCE$ and BCE, that are all frequent closed itemsets in $\mathcal{D}$.

Lemma 3. *For $p \subseteq \mathcal{I}$ such as $\|p\| > 1$, if $p \notin G_{\|p\|}$ and support(p) $\geq$ minsup then $\exists s_1, s_2 \subseteq \mathcal{I}$, $s_1 \subset s_2 \subseteq p$ and $\|s_1\| = \|s_2\| - 1$ such as $h(s_1) = h(s_2)$ and $s_1 \in G_{\|s_1\|}$.*

Proof. We show this using a recurrence. For $\|p\| = 2$, we have $p = s_2$ and $\exists s_1 \in G_1 \mid s_1 \subset s_2$ and support(s_1) = support$(s_2) \Longrightarrow h(s_1) = h(s_2)$ (Lemma 3 is obvious). Then, supposing that Lemma 3 is true for $\|p\| = i$, let's show that it is true for $\|p\| = i + 1$. Let $p \subseteq \mathcal{I} \mid \|p\| = i + 1$ and $p \notin G_{\|p\|}$. There are two possible cases:
(1) $\exists p' \subset p \mid \|p'\| = i$ and $p' \notin G_{\|p'\|}$
(2) $\exists p' \subset p \mid \|p'\| = i$ and $p' \in G_{\|p'\|}$ and support(p) = support$(p') \Longrightarrow h(p) =$

Support-Count $\longrightarrow$

G_1

Generator	Support
{A}	3
{B}	4
{C}	4
{D}	1
{E}	4

Pruning infrequent generators $\longrightarrow$

G_1

Generator	Support
{A}	3
{B}	4
{C}	4
{E}	4

AC-Generator $\longrightarrow$

G_2

Generator	Support
{AB}	2
{AC}	3
{AE}	2
{BC}	3
{BE}	4
{CE}	3

Pruning $\longrightarrow$

G_2

Generator	Support
{AB}	2
{AE}	2
{BC}	3
{CE}	3

AC-Closure $\longrightarrow$

G'

Generator	Closure	Support
{A}	{AC}	3
{B}	{BE}	4
{C}	{C}	4
{E}	{BE}	4
{AB}	{ABCE}	2
{AE}	{ABCE}	2
{BC}	{BCE}	3
{CE}	{BCE}	3

Pruning $\longrightarrow$

$Answer:\ FC$

Closure	Support
{AC}	3
{BE}	4
{C}	4
{ABCE}	2
{BCE}	3

Fig. 2. A-Close frequent closed itemset discovery for $minsup = 2$ (40%)

$h(p')$ (Lemma 2)
If (1) then according to the recurrence hypothesis, $\exists s_1 \subset s_2 \subseteq p' \subset p$ such as $h(s_1) = h(s_2)$ and $s_1 \in G_{\|s_1\|}$. If (2) then we identify s_1 to p' and s_2 to p.

Theorem 3. *The A-Close algorithm generates all frequent closed itemsets.*

Proof. Using a recurrence, we show that $\forall p \subseteq \mathcal{I} \mid \text{support}(p) \geq \text{minsup}$ we have $h(p) \in FC$. We first demonstrate the property for the 1-itemsets: $\forall p \subseteq \mathcal{I}$ where $\|p\| = 1$, if $\text{support}(p) \geq \text{minsup}$ then $p \in G_1 \Rightarrow h(p) \in FC$. Let's suppose that $\forall p \subseteq \mathcal{I}$ such as $\|p\| = i$ we have $h(p) \in FC$. We then demonstrate that $\forall p \subseteq \mathcal{I}$ where $\|p\| = i+1$ we have $h(p) \in FC$. If $p \in G_{\|p\|}$ then $h(p) \in FC$. Else, if $p \notin G_{\|p\|}$ and according to Lemma 3, we have: $\exists s_1 \subset s_2 \subseteq p \mid s_1 \in G_{\|s_1\|}$ and $h(s_1) = h(s_2)$. Now $h(p) = h(s_2 \cup p - s_2) = h(s_1 \cup p - s_2)$ and $\|s_1 \cup p - s_2\| = i$, therefore in conformity with the recurrence hypothesis we conclude that $h(s_1 \cup p - s_2) \in FC$ and so $h(p) \in FC$.

5 Experimental Results

We implemented the Apriori and A-Close algorithms in C++, both using the same prefix-tree structure that improves Apriori efficiency. Experiments were realized on a 43P240 bi-processor IBM Power-PC running AIX 4.1.5 with a CPU clock rate of 166 MHz, 1GB of main memory and a 9GB disk. Each execution uses only one processor (the application was single-threaded) and was allowed a maximum of 128MB.

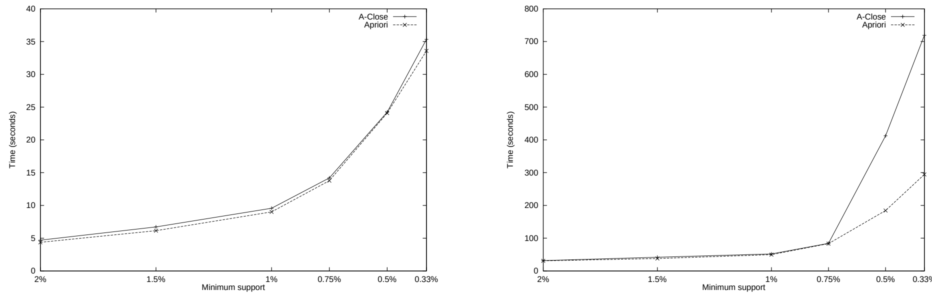
Test Data We used two kinds of datasets: synthetic data, that simulate market basket data, and census data, that are typical statistical data. The synthetic datasets were generated using the program described in [2]. The census data were extracted from the Kansas 1990 PUMS file (Public Use Microdata Samples), in the same way as [5] for the PUMS file of Washington (unavailable through Internet at the time of the experiments). Unlike in [5] though, we did not put an upper bound on the support, as this distorts each algorithm results in different ways. We therefore took smaller datasets containing the first 10,000 persons.

Parameter	T10I4D100K	T20I6D100K	C20D10K	C73D10K
Average size of the objects	10	20	20	73
Total number of items	1000	1000	386	2178
Number of objects	100K	100K	10K	10K
Average size of the maximal poten- -tially frequent itemsets	4	6	-	-

Table 2. Notation

Results on Synthetic Data Figure 3 shows the execution times of Apriori and A-Close on the datasets T10I4D100K and T20I6D100K. We can observe that both algorithms always give similar results except for executions with *minsup* = 0.5% and 0.33% on T20I6D100. This similitude comes from the fact that data are weakly correlated and sparse in such datasets. Hence, the sets of generators in A-Close and frequent itemsets in Apriori are identical, and the closure mechanism does not help in jumping iterations. In the two cases where Apriori outperforms A-Close, there was in the 4^{th} iteration a generator that has been pruned because it had the same support as one of its subsets. As a consequence, A-Close determined closures of all generators with size greater or equal than 3.

Results on Census Data Experiments were conducted on the two census datasets using different *minsup* ranges to get meaningful response times and to accommodate with the memory space limit. Results for the C20D10K and C73D10K datasets are plotted on Figure 4 and 5 respectively. A-Close always significantly outperforms Apriori, for execution times as well as number of database passes. Here, contrarily to the experiments on synthetic data, the differences between execution times can be measured in minutes for C20D10K and in hours for



Execution times on T10I4D100K   Execution times on T20I6D100K

Fig. 3. Performance of Apriori and A-Close on synthetic data

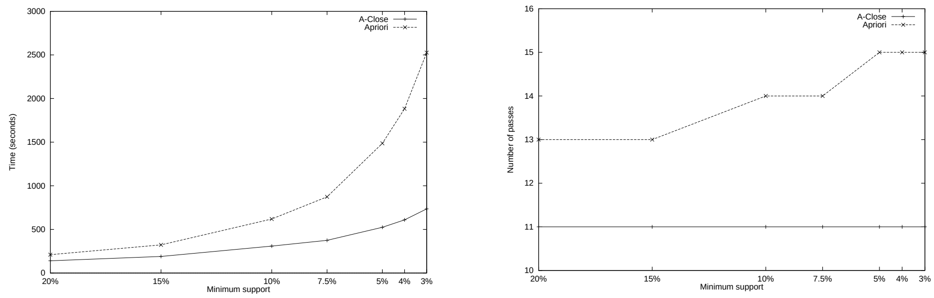
C73D10K. It should furthermore be noted that Apriori could not be run for *minsup* lower than 3% on C20D10K and lower than 70% on C73D10K as it exceeds the memory limit. Census datasets are typical of statistical databases: highly correlated and dense data. Many items being extremely popular, this leads to a huge number of frequent itemsets from which few are closed.

Scale up Properties on Census Data We finally examined how Apriori and A-Close behave as the object size is increased in census data. The number of objects was fixed to 10,000 and the *minsup* level was set to 10%. The object size varied from 10 (281 total items) up to 24 (408 total items). Apriori could not be run for higher object sizes. Results are shown in Figure 6. We can see here that, the scale up properties of A-Close are far better than those of Apriori.

6 Conclusion

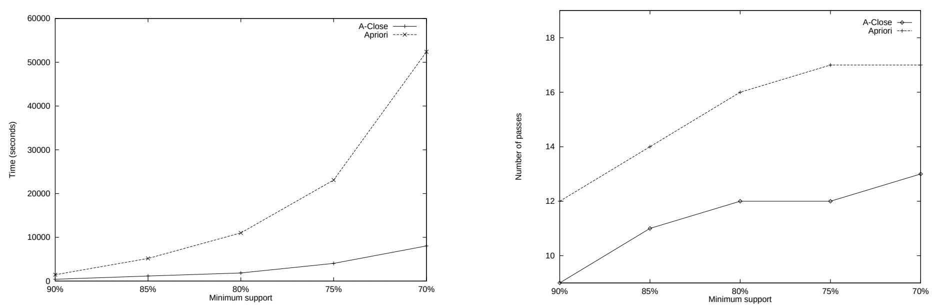
We presented a new algorithm, called A-Close, for discovering frequent closed itemsets in large databases. This algorithm is based on the pruning of the closed itemset lattice instead of the itemset lattice, which is the commonly used approach. This lattice being a sub-order of the itemset lattice, for many datasets, the number of itemsets considered will be significantly reduced. Given the set of frequent closed itemsets and their support, we showed that we can either deduce all frequent itemsets, or construct a reduced set of valid association rules needless the search for frequent itemsets.

We realized experiments in order to compare our approach to the itemset lattice exploration approach. We implemented A-Close and an optimized version of Apriori using prefix-trees. The choice of Apriori leads form the fact that, in practice, it remains one of the most general and powerful algorithms. Those experiments showed that A-Close is very efficient for mining dense and/or correlated data (such as statistical data): on such datasets, the number of itemsets considered and the number of database passes made are significantly reduced



Execution times   Number of database passes

Fig. 4. Performance of Apriori and A-Close on census data C20D10K



Execution times   Number of database passes

Fig. 5. Performance of Apriori and A-Close on census data C73D10K

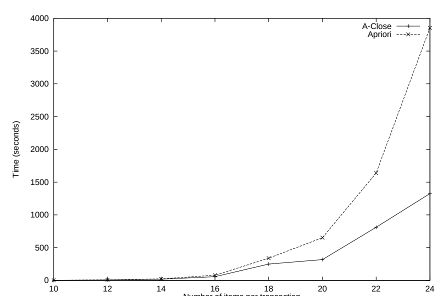


Fig. 6. Scale-up properties of Apriori and A-Close on census data

compared to those Apriori needs. They also showed that A-Close leads to equivalent performances of the two algorithms for weakly correlated data (such as synthetic data) in which many generators are closed. This leads from the adaptive characteristic of A-Close that consists in determining the first iteration for which it is necessary to compute closures of generators. Such a way, we avoid A-Close many useless closure computations.

We think these results are very interesting since dense and/or correlated data represent an important part of all existing data, and since mining such data is considered as very difficult. Statistical, text, biological and medical data are examples of such correlated data. Supermarket data are weakly correlated and quite sparse, but experimental results showed that mining such data is considerably less difficult than mining correlated data. In the first case, executions take some minutes at most whereas in the second case, executions sometimes take several hours.

Moreover, A-Close gives an efficient unsupervised classification technic: the closed itemset lattice of an order is dually isomorphic to the Dedekind-MacNeille completion of an order [7], which is the smallest lattice associated with an order. The closest work is Ganter's algorithm [9] which works only in main memory. This feature is very interesting since unsupervised classification is another important problem in data mining [6] and in machine learning.

References

1. R. Agrawal, T. Imielinski, and A. Swami. Mining association rules between sets of items in large databases. *Proceedings of the ACM SIGMOD Int'l Conference on Management of Data*, pages 207–216, May 1993.
2. R. Agrawal and R. Srikant. Fast algorithms for mining association rules. *Proceedings of the 20th Int'l Conference on Very Large Data Bases*, pages 478–499, June 1994. Expanded version in IBM Research Report RJ9839.
3. R. J. Bayardo. Efficiently mining long patterns from databases. *Proceedings of the ACM SIGMOD Int'l Conference on Management of Data*, pages 85–93, June 1998.
4. G. Birkhoff. Lattices theory. In *Coll. Pub. XXV*, volume 25. American Mathematical Society, 1967. Third edition.
5. S. Brin, R. Motwani, J. D. Ullman, and S. Tsur. Dynamic itemset counting and implication rules for market basket data. *Proceedings of the ACM SIGMOD Int'l Conference on Management of Data*, pages 255–264, May 1997.
6. M.-S. Chen, J. Han, and P. S. Yu. Data mining: An overview from a database perspective. *IEEE Transactions on Knowledge and Data Engineering*, 8(6):866–883, December 1996.
7. B. A. Davey and H. A. Priestley. *Introduction to Lattices and Order.* Cambridge University Press, 1994. Fourth edition.
8. V. Duquenne and L.-L. Guigues. Famille minimale d'implication informatives résultant d'un tableau de données binaires. *Math. Sci. Hum.*, 24(95):5–18, 1986.
9. B. Ganter and K. Reuter. Finding all closed sets: A general approach. In *Order*, pages 283–290. Kluwer Academic Publishers, 1991.
10. D. Lin and Z. M. Kedem. Pincer-search: A new algorithm for discovering the maximum frequent set. *Proceedings of the 6th Int'l Conference on Extending Database Technology*, pages 105–119, March 1998.

11. M. Luxenburger. Implications partielles dans un contexte. *Math. Inf. Sci. Hum.*, 29(113):35–55, 1991.
12. H. Mannila and H. Toivonen. Levelwise search and borders of theories in knowledge discovery. *Data Mining and Knowledge Discovery*, 1(3):241–258, 1997.
13. H. Mannila, H. Toivonen, and A. I. Verkamo. Efficient algorithms for discovering association rules. *Proceedings of the AAAI Workshop on Knowledge Discovery in Databases*, pages 181–192, July 1994.
14. A. M. Mueller. Fast sequential and parallel algorithms for association rules mining: A comparison. Technical report, Faculty of the Graduate School of The University of Maryland, 1995.
15. N. Pasquier, Y. Bastide, R. Taouil, and L. Lakhal. Pruning closed itemset lattices for association rules. *Proceedings of the BDA French Conference on Advanced Databases*, October 1998. To appear.
16. A. Savasere, E. Omiecinski, and S. Navathe. An efficient algorithm for mining association rules in larges databases. *Proceedings of the 21th Int'l Conference on Very Large Data Bases*, pages 432–444, September 1995.
17. H. Toivonen. Sampling large databases for association rules. *Proceedings of the 22nd Int'l Conference on Very Large Data Bases*, pages 134–145, September 1996.
18. H. Toivonen, M. Klemettinen, P. Ronkainen, K. Hatonen, and H. Mannila. Pruning and grouping discovered association rules. *ECML-95 Workshop on Statistics, Machine Learning, and Knowledge Discovery in Databases*, pages 47–52, April 1995.
19. R. Wille. Concept lattices and conceptual knowledge systems. *Computers and Mathematics with Applications*, 23:493–515, 1992.
20. M. J. Zaki, S. Parthasarathy, M. Ogihara, and W. Li. New algorithms for fast discovery of association rules. *Proceedings of the 3rd Int'l Conference on Knowledge Discovery in Databases*, pages 283–286, August 1997.