Frequent Regular Itemset Mining

Salvatore Ruggieri
Dipartimento di Informatica, Università di Pisa
Largo B. Pontecorvo 3, 56127 Pisa, Italy
ruggieri@di.unipi.it

ABSTRACT

Concise representations of frequent itemsets sacrifice readability and direct interpretability by a data analyst of the concise patterns extracted. In this paper, we introduce an extension of itemsets, called regular, with an immediate semantics and interpretability, and a conciseness comparable to closed itemsets. Regular itemsets allow for specifying that an item may or may not be present; that any subset of an itemset may be present; and that any non-empty subset of an itemset may be present. We devise a procedure, called **RegularMine**, for mining a set of regular itemsets that is a concise representation of frequent itemsets. The procedure computes a covering, in terms of regular itemsets, of the frequent itemsets in the class of equivalence of a closed one. We report experimental results on several standard dense and sparse datasets that validate the proposed approach.

Categories and Subject Descriptors

H.2.8 [**Database Management**]: Database Applications-Data Mining

General Terms

Algorithms

Keywords

Concise Representations, Closed and Free Itemsets

1. INTRODUCTION

The intended objective of concise representations is to alleviate the problems due to extracting, storing and post-processing a huge amount of frequent patterns. They sacrifice readability and direct interpretability by a data analyst in favor of a compact, lossless representation, where itemsets whose support is derivable from others are pruned away. Closed itemsets [2] are a concise representation of frequent itemsets, yet not the only one [4, 5, 6, 7, 12, 13, 16], surely the most well-known. Let us explain by an example what we intend for "sacrificing readability" in the case of closed itemsets. Consider the transaction database in Figure 1.

tid	transaction
1	$abcde$
2	$abcd$
3	b
4	ac

cover	closed	free	regular
$\{1,2\}$	$abcd$	d ba	$d\{abc\}^\star$
		bc	$b\{ac\}^+$
$\{1,2,3\}$	b	b	b
$\{1,2,4\}$	ac	a c	$\{ac\}^+$

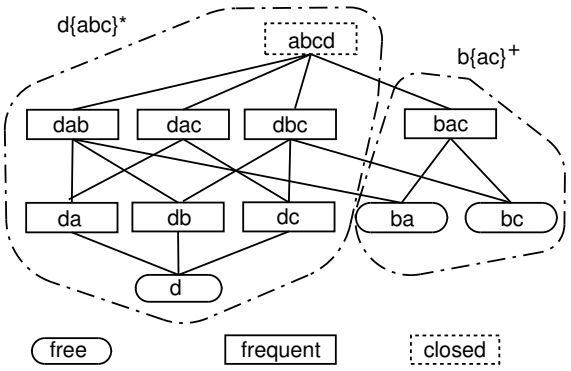

Figure 1: A sample transaction database, the θ-equivalence class of $abcd$ and its covering.

For a minimum support of 2, there is a total of 15 frequent itemsets, but only 3 frequent closed itemsets: $C_1 = abcd$ with support 2; $C_2 = b$ with support 3; and $C_3 = ac$ with support 3. Closed itemsets (and concise representations, in general) are able to answer *frequency* queries such as "is the itemset X frequent?" and, if it is, "what is the support of X?". For instance, for $X = abc$, we look at closed itemsets that include X. If there is one (C_1 includes it), then X is frequent. In such a case, the support of X is the maximal support of the closed itemsets that include it (only C_1, hence the support of X is 2).

Assume now that a data analyst has focussed her attention on itemsets with support of 2. Would it be reasonable to prompt her with C_1? On the one side, it is the only closed itemset with support 2, hence it concisely represents all itemsets the analyst is interested in. On the other side, however, the conciseness itself is the source of interpretability problems. Once prompted with C_1 the data analyst has to figure out which itemsets are represented by C_1, or in other words, what its *semantics* is. The answer is hardly useful:

KDD'10, July 25–28, 2010, Washington, DC, USA.

all subsets of C_1 except the subsets of closed itemsets whose support is greater than the support of C_1 (technically, the θ-equivalence class of C_1). Such an answer highlights that a closed itemset in isolation makes no sense for the analyst, because the itemsets it denotes, its semantics, is undeterminable. The only practical way of presenting the semantics of C_1 to the analyst seems then by enumerating the 11 itemsets in its θ-equivalence class, as shown in Figure 1. But this means loosing the so-much-acclaimed compactness property of concise representations.

In this paper, we introduce an extension of itemsets, called regular, with an immediate semantics and interpretability, and a conciseness comparable to closed itemsets. Regular itemsets allow for specifying that an item a may or may not be present, using the notation $a?$; that any subset of an itemset may be present, using the notation $\{a_1, \ldots, a_h\}^\star$; and that any non-empty subset of an itemset may be present, using the notation $\{a_1, \ldots, a_k\}^+$. We devise a procedure, called **RegularMine**, for mining a set of regular itemsets that is a concise representation of frequent itemsets. The procedure computes a covering, in terms of disjoint regular itemsets, of the θ-equivalence class of a frequent closed itemset. For the example C_1 above, we obtain two regular itemsets, namely $b\{ac\}^+$ and $d\{abc\}^\star$. They are a cover because every frequent itemset equivalent to C_1 belongs to the semantics of one of them. They are disjoint because no frequent itemset belongs to the semantics of both. For the example C_2 above, we obtain b; and for C_3 we obtain $\{ac\}^+$. We report experimental results on several standard dense and sparse datasets that show how the size of the concise representation using regular itemsets is in most cases very close to the size of closed itemsets.

The paper is organized as follows. In Section 2, we set up the notation and recall basic definitions about frequent itemsets, θ-equivalence, closed itemsets, free sets and concise representations. The syntax and semantics of regular itemsets is introduced in Section 3. In Section 4 we propose the mining procedure **RegularMine**, which is experimented in Section 5. Finally, we discuss related work in Section 6 and then summarize the contribution of the paper.

2. BASIC DEFINITIONS

2.1 Frequent Itemsets

Let $\mathcal{I}$ be a finite set of literals called *items*. We use metavariables $a, b, c, \ldots$ to denote items. A set $X = \{a_1, \ldots, a_k\} \subseteq \mathcal{I}$ is called an *itemset*, or a k-itemset if it contains k items. $X \cup Y$ is abbreviated to X, Y.

A *transaction* over $\mathcal{I}$ is a pair $T = (tid, X)$ where tid is a transaction identifier and X is an intemset. A *transaction database* $\mathcal{D}$ over $\mathcal{I}$ is a set of transactions over $\mathcal{I}$. From now on, we omit $\mathcal{I}$ and $\mathcal{D}$ when clear from the context.

A transaction $T = (tid, X)$ *supports* an itemset I if $X \subseteq I$. The *cover* of I is the set of transactions that support I:

$$cover(I) = \{tid \mid (tid, X) \in \mathcal{D}, I \subseteq X\}.$$

The *support* of I is the number of transactions in its cover: $support(I) = |cover(I)|$. An itemset is called *frequent* if its support is no less than a given minimum threshold *minsupp*. The set of frequent itemsets is defined as:

$$\mathcal{F} = \{X \subseteq \mathcal{I} \mid support(X) \geq minsupp\}.$$

2.2 Equivalence, Free and Closed Itemsets

Bastide et al. [2] introduced the relation θ between frequent itemsets supported by the same set of transactions, hence *equivalent* (and indistinguishable) with respect to the transaction database. Given two itemsets X and Y, we write $X\theta Y$, and say that X is θ-equivalent to Y, when $cover(X) = cover(Y)$. θ is an equivalence relation, namely it is reflexive, symmetric and transitive. The class of θ-equivalence of an itemset X consists of all itemsets θ-equivalent to X:

$$[X] = \{Y \subseteq \mathcal{I} \mid X\theta Y\}.$$

Minimal elements (with respect to set inclusion) of a θ-equivalence class are called *free itemsets* [4] or *generators* [2]. Formally, X is free if $X \in Min\,[X]$, where the mimimal function is defined as: $Min\, S = \{X \in S \mid \nexists Y \in S.Y \subset X\}$. It turns out that X is free iff there is no $Y \subset X$ such that $support(Y) = support(X)$. Another useful property of free itemsets is anti-monotonicity [2, 4], namely all subsets of a free itemset are free.

Maximal elements (with respect to set inclusion) of a θ-equivalence class are called *closed itemsets*. Formally, X is closed if $X \in Max\,[X]$, where the maximal function is defined as: $Max\, S = \{X \in S \mid \nexists Y \in S.Y \supset X\}$. It turns out that X is closed iff there is no $Y \supset X$ such that $support(Y) = support(X)$. Given $Y \in [X]$, it is immediate to observe that the itemset X, Y belongs to $[X]$. This implies that $Max\,[X]$ is a singleton, namely there is a bijection between classes of θ-equivalence and closed itemsets.

2.3 Concise Representations

The number of frequent itemsets can be so large that no efficient algorithm exists to enumerate all of them. This case occurs, for instance, for very low minimum support thresholds and for highly correlated data. A concise representation [7, 15] of frequent itemsets is a lossless representation, typically consisting in a subset of $\mathcal{F}$, that addresses the problem. By lossless representation, one means that starting from it: (1) frequent itemsets can be enumerated; and (2) given an itemset it can be decided whether it is frequent or not, and, if it is frequent, its support can be derived.

The set $\mathcal{CS}$ of frequent closed itemsets is a concise representation [17]. An itemset X is frequent iff there exists $Y \in \mathcal{CS}$ such that $X \subseteq Y$. In such a case, the support of X can be calculated as: $max\{support(Y) \mid X \subseteq Y \wedge Y \in \mathcal{CS}\}$.

The set $\mathcal{FS}$ of frequent free itemsets is not a concise representation. The pair of $\mathcal{FS}$ and of the free itemsets in the negative border of $\mathcal{FS}$ is a concise representation [4, 14]. The latter element in the pair is essential to determine whether an itemset X is frequent or not. If frequent, the support of X can be calculated as: $min\{support(Y) \mid X \supseteq Y \wedge Y \in \mathcal{FS}\}$.

3. REGULAR ITEMSETS

In this section, we introduce an extended formulation of itemsets, whose main purpose is to trade-off conciseness and interpretability. The basic idea consists of introducing special items that model cases when: an item a may or may not appear, and this will be denoted by $a?$; any subset of items $a_1, \ldots, a_h$ may appear, and this will be denoted by $\{a_1, \ldots, a_h\}^\star$; any non-empty subset of items $a_1, \ldots, a_k$ may appear, and this will be denoted by $\{a_1, \ldots, a_k\}^+$.

We start by extending the syntax of items and itemsets along this line.

$$\frac{\{a_1,\ldots,a_h\}^\star}{a_1?,\ldots,a_h?}\mathbf{N1} \qquad \frac{\{a\}^+}{a}\mathbf{N2} \qquad \frac{a,a?}{a}\mathbf{N3} \qquad \frac{a,\{a,X\}^+}{a,X^\star}\mathbf{N4} \qquad \frac{a?,\{a,X\}^+}{\{a,X\}^+}\mathbf{N5}$$

(a) For the normal form of extended itemsets.

$$\frac{R,X,Y^- \quad Y\cap X\neq\emptyset}{R,X,(Y\setminus X)^-}\mathbf{S1} \qquad \frac{R,X,Z^\star \quad Z\cap X\neq\emptyset}{R,X,(Z\setminus X)^\star}\mathbf{S2} \qquad \frac{R,\emptyset^-}{\mathbf{fail}}\mathbf{S3}$$

$$\frac{R,\{a\}^- \quad a\notin R}{R\setminus\{a?\}[\{a,X\}^-\rightarrow X^\star]}\mathbf{S4} \qquad \frac{R,\{a,Y\}^- \quad a\notin R \quad Y\neq\emptyset}{R\setminus\{a?\}[\{a,X\}^-\rightarrow X^\star],Y^\star \quad R\setminus\{a?\}[\{a,X\}^-\rightarrow X^-],a,Y^-}\mathbf{S5}$$

(b) For splitting non-compositional itemsets.

$$\frac{R \quad R,a}{R,a?}\mathbf{M1} \qquad \frac{R \quad R,Y^+}{R,Y^\star}\mathbf{M2} \qquad \frac{R,b,a? \quad R,a}{R,\{a,b\}^+}\mathbf{M3} \qquad \frac{R,Y^+,a? \quad R,a}{R,\{a,Y\}^+}\mathbf{M4}$$

$$\frac{R,Y^+ \quad R,a,Y^\star}{R,\{a,Y\}^+}\mathbf{M5} \qquad \frac{R,Y^+ \quad R,Z^+,Y^\star}{R,\{Z,Y\}^+}\mathbf{M6}$$

(c) For merging extended itemsets.

Figure 2: Rewriting rules. $R[X\rightarrow Y]$ means that every extended item matching X in R is replaced by Y.

3.1 Extended Itemsets

An *extended item* is defined by the following grammar:

$$E ::= a \mid a? \mid \{a_1,\ldots,a_h\}^\star \mid \{a_1,\ldots,a_k\}^+$$

where a, a_i's are items, $h\geq 0$ and $k>0$. We use e,f,g as meta-variables for extended items. Let $\mathcal{J}$ be the (finite) set of extended items. A *extended itemset* is a subset $R\subseteq\mathcal{J}$.

EXAMPLE 3.1. *The extended itemset $ab\{cd\}^\star$ represents the itemsets where a and b must necessarily appear, while c and d may or may not appear. The intended meaning of $ab\{cd\}^\star$ is then the set of itemsets $\{ab,abc,abd,abcd\}$.*

The extended itemset $ab?\{cd\}^+$ represents the itemsets where a must necessarily appear, b may or may not, and at least one between c and d appears. Its intended meaning is then $\{ac,ad,acd,abc,abd,abcd\}$.

We now formalize the intuitions underlying the example by providing a semantics mapping an extended item/itemset to the set of itemsets it denotes. The semantics $s_e()$ for extended items is defined as follows:

$$\begin{aligned} s_e(a) &= \{\{a\}\} \\ s_e(a?) &= \{\{a\},\emptyset\} \\ s_e(\{a_1,\ldots,a_h\}^\star) &= \{X \mid X\subseteq\{a_1,\ldots,a_h\}\} \\ s_e(\{a_1,\ldots,a_k\}^+) &= \{X \mid X\subseteq\{a_1,\ldots,a_k\}, X\neq\emptyset\}. \end{aligned}$$

The semantics $s()$ for extended itemsets is defined as follows:

$$s(e_1,\ldots,e_n)=\{\cup_{i=1\ldots n}X_i \mid X_i\in s(e_i), i=1\ldots n\}. \quad (1)$$

Notice that the semantics $s()$ is and-compositional, namely $s(R_1,R_2)=f(s(R_1),s(R_2))$ for some function $f()$ (actually, for $f(S_1,S_2)=\{X\cup Y \mid X\in S_1, Y\in S_2)\}$). This means that the meaning of an extended itemset can be obtained by looking (only) at the meaning of its parts. As discussed in the introduction, closed itemsets and other concise representations do not have such a property. Let us consider now some syntactic issues. Since the following holds:

$$s(\{a_1,\ldots,a_h\}^\star)=s(a_1?,\ldots,a_h?),$$

the * operator can be seen as syntactic sugar. In this sense, we will not make any further distinction between the * and the ? operators, e.g., when stating that $a?$ belongs to $b\{ac\}^\star$. Analogously, since $s(\{a\}^+)=s(a)$, we can rewrite singleton itemsets over the $^+$ operator into an item.

EXAMPLE 3.2. *According to the definition of extended itemsets, an item can appear more than once, e.g., as in $ab\{ac\}^+b?$. However, the following identities:*

$$s(a,a?)=s(a) \quad s(a,\{a,X\}^+)=s(a,X^\star)$$
$$s(a?,\{a,X\}^+)=s(\{a,X\}^+)$$

allow for removing duplicates. The extended item above is equivalent to $abc?$.

The rewriting rules describes so far are reported in Figure 2 (a) as rules $N1-N5$. We say that an extended itemset is in *normal form* if no rule can further apply. Irrespectively of the order the rules are applied, an extended itemset R can be rewritten in one and only one extended itemset in normal form, which we call the normal form of R.

3.2 Regular Itemsets

The notion of extended itemset does not take into account the cover nor the support of the itemsets in its semantics.

EXAMPLE 3.3. *Consider the transaction database $\mathcal{D}=\{(1,ab),(2,a)\}$, and the extended itemset $R=ab?$.*

Two itemsets belong to $s(R)$, namely a and ab. However, $cover(a)=\{1,2\}\neq\{1\}=cover(ab)$.

Extended itemsets are then relevant to the frequent itemset mining problem only when they denote itemsets with a common cover.

DEFINITION 3.4. *An extended itemset R is said* regular *if for every $X, Y \in s(R)$ we have that $cover(X) = cover(Y)$.*

Using the notation of θ-equivalence classes, R is regular if $s(R) \subseteq [X]$ for some itemset X. An equivalent formulation consists of requiring that all itemsets in the semantics of R have the same support.

LEMMA 3.5. *An extended itemset R is regular iff for every $X, Y \in s(R)$ we have that $support(X) = support(Y)$.*

PROOF. The only-if part is immediate, since $cover(X) = cover(Y)$ implies $support(X) = support(Y)$. Consider the if-part. Let C be the itemset obtained from R by replacing $a?$ with a, and $\{a_1, \ldots, a_k\}^+$ with $a_1, \ldots, a_k$. C is the maximal itemset in $s(R)$, hence $X \subseteq C$ and $Y \subseteq C$, and then $cover(X) \supseteq cover(C)$ and $cover(Y) \supseteq cover(C)$. Since by hypothesis $support(X) = support(C) = support(Y)$, we conclude that $cover(X) = cover(C) = cover(Y)$. □

For a regular itemset R, we define $cover(R) = cover(X)$, where X is any itemset in $s(R)$. Also, we extend the notion of support as follows: $support(R) = |cover(R)|$. Finally, we say that a regular itemset R is *frequent* if $support(R) \geq minsupp$.

EXAMPLE 3.6. *Consider the transaction database of Figure 1. The extended itemset $ab\{cd\}^\star$ from Example 3.1 is regular, and its cover is $\{1, 2\}$. The extended itemset $ab?\{cd\}^+$ is not regular since, e.g., $cover(ac) = \{1, 2, 4\} \neq \{1, 2\} = cover(abc)$.*

3.3 Concise Representations

Frequent regular itemsets are natural candidates to be adopted for a concise representation of frequent itemsets. First, a single regular itemset R represents a possibly large set $s(R)$ of itemsets. Second, the interpretation of an extended item in R is straightforward, even for non-technical data analysts: $a?$ means a may appear; $\{a_1, \ldots, a_k\}^+$ means that at least one among $a_1, \ldots, a_k$ must be present. Third, the and-compositionality property makes it possible to figure out the semantics $s(R)$ by looking at extended items in R only. We therefore formalize the notion of concise representation to the case of regular itemsets.

DEFINITION 3.7. *A finite set of regular itemsets $\mathcal{R}$ is a concise representation of the set $\mathcal{F}$ of frequent itemsets if:*

(a) $\cup_{R \in \mathcal{R}} s(R) = \mathcal{F}$, *and*

(b) *for every pair $R_1 \neq R_2 \in \mathcal{R}$, $s(R_1) \cap s(R_2) = \emptyset$.*

Intuitively, given an itemset X: (a) means that X is frequent iff X belongs to the semantics of some $R \in \mathcal{R}$; and (b) means that if X is frequent, there exists one and only one such an R, hence $support(X)$ can be obtained as $support(R)$.

A natural question arise at this stage. How large is a concise representation $\mathcal{R}$? Since the semantics of a regular itemset is included in the class of θ-equivalence of a closed itemset, we conclude that $|\mathcal{CS}| \leq |\mathcal{R}|$ is a lower bound. As for upper bounds, at this stage we cannot draw any conclusion (apart from the trivial one $|\mathcal{R}| \leq |\mathcal{F}|$), but later on we will show that, in practice, there exist concise representations whose size is close to $|\mathcal{CS}|$ and abundantly smaller than the number of frequent free sets $|\mathcal{FS}|$.

4. MINING REGULAR ITEMSETS

We tackle the problem of mining a concise representation by looking for a set of (pairwise disjoint) regular itemsets that cover all the itemsets in a θ-equivalence class. We start with a simple observation.

LEMMA 4.1. *Let C be a closed itemset, and let $Min[C] = \{X_1, \ldots, X_n\}$ be its free sets. An itemset Y belongs to $[C]$ iff there exists i such that $X_i \subseteq Y \subseteq C$.*

PROOF. The if-part follows since $X_i \subseteq Y \subseteq C$ implies $cover(X_i) \supseteq cover(Y) \supseteq cover(C) = cover(X_i)$. The only-if part is immediate. □

This result characterizes the θ-equivalence class $[C]$ given its maximal (closed) itemset and its minimal (free) itemsets. As a consequence, $[C]$ is equivalent to:

$$\cup_{i=1\ldots n}\ s(X_i(C \setminus X_i)^\star).$$

However, while the itemsets $X_1(C \setminus X_1)^\star, \ldots, X_n(C \setminus X_n)^\star$ satisfy condition *(a)* of Definition 3.7, they do not satisfy condition *(b)*, namely they are not pairwise disjoint.

EXAMPLE 4.2. *Consider the sample database and the class of equivalence $[abcd]$ reported in Figure 1. Starting from the three free itemsets $X_1 = d$, $X_2 = ba$ and $X_3 = bc$, the three regular itemsets $R_1 = d\{abc\}^\star$, $R_2 = ba\{cd\}^\star$ and $R_3 = bc\{ad\}^\star$ cover all itemsets in the class of equivalence.*

However, they overlap since $s(R_1) \cap s(R_2) = \{abd, abcd\}$, $s(R_1) \cap s(R_3) = \{bcd, abcd\}$, $s(R_2) \cap s(R_3) = \{abc, abcd\}$. We point out that the overlapping between R_i and R_j starts at the itemset X_i, X_j, namely the minimal common superset of X_i and X_j.

In the next subsections, we devise a procedure to compute a pairwise disjoint set of regular itemsets. First, we introduce a further extension of items, called non-compositional, which serves to divide $[C]$ into disjoint sets. Then we transform non-compositional itemsets into regular itemsets without the $^+$ operator. Finally, we merge pairs of regular itemsets by means of the $^+$ operator.

4.1 Non-Compositional Itemsets

A *non-compositional item* is defined by the grammar of extended items augmented with the following:

$$E \texttt{ ::= } \{a_1, \ldots, a_h\}^-$$

where $h \geq 0$. Intuitively, $\{a_1, \ldots, a_h\}^-$ represents any subset of $\{a_1, \ldots, a_h\}$ except for the the maximal one $a_1, \ldots, a_h$. Formally:

$$s_e(\{a_1, \ldots, a_h\}^-) \quad = \quad \{X \mid X \subset \{a_1, \ldots, a_h\}\}.$$

Notice that $s_e(\{a\}^-) = \{\emptyset\}$, i.e., only the empty itemset is in the semantics, while $s_e(\{\}^-) = \emptyset$, i.e., no itemset is in it.

Consider now itemsets. A *non-compositional itemset* is a finite set of non-compositional items. The semantics $s()$ defined as in (1) does not reflect the intuitive meaning of non-compositional itemsets. Consider, as an example, $\{ab\}^-\{ac\}^-$. Intuitively, we expect that $b \in s(\{ab\}^-)$ and that $a \in s(\{ac\}^-)$. However, their conjunction ab should not be in $s(\{ab\}^-\{ac\}^-)$ since it violates the constraint that a and b should not occur together. In this sense, an and-compositional semantics does not exists. Here it is a non-compositional one:

$$s'(e_1, \ldots, e_n) = \{X \in s(e_1, \ldots, e_n) \mid X \cap Y \subset Y \text{ for every } e_i \text{ of the form } Y^-\}.$$

A non-compositional itemset R is said *regular* if $cover(X) = cover(Y)$ for every $X, Y \in s'(R)$. Similarly, we can extend the notion of concise representation. However, due to the lack of and-compositionality, the class of non-compositional itemsets is not suitable, in our opinion, to be proposed to a data analyst for direct interpretation. Nevertheless, its is an intermediate representation that is useful for our purposes. The next example clarifies how the θ-equivalence class $[C]$ can be easily covered by regular non-compositional itemsets.

EXAMPLE 4.3. *Consider again the class of θ-equivalence $[C]$ in Figure 1, where $C = abcd$ is a closed itemset, and $X_1 = d$, $X_2 = ba$ and $X_3 = bc$ are the free itemsets in the class. As for X_1, we can define $R_1 = X_1, C^\star = d\{abcd\}^\star$ to cover all itemsets X such that $d \subseteq X \subseteq abcd$. As for X_2, we define $R_2 = X_2, X_1^-, C^\star = ba\{d\}^-\{abcd\}^\star$ to cover all itemsets X such that $ba \subseteq X \subseteq abcd$, but such that $d \not\subseteq X$. It turns out that $s'(R_2) = \{ba, bac\}$. Finally, for X_3 we define $R_3 = X_3, X_1^-, X_2^-, C^\star = bc\{d\}^-\{ba\}^-\{abcd\}^\star$ to cover all itemsets X such that $bc \subseteq X \subseteq abcd$, but such that $d \not\subseteq X$ and $ba \not\subseteq X$. It turns out that $s'(R_3) = \{bc\}$.*

Next we formalize the intuitions of the example.

LEMMA 4.4. *There exists a set $\mathcal{R}$ of regular non-compositional itemsets with $|\mathcal{R}| = |\mathcal{FS}|$ that is a concise representation of frequent itemsets.*

PROOF. Let C be a frequent closed itemset, and let $Min[C] = \{X_1, \ldots, X_n\}$ be its free sets. For the regular non-compositional itemsets $N_i = X_i, X_1^-, \ldots, X_{i-1}^-, C^\star$, with $i = 1 \ldots n$, we have:

(a) $\cup_{i=1\ldots n} s'(N_i) = [C]$. By Lemma 4.1, $X \in [C]$ iff for some $i \in [1, n]$, $X_i \subseteq X \subseteq C$. Let k be the minimum of such i's. By construction, we have $X \in s'(N_k)$.

(b) For every pair N_i, N_j with $i \neq j$, $s'(N_i) \cap s'(N_j) = \emptyset$. In fact, assume, without loss of generality, that $i < j$. We have $N_j = X_j, X_1^-, \ldots, X_i^-, \ldots, X_{j-1}^-, C^\star$. If $X \in s'(N_i)$ then $X \supseteq X_i$, but since X_i^- is in N_j, we have $X \notin s'(N_j)$. □

4.2 A Covering Algorithm

In this subsection, we devise a covering procedure that, given a non-compositional itemset R_{in}, computes a set $\mathcal{R}_{out}$ of extended itemsets equivalent to R_{in} and pairwise disjoint, or, formally, such that $s'(R_{in}) = \cup_{R \in \mathcal{R}_{out}} s(R)$ and such that $R_1 \neq R_2 \in \mathcal{R}_{out}$ implies $s(R_1) \cap s(R_2) = \emptyset$. When applied to the non-compositional itemsets whose semantics is the class of θ-equivalence $[C]$, it then provides a covering of $[C]$ through regular itemsets. The approach follows the rewriting rules $S1 - S5$ reported in Figure 2 (b). Let us introduce the intuitions behind them with a few examples.

EXAMPLE 4.5. *Consider Example 4.3.*

It is immediate to notice that $R_1 = d\{abcd\}^\star$ can be simplified to $d\{abc\}^\star$, i.e., by removing the item d from $\{abcd\}^\star$, due to the fact that d must necessarily occur. This is the analogous of the simplification rule $N3$ from Figure 2 (a) for extended itemsets. Similarly, $R_2 = ba\{d\}^-\{cd\}^\star$ and $R_3 = bc\{d\}^-\{ba\}^-\{ad\}^\star$. The general case is stated as rule $S2$ in Figure 2 (b).

A similar reasoning applies to the extended item $\{ba\}^-$ in R_3. Since b necessarily appear in R_3, it can be removed from $\{ba\}^-$. Thus, $R_3 = bc\{d\}^-\{a\}^-\{ad\}^\star$. In general, we have rule $S1$.

Algorithm 1 Covering

Input: a non-compositional itemset R_{in} of the form $X, Y_1^-, \ldots, Y_n^-, Z^\star$

Output: a set $\mathcal{R}_{out}$ of pairwise disjoint extended itemsets equivalent to R_{in}, obtained by the splitting rules $S1$-$S5$

```
R ← X, (Y_1 \ X)^-, ..., (Y_n \ X)^-, (Z \ X)^* //rules S1, S2
R_out ← ∅
if ∀i.Y_i \ X ≠ ∅ then //rule S3
  R ← {R}
  while R ≠ ∅ do
    let R be in R
    R ← R \ {R}
    while ∃Y^- ∈ R do
      let Y^- = {a_1, ..., a_k}^- be in R
      R' ← R \ {a_1?}
      R ← R' where every {a_1, X}^-
                    is replaced by X^* //rules S4, S5
      if k > 1 and ∄{a_1}^- ∈ R' then //rules S5, S3
        R'' ← R' where every {a_1, X}^-
                      is replaced by X^-
        R := R ∪ {(a_1, R'')}
      end if
    end while
    R_out ← R_out ∪ {R}
  end while
end if
```

EXAMPLE 4.6. *Consider $R = a\{a\}^-b?$. By rule $S1$, R is equivalent to $a\emptyset^-b?$. As already noted, $s(\emptyset^-)$ is empty, and this is inherited by any itemset containing $\emptyset^-$. Thus, $s(R) = \emptyset$. Since any extended itemset has a non-empty semantics, this means that no extended itemset is equivalent to R. This motivates rule $S3$ in Figure 2 (b).*

It is worth noting, however, that a non-compositional itemset $N_i = X_i, X_1^-, \ldots, X_{i-1}^-, C^\star$ defined in Lemma 4.4 for covering a θ-equivalence class has a non-empty semantics. In fact, since $X_1, \ldots, X_i$ are minimal sets, $X_i \not\supseteq X_j$ for $j = 1 \ldots n$, and then $X_i \in s'(N_i)$.

EXAMPLE 4.7. *Consider $R_2 = ba\{d\}^-\{cd\}^\star$ from Example 4.5. The conjunct $\{d\}^-$ implies that the item d must not appear. So, $\{d\}^-$ can be removed together with all d? in R_2, thus yielding $R_2 = ba\{c\}^\star = bac?$. Analogously, one concludes $R_3 = bc$. In general, we have rule $S4$ in Figure 2 (b). Let us briefly explain what happens if the hypothesis $a \notin R$ of such a rule is not satisfied, e.g., for the non-compositional itemset $a\{a\}^-$. Here, rule $S1$ applies, so it can be rewritten as $a\emptyset^-$ which, by rule $S3$, has an empty semantics.*

The next example explains rule $S5$.

EXAMPLE 4.8. *Consider $R = cd\{ab\}^-\{ab\}^\star$. The conjunct $\{ab\}^-$ imposes that a and b cannot be both present. Thus, we partition the semantics $s'(R)$ in two sets.*

First set: $s'(R) \cap \{X \subseteq \mathcal{I} \mid a \notin X\}$. We characterize it by removing any a? from R, and by replacing every occurrence of the form $\{Y, a\}^-$ by $Y^\star$. In fact, since a cannot be present, any other subset of Y can appear without violating $\{Y, a\}^-$. Summarizing, $R_1 = cdb?$ covers the first set.

Second set: $s'(R) \cap \{X \subseteq \mathcal{I} \mid a \in X\}$. We characterize such a set by adding a to R, by removing any a? from R, and by replacing $\{Y, a\}^-$ by Y^-. In fact, since a is now present,

the constraint imposed by any $\{Y, a\}^-$ becomes equivalent to the one imposed by Y^-. Summarizing, $R_2 = cda\{b\}^-b?$ covers the second set. By applying rule $S4$, we get $R_2 = cda$.

Figure 2 (b) summarizes the rewriting rules mentioned in the examples above. They allow for deriving zero, one or two non-compositional itemsets that are equivalent to a given one but smaller in the precise sense that at least one item a is removed from extended items of the form Y^-. As a result, the rewriting process starting from a (set of) non-compositional itemset(s) terminates with an equivalent set of extended itemsets, i.e., with no item of the form Y^-. The only rewriting rule that yields two itemsets is rule $S4$. Here, we have to ensure that the two itemsets in the rule consequence are disjoint. This is immediate because one itemset does not contain a (neither $a?$ nor $\{X, a\}^-$) whilst the other does contain a. The **Covering** Algorithm 1 implements the rewriting rules with two optimizations. First, if applied at once, rules $S1$ and $S2$ do not need to be re-checked, since all other rules remove items a appearing in $\{Y, a\}^-$ or in $\{a\}^-$ (preventing rule $S1$ to fire) and explicitly remove items $a?$ (preventing rule $S2$ to fire). Also, the initial application of $S1$ and $S2$ makes it superfluous to check the hypothesis $a \notin R$ in rules $S4$ and $S5$. As a second optimization, if applied at once, rule $S3$ can only be re-checked after rule $S5$, where an item $\emptyset^-$ can potentially be introduced. Thus, rule $S3$ can be folded within rule $S5$.

EXAMPLE 4.9. *The (size of the) covering produced may vary with the order in which free sets are considered in the construction of the non-compositional itemset. For instance, consider again Example 4.3. If we reverse the order of free sets, we obtain $R'_1 = bc\{abcd\}^\star$ which can be rewritten to $bc\{ad\}^\star$, $R'_2 = ba\{bc\}^-\{abcd\}^\star$ which can be rewritten to $bad?$, and $R'_3 = d\{bc\}^-\{ba\}^-\{abcd\}^\star = d\{bc\}^-\{ba\}^-\{abc\}^\star$. The only rule applicable to R'_3 is $S5$, which yields two itemsets: $d\{ac\}^\star$; and $db\{c\}^-\{a\}^-\{ac\}^\star$, which by rule $S4$ is equivalent to db. Summarizing, the cover obtained includes 4 regular itemsets vs. the 3 regular itemsets obtained in Examples 4.5, 4.7.*

4.3 Merging Extended Itemsets

The **Covering** procedure yields a set of extended itemsets of the form $X, Y^\star$, that is, the $^+$ operator is not exploited at all. In this subsection, we devise an algorithm to merge two or more extended itemsets by exploiting the rewriting rules $M1 - M6$ reported in Figure 2 (c).

EXAMPLE 4.10. *With reference to our running example, in Examples 4.5 and 4.7 we derived the regular itemsets $R_1 = d\{abc\}^\star$, $R_2 = bac?$ and $R_3 = bc$. Consider R_2 and R_3. Their common part is b. The semantics of R_2 is that, in addition to b, the item a is present and c may or may not be present. The semantics of R_3 is that, in addition to b, the item c is present. The semantics of both R_2 and R_3 can be rephrased as follows: in addition to b, at least one of a and c must be present. This is precisely the semantics of $b\{ac\}^+$ which, together with R_1, covers $[abcd]$ in Figure 1.*

This example is an application of rule $M3$ in Figure 2 (c). Rule $M4$ is the generalization of $M3$ to the case of extended items of the form Y^+ (vs. items b). Rule $M1$ is self-explicative, and rule $M2$ is its generalization to extended items of the form Y^+.

Algorithm 2 Merging

Input: a set $\mathcal{R}_{in}$ of pairwise disjoint extended itemsets
Output: a set $\mathcal{R}_{out}$ of pairwise disjoint extended itemsets equivalent to R_{in}, obtained by the merging rules $M1$-$M6$

```
R ← R_in
R_out ← ∅
while R ≠ ∅ do
  k ← max{|R ∩ I| | R ∈ R}
  repeat
    let R_1, R_2 ∈ R such that |R_2 ∩ I| = k and |R_1 ∩ I| ∈ [k − 1, k]
    I ← R_1 ∩ R_2
    if R_1 = I and R_2 = I, a then //rule M1
      R ← I, a?
    else if R_1 = I and R_2 = I, Y^+ then //M2
      R ← I, Y^⋆
    else if R_1 = I, b, a? and R_2 = I, a then //M3
      R ← I, {a, b}^+
    else if R_1 = I, Y^+, a? and R_2 = I, a then //M4
      R ← I, {a, Y}^+
    else if R_1 = I, Y^+ and R_2 = I, a, Y^⋆ then //M5
      R ← I, {a, Y}^+
    else if R_1 = I, Y^+ and R_2 = I, Z^+, Y^⋆ then //M6
      R ← I, {Z, Y}^+
    end if
    if any of the rules M1-M6 was applied then
      R = R \ {R_1, R_2} ∪ {R}
    end if
  until for every R_1, R_2 no rule M1-M6 applies
  K ← {R ∈ R | |R ∩ I| = k}
  R ← R \ K
  R_out ← R_out ∪ K
end while
```

Let us show next an example using rule $M5$.

EXAMPLE 4.11. *Consider the alternative cover of $[abcd]$ reported in Example 4.9. It consists of $R'_1 = bc\{ad\}^\star$, $R'_2 = bad?$, $R'_3 = d\{ac\}^\star$ and $R'_4 = db$. We can apply rule $M3$ to R'_2 and R'_4 to obtain $b\{ad\}^+$. This and R'_1 together have the following semantics: in addition to b, if c is present then any of a and d may be present or not; and if c is not present then at least one of a and d must be present. This is precisely the semantics of $b\{acd\}^+$ which, together with R'_3 represent another cover of size 2 for $[abcd]$. Rule $M5$ generalizes the reasoning of this example.*

Finally, rule $M6$ is the generalization of $M5$ to the case of extended items of the form Z^+ (vs. items a).

The rewriting rules $M1 - M6$ allow for merging two extended itemsets into a single one. The rewriting process terminates as soon as no rule can be further applied. However, such a process is inherently non-deterministic, in the sense that the order with which rules are applied is left unspecified. The **Merging** Algorithm 2 implements a *top-down* rewriting strategy, which is motivated by the following observation. For an extended itemset R, let us denote by k_R the number of pure (not extended) items in R, namely $k_R = |R \cap \mathcal{I}|$. Let $\frac{R_1 \quad R_2}{R_3}$ be any rule from $M1 - M6$. We observe that $k_{R_1}, k_{R_3} \in [k_{R_2} - 1, k_{R_2}]$. For instance, for rule $M1$, it turns out $k_R = k_{R,a?} = k_{R,a} - 1$.

Let now k be the maximal k_R for R belonging to the set of extended itemsets $\mathcal{R}$ under rewriting. The **Merging** algorithm proceeds top-down by considering first rules involving R_2 with $k_{R_2} = k$. By the previous observation, the rationale is to progressively reduce the number of pure items from extended itemsets with large values of k_{R_2}. Once no rule $M1 - M6$ can be further applied to any R_2 with $k_{R_2} = k$, the set $\mathcal{K}$ of residual extended itemsets with $k_{R_2} = k$ is removed from $\mathcal{R}$ and added to the output. The loop continues while $\mathcal{R}$ is not empty.

EXAMPLE 4.12. *Let $C = abcd$ and a, b, c, and d be the free sets in $[C]$. The* **Covering** *procedure returns the following extended itemsets to be merged: $a\{bcd\}^\star, b\{cd\}^\star, cd?, d$. First, $a\{bcd\}^\star$ and $b\{cd\}^\star$ are merged (rule $M3$) to $R_1 = \{ab\}^+\{cd\}^\star$; and $cd?$ and d are merged (rule $M3$) to $R_2 = \{cd\}^+$. Then, R_1 and R_2 are merged (rule $M6$) to the final answer $\{abcd\}^+$. As the last example, assume now that ac, ad, bc, and bd are the free sets in $[C]$. The* **Covering** *procedure returns: $ac\{bd\}^\star$, $adb?$, $bcd?$, bd. First, $ac\{bd\}^\star$ and $adb?$ are merged (rule $M3$) to $R_1 = a\{cd\}^+b?$; and $bcd?$ and bd are merged (rule $M3$) to $R_2 = b\{cd\}^+$. Then, R_1 and R_2 are merged (rule $M3$) to the final answer $\{ab\}^+\{cd\}^+$.*

4.4 Mining through Covering

We are now in the position to devise a procedure for mining a concise representation of frequent itemsets. Starting from a frequent closed itemset and the free sets in its class of θ-equivalence, by Lemma 4.4 we first derive a concise representation of the frequent itemsets in the class in terms of pairwise disjoint non-compositional itemsets. Next, by the **Covering** and the **Merging** procedures we rewrite the non-compositional itemsets into equivalent pairwise disjoint regular itemsets. The overall procedure, called **RegularMine**, is reported as Algorithm 3.

There are three sources of non-determinism in the various components of the procedure. The first one is concerned with the order by which the free itemsets $X_1, \ldots, X_n$ are considered in building the non-compositional itemsets $N_i = X_i, X_1^-, \ldots, X_{i-1}^- C^\star$, for $i = 1 \ldots n$. As shown in Examples 4.9 and 4.11, the (size of the) covering found may depend on such an order. Intuitively, smaller free itemsets should be arranged first, so that the initial N_i's cover as much as possible of the elements in $[C]$. For free sets of the same size, a lexicographic ordering leads to simpler non-compositional itemsets. This is due to the fact that N_i is at once rewritten as $X_i, (X_1 \setminus X_i)^-, \ldots, (X_{i-1} \setminus X_i)^- (C \setminus X_i)^\star$. A total order considering first itemset size and then lexicographic ordering was introduced in [8, 11]. It formalizes our intuitions, and it is adopted in the **RegularMine** algorithm.

DEFINITION 4.13. *Let $\preceq$ be a total order over items, and let $\preceq_l$ be the lexicographic ordering induced by $\preceq$ over itemsets. $\preceq$ is extended to itemsets as follows: $X \preceq Y$ iff $|X| < |Y|$ or, $|X| = |Y|$ and $X \preceq_l Y$.*

The second source of non-determinism is in the **Covering** procedure, which contains two choices (see Algorithm 1): (1) which $R \in \mathcal{R}$ to select; and (2) which $Y^- \in R$ to select. It is readily checked that (1) does not affect the output, since the splitting rules $S1$-$S5$ deal with each non-compositional itemset in isolation. On the contrary, the choice (2) can affect the (size of the) output.

Algorithm 3 RegularMine

Input: a transactional database $\mathcal{D}$
Output: a set $\mathcal{R}_{out}$ of frequent regular itemsets that is a concise representation of frequent itemsets

```
extract frequent closed itemsets CS from D
        and, for each C ∈ CS, the free sets in [C]
R_out ← ∅
for every C ∈ CS do
  let X_1, ..., X_n be the free sets in [C] ordered w.r.t. ⪯
  R = ∪_{i=1...n} Covering(X_i, X_1^-, ..., X_{i-1}^-, C*)
  R_out ← R_out ∪ Merging(R)
end for
```

EXAMPLE 4.14. *Let $C = abcd$ be a closed itemset and ab, bc and cd be the (ordered) free sets in $[C]$.*

The non-compositional itemset for the third free set is $R = cd\{ab\}^-\{b\}^-\{ab\}^\star$. By choosing $Y^- = \{ab\}^-$, we apply first rule $S5$ to obtain $cd\{b\}^-b?$ and $cda\{b\}^-b?$, which are further rewritten by rule $S4$, to cd and cda. By choosing $Y^- = \{b\}^-$, we apply rule $S4$ to obtain $cda?$. The latter choice leads to a smaller cover of R.

Intuitively, rule $S4$ should be preferred, if possible, over rule $S5$ – which rewrites a non-compositional itemset into two ones. In the actual implementation of the **Covering** procedure, we achieve that by choosing $Y^- \in R$ as one of those with the smallest size.

The third source of non-determinism is in the **Merging** algorithm, where a pair $R_1, R_2 \in \mathcal{R}$ of extended itemsets has to be selected. Notice that, since the hypotheses of the rules $M1$-$M6$ are mutually-exclusive, at most one of them applies for a given pair; thus, the choice of the rule is deterministic once the pair has been selected. The next example shows that the choice of R_1, R_2 can affect the output.

EXAMPLE 4.15. *Let $C = abcdefg$ be a closed itemset and adf, aef, bdf, bef, cdf, cef, def and dfg be the free sets in $[C]$. The* **Covering** *procedure returns the following extended itemsets: $R_1 = adf\{bceg\}^\star$, $R_2 = aef\{bcg\}^\star$, $R_3 = bdf\{ceg\}^\star$, $R_4 = bef\{cg\}^\star$, $R_5 = cdf\{eg\}^\star$, $R_6 = cefg?$, $R_7 = defg?$, and $R_8 = dfg$.*

Assume that the **Merging** *procedure applies rule $M3$ first to R_1 and R_2, yielding $R'_1 = af\{de\}^+\{bcg\}^\star$; and next to R_7 and R_8, yielding $R'_2 = df\{eg\}^+$. Let us follow from here two possible computations of the* **Merging** *procedure that yield different outputs. In the first computation, rule $M3$ is applied to rewrite R_5 and R_6 to $R'_3 = cf\{de\}^+g?$; then, to rewrite R_3 and R_4 to $R'_4 = bf\{de\}^+\{cg\}^\star$; then, to rewrite R'_1 and R'_4 to $R'_5 = f\{ab\}^+\{de\}^+\{cg\}^\star$. Finally, rule $M4$ is able to rewrite R'_5 and R'_3 to $R'_6 = f\{abc\}^+\{de\}^+g?$. The final result is then $\{R'_6, R'_2\}$.*

In the second computation, rule $M5$ is applied to rewrite R'_2 and R_5 to $R'_7 = df\{ceg\}^+$; then, to rewrite R'_7 and R_3 to $R'_8 = df\{bceg\}^+$. Finally, rule $M3$ is used to rewrite R_4 and R_6 to $R'_9 = ef\{bc\}^+g?$. The final result is then $\{R'_1, R'_8, R'_9\}$.

In the actual implementation of the **Merging** procedure, the selection of a pair R_1, R_2 for which one of the rules $M1$, $M4$, and $M5$ can be applied is preferred. The rationale is that these rules tend to smoothly add one item at a time to extended items of the form Y^+ or $Y^\star$. Rules $M2$, $M3$ and

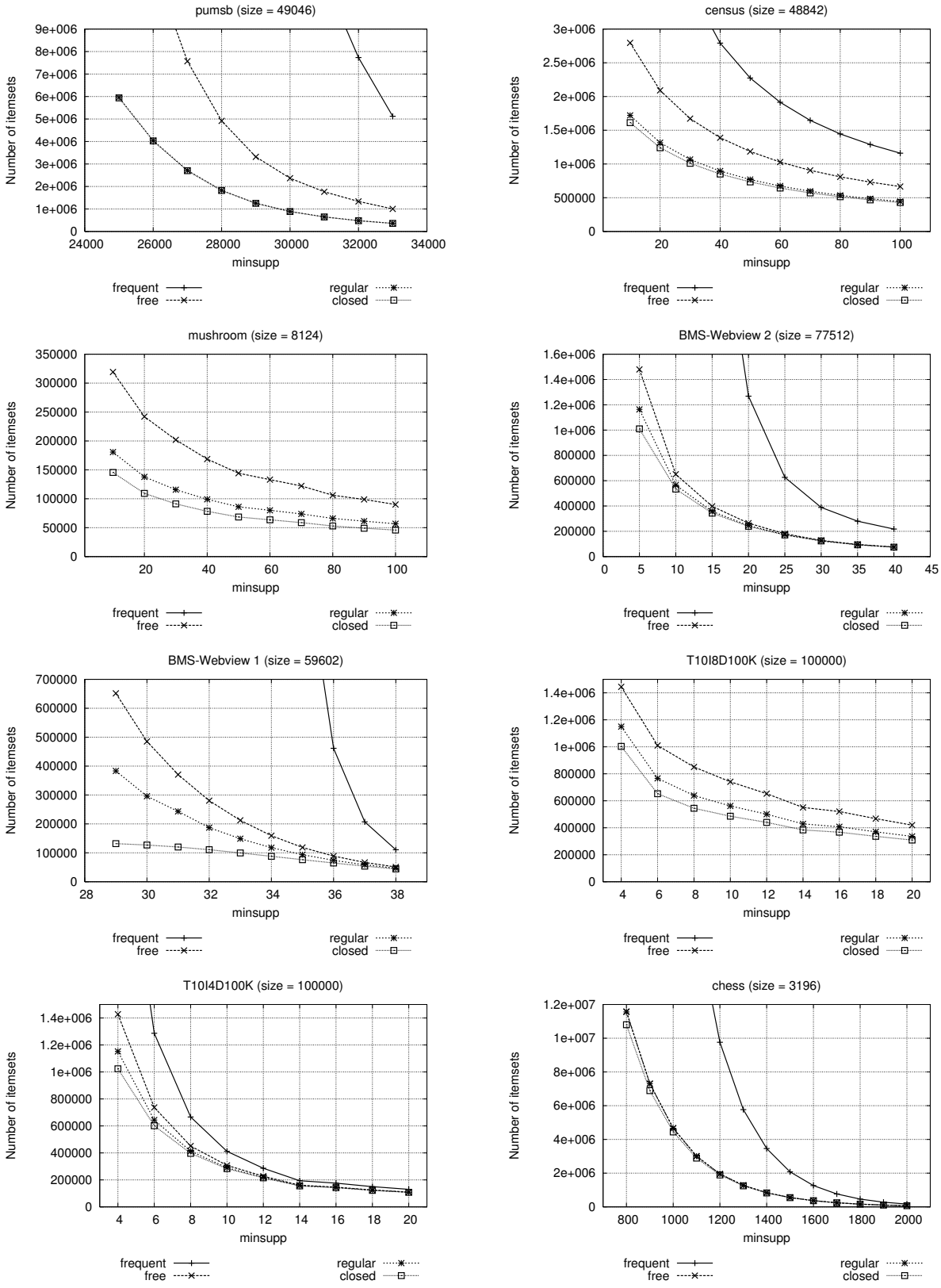

Figure 3: Number of frequent, closed, free and regular itemsets.

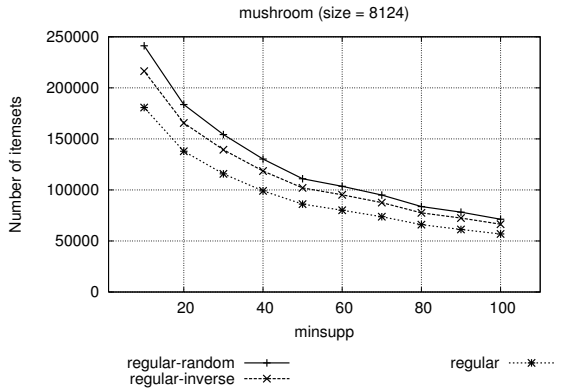

Figure 4: Number of regular itemsets extracted by RegularMine for different orderings of free itemsets.

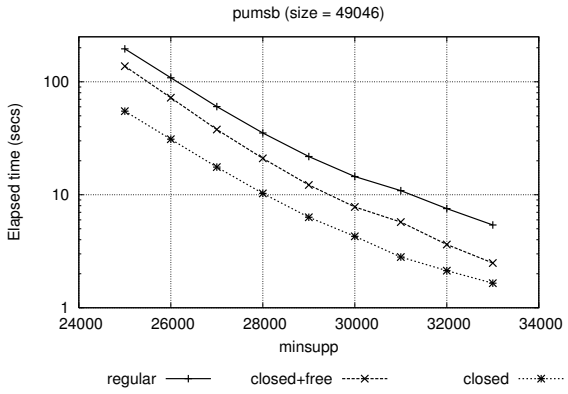

Figure 5: Execution times for mining closed, closed & free, and regular itemsets.

$M6$ instead, introduce new or merge existing extended items of the form above, which could compromise further rewritings. Of course, this choice is an heuristics. Experimentally it performs well, but there are cases, as in the previous example, where it does not yield the smallest output.

Finally, notice that Algorithm 3 does not further specify how to extract closed and free sets. To this purpose, we can resort to a large body of algorithms for frequent closed itemset mining. Most of them actually screen (and some explicitly store, e.g., [22]) the frequent free sets associated to a closed one as a sufficient condition for pruning the search space. We refer the reader to [20] for a survey.

5. EXPERIMENTAL RESULTS

We have experimented the **RegularMine** procedure on standard dense (pumsb, census, mushroom, chess) and sparse (BMS-Webview 1, BMS-Webview 2, T10I4D100K, T10I8D100K) datasets obtained from the FIMI public repository [10], or generated by the Quest synthetic data generator [1]. Figure 3 reports the number of frequent, free, closed and regular itemsets at the variation of the minimum support threshold (for mushroom and T10I8D100K, the plot of frequent itemsets is well beyond the y-range, thus it is not visible). Apart from the BMS-Webview 1 dataset, the size of regular itemsets is very close to the size of closed itemsets. For sparse datasets, this result is expected, since the number of frequent, free and closed itemsets (and even of advanced concise representations [12]) tend to coincide – apart from very low minimum support, e.g., the plot of T10I4D100K shows a relative support in the range of 0.004% - 0.02%. For dense datases, this result supports our claim that regular itemsets, when compared to closed itemsets, are a good trade-off between conciseness and interpretability.

Figure 4 shows, for the sample mushroom dataset, how the ordering of free sets in the **RegularMine** procedure affects the number of regular itemsets extracted. `regular` is the ordering of Definition 4.13; `regular-inverse` sorts free sets by descending size, rather than ascending; and `regular-random` is a random shuffle of the free sets. The rationale behind the choice of the $\preceq$ ordering is supported by its performances.

Let us consider now efficiency of **RegularMine**. When compared to frequent closed itemset mining, the additional tasks in the procedure consist of: (1) first collecting the free sets associated to a closed one; and (2) then computing a covering through the **Covering** and **Merging** procedures. To understand the relative weight of these two tasks, we report in Figure 5, for the sample pumsb dataset, the running times for extracting closed itemsets, for extracting closed and their associated free itemsets, and for the overall **RegularMine** procedure. Experiments were execute on a PC with Intel Xeon 2.8Ghz and 3Gb RAM running Linux core 2.6.17. Our implementation of **RegularMine** is written in standard C++, and it builts on the open source code of the FP-growth algorithm by Borgelt [3]. Figure 5 highlights that the overhead required by **RegularMine** over the extraction of frequent closed itemsets is mainly due to the extraction and storage of free itemsets.

6. RELATED WORK

In addition to the cornerstone concepts of closed and free itemsets, other concise representations in the literature exploit identities (e.g., inclusion-exclusion [9]) and regularities (e.g., disjunction rules $X \Rightarrow a \vee b$) in order to prune from a representation those itemsets whose support can be derived from other ones in the representation. We mention disjunction-free sets [5], non-derivable itemsets [6], closed non-derivable itemsets [16], and, more recently, approaches that maintain the disjunctive or the negative support of itemsets [12, 13], or that refer to measures other than support [19]. While these proposals achieve a higher compactness when compared to closed itemsets (actually, mainly for dense datasets), they all share, and perhaps exacerbate, the interpretability problem highlighted in the introduction for closed itemsets. Among the large body of literature about this subject, the investigation of a concise form for free itemsets seems the closest work to ours. [8] introduced and [11] systematized *succint minimal generators*. They observed that free sets in a class of θ-equivalence can be partitioned with respect to another equivalence relation, called the σ-relation. Hence, only a representative of each σ-equivalence class needs to be maintained. On a general level, we also try and reduce the covering produced by **Covering** (starting from a non-compositional itemset for each free set) by merging extended itemsets through the $^+$ operator. On a more concrete level, we rely on the total order $\preceq$ of simpler-to-complex itemsets that is introduced by [8] for defining the

σ-relation. However, there is no direct relation between succint minimal generators and regular itemsets. First, succint minimal generators are not a concise representation, since free sets alone are not. Second, to reconstruct the free sets of a θ-equivalence class, one has to look at succint minimal generators of that class and of other classes, so the meaning of a succint minimal generator is not and-compositional.

7. CONCLUSIONS

The study of patterns that are easy to understand, to manipulate and to reason about by data analysts, not necessarily data mining experts, is of primary importance for a general acceptance of the knowledge discovery methodology in everyday working life. In this paper, we have introduced an extension of itemsets, called regular itemsets, its clean semantics and a procedure, called **RegularMine**, for mining a concise representation of frequent itemsets. The main idea consists of finding a covering of a θ-equivalence class in terms of regular itemsets. Experiments support our claim that regular itemsets are a good trade-off between conciseness and direct interpretability by a data analyst.

A few open directions include the application of regular itemsets in non-redundant association rule mining [18, 21] and in case studies to validate their actionability on the field. From a computational side, our approach acts as a post-processing phase starting from the closed itemset and the free sets in a θ-equivalence class. As a future work, we intend to study how the approach can be integrated directly within the closed frequent itemset mining phase.

8. REFERENCES

[1] R. Agrawal and R. Srikant. Quest synthetic data generator. `http://www.almaden.ibm.com/cs/projects/iis/hdb/Projects/data_mining/datasets/syndata.html`.

[2] Y. Bastide, R. Taouil, N. Pasquier, G. Stumme, and L. Lakhal. Mining frequent patterns with counting inference. *SIGKDD Explorations*, 2(2):66–75, 2000.

[3] C. Borgelt. An implementation of the FP-growth algorithm. In *Proc. of OSDM 2005*, pages 1–5. ACM, 2005.

[4] J.-F. Boulicaut, A. Bykowski, and C. Rigotti. Free-sets: A condensed representation of boolean data for the approximation of frequency queries. *Data Mining and Knowledge Discovery*, 7(1):5–22, 2003.

[5] A. Bykowski and C. Rigotti. DBC: A condensed representation of frequent patterns for efficient mining. *Information Systems*, 28(8):949–977, 2003.

[6] T. Calders and B. Goethals. Non-derivable itemset mining. *Data Mining & Knowledge Discovery*, 14(1):171–206, 2007.

[7] T. Calders, C. Rigotti, and J.-F. Boulicaut. A survey on condensed representations for frequent sets. In *Constraint-Based Mining and Inductive Databases*, volume 3848 of *LNCS*, pages 64–80. Springer, 2004.

[8] G. Dong, C. Jiang, J. Pei, J. Li, and L. Wong. Mining succinct systems of minimal generators of formal concepts. In *Proc. of DASFAA 2005*, volume 3453 of *LNCS*, pages 175–187. Springer, 2005.

[9] J. Galambos and I. Simonelli. *Bonferroni-type Inequalities with Applications*. Springer, 1996.

[10] B. Goethals. Frequent Itemset Mining Implementations Repository. `http://fimi.cs.helsinki.fi`.

[11] T. Hamrouni, S. B. Yahia, and E. M. Nguifo. Succinct minimal generators: Theoretical foundations and applications. *Int. J. Foundations of Computer Science*, 19(2):271–296, 2008.

[12] T. Hamrouni, S. B. Yahia, and E. M. Nguifo. Sweeping the disjunctive search space towards mining new exact concise representations of frequent itemsets. *Data & Knowledge Engineering*, 68(10):1091–1111, 2009.

[13] M. Kryszkiewicz, H. Rybinski, and K. Cichon. On concise representations of frequent patterns admitting negation. In *Advances in Machine Learning II*, pages 259–289. Springer, 2010.

[14] G. Liu, J. Li, L. Wong, and W. Hsu. Positive borders or negative borders: How to make lossless generator based representations concise. In *Proc. of SIAM Data Mining 2006*. SIAM, 2006.

[15] H. Mannila and H. Toivonen. Multiple uses of frequent sets and condensed representations. In *Proc. of KDD 1996*, pages 189–194. AAAI Press, 1996.

[16] J. Muhonen and H. Toivonen. Closed non-derivable itemsets. In *Proc. of PKDD 2006*, volume 4213 of *LNCS*, pages 601–608. Springer, 2006.

[17] N. Pasquier, Y. Bastide, R. Taouil, and L. Lakhal. Discovering frequent closed itemsets for association rules. In *Proc. of ICDT 1999*, volume 1540 of *LNCS*, pages 398–416. Springer, 1999.

[18] N. Pasquier, R. Taouil, Y. Bastide, G. Stumme, and L. Lakhal. Generating a condensed representation for association rules. *Journal of Intelligent Information Systems*, 24(1):29–60, 2005.

[19] A. Soulet and B. Crémilleux. Adequate condensed representations of patterns. *Data Mining & Knowledge Discovery*, 17(1):94–110, 2008.

[20] S. B. Yahia, T. Hamrouni, and E. M. Nguifo. Frequent closed itemset based algorithms: A thorough structural and analytical survey. *SIGKDD Explorations*, 8(1):93–104, 2006.

[21] M. J. Zaki. Mining non-redundant association rules. *Data Mining & Knowledge Discovery*, 9(3):223–248, 2004.

[22] M. J. Zaki and C.-J. Hsiao. Efficient algorithms for mining closed itemsets and their lattice structure. *IEEE Trans. on Knowledge and Data Engineering*, 17(4):462–478, 2005.