# Mining association rules with negations

Jean-François Boulicaut  Artur Bykowski
Baptiste Jeudy

Laboratoire d'Ingénierie des Systèmes d'Information
INSA Lyon, Bâtiment 501
F-69621 Villeurbanne Cedex, France
{Jean-Francois.Boulicaut,Artur.Bykowski,Baptiste.Jeudy}@insa-lyon.fr

November 3, 2000

**Abstract**

Frequent association rules (e.g., $A \wedge B \Rightarrow C$ to say that when properties $A$ and $B$ are true in a record then, $C$ tends to be also true) have become a popular way to summarize huge datasets. The last 5 years, there has been a lot of research on association rule mining and more precisely, the tractable discovery of interesting rules among the frequent ones. We consider now the problem of mining association rules that may involve negations e.g., $A \wedge B \Rightarrow \neg C$ or $\neg A \wedge B \Rightarrow C$. Mining such rules is difficult and remains an open problem. We identify several possibilities for a tractable approach in practical cases. Among others, we discuss the effective use of constraints for mining generalized sets. We propose a generic algorithm and discuss the use of constraints (e.g., the set must contain at most $n$ negative attributes or at least $p$ positive attributes) to mine the generalized sets from which rules with negations can be derived. An experimental validation of this approach on the `mushroom` benchmark is reported.

## 1 Introduction

The design of semiautomatic methods for locating interesting information in the masses of unanalyzed or under-analyzed data, the so-called data mining techniques, has become an important research area. Mining association rules [1] is a popular data mining technique that has been proved useful for real data analysis (see e.g., its application to basket analysis or [16] for alarm data

analysis). During a typical association rule mining process, one first computes frequent patterns using (expensive) data mining algorithms. Then, we have to compute and/or query these collections for the needed post-processing phases (e.g., deriving rules and ranking them according to various objective measures of interestingness such as confidence [2], conviction [8], J-measure [22] or intensity of implication [13]).
A problem with such a process is that (a) the selection of interesting patterns has to be performed only on frequent patterns, and (b) standard association rules are not enough expressive for some applications. Mining all infrequent rules is known to be intractable, but is it interesting? Indeed, infrequent patterns can be considered as the result of noisy data. However, the frequency threshold that characterizes interesting rules for an application might be too low from the computational complexity point of view. Looking for more expressive rules, we can consider various generalizations, e.g., the introduction of taxinomies on items [14], associations between multiple relations [10] or general boolean rules [18]. In this paper, we consider generalized rules, a special case of boolean rules, namely association rules with negations. We want to find associations involving positive attributes and, eventually, negative ones (a negative attribute is the negation of a positive attribute). Mining frequent (generalized) rules turns to be intractable in practical cases, e.g., for frequency thresholds that enable the discovery of "standard" association rules. We would like to identify possibilities for a tractable approach to the computation of generalized (frequent) sets (i.e. sets that involve positive and negative attributes) and their post-processing into generalized association rules. We believe that the inductive querying framework [15, 7, 11] also called by some authors constraint-based pattern discovery [20, 17, 12] point out promising issues.
The contribution of this paper is as follows. First, we explain why a naive approach (i.e., the straightforward encoding of both positive and negative attributes and the use of standard algorithms like Apriori) cannot be used. Then we consider how it is possible to derive some generalized rules using only the information about positive attributes. Finally, we consider inductive queries that return generalized sets (from which rules with negations can be derived) and discuss their evaluation. Roughly speaking, this means that, given constraints on desired sets, one must identify which of them can be "pushed" efficiently into the discovery algorithm. Using results from [6], we consider relevant constraints for mining association rules with negations and report practical experiments on a well-known benchmark. A preliminary version of that work has been published in [4]. However, [4] neither discusses rule generation issues nor experimental validation of these ideas.
Section 2 provides a simple formalization of the kind of inductive query we have to process. In Section 3, using an abstract presentation of the classical

APRIORI algorithm, we introduce the standard approach to association rule discovery and discuss the problems when rules with negations are desired. In Section 4, we identify several kinds of constraints that can be used for the effective discovery of rules with negations. In that section, we report also an experimental study on a benchmark dataset (the so-called `mushroom` dataset) that confirms the validity of the constraint-based framework. Finally, Section 5 concludes.

## 2 Inductive Databases and Inductive Queries

We consider the formalization of inductive databases [7] and the concept of inductive query in our context.

**Definition 1 (schema and instance)** *The schema of an inductive database is a pair $\mathcal{R} = (\mathbf{R}, (\mathcal{L}, \mathcal{E}, \mathcal{V}))$, where $\mathbf{R}$ is a relational database schema, $\mathcal{L}$ is a countable collection of patterns, $\mathcal{V}$ is a set of* result values*, and $\mathcal{E}$ is the* evaluation function *that characterizes patterns. Given a database $\mathbf{r}$ over $\mathbf{R}$ and a pattern $\theta \in \mathcal{L}$, this function maps $(\mathbf{r}, \theta)$ to an element of $\mathcal{V}$. An instance $(\mathbf{r}, s)$ over $\mathcal{R}$ consists of a database $\mathbf{r}$ over the schema $\mathbf{R}$ and a subset $s \subseteq \mathcal{L}$.*

**Example 1** *Assume the minable view is* `trans(Tid,Item)` *i.e., a typical schema of data for basket analysis. Figure 1 provides a toy dataset under a boolean matrix format. For instance, if* `trans(2,A)` *and* `trans(2,C)` *define the transaction 2, row 2 contains true for columns A and C and false for column B. It also means that, in row 2 of the "complemented" matrix, $\overline{A}$ (to denote $\neg A$) and $\overline{C}$ are false while $\overline{B}$ is true.*

A typical KDD process operates on both of the components of an inductive database. Queries that concerns only the pattern part, called hereafter *inductive queries*, specify mining tasks. We use constraints to characterize the patterns that are interesting.

**Definition 2 (constraint)** *Given the schema $\mathcal{R} = (\mathbf{R}, (\mathcal{L}, \mathcal{E}, \mathcal{V}))$ and the set $DB(\mathbf{R})$ of all the relational databases over the schema $\mathbf{R}$, a constraint $\mathcal{C}$ is a function from $\mathcal{L} \times DB(\mathbf{R})$ to the set $\{true,\ false\}$. We say that a pattern $\theta \in \mathcal{L}$* satisfies *a constraint $\mathcal{C}$ in the database $\mathbf{r} \in DB(\mathbf{R})$ if $\mathcal{C}(\theta, \mathbf{r}) = true$. When it is clear from the context, we omit the reference to data and write $\mathcal{C}(\theta)$.*

The classical logical operators are defined over $\{true,\ false\}$ and can be used to construct complex constraints (such as conjunction of constraints).

**Example 2** *Assuming that $\mathcal{V}$ is [0,1] and $\mathcal{E}$ returns the frequency of $\theta$ in $\mathbf{r}$ (to be defined later), $\mathcal{C}(\theta, \mathbf{r}) \equiv \mathcal{E}(\mathbf{r}, \theta) > 0.5$ is a constraint that is satisfied by patterns $\theta$ such that $\mathcal{E}(\mathbf{r}, \theta) > 0.5$.*

**Definition 3 (inductive query)** *Given an inductive database instance* $(\mathbf{r}, s)$ *whose schema is* $(\mathbf{R}, (\mathcal{L}, \mathcal{E}, \mathcal{V}))$*, an inductive query is denoted as* $\sigma_{\mathcal{C}}(s)$ *and specifies the patterns from s that are interesting.* $\mathcal{C}$ *is a conjunction of constraints that must be fulfilled by the desired patterns. The result of the inductive query is the set of patterns of s that satisfy the constraint* $\mathcal{C}$ *and is denoted* $SAT_{\mathcal{C}}(s) = \{\theta \in s,\ \mathcal{C}(\theta) = true\}$.

We see from the previous example that checking some of the conjuncts may need for the evaluation of $\mathcal{E}$ on $\mathbf{r}$ and involve data scans.

## 2.1 Generalized Set and Rule Mining Frameworks

The databases $\mathbf{r}$ we consider are boolean matrixes. The rows of the matrixes are called *transactions* and the columns are *positive attributes*. The positive attributes are denoted with capital letters $A$, $B$, $C$... the set of the positive attributes is denoted $\texttt{Items}^+ = \{A, B, C, ...\}$. Let $\texttt{Items}^-$ be a set of same cardinality as $\texttt{Items}^+$. Its elements are denoted $\overline{A}$, $\overline{B}$, $\overline{C}$,... and called the *negative attributes*. Finally, let $\texttt{Items} = \texttt{Items}^+ \cup \texttt{Items}^-$, its elements are called the *attributes*. A *generalized set* $S$ is a subset of $\texttt{Items}$.
Given a generalized set $T \in 2^{\texttt{Items}}$, let $\texttt{NNT}(T)$ denote the number of negative attributes in $T$, $\texttt{PT}(T)$ denote the set of positive attributes in $T$, $\texttt{P}(T)$ denote the set of attributes in $T$ all turned to their positive counterparts. We often use a string notation for sets e.g., $\overline{AB}C$ for $\{\overline{A}, \overline{B}, C\}$.

**Example 3** $\texttt{NNT}(\overline{AB}C) = 2$, $\texttt{PT}(\overline{AB}C) = C$, $\texttt{P}(\overline{AB}C) = ABC$.

A generalized set $S \subseteq \texttt{Items}$ is *present* in a transaction T (a row of the boolean matrix) iff

- for each positive attribute $X \in S$, there is a 1 in the corresponding column at row T,
- for each negative attribute $\overline{X} \in S$, there is a 0 in the column corresponding to $X$ at row T.

The *frequency*, $\mathcal{F}(S, \mathbf{r})$, of a generalized set $S$ in $\mathbf{r}$ is the ratio of the number of line in which $S$ is present over the total number of line of the matrix.
A generalized set $S$ is $\gamma$-*frequent* in $\mathbf{r}$ if $\mathcal{F}(S, \mathbf{r}) \geq \gamma$. We denote by $\mathcal{C}_{freq}(S) \equiv \mathcal{F}(S, \mathbf{r}) \geq \gamma$ the constraint that is true iff $S$ is $\gamma$-frequent in $\mathbf{r}$.

**Definition 4 (generalized set mining task)** *Mining generalized set is the evaluation of the inductive query* $\sigma_{\mathcal{C}}(2^{\texttt{Items}})$ *where* $\mathcal{C}$ *is a constraint. The pattern part of the associated inductive database is* $(2^{\texttt{Items}}, \mathcal{F}, [0, 1])$ *Mining* frequent *generalized sets means that* $\mathcal{C}$ *contains* $\mathcal{C}_{freq}$.

**Example 4** *Given data from Figure 1,* $\texttt{Items} = \{A, B, C, \overline{A}, \overline{B}, \overline{C}\}$*. If* $\mathcal{C}_{freq}(X) \equiv \mathcal{F}(X, \mathbf{r}) \geq 0.5$*, the query* $\sigma_{\mathcal{C}_{freq}}(2^{\texttt{Items}})$ *returns* $\{A, B, C, \overline{C}, AB, AC, B\overline{C}\}$*. Assume that* $\mathcal{C}_{size}$ *(resp.,* $\mathcal{C}_{miss}$*) denotes the constraint* $|X| \leq 2$ *(resp.,* $\{A, \overline{B}\} \cap X = \emptyset$*) for a set X.*
$\sigma_{\mathcal{C}_{size} \wedge \mathcal{C}_{miss}}(2^{\texttt{Items}})$ *returns* $\{B, C, \overline{A}, \overline{C}, BC, B\overline{A}, B\overline{C}, C\overline{A}, C\overline{C}, \overline{AC}\}$*.*
$\sigma_{\mathcal{C}_{freq} \wedge \mathcal{C}_{size} \wedge \mathcal{C}_{miss}}(2^{\texttt{Items}})$ *returns* $\{B, C, \overline{C}, B\overline{C}\}$*.*

$\mathbf{r} =$

| | $A$ | $B$ | $C$ |
|---|---|---|---|
| 1 | 1 | 1 | 1 |
| 2 | 1 | 0 | 1 |
| 3 | 0 | 1 | 0 |
| 4 | 1 | 1 | 0 |

Its “complement” is

| | $\neg A$ | $\neg B$ | $\neg C$ |
|---|---|---|---|
| 1 | 0 | 0 | 0 |
| 2 | 0 | 1 | 0 |
| 3 | 1 | 0 | 1 |
| 4 | 0 | 0 | 1 |

Figure 1: A binary dataset $\mathbf{r}$

An *association rule* is an expression $X \Rightarrow Y$ where $X \subseteq \texttt{Items}$ and $Y \in \texttt{Items} \backslash X$. The typical “behavior” of these rules in an instance $\mathbf{r}$ is evaluated by means of two interestingness measures, namely the *support* and the *confidence.* The support of $X \Rightarrow Y$ in $\mathbf{r}$ is equal to $\mathcal{F}(X \cup \{Y\}, \mathbf{r})$ and its confidence is equal to its support divided by $\mathcal{F}(X, \mathbf{r})$.

**Definition 5 (association rule mining task)** *The standard association rule mining task concerns the discovery of the so-called* strong rules *whose support and confidence are greater or equal to user-given thresholds, resp.,* $\gamma$ *and* $\phi$ *[2]. In terms of an inductive schema,* $\mathcal{V}$ *is [0,1]* $\times$ *[0,1] and* $\mathcal{E}$ *provides the pair <support, confidence>. It corresponds to inductive queries on the language of rules whose constraint contains at least this selection criterion.*

**Example 5** *With* $\gamma$*=0.5 and* $\phi$*=0.9,* $\overline{C} \Rightarrow B$ *is discovered in data of Figure 1 while* $B \Rightarrow \overline{C}$ *is not (its confidence is too low).*

## 3 Mining Association Rules with Negations

When mining association rules, the expensive step concerns the computation of the (frequent) sets from which the rules are derived.
Most of the tractable queries involve $\mathcal{C}_{freq}$. $SAT_{\mathcal{C}_{freq}}(\texttt{Items})$ is the collection of frequent (generalized) sets and we assume that they are stored, with their frequencies in $FS_\gamma$. Notice that we need the frequency of every frequent set since our goal is to derive interesting rules (interest evaluation functions may avoid data scans if they are provided with the frequency of every frequent set).

## 3.1 The "Standard" Apriori Algorithm

We consider an abstract definition of the Apriori algorithm [2]. It takes a dataset **r** and, given the thresholds $\gamma$ (frequency) and $\phi$ (confidence), outputs every strong association rule.

```
1.   $C_1$ := set-of-all-singletons(Items)
2.   k := 1
3.   while $C_k \neq \emptyset$ do
4.       $k := k + 1$
5.       Phase 1 - frequency constraint is checked - it needs a data scan
         $\mathcal{L}_{k-1} := SAT_{\mathcal{C}_{freq}}(C_{k-1})$
6.       Phase 2 - candidate generation for level k
         $C_k^g$ := generate($\mathcal{L}_{k-1}$)
7.       Phase 3 - candidate safe pruning
         $C_k$ := safe-pruning-on($C_k^g$)
     od
8.   $FS_\gamma := \bigcup_{i=1}^{k} \mathcal{L}_i$
9.   Phase 4 - rule generation
     for each $X \in FS_\gamma$ do
10.      for each $Y \in X$ do
11.          test-for-output($X \setminus \{Y\} \Rightarrow \{Y\}$)
         od
     od
```

In step 8, all the frequent sets and their frequencies are stored in $FS_\gamma$ and the function `test-for-output` tests if the confidence of the rule is high enough w.r.t. $\phi$. In [2], `generate`($\mathcal{L}_{k-1}$) provides the candidates by fusion of two elements from $\mathcal{L}_{k-1}$ that share the same $k-2$ first elements (in lexicographic order) and `safe-pruning-on`($C_k^g$) just eliminates the candidates for which there exists a subset of length $k-1$ that is not frequent. In [2], phases 2 and 3 are merged. We provide here an abstract formulation of these procedures that will be used hereafter. The first one contains the join-based method that provides new candidates (`generate`). The second defines `safe-pruning-on`: it prunes candidates which can not be frequent according to the anti-monotonicity of $\mathcal{C}_{freq}$ (one of their subsets is not frequent).
The notation r.item$_i$ denotes the $i^{th}$ element (i.e., itemset) from r.

**Example 6** *Considering the data from Figure 1 and the thresholds $\gamma = 0.5$ and $\phi = 1$,* Apriori *outputs only $C \Rightarrow A$ and $\overline{C} \Rightarrow B$.*

Apriori can work fine for huge datasets. Its practical time complexity is $\mathcal{O}(n \times nc)$ where $n$ is the number of transactions and $nc$ the number of candidates (i.e.,

```
generate

1.   insert into $C_k^g$
2.   select r.item$_1$, r.item$_2$, $\ldots$, r.item$_{k-1}$, q.item$_{k-1}$
3.   from  $\mathcal{L}_{k-1}$ r, $\mathcal{L}_{k-1}$ q
4.   where  r.item$_1$ = q.item$_1$, r.item$_2$ = q.item$_2$, $\ldots$,
5.              r.item$_{k-2}$ = q.item$_{k-2}$, r.item$_{k-1}$ < q.item$_{k-1}$

safe-pruning-on

6.   for all  C $\in C_k$ do
7.       for all  $T \subset C$ such that $|T| = k-1$ do
8.           if  $T \notin \mathcal{L}_{k-1}$
9.               then  delete C from $C_k$
10.          fi
11.      od
12.  od
13.  output $C_k$
```

the size of $FS_\gamma$ plus the size of its negative border $\mathcal{B}d^-(FS_\gamma)$ [19]). $\mathcal{B}d^-(FS_\gamma)$ is the collection of infrequent sets whose every subset is frequent. Deriving rules is cheap when checking the selection criterion needs only $FS_\gamma$ (e.g., when the selection is based on the confidence evaluation for frequent rules).

## 3.2 Problems with Negations ... and Potential Solutions

### 3.2.1 A Naive Approach

Assume the use of Apriori on a dataset that has been "complemented". Considering the boolean matrix representation, it means that for each column $A$, one adds a column $\overline{A}$. In a row, i.e., for a given transaction, the value of $\overline{A}$ is the boolean negation of the value for $A$.
The problem with that approach is that for reasonable frequency thresholds, the number of frequent sets explodes. For instance, assume you have 100 positive attributes with a maximum attribute frequency of 0.1 and a frequency threshold $\gamma = 0.05$. It turns out that every set of negative attributes up to size 9 is frequent. In that case, it leads to more than $\binom{100}{9} > 10^{12}$ frequent sets [18]. In practice, it means that we have to take higher frequency thresholds, possibly leading to uninteresting mining phases. Furthermore, even if the extraction of the frequent sets remains tractable, most of them will involve only negative attributes and, most of the derived rules (with high confidence)

will concern only negative attributes as well. This is unfortunate for many application domains. In fact, this encoding also introduces a high correlation in the dataset: many association rules with high confidence hold in it. Notice also that the positive part of the data can be already highly-correlated so that APRIORI can even not tackle the computation of $FS^+_\gamma$ i.e., subsets of $\texttt{Items}^+$ that are frequent.

### 3.2.2 "Approximation" of Association Rules with Negations

Assume in this subsection that we compute $FS^+_\gamma$ i.e., the collection of $\gamma$-frequent sets for the positive attributes only. We already know that this problem is easier to solve. Using only that (positive) information, it remains possible to mine some association rules with negations.

**Theorem 1** *The support of a generalized set $T$ can be computed using the collection $FS^+_\gamma$ if $\mathtt{P}(T) \in FS^+_\gamma$ and is equal to:*

$$\mathcal{F}(T, \mathbf{r}) \quad = \sum_{\mathtt{PT}(T) \subseteq X \subseteq T} (-1)^{\mathtt{NNT}(X)} \mathcal{F}(\mathtt{P}(X), \mathbf{r}) \tag{1}$$

Proof sketch: Note that if $T$ is of the form $T_{n-1} \cup \{\overline{A_n}\}$,

$$\mathcal{F}(T, \mathbf{r}) = \mathcal{F}(T_{n-1}, \mathbf{r}) - \mathcal{F}(T_{n-1} \cup \{A_n\}, \mathbf{r}).$$

**Example 7** *Considering data from Figure 1, $FS^+_{0.5} = \{A, B, C, AB, AC\}$*[1]*. Since $\mathtt{P}(A\overline{B}) = AB \in FS^+_{0.5}$, $\mathcal{F}(A\overline{B}, \mathbf{r}) = \mathcal{F}(A, \mathbf{r}) - \mathcal{F}(AB, \mathbf{r})$ is known exactly ($= 0.25$). $\mathtt{P}(AB\overline{C}) = ABC \notin FS^+_{0.5}$ and $\mathcal{F}(AB\overline{C}, \mathbf{r}) = \mathcal{F}(AB, \mathbf{r}) - \mathcal{F}(ABC, \mathbf{r})$ can not be evaluated exactly.*

**Theorem 2** *The support of all subsets of a set $T$ can be derived from the collection $FS^+_\gamma$ if $\mathtt{P}(T) \in FS^+_\gamma$.*

Proof: Let $S \subset T$. Then $\mathtt{P}(S) \subseteq \mathtt{P}(T)$. If $\mathtt{P}(T) \in FS^+_\gamma$, all subsets of $\mathtt{P}(T)$ belong to $FS^+_\gamma$ including $\mathtt{P}(S)$. From the previous theorem, the support of $S$ may therefore be computed from the collection $FS^+_\gamma$.

The support of some generalized sets can be computed exactly from the positive frequent sets only. In such a case, the number of terms of the sum in formula (1) is growing exponentially with the number of negative attributes in the set. However, it can not be higher than $|FS^+_\gamma|$, which is the number of all possible terms of the sum. Therefore, the computation of the support of a generalized set remains tractable when the computation of $FS^+_\gamma$ is tractable.

[1] Notice also that the frequency in $\mathbf{r}$ is associated to each frequent set in $FS^+_{0.5}$

Consider now the generation of rules from a frequent set $X \in FS^+_\gamma$. Let $A_X$ be a generalized set involving the same properties than $X$. There are $2^{|X|}$ such generalized sets from which the Phase 4 of the APRIORI algorithm can be performed. This way, we can generate a lot of generalized rules, but the collection is not complete w.r.t. the support and confidence criteria: there is no guarantee that it computes all the strong generalized rules. Our experiments have shown that it can even be far from the intended result.
A special case of this method can be performed for free during the standard association rule discovery. It concerns a restricted form of generalized rule: rules with a set $\subset$ `Items`$^+$ at the left–hand side and an attribute from `Items` (positive or negative) at the right–hand side. After the computation of $FS^+_\gamma$, for each frequent set $X \in FS^+_\gamma$ and for each $Y \in X$ (all of them are frequent, too), it is possible to test the confidence of $X \setminus \{Y\} \Rightarrow \{Y\}$ and $X \setminus \{Y\} \Rightarrow \{\overline{Y}\}$ as well. Testing the confidence of the second rule needs for the evaluation of $1 - (\mathcal{F}(X, \mathbf{r})/\mathcal{F}(X \setminus \{Y\}, \mathbf{r})) \geq \phi$ i.e., $\mathcal{F}(X, \mathbf{r})/\mathcal{F}(X \setminus \{Y\}, \mathbf{r}) \leq 1 - \phi$ (note that $\mathcal{F}(X, \mathbf{r})/\mathcal{F}(X \setminus \{Y\}, \mathbf{r})$ is the confidence of the first rule). However, even limited to this form, some strong rules might be missed by the method. To be complete, we need to know the frequency of all generalized frequent sets and we saw that it is not realistic. Therefore, it is possible to trade precision against completeness.
Consider now the possibility to compute imprecise interestingness measures for generalized rules by using only positive information. The idea is to substitute to the unknown terms an interval that bounds the possible values for the support and the confidence. Thus, it gives rise to an incertitude to the values of these measures. Let us consider a second dataset in Figure 2.

$\mathbf{r} =$

| | A | B | C | D | E |
|---|---|---|---|---|---|
| 1 | 1 | 1 | 1 | 0 | 0 |
| 2 | 0 | 0 | 1 | 1 | 1 |
| 3 | 0 | 1 | 0 | 0 | 1 |
| 4 | 1 | 1 | 1 | 0 | 1 |
| 5 | 1 | 1 | 1 | 0 | 0 |
| 6 | 1 | 1 | 0 | 1 | 0 |
| 7 | 1 | 0 | 0 | 0 | 1 |
| 8 | 0 | 1 | 1 | 0 | 0 |
| 9 | 1 | 0 | 0 | 1 | 0 |

| Set | Frequency |
|---|---|
| $\{A\}$ | 6/9 |
| $\{B\}$ | 6/9 |
| $\{C\}$ | 5/9 |
| $\{D\}$ | 3/9 |
| $\{E\}$ | 4/9 |
| $\{A,B\}$ | 4/9 |
| $\{A,C\}$ | 4/9 |
| $\{A,E\}$ | 2/9 |
| $\{B,C\}$ | 4/9 |
| $\{B,E\}$ | 2/9 |
| $\{C,E\}$ | 1/9 |
| $\{A,B,C\}$ | 3/9 |

Figure 2: A dataset and the frequency for each set $\in FS^+_{4/9} \cup \mathcal{B}d^-(FS^+_{4/9})$

**Example 8** *Consider the rule $A\overline{E} \Rightarrow B$. support($A\overline{E} \Rightarrow B, \mathbf{r}$) $= \mathcal{F}(AB, \mathbf{r}) - \mathcal{F}(ABE, \mathbf{r})$ and confidence($A\overline{E} \Rightarrow B, \mathbf{r}$) $=$ support($A\overline{E} \Rightarrow B, \mathbf{r}$)$/(\mathcal{F}(A, \mathbf{r}) -$*

*$\mathcal{F}(AE, \mathbf{r})$). $\mathcal{F}(ABE, \mathbf{r})$ is not available (because $ABE$ is not in $FS_{4/9}$ neither in $\mathcal{B}d^-(FS_{4/9})$) but it can be estimated within $[0, 3/9]$: support($A\overline{E} \Rightarrow B, \mathbf{r}) \in [1/9, 4/9]$ and confidence($A\overline{E} \Rightarrow B, \mathbf{r}) \in [1/4, 1]$.*

This method has been proposed in [18]. It gives rise to several problems. First, a rule can be interesting (w.r.t. interestingness criteria), uninteresting or unresolved. The last case arises when intervals for support and/or confidence cross support and/or minimum confidence thresholds. It happens in Ex. 7. By substituting an interval of possible values to unknown terms in the support formula of a generalized set, some sets can not be classified as frequent or infrequent. Hence, for a complete strong rule generation in the worst case, one has to enumerate every generalized set that is not known as infrequent (i.e., that is frequent or unresolved). If we introduce "large" intervals, it gives rise to a huge amount of unresolved sets. Lot of them might be infrequent, but it is not possible to identify which ones.

**Example 9** *Continuing Ex. 7, from $FS^+_{4/9} \cup \mathcal{B}d^-(FS^+_{4/9})$, we infer 14 frequent generalized sets, 19 unresolved ones, and 210 infrequent ones. On this toy example the number of unresolved generalized sets is above the number of frequent generalized sets. This would be worse if we had $FS^+_{4/9}$ only because these numbers become respectively 11, 48 and 184.*

Problems related to incertitude amplify with the computation of the confidence or other interestingness measures. Providing better estimations is important and worthwhile.

### 3.2.3 Using Constraints

A third direction of research is the effective use of constraints during the mining process. We focus on the effective computation of $SAT_{\mathcal{C}}$ where $\mathcal{C}$ is a conjunction of atomic constraints that specify the interesting sets, and from which "interesting" rules are to be derived.
We saw that Apriori uses the constraint $\mathcal{C}_{freq}$ to prune the search space. Beside that, it is possible to have a "generate and test" approach for the other constraints (first generate all frequent generalized sets and then test other constraints on them) but this is clearly intractable in many cases.
The solution might come from the study of constraint–based discovery of frequent sets [20, 6].

## 4 A Constraint-based Approach

In this section, we apply part of the work presented in [6] to our problem. A simple observation is that the Apriori pruning is not only applicable to the frequency constraint but also to all anti-monotone constraints.

**Definition 6 (anti-monotonicity)** *A constraint $\mathcal{C}$ is* anti-monotone *if and only if for all sets* `S`, `S'`: `S'` $\subseteq$ `S` $\wedge$ `S` *satisfies* $\mathcal{C} \Rightarrow$ `S'` *satisfies* $\mathcal{C}$

**Example 10** *A systematic study of anti-monotone constraints on sets is in [20]. Continuing our running example, $\{A,B,C,D\} \supset$ `S` and `S`$\cap\{A,B,C\} = \emptyset$ are anti-monotone constraints on* `S`*. Indeed, $\mathcal{C}_{freq}$ is anti-monotone. Another interesting anti-monotone constraint is $|S \cap \texttt{Items}^-| \leq n$, denoted as $\mathcal{C}_{amnn}$, which states that every itemset must contain a̲t m̲ost n̲ n̲egative attributes. $\mathcal{C}_{am1n}$ denotes the special case where $n = 1$.*

Conjunctions of anti-monotone constraints are anti-monotone and it is easy to prove that pushing an anti-monotone constraint inside the search space exploration leads to less computations. What is challenging is the possibility of pushing other constraints. This is difficult because the generation and pruning steps must be rewritten to be complete. Moreover, pushing non anti-monotone constraints can decrease the performance of the whole process [6].

## 4.1 A Generic Algorithm $\mathcal{G}$

We consider a generalization of Apriori to emphasize the potential for optimization. Like in Apriori, it is a levelwise exploration algorithm of the set lattice and $k$ denotes the size of the sets that are processed at the current level. Assume the constraint $\mathcal{C}$ can be written as $\mathcal{C}_g \wedge \mathcal{C}_{s4} \wedge \mathcal{C}_{s6} \wedge \mathcal{C}_{dbs} \wedge \mathcal{C}_{s8} \wedge \mathcal{C}_{s11}$. The notations $s4$, $s6$, $s8$, and $s11$ refer to the steps in which these conjuncts are processed in the generic algorithm. $\mathcal{C}_{dbs}$ denotes constraints that need a database scan at checking time and $\mathcal{C}_g$ denotes constraint checked during the generation step.

**Algorithm $\mathcal{G}$: computation of $\sigma_{\mathcal{C}}(2^{\texttt{Items}})$**

```
1.  $k := 1$; $\mathcal{L}_0 := \{\emptyset\}$
2.  repeat
3.  $C_k^g :=$ generate($\mathcal{L}_{k-1}$) # Candidate generation for level k and Test $\mathcal{C}_g$
4.  $C_k^1 := C_k^g \cap SAT_{\mathcal{C}_{s4}}$ # Test $\mathcal{C}_{s4}$
5.  $C_k^2 :=$ prune($C_k^1$) # Safe pruning using anti-monotone constraints
6.  $C_k^3 := C_k^2 \cap SAT_{\mathcal{C}_{s6}}$ # Test $\mathcal{C}_{s6}$
7.  $C_k^4 := C_k^3 \cap SAT_{\mathcal{C}_{dbs}}$ #Test $\mathcal{C}_{dbs}$
8.  $\mathcal{L}_k := C_k^4 \cap SAT_{\mathcal{C}_{s8}}$ # Test $\mathcal{C}_{s8}$
9.  $k := k + 1$
10. until there is no more candidates
11. return $\bigcup_{i=1}^{k-1} \mathcal{L}_i \cap SAT_{\mathcal{C}_{s11}}$ # Test $\mathcal{C}_{s11}$
```

One problem is to choose how to split $\mathcal{C}$ into $\mathcal{C}_g \wedge \mathcal{C}_{s4} \wedge \mathcal{C}_{s6} \wedge \mathcal{C}_{dbs} \wedge \mathcal{C}_{s8} \wedge \mathcal{C}_{s11}$, i.e., at what step of the algorithm should a constraint be tested. Some of these can be the "true" constraint (the constraint that always evaluates to true), which causes a no-elimination step. First, a constraint can be pushed at the candidate generation step (step 3) if it is possible to have an optimized generation procedure (e.g., $\mathcal{C}_g$ might contain the conjunct $\mathcal{C}(S) = \ A \in S$). Second, it is possible to test constraints on steps 4, 6, 7, 8 and 11. Constraint checking that needs database scans, e.g., for $\mathcal{C}_{freq}$, is performed as step 7. Step 11 corresponds to a "generate and test approach". Depending on these choices, the generation and pruning steps might be rewritten to ensure completeness [6]. The safe pruning step (Step 5) can be seen as a constraint checking step too. Pruning means exactly removing itemsets $S$ that cannot verify an anti-monotone constraint because we already know (from a previous iteration) that a subset of $S$ does not verify it. Therefore, Step 6 can be rewritten $C_k^2 := C_k^1 \cap SAT_{\mathcal{C}_{prune}}$, where $\mathcal{C}_{prune}(S)$ is true iff $S$ cannot be pruned.
The kind of data we have to process when mining generalized sets is dense and highly-correlated. The Close algorithm [21, 3] makes the frequent set discovery tractable in such difficult cases. However, the original Close algorithm has been designed for mining frequent sets and not constrained itemsets. We revisited this algorithm in [6]. In fact, the optimization mechanism in Close can be formalized as a new pruning criterion due to a new anti-monotone constraint.

**Definition 7 (anti-monotone constraint $\mathcal{C}_{Close}$)** $\mathcal{C}_{Close}(S) = \forall S' \subset S \Rightarrow S \not\subseteq$ `closure`$(S')$ *where* `closure`*(S) is the maximal (for set inclusion) superset of S which has the same frequency as S.*

This anti-monotone constraint gives rise to a new safe pruning criterion (besides the Apriori trick based on frequency testing). An important property of the Close algorithm is its completeness, which means that it is possible to find $SAT_{\mathcal{C}_{freq}}$ knowing $SAT_{\mathcal{C}_{Close} \wedge \mathcal{C}_{freq}}$[2].

**Example 11** *Let us find* `closure`$(BC)$ *on example of Figure 2. Items $B$ and $C$ are simultaneously present in transactions* 1, 4, 5 *and* 8. *We can see that the maximal set of attributes ($\in$* `Items`*) that are true within these transactions is $\{B, C, \overline{D}\}$. Thus* `closure`$(BC) = BC\overline{D}$. *By the definition of closures, $\mathcal{F}(BC, \mathbf{r}) = \mathcal{F}(BC\overline{D}, \mathbf{r})$ (= 4/9). Since* `closure`$(B) = B$ *and* `closure`$(C) = C$, *$\mathcal{C}_{Close}(BC)$ is true and thus $BC$ can not be pruned (its frequency has to be computed). $\mathcal{C}_{Close}(BC\overline{D})$ is false and it means that it can be excluded from the counting procedure (as well as each of its supersets).*

[2]It is possible because Close outputs the closures of the sets in $SAT_{\mathcal{C}_{Close} \wedge \mathcal{C}_{freq}}$

Considering the CLOSE algorithm as an instance of our generic algorithm $\mathcal{G}$ (with the constraint $\mathcal{C}_{Close}$) enables to take advantage of CLOSE's improvements over APRIORI together with the ability to "push" constraints. To keep the completeness of CLOSE, [6] shows that Algorithm $\mathcal{G}$ can be used to search for sets which satisfy $\mathcal{C} = \mathcal{C}_{Close} \wedge \mathcal{C}_{am} \wedge \mathcal{C}_m$ where $\mathcal{C}_{am}$ is anti-monotone (e.g., it can contain $\mathcal{C}_{freq}$) and $\mathcal{C}_m$ is a monotone constraint.

**Definition 8 (monotone constraint)** *A monotone constraint $\mathcal{C}_m$ is the negation of an anti-monotone constraint. If $\mathcal{C}_m$ is monotone, $\neg\, \mathcal{C}_m$ is anti-monotone: $\mathcal{C}_m(S)$ is true $\Rightarrow \forall S' \supset S$, $\mathcal{C}_m(S')$ is true.*

**Example 12** *Continuing our running example, $\{A, B, C, D\} \subset S$ and $S \cap \{A, B, C\} \neq \emptyset$ are monotone constraints on $S$. An interesting case of a monotone constraint is $S \cap \texttt{Items}^+ \neq \emptyset$. This constraint, denoted as $\mathcal{C}_{al1p}$ says that every set has to contain $\underline{a}$t $\underline{l}$east $\underline{1}$ $\underline{p}$ositive attribute. It can be generalized as $\mathcal{C}_{alpp}$ to denote that one wants $\underline{a}$t $\underline{l}$east $\underline{p}$ $\underline{p}$ositive attributes.*

Monotone constraints are used for the effective candidate generation while anti-monotone constraints are the basis for safe pruning strategies.
In fact, CLOSE makes use of the fact that in many practical situations, there is quite much less closed frequent sets than frequent sets. In other terms, a lot of association rules with confidence 1 hold in the data (indeed, the fact that an itemset, e.g., $BC$, is not equal to its closure $\texttt{closure}(BC) = BC\overline{D}$ points out an association rule $BC \Rightarrow \overline{D}$ with confidence 1 in $\mathbf{r}$). To increase the potential for pruning, [3] propose to relax $\mathcal{C}_{Close}$ by considering almost-closures instead of closures.

**Definition 9 (anti-monotone constraint $\mathcal{C}_{Minex}$)** $\mathcal{C}_{Minex}(S) = \forall S' \subset S \Rightarrow S \not\subseteq \texttt{almost-closure}_{\delta}(S')$. *Given $\delta \ll |\mathbf{r}|$, if $Y$ denotes the $\delta$-almost-closure of a set $S$, $Y$ is the maximal (for set inclusion) superset of $S$ which has "almost" the same frequency as $S$: if $\texttt{sup}(S)$ is the absolute frequency of $S$ in $\mathbf{r}$, $\forall A \in Y \setminus S, \texttt{sup}(S \cup \{A\}) \geq \texttt{sup}(S) - \delta$. Clearly, when $\delta = 0$, an almost-closure turns to be a closure.*

Using $\mathcal{C}_{Minex}$ instead of $\mathcal{C}_{Close}$, more candidates can be pruned, but the frequency of some sets is known with some imprecision (related to the $\delta$ parameter). This has been formalized recently in [5]. It has also been shown in [5] that the real imprecision remains very low in practice.

## 4.2 An experimental validation

### 4.2.1 Dataset

We studied the use of several constraints for mining association rules with negations from a popular benchmark, the so-called `mushroom` data. This dataset is a binary matrix of 8124 lines. There are 119 columns for positive attributes. It contains 23 “1” per line that comes from the binarization of 23 attributes into exclusive attribute-value pairs. When encoding negative items, it leads to a matrix with 238 columns whose each line contains 119 “1”.

### 4.2.2 Constraints

Extracting rules is performed in two steps. First, we generate a collection of generalized sets, then we derive generalized rules. Concerning the processing of constraints, we proceed in a two-way process. First, we state the constraints for rules and then, we derive from them constraints for itemsets. Furthermore, we have weakened the derived constraints to improve the performances. Last, we discuss the effects of the selected constraints on the rule generation step, eventually adjusting it.
Concerning performances, we focus on the influence of constraints mainly for the generalized set extraction since the computational requirements of this step are orders of magnitude higher than the ones of the rule generation one. Concerning the propagation of constraints towards the rule generation step, we consider a few options, some of them need non-trivial changes to the standard rule derivation technique.

**Desired constraints** First, we decided not to consider rules involving only negative terms at the left-hand side. Formally, we will require the left-hand side to contain at least $p$ positive items.
Then, we want to extract only frequent rules. Considering the value of the frequency threshold, we wish the rules to have a frequency of few percents. In experiments, we will try to reach this value. When we do not succeed (the required resources are too high), we report the last 3 experiments corresponding to the lowest 3 values of frequency thresholds for which we succeeded.
We also use a constraint that is based on the confidence measure: we require the rules to have the confidence of at least 90%.

**From constraints on rules to constraints on itemsets** The left-hand side of each extracted rule has to contain at least $p$ positive terms. We observed that in order to evaluate the support and the confidence of a rule, we need the frequency of 2 sets: the first is made of all the items of the rule and the second is its left-hand side. In both, the number of positive items has to be at least $p$. Therefore we have to extract itemsets containing at least this number of positive items. This is expressed by the $\mathcal{C}_{alpp}$ constraint (see Ex. 11).
Since we do not choose which item must be at the right-hand side of the rule, all itemsets are candidates to produce interesting rules.

The frequency constraint on rules can be directly applied on itemsets. Therefore, we use $\mathcal{C}_{freq}$.
The constraint on rule confidence is difficult to transform into a useful constraint on itemsets (see, e.g., the discussion in [17]). Given the right-hand side constraint, the directly-derived constraint would be "extract itemset $X$ if at least one immediate subset of $X$ has a frequency of at most $\mathcal{F}(X, \mathbf{r})/0.9$ or at least one immediate superset of X has a frequency of at least $\mathcal{F}(X, \mathbf{r})$*0.9". If evaluated directly, this constraint does not lead to less computations since this evaluation is done within the rule generation step (it corresponds to the algorithm of generation of high confidence rules from itemsets). We thus do not derive any constraint on itemsets to cope with the confidence constraint.
$\mathcal{C}_{freq}$ has been successfully used to make the association rule discovery tractable on sparse matrixes. As we will show in the section 4.2.5, $\mathcal{C}_{freq}$ is severely inefficient, because typical complemented datasets are dense. In the experiments, we will use additional anti-monotone constraints $\mathcal{C}_{Close}$ or $\mathcal{C}_{Minex}$, which enable an effective pruning in such difficult cases.

**Improving the choice of constraints on itemsets** Even if we first tried to define strong constraints in order to produce as few as possible superfluous itemsets for the rule generation step, we now relax one of them for efficiency reasons.
$\mathcal{C}_{alpp}$ is clearly a monotone constraint that asks for at least $p$ positive attributes. [6] points out that it is often interesting to transform a monotone constraint of this type into a weaker constraint (i.e., accepting more sets) that, combined with pruning strategies, will reduce the number of candidate sets and will make the overall process much more efficient. In these experiments, instead of $\mathcal{C}_{alpp}$, we use: $\mathcal{C}_{alppoam1n} = \mathcal{C}_{alpp} \vee \mathcal{C}_{am1n}$
This constraint enforces $\underline{\text{a}}$t $\underline{\text{l}}$east $\underline{p}$ $\underline{\text{p}}$ositive attributes (a monotone constraint) $\underline{\text{o}}$r $\underline{\text{a}}$t $\underline{\text{m}}$ost $\underline{1}$ $\underline{\text{n}}$egative attribute (an anti-monotone constraint).
Thanks to these two disjuncts, we can:

- use the join-based procedure described in Section 3.1 to generate all candidates (but singletons),
- push the constraint into the candidate generation step (i.e., the earliest possible step), and
- prune out most (in our experiments) of the candidate sets of size $p$ and $p+1$ (corresponding to the first 2 passes of the basic version of the "generic" algorithm with $\mathcal{C}_{alpp}$ alone).

Pruning most of the sets of size $p$ and $p+1$ comes from the fact that pruning strategies such as the one in Apriori may use as few as one candidate to prune several candidates at following levels. It means that for $p = 2$ (or above) pairs that are mutually exclusive or occurring infrequently will avoid counting the frequency of a great part of candidates of size 3 and 4. The intuition is that all 3- and 4-itemsets that would be candidates under the $\mathcal{C}_{alpp}$ constraint alone are much more numerous than all frequent sets up to size 4 verifying $\mathcal{C}_{alppoam1n} \wedge \mathcal{C}_{freq}$, thanks to the $\mathcal{C}_{freq}$ conjunct.

**Outcome of the improved itemset constraints on the rule generation**
At the end of the itemset extraction step, we might apply the original $\mathcal{C}_{alpp}$ constraint (referred in the Step 11 of the generic algorithm $\mathcal{G}$ in Section 4.1), and thus end up with the initially intended result. It corresponds to a "generate and test" approach. An alternative is to keep them (with a relatively small overhead) to use them later, i.e., for some post-processing tasks. For instance, given the definition of the conviction [8] and of the J-measure [22], additional frequencies required by these measures (i.e., the frequencies of some singletons) would be thus available. We sketch later the two algorithms for rule generation that correspond to the alternative of keeping or rejecting the overhead inherited from the improved itemset constraints.

### 4.2.3 Implementing constraints on itemset extraction

To accomplish the pruning phase in the Apriori algorithm, the frequency of every immediate subset of an itemset is checked. Here, we check the frequency of every immediate subset only if it verifies the $\mathcal{C}_{alppoam1n}$ constraint.
Pushing the $\mathcal{C}_{alppoam1n}$ constraint into the candidate generation procedure requires a detailed explanation of some implementation issues and involves properties of the constraint itself.
The `generate` procedure (given in Section 3.1) contains a join-based procedure that generates a collection of candidates of size $k$ from a collection of frequent sets of size $k-1$. It considers a candidate (of size $k$) iff its two lexicographically-first subsets of size $k-1$ are frequent. Let us notice that in order obtain the two lexicographically-first $k-1$-item subsets of a $k$-itemset X, we remove from X the last or the last but one item.
If we decide that negative items are ordered after all the positive ones (i.e., $A < Z$, whereas $Z < \overline{A}$), we observe that for an itemset of size $k$ that verifies $\mathcal{C}_{alppoam1n}$, two lexicographically-first subsets of size $k-1$ verify $\mathcal{C}_{alppoam1n}$ too.
In order to prove it, let us consider first an itemset verifying the $\mathcal{C}_{alppoam1n}$ constraint that has at most 1 negative item (i.e., verifying the $\mathcal{C}_{am1n}$ disjunct).
Removing one item, we cannot increase the number of negative items, therefore all subsets (including the ones required by the joint-based procedure) verify the $\mathcal{C}_{am1n}$ disjunct.
Consider now an itemset verifying the $\mathcal{C}_{alppoam1n}$ constraint that has 2 or more

negative items. Since it does not verify the disjunct $\mathcal{C}_{am1n}$, it must verify $\mathcal{C}_{alpp}$. As we stated before, to obtain required subsets, we remove one of the last two items and we said that negative items are ordered after the positive ones. It follows that we remove one of the negative items (there are at least 2 of them at the end). Thus, the number of positive ones does not change and the subset verifies $\mathcal{C}_{alpp}$.
We have just seen, that every itemset of size $k$ verifying the $\mathcal{C}_{alppoam1n}$ constraint may be produced with `generate` procedure starting from itemsets of size $k-1$ verifying the $\mathcal{C}_{alppoam1n}$ constraint (the required subsets verify that constraint). We may turn this property into an operational statement: if one of two lexicographically-first subsets of size $k-1$ of an itemset of size $k$ verifying the $\mathcal{C}_{alppoam1n}$ constraint is missing from the output of the candidate generation procedure it is due to an other constraint. The above proof holds if all the combined constraints are anti-monotone (i.e., the conjunction or the disjunction with $\mathcal{C}_{freq}$ is anti-monotone). In that case, pruning is known to be safe.

So, the steps 8-9 of the `safe-pruning-on` procedure (see Section 3.1) are replaced with:

```
8'.         if  ($\mathcal{C}_{al(k-1)p}$(T) or $\mathcal{C}_{am1n}$(T)) and $T \notin \mathcal{L}_{k-1}$
9.              then  delete C from $C_k$
```

$\mathcal{C}_{al(k-1)p}$ denotes the constraint that checks for at least (k-1) positive attributes in a candidate set.
How to push now the constraint itself into the candidate generation step? It is relatively simple and efficient: after the generation of a $k$-item candidate with the join-based function, we check that when $k$ is lesser or equal to $p+1$, the $(k-1)^{th}$ item of the ordered candidate is a positive item.

So, the steps 4-5 of the `generate` procedure (see Section 3.1) become:

```
4.    where  r.item$_1$ = q.item$_1$, r.item$_2$ = q.item$_2$, ...,
5.           r.item$_{k-2}$ = q.item$_{k-2}$, r.item$_{k-1}$ < q.item$_{k-1}$,
5'.              ($k > p+1$ or r.item$_{k-1} \in$ Items$^+$)
```

### 4.2.4 Implementing constraints on rule derivation

The original Apriori algorithm derives frequent association rules (i.e., using $\mathcal{C}_{freq}$) as explained in Section 3.1. Let us recall this rule generation part:

```
9.     Phase 4 - rule generation
       **for each** $X \in FS_\gamma$ **do**
10.        **for each** $Y \in X$ **do**
11.            test-for-output($X \setminus \{Y\} \Rightarrow \{Y\}$)
```

When the criterion for selecting a rule is that its confidence must be greater or equal to the threshold $\phi$, `test-for-output` is defined as follows:

```
       **if** $\mathcal{F}(X, \mathbf{r}) / \mathcal{F}(X \setminus \{Y\}, \mathbf{r}) \geq \phi$
           **then** output ($X \setminus \{Y\} \Rightarrow \{Y\}$)
```

Assuming that we only have the frequent sets that verify $\mathcal{C}_{alpp}$, i.e., the input collection is now $FS_\gamma \cap SAT_{\mathcal{C}_{alpp}}$, the rule generation algorithm is the following:

```
9'.    Phase 4 - rule generation under $\mathcal{C}_{alpp} \wedge \mathcal{C}_{freq}$
       **for each** $X \in FS_\gamma \cap SAT_{\mathcal{C}_{alpp}}$ **do**
10'.       **for each** $Y \in X$ **such that** $\mathcal{C}_{alpp}(X \setminus \{Y\})$ **do**
11'.           test-for-output($X \setminus \{Y\} \Rightarrow \{Y\}$)
```

Notice, that testing $\mathcal{C}_{alpp}$ (in Line 10') is reduced here to the test if $X \setminus \{Y\}$ is a member of the input collection (every frequent set verifying $\mathcal{C}_{alpp}$ belongs to the input collection).
Let us assume now that we know the frequent sets that verify $\mathcal{C}_{alppoam1n}$, i.e., the input collection is now $FS_\gamma \cap SAT_{\mathcal{C}_{alppoam1n}}$. Since we keep more itemsets, at the rule generation step we must pay attention to this fact in order to not to produce too many rules. The algorithm is:

```
9".    Phase 4 - rule generation under $\mathcal{C}_{alppoam1n} \wedge \mathcal{C}_{freq}$
       **for each** $X \in FS_\gamma \cap SAT_{\mathcal{C}_{alppoam1n}}$ **such that** $\mathcal{C}_{alpp}(X)$ **do**
10".       **for each** $Y \in X$ **such that** $\mathcal{C}_{alpp}(X \setminus \{Y\})$ **do**
11".           test-for-output($X \setminus \{Y\} \Rightarrow \{Y\}$)
```

Given the implementation issues described in the previous subsection, notice that the tests in Lines 9" and 10" are very efficient: besides the membership test (to be done in line 10" as for rule generation under $\mathcal{C}_{alpp} \wedge \mathcal{C}_{freq}$), we must check that there are at least $p$ items and that $X.\text{item}_p$ is a positive item i.e., is a member of `Items`$^+$.
As previously pointed out, the side effect of the improved itemset constraint is that we can compute other interestingness measures for a rule $X \setminus \{Y\} \Rightarrow \{Y\}$.

For instance, consider $conv(X \setminus \{Y\} \Rightarrow \{Y\}, \mathbf{r})$, the conviction measure as defined in [8]:

$$conv(X \setminus \{Y\} \Rightarrow \{Y\}, \mathbf{r}) \;=\; \frac{\mathcal{F}(X \setminus \{Y\}, \mathbf{r}) * \mathcal{F}(\{\overline{Y}\}, \mathbf{r}))}{\mathcal{F}(X \setminus \{Y\} \cup \{\overline{Y}\}, \mathbf{r})} \tag{2}$$

It can be computed from the collection of itemsets that verify $\mathcal{C}_{alppoam1n} \wedge \mathcal{C}_{freq}$. Provided the frequencies of $X$ and of $X \setminus \{Y\}$, we can easily compute the conviction of the same rule as follows:

$$conv(X \setminus \{Y\} \Rightarrow \{Y\}, \mathbf{r}) \;=\; \frac{\mathcal{F}(X \setminus \{Y\}, \mathbf{r}) * \mathcal{F}(\{\overline{Y}\}, \mathbf{r}))}{\mathcal{F}(X \setminus \{Y\}, \mathbf{r}) - \mathcal{F}(X, \mathbf{r})}. \tag{3}$$

However, formula (3) requires the value of $\mathcal{F}(\overline{Y}, \mathbf{r})$, which will be generally present only in the collection corresponding to the improved itemset mining step (i.e., $\{\overline{Y}\}$ verifies $\mathcal{C}_{alppoam1n}$, independently of the value $p$ and of whether $Y$ is a positive or a negative item).
Details about this rule generation technique and the prototype used in these experiments are in [9].

### 4.2.5 Experimental results

The running prototype is implemented in C++. We use a PC with 512 MB of memory and a 500 MHz Pentium III processor under Linux operating system. Let us introduce the collection of constraints we used in our experiments.

$\mathcal{C}_0 = \mathcal{C}_{freq} \wedge \mathcal{C}_{al3poam1n}$
$\mathcal{C}_1 = \mathcal{C}_{freq} \wedge \mathcal{C}_{Close} \wedge \mathcal{C}_{al1poam1n}$
$\mathcal{C}_2 = \mathcal{C}_{freq} \wedge \mathcal{C}_{Close} \wedge \mathcal{C}_{al2poam1n}$
$\mathcal{C}_3 = \mathcal{C}_{freq} \wedge \mathcal{C}_{Close} \wedge \mathcal{C}_{al3poam1n}$
$\mathcal{C}_4 = \mathcal{C}_{freq} \wedge \mathcal{C}_{Minex} \wedge \mathcal{C}_{al1poam1n}$
$\mathcal{C}_5 = \mathcal{C}_{freq} \wedge \mathcal{C}_{Minex} \wedge \mathcal{C}_{al2poam1n}$
$\mathcal{C}_6 = \mathcal{C}_{freq} \wedge \mathcal{C}_{Minex} \wedge \mathcal{C}_{al3poam1n}$

In the different experiments we set different values of $p$, from the quite unconstraining requirement of 1 positive (in $\mathcal{C}_1$ and $\mathcal{C}_4$), through the moderately constraining requirement of 2 positives (in $\mathcal{C}_2$ and $\mathcal{C}_5$) to the quite restrictive requirement of 3 positives (in $\mathcal{C}_0$, $\mathcal{C}_3$ and $\mathcal{C}_6$).
The value of the frequency threshold ($\gamma$) is changed over experiments in order to observe the trend. Since we discovered an (expected) exponential trend, we draw the graphs with logarithmically scaled axes.
In these experiments, we also compare the performances of $\mathcal{C}_{Close}$ and $\mathcal{C}_{Minex}$ (note, the pairs $\mathcal{C}_1$ - $\mathcal{C}_4$, $\mathcal{C}_2$ - $\mathcal{C}_5$ and $\mathcal{C}_3$ - $\mathcal{C}_6$).
The value of the $\delta$ parameter in the $\mathcal{C}_{Minex}$ constraint is set to 200 in these experiments (recall that the number of lines is 8124).

$\mathcal{C}_{Minex}$ can take more advantage of correlations in data compared to $\mathcal{C}_{Close}$, especially when the maximal frequent sets are long (above 20 items) and obviously, the introduction of dense columns containing negative information gives rise to long frequent sets. As shown in [5], the trade off is to accept small incertitude (depending on the value of the mentioned $\delta$ parameter) on the itemset frequencies. This incertitude can be in practice very low compared with the theoretical error bound that is related to $\delta$ for a given mining task. Clearly, this incertitude also affects rule's support and confidence measures but it can be accepted in practical situations.
We included the use of $\mathcal{C}_0$ into the experiments to emphasize the intractability when using only $\mathcal{C}_{freq}$. We had to use with it the strongest version of the positive item inclusion constraint, i.e., $\mathcal{C}_{al3p}$. And even with the highest value of $\gamma$ that we used (i.e., 95%), the extraction based on $\mathcal{C}_{freq}$ was still intractable. We analyzed the partial result and we observed that there is only 1 frequent set composed of 3 positive items only. Even though, the number of sets verifying $\mathcal{C}_0$ (frequent sets made of this positive 3-itemset and various negative items) was so huge that the program could not tackle them with reasonable resources. As we have just seen, $\mathcal{C}_{freq}$ combined with the most favorable case of $\mathcal{C}_{alpp}$ still leads to an intractable extractions. We also observed that the collection of $< frequentset, support >$ pairs in this case was highly redundant. We thus used $\mathcal{C}_{Minex}$ or $\mathcal{C}_{Close}$ in the next experiments in order to get condensed representations of these generalized sets. The extractions were tractable and we have been able to lower the frequency threshold.
On Figure 3, we observe that the conjunct $\mathcal{C}_{Minex}$ drastically improves the performances when compared to $\mathcal{C}_{Close}$ under similar conditions. By using $\mathcal{C}_{Minex}$ instead of $\mathcal{C}_{Close}$, we trade some precision against the threshold frequency or the expressive power of left-hand side (lower $p$ leads to a richer descriptive "language").
Let us see it on an example. Assume that we proceed with CLOSE (i.e., constraints $\mathcal{C}_{freq}$ and $\mathcal{C}_{Close}$ pushed at step 5 in the abstract algorithm) to extract itemsets with at least 3 positive items ($p$=3) at the value 65% for the frequency threshold (see 1 on Figure 3). It takes about 4500 seconds and still the collection of rules seems to be limited to very strange cases. Indeed, each rule must involve at least 65% tuples and must have 3 or more positive items at the left-hand side.
If we want to generate rules with less constrained left-hand side (e.g., at $p = 2$), the frequent set extraction would require to rise the frequency threshold to about 90% in order to keep the extraction time similar (see 2 on Figure 3). With such a threshold, it probably outputs only trivial rules.
With a similar extraction time (about 4200 seconds), we discover frequent sets concerning at least 20% of tuples by using $\mathcal{C}_{Minex}$ (see 3 on Figure 3). If we now want rules with less restricted left-hand sides, i.e., $p$=2, resp. $p$=1, we

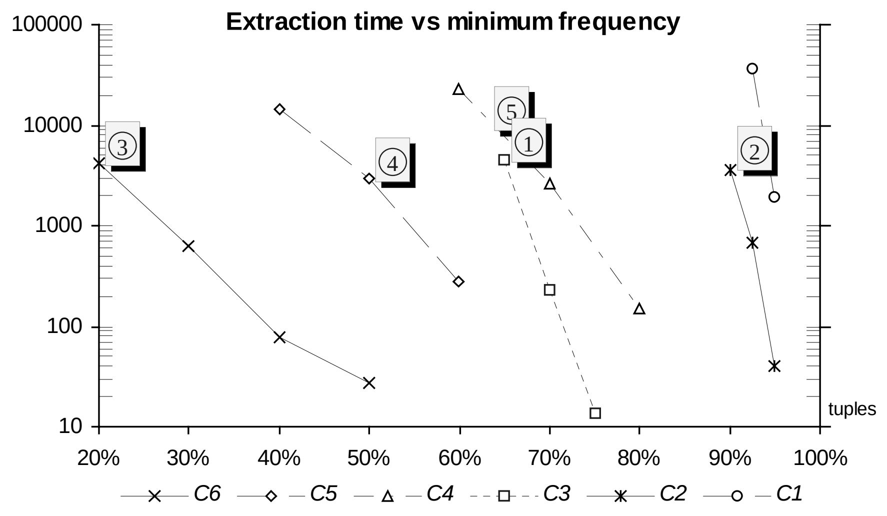

Figure 3: Experimental results on `mushroom`

can extract them within a similar time for rules concerning at least 50% (see 4 on Figure 3), resp. 65% (see 5 on Figure 3) of the tuples.
We used also the variation of frequency threshold and of the corresponding extraction times as a measure of the complexity of the process, given all other constraints. Observing the slope of the lines, we deduce that the complexity is much reduced by $\mathcal{C}_{Minex}$ (lesser relative change due to $\gamma$ than in case of $\mathcal{C}_{Close}$).

## 5 Conclusion

We discussed the possibility to derive association rules with negations using only the information about positive attributes. It gives rise to approximations of rule interestingness measures but has to be considered as a valuable direction of research. Then, we considered how given constraints on the desired rules can be "pushed" efficiently into a set mining algorithm. Using results from [6], we proposed useful constraints for mining association rules with negations, namely anti-monotone constraints (e.g., frequency constraint or constraint based on closures) and monotone constraints (e.g., ensuring the presence of positive attributes in the mined sets). This is an ongoing research and an experimental validation a benchmark dataset has been performed. We are now considering real-life datasets.

## References

[1] R. Agrawal, T. Imielinski, and A. Swami. Mining association rules between sets of items in large databases. In *Proc. ACM SIGMOD'93*, pages 207–216, 1993.

[2] R. Agrawal, H. Mannila, R. Srikant, H. Toivonen, and A. I. Verkamo. Fast discovery of association rules. In *Advances in Knowledge Discovery and Data Mining*, pages 307–328. AAAI Press, 1996.

[3] J.-F. Boulicaut and A. Bykowski. Frequent closures as a concise representation for binary data mining. In *Proc. PAKDD'00*, volume 1805 of *LNAI*, pages 62–73, Kyoto, JP, 2000. Springer-Verlag.

[4] J.-F. Boulicaut, A. Bykowski, and B. Jeudy. Towards a tractable approach for mining association rules with negations. In *Proc. FQAS'00*, pages 425 – 434, Warsaw, PL, Oct. 2000. Springer-Verlag.

[5] J.-F. Boulicaut, A. Bykowski, and C. Rigotti. Approximation of frequency queries by mean of free-sets. In *Proc. PKDD'00*, volume 1910 of *LNAI*, pages 75 – 85, Lyon, F, 2000. Springer-Verlag.

[6] J.-F. Boulicaut and B. Jeudy. Using constraints during set mining: Should we prune or not? In *Proc. BDA'00*, pages 221 – 237, Blois, F, Oct. 2000.

[7] J.-F. Boulicaut, M. Klemettinen, and H. Mannila. Modeling KDD processes within the inductive database framework. In *Proc. DaWaK'99*, volume 1676 of *LNCS*, pages 293–302, Florence, I, 1999. Springer-Verlag.

[8] S. Brin, R. Motwani, and C. Silverstein. Beyond market baskets: Generalizing association rules to correlations. In *Proc. ACM SIGMOD'97*, pages 265–276, Tucson, USA, 1997.

[9] A. Bykowski. R-miner : rule generation from frequent free-sets. Technical Report 2000-20, INSA Lyon, LISI, F-69621 Villeurbanne, Oct. 2000.

[10] L. Dehaspe and L. D. Raedt. Mining association rules in multiple relations. In *Proc. ILP'97*, volume 1297 of *LNAI*, pages 125–132. Springer-Verlag, 1997.

[11] F. Giannotti and G. Manco. Querying inductive databases via logic-based user defined aggregates. In *Proc. PKDD'99*, volume 1704 of *LNAI*, pages 125–135, Praha, CZ, 1999. Springer-Verlag.

[12] G. Grahne, L. V. S. Lakshmanan, and X. Wang. Efficient mining of constrained correlated sets. In *Proc. ICDE 2000*, pages 512–521, San Diego, USA, 2000.

[13] S. Guillaume, F. Guillet, and J. Philippé. Improving the discovery of association rules with intensity of implication (short paper). In *Proc. PKDD'98*, volume 1510 of *LNAI*, pages 318–327, Nantes, F, 1998. Springer-Verlag.

[14] J. Han and Y. Fu. Discovery of multiple-level association rules from large databases. In *Proc. VLDB'95*, pages 420–431, Zürich, CH, 1995.

[15] T. Imielinski and H. Mannila. A database perspective on knowledge discovery. *Communications of the ACM*, 39(11):58–64, Nov. 1996.

[16] M. Klemettinen. *Rule Discovery from Telecommunication Network Alarm Databases*. PhD thesis, Department of Computer Science, P.O. Box 26, FIN-00014 University of Helsinki, Jan. 1999.

[17] L. V. Lakshmanan, R. Ng, J. Han, and A. Pang. Optimization of constrained frequent set queries with 2-variable constraints. In *Proc. ACM SIGMOD'99*, pages 157–168, Philadelphia, USA, 1999.

[18] H. Mannila and H. Toivonen. Multiple uses of frequent sets and condensed representations. In *Proc. KDD'96*, pages 189–194, Portland, USA, 1996.

[19] H. Mannila and H. Toivonen. Levelwise search and borders of theories in knowledge discovery. *Data Mining and Knowledge Discovery*, 1(3):241–258, 1997.

[20] R. Ng, L. V. Lakshmanan, J. Han, and A. Pang. Exploratory mining and pruning optimization of constrained association rules. In *Proc. ACM SIGMOD'98*, pages 13–24, Seattle, USA, 1998.

[21] N. Pasquier, Y. Bastide, R. Taouil, and L. Lakhal. Efficient mining of association rules using closed itemset lattices. *Information Systems*, 24(1):25–46, 1999.

[22] P. Smyth and R. M. Goodman. An information theoretic approach to rule induction from databases. *IEEE Transactions on Knowledge and Data Engineering*, 4(4):301–316, Aug. 1992.