Intelligent Data Analysis 6(4):341-357. IOS.

Optimization of Association Rule Mining Queries

Baptiste Jeudy Jean-François Boulicaut

Institut National des Sciences Appliquées de Lyon
Laboratoire d'Ingénierie des Systèmes d'Information
Bâtiment Blaise Pascal
F-69621 Villeurbanne cedex, France
{Baptiste.Jeudy,Jean-Francois.Boulicaut}@lisi.insa-lyon.fr

Abstract

Levelwise algorithms (e.g., the Apriori algorithm) have been proved effective for association rule mining from sparse data. However, in many practical applications, the computation turns to be intractable for the user-given frequency threshold and the lack of focus leads to huge collections of frequent itemsets. To tackle these problems, two promising issues have been investigated during the last four years: the efficient use of user defined constraints and the computation of condensed representations for frequent itemsets, e.g., the frequent closed sets. We show that the benefits of these two approaches can be combined into a levelwise algorithm. It can be used for the discovery of association rules in difficult cases (dense and highly-correlated data). For instance, we report an experimental validation related to the discovery of association rules with negations.

keywords: association rules, constraints, condensed representations, inductive queries.

1 Introduction

Among the possible views on data mining processes, we consider a database perspective. From the user point of view, one can see a continuous spectrum of information needs, starting from very simple database queries (say, SQL queries like "What is the average amount of transactions"), moving to more complex aggregate information (say, OLAP queries like "What are the sales by product groups and regions") and data mining queries like "Give me interesting trends on sales" or "Which products are likely to be bought by someone who already bought a DVD player".

This suggests that it is useful to consider knowledge discovery processes as querying processes. It has motivated the *Inductive Database Framework*: an inductive database is a database that conceptually contains, in addition to normal data, all the patterns from a specified class of patterns that are true in the data. Inductive queries can be used to query the normal data or the patterns [13, 8].

Inductive databases are also suitable for supporting the *iterative* and *interactive* nature of discovery processes. For example, the result of a data mining query makes explicit some information that in turn motivates a new preprocessing step on the data, a modification of the mining algorithm parameters, or changing the algorithm itself. Several languages like `MINE RULE` [19], `DMQL` [12], or `M-SQL` [14] can be used to express more or less specific inductive queries.

However, there is still much work to do on the optimization of the evaluation of inductive queries. The bottleneck is the data mining algorithm itself. Indeed, its evaluation can last for several hours.

In this work, we focus on association rule mining. In a transactional database for the popular basket analysis task, an association rule $X \Rightarrow Y$, where X and Y are sets of products, describes the fact that customers are likely to buy products in Y when they buy products in X. The quality of a rule is given by different measures such as the frequency, i.e., X and Y occur together in many transactions, or the confidence, i.e., there is a high probability to find products from Y when we have the products from X in a transaction. The standard association rule mining process concerns the discovery of every association rule whose frequency and confidence are above user-defined thresholds [1, 2].

The computationally expensive step is clearly the computation of the so-called frequent itemsets from which the potentially interesting association

rules are derived. Levelwise algorithms, e.g., the well-know APRIORI algorithm [2], have been proved to be effective for frequent itemset mining when the data are sparse and lowly correlated. However, in many applications, the data are dense and/or highly correlated and frequent itemset computation turns to be intractable. Furthermore, even if it is tractable, the lack of focus leads to huge collections of frequent itemsets from which too many uninteresting patterns or rules will be derived.

During the last four years, two promising issues have been investigated to tackle these problems.

One can assume that only a subset of the collection of frequent itemsets is interesting: it leads to *constraint-based extraction of the frequent itemsets* [26, 20, 16]. These studies have considered various kinds of constraints, including "syntactic" constraints (e.g., an item must not appear in the itemsets) and constraints related to the so-called objective measures of itemset interestingness (e.g., the itemsets must be frequent). Using constraints enables to decrease the size of the output while improving the user guidance. The problem is to "push" efficiently the constraint checking step during itemset extraction, i.e., not to apply a simple "generate and test" strategy. Nice results have been discovered concerning the so-called anti-monotone, succinct and monotone constraints, i.e., a wide range of constraints [20].

Another promising approach concerns the *condensed representation of frequent itemsets* [17]. Instead of mining all the frequent patterns, only a particular subset of the frequent patterns collection is extracted. Ideally, this subset is much smaller than the original collection and can be extracted more efficiently, while allowing a fast regeneration of the whole collection of frequent patterns. For instance, several researchers have investigated the use of closed frequent sets [22, 4, 6, 24, 27]. In our group, we have recently designed a condensed representation based on the so-called δ-free sets [6]. Such a representation can be extracted from dense data (for which the previous approaches fail) and enables a fairly good approximation of the frequency of each frequent itemset.

To the best of our knowledge, combining these two frameworks has not been studied yet[1].

In this paper, we show that the benefit of these two approaches can be combined into levelwise algorithms. Doing so, new mining tasks can

[1]This paper is a significant extension of [7] and a complete study is to appear in the forthcoming P.D. thesis [15]

be considered like frequent itemset mining for low frequency thresholds or the discovery of frequent generalized itemsets (sets that combine positive and negative items). An experimental validation related to the discovery of association rules with negations is reported.

This paper is organized as follows. Section 2 introduces the notations and the problem settings. In Section 3 and 4 we survey two known strategies to compute the result of an association rule mining query. In Section 3, we focus on the efficient use of constraints. We also provide a new framework for constrained itemset mining. In Section 4, we consider the use of free sets and δ-free sets. In Section 5, we use our framework to introduce a new query evaluation strategy: the computation of constrained condensed representations. Finally, this new strategy is experimentally evaluated in Section 6 and Section 7 is a short conclusion.

2 Problem Settings and Notations

2.1 Notations

Assume that $\mathtt{Items}$ is a finite set of symbols denoted by capital letters, e.g., $\mathtt{Items} = \{\mathtt{A}, \mathtt{B}, \mathtt{C}, \ldots\}$. A *transaction* t is a subset of $\mathtt{Items}$. A *transactional database* T is a finite and non empty multiset $T = \{t_1, t_2, \ldots, t_n\}$ of transactions. An *itemset* is a subset of $\mathtt{Items}$ and a k-itemset is an itemset of size k; the set of k-itemset is denoted $\mathtt{Items}_k$. A transaction t *supports* an itemset S iff $S \subseteq t$. The *support* (denoted support(S)) of an itemset S is the multiset of all transactions of T that support S (e.g., support$(\emptyset) = T$). The *frequency* of an itemset S is defined by $\mathcal{F}(S) = |\text{support}(S)|/|\text{support}(\emptyset)|$ where $|.|$ denote the cardinality of the multiset (each transaction is counted with its multiplicity). An itemset S is γ-*frequent* in T if $\mathcal{F}(S) \geq \gamma$. Figure 1 provides an example of a transactional database and the supports and the frequencies of some itemsets. Notice that we use a string notation for sets, e.g., $\mathtt{AB}$ for $\{\mathtt{A}, \mathtt{B}\}$.

An *association rule* is denoted $X \Rightarrow Y$ where $Y \neq \emptyset$, $X \cap Y = \emptyset$, $X \subseteq \mathtt{Items}$ is the *body* of the rule and $Y \subseteq \mathtt{Items}$ is the *head* of the rule. The *support* and *frequency* of a rule are defined as the support and the frequency of the itemset $X \cup Y$. A transaction t *supports* a rule $X \Rightarrow Y$ if it supports $X \cup Y$. A transaction t is an exception for a rule $X \Rightarrow Y$ if it supports X and it does not support Y. The *confidence* of the rule is conf$(X \Rightarrow Y) =$

$\mathcal{F}(X \Rightarrow Y)/\mathcal{F}(X) = \mathcal{F}(X \cup Y)/\mathcal{F}(X)$. The confidence of the rule gives the conditional probability that a transaction supports $X \cup Y$ when it supports X. A rule with a confidence of one has no exception and is called a *logical rule*.

We now define constraints on itemsets and rules.

Definition 1 (constraint) *If $\mathcal{T}$ denotes the set of all transactional databases and $2^{\texttt{Items}}$ the set of all itemsets, an itemset constraint $\mathcal{C}$ is a predicate over $2^{\texttt{Items}} \times \mathcal{T}$. Similarly, a rule constraint is a predicate over $\mathcal{R} \times \mathcal{T}$ where $\mathcal{R}$ is the set of association rules. An itemset $S \in 2^{\texttt{Items}}$ (resp. a rule R)* satisfies *a constraint $\mathcal{C}$ in the database $T \in \mathcal{T}$ iff $\mathcal{C}(S,T) = true$ (resp. $\mathcal{C}(R,T) = true$). When it is clear from the context, we write $\mathcal{C}(S)$ (resp. $\mathcal{C}(R)$). Given a subset I of* Items*, we define $\mathrm{SAT}_{\mathcal{C}}(I) = \{S \in I, S \text{ satisfies } \mathcal{C}\}$ for an itemset constraint (resp. if J is a subset of $\mathcal{R}$, $\mathrm{SAT}_{\mathcal{C}}(J) = \{R \in J, R \text{ satisfies } \mathcal{C}\}$ for a rule constraint). $\mathrm{SAT}_{\mathcal{C}}$ denotes $\mathrm{SAT}_{\mathcal{C}}(2^{\texttt{Items}})$ or $\mathrm{SAT}_{\mathcal{C}}(\mathcal{R})$.*

We can now define important constraints: the frequency constraint for itemsets and the frequency and confidence constraints for association rules.

$$\mathcal{C}_{\gamma-\mathrm{freq}}(S) \equiv \mathcal{F}(S) \geq \gamma,$$
$$\mathcal{C}_{\gamma-\mathrm{freq}}(X \Rightarrow Y) \equiv \mathcal{F}(X \Rightarrow Y) \geq \gamma,$$
$$\mathcal{C}_{\theta-\mathrm{conf}}(X \Rightarrow Y) \equiv \mathrm{conf}(S) \geq \theta$$

where γ is the frequency threshold and θ the confidence threshold. A rule that satisfies $\mathcal{C}_{\gamma-\mathrm{freq}}$ is said γ-frequent. A rule that satisfies $\mathcal{C}_{\theta-\mathrm{conf}}$ is said θ-valid.

Example 1 *Consider the dataset of Figure 1 where* Items$= \{\texttt{A}, \texttt{B}, \texttt{C}, \texttt{D}\}$. *The 0.6-frequent itemsets are:*

$$\mathrm{SAT}_{\mathcal{C}_{0.6-\mathrm{freq}}} = \{\texttt{A}, \texttt{B}, \texttt{C}, \texttt{AC}, \texttt{BC}\}.$$

Assume that $\mathcal{C}_{size}(S) \equiv |S| \leq 2$ and $\mathcal{C}_{miss}(S) \equiv \texttt{B} \notin S$, then

$$\mathrm{SAT}_{\mathcal{C}_{size} \wedge \mathcal{C}_{miss}} = \{\texttt{A}, \texttt{C}, \texttt{D}, \texttt{AC}, \texttt{AD}, \texttt{CD}\} \text{ while } \mathrm{SAT}_{\mathcal{C}_{\gamma-\mathrm{freq}} \wedge \mathcal{C}_{\texttt{size}} \wedge \mathcal{C}_{\texttt{miss}}} = \{\texttt{A}, \texttt{C}, \texttt{AC}\}.$$

The 0.6-frequent and 0.7-valid rules are

$$\mathrm{SAT}_{\mathcal{C}_{0.6-\mathrm{freq}} \wedge \mathcal{C}_{0.7-\mathrm{conf}}} = \{\emptyset \Rightarrow \texttt{C}; \texttt{A} \Rightarrow \texttt{C}; \texttt{B} \Rightarrow \texttt{C}\}.$$

Consider now the rule constraint $\mathcal{C}_r(X \Rightarrow Y) \equiv \texttt{B} \in X$, then

$$\mathrm{SAT}_{\mathcal{C}_{0.6-\mathrm{freq}} \wedge \mathcal{C}_{0.7-\mathrm{conf}} \wedge \mathcal{C}_r} = \{\texttt{B} \Rightarrow \texttt{C}\}.$$

$$T = \begin{array}{c|c} t_1 & \texttt{ABCD} \\ t_2 & \texttt{BC} \\ t_3 & \texttt{AC} \\ t_4 & \texttt{CD} \\ t_5 & \texttt{ABCD} \\ t_6 & \texttt{ABC} \end{array}$$

Itemset	Support	Frequency
A	$\{t_1, t_3, t_5, t_6\}$	0.67
B	$\{t_1, t_2, t_5, t_6\}$	0.67
AB	$\{t_1, t_5, t_6\}$	0.5
AC	$\{t_1, t_3, t_5, t_6\}$	0.67
CD	$\{t_1, t_4, t_5\}$	0.5
ACD	$\{t_1, t_5\}$	0.33

Figure 1: Supports and frequencies of some itemsets in a transactional database.

2.2 Problem Settings

Definition 2 (constrained association rules mining task) *Given a transactional database T and a constraint $\mathcal{C}$, the constrained association rule mining task is the computation of the collection of the association rules that satisfy $\mathcal{C}$ (i.e., $\mathrm{SAT}_{\mathcal{C}}$) together with their frequencies and confidences. It provides $\{(R, \mathcal{F}(R), \mathrm{conf}(R)),\ R \in \mathrm{SAT}_{\mathcal{C}}\}$.*

The standard association rule mining problem introduced in [1] is to find all γ-frequent and θ-valid association rules. It is generally done in two steps: first the computation of all the frequent itemsets and then the computation of all the valid association rules that can be made from disjoint subsets of each frequent itemset. This second step is far less expensive than the first one because no access to the database is needed: only the collection of the frequent itemsets and their frequencies are needed. This is an important point, and we consider in the rest of this paper that the generation of association rules does not need to access the transactional database (it is still the case when using other objective measures such as the conviction [9] or the J-measure [25]). However, we allow any other constraint on the association rules and we do not require the occurrence of the frequency and/or confidence constraints.

Given an association rule constraint $\mathcal{C}$, let us study different strategies to support constrained association rules mining task.

2.3 Generate and Test Strategy

Assume one wants to perform the association rule mining task with a constraint $\mathcal{C}$ that is the conjunction of $\mathcal{C}_{\gamma-\mathrm{freq}}$, $\mathcal{C}_{\theta-\mathrm{conf}}$ and other constraints (i.e., $\mathcal{C} = \mathcal{C}_{\gamma-\mathrm{freq}} \wedge \mathcal{C}_{\theta-\mathrm{conf}} \wedge \ldots$).

This generate and test strategy consists of three steps.

- Step 1: compute γ-frequent itemsets;
- Step 2: generate all θ-valid association rules from frequent itemsets;
- Step 3: remove rules that do not satisfy the constraint $\mathcal{C}$ and output the remaining ones.

This strategy provides a correct set of association rules but it is inefficient. Even if the user is only interested in a small subset of the association rules, all frequent and valid association rules must be generated in Step 2. It is desirable that the needed resources (e.g., time, memory) are commensurate with the size of the output of the algorithm. To overcome this problem, it is necessary to "push" the constraints before Step 3. Since the more expensive step of this process is the generation of frequent itemsets (Step 1), the constraints should be pushed during Step 1.

3 Pushing Constraints

Strategies for pushing constraints have been studied for association rule mining (e.g., [26, 20, 16]), correlation discovery [11], sequential pattern mining (e.g., [28]), etc. However, not all constraints can be pushed. Assume that one wants to perform the association rule mining task with a constraint $\mathcal{C} = \mathcal{C}_1 \wedge \mathcal{C}_2$ where $\mathcal{C}_1$ can be pushed during the itemset extraction and $\mathcal{C}_2$ can not. A strategy that efficiently uses constraints is:

- Step 1: compute itemsets that satisfy $\mathcal{C}_1$;
- Step 2: generate all association rules;
- Step 3: remove rules that do not satisfy the constraint $\mathcal{C}_2$ and output the remaining ones.

This section provides a high level generic algorithm that mines itemsets under constraints (Step 1). It generalizes previously published algorithms.

APRIORI **algorithm**

1. $C_1 := \texttt{Items}_1$; $\mathcal{L}_0 = \{\emptyset\}$
2. $k := 1$
3. **while** $C_k \neq \emptyset$ **do**
4. Phase 1 - candidate safe pruning
   $C_k := \texttt{safe-pruning-on}(C_k, \mathcal{L}_{k-1})$
5. Phase 2 - frequency constraint checking (data scan)
   $\mathcal{L}_k := \mathrm{SAT}_{\mathcal{C}_{\gamma-\mathrm{freq}}}(C_k)$
6. Phase 3 - candidate generation for level k+1
   $C_{k+1} := \texttt{generate}_{apriori}(\mathcal{L}_k)$
7. $k := k + 1$

   od
8. **output** $\bigcup_{i=0}^{k-1} \mathcal{L}_i$

3.1 From Rule Constraints to Itemset Constraints

The transformation of rule constraints into itemset constraints is a difficult step. Few theoretical work has been done on this subject. Generally, this is done in an ad-hoc manner. Notice that not all rule constraints can be pushed, e.g., it is well known that the confidence constraint cannot be pushed.

Example 2 *Let us define a constraint* $\mathcal{C}_a(X \Rightarrow Y) \equiv Y \subseteq \{\texttt{A}, \texttt{B}\}$. $\mathcal{C}_a$ *is equivalent to* $\mathcal{C}_a \wedge \mathcal{C}'_a$ *where* $\mathcal{C}'_a(X \Rightarrow Y) \equiv |(X \cup Y) \cap \{\texttt{A}, \texttt{B}\}| \geq 1$ *is an itemset constraint. The constraint* $\mathcal{C}'_a$ *can be pushed during itemset extraction.*

3.2 Anti-Monotone Constraints and APRIORI Algorithm

We consider an abstract definition of the APRIORI algorithm [2] to support our discussion on the effective use of constraints. This algorithm performs the constrained itemset mining task when $\mathcal{C} = \mathcal{C}_{\gamma-\mathrm{freq}}$.

In this algorithm, and in the following ones, the frequencies of itemsets are not explicit for the sake of clarity (e.g., Line 5 of the algorithm should be $\mathcal{L}_k := \{(S, \mathcal{F}(S)),\ S \in \mathrm{SAT}_{\mathcal{C}_{\gamma-\mathrm{freq}}}(C_k)\}$ since APRIORI outputs the frequency of each frequent itemset).

APRIORI is a levelwise exploration of the lattice of itemsets (w.r.t. set inclusion). C_k denotes the k-itemsets that can be frequent (candidate itemsets).

During Phase 1, some of these k-itemsets are pruned. `safe-pruning-on` eliminates the candidates for which a subset of length k is not frequent: if S is not frequent, a superset of S cannot be frequent. During Phase 2, a database scan is performed to compute the frequency of the candidate itemsets. The infrequent ones are discarded. In Phase 3, frequent k-itemsets are used to compute candidate $k+1$-itemsets. $\texttt{generate}_{apriori}(\mathcal{L}_k)$ provides the candidates by fusion of two elements from $\mathcal{L}_k$ that share the same $k-1$ first items: $\texttt{generate}_{apriori}(\mathcal{L}_k)$=$\{A \cup B$, where $A, B \in \mathcal{L}_k$, A and B share the $k-1$ first items (in lexicographic order)$\}$.

It can be proved by induction on k that Apriori is correct and complete, i.e., $\cup_{i=0}^{k-1}\mathcal{L}_i = \mathrm{SAT}_{\mathcal{C}_{\gamma-\mathrm{freq}}}$. Indeed, its completeness relies on the anti-monotonicity of $\mathcal{C}_{\gamma-\mathrm{freq}}$.

Definition 3 (Anti-monotonicity) *An* anti-monotone *constraint is a constraint $\mathcal{C}$ such that for all itemsets S, S': ($S' \subseteq S \wedge S$ satisfies $\mathcal{C}$) $\Rightarrow$ S' satisfies $\mathcal{C}$.*

Example 3 *Some examples of anti-monotone constraints are: $\mathcal{C}_{\gamma-\mathrm{freq}}$, $\mathcal{C}(S) \equiv$ `A` $\notin S$ and $\mathcal{C}(S) \equiv \sum_{I \in S} I.price < 500$. This last constraint means that the sum of the prices of the items must be lower than five hundred. Other anti-monotone constraints are presented in [20]. Notice that a disjunction or a conjunction of anti-monotone constraints is an anti-monotone constraint.*

It is therefore rather straightforward to replace Step 5 of the Apriori algorithm with $\mathcal{L}_k := \mathrm{SAT}_{\mathcal{C}_{am}}(C_k)$ where $\mathcal{C}_{am}$ is any anti-monotone constraint without loosing completeness or correctness. This means that it is possible to push anti-monotone constraints using Apriori, and it can be shown that doing this saves computations (see Appendix).

3.3 Pushing Monotone Constraints

If the effective use of anti-monotone constraints is easy to understand, it is far more complex in the general case. In other terms, given an arbitrary constraint $\mathcal{C}$, it is not possible to use it in Apriori by simply replacing Step 5 with $\mathcal{L}_k := \mathrm{SAT}_{\mathcal{C}}(C_k)$. Doing this leads to the loss of the completeness of Apriori as shown in the next example. Indeed, there are two problems: the generation step and the pruning step. The generation step must be complete, i.e., it must not miss any itemset satisfying $\mathcal{C}$, and also the pruning step

(Phase 1) must be correct, i.e., it must not prune an itemset that satisfy the constraint.

Example 4 *Assume the constraint is* $\mathcal{C}(S) \equiv \mathtt{C} \in S$ *and that we replace Step 5 of the* Apriori *algorithm with* $\mathcal{L}_k := \mathrm{SAT}_{\mathcal{C}}(C_k)$. *The itemset* ABC *should be generated by* $\mathtt{generate}_{apriori}$ *from* AB *and* AC *but since* $\mathcal{C}(\mathtt{AB}) = false$, ABC *is not generated whereas* $\mathcal{C}(\mathtt{ABC}) = true$.

If the constraint is $\mathcal{C}(S) \equiv \mathtt{A} \in S$. *The itemset* ABC *is then correctly generated by* $\mathtt{generate}_{apriori}$ *from* AB *and* AC *but since* $\mathcal{C}(\mathtt{BC}) = false$, ABC *is incorrectly pruned whereas* $\mathcal{C}(\mathtt{ABC}) = true$.

To overcome these problems, we present an extension of Apriori to push conjunctions of anti-monotone and monotone constraints, i.e., when $\mathcal{C}$ can be written as $\mathcal{C}_{am} \wedge \mathcal{C}_m$.

Definition 4 (Monotonicity) *A* monotone *constraint is a constraint* $\mathcal{C}$ *such that for all itemsets* S, S': $(S \subseteq S' \wedge S$ *satisfies* $\mathcal{C}) \Rightarrow S'$ *satisfies* $\mathcal{C}$.

Example 5 $\mathcal{C}'_a$ *(see Example 2),* $\{\mathtt{A}, \mathtt{B}\} \subset \mathtt{S}$, $\sum_{I \in S} I.price > 100$ *and* $S \cap \{\mathtt{A}, \mathtt{B}, \mathtt{C}\} \neq \emptyset$ *are monotone constraints. Conjunctions or disjunctions of monotone constraints and negation of anti-monotone constraints are also monotone constraints.*

This generic algorithm uses procedures $\mathtt{prune}_{\mathtt{m}}$ and $\mathtt{generate}_m$ for the pruning step and the generation step respectively. The generation procedure assumes that the monotone constraint $\mathcal{C}_m$ is not trivial (i.e., $\mathcal{C}_m(\emptyset) \neq$ true).

Generic algorithm

1. $C_1 := \mathtt{generate}_m(\emptyset, 0)$; $\mathcal{L}_0 = \emptyset$
2. $k := 1$
3. **while** $C_k \neq \emptyset$ **do**
4. Phase 1 - candidate safe pruning
   $C_k := \mathtt{prune}_{\mathtt{m}}(C_k, \mathcal{L}_{k-1})$
5. Phase 2 anti-monotone constraint checking
   $\mathcal{L}_k := \mathrm{SAT}_{\mathcal{C}_{am}}(C_k)$
6. Phase 3 - candidate generation for level k+1
   $C_{k+1} := \mathtt{generate}_m(\mathcal{L}_k, k)$
7. $k := k + 1$

   od
8. **output** $\bigcup_{i=1}^{k-1} \mathcal{L}_i$

It is not necessary to check $\mathcal{C}_m$ during Phase 2 to ensure the correctness of this algorithm since $\texttt{generate}_m$ ensures that all candidate itemsets satisfy $\mathcal{C}_m$.

This algorithm is correct and complete, i.e., it outputs exactly itemsets in $\text{SAT}_{\mathcal{C}_{am} \wedge \mathcal{C}_m}$ together with their frequencies (proof in Appendix).

Generation Procedure. $\texttt{generate}_m$ deals efficiently with monotone constraints. It generates only candidate itemsets that satisfy the monotone constraint $\mathcal{C}_m$. It also ensures the completeness of the generic algorithm.

Let $\texttt{generate}_1(\mathcal{L}_k) = \{A \cup B$, where $A \in \mathcal{L}_k$ and B is a 1-itemset$\}$ and let $ms = \text{Max}_{S \in \mathcal{B}d^-_{\mathcal{C}'_{am}}} |S|$. $\mathcal{B}d^-_{\mathcal{C}'_{am}}$ is the negative border [18], i.e., the set of minimal itemsets (w.r.t. set inclusion) that do not satisfy $\mathcal{C}'_{am}$[2].

procedure $\texttt{generate}_m(\mathcal{L}, k)$
if $k = 0$ **then return** $\mathcal{B}d^-_{\mathcal{C}'_{am}} \cap \texttt{Items}_1$
elsif $k < ms$ **then return** $\texttt{generate}_1(\mathcal{L}) \cup (\mathcal{B}d^-_{\mathcal{C}'_{am}} \cap \texttt{Items}_{k+1})$
elsif $k = ms$ **then return** $\texttt{generate}_1(\mathcal{L})$
elsif $k > ms$ **then return** $\texttt{generate}_{apriori}(\mathcal{L})$
fi

Safe Pruning Procedure. The pruning procedure must also be rewritten to deal with monotone constraints. The aim of this procedure is to save constraint checking during Phase 2. Therefore, in order to be effective, this procedure must be less expensive than this constraint checking (this is the case when there is a data scan during Phase 2). It is also desirable that as most itemsets as possible are pruned by this procedure.

Indeed, an itemset S can be safely pruned only if one of its subset already considered by the algorithm does not satisfy $\mathcal{C}_{am}$. This is equivalent to the fact that one immediate subset of S (i.e., whose size is $|S| - 1$) satisfy $\mathcal{C}_m$ and not $\mathcal{C}_{am}$. This is used in the following procedure.

procedure $\texttt{prune}_m(C, \mathcal{L})$
$C' := C$
for all $S \in C$ **do for** all $S' \subset S$ such that $|S'| = |S| - 1$
do if $S' \notin \mathcal{L}_k$ and $\mathcal{C}_m(S') = \text{true}$
then delete S from C' **od**

[2]In other terms, all the subsets of the itemsets from $\mathcal{B}d^-_{\mathcal{C}'_{am}}$ satisfy $\mathcal{C}'_{am}$.

od
return C'

Our generic algorithm can be considered as a generalization of several algorithms like [26, 10, 20]. Conjunctions of monotone and anti-monotone constraints encompass every kind of constraints that have been "pushed" inside a levelwise algorithm (another kind of interesting constraint, the convertible constraints [23], can be pushed in depth-first exploration algorithms). The framework of succinct constraints introduced in [20] allows to find an effective generation procedure (i.e., an effective computation of the negative border $\mathcal{B}d^-_{\mathcal{C}'_{am}}$ in procedure $\texttt{generate}_m(\mathcal{L}, k)$).

3.4 Efficiency Issues

Pushing anti-monotone constraints leads to less constraint checking. However this is not true in the case of monotone constraints.

The motivation for pushing monotone constraint is that testing some anti-monotone constraints is very expensive, e.g., those like $\mathcal{C}_{\gamma-\text{freq}}$ that need to scan the data. One solution is to remove as soon as possible some candidates by pushing monotone constraints during the search space exploration.

However, pushing monotone constraints leads to less effective pruning. Pruning, in the case of anti-monotone constraints, is based on the fact that, if an itemset S does not satisfy the anti-monotone constraint $\mathcal{C}_{am}$, then every superset of S do not satisfy it either. However, if S was not generated because it does not satisfy some monotone constraint, $\mathcal{C}_{am}(S)$ is unknown. Then, it is not possible to prune the supersets of S in the case where $\mathcal{C}_{am}(S)$ is false.

The tradeoff is as follows: when a monotone constraint is pushed, it might save tests on more costly constraints. However, the results of these tests could have lead to more effective pruning. The following example shows that the "generate and test" strategy is sometimes more efficient.

Example 6 *Assume the constraint* $\mathcal{C}(S) \equiv |S| \geq 10 \wedge \mathcal{C}_{\gamma-\text{freq}}(S)$*, i.e., S contains more than ten items and S is frequent. If the constraint* $|S| \geq 10$ *(a monotone constraint) is pushed into the candidate generation step, no candidate of size lower than ten is generated. Every candidate of size ten is generated and its frequency is tested in one database pass. This leads to at least* $\binom{n}{10}$ *candidates and, as soon as n is large, this turns to be intractable for any frequency threshold.*

A "generate and test" strategy computes every frequent itemset and needs several passes over the database. Then all itemsets whose size is lower than ten are removed. This strategy remains tractable for a reasonable frequency threshold even for a large n.

Previous works have shown that pushing monotone constraint can be efficient. However, Example 6 shows that it is *not* always the case. There is no theoretical result about this issue and it is still an open problem to decide whether it is profitable to push a particular monotone constraint or not. Indeed, this generally depends on the dataset.

4 Using Condensed Representations

The Apriori algorithm explores the itemset lattice to find all the frequent itemsets. However, the number of frequent itemsets can be exponential in the size of `Items`. If the size of `Items` is n, the size of the itemset lattice is 2^n and many of these itemsets can be frequent for the given frequency threshold. This is the case in dense/highly-correlated data like for instance census data. In this section, we study strategies based on the extraction of condensed representations to efficiently mine association rules in such dense/highly-correlated data.

In these strategies, a new step is inserted before the generation of the frequent itemsets. The whole strategy is then:

- Step 1: compute condensed representation from transactional data (using the frequency constraint);
- Step 1': compute frequent itemsets from the condensed representation;
- Step 2: generate all association rules with a sufficient confidence from frequent itemsets (using the confidence constraint);
- Step 3: finally remove rules that do not satisfy the whole constraint $\mathcal{C}$ and output the remaining rules.

Condensed representation is a general concept (see, e.g., [17]). The rationale behind condensed representations for frequent itemsets is to try to count the frequency of less itemsets and then to be able to deduce the frequency of the remaining itemsets efficiently. Thus, we are looking for condensed

representation that are subsets of the collection of frequent itemsets. The desirable properties of such a condensed representation are:

- It can be computed efficiently and it should be smaller than the whole collection of frequent itemsets,
- The whole collection of frequent itemsets can be efficiently regenerated from the condensed representation.

These properties means it is desirable that Step 1 and Step 1' of this strategy are done more efficiently than the direct generation of frequent itemsets with APRIORI.

Several algorithms exist that uses various condensed representations of frequent itemsets: CLOSE [22], CLOSET[24], CHARM [27], MIN-EX [4, 6], or PASCAL [3]. These algorithms provide different condensed representations[3]: frequent closed itemsets (CLOSE, CLOSET, CHARM), frequent free itemsets (or key patterns) (MIN-EX, PASCAL), or frequent δ-free itemsets (MIN-EX).

These algorithms enable tractable frequent itemsets extractions from dense and highly-correlated data, i.e., tractable extractions for frequency thresholds on which APRIORI-like algorithms are clearly intractable.

4.1 Mining Free Itemsets

Closed and free itemsets are closely related. Indeed, all the algorithms that compute closed or free itemsets compute in fact both. For our discussion we will follow the point of view of [6] and for the sake of clarity focus only on free itemsets. Readers interested in the relation between free and closed itemsets should refer to the cited works.

These algorithms use logical rules (association rules with a confidence of one) to infer some frequencies. Given an itemset $S = X \cup Y$ and a logical rule $X \Rightarrow Z$ then the frequency of $X \cup Y \cup Z$ does not need to be computed since it has the same frequency as S. When comparing to APRIORI, there is the overhead of computing the logical rules, but these rules can save the frequency computation of many itemsets.

The itemsets whose frequencies cannot be inferred by such logical rules are called *free itemsets*.

[3]Notice that most of these works (with the exceptions of [4, 6]) do not formalize their contributions within the framework of condensed representation w.r.t. frequency queries

Definition 5 (free itemset) *An itemset S is free if no logical rule holds between its items, i.e., it does not exist two distinct subsets X, Y of S such that $X \Rightarrow Y$ is a logical rule.*

An algorithm using free itemsets to mine frequent itemsets can then easily be designed. We can replace Phase 2 of Apriori by:

for all $S \in C_k$ **do**
  if S is free **then** compute $\mathcal{F}(S)$ and logical rule whose body is S from the database
  else compute $\mathcal{F}(S)$ using already computed free itemsets and logical rules
od

However, in real implementations there are two distinct steps. First, the mining of all frequent free itemsets and then the regeneration of all frequent itemsets.

Example 7 *In the data of Figure 1, if the frequency threshold is* 0.1*, then all subsets of* ABCD *are frequent. The frequent free itemsets are* $\{\emptyset, \texttt{A}, \texttt{B}, \texttt{D}, \texttt{AB}, \texttt{AD}, \texttt{BD}\}$*. The frequencies of all other itemsets can be deduced from the frequencies of the free ones. E.g., the frequencies of* BC *and* ABD *can be computed from the frequencies of* B *and* BD*: Since* $\texttt{B} \Rightarrow \texttt{C}$ *and* $\texttt{BD} \Rightarrow \texttt{A}$ *are logical rules,* $\mathcal{F}(\texttt{BC}) = \mathcal{F}(\texttt{B})$ *and* $\mathcal{F}(\texttt{ABD}) = \mathcal{F}(\texttt{BD})$*.*

4.2 δ-free Itemsets

In the Min-Ex algorithm introduced in [4, 6], the concept of free itemset is extended, providing new possibilities for pruning. However, this efficiency improvement is done at the expense of precision: the frequencies of the frequent itemsets are only known with a bounded error.

Algorithms using free itemsets use logical rules to prune some itemsets because their frequency can be inferred from the frequency of free itemsets. However, to be efficient, these algorithms need that such logical rules hold in the data. If it is not the case, then the frequent free sets are exactly the frequent sets and there is no improvement over Apriori.

The Min-Ex algorithm uses rules with few exceptions to further prune the itemset lattice. Given an itemset $S = X \cup Y$ and a rule $Y \Rightarrow Z$ with less than δ exceptions, then the frequency of $X \cup Y \cup Z$ can be approximated by

the frequency of S. The MIN-EX algorithm uses this property to prune the itemset S. More formally, MIN-EX computes only the δ-free itemsets.

Definition 6 (δ-free itemsets) *Let δ be an integer and S an itemset. S is δ-free if it does not exist two distinct subsets X and Y of S such that $X \Rightarrow Y$ has less ($\leq$) than δ exceptions.*

Example 8 *In the database of Figure 1, The 0.1-frequent 1-free itemsets are $\{\emptyset, \mathtt{A}, \mathtt{B}, \mathtt{D}\}$. $\mathtt{AB}$ is not 1-free because the rule $\mathtt{A} \Rightarrow \mathtt{B}$ has only one exception.*

Notice that with $\delta = 0$, 0-free itemsets are exactly free itemsets. If $\delta \geq \delta'$, then every δ-free itemset is also a δ'-free itemset. This means that larger values of δ lead to more efficient pruning (there are less δ-free itemsets) but also larger errors on the frequencies of itemsets when they are regenerated from the δ-free ones. The bounding error is given by the next proposition from [6].

Proposition 1 *Let S be a frequent itemset and n be the number of rows in the database. Then there exists a frequent δ-free itemset F such that:*

$$\mathcal{F}(F) \geq \mathcal{F}(S) \geq \mathcal{F}(F) - \frac{\delta}{n}(|S| - |F|).$$

This means that the error done on $\mathcal{F}(S)$ when using frequent δ-free itemsets to estimate it is bounded by $\frac{\delta}{n}(|S| - |F|)$.

Typical δ values range from zero to a few hundreds. With a database size of several tens of thousands of rows, the typical error is below few percents.

4.3 Discussion

Frequent closed/free itemsets are interesting for several reasons:

- They are far less numerous than frequent itemsets in dense/correlated data (and therefore faster to compute, easier to store and manipulate),
- In the degenerated cases where there is almost the same number of frequent closed itemsets and frequent itemsets, then the overhead with respect to APRIORI is typically less than 5%,
- It is possible to generate efficiently all the frequent itemsets (and their frequencies) from the closed ones (Step 1'),

- It is possible to derive (non redundant) association rules directly from closed frequent itemsets without generating all frequent non-closed ones (see, e.g., [27, 21]).

If one wants better efficiency and can trade some precision, δ-free itemsets are even less numerous than closed/free ones. This can be very interesting especially when there is few logical rules holding in the data, i.e., when computing closed itemsets does not bring a significant speedup.

Also, the δ-free itemset discovery task enables to mine at frequency thresholds for which no other technique is available, i.e., for applications where providing inaccurate results is better than no result at all.

5 Extraction of Condensed Representations under Constraints

We have been discussing the interest of using condensed representations. However, to the best of our knowledge, no algorithm exists that push user-defined constraints when mining condensed representations of frequent itemsets. In this section, we propose to use the generic framework we introduced in Section 3 to propose a new strategy for constrained association rule mining:

- Step 1: compute condensed representation under constraints from the transactional data;
- Step 1': compute constrained itemsets from the condensed representation;
- Step 2: generate constrained association rules with a sufficient confidence from constrained itemsets (using the confidence constraint);
- Step 3: finally remove rules that do not satisfy the whole constraint $\mathcal{C}$ and output the remaining rules.

This strategy combines the advantages of using condensed representation and pushing constraints.

5.1 Constraint for Free Itemsets

In this section we presents an original point of view on condensed representations extraction algorithms: One can consider these algorithms as instantiations of our generic algorithm (Section 3) for itemset mining under constraint when introducing new constraints. This novel approach will enable a simple definition of the extraction of constrained condensed representations in the next section.

Assume a constraint for free itemsets denoted by $\mathcal{C}_{\text{free}}$. It is defined by:

$$\mathcal{C}_{\text{free}}(S) \equiv S \text{ is free.}$$

This constraint is anti-monotone (see Appendix) and can therefore be used in our generic algorithm of Section 3. The CLOSE algorithm can be considered as an instantiation of our generic algorithm with the constraint $\mathcal{C}_{\gamma-\text{freq}} \wedge \mathcal{C}_{\text{free}}$.

The implementation issues related to $\mathcal{C}_{\text{free}}$ constraint are discussed in the Appendix.

5.2 Contextual Free Itemsets

Now, it seems straightforward to search for itemsets which satisfy a constraint $\mathcal{C} = \mathcal{C}_{\text{free}} \wedge \mathcal{C}_{am} \wedge \mathcal{C}_m$ using the generic algorithm. However, two problems arise.

- The $\mathcal{C}_{\text{free}}$ constraint checking problem: Given an itemset S, some of its immediate subsets might not satisfy $\mathcal{C}_m$. To compute $\mathcal{C}_{\text{free}}(S)$, the closure of *all* its immediate subsets are needed. However, if a subset of S does not satisfy $\mathcal{C}_m$, its closure is unknown.

- The regeneration problem: After the computation of the free itemsets, our aim is to regenerate itemsets that satisfy $\mathcal{C}_{am} \wedge \mathcal{C}_m$. However, the knowledge of the free itemsets that satisfy $\mathcal{C}_{am} \wedge \mathcal{C}_m$ is not enough for this purpose.

To cope with these problems, we have to mine "contextual" free itemsets.

Definition 7 (contextual free itemset) *An itemset S is contextual free with respect to a monotone constraint $\mathcal{C}_m$ if it does not exist two distinct subsets X, Y of S such that X satisfy $\mathcal{C}_m$ and $X \Rightarrow Y$ is a logical rule.*

The associated constraint is $\mathcal{C}_{free \wedge \mathcal{C}_m}$. $\mathcal{C}_{free \wedge \mathcal{C}_m}(S)$ is true if S is contextual free with respect to $\mathcal{C}_m$. This new constraint solves the two problems. It is possible to compute $\mathrm{SAT}_{\mathcal{C}_{free \wedge \mathcal{C}_m} \wedge \mathcal{C}_{am} \wedge \mathcal{C}_m}$ with our generic algorithm and the collection of the itemsets from $\mathrm{SAT}_{\mathcal{C}_{free \wedge \mathcal{C}_m} \wedge \mathcal{C}_{am} \wedge \mathcal{C}_m}$ with their frequencies provides the needed information to generate the itemsets from $\mathrm{SAT}_{\mathcal{C}_{am} \wedge \mathcal{C}_m}$ and their frequencies.

5.3 Constraints for δ-free Itemsets

It is possible to extend the definitions of $\mathcal{C}_{\text{free}}$ and $\mathcal{C}_{free \wedge \mathcal{C}_m}$ constraints to use δ-freeness instead of freeness.

We can define $\mathcal{C}_{\delta - Free}$, $\mathcal{C}_{\delta - Free \wedge \mathcal{C}_m}$. The latter one can be used in the generic algorithm to mine $\mathrm{SAT}_{\mathcal{C}_{\delta - Free \wedge \mathcal{C}_m} \wedge \mathcal{C}_{am} \wedge \mathcal{C}_m}$. From these itemsets and their frequencies, it is then possible to regenerate itemsets belonging to $\mathrm{SAT}_{\mathcal{C}_{am} \wedge \mathcal{C}_m}$ and their frequencies with a bounded error.

6 An Experimental Validation

We consider an experiment motivated by the search for association rules with negations [5]. Only some results concerning the discovery of generalized sets (from which association rules with negations are derived) are given here.

6.1 Generalized Sets

Let $\texttt{Items}^+ = \{\texttt{A}, \texttt{B}, ...\}$ be a finite set of symbols called the positive items and a set $\texttt{Items}^-$ of same cardinality as $\texttt{Items}^+$ whose elements are denoted $\overline{\texttt{A}}$, $\overline{\texttt{B}}$, ... and called the negative items. Given a transactional database $\mathcal{T}$ over $\texttt{Items}^+$, let us define a complemented transactional database over $\texttt{Items} = \texttt{Items}^+ \cup \texttt{Items}^-$ as follows: for a given transaction $t \in \mathcal{T}$, we add to t negative items corresponding to positive items not present in t. Generalized itemsets are subsets of $\texttt{Items}$ and can contain positive and negative items.

6.2 Datasets

Our experiments were done on two datasets. The first one is a benchmark, the so-called `mushroom` data. This dataset is a binary matrix of 8124 rows. Each row contains 23 discrete attributes. Theses attributes are binarized

into exclusive attribute-value pairs. This leads to a binary matrix with 119 columns and 23 "1" per row. When encoding negative items, it leads to a matrix with 238 columns whose each row contains 119 "1".

The second dataset is from the French national institute of statistics (INSEE). In this dataset, each row represents a French town and each column represents a kind of service (e.g., bank, insurance company, etc), a "1" in "bank" column means that there is at least one bank in the town. In this dataset, there are about 37000 rows and 59 columns with an average number of "1" per row of 4. When encoding negative items, it leads to a matrix with 118 columns whose each row contains 59 "1".

The former dataset is quite small but it is known to be tough due to the high correlation between the attributes and its density (for positive attributes). The latter dataset is larger but it is sparse (4 "1" per row on average for positive attributes) and less correlated. These two different datasets let us compare our approach on different types of datasets.

6.3 Experiments

Extracting all generalized frequent itemsets is generally intractable and not very interesting from the user point of view. Many frequent generalized itemsets involve only negative items. Rather, we assume that the user wants to extract frequent generalized itemsets that do not involve only negative items. We introduce the constraint $\mathcal{C}_{alpp}(S)$ which is true when S involves at least p positive items. This is obviously a monotone constraint.

In our experiments we compare different strategies to extract frequent generalized itemsets with at least 3 positive items.

- Strategy I: this is the generate and test strategy. First extract all frequent generalized itemsets (with Apriori) and then remove those that have less than 3 positive attributes (Section 2.3).
- Strategy II: push the $\mathcal{C}_{al3p}$ constraint during extraction (Section 3).
- Strategy III: using condensed representation to extract frequent generalized itemsets (Section 4). There are two sub-strategies depending on the condensed representation used.
  - Strategy IIIa uses free itemsets (Section 4.1);
  - Strategy IIIb uses δ-free itemsets (Section 4.2).

- Strategy IV: this is our new strategy, pushing the $\mathcal{C}_{al3p}$ constraint during the condensed representation extraction. There are two sub-strategies depending on the condensed representation used.
  - Strategy IVa uses free itemsets (Section 4.1);
  - Strategy IVb uses δ-free itemsets (Section 4.2).

The experiments were conducted on a 500 MHz Pentium III with 768 MB of memory. In the next table we present, for each strategy and each dataset, the lowest frequency threshold for which the extraction succeeded within 1000 seconds.

Frequency threshold achieved for 1000s

	mushroom	Insee
Strat. I	$> 95\%$	94%
Strat. II	$> 95\%$	3.6%
Strat. IIIa	$> 95\%$	35%
Strat. IIIb	75%	34%
Strat. IVa	70%	3.9%
Strat. IVb	28%	3.2%

On the mushroom dataset, Strategies I, II and IIIa are intractable, meaning that even with a high frequency threshold of 95%, the extraction does not finish within 1000s. Strategy IIIb is tractable but does not reach a low threshold. On this dataset, only our Strategy IVb does achieve a good threshold (28%). Further comparison between Strategies IIIb, IVa and IVb is shown on Figure 2. We can see that for any frequency threshold Strategy IVb is several order of magnitude faster than other strategies.

On the INSEE dataset, the use of constraint (Strategy II) is very profitable whereas condensed representations are less interesting (Strategy III). In this experiments, Strategy IVa is worse than Strategy II. This is because there are few logical rules in this dataset, therefore the use of free sets is not valuable. However combining constraints and δ-free sets (Strategy IVb) is still the best strategy (although this is less impressive than in the mushroom dataset). On Figure 3, we see that Strategy IVb is better than other strategies only for lowest thresholds.

Finally, on dense and correlated datasets like the mushroom dataset, our strategy is very efficient and allow extraction at frequency thresholds never

reached before. On sparse dataset (where condensed representations are known to not perform very well), our strategy is not surprisingly less efficient. However, there is still a gain for lowest frequency thresholds.

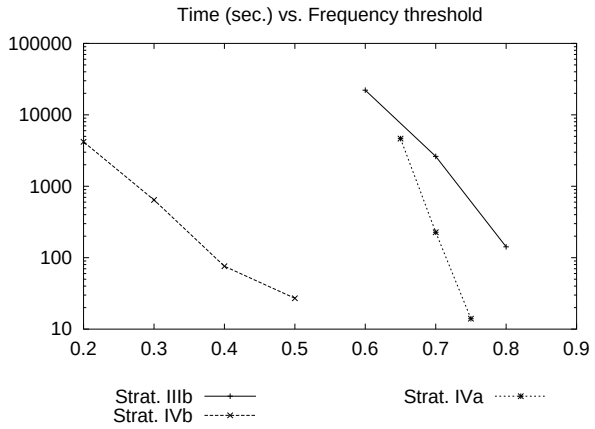

Figure 2: Mining generalized sets from mushroom dataset.

7 Conclusion

In this paper, we focus on the evaluation of association rule queries. We considered two important strategies: pushing constraints and computing condensed representations. We proposed a generic algorithm for itemset mining under a conjunction of monotone and anti-monotone constraints and an original point of view on the extraction of condensed representation. Then the introduction of a new kind of condensed representation (contextual free itemsets) allows us to combine the two previous strategies into a completely new one: the extraction of condensed representations under constraints.

The experiments show the added value of this approach in dense and correlated data: none of the two existing strategies were tractable at low frequency thresholds whereas the extraction of constrained condensed representations was possible.

We think that the study of condensed representations for frequent itemsets is promising: using them generally dramatically increase performance (w.r.t. time and space) and, when we get no improvement, the overhead

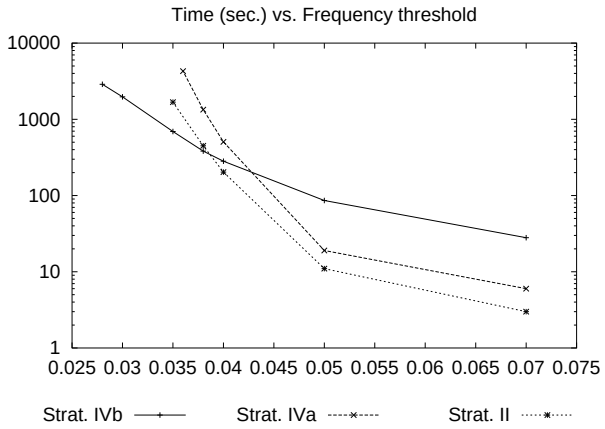

Figure 3: Mining generalized sets from INSEE dataset.

is small. If the efficiency is of a primary importance, it is even possible to trade more efficiency against precision using condensed representations like the (δ-free sets).

It would be interesting now to revisit earlier works on the direct extraction of association rules[27, 21] when considering condensed representations without a regeneration of all the frequent itemsets. It might be possible to extract constrained rules directly from constrained condensed representations.

Acknowledgements. The authors thank Arthur Bykowski for his implementation of the MIN-EX algorithm. This research is partly funded by the European contract cInQ IST-2000-26469 (Future and Emerging Technologies arm of the IST programme).

References

[1] R. Agrawal, T. Imielinski, and A. Swami. Mining association rules between sets of items in large databases. In *Proceedings of ACM SIGMOD Conference on Management of Data SIGMOD'93*, pages 207–216, Washington, D.C., USA, May 1993. ACM.

[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, Menlo Park, CA, 1996.

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

[4] J.-F. Boulicaut and A. Bykowski. Frequent closures as a concise representation for binary data mining. In *Proceedings of the Fourth Pacific-Asia Conference on Knowledge Discovery and Data Mining PAKDD'00*, volume 1805 of *Lecture Notes in Artificial Intelligence*, pages 62–73, Kyoto, JP, Apr. 2000. Springer-Verlag.

[5] J.-F. Boulicaut, A. Bykowski, and B. Jeudy. Mining association rules with negations. Technical Report 2000-19, INSA Lyon - LISI, Institut National des Sciences Appliquées de Lyon, Bâtiment Blaise Pascal, F-69621 Villeurbanne, France, 2000.

[6] J.-F. Boulicaut, A. Bykowski, and C. Rigotti. Approximation of frequency queries by mean of free-sets. In *Proceedings of the Fourth European Conference on Principles and Practice of Knowledge Discovery in Databases PKDD'00*, volume 1910 of *Lecture Notes in Artificial Intelligence*, pages 75–85, Lyon, F, Sept. 2000. Springer-Verlag.

[7] J.-F. Boulicaut and B. Jeudy. Mining free-sets under constraints. In *Proceedings of the International Database Engineering & Applications Symposium IDEAS'01*, pages 322–329, Grenoble, F, July 2001. IEEE Computer Society.

[8] J.-F. Boulicaut, M. Klemettinen, and H. Mannila. Modeling KDD processes within the inductive database framework. In *Proceedings of the First International Conference on Data Warehousing and Knowledge Discovery DaWaK'99*, volume 1676 of *Lecture Notes in Computer Science*, pages 293–302, Florence, I, Sept. 1999. Springer-Verlag.

[9] S. Brin, R. Motwani, and C. Silverstein. Beyond market baskets: Generalizing association rules to correlations. In J. M. Peckman, editor, *Proceedings of ACM SIGMOD Conference on Management of Data SIGMOD'97*, pages 265–276, Tucson, AZ, May 1997. ACM.

[10] B. Goethals and J. van den Bussche. On implementing interactive association rule mining. In *Proceedings of the ACM SIGMOD Workshop on Research Issues in Data Mining and Knowledge Discovery DMKD'99*, Philadelphia, USA, May 1999.

[11] G. Grahne, L. V. S. Lakshmanan, and X. Wang. Efficient mining of constrained correlated sets. In *Proceedings of the 16th International Conference on Data Engineering ICDE 2000*, pages 512–521, San Diego, USA, 2000. IEEE Computer Society Press.

[12] J. Han, Y. Fu, W. Wang, K. Koperski, and O. Zaiane. DMQL: a data mining query language for relational databases. In *SIGMOD Workshop on Research Issues on Data Mining and Knowledge Discovery DMKD'96, Montreal (Canada)*, June 1996.

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

[14] T. Imielinski and A. Virmani. MSQL: A query language for database mining. *Data Mining and Knowledge Discovery*, 3(4):373–408, 1999.

[15] B. Jeudy. *Extraction sous contraintes de représentations condensées : application à l'optimisation de requêtes inductives sur des données transactionnelles*. PhD thesis, Institut National des Sciences Appliquées de Lyon, Bâtiment Blaise Pascal, F-69621 Villeurbanne cedex, 2002. In preparation.

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

[17] H. Mannila and H. Toivonen. Multiple uses of frequent sets and condensed representations. In *Proceedings of the Second International Conference on Knowledge Discovery and Data Mining KDD'96*, pages 189–194, Portland, USA, Aug. 1996. AAAI Press.

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

[19] R. Meo, G. Psaila, and S. Ceri. A new SQL-like operator for mining association rules. In *Proceedings of the 22nd International Conference on Very Large Data Bases VLDB'96*, pages 122–133, Mumbay, India, Sept. 1996. Morgan Kaufmann.

[20] R. Ng, L. V. Lakshmanan, J. Han, and A. Pang. Exploratory mining and pruning optimizations of constrained associations rules. In *Proceedings of ACM SIGMOD Conference on Management of Data SIGMOD'98*, pages 13–24, Seattle, Washington, USA, 1998. ACM Press.

[21] N. Pasquier, Y. Bastide, R. Taouil, and L. Lakhal. Closed set based discovery of small covers for association rules. In *Proceedings og the 15th Journées Bases de Données Avancées BDA'99*, pages 361–381, Bordeaux, F, Oct. 1999.

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

[23] J. Pei, J. Han, and L. V. S. Lakshmanan. Mining frequent itemsets with convertible constraints. In *Proceedings of the 17th International Conference on Data Engineering ICDE'01*, Heidelberg, Germany, Apr. 2001. IEEE Computer Society Press.

[24] J. Pei, J. Han, and R. Mao. CLOSET an efficient algorithm for mining frequent closed itemsets. In *Proceedings of the ACM SIGMOD Workshop on Research Issues in Data Mining and Knowledge Discovery DMKD'00*, Dallas, USA, May 2000.

[25] 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.

[26] R. Srikant, Q. Vu, and R. Agrawal. Mining association rules with item constraints. In *Proceedings of the Third International Conference on Knowledge Discovery and Data Mining KDD'97*, pages 67–73, Newport Beach, CA, USA, 1997. AAAI Press.

[27] M. J. Zaki. Generating non-redundant association rules. In *Proceedings of the 6th ACM SIGKDD International Conference on Knowledge Dis-*

covery and Data Mining SIGKDD'00, pages 34–43, Boston, USA, Aug. 2000. AAAI Press.

[28] M. J. Zaki. Sequence mining in categorical domains: incorporating constraints. In *Proceedings of the 9th International Conference on Information and Knowledge Management CIKM'01*, pages 422–429, Washington DC, USA, Nov. 2001. ACM.

Appendix

Pushing Anti-Monotone Constraints into APRIORI

We prove that pushing anti-monotone constraints decrease the number of frequency counting.

Assume that $\mathcal{C}$ is the conjunction of two anti-monotone constraints $\mathcal{C}_{am}$ and $\mathcal{C}'_{am}$. The optimization problem is to know whether it is more efficient (strategy `push`) to use the APRIORI algorithm with the constraint $\mathcal{C} = \mathcal{C}_{am} \wedge \mathcal{C}'_{am}$ (to evaluate $\mathcal{C}(S)$, $\mathcal{C}_{am}(S)$ is evaluated and then $\mathcal{C}'_{am}(S)$ *if* $\mathcal{C}_{am}(S)$ *is true*), or (strategy `g&t`) to use a "generate and test" strategy, i.e., to first generate $\mathrm{SAT}_{\mathcal{C}_{am}}$ with APRIORI and then to test each $S \in \mathrm{SAT}_{\mathcal{C}_{am}}$ against the constraint $\mathcal{C}'_{am}$ (the two algorithms are in Figure 4). Given an algorithm $\mathcal{A}$, we denote $\mathtt{Test}_{\mathcal{A}}(\mathcal{C})$ the set of the itemsets that are tested against the constraint $\mathcal{C}$ by $\mathcal{A}$. For instance, it is well known that, in the case of the APRIORI algorithm, $\mathtt{Test}_{apriori}(\mathcal{C}_{\gamma-\mathrm{freq}}) = \mathrm{SAT}_{\mathcal{C}_{\gamma-\mathrm{freq}}} \cup \mathcal{B}d^{-}_{\mathcal{C}_{\gamma-\mathrm{freq}}}$ [18].

Under these hypotheses, we have the following theorem.

Theorem 1 *If* $\mathtt{Test}_{\mathtt{push}}$ *corresponds to strategy* `push` *and if* $\mathtt{Test}_{\mathtt{g\&t}}$ *corresponds to strategy* `g&t`, $\mathtt{Test}_{\mathtt{push}}(\mathcal{C}_{am}) \subseteq \mathtt{Test}_{\mathtt{g\&t}}(\mathcal{C}_{am})$ *and* $\mathtt{Test}_{\mathtt{push}}(\mathcal{C}'_{am}) \subseteq \mathtt{Test}_{\mathtt{g\&t}}(\mathcal{C}'_{am})$.

Proof: *Strategy* `g&t`*: it is the* APRIORI *algorithm with* $\mathcal{C}_{am}(S)$ *therefore* $\mathtt{Test}_{\mathtt{g\&t}}(\mathcal{C}_{am}) = \mathrm{SAT}_{\mathcal{C}_{am}} \cup \mathcal{B}d^{-}_{\mathcal{C}_{am}}$ *and* $\mathcal{C}'_{am}$ *is tested on the result of this algorithm,* $\mathtt{Test}_{\mathtt{g\&t}}(\mathcal{C}'_{am}) = \mathrm{SAT}_{\mathcal{C}_{am}}$.
Strategy `push`*: due to the order in which* $\mathcal{C}_{am}$ *and* $\mathcal{C}'_{am}$ *are tested,* $\mathtt{Test}_{\mathtt{push}}(\mathcal{C}_{am}) = \mathrm{SAT}_{\mathcal{C}_{am} \wedge \mathcal{C}'_{am}} \cup \mathcal{B}d^{-}_{\mathcal{C}_{am} \wedge \mathcal{C}'_{am}}$ *and* $\mathtt{Test}_{\mathtt{push}}(\mathcal{C}'_{am}) = \mathrm{SAT}_{\mathcal{C}_{am} \wedge \mathcal{C}'_{am}} \cup (\mathcal{B}d^{-}_{\mathcal{C}_{am} \wedge \mathcal{C}'_{am}} \cap \mathrm{SAT}_{\mathcal{C}_{am}})$. *The second statement of the theorem follows. For the first one, we must prove that* $\mathrm{SAT}_{\mathcal{C}_{am} \wedge \mathcal{C}'_{am}} \cup \mathcal{B}d^{-}_{\mathcal{C}_{am} \wedge \mathcal{C}'_{am}} \subseteq \mathrm{SAT}_{\mathcal{C}_{am}} \cup \mathcal{B}d^{-}_{\mathcal{C}_{am}}$*: It*

push strategy (push)

1. C_1^g := $\texttt{Items}_1$; $\mathcal{L}_0 = \{\emptyset\}$
2. $k := 1$
3. **while** $C_k \neq \emptyset$ **do**
4. $\quad C_k := \texttt{safe-pruning-on}(C_k^g, \mathcal{L}_{k-1})$
5. $\quad C_k^1 := \mathrm{SAT}_{\mathcal{C}_{am}}(C_k)$
6. $\quad \mathcal{L}_k := \mathrm{SAT}_{\mathcal{C}'_{am}}(C_k^1)$
7. $\quad C_{k+1}^g := \texttt{generate}_{apriori}(\mathcal{L}_k)$
8. $\quad k := k + 1$

od

9. **output** $\bigcup_{i=0}^{k-1} \mathcal{L}_i$

generate and test strategy (g&t)

1. C_1^g := $\texttt{Items}_1$; $\mathcal{L}_0 = \{\emptyset\}$
2. $k := 1$
3. **while** $C_k \neq \emptyset$ **do**
4. $\quad C_k := \texttt{safe-pruning-on}(C_k^g, \mathcal{L}_{k-1})$
5. $\quad \mathcal{L}_k := \mathrm{SAT}_{\mathcal{C}_{am}}(C_k^1)$
6. $\quad C_{k+1}^g := \texttt{generate}_{apriori}(\mathcal{L}_k)$
7. $\quad k := k + 1$

od

8. **output** $\mathrm{SAT}_{\mathcal{C}'_{am}}\left(\bigcup_{i=0}^{k-1} \mathcal{L}_i\right)$

Figure 4: The two algorithms push and g&t

is clear that $\mathrm{SAT}_{\mathcal{C}_{am} \wedge \mathcal{C}'_{am}} \subseteq \mathrm{SAT}_{\mathcal{C}_{am}}$. *Let* $T \in \mathcal{B}d^-_{\mathcal{C}_{am} \wedge \mathcal{C}'_{am}}$. $\forall S \subset T$, $\mathcal{C}_{am}(S) \wedge \mathcal{C}'_{am}(S)$ *is true, so* $\mathcal{C}_{am}(S)$ *is true. Therefore if* $\mathcal{C}_{am}(T)$ *is false,* $T \in \mathcal{B}d^-_{\mathcal{C}_{am}}$. *If* $\mathcal{C}_{am}(T)$ *is true,* $T \in \mathrm{SAT}_{\mathcal{C}_{am}}$. *In either case,* $T \in \mathrm{SAT}_{\mathcal{C}_{am}} \cup \mathcal{B}d^-_{\mathcal{C}_{am}}$ *and the theorem is true.*

Correctness and Completeness of the Generic Algorithm

Theorem 2 *Assuming that* $\mathcal{C}_{am}$ *and* $\mathcal{C}_m$ *are respectively anti-monotone and monotone constraints and that* $\mathcal{C}_m$ *is not a trivial constraint, i.e., that* $\mathcal{C}_m$ *is not always true, the generic algorithm of Section 3.3 is correct and complete, i.e., it outputs exactly* $\mathrm{SAT}_{\mathcal{C}_{am} \wedge Cm}$.

Proof: *We prove by induction on* k *that* $\mathcal{L}_k = \mathrm{SAT}_{\mathcal{C}} \cap \texttt{Items}_k$.
$k = 0$*: Since* $\mathcal{C}_m$ *is not a trivial constraint,* $\emptyset \notin \mathrm{SAT}_{\mathcal{C}}$. *Step 1 ensures that* $\mathcal{L}_0 = \emptyset$.
$k = 1$*: from Step 1 and the definition of* $\texttt{generate}_m$, *we can see that* $C_1^g = \mathcal{B}d^-_{\mathcal{C}'_{am}} \cap \texttt{Items}_1$. *Since* $\mathcal{C}_m(\emptyset)$ *is false,* $\mathcal{B}d^-_{\mathcal{C}'_{am}} \cap \texttt{Items}_1 = \mathrm{SAT}_{\mathcal{C}_m} \cap \texttt{Items}_1$. *Therefore,* C_1^g *is a superset of* $\mathrm{SAT}_{\mathcal{C}} \cap \texttt{Items}_1$ *and a subset of* $\mathrm{SAT}_{\mathcal{C}_m}$. *None of the items of* C_1^g *is pruned during Step 4 (because* $\mathcal{C}_m(\emptyset)$ *is false) therefore* $\mathrm{SAT}_{\mathcal{C}} \cap \texttt{Items}_1 \subseteq C_1 \subseteq \mathrm{SAT}_{\mathcal{C}_m}$. *Steps 5 gives the other inclusion.*
k+1, *completeness: let* $S \in \mathrm{SAT}_{\mathcal{C}} \cap \texttt{Items}_{k+1}$. *First, let us prove that* S *is generated, i.e.,* S *belongs to the set returned by the call* $\texttt{generate}_m(\mathcal{L}_k, k)$. *Since* $S \in \mathrm{SAT}_{\mathcal{C}}$, $\mathcal{C}'_{am}(S)$ *is false, then there are two cases (by definition of* $\mathcal{B}d^-_{\mathcal{C}'_{am}}$*):*

- $S \in \mathcal{B}d^-_{\mathcal{C}'_{am}}$ *and therefore* S *is generated;*
- $S \notin \mathcal{B}d^-_{\mathcal{C}'_{am}}$. *Then* $\exists S' \subset S \cap \texttt{Items}_k$ *such that* $\mathcal{C}'_{am}(S')$ *is false. Since* $\mathcal{C}_{am}(S)$ *is true,* $\mathcal{C}_{am}(S')$ *is true. Then, by induction,* $S' \in \mathcal{L}_k$. *Therefore,* $S \in \texttt{generate}_1(\mathcal{L}_k)$. *If* $k > ms$, $\forall S'$ *such that* $S' \subset S$ *and* $|S'| = k$, $\mathcal{C}'_{am}(S')$ *is false. Since* $\mathcal{C}_{am}(S)$ *is true,* $\forall S'$ *such that* $S' \subset S$ *and* $|S'| = k$, $\mathcal{C}(S')$ *is true. Then, by induction,* $\forall S'$ *such that* $S' \subset S$ *and* $|S'| = k$, $S' \in \mathcal{L}_k$. *Therefore,* $S \in \texttt{generate}_{apriori}(\mathcal{L}_k)$. *In all cases,* S *is generated.*

Then, we must prove that S *is not pruned during Step 4. If* S *is pruned,* $\exists S' \subset S$, $|S'| = |S| - 1$ *such that* $\mathcal{C}(S')$ *is false (by induction) and* $\mathcal{C}_m(S')$ *is true.* $\neg\mathcal{C}(S') \wedge \mathcal{C}_m(S') \Rightarrow \neg\mathcal{C}_{am}(S') \Rightarrow \neg\mathcal{C}_{am}(S) \Rightarrow \neg\mathcal{C}(S)$. *Since* $\mathcal{C}(S)$ *is true,* S *is not pruned.* S *is not rejected in Step 5, therefore* $S \in \mathcal{L}_{k+1}$ *and* $\mathrm{SAT}_\mathcal{C} \cap \texttt{Items}_{k+1} \subset \mathcal{L}_{k+1}$.
Correctness: Let $S \in \mathcal{L}_{k+1}$. S *is generated as a superset of an itemset of* $\mathcal{L}_k = \mathrm{SAT}_\mathcal{C} \cap \texttt{Items}_k$. *Therefore* $\mathcal{C}_m(S)$ *is true and, since* $\mathcal{C}_{am}(S)$ *is true (because of Step 5),* $\mathcal{C}(S)$ *is true. Therefore,* $\mathcal{L}_{k+1} \subset \mathrm{SAT}_\mathcal{C} \cap \texttt{Items}_{k+1}$.

Condensed Representations under Constraints

In this section, we prove that the $\mathcal{C}_{\text{free}}$ and $\mathcal{C}_{free \wedge \mathcal{C}_m}$ constraints are anti-monotone.

Definition 8 (closures and closed itemsets) *The closure of an itemset* S *(denoted by* $\texttt{closure}(S)$*) is the maximal (for set inclusion) superset of* S *which has the same support as* S. *A closed itemset is an itemset that is equal to its closure.*

The closure operator has some interesting properties.

Proposition 2 *The closure operator has the following properties:*

- *(i)* $S \subseteq \texttt{closure}(S)$;
- *(ii) it is monotone increasing:* $S \subseteq S' \Rightarrow \texttt{closure}(S) \subseteq \texttt{closure}(S')$;
- *(iii) it is idempotent:* $\texttt{closure}(S) = \texttt{closure}(\texttt{closure}(S))$.

Let us give a more formal definition for the $\mathcal{C}_{\text{free}}$ constraint and $\mathcal{C}_{free \wedge \mathcal{C}_m}$ constraint.

Proposition 3 ($\mathcal{C}_{\text{free}}$ and $\mathcal{C}_{free\wedge\mathcal{C}_m}$ constraint)

$$\mathcal{C}_{\text{free}}(S) \quad \Leftrightarrow \quad (S' \subset S \Rightarrow S \not\subseteq \texttt{closure}(S')$$

$$\Leftrightarrow \quad (S' \subset S \wedge |S'| = |S| - 1) \Rightarrow S \not\subseteq \texttt{closure}(S')$$

and

$$\mathcal{C}_{free\wedge\mathcal{C}_m}(S) \quad \Leftrightarrow \quad (S' \subset S \wedge \mathcal{C}_m(S')) \Rightarrow S \not\subseteq \texttt{closure}(S')$$

$$\Leftrightarrow \quad (S' \subset S \wedge |S'| = |S| - 1 \wedge \mathcal{C}_m(S')) \Rightarrow S \not\subseteq \texttt{closure}(S').$$

Proposition 4 *The constraints $\mathcal{C}_{\text{free}}$ and $\mathcal{C}_{free\wedge\mathcal{C}_m}$ are anti-monotone.*

> **Proof:** *Anti-monotonicity: Let S such that $\mathcal{C}_{free\wedge\mathcal{C}_m}(S) = true$. Let $T \subset S$ and $\Delta = S \backslash T$. Let $T' \subset T$. Since $\mathcal{C}_{free\wedge\mathcal{C}_m}(S) = true$, $S \not\subseteq \texttt{closure}(T' \cup \Delta)$, then $T \cup \Delta \not\subseteq \texttt{closure}(T' \cup \Delta)$. $\Delta \subseteq \texttt{closure}(T' \cup \Delta)$ (by prop.2.(i) and (ii)). Therefore $T \not\subseteq \texttt{closure}(T' \cup \Delta)$ and finally $T \not\subseteq \texttt{closure}(T')$ by prop.2.(ii). Therefore $\mathcal{C}_{free\wedge\mathcal{C}_m}(T) = true$. Re: only the prop.2.(i) and (ii) of the closure operator are used in this proof.*

The next theorem states that the set $\text{SAT}_{\mathcal{C}_{am}\wedge\mathcal{C}_m}$ can be efficiently computed using the same method as in CLOSE using $\text{SAT}_{\mathcal{C}_{free\wedge\mathcal{C}_m}\wedge\mathcal{C}_{am}\wedge\mathcal{C}_m}$, i.e., the regeneration problem is solved.

Theorem 3 *Let $S \in \text{SAT}_{\mathcal{C}_{am}\wedge\mathcal{C}_m}$ then $\exists S' \in \text{SAT}_{\mathcal{C}_{free\wedge\mathcal{C}_m}\wedge\mathcal{C}_{am}\wedge\mathcal{C}_m}$ such that $S' \subseteq S \subseteq \texttt{closure}(S')$.*

> **Proof:** *By induction on $|S|$. $|S| = 0$: $\mathcal{C}_{free\wedge\mathcal{C}_m}(\emptyset) = true$ therefore $S \in \text{SAT}_{\mathcal{C}_{free\wedge\mathcal{C}_m}\wedge\mathcal{C}_{am}\wedge\mathcal{C}_m}$.*
> *$|S| = k + 1$: if $\mathcal{C}_{free\wedge\mathcal{C}_m}(S) = true$, ok. Else, $\exists S' \subset S$ such that $S \subseteq \texttt{closure}(S')$ and $\mathcal{C}_m(S')$ is true. There are two cases:*
>
> - *$\mathcal{C}_{free\wedge\mathcal{C}_m}(S')$ is true. In this case, $S' \in \text{SAT}_{\mathcal{C}_{free\wedge\mathcal{C}_m}\wedge\mathcal{C}_{am}\wedge\mathcal{C}_m}$ and $S' \subset S \subseteq \texttt{closure}(S')$.*
> - *Else, by induction, $\exists S' \in \text{SAT}_{\mathcal{C}_{free\wedge\mathcal{C}_m}\wedge\mathcal{C}_{am}\wedge\mathcal{C}_m}$ such that $S'' \subseteq S' \subseteq \texttt{closure}(S'')$. Therefore $\texttt{closure}(S') = \texttt{closure}^2(S'')$ and $S'' \subseteq S' \subseteq S \subseteq \texttt{closure}(S') = \texttt{closure}^2(S'')$. By prop. 2.(iii), $\texttt{closure}^2(S'') = \texttt{closure}(S'')$. This finish the proof.*

By replacing the closure operator in the definition of $\mathcal{C}_{\text{free}}$ and $\mathcal{C}_{free\wedge\mathcal{C}_m}$ by $\texttt{closure}_\delta$ we define $\mathcal{C}_{\delta-Free}, \mathcal{C}_{\delta-Free\wedge\mathcal{C}_m}$. These constraints are anti-monotone and we get the following theorem.

Theorem 4 *Let* $S \in \mathrm{SAT}_{\mathcal{C}_{am} \wedge \mathcal{C}_m}$ *then* $\exists S' \in \mathrm{SAT}_{\mathcal{C}_{\delta-Free} \wedge \mathcal{C}_m \wedge \mathcal{C}_{am} \wedge \mathcal{C}_m}$ *and an integer* n *such that* $S' \subseteq S \subseteq \mathtt{closure}_\delta^n(S')$ *and* $n \leq |S| - |S'|$.

Proof: *By induction on* $|S|$. $|S| = 0$*:* $\mathcal{C}_{\delta-Free \wedge \mathcal{C}_m}(\emptyset) = true$ *therefore* $S \in \mathrm{SAT}_{\mathcal{C}_{free} \wedge \mathcal{C}_m \wedge \mathcal{C}_{am} \wedge \mathcal{C}_m}$.
$|S| = k + 1$*: if* $\mathcal{C}_{\delta-Free \wedge \mathcal{C}_m}(S) = true$*, ok. Else,* $\exists S' \subset S$ *such that* $S \subseteq \mathtt{closure}_\delta(S')$ *and* $\mathcal{C}_m(S')$ *is true. There are two cases:*

- $\mathcal{C}_{\delta-Free \wedge \mathcal{C}_m}(S')$ *is true. In this case,* $S' \in \mathrm{SAT}_{\mathcal{C}_{\delta-Free} \wedge \mathcal{C}_m \wedge \mathcal{C}_{am} \wedge \mathcal{C}_m}$ *and* $S' \subset S \subseteq \mathtt{closure}_\delta(S')$.
- *Else, by induction,* $\exists S'' \in \mathrm{SAT}_{\mathcal{C}_{\delta-Free} \wedge \mathcal{C}_m \wedge \mathcal{C}_{am} \wedge \mathcal{C}_m}$ *and* $n \leq |S'| - |S''|$ *such that* $S'' \subset S' \subseteq \mathtt{closure}_\delta^n(S'')$*. Therefore* $\mathtt{closure}_\delta(S') = \mathtt{closure}_\delta^{n+1}(S'')$ *and* $S'' \subseteq S' \subseteq S \subseteq \mathtt{closure}_\delta(S') = \mathtt{closure}_\delta^{n+1}(S'')$. *This finish the proof.*