Deducing Bounds on the Support of Itemsets

Toon Calders*

University of Antwerp
Universiteitsplein 1, B-2610 Wilrijk, Belgium
toon.calders@ua.ac.be

Abstract. Mining *Frequent Itemsets* is the core operation of many data mining algorithms. This operation however, is very data intensive and sometimes produces a prohibitively large output. In this paper we give a complete set of rules for deducing tight bounds on the support of an itemset if the supports of all its subsets are known. Based on the derived bounds $[l, u]$ on the support of a candidate itemset I, we can decide not to access the database to count the support of I if l is larger than the support threshold (I will certainly be frequent), or if u is below the threshold (I will certainly fail the frequency test). We can also use the deduction rules to reduce the size of an adequate representation of the collection of frequent sets; all itemsets I with bounds $[l, u]$, where $l = u$, do not need to be stored explicitly. To assess the usability in practice, we implemented the deduction rules and we present experiments on real-life data sets.

1 Introduction

Mining frequent itemsets is a core operation in many data mining problems. Since their introduction [1], many algorithms have been proposed to find frequent itemsets, especially in the context of association rule mining [1, 2, 12].

The *frequent itemset problem* is stated as follows. Assume we have a finite set of items $\mathcal{I}$. A *transaction* is a subset of $\mathcal{I}$, together with a unique identifier. A *transaction database* $\mathcal{D}$ is a finite set of transactions. A subset of $\mathcal{I}$ is called an *itemset*. We say that an *itemset I is s-frequent in a transaction database $\mathcal{D}$* if the number of transactions in $\mathcal{D}$ that contain all items of I is at least s. The number of transactions that contain all items of I is called the *absolute support of I*. The frequent itemset problem is, given a support threshold s and a transaction database $\mathcal{D}$, find all s-frequent itemsets. In the remainder of the paper, we will always assume that we are working over a transaction database $\mathcal{D}$ with items in $\mathcal{I}$.

All algorithms for mining frequent itemsets rely heavily on the following *monotonicity principle* [16] to prune the search space:

> Let $J \subseteq I$ be two itemsets. In every transaction database $\mathcal{D}$, the support of I will be at most as high as the support of J.

* Research Assistant of the Fund for Scientific Research - Flanders (FWO-Vlaanderen).

Thus, based on the support of a set that is below the support threshold, we can *deduce*, using the monotonicity rule, that also the support of its supersets will be below the threshold. This simple rule of *deduction* has successfully been used in practice. Because of the success of this simple rule, much more attention went into efficient counting schemes than into finding additional ways to prune the search space. The standard example of an algorithm exploiting this monotonicity is the well-known Apriori-algorithm [2]. Apriori traverses the itemset-lattice level by level; in the ith loop, itemsets of cardinality i are counted in the database. Because of the monotonicity principle, all itemsets in loop i that have at least one subset that failed the support-test can be *pruned*; we know *a priori* that they will be infrequent. In this way we will never count itemsets that could be pruned using the monotonicity rule.

In this paper we present deduction rules, additional to the monotonicity rule, that calculate lower and upper bounds on the support of a candidate. As such, we continue work initiated in [9]. Based on the supports of all subsets of an itemset I, the deduction rules we present, will compute bounds $[l, u]$ on the support of I. We show that the rules calculate the best possible such bounds; that is, both l and u are possible as supports of I, and thus, the interval cannot be made more tight. Based on these bounds we can limit the number of candidates we need to count. For example, if l is above the support threshold, then we know *without counting its support in the database* that I is frequent. If there is no need to know the support of I exact, we can thus, in this case, omit counting I. If u is below the threshold, then we know for sure that I is not frequent, and we can prune it.

Besides reducing the number of candidate itemsets, we can also use the deduction rules to make *concise representations* [15] of the frequent itemsets. We call an itemset *derivable* if its lower and upper bound are the same. Thus, an itemset is derivable if its support is uniquely determined by the supports of its subsets. Therefore, for the derivable itemsets, it is not necessary to count their supports. There is also no need to store them; we can later always find the missing supports with the deduction rules. Based on this observation, the NDI-representation is defined. We shortly discuss relations with other concise representations in the literature, including *free sets* [5], *closed sets* [18, 4, 19], and *disjunction-free sets* [6].

The organization of the paper is as follows. In Section 2 we give an example showing that the monotonicity rule is not complete for the deduction of supports. This example also gives a sketch of the general approach we follow to derive the deduction rules. In Section 3 we formally define important notions we will use throughout the paper. In Section 4, the deduction rules are given, and it is proven that they are complete. In Section 5 we present a concise representation based on the deduction rules. Section 6 gives the results of experiments with the deduction rules. In Section 7 we discuss related work and Section 8 concludes the paper.

2 Motivating Example

Apriori does not prune perfectly. Consider the following database.

$$\mathcal{D} \;=\; \begin{array}{|c|c|}\hline \texttt{TID} & \texttt{Items} \\ \hline 1 & A, B \\ 2 & A, C \\ 3 & B, C \\ \hline \end{array} \tag{1}$$

Suppose we are running the Apriori-algorithm on this database $\mathcal{D}$ with minimal absolute support equal to 1. Apriori starts with counting the supports of the singleton-itemsets in $C_1 = \{\{A\}, \{B\}, \{C\}\}$. Since they are all frequent, in its second loop, Apriori will consider the candidates in $C_2 = \{\{A, B\}, \{A, C\}, \{B, C\}\}$. Again all candidates are frequent, and thus, Apriori counts $C_3 = \{\{A, B, C\}\}$ in its third loop. However, the following observation shows that from the supports counted so far, we can derive that $\{A, B, C\}$ must be infrequent.

Let for each itemset I, $\mathcal{F}_I(\mathcal{D})$ denote the set of transactions

$$\mathcal{F}_I(\mathcal{D}) \;=_{def}\; \{(tid, I') \in \mathcal{D} \mid I' = I\} \;,$$

and let f_I be the cardinality of $\mathcal{F}_I(\mathcal{D})$. Hence, in the database $\mathcal{D}$ given in (1), $\mathcal{F}_{AB}(\mathcal{D}) = \{(1, AB)\}$, $\mathcal{F}_{AC}(\mathcal{D}) = \{(2, AC)\}$, and $\mathcal{F}_{BC}(\mathcal{D}) = \{(3, BC)\}$. For all other itemsets I, $\mathcal{F}_I(\mathcal{D})$ is empty. Notice that $\{\mathcal{F}_I(\mathcal{D}) \mid I \subseteq \mathcal{I}\}$ forms a partition of $\mathcal{D}$. This partition is illustrated in Fig. 1. The dots in this figure represent the transactions of $\mathcal{D}$. With every item a set is associated. The set associated with item A consists of all transactions that contain A. The partition defined by the sets $\mathcal{F}_I(\mathcal{D})$ is indicated in the figure.

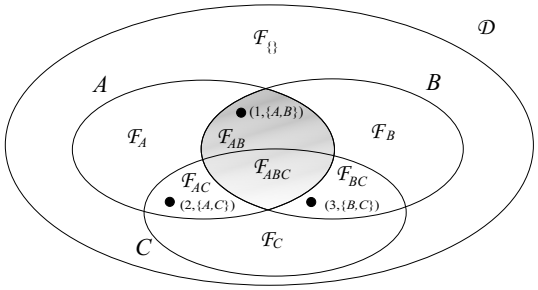


Fig. 1. Partition of $\mathcal{D}$.

The next lemma expresses the supports of the itemsets in function of the numbers f_I, $I \subseteq \mathcal{I}$.

Lemma 1. *For each itemset I,*

$$support(I, \mathcal{D}) \;=\; \sum_{I \subseteq I' \subseteq \mathcal{I}} f_{I'}(\mathcal{D}) \;.$$

Proof.

$$\begin{aligned} support(I,\mathcal{D}) &= |\{(tid,I') \in \mathcal{D} \mid I \subseteq I' \subseteq \mathcal{I}\} \\ &= \sum_{I \subseteq I' \subseteq \mathcal{I}} |\{(tid,I'') \in \mathcal{D} \mid I'' = I'\}| \\ &= \sum_{I \subseteq I' \subseteq \mathcal{I}} |\mathcal{F}_{I'}(\mathcal{D})| \;=\; \sum_{I \subseteq I' \subseteq \mathcal{I}} f_{I'}(\mathcal{D}) \;. \end{aligned}$$

□

In Fig. 1, the grey region indicates the set of all transactions that contain the itemset AB. The transactions that contain AB are exactly those of the form $(tid, \{A, B\})$ and $(tid, \{A, B, C\})$. Hence, the set of transactions in $\mathcal{D}$ containing AB is $\mathcal{F}_{AB}(\mathcal{D}) \cup \mathcal{F}_{ABC}(\mathcal{D})$.

In the running example, after the second loop, we have the following information:

$$\begin{array}{lll} support(\{\},\mathcal{D}) = 3 & support(A,\mathcal{D}) = 2 & support(B,\mathcal{D}) = 2 \\ support(C,\mathcal{D}) = 2 & support(AB,\mathcal{D}) = 1 & support(AC,\mathcal{D}) = 1 \\ support(BC,\mathcal{D}) = 1 & & \end{array} \tag{2}$$

Therefore, the following equalities hold [1]:

$$\begin{cases} f_{\{\}} + f_A + f_B + f_C + f_{AB} + f_{AC} + f_{BC} + f_{ABC} = 3 & (\{\}) \\ f_A + f_{AB} + f_{AC} + f_{ABC} = 2 & (A) \\ f_B + f_{AB} + f_{BC} + f_{ABC} = 2 & (B) \\ f_C + f_{AC} + f_{BC} + f_{ABC} = 2 & (C) \\ f_{AB} + f_{ABC} = 1 & (AB) \\ f_{AC} + f_{ABC} = 1 & (AC) \\ f_{BC} + f_{ABC} = 1 & (BC) \end{cases} \tag{3}$$

For example, the equation $f_A + f_{AB} + f_{AC} + f_{ABC} = 2$ expresses that the support of A equals 2. (3) expresses the same information as (2).

Furthermore, since $f_I = |\mathcal{F}_I|$, it is also true that

$$f_{\{\}}, f_A, f_B, f_C, f_{AC}, f_{BC}, f_{AB}, f_{ABC} \geq 0 \tag{4}$$

We now show how we can derive from (3) and (4) that f_{ABC} must be 0, and hence that the support of ABC is 0. Rewriting (3) gives:

$$\begin{cases} f_{\{\}} = -f_{ABC} & f_{AB} = 1 - f_{ABC} \\ f_A = f_{ABC} & f_{AC} = 1 - f_{ABC} \\ f_B = f_{ABC} & f_{BC} = 1 - f_{ABC} \\ f_C = f_{ABC} & \end{cases} \tag{5}$$

Since both $f_{\{\}}$ and f_A are greater than or equal to 0 (Cfr. (4)), the first two lines of (5) imply respectively $f_{ABC} \leq 0$ and $f_{ABC} \geq 0$. Thus, from the information in (2), it can be derived that $support(ABC,\mathcal{D})$ must be 0, and hence we know *a priori* that ABC cannot be frequent. Nevertheless, Apriori does not prune ABC. This example shows that pruning can be improved beyond monotonicity.

[1] A similar representation is also used in [9, 7, 8].

3 Definitions

In this section we formalize the notions used in the example of last section. We introduce *support expressions* to model information about the supports of the itemsets. The notions of *implication* and *tight implication* express what can be derived from a set of support expressions.

Definition 1. *A* support expression over $\mathcal{I}$ *is an equality*

$$support(I) = s \ ,$$

with I an itemset over $\mathcal{I}$ and s an integer greater than or equal to 0.

A transaction database $\mathcal{D}$ over $\mathcal{I}$ is said to satisfy a support expression $support(I) = s$ *if and only if* $support(I, \mathcal{D}) = s$.

A transaction database is said to satisfy a set of support expressions $\mathcal{S}$ *if and only if it satisfies every expression in $\mathcal{S}$.* □

During the execution of the Apriori-algorithm, the support of ever larger itemsets is counted. In the theory we develop, the knowledge of the supports of the itemsets accumulated in the previous counting steps is modelled as a set of support expressions. In the example in Section 2, the knowledge given in (2) is expressed by the following set of support expressions:

$$\mathcal{S} = \left\{ \begin{array}{lll} support(\{\}) = 3, & & \\ support(A) = 2, & support(B) = 2, & support(C) = 2, \\ support(AB) = 1, & support(AC) = 1, & support(BC) = 1 \end{array} \right\} \ .$$

In the candidate generation and pruning phases, it is decided which sets to count in the next iteration. The decision of which sets will be counted is based solely on the supports of the itemsets counted so far. For example, a set ABC is a candidate in the next loop, only if all three sets AB, AC, and BC were found frequent. If, for example, AB is infrequent, then it can be derived that the support of ABC is below the support threshold as well. Indeed, from support expression $support(AB) = s$ it follows that the support of ABC must be in the interval $[0, s]$. Such deductions are formalized as *logical implication* in the next definition.

Definition 2. *Let I be an itemset over $\mathcal{I}$, and let $l, u \geq 0$ be integers.*

A set of support expressions $\mathcal{S}$ is said to imply bounds $[l, u]$ on the support of I, *denoted $\mathcal{S} \models support(I) \in [l, u]$, if in every transaction database $\mathcal{D}$ that satisfies $\mathcal{S}$, $l \leq support(I, \mathcal{D}) \leq u$ holds.*

The bounds $[l, u]$ are said to be tight, *denoted $\mathcal{S} \models_{\text{tight}} support(I) \in [l, u]$, if there does not exist a smaller interval $[l', u'] \subset [l, u]$ such that $\mathcal{S} \models support(I) \in [l', u']$.* □

Implication denotes what we can derive from a set of support expressions. Given a set of support expressions $\mathcal{S}$, the deduction of $support(I) \in [l, u]$ is correct —or *sound*— if and only if it is true in *every* database that satisfies $\mathcal{S}$.

Tight implication denotes that the bounds cannot be improved; $\mathcal{S} \models_{\text{tight}} \mathit{support}(I) \in [l,u]$ indicates that, given $\mathcal{S}$, both $\mathit{support}(I) = l$ as $\mathit{support}(I) = u$ are possible. Hence, based on $\mathcal{S}$, we cannot improve the interval $[l,u]$. Therefore, a deduction mechanism is *complete* if, given $\mathcal{S}$ and a target set I, it *always* produces the *tight* interval for I.

Example 1.

$$\mathcal{S} = \left\{ \begin{array}{lll} \mathit{support}(\{\}) = 3, & & \\ \mathit{support}(A) = 2, & \mathit{support}(B) = 2, & \mathit{support}(C) = 2, \\ \mathit{support}(AB) = 1, & \mathit{support}(AC) = 1, & \mathit{support}(BC) = 1 \end{array} \right\} .$$

From the monotonicity rule we know that $\mathcal{S} \models \mathit{support}(ABC) \in [0,1]$. The interval $[0,1]$ however, is not tight for the support of ABC. From the reasoning in Section 2, we know that in every database that satisfies $\mathcal{S}$, the support of ABC must be 0. Hence, $\mathcal{S} \models_{\text{tight}} \mathit{support}(ABC) \in [0,0]$. □

The next lemma makes a similar connection between support expressions and systems of linear inequalities as in the example in Section 2.

Lemma 2. *Let $\mathcal{S}$ be a collection of support expressions over $\mathcal{I}$. There exists a transaction database $\mathcal{D}$ over $\mathcal{I}$ that satisfies $\mathcal{S}$, if and only if the following system of inequalities has an integer solution in the variables x_I, $I \subseteq \mathcal{I}$:*

$$\mathit{Sys}(\mathcal{S}) \;=_{def}\; \left\{ \begin{array}{ll} x_I \geq 0 & \forall I \subseteq \mathcal{I} \\ \sum_{I \subseteq I' \subseteq \mathcal{I}} x_{I'} = s_I & \forall (\mathit{support}(I) = s_I) \in \mathcal{S} \end{array} \right.$$

Proof. If: Consider an integer solution of $\mathit{Sys}(\mathcal{S})$. In such a solution all x_I's are integers greater than or equal to 0. Let now $\mathcal{D}$ be the transaction database that for all $I \subseteq \mathcal{I}$ contains x_I transactions of the form (tid, I). Hence, for all $I \subseteq \mathcal{I}$, $f_I(\mathcal{D}) = x_I$. Using Lemma 1, we obtain:

$$\forall I \subseteq \mathcal{I} : \mathit{support}(I, \mathcal{D}) \;=\; \sum_{I \subseteq I' \subseteq \mathcal{I}} f_{I'}(\mathcal{D}) \;=\; \sum_{I \subseteq I' \subseteq \mathcal{I}} x_{I'} \qquad (6)$$

For all support expressions $\mathit{support}(I) = s_I$ in $\mathcal{S}$, $\mathit{Sys}(\mathcal{S})$ contains the equality $\sum_{I \subseteq I' \subseteq \mathcal{I}} x_{I'} = s_I$, and hence, via (6), $\mathit{support}(I, \mathcal{D}) = s_I$. Thus, $\mathcal{D}$ satisfies $\mathcal{S}$.
Only if: Let $\mathcal{D}$ be a transaction database that satisfies $\mathcal{S}$. Then $x_I = f_I(\mathcal{D})$, for all $I \subseteq \mathcal{I}$ is an integer solution of the system $\mathit{Sys}(\mathcal{S})$. Indeed, for all I, $f_I(\mathcal{D})$ is greater than or equal to 0. Furthermore, since $\mathcal{D}$ satisfies $\mathcal{S}$, for all support expressions $\mathit{support}(I) = s_I$ in $\mathcal{S}$, $\mathit{support}(I, \mathcal{D}) = s_I$. Because of Lemma 1, $\mathit{support}(I, \mathcal{D}) = \sum_{I \subseteq I' \subseteq \mathcal{I}} f_{I'}(\mathcal{D})$, and hence $\sum_{I \subseteq I' \subseteq \mathcal{I}} f_{I'}(\mathcal{D}) = s_I$. □

Example 2. There exists a transaction database $\mathcal{D}$ with $\mathit{support}(\{\}, \mathcal{D}) = 3$, $\mathit{support}(A, \mathcal{D}) = 2$, $\mathit{support}(B, \mathcal{D}) = 2$, and $\mathit{support}(AB, \mathcal{D}) = 0$ if and only if the following system of inequalities has a solution:

$$\left\{ \begin{array}{ll} x_{\{\}}, x_A, x_B, x_{AB} \geq 0 & x_B + x_{AB} = 2 \\ x_{\{\}} + x_A + x_B + x_{AB} = 3 & x_{AB} = 0 \\ x_A + x_{AB} = 2 & \end{array} \right.$$

From the last three equalities we derive that $x_A = x_B = 2$, and $x_{AB} = 0$. This however conflicts with $x_{\{\}} + x_A + x_B + x_{AB} = 3$, since all variables must be greater than or equal to 0. Hence, we conclude that there does not exist a transaction database satisfying the given support expressions. □

Problem Statement In the remainder we will concentrate on implication problems for a set I, based on a set $\mathcal{S}$ of support expressions that contains exactly one expression for each strict subset of I. We do not consider cases in which $\mathcal{S}$ contains support expressions for supersets of I, or in which subsets are missing. Hence, given an integer s_J for all $J \subset I$, tight implication of the following type is studied:

$$\{support(J) = s_J \mid J \subset I\} \models_{\text{tight}} support(I) \in [l, u] \ .$$

Notice that the information $\{support(J) = s_J \mid J \subset I\}$ is available for every candidate itemset I in the Apriori-algorithm.

4 Deduction Rules

In this section we describe sound and complete rules for deducing tight bounds on the support of a set I if the supports of all its subsets are given. Because we do not consider itemsets that are not subsets of I, we can assume that all items in the database are elements of I. Since "projecting away" the other items in a transaction database does not change the supports of subsets of I, we can assume without loss of generality that $\mathcal{I} = I$. The correctness of this observation follows from the next lemma.

Definition 3. *Let $I \subseteq \mathcal{I}$ be an itemset.*

- *The* projection of a transaction (tid, I') over $\mathcal{I}$ on I, *denoted $\pi_I(tid, I')$, is the transaction $(tid, I' \cap I)$.*
- *The* projection *of a transaction database $\mathcal{D}$ over $\mathcal{I}$ on I, denoted $\pi_I\mathcal{D}$, is defined as $\pi_I\mathcal{D} =_{def} \{\pi_I T \mid T \in \mathcal{D}\}$.*

Lemma 3. *Let $\mathcal{I}$ be a set of items, and let $J \subseteq I$ be itemsets. For every transaction database $\mathcal{D}$ over $\mathcal{I}$ it holds that*

$$support(J, \mathcal{D}) = support(J, \pi_I\mathcal{D}) \ .$$

Proof. For $J \subseteq I$,

$$\begin{aligned} support(J, \mathcal{D}) &= |\{(tid, I') \in \mathcal{D} \mid J \subseteq I'\}| \\ &= |\{(tid, I') \in \mathcal{D} \mid J \subseteq (I' \cap I)\}| \qquad (J \subseteq I) \\ &= |\{(tid, I'') \in \pi_I\mathcal{D} \mid J \subseteq I''\}| \\ &= support(J, \pi_I\mathcal{D}) \end{aligned}$$

□

This lemma allows for an important reduction of the system $Sys(\mathcal{S})$ associated with a set of support expressions $\mathcal{S}$ that contains an expression for every strict subset of I. Instead of having a variable x_J for every itemset $J \subseteq \mathcal{I}$, with Lemma 3 we can restrict the variables to only those x_J such that $J \subseteq I$.

Corollary 1. *Given an itemset $I \subseteq \mathcal{I}$, and integer $s_J \geq 0$, for every $J \subseteq I$. There exists a transaction database $\mathcal{D}$ satisfying $\forall J \subseteq I : support(J, \mathcal{D}) = s_J$ if and only if the following system of inequalities has a solution:*

$$\begin{cases} x_J \geq 0 & \forall J \subseteq I \\ \sum_{J \subseteq I' \subseteq I} x_{I'} = s_J & \forall J \subseteq I \end{cases}$$

Proof. Because of Lemma 3, the existence of a database $\mathcal{D}$ over $\mathcal{I}$ satisfying the given expressions implies the existence of such a database over I, namely $\pi_I \mathcal{D}$. The corollary now follows from Lemma 2. □

Let $I \subseteq \mathcal{I}$ be an itemset. We assume that all supports of the strict subsets J of I are known, let s_J denote $support(J, \mathcal{D})$. We will now derive optimal bounds on the support of I. These bounds can be determined as follows: the best possible lower bound is the smallest integer l such that the system of support expressions

$$\{support(J) = s_J \mid J \subset I\} \cup \{support(I) = l\}$$

is satisfiable. The best upper bound is the largest integer u such that

$$\{support(J) = s_J \mid J \subset I\} \cup \{support(I) = u\}$$

is satisfiable. Let now s_I be an arbitrary integer. From Corollary 1, we know that the system of support constraints

$$\{support(J) = s_J \mid J \subset I\} \cup \{support(I) = s_I\}$$

is satisfiable if and only if the following system of inequalities has an integer solution:

$$\begin{cases} x_J \geq 0 & \forall J \subseteq I \\ \sum_{J \subseteq I' \subseteq I} x_{I'} = s_J & \forall J \subseteq I \end{cases}$$

This system can be solved for the x_J's as follows:

$$\begin{cases} s_I = x_I \\ s_{I-A} = x_I + x_{I-A} \\ s_{I-B} = x_I + x_{I-B} \\ s_{I-AB} = x_I + x_{I-A} \\ \qquad\quad + x_{I-B} + x_{I-AB} \\ \dots \end{cases} \rightarrow \begin{cases} x_I = s_I \\ s_{I-A} = x_I + x_{I-A} \\ s_{I-B} = x_I + x_{I-B} \\ s_{I-AB} = x_I + x_{I-A} \\ \qquad\quad + x_{I-B} + x_{I-AB} \\ \dots \end{cases}$$

$$\rightarrow \begin{cases} x_I = s_I \\ x_{I-A} = s_{I-A} - s_I \\ x_{I-B} = s_{I-B} - s_I \\ s_{I-AB} = x_I + x_{I-A} \\ \qquad\quad + x_{I-B} + x_{I-AB} \\ \dots \end{cases} \rightarrow \begin{cases} x_I = s_I \\ x_{I-A} = s_{I-A} - s_I \\ x_{I-B} = s_{I-B} - s_I \\ x_{I-AB} = s_{I-AB} - s_{I-A} \\ \qquad\quad - s_{I-B} + s_{I-AB} \\ \dots \end{cases}$$

In general, $x_J = \sum_{J \subseteq J' \subseteq I} (-1)^{|J'-J|} s_{J'}$, as the following lemma shows.

Lemma 4. *Let I be an itemset, and for all $J \subseteq I$, s_J, x_J be integers. The following are equivalent*

(1) $\forall J \subseteq I : s_J = \sum_{J \subseteq I' \subseteq I} x_{I'}$
(2) $\forall J \subseteq I : x_J = \sum_{J \subseteq J' \subseteq I} (-1)^{|J'-J|} s_{J'}$

(The proof of this lemma can be found in Appendix A.)

Therefore, the system of support constraints

$$\{support(J) = s_J \mid J \subset I\} \cup \{support(I) = s_I\}$$

is satisfiable if and only if the following system of inequalities has an integer solution:

$$\begin{cases} x_J \geq 0 & \forall J \subseteq I \\ x_J = \displaystyle\sum_{J \subseteq J' \subseteq I} (-1)^{|J'-J|} s_{J'} & \forall J \subseteq I \end{cases}$$

Hence, if

$$\sum_{J \subseteq J' \subseteq I} (-1)^{|J'-J|} s_{J'} \geq 0 \qquad \forall J \subseteq I$$

or, equivalent,

$$\begin{cases} s_I \leq \displaystyle\sum_{J \subseteq J' \subset I} (-1)^{|I-J'|+1} s_{J'} & \forall J \subseteq I, |I-J| \text{ odd} \\ s_I \geq \displaystyle\sum_{J \subseteq J' \subset I} (-1)^{|I-J'|+1} s_{J'} & \forall J \subseteq I, |I-J| \text{ even} \end{cases}$$

Let $\sigma_I(J, \mathcal{D})$ denote the sum

$$\sigma_I(J, \mathcal{D}) \ =_{def} \ \sum_{J \subseteq J' \subset I} (-1)^{|I-J'|+1} support(J', \mathcal{D})$$

and let $\mathcal{R}_I(J, \mathcal{D})$ denote the rule $support(I) \leq \sigma_I(J, \mathcal{D})$ if $|I-J|$ is odd, and $support(I) \geq \sigma_I(J, \mathcal{D})$ if $|I-J|$ is even. We obtain the following theorem that states that the bounds for itemset I found by the rules $\mathcal{R}_I(J)$, for all $J \subseteq I$, are the best bounds possible; that is, the interval found is tight.

Theorem 1. *Let $\mathcal{D}$ be a transaction database, and let I be an itemset. s_J denotes $support(J, \mathcal{D})$.*

$$\{support(J) = s_J \mid J \subset I\} \quad \models_{\text{tight}} \quad support(I) \in [l, u]$$

with

$$\begin{aligned} l &= \max\{\sigma_I(J, \mathcal{D}) \mid J \subset I, J \text{ even}\} \\ u &= \min\{\sigma_I(J, \mathcal{D}) \mid J \subset I, J \text{ odd}\} \end{aligned}$$

Hence, the rules $\mathcal{R}_I(J), J \subseteq I$ are sound and complete for implication of the support of I, based on the supports of the strict subsets of I.
(The proof of this theorem can be found in Appendix A.)

$$
\begin{cases}
support(ABCD) \geq s_{ABC} + s_{ABD} + s_{ACD} + s_{BCD} \\
\qquad\qquad\quad -s_{AB} - s_{AC} - s_{AD} - s_{BC} - s_{BD} - s_{CD} \\
\qquad\qquad\quad +s_A + s_B + s_C + s_D - 1 \\
support(ABCD) \leq s_A - s_{AB} - s_{AC} - s_{AD} + s_{ABC} + s_{ABD} + s_{ACD} \\
support(ABCD) \leq s_B - s_{AB} - s_{BC} - s_{BD} + s_{ABC} + s_{ABD} + s_{BCD} \\
support(ABCD) \leq s_C - s_{AC} - s_{BC} - s_{CD} + s_{ABC} + s_{ACD} + s_{BCD} \\
support(ABCD) \leq s_D - s_{AD} - s_{BD} - s_{CD} + s_{ABD} + s_{ACD} + s_{BCD} \\
support(ABCD) \geq s_{ABC} + s_{ABD} - s_{AB} \\
support(ABCD) \geq s_{ABC} + s_{ACD} - s_{AC} \\
support(ABCD) \geq s_{ABD} + s_{ACD} - s_{AD} \\
support(ABCD) \geq s_{ABC} + s_{BCD} - s_{BC} \\
support(ABCD) \geq s_{ABD} + s_{BCD} - s_{BD} \\
support(ABCD) \geq s_{ACD} + s_{BCD} - s_{CD} \\
support(ABCD) \leq s_{ABC} \\
support(ABCD) \leq s_{ABD} \\
support(ABCD) \leq s_{ACD} \\
support(ABCD) \leq s_{BCD} \\
support(ABCD) \geq 0
\end{cases}
$$

Fig. 2. Tight bounds on $support(ABCD)$

The rules $\mathcal{R}_{ABCD}(J)$ have been given in Figure 2.

Example 3. Consider the following transaction database.

$\mathcal{D} =$

TID	items
1	A, B
2	A, C, D
3	A, B, D
4	C, D
5	B, C, D
6	A, D
7	B, D
8	B, C, D
9	B, C, D
10	A, B, C, D

$$
\begin{array}{llllll}
s_{\{\}} = 10, & s_A = 5, & s_B = 7, \\
s_C = 6, & s_D = 9, & s_{AB} = 3, \\
s_{AC} = 2, & s_{AD} = 4, & s_{BC} = 4, \\
s_{BD} = 6, & s_{CD} = 6, & s_{ABC} = 1, \\
s_{ABD} = 2, & s_{ACD} = 2, & s_{BCD} = 4.
\end{array}
$$

Figure 2 gives the rules to determine tight bounds on the support of $ABCD$. Based on these deduction rules we derive the following bounds on the support of $ABCD$ *without counting in the database* $\mathcal{D}$.

$support(ABCD, \mathcal{D}) \geq 1$ (Rule $support(ABCD) \geq s_{ABC} + s_{ACD} - s_{AC}$)
$support(ABCD, \mathcal{D}) \leq 1$ (Rule $support(ABCD) \leq s_{ABC}$)

Therefore, we can conclude, without having to count, that the support of $ABCD$ in $\mathcal{D}$ must be 1. In the experiments we will see that this exactness is not very unusual; even in real-life data, and for small itemsets, we will be able to derive very narrow intervals. □

5 Concise Representation

5.1 Derivable Itemsets

Definition 4. *Let I be an itemset, and $\mathcal{D}$ a transaction database. Let for all itemsets $J \subseteq I$, s_J denote $support(J, \mathcal{D})$. We say that I is a derivable itemset w.r.t. $\mathcal{D}$, if*

$$\{support(J) = s_J \mid J \subset I\} \models_{\text{tight}} support(I) \in [s_I, s_I]$$

□

Notice that I derivable means that we do not have to count I in the database to know its support. Based on the supports of the subsets of I we can derive the support of I exactly.

5.2 NDI-representation

Based on the notion of derivable itemsets we propose a *concise representation*. A concise representation [15] is a subset of the set of frequent itemsets, extended with supports, that allows for deriving the whole set of frequent sets and their supports. Such a concise representation is typically much smaller than the whole set of frequent itemsets, even though it contains the same amount of information. Therefore, in situations where the number of frequent itemsets is very large, it is often better to only mine a concise representation. Let $[l_I, u_I]$ be the bounds we can derive for itemset I, based on the supports of its subsets. We now define the NDI-representation as follows:

$$\text{NDI}(\mathcal{D}, s) \ =_{def} \ \{(I, support(I, \mathcal{D})) \mid (l_I \neq u_I), support(I, \mathcal{D}) \geq s\}.$$

That is, NDI only contains those sets that are both frequent in $\mathcal{D}$ and not derivable w.r.t. $\mathcal{D}$.

Theorem 2. *Let $\mathcal{D}$ be a transaction database, and let s be a support threshold.* $\text{NDI}(\mathcal{D}, s)$ *is a concise representation for the s-frequent itemsets in $\mathcal{D}$.*

5.3 Algorithm

In [8], the following theorem has been proven:

Theorem 3 (Anti-monotonicity of derivability). *Let $I \subseteq J$ be itemsets over $\mathcal{I}$, and $\mathcal{D}$ be a transaction database over $\mathcal{I}$. If I is a derivable itemset, then J must be a derivable itemset as well.*

Based on this theorem we come up with the following algorithm to find all frequent non-derivable itemsets.

```
(1)  NDI(𝒟,s)
(2)      i := 1; NDI := {}; C_1 := {{i} | i ∈ ℐ};
(3)      for all I in C_1 do I.l := 0; I.u := |𝒟|;
(4)      while C_i not empty do
(5)          Count the supports of all candidates in C_i in one pass over 𝒟;
(6)          F_i := {I ∈ C_i | support(I, 𝒟) ≥ s};
(7)          NDI := NDI ∪ F_i;
(8)          PreC_{i+1} := AprioriGenerate(F_i);
(9)          C_{i+1} := {};
(10)         for all J ∈ PreC_{i+1} do
(11)             Compute bounds [l, u] on support of J;
(12)             if l ≠ u then J.l := l; J.u := u; C_{i+1} := C_{i+1} ∪ {J};
(13)         i := i + 1
(14)     end while
(15)     return NDI
```

For a more elaborated description of the algorithm we refer the interested reader to [8].

6 Experiments

The experiments were performed on a 1.5GHz Pentium IV PC with 256MB of main memory. To empirically evaluate the proposed NDI-algorithm and deduction rules, we performed several tests on the datasets summarized in the following table. For each dataset the table shows the number of transactions, the number of items, and the average transaction length.

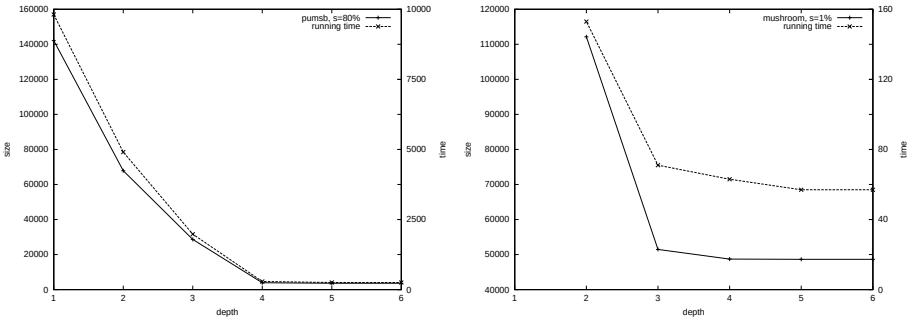
Dataset	# trans.	# items	Avg. length
BMS-POS	515 597	1 656	6.53
T40I10D100K	100 000	1 000	39.6
Connect-4	67 557	125	42
BMS-Webview-1	59 602	497	2.51
Pumsb	49 046	2 112	74
Mushroom	8 124	120	23

These datasets are all well-known benchmarks for frequent itemset mining. The *BMS-Webview* and *BMS-POS* datasets are click-stream data from a small dot-com company that no longer exists. These two datasets were donated to the research community by Blue Martini Software. The *Pumsb*-dataset is based on census data, the *Mushroom* dataset contains characteristics from different species of mushrooms. The *Connect-4* dataset contains different game positions. The Pumsb dataset is available in the UCI KDD Repository [13], and the Mushroom and Connect-4 datasets can be found in the UCI Machine Learning Repository [3]. The *T40I10D100K* dataset was generated using the IBM synthetic data generator.

The NDI-algorithm differs slightly from the algorithm presented in Section 5. In order to avoid the generation of pairs of items in the second loop, the candidates are only generated while iterating over the dataset. In this way the generation of pairs that do not appear in the database is avoided.

6.1 Overhead of Rule Evaluation

We first study the influence of limiting the depth of the rules we evaluate. In Fig. 3, the number of sets that are derivable when we evaluate rules up to depth k, and the time needed to find them are indicated for different k. As can be seen, the number of NDIs drops quickly from depth 1 to depth 2. In the Mushroom experiment, the test with $k = 1$ was even not feasible. From depth 3 on, higher depths result in only a slight decrease of the number of NDIs. This is not that remarkable since the number of NDIs of these sizes is small. The running times in Fig. 3 show that for these limited depths, the cost of evaluating all rules is rather small.



(a) Pumsb (b) Mushroom

Fig. 3. Size of the representations when k is limited.

In Tab. 1, four experiments with the BMS-POS dataset are reported in detail. For each iteration of the algorithm, the number of candidates and the number these candidates that turn out to be frequent NDIs are reported, together with the computation time of respectively candidate generation with rule evaluation and counting. In Tab. 1 (a) and (b), all rules are evaluated, while in Tab. 1 (c) and (d) no rules are evaluated. Hence, the experiments in Tab. 1 (c) and (d) are in fact plain Apriori. The BMS-POS dataset is interesting, since it is the only dataset that contains almost no derivable itemsets. Therefore, these examples show very well the cost of evaluating the rules. The rule evaluation time is included in the generation time for the candidates. Tab. 1 shows that the evaluation times for the rules are very reasonable. This is especially so when the number of transactions becomes very high. For iteration 2, the number of candidates and the generation time of these candidates is not reported, because they are generated on the fly to avoid pairs of items with support 0.

	$\|C_k\|$		$\|NDI\|$		Width
1	1656	6.437s	510	0s	515597
2			9553	35.828s	116102
3	187839	6.625s	39912	80.859s	29167
4	157481	14.329s	74768	120.828s	11831
5	117797	28.437s	77353	126.922s	3462
6	62233	41.922s	47499	93.266s	998
7	18369	38.656s	16276	46.297s	250
8	2230	18s	2141	16.547s	88
9	10	1.906s	9	8.579s	19
	268021	686.9			

(a) BMS-POS, support = 361, all rules

	$\|C_k\|$		$\|NDI\|$		Width
1	1656	6.609s	461	36.047s	515597
2			7554	75.437s	116102
3	126338	4.719s	27904	103.188s	29167
4	95701	8.922s	46115	98.25s	11831
5	63578	15.5s	42047	67.641s	3462
6	29226	20.578s	22300	29.937s	998
7	7075	14.86s	6315	12.031s	250
8	704	5.234s	685	7.985s	88
9	1	0.343s	1	8.579s	9
	153382	508.234s			

(b) BMS-POS, support = 465, all rules

	$\|C_k\|$		$\|NDI\|$		Width
1	1656	5.531s	510	0s	n/a
2			9553	35.719s	n/a
3	187839	5.031s	39912	80.672s	n/a
4	157481	7.375s	74768	126s	n/a
5	117929	8.984s	77361	128.484s	n/a
6	63981	7.406s	47741	95.422s	n/a
7	21335	3.812s	17293	49.203s	n/a
8	3765	1.109s	3283	19.797s	n/a
9	255	0.282s	228	9.625s	n/a
	270649	594.141s			

(c) BMS-POS, support = 361, no rules

	$\|C_k\|$		$\|NDI\|$		Width
1	1656	5.5s	461	0s	n/a
2			7554	34.86s	n/a
3	126338	3.344s	27904	73.547s	n/a
4	95701	4.359s	46115	102.469s	n/a
5	63641	4.656s	42048	95.297s	n/a
6	30114	3.421s	22341	63.532s	n/a
7	7967	1.422s	6480	29.953s	n/a
8	962	0.359s	849	13.079s	n/a
9	30	0.188s	29	8.312s	n/a
	153781	453.093s			

(d) BMS-POS, support = 465, no rules

Table 1. Example runs on the BMS-POS dataset.

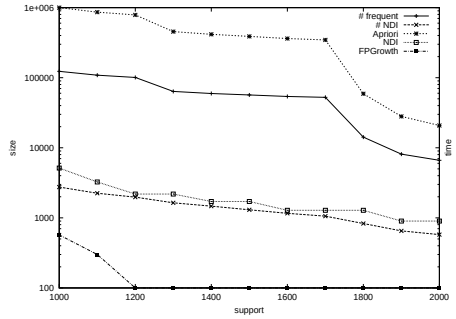
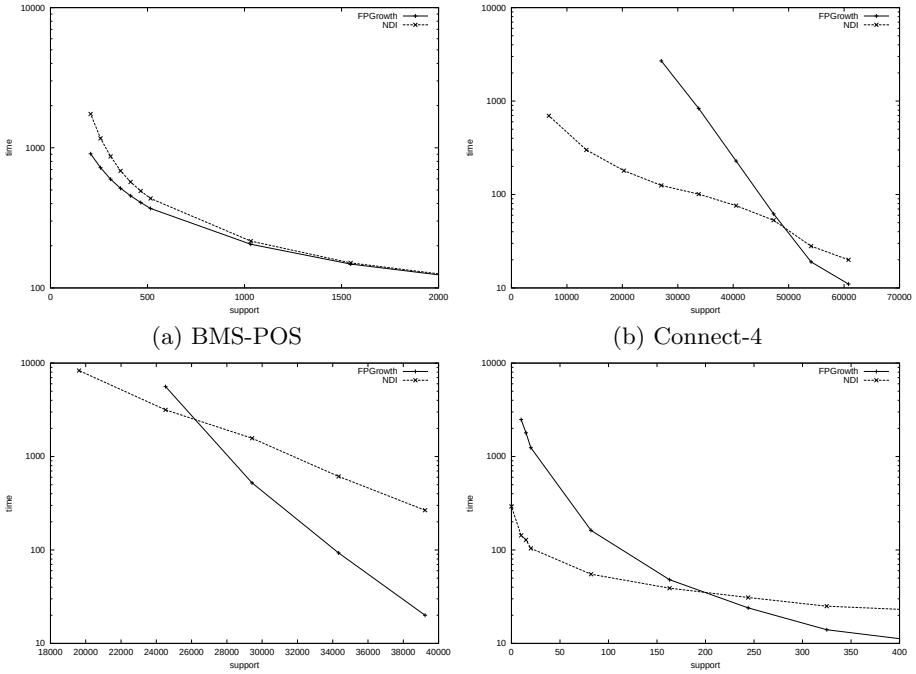


Fig. 4. Running time on the Mushroom dataset.

6.2 Comparison with Mining All Frequent Sets

Since the overhead of calculating the rules is small, the running times of the Apriori-algorithm and the NDI-algorithm are almost linear in the size of their respective output. Therefore, the gain in speed of the NDI-algorithm over the extraction of all frequent itemsets with the Apriori-algorithm is more or less the ratio between the number of frequent sets and the number of frequent NDIs. This claim is supported by Fig. 4. In Fig. 4, the running time of the NDI-algorithm, Apriori, and FPGrowth is given, together with the number of the frequent and the non-derivable sets, for different minimal supports. In this example, the execution time of FPGrowth is much lower than for the other algorithms. As long as the number of frequent sets is not too high, FPGrowth will be more efficient



(a) BMS-POS (b) Connect-4

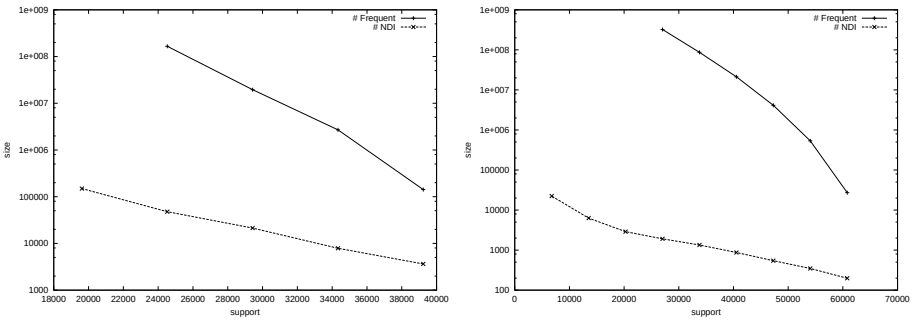
(c) Pumsb (d) Mushroom

Fig. 5. Comparison of running times of NDI and FPGrowth.

than mining the NDIs. As soon as the number of frequent sets becomes very high however, NDI will become more efficient.

Since in most of the experiments we present the number of frequent sets is very high, the execution of the Apriori-algorithm was not always possible. Instead we present a comparison with FPGrowth, and we report for some of the experiments the number of frequent sets and the number of NDIs. The results are presented in Fig. 5. For the Connect-4 and the Pumsb dataset it was not possible to perform the FPGrowth algorithm within reasonable time for the lowest supports.

In Fig. 5, it can clearly be seen that in most datasets once the support threshold becomes too low, and the number of frequent sets explodes, the NDI-algorithm becomes more efficient than mining all frequent itemsets. In Fig. 6, the number of frequent NDIs is compared with the total number of frequent sets. The only exceptional dataset in this perspective was the BMS-POS dataset, in which the performance of the NDI and FPGrowth algorithms stays more or less comparable. The explanation for this is in Tab. 1. In the BMS-POS dataset there are almost no derivable itemsets of low cardinality.



(a) Connect-4 (b) Pumsb

Fig. 6. Comparison of the number of frequent sets and NDIs.

6.3 Comparison with Other Concise Representations

We compared the NDI-representation with the following other concise representations:

- *Free sets representation* FreeRep [5],
- *Disjunction-free sets representation* DFreeRep [6],
- *Disjunction-free generators representation* DFreeGenRep [14], and
- *Closed sets representation* Closed [18].

In Figure 7, the sizes of these representations and $|NDI|$ are reported on different datasets. The experiments show that on these datasets, the NDI-representation is often the smallest representation. Only in the BMS-Webview-1 dataset, the NDI-representation is slightly larger than the Closed sets representation.

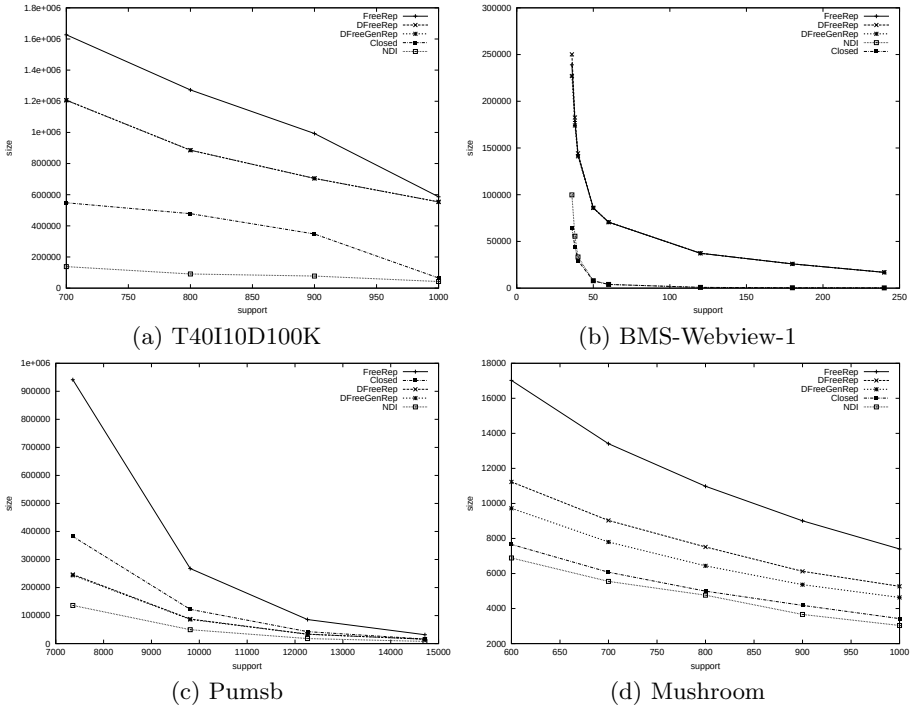
7 Related Work

7.1 Concise Representations

Closed itemsets [18] received a lot of attention in the literature [4, 19, 20]. They can be introduced as follows: the *closure* of an itemset I is the largest superset of I such that its support equals the support of I. This superset is unique and is denoted by $cl(I)$. An itemset is called *closed* if it equals its closure. In [18], the authors show that the set of frequent closed sets is a concise representation for the frequent itemsets.

Free sets [5] *or Generators* [14] (Free sets [5] and generators [18, 14] are the same.) An itemset I is called *free* if it has no subset with the same support. The free-set representation is based on the fact that is $support(A) = support(AB)$, also $support(AC) = support(ABC)$. This deduction can also be made with the following two deduction rules presented in this paper:

$$support(ABC) \leq support(AC) \ , \ and$$
$$support(ABC) \geq support(AB) + support(AC) - support(A) \ .$$



(a) T40I10D100K (b) BMS-Webview-1

(c) Pumsb (d) Mushroom

Fig. 7. Size of different concise representations.

Disjunction-free sets [6] *or disjunction-free generators* [14] are an extension of free sets. A set I is called disjunction-free if there does not exist two items A, B in I such that

$$support(I) = support(I - A) + support(I - B) - support(I - AB) \ .$$

Free sets are a special case of disjunction-free sets, namely when $A = B$. The representation is based on the fact that when

$$support(ABC) = support(AB) + support(AC) - support(A) \ ,$$

it is also true that

$$support(ABCD) = support(ABD) + support(ACD) - support(AD) \ .$$

Again, this deduction follows from the rules presented in this paper:

$$s_{ABCD} \geq s_{ABD} + s_{ACD} - s_{AD} \ , \ and$$
$$s_{ABCD} \leq s_{ABC} + s_{ABD} + s_{ACD} - s_{AB} - s_{AC} - s_{AD} + s_A \ .$$

7.2 Deduction

Another application of deduction rules is developed in [11]. Based on the observation that highly frequent items tend to blow up the output of a data mining

query by an exponential factor, the authors develop a technique to leave out these highly frequent items, and to reintroduce them after the mining phase by using a deduction rule, the *multiplicative* rule. The multiplicative rule can be stated as follows: let I, J be itemsets, then

$$support(I \cup J, \mathcal{D}) \geq support(I, \mathcal{D}) + support(J, \mathcal{D}) - support(\{\}, \mathcal{D}) \ .$$

This rule can be derived from the rules in our framework.

Also in the field of artificial intelligence, much work has been done around inferring knowledge. Interesting related work in artificial intelligence concentrates on logics for reasoning about probabilities, such as the probabilistic logic of *Nilsson* [17] and of *Fagin et al.* [10].

8 Conclusions and Further Work

We presented sound and complete rules for deducing bounds on the support of an itemset. These rules have many possible applications, such as improving the pruning in the Apriori-algorithm, making concise representations, and deducing the result of a data mining query based on previous query results. We evaluated the rules against different real-life data set. The experiments showed the usefulness of the deduction rules for mining concise representations of the frequent itemsets.

For the deduction rules presented in this paper, we need to know the supports of all subsets exactly. Interesting further work includes finding deduction rules for situations in which some of the subsets are missing, and when we only have partial knowledge of the supports.

Acknowledgement

We thank *Bart Goethals* of the *Helsinki Institute for Information Technology* for the implementation of many of the algorithms used in the experiments section and for his help with the experiments. The number of closed sets is produced by the ChARM-algorithm of *Mohammed Zaki* [20].

References

1. R. Agrawal et al. Mining association rules between sets of items in large databases. In *Proc. ACM SIGMOD*, pages 207–216, 1993.
2. R. Agrawal and R. Srikant. Fast algorithms for mining association rules. In *Proc. VLDB*, pages 487–499, 1994.
3. C.L. Blake and C.J. Merz. *UCI Repository of machine learning databases [http://www.ics.uci.edu/∼mlearn/MLRepository.html].* Irvine, CA: University of California, Dept. of Inf. and CS., 1998.
4. J.-F. Boulicaut and A. Bykowski. Frequent closures as a concise representation for binary data mining. In *Proc. PaKDD*, pages 62–73, 2000.

5. J.-F. Boulicautet al. Approximation of frequency queries by means of free-sets. In *Proc. PKDD*, pages 75–85, 2000.
6. A. Bykowski and C. Rigotti. A condensed representation to find frequent patterns. In *Proc. PODS*, 2001.
7. A. Bykowski et al. Model-independent bounding of the supports of boolean formulae in binary data. In *Proc. ECML-PKDD Workshop KDID*, pages 20–31, 2002.
8. T. Calders and B. Goethals. Mining all non-derivable frequent itemsets. In *Proc. PKDD*, pages 74–85. Springer, 2002.
9. T. Calders and J. Paredaens. Axiomatization of frequent sets. In *Proc. ICDT*, pages 204–218, 2001.
10. R. Fagin et al. A logic for reasoning about probabilities. *Information and Computation*, 87(1,2):78–128, 1990.
11. D. Groth and E. Robertson. Discovering frequent itemsets in the presence of highly frequent items. In *Proc. Workshop RBDM, in Conjunction with 14th Intl. Conf. On Applications of Prolog*, 2001.
12. J. Han, J. Pei, and Y. Yin. Mining frequent patterns without candidate generation. In *Proc. ACM SIGMOD*, pages 1–12, 2000.
13. S. Hettich and S. D. Bay. *The UCI KDD Archive. [http://kdd.ics.uci.edu]*. Irvine, CA: University of California, Dept. of Inf. and CS., 1999.
14. M. Kryszkiewicz. Concise representation of frequent patterns based on disjunction-free generators. In *Proc. ICDM*, pages 305–312, 2001.
15. H. Mannila and H. Toivonen. Multiple uses of frequent sets and condensed representations. In *Proc. KDD*, 1996.
16. H. Mannila and H. Toivonen. Levelwise search and borders of theories in knowledge discovery. *DMKD*, 1(3):241–258, 1997.
17. N. Nilsson. Probabilistic logic. *Artificial Intelligence*, 28:71–87, 1986.
18. N. Pasquier et al. Discovering frequent closed itemsets for association rules. In *Proc. ICDT*, pages 398–416, 1999.
19. J. Pei et al. Closet: An efficient algorithm for mining frequent closed itemsets. In *ACM SIGMOD Workshop DMKD*, Dallas, TX, 2000.
20. M.J. Zaki and C. Hsiao. ChARM: An efficient algorithm for closed association rule mining. In *Proc. ICDM*, 2002.

A Proofs

Proof of Lemma 4

Proof. (1)⇒(2): Proof by induction on $|I - J|$ that $x_J = \sum_{J \subset I' \subseteq I} (-1)^{|I-J|} s_{I'}$.
Base case ($J = I$): for $J = I$, (2) reduces to $x_I = s_I$. $x_I = s_I$ is also in (1).
General case (J arbitrary): By induction hypothesis, for all I', such that $J \subset I' \subseteq I$, $x_{I'} = \sum_{I' \subseteq J' \subseteq I} (-1)^{|J'-I'|} s_{J'}$. From (1) we know that $\sum_{J \subseteq I' \subseteq I} x_{I'} = s_J$.
Hence,

$$x_J = s_J - \sum_{J \subset I' \subseteq I} x_{I'} = s_J - \sum_{J \subset I' \subseteq I} \sum_{I' \subseteq J' \subseteq I} (-1)^{|J'-I'|} s_{J'}$$

$$= s_J - \sum_{J \subset J' \subseteq I} \left(\sum_{J \subset I' \subseteq J'} (-1)^{|J'-I'|} \right) s_{J'}$$

$$
\begin{aligned}
&= s_J - \sum_{J \subset J' \subseteq I} \left(\sum_{J \subset I' \subseteq J'} (-1)^{|I'-J|} \right) (-1)^{|J'-J|} s_{J'} \\
&= s_J - \sum_{J \subset J' \subseteq I} \left(\sum_{i=1} |J'-J| \binom{|J'-J|}{i} (-1)^i \right) (-1)^{|J'-J|} s_{J'} \\
&= s_J - \sum_{J \subset J' \subseteq I} \left(-1 + \sum_{i=0}^{|J'-J|} \binom{|J'-J|}{i} (-1)^i \right) (-1)^{|J'-J|} s_{J'} \\
&= s_J + \sum_{J \subset J' \subseteq I} (-1)^{|J'-J|} s_{J'} \;=\; \sum_{J \subseteq J' \subseteq I} (-1)^{|J'-J|} s_{J'}
\end{aligned}
$$

(2)⇒(1): Proof by induction on $I - J$ that $s_J = \sum_{J \subseteq I' \subseteq I} x_{I'}$.
Base case ($J = I$): for $J = I$, (1) reduces to $s_I = x_I$. $x_I = s_I$ is also in (2).
General case (J arbitrary): By induction hypothesis, for all J', such that $J \subset J' \subseteq I$, $s_{J'} = \sum_{J' \subseteq I' \subseteq I} x_{I'}$. Since $x_J = \sum_{J \subseteq J' \subseteq I} (-1)^{|J'-J|} s_{J'}$, also
$s_J = x_J - \sum_{J \subset J' \subseteq I} (-1)^{|J'-J|} s_{J'}$. Hence,

$$
\begin{aligned}
s_J &= x_J - \sum_{J \subset J' \subseteq I} (-1)^{|J'-J|} \sum_{J' \subseteq I' \subseteq I} x_{I'} = x_J - \sum_{J \subset I' \subseteq I} \left(\sum_{J \subset J' \subseteq I'} (-1)^{|J'-J|} \right) x_{I'} \\
&= x_J - \sum_{J \subset I' \subseteq I} (-1)\, x_{I'} \;=\; \sum_{J \subseteq I' \subseteq I} x_{I'}
\end{aligned}
$$

Proof of Theorem 1

Proof. By definition, the integers l, u such that $[l, u]$ is the tight interval implied for $\mathit{support}(I)$ by $\{\mathit{support}(J) = s_J \mid J \subset I\}$, are the minimal and maximal integer s_I such that the system

$$
\mathcal{S} \;=\; \{\mathit{support}(J) = s_J \mid J \subset I\} \cup \{\mathit{support}(I) = s_I\}
$$

is satisfiable. Using Corollary 1 and Lemma 4, we obtain that $\mathcal{S}$ has a solution if and only if

$$
\begin{aligned}
x_J &\geq 0 && \forall J \subseteq I \\
x_J &= \sum_{J \subseteq J' \subseteq I} (-1)^{|J'-J|} s_{J'} && \forall J \subseteq I
\end{aligned}
$$

has a solution. This system has a solution if and only if

$$
\begin{aligned}
s_I &\leq \sum_{J \subseteq J' \subset I} (-1)^{|I-J'|+1} s_{J'} && \forall J \subseteq I, |I-J| \text{ odd} \\
s_I &\geq \sum_{J \subseteq J' \subset I} (-1)^{|I-J'|+1} s_{J'} && \forall J \subseteq I, |I-J| \text{ even}
\end{aligned}
$$

Hence, the maximal solution is the minimum of the upper bounds as given by $\mathcal{R}_I(J)$, J odd, and the minimal solution is the maximum of the lower bounds as given by $\mathcal{R}_I(J)$, J even. □